$8.95

101 ATARI Computer Programming Tips & Tricks

The exciting software ideabook, overflowing with pro techniques, hints, secrets, shortcuts, with 101 ready-to-run programs for the ATARI 400/800 personal computers.

by Alan North

101 ATARI Computer Programming Tips & Tricks

101 ATARI Computer Programming Tips & Tricks

by Alan North

ARCsoft Publishers
WOODSBORO, MARYLAND

FIRST EDITION
FIRST PRINTING

Library of Congress LC number: 82-18179

ISBN 0-86668-022-5

Preface

The ATARI computer is one of the world's most popular computer systems. Its lightweight, desktop design and powerful version of the BASIC programming language place it at the forefront of the new wave of personal computers for office, school and home use.

Anything but a toy, its hardware configuration and system software make it a highly useful tool in the business environment and classroom as well as for practical jobs around the home.

In fact, the system software built into your ATARI computer is so flexible and versatile that the need for this book became apparent. There are so many computing tasks which can be accomplished with the ATARI system that an introduction to the many techniques is needed.

Applications software are the programs which make a computer do what you want. This book is written for those newcomers and beginners, as well as advanced novices and student programmers, who would like to tap the vast resources available in the ATARI computer package. This book, it is hoped, will guide by example, instruct, and provide insights into the many ways the BASIC language built into the ATARI computer can be put to use.

This book is a companion volume to *31 New ATARI Computer Programs for Home, School & Office.*

—Alan North

Table of Contents

Text on Text

Introduction

This book is designed as an idea starter and thought provoker for beginners and newcomers, as well as advanced novices and student programmers, who have read the manual which came with the ATARI Model 400 or Model 800 personal computer and have read something very elementary on BASIC programming and now need some good ideas.

This handbook gives you 101 different, complete programs which will stand alone and run by themselves for learning purposes or which can be, if you wish, incorporated into larger sets of instructions to your computer.

These programs are written to be typed into your ATARI 400/800 just as you find them here with no programming needed. We assume you know how to turn on your computer and how to go about typing in a program but you don't have to be a computer programmer to get started using these pieces of software. Many tips in this book, in fact, will be of interest to old-timers in the program-writing game since we have presented many powerful new twists aimed at making the computer do more work more quickly.

Amidst our 101 tips are countless secrets, shortcuts, tricks, hints, techniques and make-it-easier instructions. Each is intended to make you a more versatile programmer and to make your programming effort lighter. Use this book to stimulate your thinking about how to approach various software problems. Use it to get good ideas for new and different approaches to all your programming goals. As you grow and develop as a program writer, modify these programs and make your computer do even more!

The printout

As each program in this book was tested on an ATARI Model 400/800, it was printed on paper directly from the computer's program memory. In this manner, we have attempted to prevent those common typographical errors which occur when computer programs are typeset by hand and proofread. The human hand and eye have not come between the computer and the printed page.

You should find no errors at all in the programs in this book. Should you find something which appears to be in error, please send a postcard or letter to the author in care of *ARCsoft Publishers*, P.O. Box 132, Woodsboro, MD 21798 USA. He would like to know of problems so they can be corrected in future editions of this work.

As you read through this book, you will find very few uses of the BASIC word REM. REM stands for "remark" and is the BASIC instruction to the computer to "ignore this line." REMs are used by some programmers to pad out programs and make them look longer, and to include sometimes-useful reminders about what certain segments of a program do. The trouble is REMs waste lots of precious memory space and add considerably to the typing time when entering a program into the computer.

The author's training in writing BASIC-language computer programs required a sharp editing pencil. REMarks and software explanations were out. Honing, fine-tuning and waste trimming were in. Use of coding-form program-writing worksheets, such as the *ATARI Computer BASIC Coding Form* published by *ARCsoft Publishers,* was in. The objective always was—and still is—to make the most efficient use of available memory.

Even though they may be headed toward the same goal, no two programmers will write the exact same list of BASIC program lines from scratch. As you load these pieces of software into your ATARI computer, you'll make modifications to suit your personal needs and interests. For instance, exact wording of PRINT statements can be changed. Or two or more programs can be combined into one grand scheme. Your applications for these programs will vary.

Standalone vs. subroutine

All of the programs in this book can be used as portions of larger lists of instructions to your computer. That is, they can be written in as GOSUB or GOTO objects. To do so, make appropriate changes to the first line (usually numbered 10 in this book) and the last line of each program. If you wish to create a subroutine, remember that every GOSUB must have a RETURN. RETURN must be the last line of a subroutine.

If you work one of these programs into a larger set of instructions, be especially careful of your memory (variables) names. They must agree with and fit into those you are using in the main program. Also, be careful of line numbering.

If you want to load more than one of these programs into your ATARI computer at the same time, be sure to use different sets of line numbers.

Buzzer and clear screen

The crooked, upward-pointing arrow used on the video display by the ATARI is not found on line printers. So, we have used the right-pointing bracket to represent that symbol in this book. In all cases, you will find a REM in the same line indicating exactly what we have done. For example, you may see a line like this:

```
10  PRINT "}":REM CLEAR SCREEN
```

Such a line is an instruction to the computer to wipe everything from the video display screen. That is, clear the screen.

To type in such an instruction, type in the line number, the word print and the first quotation mark. Then press the ESC key, press and hold the CTRL key, and press the CLEAR key. After that, type in the second quotation mark and the rest of the line.

The crooked arrow display for screen-clear is ESC and CTRL and CLEAR.

If you wish to make the internal *buzzer* sound off, use ESC and CTRL and the number 2 key. Here's a line which you might see in this book:

```
10 PRINT "}":REM BUZZER
```

Here, you type in the line number, the word PRINT and the first quotation mark. Then press ESC, press and hold CTRL and press 2. After that, type in the second quotation mark and the rest of the line.

The crooked-arrow video display for buzzer is ESC and CTRL and 2.

Other computers

The 101 programs in this book will run on other computers such as the TRS-80 Models I, II, III, 16, Color, Pocket, Apple, Commodore PET, 64, VIC-20, MAX, Timex 1000, Sinclair ZX-81, MicroAce, IBM Personal Computer, Texas Instruments TI-99/4A and many other systems.

However, to do so you will have to make some minor modifications to program lines. Each computer hardware manufacturer offers a version of BASIC slightly different from all others. Thus, BASIC programs aren't exactly interchangeable from system to system.

Video-screen graphics commands, especially, vary from machine to machine. Also, if you use other than an ATARI Model 400 or Model 800, line number, logical tests, multiplication symbols, print statements, and other instructions may vary. If you want to run these programs on a computer other than the ATARI 400/800, check the owner's manual for that computer to see which BASIC words are built into that computer, and how they will work on that computer, before running.

Many programmers use the words ENTER and RETURN interchangeably when writing about typing programs into a computer. You may spot examples of this common use in this book. When we say ENTER a program or press RETURN, we mean tell the computer you have ended a line of a program or of data and are sending the info in.

By the way, the author would like to hear of suggestions you may have for improvements in future editions of this work or ideas you may have for future volumes in this series for the ATARI Computer. Please address your comments, suggestions, thoughts and ideas to the author in care of *ARCsoft Publishers.*

Book parts

Besides this Introduction, this book has seven important sections: *Fun & Games,* a set of new and different ideas you can put to use in writing games for your ATARI computer;

Text on Text, programming tips for handling words in your ATARI;

Gee Whiz, a group of quick-to-type programs you can use to excite your friends and family;

Number Crunching, tips and tricks for handling numbers in your ATARI;

Money Matters, interesting programs to help you handle your household budget as well as store and office routines for businessmen;

Colorful Graphics, designed to help you get more from your machine;

And a handy *Appendix* of BASIC words and error messages as used in the ATARI computers.

Naturally, these sections have been arbitrarily divided and may not be exactly as you would have sorted the programs herein. These divisions are not hard and fast lines, never to be crossed. Rather, programs shown here for use in games might be useful in a business program. Or part of a business program might be useful in a classroom or in a game. Try them all. You'll learn something from each!

Fun & Games

1 Coin Toss

Here's a handy way to settle arguments. Toss a coin. Only this time, let the computer do the work!

Type in the program. Run it. The computer will report *heads* or *tails* after each toss.

For a new toss, press the RETURN key on your computer's keyboard.

Line 10 clears the screen. A random number—either zero or one—is generated at line 20 and tested to see if it is a zero. If it is, the computer prints *heads*. If not, the computer drops to line 30 where it prints *tails*. Lines 50, 60 and 70 accomplish the restart when you press RETURN.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 IF (INT(2*(RND(1))))<1 THEN PRINT
   "HEADS":GOTO 40
30 PRINT "TAILS"
40 PRINT :PRINT
50 PRINT "FOR MORE, PRESS RETURN"
60 DIM K$(1):INPUT K$
70 CLR :GOTO 10
```

2 Traditional Dice Roll

Here's a simple, brief way to roll and display results for two dice.

Lines 100-110 get a random number between 1 and 6 and store it in A. Lines 200-210 get another random number from 1 to 6 and store it in B.

Lines 300-310 print the contents of A and B along with a suitable message.

Program Listing

```
10 DIM K$(1)
20 PRINT "}":REM SCREEN CLEAR
100 A=INT(7*(RND(1)))
110 IF A<1 THEN 100
200 B=INT(7*(RND(1)))
210 IF B<1 THEN 200
300 PRINT "FIRST DICE:",A
310 PRINT "SECOND DICE:",B
400 FOR L=1 TO 10:PRINT :NEXT L
410 PRINT "TO ROLL AGAIN, PRESS RETURN"
420 INPUT K$
430 GOTO 20
```

3 See Two Dice

Here's a quick way to add real dice to any fun program you are designing for your Atari.

This program rolls two dice and lets you see the results, as with real dice. This is especially useful in those games where it is important to see the value of each.

The subroutine in lines 100-140 generates the necessary pair of random numbers. Lines 60, 70 and 80 make the display you want.

Note that lines 60 and 80 each have nine asterisks. Line 140 is RETURN and must be the last line in the program.

After you type in and RUN the program, press any key on your Atari's keyboard to roll the dice.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 DIM K$(1)
30 PRINT "TO ROLL TWO DICE, PRESS RETURN"
40 INPUT K$:PRINT :PRINT
50 GOSUB 100
```

```
60 PRINT "*********"
70 PRINT "* ";DL;" * ";DR;" *"
80 PRINT "*********"
90 PRINT :PRINT :GOTO 30
100 DL=INT(7*(RND(1)))
110 IF DL<1 THEN 100
120 DR=INT(7*(RND(1)))
130 IF DR<1 THEN 120
140 RETURN
```

4 See Four Dice

Two dice not enough for your game? Here's how to see four dice after a roll!

Naturally, this program works just like the program in tip number two except that the FOR/NEXT loop in lines 50-140 makes the Atari roll and display four times rather than two times. If you need six, eight or ten dice on display, change the number two in line 50 to three, four or five.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 DIM K$(1)
30 PRINT "TO ROLL DICE, PRESS RETURN"
40 INPUT K$:PRINT :PRINT
50 FOR L=1 TO 2
60 DQ=INT(7*(RND(1)))
70 IF DQ<1 THEN 60
80 DR=INT(7*(RND(1)))
90 IF DR<1 THEN 80
100 PRINT "*********"
110 PRINT "* ";DQ;" * ";DR;" *"
120 PRINT "*********"
130 PRINT
140 NEXT L
150 PRINT :CLR :GOTO 20
```

5 Secret Message

Secret messages can be lots of fun! They often are composed of codes in which letters of the alphabet have been replaced by numbers.

In this easy-to-use program, the computer generates a list of pseudorandom numbers and assigns one number to each letter of the alphabet. You use the numbers, in lieu of letters, to write notes to your friends.

There is very little chance of the same number being assigned to two different letters because available numbers range from zero to 999.

When typing this program into your Atari , be sure to separate the alphabet letters with commas in line 100.

By the way, note the nice two-column screen printing format! Line 250 does that.

Program Listing

```
 10 PRINT "}":REM CLEAR SCREEN
 20 DIM L$(1),J$(1)
100 DATA A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,
    P,Q,R,S,T,U,V,W,X,Y,Z
200 FOR N=1 TO 13
210 C=INT(1000*(RND(1)))
220 READ L$
230 D=INT(1000*(RND(1)))
240 READ J$
250 PRINT L$;"  ";C,J$;"  ";D
260 NEXT N
```

Sample Run

```
A 981        B 203
C 561        D 491
E 662        F  61
G 536        H  92
I 421        J 768
K 257        L 678
M 662        N 249
```

```
O 486        P 866
Q 115        R 601
S 263        T 322
U 654        V 962
W 821        X 292
Y 524        Z 623
```

6 Sound Off

You can make your ATARI computer beep (or buzz, depending upon how you hear it) on command.

Line 10 clears the screen. The FOR/NEXT loop in lines 20 to 50 make the ATARI sound its built-in beeper 10 times. You can change the number of times the sound is made by changing the number 10 at the end of line 20. Line 30 actually prints the word *buzz* on the screen. Line 40 makes the buzzer (or beeper, if you like) sound off.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 FOR L=1 TO 10
30 PRINT "BUZZ"
40 PRINT "}":REM BUZZER
50 NEXT L
```

7 Ring The Bell I

This is the first of two programs designed to allow you to force sound from your ATARI computer. Here, we use the ASCII number 253 in the PRINT CHR$(*ASCII number*) statement in line 30.

Line 10 clears the text screen. The FOR/NEXT loop in lines 20 to 50 causes the buzzer (or bell or beeper, if you like) to sound. The number of times it sounds is controlled by the number five at the end of line 20. You may change that.

Line 40 is a time-delay loop, placed here to slow down the repetition of the buzzer. The length of time in the delay is controlled by the number 100 in line 40. Use a number larger than 100 for more delay. A number smaller than 100 will reduce the delay. Remember this time-delay technique. You might like to use it in another program.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 FOR N=1 TO 5
30 PRINT CHR$(253)
40 FOR T=1 TO 100:NEXT T
50 NEXT N
```

8 Ring the Bell II

Here's another, slightly different version of our bell ringer program. This one allows you to set a time delay between rings.

The N value in line 20 sets the actual number of repeat rings.

The L value in line 40 establishes the time delay between rings. To insert more time, increase the number 100 in line 40. To shorten the time between beeps, lower the number 100 in line 40.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 FOR N=1 TO 5
30 PRINT "}":REM BUZZER
40 FOR L=1 TO 100:NEXT L
50 NEXT N
```

9 Mystery Clues

Want to create your own murder mystery? Figure out whodunit and write your program backwards from there. When your players make wrong guesses, give them tantalizing clues.

