Exciting New Computers From Atari And Commodore

COMPUTE!

$2.95
April
1985
Issue 59
Vol. 7, No. 4
$3.50 Canada
02193
ISSN 0194-347X

The Leading Magazine Of Home, Educational, And Recreational Computing

TurboDisk:
Load Programs 300 Percent Faster
Free Inside For Commodore 64 & VIC-20

Mindbusters
Fascinating Puzzle Game For Apple, Commodore 64, VIC-20, PC/PCjr, Atari, TI

IBM Graphics Printer Switch Settings For PC & PCjr

SpeedScript 3.0
Enhanced Word Processor For Commodore VIC-20

COMPUTE!'s Guide To Computer Camps

And Much More

DIR 3449
AHITZIZ00N7118 04 C0 JAN86
THERESA HITZ
2007 N 71ST
WAUWATOSA WI 53213

0 71486 02193

April 1985 Vol. 7, No. 4

FEATURES

REVIEWS

COLUMNS AND DEPARTMENTS

THE JOURNAL

NOTE: See page 148 before typing in programs.

AP Apple, **Mac** Macintosh, **AT** Atari, **V** VIC-20, **64** Commodore 64, **+4** Commodore Plus/4, **16** Commodore 16, **P** PET/CBM, **TI** Texas Instruments, **PC** IBM PC, **PCjr** IBM PCjr, **CC** Radio Shack Color Computer. *General interest.

TOLL FREE Subscription Order Line
800-334-0868 (in NC 919-275-9809)

COMPUTE! Publications, Inc.
One of the ABC Publishing Companies:
ABC Publishing, President, Robert G. Burton
1330 Avenue of the Americas, New York, New York 10019
Address all inquiries to:
P.O. Box 5406, Greensboro, NC 27403

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published monthly by COMPUTE! Publications, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919) 275-9809. Editorial Offices are located at 324 West Wendover Avenue, Greensboro, NC 27408. Domestic Subscriptions: 12 issues, $24. Send subscription orders or change of address (P.O. form 3579) to **COMPUTE!** Magazine, P.O. Box 914, Farmingdale, NY 11737. Second class postage paid at Greensboro, NC 27403 and additional mailing offices. Entire contents copyright ©1985 by COMPUTE! Publications, Inc. All rights reserved, ISSN 0194 357X.

be available by this summer, and Atari is thinking about making copies available to current owners at little or no cost.

Lowercase On The TI

Lowercase characters on the TI-99/4A appear as small capital letters. In some of my programs, I'd like to have a normal lowercase character set. I've tried many times to redefine the lowercase letters, but my results have been disappointing. Could you provide me with some character definitions for lowercase letters?

Jim Tope

The following program redefines the lowercase character set with lowercase letters:

```
1ØØ GOSUB 1ØØØ
11Ø CALL CLEAR
12Ø PRINT "abcdefghijklmnopqrstuvwx
    yz"
13Ø FOR I=1 TO 2ØØØ
14Ø NEXT I
15Ø STOP
999 REM LOWERCASE SET
1ØØØ FOR I=97 TO 122
1Ø1Ø READ A$
1Ø2Ø CALL CHAR(I,A$)
1Ø3Ø NEXT I
1Ø4Ø RETURN
1Ø5Ø DATA ØØØØØØ38Ø43C643C,ØØ4Ø4Ø4Ø
     78444478,ØØØØØØ1C2Ø2Ø2Ø1C
1Ø6Ø DATA ØØØ4Ø4Ø43C44443C,ØØØØØØ38
     4478403C,ØØ1824207Ø2Ø2Ø2Ø
1Ø7Ø DATA ØØØØØØ38443CØ438,ØØ4Ø4Ø4Ø
     78444444,ØØ1ØØØ1Ø1Ø1Ø1Ø1Ø
1Ø8Ø DATA ØØØ4ØØØ4Ø4Ø42418,ØØ2Ø2Ø24
     283Ø2824,ØØ1Ø1Ø1Ø1Ø1Ø1Ø1Ø
1Ø9Ø DATA ØØØØØØ6854544444,ØØØØØØ58
     64444444,ØØØØØØ3844444438
11ØØ DATA ØØØØØØ7844784Ø4Ø,ØØØØØØ3C
     443CØ4Ø4,ØØØØØØ586440404Ø
111Ø DATA ØØØØØØ3C4Ø38Ø478,ØØ1Ø1Ø38
     1Ø1Ø1ØØC,ØØØØØØ4444444438
112Ø DATA ØØØØØØ444428281Ø,ØØØØØØ44
     44545428,ØØØØØØ442810282844
113Ø DATA ØØØØØØ44443CØ438,ØØØØØØ3C
     Ø4Ø81Ø3C
```

To use this lowercase character set in your programs, add the subroutine beginning at line 1000 containing the character definitions.

More Commodore Overheating

I have had a Commodore 64 for nine months, and am now experiencing problems. After an hour or so of use the bottom of the computer gets very warm, the computer locks up, and I lose everything not saved. Is there any remedy short of sending it back to Commodore?

Chuck Kutz-Marks

Your problem seems to be related to overheating, but it's impossible to tell from a letter exactly what is causing the problem. It could be caused by any one of a number of faults. Your best choice i ably to return your computer to Commodore, first you may want to try some simple troubleshooting.

Try borrowing a power supply from a fr who has a 64 and see if the problem occurs not, then your power supply has developed mal fault and needs to be replaced. Several dent sources sell power supplies.

You may also want to remove the foil-c cardboard shield found inside most 64s. It's signed to cut down interference between the puter and a TV set, but it also traps heat.

If you or a friend is handy with hardw could locate the components responsible for excessive heat and install a heat sink to dra and dissipate the heat. But don't attempt th you're experienced at this kind of repair.

If you continue to have problems, your is to contact Commodore's Customer Servic Department by calling 215-431-9100 and a return your 64 for service. Although this w several weeks, it's probably your cheapest alternative.

