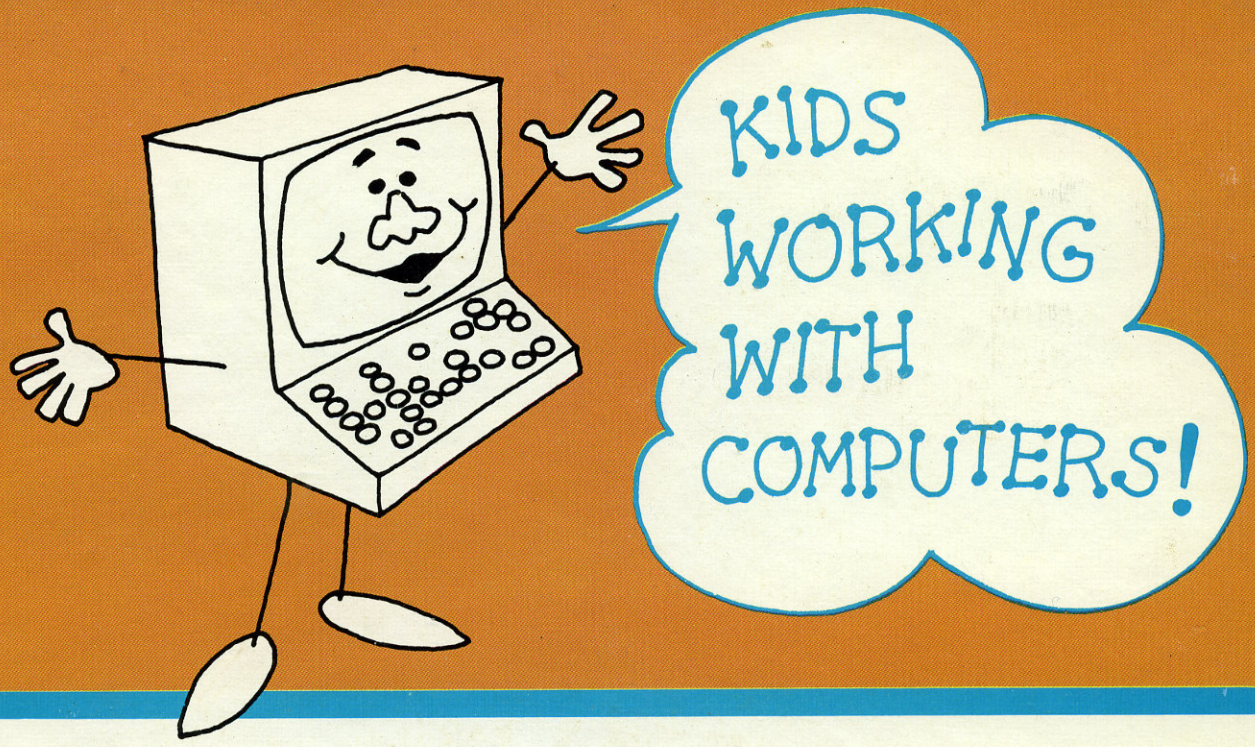
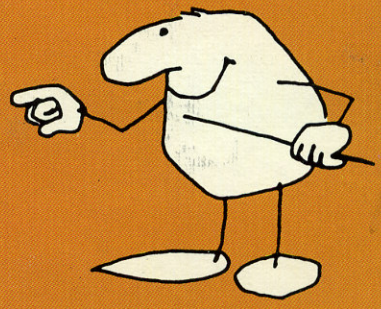
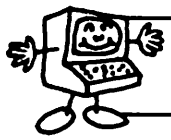


THE TEXAS INSTRUMENTS BASIC MANUAL

Thomas Milton Kemnitz
Lynne Mass





Kids Working with Computers

THE TEXAS INSTRUMENTS BASIC MANUAL

Thomas Milton Kemnitz **Lynne Mass**

CHILDRENS PRESS™
CHICAGO

Library of Congress Cataloging in Publication Data

Kemnitz, Thomas Milton.
The Texas Instruments BASIC manual.

(Kids working with computers)
Includes index.
Summary: Explains how to use the computer language BASIC to write fundamental programs for the TI 99/4A computer.
1. TI 99/4A (Computer)—Programming—Juvenile literature. 2. BASIC (Computer program language)—Juvenile literature. [1. TI 99/4A (Computer)—Programming. 2. BASIC (Computer program language) 3. Programming (Computers) 4. Computers] I. Mass, Lynne. II. Title. III. Series.