Here's a short program which you can load into your computer in a matter of minutes. Key it in and try it out. It shows how you can add clues to your mysteries.

For simplicity, we assume here the Butler did it. Note that, in line 20, we are making him equal to X$. At line 30, the computer stops to ask you whom you think did it. Your answer is recorded in A$.

In line 40, your answer, lodged in A$, is compared with the computer's already-certain knowledge that the Butler did it. A$ is compared with X$. If they agree, and only if they agree, the computer displays the message, "You guessed it." If you got it right, things will end right there.

If, however, you missed it, program execution (sorry about using that word in a murder mystery!) drops to line 50 where we hear the computer, "Clue: servant." After deftly dropping that clue, the computer moves back and runs through the whole affair another time. It will keep running through it until you answer, "Butler," in response to its question in line 30.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
15 DIM A$(6),X$(6)
20 X$="BUTLER"
30 PRINT "WHODUNIT ";:INPUT A$
40 IF X$=A$ THEN PRINT :PRINT "YES,
   YOU GUESSED IT":PRINT :GOTO 20
50 PRINT :PRINT "CLUE: ","SERVANT":PRINT
   :PRINT :GOTO 20
```

10 Original Hi/Lo Game

Here it is. Where everybody started in microcomputer programming back in the Seventies. The first game ever played was a high-low guess-the-number routine.

The computer selects a secret number. You try to guess it. The computer tells you whether or not you are too high, too low, or right on the number.

Here's how it works: the secret number can be zero to 1000. Line 100 generates a random number (the secret number) and stores it. Line 210 asks you to guess the number.

Lines 300-310 decide if you are right or wrong. Line 220 keeps track of the number of attempts.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 T=0
100 R=INT(1000*(RND(1)))
200 PRINT "}":REM BUZZER
210 PRINT "GUESS THE NUMBER":INPUT B
220 T=T+1
230 PRINT :PRINT "THAT WAS TRY NO. ";T
300 IF B>R THEN PRINT :PRINT "TOO HIGH
    GUESS AGAIN":INPUT B:GOTO 220
310 IF B<R THEN PRINT :PRINT "TOO LOW
    GUESS AGAIN":INPUT B:GOTO 220
400 PRINT "}":REM BUZZER
410 PRINT "YES !   YOU GOT IT"
420 PRINT R;" IS THE NUMBER"
430 PRINT "YOU GOT IT IN ";T;" TRIES"
500 PRINT :PRINT :PRINT
510 GOTO 20
```

11 Code Groups

Need some secret codes for your latest sensitive mission? How about sets of five random letters for use in Morse code practice?

This program has the computer generate an endless string of random combinations of five letters.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 FOR L=1 TO 5
30 N=INT(91*(RND(1)))
40 IF N<65 THEN 30
50 PRINT CHR$(N);
60 NEXT L
70 PRINT :GOTO 20
```

Sample Run

```
CYGQH     |   MWHOJ     |   KAFDH
XMIAJ     |   CTWRQ     |   BTPNC
BZPDO     |   VFEZK     |   QDFGN
REMSF     |   HALVN     |   ANBWO
NTVEA     |   NXECR     |   RUNSX
IDPLG     |   PODAA     |   TPKII
```

12 60-Second Timer

A one-minute timer can be very handy for fun-n-games. This easy-to-use clock "ticks" as it counts off seconds up to 60. When it reaches 60 seconds, it rings an alarm.

The number of seconds counted can be changed by changing the number 60 in line 20.

The clock can be calibrated by changing the number 200 in line 50. Line 50 is a time-delay loop set for approximately one second.

Lines 70-90 provide a rapid burst of five beeps when the clock reaches 60 seconds. To change the length of this alarm, change the number 5 in line 70.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 FOR T=1 TO 60
30 PRINT "}":REM BUZZER
40 PRINT T,"SECONDS"
50 FOR L=1 TO 200:NEXT L
60 NEXT T
70 FOR E=1 TO 5
80 PRINT "}":REM BUZZER
90 NEXT E
```

13 Find Highest/Lowest

Suppose we have a list of people and each person has been assigned a number or score. This program accepts the names and scores and sorts out the persons with the highest and lowest scores. Here's how it works.

At line 20, the memory is DIMensioned to hold data. Lines 30-110 take in the info on each person. As each person's score is entered, lines 70-100 determine if it is higher or lower than all previous scores. If higher or lower, it is so noted.

To complete data entry, simply press RETURN without data. That will prompt the computer, at lines 130 and 140, to print the lowest score and the highest score.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 DIM N$(20),LN$(20),HN$(20),K$(1)
30 PRINT "}":REM BUZZER
40 PRINT "NAME:",:INPUT N$
50 IF N$="" THEN 120
60 PRINT "SCORE:",:INPUT S
70 X=X+1
80 IF X=1 THEN LS=S:LN$=N$:HS=S:HN$=N$
90 IF S<LS THEN LS=S:LN$=N$
100 IF S>HS THEN HS=S:HN$=N$
110 GOTO 40
120 PRINT :PRINT
130 PRINT LN$;" SCORED LOWEST AT ";LS
140 PRINT HN$;" SCORED HIGHEST AT ";HS
150 PRINT :PRINT :PRINT :PRINT
160 PRINT "TO DO MORE, PRESS RETURN"
170 INPUT K$
180 CLR :GOTO 10
```

14 Sorting Scores

Here's how to sort a set of scores. Any numbers can be used. Zero is assumed to be lower than any positive number and a negative number is lower than zero.

Key in as many numbers as you like. Then key a zero when you want your Atari to compute final results. Obviously, a zero cannot be in the set of numbers you are sorting since we use zero to get out of the input loop.

At the end of the RUN, the computer will tell you which number is lowest and which is highest.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 DIM K$(1)
30 PRINT "TYPE IN A SCORE: ";:INPUT N
40 IF N=0 THEN 100
50 S=S+1
60 IF S=1 THEN LN=N:HN=N
70 IF N<LN THEN LN=N
80 IF N>HN THEN HN=N
90 GOTO 30
100 PRINT
110 PRINT "LOW NUMBER IS ";LN
120 PRINT "HIGH NUMBER IS ";HN
130 PRINT :PRINT :PRINT
140 PRINT "FOR A DIFFERENT"
150 PRINT "SET OF NUMBERS,"
160 PRINT "PRESS RETURN"
170 INPUT K$
180 CLR :GOTO 10
```

Sample Run

```
RUN       RETURN

TYPE IN A SCORE
22        RETURN
55        RETURN
```

```
77        RETURN
11        RETURN
 Ø        RETURN

LOW NUMBER IS 11
HIGH NUMBER IS 77
```

15 Keeping Game Scores

Writing a computer football game? Spelling bee? Cave adventure? No matter what kind of fun you are preparing, you'll need a way to keep score. Here's how.

The wealthy English duke has just been killed in our little mystery game. In lines 10 through 160 of our program listing, below, you play the game, attempting to find out whodunit.

The trick here is in the scorekeeping. Note line 170. If you guessed correctly in response to the query in line 160, at line 170 the computer will give you credit by adding one point to your score stored in memory location R. It does that by comparing your line 160 answer stored in P$ with the correct answer stored in A$.

If you blew it and guessed wrong, the program drops below line 170 to line 180 where it increases your "wrong score" by adding one point to W.

If you got a W+1 at line 180, the program moves back to line 100 and gets you to try again. If you scored a victory and got an R+1 at line 170, the program jumps to line 200 where it stops to display your total right and wrong score. After that, it's back to line 40 for a complete new run-through.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 DIM A$(7),B$(7),C$(7),D$(7)
30 DIM E$(7),F$(7),G$(7),P$(7)
40 S=INT(7*(RND(1)))
50 IF S<1 THEN 40
```

```
60 FOR L=1 TO S
70 READ A$
80 NEXT L
90 RESTORE :PRINT "*********************"
100 PRINT "WHO KILLED THE DUKE ?"
110 READ B$,C$,D$,E$,F$,G$
120 RESTORE
130 PRINT :PRINT "WAS IT THE..."
140 PRINT B$,C$,D$,E$,F$,G$
150 PRINT :PRINT "}":REM BUZZER
160 PRINT "WHODUNIT ";:INPUT P$
170 IF A$=P$ THEN R=R+1:PRINT
    :PRINT "OK  YOU ARE RIGHT"
    :PRINT "IT WAS THE ";A$:GOTO 200
180 PRINT :PRINT "NO! NOT THE ";P$:W=W+1
    :PRINT :GOTO 100
200 PRINT
210 PRINT "YOUR SCORE IS..."
220 PRINT R;" RIGHT ",W;"WRONG"
230 PRINT :PRINT :GOTO 40
300 DATA BUTLER,NANNY,GARDNER,BURGLAR
    ,SON,WIFE
```

16 Batting Average

Once you know the number of times you were right and wrong in a game, as in Tip Number 15, it's fun to convert those raw numbers to a batting average. Numbers right and numbers wrong take on a new meaning when changed to a batting average. Folks seem to be able to understand a batting average better.

Our program, starting at line 900, is a partial listing designed to be tacked onto the end of your longer game program to display the final results of play. It will show the number of tries, number of right answers, percentage right, and batting average.

You'll want to test load this program so add lines 10 and 800 as shown. Line 800 will give you the R and T values you'll need going into the program at line 900.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
800 PRINT "NUMBER RIGHT: ";
810 INPUT R
820 PRINT "NUMBER TRIES: ";
830 INPUT T
900 PRINT R;" RIGHT"
910 PRINT "IN ";T;" TRIES"
920 D=R/T:P=100*D:B=INT(10*P)
930 PRINT "THAT'S ";P;" PERCENT"
940 PRINT "YOU ARE BATTING ";B
```

17 Computer Rating Service

Of course, once you know a player's batting average it still might need some interpretation. In this program, the computer takes a look at a batting average and makes a comment.

Remember that this listing, starting here with line 800, is a partial program to be tacked on the end of a longer game. Note that, at 800, you already have values for G (number right) and E (number of tries). Line 810 converts those raw numbers to a batting average (H).

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
700 PRINT "NUMBER RIGHT: ";
710 INPUT G
720 PRINT "NUMBER OF TRIES: ";
730 INPUT E
800 PRINT G;" RIGHT IN ";E;" TRIES"
810 H=INT((G/E)*1000)
820 PRINT "YOU ARE BATTING ";H
830 PRINT "YOU ARE ";
840 IF H<100 THEN GOTO 910
850 IF H<300 THEN GOTO 920
860 IF H<500 THEN GOTO 930
```

```
870 IF H<700 THEN GOTO 940
880 IF H<900 THEN GOTO 950
900 PRINT "HALL OF FAME":GOTO 960
910 PRINT "THE PITS":GOTO 960
920 PRINT "POOR":GOTO 960
930 PRINT "AVERAGE":GOTO 960
940 PRINT "TOP NOTCH":GOTO 960
950 PRINT "DAMN NEAR PERFECT!"
960 PRINT "YOUR BATTING AVERAGE IS ";H
970 END
```

18 Box Score

To dress up scores during and at the end of a game program, use this method of putting those scores in a box. The box around the score will highlight it and jazz up your video display.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 DIM N$(20),S$(8)
30 PRINT "PLAYER'S NAME: ";
40 INPUT N$
50 PRINT "PLAYER'S SCORE: ";
60 INPUT S
70 PRINT
80 S$=STR$(S)
90 LN=LEN(N$)+LEN(S$)
100 LT=LN+14
110 FOR L=1 TO LT
120 PRINT "*";
130 NEXT L
140 PRINT
150 PRINT "* ";N$;"'S SCORE: ";S$;" *"
160 FOR L=1 TO LT
170 PRINT "*";
180 NEXT L
190 PRINT :PRINT :GOTO 30
```

Sample Run

```
*************************
* EDWARD'S SCORE: 1234 *
*************************
```

Text On Text

19 Create a Quiz

One of the most fascinating uses for your Atari is in having it carry on a video conversation with your friends, relatives and neighbors. One useful way to promote such conversation is through a quiz. An instructional, educational quiz, such as we have here.

Quiz data—the computer's storehouse of knowledge—is in lines 20 to 70. Be careful, when you type them into your Atari, to include the commas separating the two halfs of each data line. Spelling and spacing must be exact.

Lines 90 and 100 obtain a random number in the range of 1 to 11. Line 110 selects the data line for a question. Lines 120 through 140 get the appropriate word FIRST, SECOND, THIRD, FOURTH, FIFTH or SIXTH from the selected data line. Lines 160 to 180 print the quiz question on the screen, while line 190 reads the DATA line to learn the correct answer. You provide your response when the computer asks for it at line 200. Lines 220-240 decide whether you are right or wrong.

Of course, the quiz can be made much longer. In this example, it could be expanded to encompass all past U.S. presidents.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
15 DIM S$(9),C$(20),D$(20)
20 DATA FIRST,GEORGE WASHINGTON
30 DATA SECOND,JOHN ADAMS
40 DATA THIRD,THOMAS JEFFERSON
50 DATA FOURTH,JAMES MADISON
60 DATA FIFTH,JAMES MONROE
70 DATA SIXTH,JOHN QUINCY ADAMS
80 PRINT "HOW MANY  U.S. PRESIDENTS"
85 PRINT "CAN YOU NAME"
90 R=INT(12*(RND(1)))
100 IF R<1 THEN 90
110 IF INT(R/2)=R/2 THEN R=R+1
```

```
120 FOR L=1 TO R
130 READ S$
140 NEXT L
150 PRINT :PRINT
160 PRINT "WHO WAS THE"
170 PRINT S$
180 PRINT "PRESIDENT OF THE U.S."
190 READ C$
200 INPUT D$
210 PRINT
220 IF D$=C$ THEN PRINT "THAT'S CORRECT  "
    :GOTO 240
230 PRINT "THAT'S WRONG"
240 PRINT "THE ";S$;" PRESIDENT WAS  "
250 PRINT C$
260 RESTORE
270 PRINT :PRINT
280 GOTO 90
```

20 Killing Time

Sometimes, it may seem to you as if the computer will never get to the result of a job. You understand the processing delay but your non-computer friends may not. They could be confused by the wait and think the computer is "broken."

To keep their minds off the slowness, give them something to look at while the computer is "thinking."

The added, extra lines, numbered 50, 60, 70 and 80, take up more processing time but make for less confusion. Computing may take a bit longer but your fun will be increased.

If you delete lines 50-80 you'll see how the program runs faster but the blank screen is confusing.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "GIVE ME A NUMBER ";:INPUT N
```

```
30 FOR L=1 TO N
40 X=X+L
50 FOR T=1 TO 100:NEXT T
60 PRINT "}":REM SCREEEN CLEAR
70 FOR T=1 TO 100:NEXT T
80 PRINT "I AM THINKING"
90 NEXT L
100 PRINT "}":REM SCREEN CLEAR
110 PRINT "I HAVE THE ANSWER":PRINT
120 PRINT "THE TOTAL OF ALL NUMBERS"
130 PRINT "FROM 1 TO ";N;" IS ";X
200 PRINT :PRINT :PRINT
210 CLR :GOTO 20
```

21 Word-Error Trapping

The same kind of error trapping is available for strings. Suppose the program, as in this example, asks at line 10 for a word. It is looking for YES or NO. If it gets a YES, then line 20 sees that it got what it wanted and moves operations along to line 100.