Mixing Atari Graphics Modes

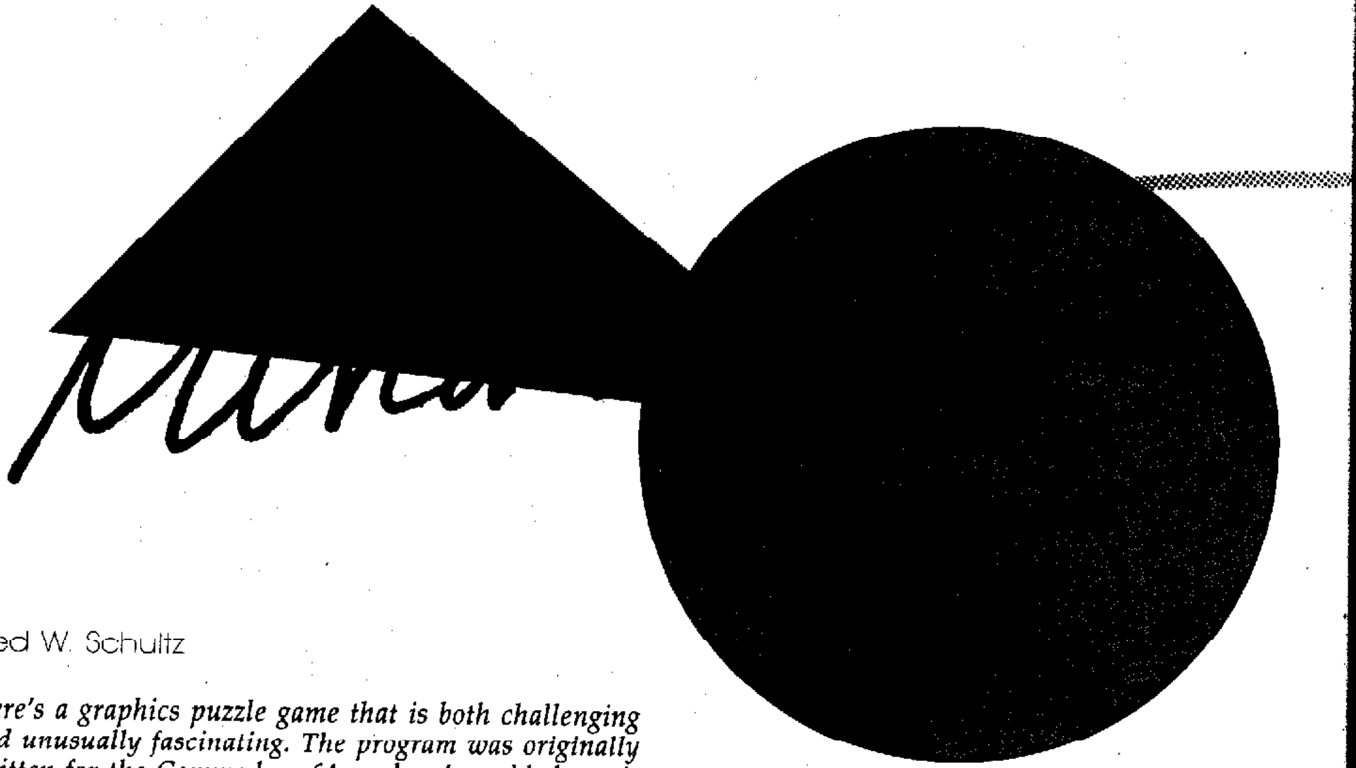
I own an Atari 1200XL computer. I've m few BASIC programs of my own and I'v trying to get two graphics modes on the the same time. For example, having GR 1 at the top and GRAPHICS 2 at the bo Can you help?

James

A full explanation of modifying graphics m beyond the scope of this column, but try th ing program. Set the variable G2 to the nu GRAPHICS 2 lines you'd like, then GOSU Lines 100–200 demonstrate the subroutine. program modifies a GRAPHICS 1 display POKEing in the display list bytes for GRA You must not set G2 to less than 1 or grea 11.

COMPUTE! *has published several articl topic in back issues, some of which are no available. For more information, refer to "Design Custom Graphics Modes" in* COM First Book of Atari Graphics.

```
HO 1ØØ G2=8:GOSUB 5ØØ
PI 11Ø FOR I=1 TO 24:? #6;"LI
       EXT I
FP 12Ø GOTO 12Ø
HG 5ØØ GRAPHICS 17:IF G2<1 OR
       HEN RETURN
GI 51Ø DLIST=PEEK(56Ø)+256*PE
KG 52Ø FOR I=29-G2*2 TO 28-G2
       IST+I,7:NEXT I
EA 53Ø POKE DLIST+I,65:POKE D
       ,PEEK(56Ø):POKE DLIST+
       (561)
HJ 54Ø RETURN
```



Ned W. Schultz

Here's a graphics puzzle game that is both challenging and unusually fascinating. The program was originally written for the Commodore 64, and we've added versions for the unexpanded VIC-20, Atari, Apple II-series computers, IBM PC (color or monochrome), PCjr, and TI-99/4A.

Are you ready to pit your brain against the computer's? "Mindbusters" presents you with three graphics puzzles that are guaranteed to keep your mind's microprocessors and memory chips whirring for hours.

After you type, save, and run your copy of Mindbusters, you can choose to solve one of three puzzles: a mind bender, a mind bruiser, or a mind blower. Warm up with the mind bender—it's the easiest. When you're prepared to press your brain to its limits, you're ready for the mind blower.

Following your selection, the program constructs a puzzle and displays it at the upper-left corner of the screen. Your job is to match that puzzle in the workspace at the lower-right corner of the screen. What's more, you try to solve the puzzle in as little time as possible. A timer ticks away as you work. There's no limit to how much time you can take, but the timer lets you compare your progress to a previous performance, or against another player if you wish. Your fastest time during the current session will be displayed on the screen.

Each puzzle is composed of several horizontal rows of odd shapes. A tiny arrow to the right of the workspace points to the row you're currently working on. To work on different rows, you can move the arrow up and down with the I and M keys (use the up/down cursor keys on the IBM and TI, and be sure to press ALPHA LOCK on the TI). To move the row of shapes next to the arrow left or right, press the J or K key (left/right cursor keys on the IBM and TI). When you think you've matched a row to the puzzle pattern, start working on another row.

When you succeed in correctly matching all the rows, the program automatically signals that you've solved the puzzle. Then you can play again if you like.

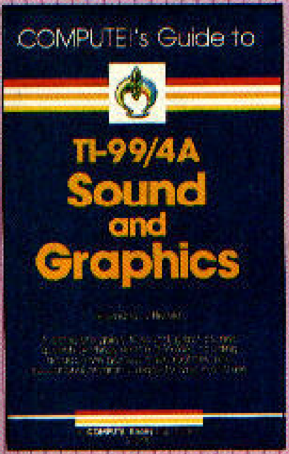
Helpful Hints

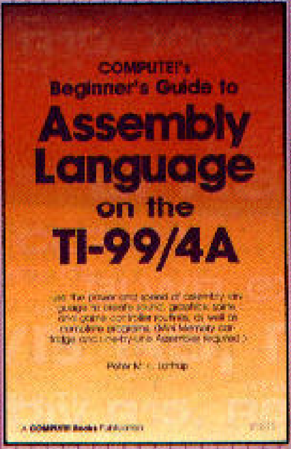
Because Mindbusters can generate a tremendous number of different puzzles, there are very few tricks to mastering it. I suggest you work from top to bottom or vice versa. The best tip I can offer after hours of my own mindbusting is to concentrate, concentrate, concentrate.