QA76.8.T133K46 1985 001.64'2 85-433
ISBN 0-516-08428-3

Library bound edition published by Childrens Press under license from Trillium Press.

1 2 3 4 5 6 7 8 9 10 R 94 93 92 91 90 89 88 87 86 85

CONTENTS

1

Meet the Computer

Let's get to know our computer. To start:

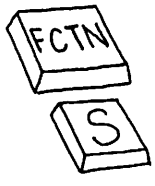
1. Plug in the computer and turn on the monitor.
2. When the screen tells you to do so, press any key to begin.
3. Press 1 for TI Basic.
4. The screen will show:

   ```
   TI BASIC READY
   ```

   The blinking black box is called the **cursor.**

5. Type **CALL CLEAR** and push **ENTER.**

You are now ready for your first **program.**



If you make a mistake, hold down **FCTN,** push S and then correct the error.

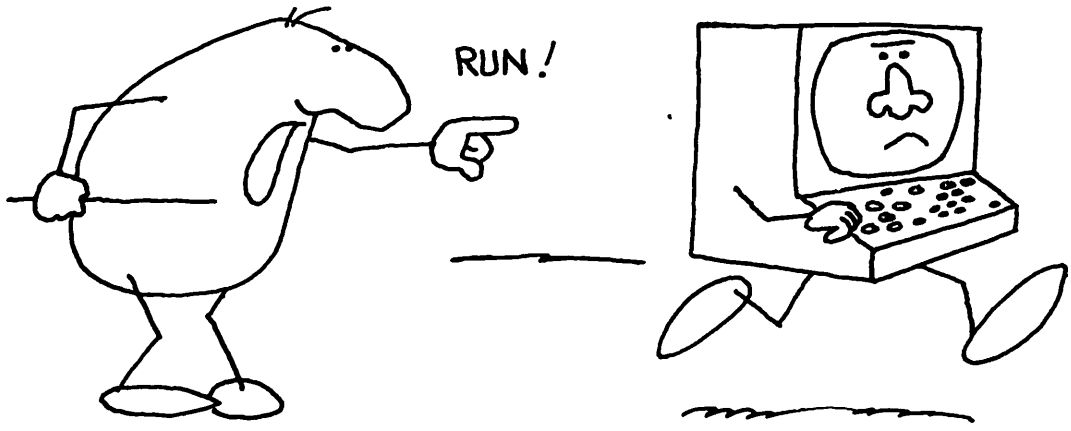
Now type exactly what you see. Always remember to push ENTER at the end of each line.

```
10 PRINT ''HELLO, MY NAME IS (type your name).''
20 PRINT ''I AM LEARNING TO RUN THE TI 99/4A.''
30 PRINT ''MY TEACHER IS (type in teacher's
   name).''
40 PRINT ''(make up something about yourself).''
```

Now that you have typed your program, type **LIST** and push ENTER. What happens?

Try typing **RUN**. Did you remember to push ENTER?

What happens if you type **NEW** [ENTER] and then RUN [ENTER]?



REVIEW

What do these words do:

CALL CLEAR

LIST

RUN

NEW

Why are they called **commands?**

2

Do I Need Quotation Marks?

Type the following program:

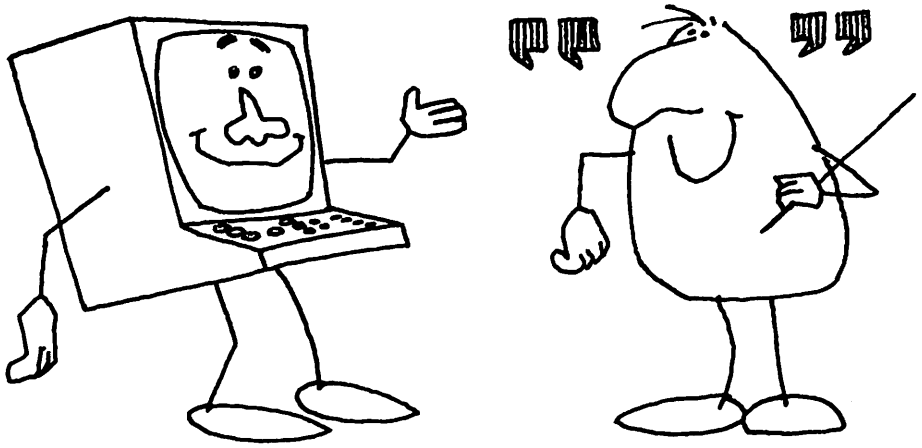
```
10 PRINT ''HI, MOM.''
20 PRINT ''HI, DAD.''
```

Don't forget LIST and RUN.