If it gets a NO, then line 20 hasn't received what it wants so program execution moves on to line 30. Here, at line 30, the program finds something useful and shoots operations down to line 200.

If, however, neither YES nor NO were entered at line 10, then neither lines 20 nor 30 would be satisfied so action would drop to line 40. Here, the error is trapped by commanding the operator to give one of the two correct answers. Then, at line 50, the operation is returned to line 10 for a new try at the correct input.

Program Listing

```
5 DIM A$(3):PRINT "}":REM CLEAR SCREEN
10 PRINT "WANT TO PLAY AGAIN ";:INPUT A$
20 IF A$="YES" THEN 100
30 IF A$="NO" THEN 200
40 PRINT "PLEASE ANSWER ONLY YES OR NO"
```

```
50 PRINT :GOTO 10
100 PRINT "THANK YOU FOR THAT ";A$
110 PRINT :PRINT :GOTO 10
200 PRINT "THANK YOU FOR THAT ";A$
210 PRINT :PRINT :GOTO 10
```

22 Character Numbers

This brief program displays the ASCII value for each keyboard character, side-by-side with the character it stands for. You will be able quickly to tell what each number prints.

Line 40 is a timing loop to slow down the presentation so you can digest the information. To make it even slower, increase the number 400 in line 40. To make it faster, decrease the number 400 in line 40.

The computer's chime will ring after it reaches 255. Line 60 provides the beep.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 FOR N=0 TO 255
30 PRINT N,CHR$(N)
40 FOR T=1 TO 400:NEXT T
50 NEXT N
60 PRINT "}":REM BUZZER
```

23 One-Time Password

If you don't want unauthorized use of your programs, insert a requirement that a user know a password. This particular routine allows only one try at entering a correct password.

For our password, we have selected "elephant" and stored it in line 30. You can change the password to whatever you like.

If a correct attempt at entering the password is made, program action will progress to line 100. Otherwise, action drops to line 40 and action ends.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
15 DIM A$(8)
20 PRINT "WHAT IS THE PASSWORD  ";:INPUT  A$
30 IF A$="ELEPHANT" THEN 100
40 END
100 PRINT "YOU GOT IT RIGHT"
110 PRINT "NOW THE PROGRAM WOULD RUN"
```

24 Three-Tries Password

Here the software lets you try three times to enter the correct password. You don't get to go forward with the program if you don't get it right in three tries.

Again the password is "elephant" and is stored in line 30. You can change the password to whatever suits you.

Lines 40 to 60 allow the three attempts. If no good after three tries, then END.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
15 DIM A$(8)
20 PRINT "WHAT IS THE PASSWORD  ";:INPUT A$
30 IF A$="ELEPHANT" THEN 100
40 B=B+1
50 IF B=3 THEN END
60 GOTO 20
100 PRINT "YOU GOT IT RIGHT"
110 PRINT "NOW THE PROGRAM WOULD RUN"
```

25 Multiple Passwords

Here's a really complex password entry system. It has a unique "account number" and a password for each person. This will allow several different persons access to the program but each person will have a different combination to the lock!

account number	**password**
12345	zebra
23456	goose
34567	trout
45678	snake

Each individual user must correctly enter his unique account number and then his own personal password. If account number is wrong, then the password never can be right. If account number is okay but password doesn't match, the user gets no run.

You can add users to this program by adding lines to the 300-340 subroutine.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
15 DIM PS$(5),PW$(5)
20 FOR L=1 TO 3
30 PRINT "YOUR ACCOUNT NUMBER:  ",:INPUT UA
40 GOSUB 300
50 PRINT "PASSWORD:",,:INPUT PS$
55 IF PS$="" THEN 50
60 IF PS$=PW$ THEN 100
70 NEXT L
80 END
100 PRINT "YOU GOT IT ALL RIGHT"
110 PRINT "NOW THE PROGRAM WOULD RUN"
120 END
300 IF UA=12345 THEN PW$="ZEBRA"
310 IF UA=23456 THEN PW$="GOOSE"
```

```
320 IF UA=34567 THEN PW$="TROUT"
330 IF UA=45678 THEN PW$="SNAKE"
340 RETURN
```

26 Name In A Box

Put your name up in lights! Or, at least, on the video display screen of your Atari computer.

This short program creates a box on the screen and puts a name you have specified into that box. The name is highlighted.

You can change what the box is composed of by changing the asterisks in lines 80, 110 and 130.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 DIM N$(26),K$(1)
30 PRINT :PRINT "WHAT IS YOUR NAME"
40 INPUT N$
50 LN=LEN(N$):LT=LN+4
60 PRINT :PRINT "}":REM CLEAR SCREEN
70 FOR L=1 TO LT
80 PRINT "*";
90 NEXT L
100 PRINT
110 PRINT "* ";N$;" *"
120 FOR L=1 TO LT
130 PRINT "*";
140 NEXT L
150 FOR L=1 TO 10
160 PRINT
170 NEXT L
180 CLR :GOTO 20
```

27 Entering: Zero Stop

Here's another way to conclude an entry loop: have the computer be on the lookout for a plain zero. When a zero is entered, the computer will jump out of the entry cycle and on to further action.

This program totals numbers as they are added and accumulates them in memory location B. If one of the numbers entered is a zero alone, then line 110 will spot it and send the computer on down to line 200, breaking the entry cycle.

Naturally, you can't use a zero in a string of numbers to be added since zero causes the computer to quit entering and get on with displaying.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 B=0
100 PRINT "GIVE ME A NUMBER":INPUT A
110 IF A=0 THEN 200
120 B=B+A
130 PRINT :GOTO 100
200 PRINT :PRINT :PRINT
210 PRINT "THE TOTAL OF THOSE NUMBERS
    IS  ";B
300 PRINT :PRINT :PRINT :PRINT
310 GOTO 20
```

28 Entering: Letter Stop

One way to conclude an input series, and get out of its entry loop, is to use a key letter to promote a jump. In this brief example, we input numbers, at line 100, as string values. If we give the computer an X rather than a number, it will jump down to line 200 for new action.

Numbers keyed in are stored first as strings. Then line 120 changes them to number values for the addition in line 130.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 CLR
30 DIM A$(20)
100 PRINT "GIVE ME A NUMBER:",:INPUT A$
110 IF A$="X" THEN 200
120 B=VAL(A$)
130 C=C+B
140 GOTO 100
200 PRINT :PRINT :PRINT
210 PRINT "THE TOTAL OF THOSE NUMBERS
    IS ";C
300 PRINT :PRINT :PRINT :PRINT
310 GOTO 20
```

29 Title Billboard

This program opens a light window against a dark background. It can be placed anywhere on the graphics screen.

Move the window by changing the range of values for X and for Y in lines 1030 and 1040.

Move the lettering around by changing the PLOT and DRAWTO locations specified in lines 1090 to 1290.

Line 1300 is a freeze-frame, used to hold the picture. It is an endless loop which will remain forever at line 1300 until you press the BREAK key. Remember how to make this freeze-frame. You might like to use it in some program in the future.

Program Listing

```
1000 GRAPHICS 10
1020 COLOR 1
1030 FOR X=15 TO 75
1040 FOR Y=10 TO 45
1050 PLOT X,Y
1060 NEXT Y
1070 NEXT X
```

```
1080 COLOR 8
1090 PLOT 20,20
1100 DRAWTO 30,20
1110 DRAWTO 25,20
1120 DRAWTO 25,30
1130 PLOT 35,20
1140 DRAWTO 35,30
1150 PLOT 40,20
1160 DRAWTO 50,20
1170 DRAWTO 45,20
1180 DRAWTO 45,30
1190 PLOT 55,20
1200 DRAWTO 55,30
1210 DRAWTO 60,30
1220 PLOT 65,20
1230 DRAWTO 70,20
1240 DRAWTO 65,20
1250 DRAWTO 65,25
1260 DRAWTO 70,25
1270 DRAWTO 65,25
1280 DRAWTO 65,30
1290 DRAWTO 70,30
1300 GOTO 1300
```

30 Marching Numbers

This little program does a big job! It creates the unusual display of numbers from one to nine marching across the screen. Try it; you'll like it.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 X=0
30 X=X+1
40 IF X>9 THEN 20
50 PRINT X;
60 GOTO 30
```

31 Wipeout!

Warning: handle with care!

Careless operation of this program can cause you a lot of extra work.

Key in the program. Run it. When it asks for the password, be sure to give it what it wants or it will erase itself. That's right, the entire contents of program memory down the tubes!

Here's how it works:

The password, in this case Tracey , is asked by line 20. You give it a password. In line 30, your answer is compared to the true password. If correct, action goes to line 50. If incorrect, the program goes to line 40.

The NEW statement in line 40 erases everything from program memory.

You can change the password in line 30 to any letters, numbers or keyboard symbols of your choice. Watch out when testing. A wrong password can cause a lot of retyping.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
15 DIM P$(20)
20 PRINT "WHAT IS THE PASSWORD: ";
   :INPUT P$
30 IF P$="TRACEY" THEN 50
40 NEW
50 PRINT "CORRECT"
60 PRINT P$;" IS THE PASSWORD"
```

32 Sentence Writer

Practice your English!

Exhibit your knowledge of nouns and verbs. This program leads the computer to solicit individual words from you and use those words to create sentences.

Besides helping you better understand verbs, nouns and simple declarative sentence structures, the program demonstrates the computer's ability to simulate conversation and communication.

Lines 20, 30 and 40 take in the words.

You may modify the program to suit your own interests or needs.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
15 DIM N$(15),V$(15),S$(15),K$(1)
20 PRINT "A PLURAL NOUN:",:INPUT N$
30 PRINT "A VERB:",,:INPUT V$
40 PRINT "A SINGULAR NOUN:",:INPUT S$
50 PRINT :PRINT
60 PRINT "THE ";N$;" ";V$;" ";S$;"."
70 FOR L=1 TO 10:PRINT :NEXT L
80 PRINT "PRESS RETURN"
90 INPUT K$
100 CLR :GOTO 10
```

Sample Run

```
A PLURAL NOUN = ?
DOGS
A VERB = ?
LOVE
A SINGULAR NOUN = ?
FOOD

THE DOGS LOVE FOOD.

A PLURAL NOUN = ?
BOXES
A VERB = ?
HOLD
A SINGULAR NOUN = ?
WATER

THE BOXES HOLD WATER
```

33 Categorizing

A large quantity of numbers can be categorized and thereby cut down into a smaller quantity of numbers. See our example: it takes test scores and divides them into ranges labeled A, B, C, D, and F.

The program assumes exam or test scores in a range of zero to 100. The letter grades include zero to 59, F; 60-69, D; 71-79, C; 80-89, B; 90-100, A.

Key in as many scores as you like and then enter the letter X to stop the entry cycle.

Lines 100-140 sort all scores into the A through F categories. Lines 150-170 sort highest and lowest scores. Line 200 finds mid-range and average scores.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 DIM G$(10),K$(1)
30 PRINT "ENTER A GROUP OF SCORES"
40 PRINT "FROM 0 TO 100, ONE AT A TIME"
50 PRINT "ENTER X AFTER LAST SCORE"
60 PRINT :PRINT "SCORE:  ";:INPUT G$
70 IF G$="X" THEN 200
80 G=VAL(G$)
90 N=N+1
100 IF G<60 THEN F=F+1:GOTO 150
110 IF G<70 THEN D=D+1:GOTO 150
120 IF G<80 THEN C=C+1:GOTO 150
130 IF G<90 THEN B=B+1:GOTO 150
140 A=A+1
150 IF N=1 THEN L=G:H=G
160 IF G<L THEN L=G
170 IF G>H THEN H=G
180 S=S+G
190 GOTO 60
200 P=S/N:M=L+((H-L)/2)
210 PRINT "}":REM SCREEN CLEAR
220 PRINT "THERE WERE ";N;" SCORES"
230 PRINT "RANGING FROM ";L;" TO ";H
```

```
240 PRINT :PRINT "MID-RANGE SCORE:",M
250 PRINT "AVERAGE SCORE:",P
260 PRINT :PRINT "TOTALS FOR EACH LETTER
    GRADE:"
270 PRINT :PRINT "A:",A
280 PRINT "B:",B
290 PRINT "C:",C
300 PRINT "D:",D
310 PRINT "F:",F
400 PRINT :PRINT :PRINT
410 PRINT "FOR MORE, PRESS RETURN"
420 INPUT K$
430 CLR :GOTO 10
```

34 Alphabet Soup

Sure, everybody knows there are 26 letters in the alphabet. But, do you know which letter is number 20? Number 5? Number 17? Well, your Computer knows!

Type in this short ready-to-run program. RUN it. The computer will spit out number-and-letter combinations all day long. The number on the left is the position in the alphabet of the letter on the right.

It's a fun way to demonstrate to your friends just how "smart" the computer is!

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 DIM X$(1)
30 P=INT(91*(RND(1)))
40 IF P<65 THEN 30
50 X$=CHR$(P)
60 PRINT P-64;":  ",X$
70 CLR
80 GOTO 20
```

Sample Run

```
                                          19: S
21: U          17: Q                      18: R
 3: C          11: K                      14: N
25: Y           8: H                      20: T
 6: F          10: J                       5: E
 2: B           7: G                      19: S
```

35 Create A Table

This program generates a table of values, as a demonstration on how to set up a table on the video display.