Important: When typing in the program, be extra careful with the long strings of characters at the beginning of the listing. These strings become the puzzle shapes. If you mistype or transpose a couple of characters when typing these strings, the program may still run, but it won't know when you've solved the puzzle. If you're using COMPUTE!'s "Automatic Proofreader" to enter the listing, remember that the Proofreader (except the IBM version) does not catch character-transposition errors.

A special offer from COMPUTE! Books for owners and users of TI-99/4A computers.

Buy any two of these books and get 15% off.
Buy all four of these titles for only $39.95 (a savings of $13.85—over 25% off).





COMPUTE!'s Guide to TI-99/4A Sound and Graphics

Raymond J. Herold

Using dozens of examples and clear, nontechnical explanations, *COMPUTE!'s Guide to TI Sound and Graphics* introduces you to the tremendous sound and graphics capabilities of your TI home computer. This book includes exciting arcade-style games, challenging educational programs, a versatile sprite editor, and many more useful, ready-to-type-in programs. Whether you're a beginning TI user or an experienced programmer, this is a book you'll refer to again and again.

$12.95
ISBN 0-942386-46-9

COMPUTE!'s TI Collection, Volume One

An anthology of *COMPUTE!'s* best games, applications, utilities, and tutorials for the TI-99/4A, this book includes more than 30 programs, most never before published.

"SuperFont" is an exceptionally powerful and easy-to-use character editor. Other utilities, such as "Sprite Editor" and "Sound Shaper," make graphics and sound programming simple. Games like "Worm of Bemer" will provide hours of fun, and applications like "Mailing List" let you use your TI to organize your home.

$12.95
ISBN 0-942386-71-X

COMPUTE!'s Beginner's Guide to Assembly Language on the TI-99/4A

Peter M. L. Lottrup

A clearly written, step-by-step tutorial on machine language (ML) programming for the TI-99/4A. Using the Line-by-Line Assembler, the basic concepts of machine language programming are fully explained and illustrated. Many practical, easy-to-follow ML examples are included, from elementary keyboard, joystick, and data handling routines to sophisticated redefined character, sprite manipulation, and even high-resolution bitmap mode techniques.

$14.95
ISBN 0-942386-74-4

33 Programs for the TI-99/4A

Brian Flynn

Contains something for everyone: chapters on games ("Rings and Poles"), money management ("IRA Planner"), business ("Internal Rate of Return"), simple statistics ("Mean, Variance, and Standard Deviation"), and more. A wide variety of applications software, plus games. Thirty-three ready-to-type-in programs at a low cost.

$12.95
199 pages, split wire bound
ISBN 0-942386-42-6

SAVE 25%

Order today!
This offer expires
May 30, 1985

Yes! I want to save money while enjoying COMPUTE! Books.
______ COMPUTE!'s Guide to TI-99/4A Sound and Graphics $12.95
______ COMPUTE!'s TI Collection, Volume One $12.95
______ COMPUTE!'s Beginner's Guide to Assembly Language on the TI-99/4A $14.95
______ 33 Programs for the TI-99/4A $12.95

All orders must be prepaid.
□ Payment enclosed (check or money order).
□ Charge: □ Visa □ MasterCard □ American Express
Acct. No. ______ Exp. Date ______
Signature ______
Name ______
Address ______
City ______ State ______ Zip ______

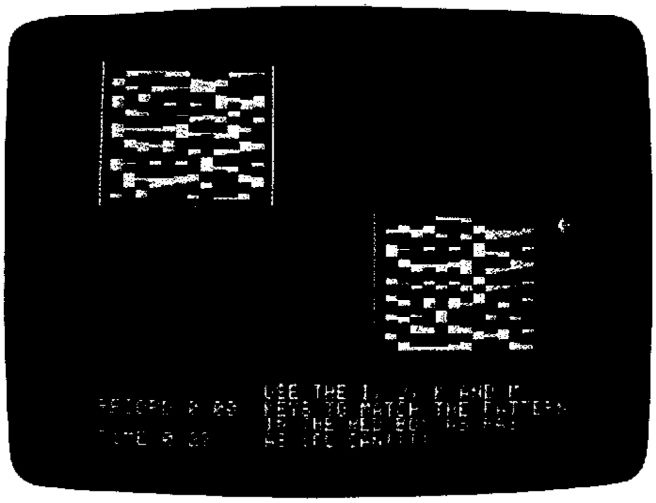
To order call toll-free, 1-800-334-0868 (in NC, call 919-275-9809) or mail this coupon with your payment to: COMPUTE! Books, P.O. Box 5406, Greensboro, NC 27403.

Please send me: □ 4 books for $39.95
□ 2 books at 15% off

NC residents add 4.5% tax ______
Add $2.00 per book for shipping and handling ______
Total paid ______

Please allow 4–6 weeks for delivery.

7541CC



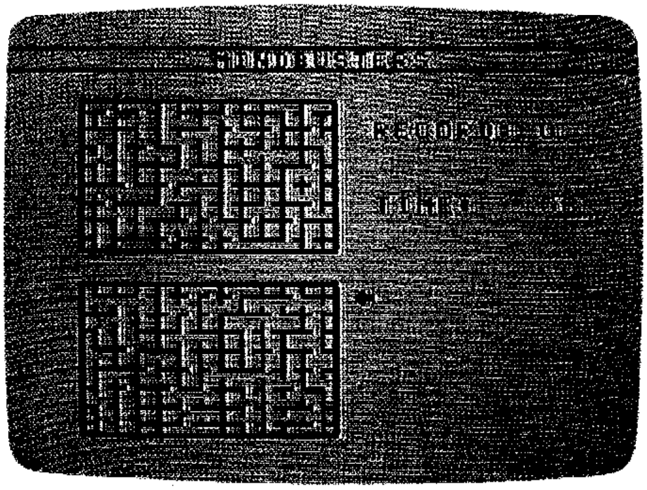
USE THE I, J, K AND M
KEYS TO MATCH THE PATTERN