Subroutine lines 900 and 910 generate random numbers in the range of zero to 99. Lines 20 and 30 find how many times through the random number generator it takes to get a number greater than 50. The answer is stored in A.

Lines 40 and 50 do it again and store the answer in B. Lines 60 and 70 do it and store in C.

Line 10 prints the table heading and line 100 displays the results. Line 110 causes the whole operation to repeat until you have a table of 20 lines on the screen.

Program Listing

```
5 PRINT "}":REM SCREEN CLEAR
10 PRINT "A","B","C":PRINT
20 GOSUB 900
30 IF X>50 THEN A=A+1:GOTO 20
40 GOSUB 900
50 IF X>50 THEN B=B+1:GOTO 20
60 GOSUB 900
70 IF X>50 THEN C=C+1:GOTO 20
100 PRINT A,B,C:IF T=19 THEN 200
110 A=0:B=0:C=0:T=T+1:GOTO 20
200 GOTO 200
900 X=INT(100*(RND(1)))
910 RETURN
```

36 Question & Answer

Here's how to use the DATA statement, and the computer's ability to search for data, to create a Q&A.

We put DATA in lines 20-130. It could be anywhere in the program. For instance, at the end at lines 400-510.

The computer sees two items in each data line. Program lines 140 and 160 force the machine to take only odd-numbered data from the list. That is, S$ in line 180 is always the first piece of data in a data line. And C$ in line 210 is always the second item in a data line. Line 230 checks to see if you answered the line 220 question correctly.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
15 DIM S$(9),C$(2),D$(2)
20 DATA JANUARY,31
30 DATA FEBRUARY,28
40 DATA MARCH,31
50 DATA APRIL,30
60 DATA MAY,31
70 DATA JUNE,30
80 DATA JULY,31
90 DATA AUGUST,31
100 DATA SEPTEMBER,30
110 DATA OCTOBER,31
120 DATA NOVEMBER,30
130 DATA DECEMBER,31
140 R=INT(25*(RND(1)))
150 IF R<1 THEN 140
160 IF INT(R/2)=R/2 THEN R=R-1
170 FOR L=1 TO R
180 READ S$
190 NEXT L
200 PRINT "MONTH IS ";S$
210 READ C$
220 PRINT "HOW MANY DAYS ";:INPUT D$
230 IF D$=C$ THEN PRINT :PRINT "CORRECT"
    :GOTO 300
```

```
240 PRINT :PRINT "WRONG"
300 PRINT "NUMBER OF DAYS IS ";C$
310 RESTORE
320 FOR L=1 TO 5:PRINT :NEXT L
330 CLR :GOTO 15
```

Gee Whiz

37 Gee Whiz I: Smart Adder

These six programs, in this section of the book, make up our *Gee Whiz* series. One of the fun ways to use your Atari is in wowing your friends. Next time they ask, "But, what can it do?", show them its uncanny abilities at adding, spelling, writing upside down, even cracking jokes. Try these six *Gee Whiz* programs on your friends. You'll love their reactions.

Smart Adder is the first in the series. When your neighbor drops in for a cup of coffee, bring out the Atari for a demonstration of its lightning speed.

This program adds long strings of numbers in a flash. You give the computer a number. It starts at 1 and adds all numbers up to and including your number. For instance, if you give it a five, it will add 1 plus 2 plus 3 plus 4 plus 5 and display the result.

Ask your neighbor how fast he or she can add all the numbers to 100. It should take several minutes. While he's working on it, let your Atari do it in a split second. Your neighbor's reaction is bound to be, "Gee whiz!"

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 PRINT "GIVE ME A NUMBER:   ";:INPUT N
30 IF N<1 THEN 20
40 FOR L=1 TO N:X=X+L:NEXT L
50 PRINT "}":REM BUZZER
60 PRINT "THE TOTAL OF 1 TO ";N;" IS ";X
70 PRINT :PRINT :CLR :GOTO 20
```

38 Gee Whiz II: Three-Digit Mystery

Have your neighbor secretly select any three-digit number in which all three digits are the same. Then have

him tell the computer only the *sum* of those three digits.
The computer will identify his secret number!

Program Listing

```
10  PRINT "}":REM SCREEN CLEAR
20  PRINT "SELECT A THREE-DIGIT NUMBER"
30  PRINT "ALL THREE DIGITS THE SAME"
40  PRINT
50  PRINT "ADD THE THREE DIGITS TOGETHER"
60  PRINT
70  PRINT "WHAT IS THE SUM"
80  PRINT "OF THE THREE DIGITS    ";
90  INPUT N
100  IF N<3 OR N>27 THEN 90
110  Q=37*N
120  PRINT :PRINT
130  PRINT "}":REM BUZZER
140  PRINT "YOUR NUMBER IS ";Q
150  PRINT :PRINT :GOTO 20
```

39 Gee Whiz III: Yes/No Decision Maker

This is handy for the busy executive who doesn't have time for decisions.

Line 10 clears the screen. Line 20 generates a random number from zero to 100. Line 30 selects a *yes* answer if the random number is greater than 49. Otherwise, line 40 chooses a *no* answer.

Program Listing

```
10  PRINT "}":REM SCREEN CLEAR
20  X=100*(RND(1))
30  IF X>49 THEN PRINT "YES":GOTO 50
40  PRINT "NO"
50  END
```

40 Gee Whiz IV: First Alphabet Spotter

There are 26 letters in the alphabet. Each has a number. For instance, number 1 is A. Number 20 is T. This *Gee Whiz* program has the computer ask you for a number from 1 to 26 and then, faster than a jackrabbit, tell you what letter it goes with.

Naturally, you'll know how it works but to your non-computer friends it will seem like the Atari is a genius!

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "GIVE ME THE NUMBER OF"
30 PRINT "A LETTER FROM THE ALPHABET"
40 PRINT "FROM 1 TO 26"
50 INPUT N
60 X=N+64
70 PRINT :PRINT :PRINT
80 PRINT "LETTER NUMBER ";N;" IS ";CHR$(X)
90 PRINT :PRINT :PRINT
100 GOTO 20
```

41 Gee Whiz V: Second Alphabet Spotter

This is a variation on the previous program. This *Gee Whiz* program has the computer ask you for a number from 1 to 26 and then, faster than a jackrabbit, tell you what letter it goes with.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 DIM A$(1)
```

```
30 DATA A,B,C,D,E,F,G,H,I,J,K,L,M
40 DATA N,O,P,Q,R,S,T,U,V,W,X,Y,Z
50 PRINT "GIVE ME THE NUMBER OF"
60 PRINT "A LETTER FROM THE ALPHABET"
70 PRINT "FROM 1 TO 26"
80 INPUT N
90 FOR L=1 TO N
100 READ A$
110 NEXT L
120 PRINT :PRINT
130 PRINT "}":REM BUZZER
140 PRINT "LETTER NUMBER ";N;" IS ";A$
150 RESTORE
160 PRINT :PRINT
170 GOTO 50
```

42 Gee Whiz VI: Guess The Number

Here it is! The world's oldest, longest running, most popular game: Guess The Number.

When you start the program running, the computer thinks of a number and stores that away. You try to guess the number. If your number is too high, the computer says, "TOO HIGH."

If you are too low, the computer will report "TOO LOW." The possible numbers range from zero to 100.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 DIM Q$(1)
30 Q$="*"
40 GOTO 110
50 N=INT(101*(RND(1)))
60 PRINT "GUESS THE NUMBER",:INPUT G
70 IF G>N THEN PRINT "TOO HIGH":GOTO 60
80 IF G<N THEN PRINT "TOO LOW":GOTO 60
90 PRINT "RIGHT !"
100.PRINT :PRINT "LET'S GO AGAIN"
```

```
110 FOR L=1 TO 38
120 PRINT Q$;
130 NEXT L
140 GOTO 50
```

Sample Run

```
********************
GUESS THE NUMBER ?
46
TOO HIGH
GUESS THE NUMBER ?
21
TOO LOW
GUESS THE NUMBER ?
25
RIGHT !
LETS GO AGAIN
********************
```

Number Crunching

43 Memory Tester

Most everybody can remember numbers. At least short numbers with few digits. But how long a number can you recall in a flash?

The computer will briefly display a number. It then will remove the number from your view and ask you to repeat what it was. If you miss three times, the computer will tell you to FORGET IT, give you your score and end the game. Then it will start over.

On the other hand, if you recall correctly, the computer will say so and then give you a new number. The new number will have more digits than the previous number. Each time you guess correctly, the number gets longer.

No matter how good you are, at some point you won't be able to recall *all* the digits in proper sequence.

How many digits can you quickly recall?

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 Z=1
30 S=10*RND(1)
40 N=INT(S*Z)
50 PRINT "REMEMBER---->",N
60 IF W=3 THEN PRINT " FORGET IT !"
   :GOTO 140
70 FOR T=1 TO 500:NEXT T
80 PRINT "}":REM CLEAR SCREEN
90 PRINT "WHAT WAS IT",:INPUT S
100 IF S<>N THEN PRINT "***YOU ARE WRONG !"
    :W=W+1:GOTO 60
110 PRINT "YOU ARE RIGHT !":R=R+1:Z=Z*10
120 PRINT R;" RIGHT SO FAR"
130 PRINT :GOTO 30
140 PRINT :PRINT "YOU HAD ";R;" RIGHT"
150 PRINT :PRINT "LET'S START OVER"
160 PRINT "PRESS RETURN"
170 DIM K$(1)
180 INPUT K$
190 CLR :GOTO 10
```

44 Number Reverser

Give your Atari any three-digit number and, as a result of this particular programming trick, it will reverse the original number. For example, 789 will be transformed into 987. Or 123 into 321. It takes your three-digit number apart and reassembles it in reverse order.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "ENTER A THREE-DIGIT NUMBER:  "
30 INPUT N
40 IF N<100 OR N>999 THEN 20
50 A=INT(N/100)
60 B=INT(10*((N/100)-A))
70 C=INT(10*((N/10)-(INT(N/10))))
80 PRINT C;B;A
90 PRINT :PRINT
100 CLR :GOTO 20
```

45 Exam Score Sorter

A quick way to sort and count a book full of letter grades, this program permits one-key entry of a mixed series of data.

We use the familiar letter grades A, B, C, D, and F. You may substitute any other set of five characters you wish in the IF statements in lines 80 to 120.

The letter X is used to conclude the series and lead the computer to display final results.

You press the appropriate key and the computer knows immediately what grade you have indicated. Key in as many grades as you like in any mixed order.

When you have completed entering all grades, type in the letter X. The computer will report the total of A's, B's, C's, D's, and F's.

We use exam-score sorting as our example here but this same program would be good for data collection in the field in many professions. And you can stretch out the possible categories to 25 or more.

To make the program run again, press the RETURN key on the computer's keyboard.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 DIM G$(1),K$(1)
30 PRINT "ENTER LETTER GRADES:"
40 INPUT G$
50 IF G$="X" THEN 140
80 IF G$="A" THEN A=A+1:GOTO 40
90 IF G$="B" THEN B=B+1:GOTO 40
100 IF G$="C" THEN C=C+1:GOTO 40
110 IF G$="D" THEN D=D+1:GOTO 40
120 IF G$="F" THEN F=F+1:GOTO 40
130 GOTO 40
140 PRINT "}":REM SCREEN CLEAR
150 PRINT "A:",A
160 PRINT "B:",B
170 PRINT "C:",C
180 PRINT "D:",D
190 PRINT "F:",F
200 PRINT :PRINT :PRINT
210 PRINT "FOR MORE, PRESS RETURN"
220 INPUT K$
230 CLR :GOTO 10
```

Sample Run

```
ENTER LETTER GRADES
A
C
F
D
B
A
B
C
C
C
D
```

```
F

A: 2
B: 2
C: 4
D: 2
F: 2
```

46 Number-Error Trapping

Good programs, those which are well written, need *error trapping*. It's a technique for making sure persons communicating with the computer don't key in inappropriate data or make mistakes which would cause computation problems for the computer.

For instance, see the example program here. In line 10 the computer asks for a number. In line 20, if the number is too low, it says so and goes back to line 10 to repeat its request.

At line 30, if the number received at line 10 is too large, it says so and goes back to line 10 for a better choice.

The result is only printed at line 40 when a satisfactory number has been keyed in back at line 10.

You can set your own limits by changing the 10 in line 20 and the 100 in line 30.

Program Listing

```
5 PRINT "}":REM CLEAR SCREEN
10 PRINT "GIVE ME A NUMBER ";:INPUT A
20 IF A<10 THEN PRINT "TOO LOW":GOTO 10
30 IF A>100 THEN PRINT " TOO  HIGH"
   :GOTO  10
40 PRINT A
```

47 Standard Deviation

Here's a way to determine mean and standard deviation. In this particular program, you exit the entry cycle by entering the large number 999999999 (nine 9's) so you can't use 999999999 as one of your data points.

This is a great opportunity to experiment with standard deviation computations. Try a series of data points such as 3, 5, 3, 7, and 4. They should result in

```
DATA POINTS TOTAL:   22
MEAN:                4.4
VARIANCE:            2.2399989
STD DEVIATION:       1.49666258
```

Program Listing

```
10 PRINT ")":REM SCREEN CLEAR
20 PRINT "DATA POINT:  ";:INPUT X
30 IF X=999999999 THEN 60
40 T=T+X:S=S+X^2:N=N+1
50 GOTO 20
60 A=T/N:V=S/N-A^2:D=SQR(V)
70 PRINT :PRINT
80 PRINT "DATA POINTS TOTAL:",T
90 PRINT "MEAN:",,A
100 PRINT "VARIANCE:",V
110 PRINT "STD DEVIATION:",D
120 PRINT :PRINT
130 CLR :GOTO 20
```

48 Percentages

Usually it's more convenient to enter percentages as percent rather than having to convert to decimals in your head first. Of course, the computer needs that converted decimal value to do its work. How to get it?

This program does the trick. You give it a percentage and it converts that to a decimal. The computer does the hard work for you!

Line 30 makes the actual conversion. Use this idea as part of a larger check-balancing, accounting or bookkeeping program and save lots of mental effort.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 PRINT "PERCENTAGE:";:INPUT P
30 D=0.01*P
40 PRINT "DECIMAL:",D
50 PRINT :GOTO 20
```

49 Logic Functions

You can make your computer do things based on its decision that something exists. That is, in the first program listing here, it only will print the value of C if it finds that B has an existing value. If B is found to have no value, does not exist, C will not be printed.

The decision is in line 40. The machine only prints C if B does not equal zero. Since, in line 20, we set B = 10, the computer will find that something exists in B and, thus, go ahead and do the work assigned in the last half of line 40. If nothing had been stored in B, the last half of line 40 would have been ignored.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 B=10
30 C=10*B
40 IF B THEN PRINT C
```

In the second program here, the Atari only displays the results of the tests in lines 40 and 50 if the results of one or both is "true."

By doing the simple math in your head, you can see that the information in the right-hand side of line 20 is true. The information in the right-hand side of line 30 is false.

Line 20 says that 6 + 8 is greater than 3 times 4. That is, 14 is greater than 12. That is true.

Line 30 says that 5 + 2 is greater than 9 + 2. That is, 7 is greater than 11. That is false.

After reading line 20, the computer will store a 1 in B since the statement is true. Upon reading line 30, the computer will store a zero in C since the statement is false.

As action drops to line 40, the computer will find the 1 it stored in B and, thus, complete the action called for at the right-hand end of line 40. It will display the message, "B OKAY."

At line 50, however, the computer will find "nothing" (zero) in C and will not complete the right-hand end of that instruction. It only will do the right-hand end if it finds something in the left-hand end.

These logic functions are great for quick tests.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 B=(6+8)>(3*4)
30 C=(5+2)>(9+2)
40 IF B THEN PRINT "B OKAY"
50 IF C THEN PRINT "C OKAY"
```

50 Above & Below a Line

Here's a way to count numbers above and below a cut-off line. The computer solicits numbers between 1 and 100. Any numbers you key in which are below 1 or above 100 are trapped out by line 40. Entering a zero ends the input cycle.

Line 50 counts the total numbers. Line 60 counts only those numbers between 1 and 50. Line 80 counts the numbers from 51 to 100. Lines 90 to 130 present results.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "GIVE ME A NUMBER:";:INPUT Z
30 IF Z=0 THEN 80
40 IF Z<1 OR Z>100 THEN 20
50 N=N+1
60 IF Z<51 THEN B=B+1
70 GOTO 20
80 A=N-B
90 PRINT :PRINT
100 PRINT "NUMBERS TOTAL:",N
110 PRINT "1 TO 50",,B
120 PRINT "51 TO 100:",A
130 PRINT :PRINT :PRINT :PRINT
140 CLR :GOTO 20
```

51 Factoring

This program finds and lists the factors of any number you specify. It can be used as a subroutine in a larger program, with appropriate attention to line numbers, variable names, and RETURN.

The number of individual factors are limited by the DIM statement in line 20.

The list will exclude the number itself divided by 1.

For a quick sample run, try the number 18. You should find factors are 9, 6, 3 and 2.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 DIM Q(1700)
25 FOR L=0 TO 999:Q(L)=0:NEXT L
30 PRINT "NUMBER:",:INPUT N
40 FOR L=2 TO N/2
50 M=N/L
60 IF M=INT(M) THEN Q(L)=M
70 NEXT L
```

```
80 PRINT :PRINT "FACTORS ARE:"
90 FOR L=1 TO N/2
100 IF Q(L)>1 THEN PRINT Q(L):GOTO 120
110 Z=Z+1
120 NEXT L
130 IF N=1 THEN PRINT "NONE":GOTO 150
140 IF Z=INT(N/2) THEN PRINT "NONE"
150 PRINT :PRINT :GOTO 25
```

52 Which is Smallest?

How can the computer tell which number is smaller or larger? Here's how.

Type in the program and RUN it. It will ask for, and accept a continuous string of numbers until you end the input routine by keying in a zero.

Lines 40 to 60 make the decision as to which number is lowest.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "GIVE ME A NUMBER:",:INPUT Z
30 IF Z=0 THEN 80
40 N=N+1
50 IF N=1 THEN D=Z
60 IF Z<D THEN D=Z
70 GOTO 20
80 PRINT :PRINT
90 PRINT "THE SMALLEST NUMBER IS ";D
100 PRINT :PRINT :PRINT :PRINT
110 CLR :GOTO 20
```

53 Which is Largest?

Suppose you have a group of numbers and you would like to know which number is largest within the group?

Here's a software routine for your Atari so it can locate the largest number.

You can key in as many numbers as you wish. To end that entry cycle, type in a zero. The computer will see that zero as its cue to leave the entering routine and get on with computing.

Line 40 tests each new number as you enter it. If a new number is larger, that new number is stored in memory location D. At the end of the entry cycle, the largest number is left stored in D. Line 70 recalls that largest number and prints it.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 PRINT "GIVE ME A NUMBER",:INPUT Z
30 IF Z=0 THEN 60
40 IF Z>D THEN D=Z
50 GOTO 20
60 PRINT :PRINT
70 PRINT "THE LARGEST NUMBER IS ";D
80 PRINT :PRINT :PRINT :PRINT
90 CLR :GOTO 20
```

54 Reciprocals

Key in any number. The computer will display its reciprocal. The actual conversion is done here at line 30.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
15 PRINT "NUMBER TO BE CONVERTED"
20 PRINT "TO ITS RECIPROCAL:  ";
25 INPUT N
30 R=1/N
35 PRINT :PRINT
40 PRINT "RECIPROCAL OF ";N;" IS ";R
45 PRINT :PRINT
50 CLR :GOTO 15
```

55 Dump the Integer

Look at the number 123.456 with an eye toward how to get rid of the portion left of the decimal point. Keep only .456 and dump 123. Here's a short program to accomplish that.

Try 5.67. It will come out .67. Or 500.5 which will come out .5.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "GIVE ME A NUMBER"
30 PRINT "WITH A DECIMAL:"
40 INPUT N
50 X=N-INT(N)
60 PRINT
70 PRINT "THE FRACTIONAL PORTION"
80 PRINT "OF ";N;" IS ";X
90 PRINT :PRINT :PRINT :PRINT
100 CLR :GOTO 20
```

56 Averages

Key in numbers in any order. A zero will end entry. The computer will tell you the average number of all numbers you entered.

Line 40 finds the total number of all numbers entered. Line 50 finds the total of entered numbers. Line 70 computes the average.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 PRINT "GIVE ME A NUMBER",:INPUT Z
30 IF Z=0 THEN 70
40 N=N+1
```

```
50 T=T+Z
60 GOTO 20
70 A=T/N
100 PRINT :PRINT
110 PRINT "THE AVERAGE IS ";A
120 PRINT :PRINT :PRINT :PRINT
130 CLR :GOTO 20
```

57 Mid-Range Number

Here's how to find the middle of a range of numbers. You key in as many numbers in a series as you wish. After the last number, key in a zero to move the program out of the entry cycle.

Lines 40 to 70 select the highest and lowest numbers in the range. They actually define the range. Then line 90 finds the middle point of that range.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "GIVE ME A NUMBER",:INPUT Z
30 IF Z=0 THEN 90
40 N=N+1
50 IF N=1 THEN H=Z:L=Z
60 IF Z<L THEN L=Z
70 IF Z>H THEN H=Z
80 GOTO 20
90 M=L+((H-L)/2)
100 PRINT :PRINT
110 PRINT "MID-RANGE NUMBER:",M
120 PRINT :PRINT :PRINT :PRINT
130 CLR :GOTO 20
```

58 Rounding Off

The technique for rounding off numbers is easy.

This program, which can stand alone or be worked into a larger program as a subroutine, rounds a decimal to the nearest whole number.

There are two views on how to round off. One holds that "if the number is more than five, you round up." Which means that exactly 0.5 rounds down.

Another view is that "any number less than five rounds down." In that case exactly 0.5 rounds up.

The first set of program lines below is for the fellow with the "more than five rounds up" idea.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "GIVE ME A NUMBER"
30 PRINT "TO BE ROUNDED OFF"
40 INPUT N
50 IF N>INT(N) THEN 80
60 R=N
70 GOTO 130
80 D=N-INT(N)
90 IF D>0.5 THEN 120
100 R=INT(N)
110 GOTO 130
120 R=INT(N)+1
130 PRINT "}":REM BUZZER
140 PRINT N;" ROUNDS OFF TO ";R
150 PRINT :PRINT :PRINT
160 CLR :GOTO 20
```

The second set of program lines rounds off on the "less than five rounds down" theory.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "GIVE ME A NUMBER"
30 PRINT "TO BE ROUNDED OFF"
40 INPUT N
50 IF N>INT(N) THEN 80
60 R=N
70 GOTO 130
```

```
80  D=N-INT(N)
90  IF D<0.5 THEN 120
100 R=INT(N)+1
110 GOTO 130
120 R=INT(N)
130 PRINT "}":REM BUZZER
140 PRINT N;" ROUNDS OFF TO ";R
150 PRINT :PRINT :PRINT
160 CLR :GOTO 20
```

59 Two-Digit Round Off

It is possible to round off to the nearest hundredths place. That is, to two digits after the decimal point. Here's how:

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "GIVE ME A NUMBER TO"
30 PRINT "MORE THAN 2 DECIMAL PLACES:"
40 INPUT N
50 R=INT(100*N+0.5)/100
60 PRINT
70 PRINT N;" ROUNDS OFF TO ";R
80 PRINT "OR"
90 PRINT "$";N;" BECOMES $";R
100 PRINT :PRINT :PRINT :PRINT
110 CLR :GOTO 20
```

60 Percent to Decimal

Checking, interest, sales tax, and other financial programs are more "user friendly" if you don't have to make manual conversions in your head. For example, if you know your savings account earns 8 percent interest, and you need to multiply by the decimal value for 8 percent (which is 0.08), it is easier to be able to enter 8 and let the Atari figure out the decimal value.

Here's another way to change percentages to decimals inside a program to simplify entry by permitting percents to be entered as simple numbers.

For some examples, try entering a price of 2.50 and a sales tax percentage of 6. Your Atari will find the bill totals $2.65. Or try $7.80 and 5 percent tax. The bill will be $8.19. Try $123.75 at 8 percent tax. The bill will total $133.65.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 PRINT "PRICE $",,:INPUT P
30 PRINT "SALES TAX %",:INPUT R
40 T=0.01*R
50 S=T*P:B=P+S
60 PRINT "SALES TAX","$";S
70 PRINT "TOTAL BILL","$";B
80 PRINT :PRINT :PRINT :PRINT
90 CLR :GOTO 20
```

61 Every 10th Answer

This program generates a random number in the range of zero to 999. However, it has a difference. It only shows you every tenth number it generates.

Line 20 generates the numbers. Line 40 selects the tenth number from each set.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 T=INT(1000*(RND(1)))
30 V=V+1
40 IF 0.1*V=INT(0.1*V) THEN PRINT V,T
50 GOTO 20
```

62 Random Sampler

This program strengthens your confidence in the random number generator built into your Atari computer.

It generates 100 numbers between zero and 100 and tells you how many of those are above 49 and how many are below 50. See the sample RUN for several sets of results in our recent test.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 FOR L=1 TO 100
30 X=INT(100*(RND(1)))
40 IF X<50 THEN Y=Y+1
50 IF X>49 THEN N=N+1
60 NEXT L
70 PRINT Y;" YES"
80 PRINT N;" NO"
90 PRINT :PRINT :PRINT
100 CLR :GOTO 20
```

Sample Run

```
51  YES        46  YES
49  NO         54  NO

42  YES        52  YES
58  NO         48  NO

56  YES        48  YES
44  NO         52  NO
```

63 Random Numbers: Zero To Nine

Although you see four program lines below, what we really have here is a very convenient single-line program for you to insert in a larger game or educational-testing program.

Line 20 is the winner here. It prints a random number from zero to nine every time. For your use here, we print that number on the screen. You could just as easily have the computer store that random number in a memory location for later recall and use.

We have added lines 10, 30 and 40 to make your Atari show you a whole series of random numbers from zero to nine. Remember, line 20 is the important single-line program element here.

If you would like random numbers in the range from zero to 99, make it 100* in line 20. For zero to 999, use 1000* in line 20.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT INT(10*(RND(1)))
30 FOR L=1 TO 200:NEXT L
40 GOTO 20
```

64 Random Numbers: Distribution

Ever wonder how "random" are the numbers generated by the random-number generator in your Atari when you use the RND instruction? Try this program.

It generates 100 random numbers in a range from zero to nine and counts how many there are of each number between zero and nine.

By the way, while it is doing that it will display the message "counting" so you can tell it is working.

At the end of its run, the Atari prints a neat chart, on the video display, of results. See our sample run.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
```

```
20 FOR L=1 TO 100
30 N=INT(10*(RND(1)))
40 IF N=0 THEN A=A+1
50 IF N=1 THEN B=B+1
60 IF N=2 THEN C=C+1
70 IF N=3 THEN D=D+1
80 IF N=4 THEN E=E+1
90 IF N=5 THEN F=F+1
100 IF N=6 THEN G=G+1
110 IF N=7 THEN H=H+1
120 IF N=8 THEN I=I+1
130 IF N=9 THEN J=J+1
140 PRINT "COUNTING"
150 NEXT L
160 PRINT "}":REM CLEAR SCREEN
170 PRINT "}":REM BUZZER
200 PRINT "0",A
210 PRINT "1",B
220 PRINT "2",C
230 PRINT "3",D
240 PRINT "4",E
250 PRINT "5",F
260 PRINT "6",G
270 PRINT "7",H
280 PRINT "8",I
290 PRINT "9",J
300 END
```

Sample Run

```
RUN RETURN

COUNTING

0    8
1   10
2   14
3   16
4    6
5    7
6    7
7   15
```

```
8    4
9   13

RUN   RETURN

COUNTING

0    6
1   12
2   11
3   13
4    7
5   12
6    9
7   10
8    8
9   12
```

65 Random Numbers: Averages

This program generates 100 random numbers and totals them. Then it finds the average of all 100 numbers.

In fact, the average number itself is a useful new random number.

To make the program run again, press the RETURN key on the computer's keyboard.

Program Listing

```
10  PRINT "}":REM CLEAR SCREEN
20  FOR L=0 TO 99
30  N=INT(10*(RND(1)))
40  NT=NT+N
50  PRINT "AVERAGING"
60  NEXT L
70  PRINT "}":REM SCREEN CLEAR
80  AV=NT/100
90  PRINT "TOTAL OF 100 RANDOM NUMBERS"
100 PRINT "BETWEEN ZERO AND NINE IS ";NT
```

```
110 PRINT "AVERAGE IS ";AV
200 DIM K$(1)
210 PRINT "FOR MORE, PRESS RETURN"
220 INPUT K$
230 CLR :GOTO 10
```

66 Random Numbers: Sorting High/Low

It's important to be able to sort a group of numbers to see what the highest and lowest values are. This program does that.

The random number generator is in line 30. It gives numbers in a range of zero to 999. Line 50 determines the lowest number in the set and line 60 finds the highest number.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 FOR L=0 TO 99
30 N=INT(1000*(RND(1)))
40 IF L=0 THEN LN=N:HN=N
50 IF N<LN THEN LN=N
60 IF N>HN THEN HN=N
70 PRINT "SORTING"
80 NEXT L
90 PRINT CHR$(253):REM BUZZER
100 PRINT "}":REM CLEAR SCREEN
110 PRINT "LOW NUMBER IS ",LN
120 PRINT "HIGH NUMBER IS ",HN
200 END
```

Sample Run

```
RUN  RETURN

SORTING
```

```
LOW NUMBER  IS 32
HIGH NUMBER IS 983

LOW NUMBER IS 14
HIGH NUMBER IS 980

LOW NUMBER IS 17
HIGH NUMBER IS 985

LOW NUMBER IS 9
HIGH NUMBER IS 991

LOW NUMBER IS 1
HIGH NUMBER IS 994
```

Money Matters

67 Money Grows

This section of the book includes a number of programs relating to household money management and to small-business applications. This first program shows you how your money grows when deposited in a savings account at a certain annual interest rate, compounded monthly.