Apple "Mindbusters."

```
      - 16300,Ø: POKE  - 16303,Ø: FOR I
      = 1 TO 50: NEXT
540  GOTO 520
550  POKE  - 16368,Ø:A = A - 176: IF A <
     1 OR A > 3 THEN 520
560  POKE 230,32: CALL  - 3086
570  IF A = 1·THEN D$ = A$
580  IF A = 2 THEN D$ = B$
59ØL IF A = 3 THEN D$ = C*
600  RETURN
610  REM  SHAPE DATA
620  FOR I = 36096 TO 36263: READ A:CS =
     CS + A: POKE I,A: NEXT
630  IF CS <  > 11534 THEN  PRINT "ERRO
     R IN FIRST SET OF DATA STATEMENTS.
     ": STOP
640  DATA  128,128,128,128,128,128,128,
     128
650  DATA    Ø,Ø,Ø,Ø,255,255,255,255
660  DATA  Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
670  DATA  Ø,Ø,Ø,Ø,Ø,Ø,Ø,255
680  DATA  Ø,Ø,Ø,Ø,Ø,Ø,255,255
690  DATA  255,255,Ø,Ø,Ø,Ø,Ø,Ø
700  DATA  255,255,255,Ø,Ø,Ø,Ø,Ø
710  DATA  Ø,Ø,Ø,Ø,Ø,255,255,255
720  DATA  24,24,24,31,31,24,24,24
730  DATA  24,24,24,31,31,Ø,Ø,Ø
740  DATA  Ø,Ø,Ø,248,248,24,24,24
750  DATA  Ø,Ø,Ø,31,31,24,24,24
760  DATA  24,24,24,255,255,Ø,Ø,Ø
770  DATA  Ø,Ø,Ø,255,255,24,24,24
780  DATA  24,24,24,248,248,24,24,24
790  DATA  24,24,24,248,248,Ø,Ø,Ø
800  DATA  24,24,24,255,255,24,24,24
810  DATA  204,153,51,102,204,153,51,10
     2
820  DATA  51,153,204,102,51,153,204,10
     2
830  DATA  8,12,14,127,127,14,12,8
840  DATA   255,Ø,Ø,Ø,Ø,Ø,Ø,Ø
850  REM  HROUT ML ROUTINE
860  FOR I = 768 TO 856: READ A:CK = CK
      + A: POKE I,A: NEXT
870  IF CK <  > 8413 THEN  PRINT "ERROR
      IN SECOND SET OF DATA STATEMENTS.
     ": STOP
880  RETURN
890  DATA  216,120,133,69,134,70,132,71
900  DATA  166,7,10,10,176,4,16,62
910  DATA  48,4,16,1,232,232,10,134
920  DATA  27,24,101,6,133,26,144,2
930  DATA  230,27,165,40,133,8,165,41
940  DATA  41,3,5,230,133,9,162,8
950  DATA  160,Ø,177,26,36,50,48,2
960  DATA  73,127,164,36,145,8,230,26
970  DATA  208,2,230,27,165,9,24,105
980  DATA  4,133,9,202,208,226,165,69
990  DATA  166,70,164,71,88,76,240,253
1000  DATA  255,255,255,255,255,255,255
     ,255
```

Program 6: Mindbusters For TI-99/4A

```
100 GOTO 150
110 FOR M=1 TO LEN(H$)
120 CALL HCHAR(R,C+M,ASC(SEG$(H$,M,
    1)))
130 NEXT M
140 RETURN
150 CALL CLEAR
160 SCR=3
170 HIGH=Ø
180 GOSUB 1470
190 CALL SCREEN(15)
200 FOR I=9 TO 12
210 CALL COLOR(I,1,1)
220 NEXT I
230 GOSUB 1730
240 GOSUB 1760
250 GOSUB 1730
260 PRINT
270 FOR J=1 TO 2
280 PRINT "   "&CHR$(135)&CHR$(129)&
    CHR$(129)&CHR$(129)&CHR$(129)&C
    HR$(129)&CHR$(129)&CHR$(129)&CH
    R$(129);
290 PRINT CHR$(129)&CHR$(129)&CHR$(
    129)&CHR$(129)&CHR$(132)
300 FOR I=1 TO 8
310 PRINT "   "&CHR$(130)&"
    {12 SPACES}"&CHR$(134)
320 NEXT I
330 PRINT "   "&CHR$(131)&CHR$(128)&
    CHR$(128)&CHR$(128)&CHR$(128)&C
    HR$(128)&CHR$(128)&CHR$(128)&CH
    R$(128);
340 PRINT CHR$(128)&CHR$(128)&CHR$(
    128)&CHR$(128)&CHR$(133)
350 NEXT J
360 CALL HCHAR(1,1,136,32)
370 CALL HCHAR(3,1,137,32)
380 H$="USE ARROW"
390 R=7
400 C=19
410 GOSUB 110
420 H$="KEYS TO"
430 R=9
440 GOSUB 110
450 R=11
460 H$="MATCH THE"
470 GOSUB 110
480 R=13
490 H$="1ST GRID"
500 GOSUB 110
510 H$="WITH THE 2ND"
520 R=15
```