The program will have the computer ask for the initial amount of principal saved by depositing in the account. Then the annual interest rate and the number of months to be displayed. The result of the run is a display of the changing principal as months pass and interest is added on.

Line 10 clears the text screen. Lines 20 and 40 take in data from you. Lines 50 to 90 put out the results. Very handy!

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 PRINT "PRINCIPAL $",,:INPUT P
30 PRINT "ANNUAL INTEREST RATE %",:INPUT R
40 PRINT "NUMBER MONTHS",,:INPUT M
50 FOR Q=1 TO M
60 I=(P*(0.01*R))/12
70 P=P+I
80 PRINT Q;" MONTH = $";P
90 NEXT Q
100 PRINT :PRINT
110 PRINT "FOR ANOTHER, PRESS RETURN"
120 DIM K$(1):INPUT K$
130 CLR :GOTO 10
```

68 Shopper's Friend

This program finds the computer asking for certain information and then telling you which product brand name is the best buy.

The computer wil ask for the brand name of a product, the quantity in the product package, and the price of the package. Then it will ask for the name, quantity and price for a second product.

After digesting all this information, it will tell you the brand name of the best-buy product and show you the unit prices for both brand names so you can agree with the computer's judgment.

For example, suppose you were looking at corn flakes in boxes, one by Post and one by Kellogg. Suppose the Post box contained 24 ounces of flakes and was priced on the grocery shelf at $1.98 while the Kellogg box held 18 ounces and was priced at $1.59. Which would be the better buy based on unit price per ounce of flakes?

Run the data through your Atari and you'll find it computes the Post corn flakes to be the best buy with a unit price of 8¢ vs. the Kellogg unit price of 9¢.

By the way, if the unit prices turn out to be equal, the computer will say they are equal.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 DIM X$(20),Y$(20),K$(1)
30 PRINT "          SHOPPER'S FRIEND"
40 PRINT "FIRST BRAND:",:INPUT X$
50 PRINT "QUANTITY:",:INPUT M
60 PRINT "PRICE:",,:INPUT N
70 PRINT "SECOND BRAND:",:INPUT Y$
80 PRINT "QUANTITY:",:INPUT Q
90 PRINT "PRICE:",,:INPUT R
100 IF N/M=R/Q THEN 500
110 IF N/M<R/Q THEN 300
200 PRINT :PRINT Y$;" IS BEST BUY"
210 GOTO 400
300 PRINT :PRINT X$;" IS BEST BUY"
400 PRINT :PRINT X$;" UNIT:","$";N/M
410 PRINT Y$;" UNIT:","$";R/Q
420 FOR L=1 TO 8:PRINT :NEXT L
430 PRINT "TO DO ANOTHER, PRESS RETURN"
440 INPUT K$
```

```
45Ø CLR :GOTO 1Ø
5ØØ PRINT :PRINT X$;" = ";Y$
51Ø GOTO 4ØØ
```

69 Car Payments

Shopping for a new car? Use your Atari computer to compute quickly the potential monthly car payment on various models.

Imagine you want an $8000 car and are prepared to put up $1000 against the purchase. You want to arrange to finance the car for 36 months. You know the current annual interest rate on car loans is 15 percent.

Key in those few numbers and the computer instantly tells you the car payment will be $242.66 per month.

Program Listing

```
1Ø PRINT "}":REM CLEAR SCREEN
2Ø DIM K$(1)
3Ø FOR L=1 TO 38:PRINT "*";:NEXT L
4Ø PRINT "*          AUTOMOBILE PAYMENT
               *";
5Ø FOR L=1 TO 38:PRINT "*";:NEXT L
6Ø PRINT "PURCHASE PRICE $:",,:INPUT T
7Ø PRINT "DOWN PAYMENT $:",,:INPUT R
8Ø PRINT "NUMBER OF MONTHS:",,:INPUT N
9Ø PRINT "ANNUAL INTEREST %:",,:INPUT I
1ØØ I=(Ø.Ø1*I)/12
11Ø P=(T-R)*I/(1-1/(1+I)^N)
12Ø PP=INT(1ØØ*P+Ø.5)/1ØØ
13Ø PRINT :PRINT "PAYMENT:","$";PP
14Ø FOR L=1 TO 38:PRINT "*";:NEXT L
15Ø PRINT :PRINT
16Ø PRINT "FOR MORE, PRESS RETURN"
17Ø INPUT K$
18Ø CLR :GOTO 1Ø
```

70 To Nearest 95 Cents

Many companies like to price their goods at a figure ending in 95 cents. For instance, a ten dollar item might be marked $9.95 or $10.95.

Here's a program which demonstrates how to make all prices come out to the nearest 95 cents. See line 40. It merely takes the integer portion of the dollars number and adds 0.95 to it.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 PRINT "MANUFACTURING COST= $":INPUT C
30 PRINT "PRICING MULTIPLIER= ":INPUT M
40 P=INT(C*M)+0.95
50 PRINT
60 PRINT "RETAIL PRICE= $";P
70 PRINT :PRINT
80 CLR :GOTO 20
```

Sample Run

```
MFG COST = $
1.12  RETURN
PRICING MULTIPLIER =
10  RETURN
RETAIL PRICE = $11.95
RETURN
```

71 To the Nearest Penny

This program is useful when you have a dollar-and-cents figure with more than two decimal places. For example, $151.6972. You need to transform $151.6972 to the more common $151.70

This small program would make a good subroutine in a larger set of instructions. To do so, insert GOSUB at the appropriate place in the larger set of program lines. Modify the line numbers of this small program so the subroutine will be located in an unused position in the larger listing. Change the last line of this small program to RETURN. Delete the first line.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 PRINT "ENTER A NUMBER TO"
30 PRINT "MORE THAN TWO DECIMAL PLACES"
40 PRINT :PRINT "ORIGINAL AMOUNT = ";
50 INPUT N
60 R=INT(100*N+0.5)/100
70 PRINT :PRINT "TO THE NEAREST PENNY,"
80 PRINT "$";N;" IS $";R
90 PRINT :PRINT
100 CLR :GOTO 40
```

72 Mark Up

Mr. Storekeeper, here's just what you have needed to compute mark ups. This program causes your APPLE to find the retail price for which your percentage off would give the wholesale cost.

For instance, if you got 40 percent off on an item and paid $60, how much was it priced at, at retail? The answer is $100. To put that another way, if retail price or suggested retail price is $100 and you got 40 percent off at wholesale, what is the wholesale price? The answer is $60.

Try $40 wholesale which is 60 percent off. The answer is $100 retail. Or try $10 wholesale at 90 percent off. Retail would be $100. Or $75 wholesale at 25 percent off gives $100 retail.

Here's a toughie! Try $19.95 wholesale cost. Mark-up percentage is 40. The correct retail answer is $33.25.

Program Listing

```
10 PRINT "}":REM CLEAR SCREEN
20 PRINT "WHOLESALE COST $":INPUT W
30 PRINT "MARK-UP PERCENTAGE %":INPUT P
40 D=1-0.01*P:R=W/D
50 PRINT "RETAIL PRICE = $";R
60 PRINT :PRINT
70 CLR :GOTO 20
```

73 Percentage Off

From earlier tips in this book, you know how to make your Atari convert percentages to decimals. But what if you want to know "percentage off?"

For example, how much is 40 percent off? This program can be used to interpret 40 percent off and compute the decimal value needed. Try 40 percent off $100. The computer will change 40 percent off into decimal value 0.60. If you multiply 0.60 times $100 you find $60 is 40 percent off $100.

Line 50 makes the important translation.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "LIST PRICE $":INPUT L
30 PRINT "PERCENTAGE OFF %":INPUT P
40 PRINT "}":REM CLEAR SCREEN
50 D=1-0.01*P
60 PRINT "TO COMPUTE WITH ";P;"% OFF"
70 PRINT "THE DECIMAL WILL BE ";D
80 PRINT
90 PRINT P;"% OFF $";L
100 PRINT "RESULTS IN A COST OF $";D*L
110 PRINT :PRINT
120 CLR :GOTO 20
```

74 Dollars & Cents

If the result of your computation is a "money" answer, and you don't know whether to display it in dollars or cents, let the computer decide.

This program decides whether to display the output in dollars or cents. Line 50 in the program makes the decision.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "QUANTITY:",:INPUT P
30 PRINT "TOTAL COST $",:INPUT C
40 T=C/P
50 IF T<1 THEN T=100*T:GOTO 80
60 PRINT "EACH COST $";T
70 PRINT :PRINT :CLR :GOTO 20
80 PRINT "EACH COST ";T;" CENTS"
90 PRINT :PRINT :CLR :GOTO 20
```

75 Wages & Hours

These useful lines compute total hours worked at regular pay and number of hours worked at time-and-a-half overtime. The computer then finds gross pay and rounds off to the nearest cent.

The program knows that overtime starts after 40 hours. It makes payroll bookkeeping quick and simple.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "HOURLY PAY RATE = $",:INPUT P
30 PRINT "NUMBER HOURS WORKED = ",:INPUT H
40 IF H>40 THEN OT=H-40:GOTO 100
```

```
50 W=H*P
60 PRINT "GROSS WAGES =","$";W
70 END
100 W=(40*P)+(OT*P*1.5)
110 GOTO 60
```

Sample Run

```
RUN RETURN

HOURLY PAY RATE = $
5.75 RETURN
NUMBER HOURS WORKED =
61 RETURN
GROSS WAGES = $411.125
```

76 Invoicing

There's a lot of repetitious math work to be done before you mail invoices to your customers. This software has the computer collect a few pertinent bits of data from you and then present all the various totals you need to plug into an invoice.

It gives you a total retail price for all goods sold on the invoice, total sales tax if applicable, shipping charges and the grand total amount due you from your customer.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 PRINT "QUANTITY SOLD",:INPUT Q
30 PRINT "UNIT PRICE = $",:INPUT P
40 PRINT "SALESTAX RATE PERCENT=",:INPUT S
50 PRINT "SHIPPING CHARGES = $",:INPUT H
60 S=0.01*S:C=Q*P:T=C*S:F=C+T+H
70 CC=INT(100*C+0.5)/100
80 TT=INT(100*T+0.5)/100
90 FF=INT(100*F+0.5)/100
100 PRINT "}":REM CLEAR SCREEN
```

```
110 PRINT "TOTAL PRICE = $";CC
120 PRINT "SALES TAX = $";TT
130 PRINT "SHIPPING CHARGES = $";H
140 PRINT
150 PRINT "INVOICE TOTAL = $";FF
200 END
```

Sample Run

```
RUN  RETURN

QUANTITY SOLD
178  RETURN
UNIT PRICE = $
55.98 RETURN
SALES TAX RATE PERCENT =
6    RETURN
SHIPPING CHARGES = $
100  RETURN

TOTAL PRICE = $9964.44
SALES TAX = $597.87
SHIPPING CHARGES = $100

INVOICE TOTAL = $10662.31
```

77 Unit Price

Suppose you find 895 green Widgets and buy them for $695. How much did each green Widget cost? Rounded off, $7.77.

Unit price is total price divided by quantity. The quantity can be expressed in weight, total numbers, etc. It works the same whether you are talking about pounds of coffee, yards of concrete, gallons of ice cream, boxes of books, or units of Widgets.

This program asks for the name of the item, quantity purchased and total price paid. It then displays quantity, name, total and unit price.

Program Listing

```
10  PRINT "}":REM SCREEN CLEAR
20  DIM N$(20),K$(1)
30  PRINT "ITEM NAME IS ":INPUT N$
40  PRINT "QUANTITY OF ITEMS = ":INPUT Q
50  PRINT "TOTAL PRICE PAID FOR ITEMS $"
60  INPUT P
70  U=P/Q
80  UU=INT(100*U+0.5)/100
90  PRINT "}":REM BUZZER
100  PRINT N$;" UNIT PRICE = $";UU
110  PRINT :PRINT :PRINT
120  PRINT "TO DO MORE, PRESS RETURN"
130  INPUT K$
140  CLR :GOTO 10
```

Sample Run

```
RUN  RETURN

ITEM NAME IS
WIDGETS RETURN
QUANTITY OF  ITEMS =
999 RETURN
TOTAL PRICE PAID FOR ITEMS = $
14653 RETURN

BEEP WIDGETS UNIT PRICE = $14.67
```

78 Inventory Counter

The computer makes it very convenient to tally the number of items in your inventory.

This workaholic program is set up for up to ten different inventory items. You code them with the numbers 1 through 10.

You may enter any quantity for any item code in any mixed sequence. You may repeat item codes and add to quantities as often as you like.

When you finish entering quantities, enter a zero in response to the "item code" query. The computer will respond with a display of grand totals for each of the ten item codes.

Here's the routine:

Line 20 asks for the item code number from one through 10. If you enter a zero, action jumps to line 70 for a display of grand totals. Otherwise, the computer proceeds to line 40.

You are limited to item-code responses from zero through ten. If you try to enter a number larger than 10, the computer will discover that via the test in line 40 and ship the whole operation back to line 20 where it asks you again for a valid number. If, at line 40, it finds a valid item code number from one to ten it will allow the execution to go on to line 50.

At line 50, the computer asks for the quantity of the item. Line 60 causes the computer to jump to the appropriate line to add that quantity to the various item totals.

Note that the portion of the program from line 201 through line 210 has line numbers stepping up by ones rather than the conventional tens. This is because of the way line 60 jumps. It takes the item code number from one through ten and adds that to 200 to create a number from 201 to 210. It then jumps to that number.

Suppose your item code was 5. The computer would add 5 to 200 and get 205. It then would jump to line 205. A neat sorting trick!

Each of the lines 201 through 210 end with a jump to line 20. This allows you to continue to enter item codes and quantities as long as you like. When you finish, enter zero for item code and line 30 will push action to line 70.

At line 70, the computer finds instructions, through line 90, to display the grand totals.

Lines 100 to 130 ask if you want to do more.

To make the program run again, press the RETURN key on the computer's keyboard.

Program Listing

```
1Ø PRINT "}":REM SCREEN CLEAR
2Ø PRINT "ITEM CODE:  ",:INPUT C
3Ø IF C=Ø THEN 7Ø
```

```
40 IF C>10 THEN 20
50 PRINT "QUANTITY:  ",:INPUT Q
60 GOTO C+200
70 PRINT "}":REM CLEAR SCREEN
80 GOSUB 300
100 PRINT :PRINT "FOR MORE,PRESS RETURN"
110 DIM K$(1)
120 INPUT K$
130 CLR :GOTO 10
201 J=J+Q:GOTO 20
202 K=K+Q:GOTO 20
203 L=L+Q:GOTO 20
204 M=M+Q:GOTO 20
205 N=N+Q:GOTO 20
206 R=R+Q:GOTO 20
207 S=S+Q:GOTO 20
208 T=T+Q:GOTO 20
209 U=U+Q:GOTO 20
210 V=V+Q:GOTO 20
220 END
300 PRINT "1:",J
310 PRINT "2:",K
320 PRINT "3:",L
330 PRINT "4:",M
340 PRINT "5:",N
350 PRINT "6:",R
360 PRINT "7:",S
370 PRINT "8:",T
380 PRINT "9:",U
390 PRINT "10:",V
400 RETURN
```

79 Daily Code

Need to have a secret code each day of the year? This software generates a list of code numbers. Of course, you can change the list every day if you wish.

Program Listing

```
10 PRINT "}":REM SCREEN CLEAR
20 GOSUB 200
100 PRINT "SUNDAY:",,C:GOSUB 200
110 PRINT "MONDAY:",,C:GOSUB 200
120 PRINT "TUESDAY:",,C:GOSUB 200
130 PRINT "WEDNESDAY:",C:GOSUB 200
140 PRINT "THURSDAY:",C:GOSUB 200
150 PRINT "FRIDAY:",,C:GOSUB 200
160 PRINT "SATURDAY:",C:END
200 C=INT(10000*(RND(1)))
210 IF C<1000 THEN 200
220 RETURN
```

Colorful Graphics

80 Aztec Art

In this *Colorful Graphics* section of the book, you will find a number of interesting new and different applications for the graphics capabilities of the ATARI computer. These can be modified, combined or otherwise changed to suit your own needs. Our titles represent only the thoughts we had when we watched these programs run. You might like to dream up new and different titles for your own creations made by modifying these programs.

Colors can be changed. Screen locations can be changed. Movement can be reversed. Try all of these programs. You'll like them!

Our first program reminded us of *Aztec* artwork.

Program Listing

```
10 GRAPHICS 3+16
20 FOR L=1 TO 0 STEP -1
30 FOR N=1 TO 12
40 COLOR L
50 PLOT 20,10
60 X=20-(10*COS(N/6*3.14))
70 Y=10+(10*SIN(N/6*3.14))
80 DRAWTO X,Y
90 NEXT N
100 NEXT L
110 GOTO 20
```

81 Sine Wave

Note that some complex screen art can be created with only a few program lines. You will want to refine these programs to make them even shorter.

By the way 3+16 used in line 10 is there to select graphics screen three and make it a full screen, rather than one with some text window at the bottom. The +16 accomplishes the full-screen trick.

Program Listing

```
10 GRAPHICS 3+16
20 COLOR 1
30 FOR X=1 TO 39
40 Y=SIN(X)+10
60 PLOT X,Y
70 NEXT X
80 GOTO 80
```

82 Hold That Pose

The illusion of motion is important in computer graphics. But, to make a character move, you have to draw it first!

Here we have created a friendly little stick man. Lines 100 to 140 create the head. Lines 150 and 160, the eyes. Line 170, the nose. Lines 180 and 190, the mouth.

Lines 200 to 280 create the body, legs and arms. But, having created him on the graphics screen, how can we keep him there? By using a freeze frame as we have done here in line 300.

Line 300 is an endless loop, holding action at line 300 forever, until you press the break key.

Run this program first. Then, go on to Tip Number 83.

Program Listing

```
10 GRAPHICS 9
100 PLOT 35,50
110 DRAWTO 45,50
120 DRAWTO 45,70
130 DRAWTO 35,70
140 DRAWTO 35,50
150 PLOT 37,55
160 PLOT 43,55
170 PLOT 40,60
180 PLOT 37,65
190 DRAWTO 43,65
```

```
200 PLOT 40,70
210 DRAWTO 40,100
220 DRAWTO 45,100
230 DRAWTO 35,100
240 DRAWTO 40,100
250 DRAWTO 40,130
260 DRAWTO 45,140
270 DRAWTO 40,130
280 DRAWTO 35,140
300 GOTO 300
```

83 Okay, Now Wave

This is a development of the stick man drawn and frozen on the screen in Tip Number 82. Here, we add program lines to draw and seem to move the stick man's left arm on the right side of the TV screen.

This program is the same as the previous program, through line 280. Change line 300 and add what we show here through line 430. Here's how it works to create the illusion of arm movement:

Lines 300 to 320 draw the arm in an upward position. Line 330 is a time-delay pause so you have time to see the arm in an upward position. Lines 340 to 360 erase the same arm.

Lines 370 to 390 draw a downward arm. Lines 400 to 420 erase the downward arm. Line 430 makes everything start over again with the upward arm. The result is the appearance of arm motion. Very clever, these stick men!

Program Listing

```
10 GRAPHICS 9
20 COLOR 8
100 PLOT 35,50
110 DRAWTO 45,50
120 DRAWTO 45,70
130 DRAWTO 35,70
140 DRAWTO 35,50
```

```
150 PLOT 37,55
160 PLOT 43,55
170 PLOT 40,60
180 PLOT 37,65
190 DRAWTO 43,65
200 PLOT 40,70
210 DRAWTO 40,100
220 DRAWTO 45,100
230 DRAWTO 35,100
240 DRAWTO 40,100
250 DRAWTO 40,130
260 DRAWTO 45,140
270 DRAWTO 40,130
280 DRAWTO 35,140
300 COLOR 8
310 PLOT 45,100
320 DRAWTO 50,85
330 FOR T=1 TO 200:NEXT T
340 COLOR 0
350 PLOT 50,85
360 DRAWTO 45,100
370 COLOR 8
380 DRAWTO 50,115
390 FOR T=1 TO 200:NEXT T
400 COLOR 0
410 PLOT 50,115
420 DRAWTO 45,100
430 GOTO 300
```

84 Moving Illusion

Beware! This constantly-moving image may drive you batty.

Program Listing

```
10 PRINT "}":REM BUZZER
20 GRAPHICS 10
30 POKE 704,96
40 POKE 705,22
```

```
50 POKE 706,38
60 POKE 707,54
70 POKE 708,70
80 POKE 709,86
90 POKE 710,104
100 POKE 711,120
110 POKE 712,180
120 FOR X=1 TO 64
130 LL=191-X:LR=79-X
140 M=M*(M<8)+1
150 COLOR M
160 PLOT X,X
170 DRAWTO X,LL
180 DRAWTO LR,LL
190 DRAWTO LR,X
200 DRAWTO X,X
210 NEXT X
220 A=PEEK(712):B=PEEK(711):C=PEEK(710)
230 D=PEEK(709):E=PEEK(708):F=PEEK(707)
240 G=PEEK(706):H=PEEK(705)
250 POKE 712,B
260 POKE 711,C
270 POKE 710,D
280 POKE 709,E
290 POKE 708,F
300 POKE 707,G
310 POKE 706,H
320 POKE 705,A
330 GOTO 220
```

85 Super Moving Illusion

If you liked Tip Number 84 above, you'll love this one! Here the background color changes as well as the color of the lines. Very striking!

Program Listing

```
10 PRINT "}":REM BUZZER
20 GRAPHICS 10
30 POKE 704,96
40 POKE 705,22
50 POKE 706,38
60 POKE 707,54
70 POKE 708,70
80 POKE 709,86
90 POKE 710,104
100 POKE 711,120
110 POKE 712,180
120 FOR X=1 TO 64
130 LL=191-X:LR=79-X
140 M=M*(M<8)+1
150 COLOR M
160 PLOT X,X
170 DRAWTO LR,X
180 DRAWTO LR,LL
190 DRAWTO X,LL
200 DRAWTO X,X
210 NEXT X
220 J=96
230 A=PEEK(712):B=PEEK(711):C=PEEK(710)
240 D=PEEK(709):E=PEEK(708):F=PEEK(707)
250 G=PEEK(706):H=PEEK(705)
260 POKE 712,B
270 POKE 711,C
280 POKE 710,D
290 POKE 709,E
300 POKE 708,F
310 POKE 707,G
320 POKE 706,H
330 POKE 705,A
340 J=J+1
350 POKE 704,J
360 IF J=255 THEN J=1
370 GOTO 230
```

86 The Seamstress

It's either somebody sewing or a spider weaving a web or something...

Program Listing

```
10 GRAPHICS 8+16
20 COLOR 255
30 PLOT 160,96
40 X=INT(320*(RND(1)))
50 Y=INT(192*(RND(1)))
60 DRAWTO X,Y
70 GOTO 40
```

87 Swimming Fish

Here's another interesting demonstration of your ability to erase images from the screen. The erasing of the fish, here, creates an illusion of motion across the screen.

The fish appears, swims right to left across the screen, and concludes with an *end of swim* message.

Program Listing

```
10 GRAPHICS 8+16
20 X=319:Y=96
30 COLOR 0
40 PLOT X-20,Y
50 DRAWTO X-10,Y-10
60 DRAWTO X,Y+10
70 DRAWTO X-10,Y
80 DRAWTO X,Y-10
90 DRAWTO X-10,Y+10
100 DRAWTO X-20,Y
110 X=X-10
120 IF X<21 THEN 300
130 COLOR 255
```

```
140 PLOT X-20,Y
150 DRAWTO X-10,Y-10
160 DRAWTO X,Y+10
170 DRAWTO X-10,Y
180 DRAWTO X,Y-10
190 DRAWTO X-10,Y+10
200 DRAWTO X-20,Y
210 GOTO 30
300 PRINT "END OF SWIM"
```

88 Circling Dot

More round graphics on your rectangular picture tube.

Program Listing

```
10 GRAPHICS 3+16
20 FOR N=1 TO 12
30 COLOR 1
40 PLOT 20,10
50 X=20-(10*COS(N/6*3.14))
60 Y=10+(10*SIN(N/6*3.14))
70 PLOT X,Y
80 FOR T=1 TO 50:NEXT T
90 COLOR 0
100 PLOT X,Y
110 NEXT N
120 GOTO 20
```

89 Box The Screen

Here's how to draw a box around the graphics display-screen area on your TV monitor. Lines 30 to 70 draw the vertical sides of the box while lines 80 to 120 draw the horizontal bottom and top. Line 130 is a freeze-frame loop to hold the picture so you can see it.

Program Listing

```
10 GRAPHICS 3+16
20 COLOR 1
30 FOR Y=0 TO 23 STEP 23
40 FOR X=0 TO 39
50 PLOT X,Y
60 NEXT X
70 NEXT Y
80 FOR X=0 TO 39 STEP 39
90 FOR Y=0 TO 23
100 PLOT X,Y
110 NEXT Y
120 NEXT X
130 GOTO 130
```

90 Window Twinklers

Well, what would you call them?

Program Listing

```
10 GRAPHICS 3+16
20 COLOR 1
30 FOR Y=0 TO 23 STEP 23
40 FOR X=0 TO 39
50 PLOT X,Y
60 NEXT X
70 NEXT Y
80 FOR X=0 TO 39 STEP 39
90 FOR Y=0 TO 23
100 PLOT X,Y
110 NEXT Y
120 NEXT X
130 C=INT(5*(RND(1)))
140 COLOR C
150 X=INT(40*(RND(1)))
160 IF X<1 OR X>38 THEN 150
170 Y=INT(24*(RND(1)))
```

```
180 IF Y<1 OR Y>22 THEN 170
190 PLOT X,Y
200 GOTO 130
```

91 Horn-In-Funnel Art

The background color keeps changing.

Program Listing

```
10 GRAPHICS 10
20 FOR Z=1 TO 96 STEP 5
30 POKE 704,Z
40 POKE 705,22
50 POKE 706,38
60 POKE 707,54
70 POKE 708,70
80 POKE 709,86
90 POKE 710,104
100 POKE 711,120
110 POKE 712,180
120 FOR X=1 TO 64
130 LL=191-X:LR=79-X
140 C=C*(C<8)+1
150 COLOR C
160 PLOT X,X
170 DRAWTO X,LL
180 DRAWTO LR,LL
190 DRAWTO LR,X
200 DRAWTO X,X
210 NEXT X
220 NEXT Z
230 GOTO 230
```

92 Blackboard

This program appears to draw a blackboard on the screen. You can write messages on it, draw football plays, etc. Use PLOT and DRAWTO to create art or words on this electronic chalkboard.

Program Listing

```
10 GRAPHICS 10
20 POKE 704,112
30 COLOR 255
40 PLOT 70,100
50 DRAWTO 70,10
60 DRAWTO 10,10
70 POSITION 10,100
80 POKE 765,1
90 XIO 18,#6,0,0,"S:"
100 GOTO 100
```

93 Snowfall

White flakes sprinkle down the screen, over and over—until you press the BREAK key. It may be useless but it's a lot of fun to watch!