"Mindbusters" on the TI-99/4A.

```
530 GOSUB 11Ø
54Ø H$="AS FAST AS"
55Ø R=17
56Ø GOSUB 11Ø
57Ø H$="YOU CAN !!!"
58Ø R=19
59Ø GOSUB 11Ø
6ØØ R=5
61Ø C=5
62Ø FOR N=1 TO 8
63Ø RANDOMIZE
64Ø PP(N)=INT(RND*56)+1
65Ø H$=SEG$(D$,PP(N),12)
66Ø GOSUB 11Ø
67Ø R=R+1
68Ø NEXT N
69Ø R=R+2
7ØØ FOR N=1 TO 8
71Ø RANDOMIZE
72Ø P(N)=INT(RND*56)+1
73Ø H$=SEG$(D$,P(N),12)
74Ø GOSUB 11Ø
75Ø R=R+1
76Ø NEXT N
77Ø CALL SOUND(1ØØ,44Ø,3)
78Ø CALL COLOR(KSET(Z),F(Z),1)
79Ø IF Z<>2 THEN 81Ø
8ØØ CALL COLOR(11,13,1)
81Ø FOR R=5 TO 2Ø
82Ø CALL HCHAR(R,2Ø,32,12)
83Ø NEXT R
84Ø TIME=Ø
85Ø R1=15
86Ø C1=19
87Ø CALL HCHAR(R1,C1,91)
88Ø H$="RECORD: "&STR$(HIGH)
89Ø R=6
9ØØ C=19
91Ø GOSUB 11Ø
92Ø H$="TIME:{3 SPACES}"&STR$(TIME)
93Ø R=1Ø
94Ø GOSUB 11Ø
95Ø CALL KEY(Ø,K,S)
96Ø TIME=TIME+.3
97Ø H$=STR$(INT(TIME))
98Ø C=27
99Ø R=1Ø
1ØØØ GOSUB 11Ø
1Ø1Ø IF (K<>67)*(K<>88)THEN 1Ø7Ø
1Ø2Ø CALL HCHAR(R1,C1,32)
1Ø3Ø R1=R1-(R1<>15)*(K=69)+(R1<>22)
     *(K=88)
1Ø4Ø CALL HCHAR(R1,C1,91)
1Ø5Ø TIME=TIME+.1
1Ø6Ø GOTO 95Ø
1Ø7Ø IF K<>68 THEN 11ØØ
1Ø8Ø P(R1-14)=P(R1-14)+(P(R1-14)<>1
     )
1Ø9Ø GOTO 112Ø
11ØØ IF K<>83 THEN 95Ø
111Ø P(R1-14)=P(R1-14)-(P(R1-14)<>5
     6)
112Ø H$=SEG$(D$,P(R1-14),12)
113Ø R=R1
114Ø C=5
115Ø GOSUB 11Ø
116Ø TIME=TIME+1
117Ø FOR X=1 TO 8
118Ø IF PP(X)<>P(X)THEN 95Ø
119Ø NEXT X
12ØØ H$="PUZZLE"
121Ø R=16
122Ø C=22
123Ø GOSUB 11Ø
124Ø H$="SOLVED!"
125Ø FOR I=22Ø TO 88Ø STEP 2Ø
126Ø CALL SOUND(5Ø,I,3)
127Ø NEXT I
128Ø R=18
129Ø GOSUB 11Ø
13ØØ H$="PLAY"
131Ø R=2Ø
132Ø C=23
133Ø GOSUB 11Ø
134Ø H$="AGAIN (Y/N)?"
135Ø C=2Ø
136Ø R=22
137Ø GOSUB 11Ø
138Ø CALL KEY(Ø,K,S)
139Ø IF S=Ø THEN 138Ø
14ØØ IF K=89 THEN 143Ø
141Ø IF K<>78 THEN 138Ø
142Ø STOP
143Ø IF (INT(TIME)>HIGH)*(HIGH<>Ø)T
     HEN 145Ø
144Ø HIGH=INT(TIME)
145Ø CALL CLEAR
146Ø GOTO 19Ø
147Ø FOR I=1 TO 29
148Ø READ A,A$
149Ø CALL CHAR(A,A$)
15ØØ NEXT I
151Ø CALL COLOR(14,14,1)
152Ø A$="geafebffagdafebffagafebfad
     adaeefadddgafefagfagcededfafeb
     dfccgedeafdf"
153Ø B$="mnhlphphilonpkhkllipklnppn
     phmiopjnmijnhpolpjnmlhiphphmom
     nnpopmhopihp"
154Ø C$="yyxxxyyxxyxyxyxyxyyyyyxyxx
     xyxxxyxyxxxxyxyxyyyyxyxyxxxyxy
     yyyyyxxyxxyx"
155Ø F(1)=5
156Ø KSET(1)=9
157Ø F(2)=13
158Ø KSET(2)=1Ø
159Ø F(3)=2
16ØØ KSET(3)=12
161Ø RETURN
```

```
1620 DATA 97,ØØØØØØØØFFFFFFFF,98,FF
     ØØØØØØØØØØØØØØ,99,ØØØØØØØØØØØØØØ
     ØØFF
1630 DATA 100,ØØØØØØØØØØØØFFFF,1Ø1,
     FFFFØØØØØØØØØØØØ,1Ø2,FFFFFFØØØ
     ØØØØØØØ
1640 DATA 1Ø3,ØØØØØØØØØØFFFFFF,1Ø4,
     1818181F1F181818,1Ø5,1818181F1
     FØØØØØØ
1650 DATA 1Ø6,ØØØØØØF8F8181818,1Ø7,
     ØØØØØØ1F1F181818,1Ø8,181818FFF
     FØØØØØØ
1660 DATA 1Ø9,ØØØØØØFFFF181818,11Ø,
     181818F8F8181818,111,181818F8F
     8ØØØØØØ
1670 DATA 112.181818FFFF181818.12Ø.
     CC993366CC993366,121,3399CC663
     399CC66
1680 DATA 128,FFFFØØØØØØØØØØØØ,129,
     ØØØØØØØØØØØØFFFF,13Ø,Ø3Ø3Ø3Ø3Ø
     3Ø3Ø3Ø3
1690 DATA 131,Ø3Ø3ØØØØØØØØØØØØ,132,
     ØØØØØØØØØØØØCØCØ,133,CØCØØØØØØ
     ØØØØØØØ
1700 DATA 134,CØCØCØCØCØCØCØCØ,135,
     ØØØØØØØØØØØØØ3Ø3,91,ØØ1Ø3Ø7FFF
     7F3Ø1Ø
1710 DATA 136,ØØØØØØØØØØØØFFFF,137,
     FFFFØØØØØØØØØØØØ
1720 PRINT "   }{{{{{{{{{{{{{{"&CHR$(
     127)
1730 CALL CLEAR
1740 PRINT TAB(1Ø);"MINDBUSTERS"
1750 RETURN
1760 PRINT : : : :
1770 PRINT TAB(7);"DO YOU WANT TO:"
     : : :
1780 PRINT TAB(6);"1 BEND YOUR MIND
     ?": :
1790 PRINT TAB(6);"2 BRUISE YOUR MI
     ND?": :
1800 PRINT TAB(6);"3 BLOW YOUR MIND
     ?": : : : : :
1810 CALL HCHAR(5,1,136,32)
1820 CALL HCHAR(7,1,137,32)
1830 CALL KEY(Ø.K.S)
1840 CALL SCREEN(SCR)
1850 SCR=SCR-(SCR<16)+(SCR=16)*14
1860 IF S=Ø THEN 183Ø
1870 CALL SCREEN(15)
1880 Z=K-48
1890 IF (Z<1)+(Z>3)THEN 183Ø
1900 IF Z>1 THEN 193Ø
1910 D$=A$
1920 RETURN
1930 IF Z=3 THEN 196Ø
1940 D$=B$
1950 RETURN
1960 D$=C$
1970 RETURN
```

SEE PAGES 66–85 IN THIS ISSUE FOR PROTECTO ENTERPRIZES' SUPER SALE AND YOU WILL SEE WHY WE SAY . . .