Program Listing

```
10 GRAPHICS 3+16
20 FOR L=1 TO 84
30 PRINT #6,"*";
40 NEXT L
50 GRAPHICS 0
60 GOTO 10
```

94 Making Things Move

Movement on the computer display screen is an illusion. As in any television picture, the turning on and turning off of dots in a pattern across a screen can seem to provide motion to an object drawn on the face of the tube.

There are a number of ways to get the look of motion. Let's send a dot across the screen:

Program Listing

```
10 GRAPHICS 3+16
20 FOR X=0 TO 39
30 COLOR 65
40 PLOT X,10
50 FOR T=1 TO 50:NEXT T
60 COLOR 0
70 PLOT X,10
80 NEXT X
90 GOTO 20
```

95 Draw A Box

Line 10 establishes the graphics screen. Line 20 sets the background color. Lines 30 to 70 draw a box. Line 100 is a freeze-frame device. Now you know how to draw a box!

Program Listing

```
10 GRAPHICS 10
20 POKE 704,96
25 COLOR 255
30 PLOT 10,10
40 DRAWTO 70,10
50 DRAWTO 70,100
60 DRAWTO 10,100
70 DRAWTO 10,10
100 GOTO 100
```

96 Luminance Demonstrator

Take a look at 116 shades of a color!

Line 10 selects graphics mode 9 which permits 116 shades of a color. Line 20 sets the background to the color desired. We have selected purple for this example.

The FOR/NEXT loop in lines 30 to 70 causes the computer to run through the shades. Line 40 selects the shade. Line 50 draws the screen pattern. Line 80 is a freeze frame to hold the picture.

Program Listing

```
10 GRAPHICS 9
20 SETCOLOR 4,6,0
30 FOR X=0 TO 15
40 COLOR X
50 PLOT X,0
60 DRAWTO X,191
70 NEXT X
80 GOTO 80
```

97 Draw A Line

It may be an odd way to achieve the goal but these programs draw a line across the screen.

The first program, below, paints from top to bottom down the screen, then left to right, leaving a dark horizontal line at about the middle of the screen.

Program Listing

```
10 GRAPHICS 3+16
20 FOR X=0 TO 39
25 FOR Y=0 TO 23
30 COLOR 65
40 PLOT X,Y
50 FOR T=1 TO 50:NEXT T
60 COLOR 0
70 PLOT X,10
75 NEXT Y
80 NEXT X
90 GOTO 90
```

The second program, below, paints left to right and then downward, leaving the same dark horizontal line across the screen.

Program Listing

```
10 GRAPHICS 3+16
15 FOR Y=0 TO 23
20 FOR X=0 TO 39
30 COLOR 65
40 PLOT X,Y
50 FOR T=1 TO 50:NEXT T
60 COLOR 0
70 PLOT X,10
80 NEXT X
85 NEXT Y
90 GOTO 90
```

98 Flashing Graphics Cursor

Use this flashy little indicator to spot whatever you like on the graphics screen. Change the size of the cursor spot by changing the ranges of X and Y values in lines 1030 and 1040.

Program Listing

```
1000 GRAPHICS 10
1010 FOR L=1 TO 8
1020 COLOR L
1030 FOR X=10 TO 12
1040 FOR Y=10 TO 20
1050 PLOT X,Y
1060 NEXT Y
1070 NEXT X
1080 NEXT L
1090 GOTO 1010
```

99 Boxed Title

Here's a neat, different way to place a box around a program title—or anything else you might like to highlight.

Line 10 selects graphics mode 2. The + 16 makes it full-screen. Lines 20 and 30 position and print the title.

Lines 40 to 80 create the box around the title. Line 100 is a freeze- frame loop, used here so you can review your handywork.

Program Listing

```
10 GRAPHICS 2+16
20 PRINT #6:PRINT #6:PRINT #6
30 PRINT #6;"         TITLE"
40 PLOT 1,1
50 DRAWTO 19,1
60 DRAWTO 19,4
70 DRAWTO 1,4
80 DRAWTO 1,1
100 GOTO 100
```

100 Draw Bar Graphs

Drawing graphs on the video screen are a popular form of communication today. This program establishes a bar graph on the ATARI display.

We have selected the business-like example, shown here, to demonstrate how you go about setting up a bar graph on the TV screen.

Lines 10 to 50 are used to input data needed to be graphed. We have selected annual profits as the subject for our graph.

Line 10 asks for total profits in the year 1978; line 20 for 1979; etc. For the purposes of our graph, the maximum value you can enter will be 13. When the computer asks for the annual-profits data, give it numbers from 1 to 13.

Line 100 establishes full-screen graphics mode number 2. Lines 120 and 130 print the heading.

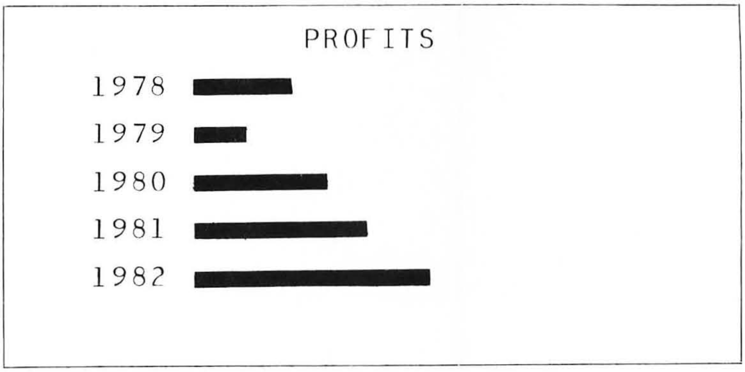
Lines 200 to 230 create the first-year bar on the graph. Lines 300 to 330 create the second-year bar, and so on through line 630 which concludes the last-year bar on the graph.

Line 1000 is a freeze frame to hold the picture on the screen for you to see. Press BREAK to interrupt the freeze-frame loop.

Program Listing

```
10 PRINT "1978 PROFITS:",:INPUT A
20 PRINT "1979 PROFITS:",:INPUT B
30 PRINT "1980 PROFITS:",:INPUT C
40 PRINT "1981 PROFITS:",:INPUT D
50 PRINT "1982 PROFITS:",:INPUT E
100 GRAPHICS 2+16
120 PRINT #6;"          PROFITS"
130 PRINT #6
200 PRINT #6;"1978"
210 PLOT 6,2
220 DRAWTO 6+A,2
230 POSITION 0,4
300 PRINT #6;"1979"
310 PLOT 6,4
320 DRAWTO 6+B,4
330 POSITION 0,6
400 PRINT #6;"1980"
410 PLOT 6,6
420 DRAWTO 6+C,6
430 POSITION 0,8
500 PRINT #6;"1981"
510 PLOT 6,8
520 DRAWTO 6+D,8
530 POSITION 0,10
600 PRINT #6;"1982"
610 PLOT 6,10
620 DRAWTO 6+E,10
630 POSITION 0,12
1000 GOTO 1000
```

Sample Run



```
              PROFITS
    1978
    1979
    1980
    1981
    1982
```

101 Background Cycler

Need a fast way to review the various background colors available on the graphics screen? This program uses graphics mode 10 and cycles thtough the many background colors.

The appropriate background color is POKE'd into memory location 704 at line 1020.

The FOR/NEXT loop in lines 1010 to 1040 causes the cycling through the background-color numbers.

Line 1030 is a time-delay loop, placed there to allow you to study each color briefly before it is replaced by a new color. To add more delay, increase the number 100 in line 1030. To speed things up, decrease the number 100 in line 1030.

Program Listing

```
1000 GRAPHICS 10
1010 FOR L=15 TO 255 STEP 16
1020 POKE 704,L
1030 FOR T=1 TO 100:NEXT T
1040 NEXT L
1050 GOTO 1010
```

Appendix

Appendix A: ATARI BASIC Words

Here is a handy list of all the words in the ATARI version of the BASIC computer language:

ABS	absolute value function
ADR	string memory address function
AND	logical expression true only if both are
ASC	finds ASCII character number
ATN	arctangent
BYE	leave BASIC; write directly on screen
CLOAD	load program from tape
CHR$	converts ASCII number to character
CLOG	base 10 logarithm
CLOSE	close file at end of I/O job
CLR	undimensions strings and arrays
COLOR	select color register for graphics
COM	same as DIM
CONT	continue, after BREAK or STOP
COS	cosine
CSAVE	sends data from computer to tape
DATA	stores info in program line

DEG	switch to trig degrees from radians
DIM	reserves memory for array or string
DOS	displays disk menu
DRAWTO	draws line between points
END	concludes a program run
ENTER	input data or programs
EXP	e to a power
FOR	sets range of FOR/NEXT loop
FRE	shows remaining available memory
GET	input single byte of data, with disk
GOSUB	branch to subroutine
GOTO	branch to a line
GRAPHICS	selects graphics mode
IF	decision maker
INPUT	stops for keyboard input of data
INT	portion of number left of decimal
LEN	number of characters in a string
LET	optional; assigns value to variable
LIST	display contents of program memory
LOAD	send program from disk to computer
LOCATE	stores specific graphics point
LOG	natural logarithm
LPRINT	print on paper with line printer
NEW	erase all program and data memory
NEXT	other half of FOR/NEXT loop
NOT	not true = 1; true = 0
NOTE	used with disk
ON	use with GOTO or GOSUB branches
OPEN	opens a file for I/O
OR	if either true, a 1; 0 if both false
PADDLE	position of game paddle
PEEK	look in one memory location
PLOT	light a single dot on video screen
POINT	for disk operations
POKE	put info in one memory location
POP	abnormal departure from GOSUB loop
POSITION	move to specific video screen point
PRINT	causes output from the computer
PTRIG	status of trigger button on paddle
PUT	output one data byte from computer
RAD	trig info in rads, not degrees
READ	get info from DATA lines

REM	ignore these remarks
RESTORE	allows DATA lines to be re-read
RETURN	back from subroutine to main program
RND	random number between 0 and 1
RUN	execute a program
SAVE	store data or program on disk
SETCOLOR	store color data
SGN	find sign of a number
SIN	sine
SOUND	control pitch, volume, tone
SQR	square root
STATUS	asks state of I/O device
STEP	size of skip in FOR/NEXT loop
STICK	position of stick game controller
STRIG	determines if stick trigger button is pressed
STOP	temporary program run halt
STR$	changes number to character string
THEN	IF true, does what follows
TO	part of FOR/NEXT loop range
TRAP	on input error, jumps to a line
USR	machine-language subroutine
VAL	converts string to a number
XIO	I/O in graphics or disk work

Appendix B: Error Messages

These are the messages your ATARI is trying to send you when it presents an error number. Naturally, it tells you the line number where the error is located.

Number	Message
2	you have used up all available memory
3	the number can't have the value it has
4	you are only allowed 128 different variable names
5	your string is longer than the space you dimensioned
6	you are tryng to READ more DATA than you have available
7	a number is greater than 32767
8	you are trying to put string data in a number variable
9	you have not dimensioned an array or string properly

10	you simply have too many GOSUBs
11	you either have tried to divide by zero or else you want a result too large or too small for the computer to handle
12	you are using GOSUB or GOTO or THEN to jump to a nonexistent line number
13	you have a NEXT but there was no FOR
14	the statement simply is too long or too complex for BASIC to handle
15	again you have tried a NEXT or a RETURN without the necessary FOR or GOSUB
16	there is a RETURN without a GOSUB
17	don't POKE there; try NEW to get out of this mess; if that doesn't work, kill the power and turn it back on and re-enter the program without POKEs
18	you are trying to use an invalid string character
19	there's not enough memory to complete the LOAD
20	you have a device number larger than 7 or equal to zero
21	you are trying to LOAD a no-LOAD file
128	you hit the BREAK key while the machine was doing something
129	the input/output control block already is open
130	you are calling for a nonexistent device
131	you have sent a read command to a write-only device, the printer
132	you can't use that command for that device
133	you forgot to open the file or device
134	that is an illegal device number
135	you are trying to send a write command to a read-only device
136	you have read to the end of the file
137	you are trying to read a record longer than 256 characters
138	the device is turned off, dummy
139	you are getting garbage at a serial port or else you have a bad disk drive
140	input framing error

141	your cursor is out of range for that mode
142	serial bus data frame overrun
143	serial bus data frame checksum error
144	you are trying to write on a write-protected disk
145	something's wrong with what you just wrote
146	that function is not implemented
147	you don't have enough memory available for the graphics mode you selected
160	that's a drive number error
161	you have too many files OPENed
162	the disk is full
163	you've blown the whole bit
164	file numbers don't match, the disk links are messed up
165	that's a file name error
166	there's an error in the POINT data length
167	that file is locked
168	that command is invalid
169	the directory is full
170	that file cannot be located anywhere
171	the POINT is invalid

ATARI Computer Book II

101 ATARI Computer Programming Tips & Tricks

by Alan North

Here's a giant collection of practical, useful, efficient programming techniques and operating shortcuts, for the ATARI model 400, 600 and 800 personal computers, right out of a master programmer's notebook.

Loaded with hints, secrets, tips, tricks and easy-to-follow instructions, this book shows you how to handle routine programming chores on the Atari more quickly, do special effects, make your computer work for you—faster and more efficiently.

Each of the 101 computer programming tips in this book features a complete ready-to-run program. Each will run immediately, as you find it in this book, or it easily can be included in a larger set of instructions to your computer. All 101 programs have been tested thoroughly on the ATARI 400/800 Computer and are ready to type in and run.

Learn insider's how-to secrets for using the special, exciting BASIC words READ, DATA, RESTORE, INPUT, LEN, GOSUB, RETURN, VAL, ASC, CHR$, PRINT, DIM, FOR, NEXT, IF, THEN, DRAWTO, SETCOLOR, and many more.

Sections in this book include a detailed *Introduction; Fun and Games; Text on Text; Gee Whiz; Number Crunching; Money Matters; Colorful Graphics;* and a handy *Appendix.*

You will find special techniques for graphics; colorful program titles and billboards; coin toss; dice throw; secret message; bell ringer; mystery clues; code groups; 60-second timer; sorting; ordering; searching; averaging; passwords; sentence writer; creating tables; memory tester; exam sorter; rounding off; random numbers; shopper's friend; interest on money; car payments; wages 7 hours; percentage off and mark up; invoicing; inventory counter; lists; two dozen exciting graphics programs; and many more.

You'll learn what to do, how to do it, when to make changes and when not to, all from this info-packed ATARI computer programmer's handbook of tips and tricks.

ARCsoft Publishers
Woodsboro, Maryland

ISBN 0-86668-022-5