WE LOVE OUR CUSTOMERS

PROTECTO
ENTERPRIZES

Required Reading for your COMMODORE 64

THE GRAPHICS BOOK FOR THE COMMODORE
SCIENCE & ENGINEERING FOR THE COMMODORE
IDEAS FOR USE ON YOUR COMMODORE
ADVENTURE GAMEWRITERS HANDBOOK FOR

TRICKS & TIPS Treasury of programming techniques. Advanced graphics, easy input, enhanced BASIC CP/M, character sets, more. **275 pages $19.95**

GRAPHICS BOOK Basic to advanced, best reference available. Sprite animation, Hires, Multicolor, lightpen, IRQ, 3D graphics. **350 pages $19.95**

SCIENCE & ENGINEERING Discuss variable types, computational accuracy, sort algorithms, more. Topics from chemistry, physics, biology, astronomy, electronics. **250 pages $19.95**

ANATOMY OF C-64 Insider's guide to the '64 internals. Graphics, sound, I/O, kernal, more. Complete commented ROM listings. **300 pages $19.95**

PEEKS & POKES Programming quickies that will simply amaze you. This guide is packed full of techniques for the BASIC programmer. **180 pages $14.95**

ANATOMY OF 1541 DISK DRIVE Bestselling handbook on floppy disk. Explains disk files with examples and utilities. Includes complete commented 1541 ROM listings. **320 pages $19.95**

Other titles available. Phone for name of nearest dealer Phone:(616) 332-4154

Postage & handling $4.00 ($8.00 foreign). Money order, checks, VISA, MC, AMEX. Michigan residents (incl. 4% sales tax).

CANADA: Book Center, Montreal (514) 332-4154

You Can Count On
Abacus Software

P.O. Box 7211 Grand Rapids, MI 49510 - Telex 709-101 - Phone 616/241-5510

C. Regena

Matching Quiz

This month's column presents a general matching-quiz program that can be adapted to any topic. It contains no graphics or sound, so it should be easy to translate to other computers. Feel free to add your own graphics and sound to enhance your particular quiz.

The sample program is a quiz of terms and their definitions. This particular quiz can be used in a computer literacy class for learning general computer terminology.

First the program prints a definition on the screen followed by 12 possible terms. The user must press the letter corresponding to the term defined. If the answer is correct, the program continues and that definition will not appear again. If the answer is incorrect, the program gives the correct answer and the definition *will* appear again.

The score is kept by keeping track of how many times an answer is attempted. A perfect score in this case would be 12. Each time a definition is shown, the score is incremented.

If you want to use this matching quiz for several different topics, type in and save the program consisting of lines 100 through 710. Now, to build a custom program, start with this basic structure and then add DATA statements starting at line 720. Then save the quiz on a different tape or with a different name on the disk. Different quizzes will simply have different DATA statements. You may also need to change the instructions.

Creating DATA Statements

Notice that each DATA statement contains two items separated by a comma. The first item is the term, and the second item is the corresponding definition. If the definition contains a comma, it must be surrounded by quotation marks. Otherwise, the computer will mistake the characters after the comma for another DATA element.

On a quiz for a different topic, use the same idea—put matching parts in the same DATA statement.

Line 110 DIMensions arrays for the quiz. Since this quiz has 12 definitions and terms, the numbers in the DIM statement are 12. You will need to adjust this for the number of items in your own quiz. Line 120 sets the variable N to 12 for the 12 items in this example program. If you have a different number of items, be sure to change this line.

Lines 130–200 clear the screen and print the instructions. Lines 210–230 READ from the DATA the 12 words (W$) and their corresponding definitions (D$). Within the FOR-NEXT loop, a counter with the variable name A varies from 1 to 12. Line 220 looks for DATA statements and reads in order first a word W$(A), then the definition D$(A). The number A keeps them matched up properly. Make sure when you type your DATA statements that you have matched pairs of items (separated by commas).

Program Setup

Lines 240–270 wait for the user to press ENTER before clearing the screen to start the quiz. Line 280 initializes the score (SC) to zero at the beginning of each quiz.

Lines 290–310 set up a temporary word file array, T$(A), which is the same as the original W$ array. This temporary array is used in choosing the terms for the quiz.

Lines 320–550 perform the quiz for the number of items to be matched, N, or in this case 12. Line 330 increments the score SC for each time a definition is shown.

Line 340 clears the screen. Lines 350–370 randomly choose one of the terms which has not previously been matched correctly. The term chosen is denoted by the number R. Line 380 prints the definition D$(R) corresponding to the term chosen.

Lines 390–420 print all of the terms possible for answers with a letter to indicate the answer. Line 430 sounds a prompting tone. Lines 440–460 accept the user's answer, making sure the key pressed is an acceptable letter of one of the terms, then prints the letter chosen.

Evaluating The Answer

Line 470 tests the user's response with the correct answer stored in R. If the answer is incorrect, lines 480–510 print the correct answer, wait for the user to press ENTER, then branch back to line 330 to increment the score and print the next definition. If the answer is correct, lines 520–540 print the message CORRECT!, set T$(R) equal to the null string so the term cannot be chosen again, and then wait for the user to press ENTER. Line 550 increments P for the loop counter to go to the next problem.

After the quiz is complete and all terms have been correctly matched, line 560 clears the screen. Lines 570–580 print the possible score and the user's score. Lines 590–600 print a message if there is a perfect score.

Lines 610–670 present the option to try the quiz again or to end the program.

Lines 680–710 contain the subroutine to wait for the user to press the ENTER key before continuing the program.

Lines 720–840 in this program contain the data for the quiz. Notice that some of the definitions contain extra spaces. These are used to print the definition on the 28-column screen without splitting words.

Customizing The Quiz

Now to change the topic of the quiz. Decide how many items will need to be matched. Keep in mind how it will look when printed on the 24-row screen. Change the DIMension statement of line 110 and the definition of N in line 120 to reflect the number of items.

Next add the DATA statements starting with line 720. For example, if you want a quiz on BASIC programming commands, a typical DATA statement might be:

720 DATA GOTO,Command to transfer program control

A history quiz might contain:

720 DATA 1492,Columbus discovered America.

An algebra quiz could use:

720 DATA x=2,x+5=5x−3

A states and capitals quiz could use:

720 DATA Providence,Rhode Island

When typing the DATA statements, make sure there are matching pairs. If there are short words, you may put more than one matching pair in a DATA statement—just be sure to use commas to separate each item. With longer phrases, make sure you use spaces to print the phrase properly on the screen without splitting words.

Remember that you can add your own sound effects and graphics for positive reinforcements on correct answers. You may also wish to use graphics and sound as part of the matching process.

If you wish to save typing effort and obtain a copy of this program, send a blank cassette or disk, a stamped, self-addressed mailer, and $3 to:

C. Regena
P.O. Box 1502
Cedar City, UT 84720

Please be sure to specify the title of the program and the type of computer you use.

Matching Quiz For TI

Please refer to "COMPUTE!'s Guide To Typing In Programs" before entering this listing.

```
100 REM  MATCHING QUIZ
110 DIM W$(12),T$(12),D$(12)
120 N=12
130 CALL CLEAR
140 PRINT TAB(9);"CHAPTER 1"
150 PRINT :::"A DEFINITION WILL BE
    GIVEN."
160 PRINT :"CHOOSE THE TERM WHICH"
170 PRINT :"MATCHES THE DEFINITION.
    "
180 PRINT :"PRESS THE LETTER OF THE
    "
190 PRINT :"ANSWER."
200 PRINT :"THERE WILL BE";N;"PROBL
    EMS."
210 FOR A=1 TO N
220 READ W$(A),D$(A)
230 NEXT A
240 PRINT ::"PRESS <ENTER> TO START
    ."
250 CALL KEY(0,K,S)
260 IF K<>13 THEN 250
270 CALL CLEAR
280 SC=0
290 FOR A=1 TO N
300 T$(A)=W$(A)
310 NEXT A
320 FOR P=1 TO N
330 SC=SC+1
340 CALL CLEAR
350 RANDOMIZE
360 R=INT(N*RND)+1
370 IF T$(R)="" THEN 360
380 PRINT D$(R)::
390 FOR A=1 TO N
400 PRINT CHR$(64+A);"   ";W$(A)
410 NEXT A
420 PRINT
430 CALL SOUND(150,1500,2)
440 CALL KEY(0,K,S)
```

```
450 IF (K<65)+(K>64+N)THEN 440
460 PRINT CHR$(K)::
470 IF K-64=R THEN 520
480 PRINT "THE CORRECT ANSWER IS"
490 PRINT CHR$(R+64);"--";W$(R)
500 GOSUB 680
510 GOTO 330
520 PRINT "CORRECT!"
530 T$(R)=""
540 GOSUB 680
550 NEXT P
560 CALL CLEAR
570 PRINT "THERE WERE";N;"DEFINITIO
    NS."
580 PRINT :"YOUR SCORE: ";SC;"ANSWE
    RS":::
590 IF SC<>N THEN 610
600 PRINT "GOOD WORK!":::
610 PRINT "PRESS 1 TO TRY AGAIN"
620 PRINT "{6 SPACES}2 TO END PROGR
    AM"
630 CALL KEY(0,K,S)
640 IF K=49 THEN 270
650 IF K<>50 THEN 630
660 PRINT ::"2  END":::
670 STOP
680 PRINT :"PRESS <ENTER>.";
690 CALL KEY(0,K,S)
700 IF K<>13 THEN 690
710 RETURN
720 DATA DOCUMENTATION,THE BOOKS AN
    D MANUALS THAT  ACCOMPANY A COM
    PUTER-RELATEDPRODUCT
730 DATA SYSTEM,A SET OR ARRANGEMEN
    T OF{5 SPACES}PARTS ACTING TOGE
    THER TO{4 SPACES}PERFORM A FUNC
    TION
740 DATA INFORMATION SYSTEM,"A SYST
    EM THAT TAKES INPUT,  PROCESSES
     IT, AND PRODUCES  INFORMATION
    AS OUTPUT"
750 DATA COMMUNICATION SYSTEM,"A SY
    STEM THAT CONSISTS OF A SENDER,
    A PHYSICAL CHANNEL,  AND A RECE
    IVER"
760 DATA HARDWARE,THE PHYSICAL COMP
    ONENTS{5 SPACES}ASSOCIATED WITH
     A COMPUTER  OR OTHER SYSTEM
770 DATA SOFTWARE,PROGRAMS THAT CON
    TROL THE{3 SPACES}FUNCTIONS OF
    SYSTEMS
780 DATA NETWORK,TWO OR MORE COMMUN
    ICATING{3 SPACES}DEVICES THAT A
    RE CONNECTED  TOGETHER
790 DATA APPLICATION,WHAT IS DONE W
    ITH COMPUTERS
800 DATA CIRCUIT,AN INTERCONNECTED
    SET OF{4 SPACES}COMPONENTS THAT
     PERFORM AN ELECTRONIC FUNCTION
810 DATA BINARY SIGNAL,A COMPUTER C
    IRCUIT THAT IS  REPRESENTED BY
    TWO DIFFERENTLEVELS OF CURRENT
820 DATA DATA,"FACTS, NUMBERS, AND
    SYMBOLS PROCESSED BY A COMPUTER
     TO  PRODUCE INFORMATION"
830 DATA BINARY DIGIT (BIT),A BASIC
     BUILDING BLOCK OR{3 SPACES}UNI
    T OF INFORMATION USED IN COMPUT
    ER SYSTEMS
840 END
```

THE BEGINNER'S PAGE

Tom R. Halfhill, Editor

Programs Within Programs

Imagine what your life would be like if every time you had to perform a routine task—such as starting your car or switching on a TV—you had to think really hard about it, almost as if you were learning the task for the first time. Starting a car doesn't seem too difficult, but it does require you to execute a number of smaller tasks in exactly the same sequence each time. You have to find the right key, unlock the door, grasp the handle, pull open the door, climb into the seat, stick the key into the ignition, twist the key, and press the gas pedal.

Yet, unless the car is brand-new or belongs to someone else, you can probably do all of this with your eyes closed, like a blindfolded soldier reassembling his rifle. That's because you've performed the actions so many times that they're carved into your unconscious. You just think *start the car*, and a little "program" takes over.

When you think about it, your brain stores thousands of such tiny programs. They let you perform everyday tasks almost on autopilot. Without them, every routine action would be like a new learning experience. Life might be more interesting, like a young child's, but you'd be a lot less efficient.

Computer programs can benefit from the same sort of efficiency. After all, a program at its most basic level is just a list of instructions telling the computer how to perform some kind of job. That job might be something as simple as adding two numbers or something as complex as modeling the economy of a large nation. Still, even simple jobs can often be broken down into several smaller tasks which are executed repeatedly. So why make the computer do things the hard way? Why not equip your programs with the same kind of subprograms that your brain seems to use to automate routine tasks?

This concept of smaller programs within larger programs is so powerful that virtually every computer language offers some way to do it. By identifying these repetitive tasks and turning them into subprograms or *subroutines*, you can write programs that run faster, consume less memory, and are easier to understand and modify.

When To Use A Subroutine

Your brain acquires a subroutine by rote—it subconsciously memorizes a task that you perform over and over again. Today's computers aren't quite intelligent enough to learn this way, so you have to spell it out for them more literally with BASIC commands.

First you have to decide when to take a piece of a program and make it into a subroutine. This judgment comes naturally after a while, but as a general rule, any small task which is performed more than once in a program is a candidate for a subroutine.

Once you've identified this task, you write the little routine and make the program detour to those lines whenever you need to perform that task. At the end of each subroutine, you use the command RETURN to automatically go back into the main program and proceed with other things.

Let's try an example. Assume you're writing a program that frequently pauses and asks the user to press a key. With no subroutines, this is how clumsy the program would be:

```
90 DIM A$(1):REM This line for Atari only
100 PRINT "During the Civil War,"
110 PRINT "more American soldiers died"
120 PRINT "than in all other"
130 PRINT "American wars combined."
140 PRINT "PRESS C AND RETURN TO
    CONTINUE";
150 INPUT A$
160 IF A$<>"C" THEN GOTO 140
170 PRINT "Poor medical care accounted"
180 PRINT "for many casualties,"
190 PRINT "but outmoded military tactics"
200 PRINT "were also to blame."
210 PRINT "PRESS C AND RETURN TO
    CONTINUE";
220 INPUT A$
230 IF A$<>"C" THEN GOTO 210
...
```

Notice how the lines which ask the user to press a key (lines 140–160 and 210–230) are simply repetitious; only the line number references are different.

In each case these lines keep printing the prompt PRESS C AND RETURN TO CONTINUE until the user presses the C key. (Make sure to press a capital C if you try running this example. If you have a TI-99/4A, change every occurrence of THEN GOTO to THEN in this and all following examples.) A little three-line routine like this one might not seem like much, but if it's repeated throughout a long program, considerable space and programming time would be wasted. This is an ideal candidate for a subroutine.

Why Not GOTO?

At this point, you might be thinking about building a subroutine with the GOTO command. After all, a subroutine requires a detour from the main program, and GOTO is a programming detour (see last month's column). Why not just jump to the subroutine with GOTO and then exit from it the same way? The program might look like this:

```
90 DIM A$(1):REM This line for Atari only
100 PRINT "During the Civil War,"
110 PRINT "more American soldiers died"
120 PRINT "than in all other"
130 PRINT "American wars combined."
140 GOTO 1000
150 PRINT "Poor medical care accounted"
160 PRINT "for many casualties,"
170 PRINT "but outmoded military tactics"
180 PRINT "were also to blame."
190 GOTO 1000
200 PRINT "For instance, many battles"
210 PRINT "were fought with mass charges"
220 PRINT "of infantry and cavalry."
230 GOTO 1000
...
1000 PRINT "PRESS C AND RETURN TO
     CONTINUE";
1010 INPUT A$
1020 IF A$<>"C" THEN GOTO 1000
1030 GOTO 150
```

At first this seems to fit the bill. The lines which await the user's keystroke are grouped together in a neat subroutine at the end of the program. All it takes is a simple instruction—GOTO 1000—to activate (or *call*) the subroutine.

If you try running the program, however, a problem soon becomes apparent. The subroutine works great the first time it's called. The first paragraph of text appears on the screen, followed by the prompt, and the program continues printing when you press C. But after the second time the subroutine is called, the program prints the second paragraph all over again! In fact, it keeps printing the same paragraph no matter how many times you press C—it never reaches the third paragraph at all.

GOTO is the culprit. GOTO 1000 works okay for *calling* the subroutine, because the routine is always at line 1000. But GOTO doesn't work so well when *returning* from the subroutine. The line number in the routine's final GOTO statement is fixed (GOTO 150), but the line number where the program should continue after calling the routine keeps changing. What's needed is a substitute for GOTO that always knows how to pick up where the program left off. That substitute is the pair of commands GOSUB and RETURN.

GOSUB: A GOTO With Brains

If you understood how the above programs work, you'll have no trouble at all grasping GOSUB and RETURN. GOSUB (which means *GOto SUBroutine*) is merely a smarter version of GOTO. The statement GOSUB 1000 does the same thing as GOTO 1000—it detours the program to line 1000. However, it also makes the computer remember *where it detoured from.* Then, when a RETURN statement is encountered, the program automatically returns from the subroutine and begins executing the statement which immediately follows the original GOSUB.

Here's how the previous example would look after GOSUB and RETURN are substituted for the GOTO statements that caused the problem:

```
90 DIM A$(1):REM This line for Atari only
100 PRINT "During the Civil War,"
110 PRINT "more American soldiers died"
120 PRINT "than in all other"
130 PRINT "American wars combined."
140 GOSUB 1000
150 PRINT "Poor medical care accounted"
160 PRINT "for many casualties,"
170 PRINT "but outmoded military tactics"
180 PRINT "were also to blame."
190 GOSUB 1000
200 PRINT "For instance, many battles"
210 PRINT "were fought with mass charges"
220 PRINT "of infantry and cavalry."
230 GOSUB 1000
240 END
...
1000 PRINT "PRESS C AND RETURN TO
     CONTINUE";
1010 INPUT A$
1020 IF A$<>"C" THEN GOTO 1000
1030 RETURN
```

Think how much memory (and programming time) you could save by simply inserting a GOSUB 1000 statement whenever you want the user to press a key to continue, instead of redundantly entering the routine itself each time you need it. The memory savings are even more dramatic with longer subroutines.

For that reason alone, GOSUB and RETURN are worth their weight in RAM chips. Yet memory conservation is only one advantage of using subroutines in your programs. We already mentioned how they can increase execution speed and help make programs easier to understand and modify. But they can also drastically reduce the time you spend writing and debugging a program. Once you get a subroutine up and running without bugs, you can call it with confidence whenever necessary. If an error does result, you can be fairly certain that something outside the subroutine is causing the error. This narrows down your search for the elusive bug.

Subroutines can also make it less intimidating to write large, complex programs. By breaking a big job down into many smaller jobs, and then tackling them one at a time, the program seems to fall together much more easily. In fact, many programmers keep a library of frequently used subroutines and stick them into new programs wherever needed.

Questions Beginners Ask

Q In manuals, books, and articles, I keep seeing the term "default." What does default mean?

A *Default* means the way something starts out, its normal condition. For example, many computer games default to one-player mode. If there are two players, you have to let the game know by pressing a special key.

In computer terminology, default can refer to the standard setting of a switch, the screen colors when you first turn on the computer, the number stored in a memory location before it's altered by a program, and many other things. For example, the LOAD command on a Commodore 64 or VIC-20 defaults to tape instead of disk. If you type:

```
LOAD"PROGRAM NAME"
```

the computer assumes you are loading from the cassette recorder and responds PRESS PLAY ON TAPE. To load a program from the disk drive, you have to add a device number to the command which overrides the default:

```
LOAD"PROGRAM NAME",8
```

Another example is a dot-matrix printer which defaults to a standard typeface. To print in a special typeface such as bold or italics, you must send the printer a command (usually from within a program) which overrides the default setting.