Type NEW. Now try the same words without quotation marks:

```
10 PRINT HI, MOM.
20 PRINT HI, DAD.
```

How are these two programs different?



3

The Disappearing Act

Program the computer with:

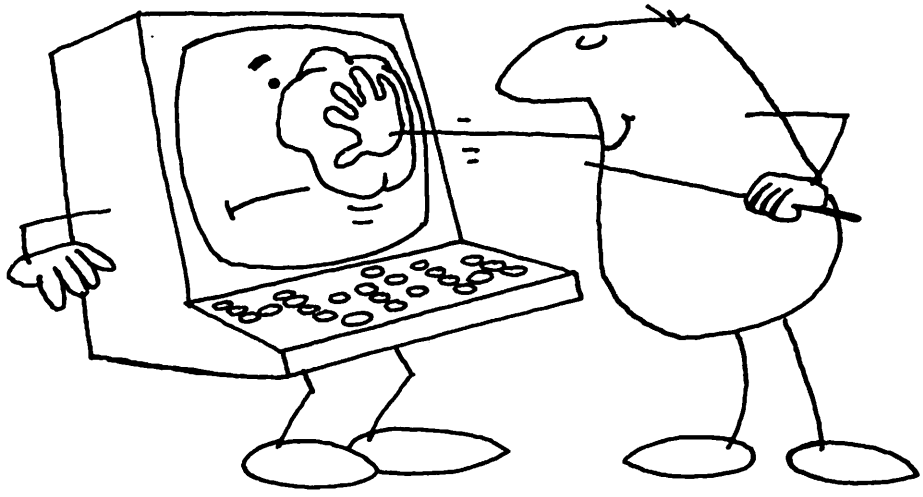
```
10 PRINT ''RUBBER BABY BUGGY BUMPERS.''
```

Clear the screen by typing CALL CLEAR. Does this erase your program? How can you test this? You can see what is in the **memory** by typing LIST.

Now type in these commands:

```
NEW
CALL CLEAR
```

Does this clear the screen also? Is your program erased?



4

Computer Loops

Type the following program and RUN it:

```
10 PRINT ''HELP—MY COMPUTER WENT CRAZY.''
20 GOTO 10
```

When you have seen enough, hold down FCTN and press 4. This is called a **LOOP** and is controlled by the **GOTO** command.

Now try this program:

```
10 FOR X = 1 TO 5
20 PRINT ''HELP—MY COMPUTER WENT CRAZY.''
30 NEXT X
40 PRINT ''NO, I HAVE IT UNDER CONTROL.''
```

Lines 10 through 30 of this program are called a **FOR/NEXT** loop. The statement in line 10 says, "I am going to do something five times." The statement in line 20 tells what will be done. The statement in line 30 sends the computer back to the counter.

Retype line 10:

```
10 FOR X = 1 TO 12
```

This will erase what was there before, just as it would on a tape recorder.

What do you think will happen after you type the command RUN?

Try typing it again with no space between the X and the = sign. RUN it. Did that make a difference?

Try it with no space between 1 and TO. Did that make a difference?

Try it with no space between = and 1. Did that make a difference?

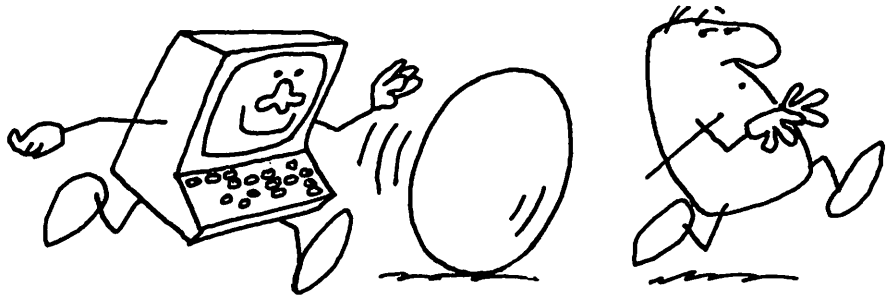
Try it with no space between FOR and X. Did that make a difference?

Try it with no space between TO and 12. Did that make a difference?

Just for Fun

Try this program:

```
10 FOR B = 1 TO 588
20 PRINT ''A'';
30 NEXT B
40 PRINT ''STOP!'': ''PHEW!''
```



5

The Egg Timer

Type the following program:

```
 2 CALL CLEAR
10 PRINT ''WAIT RIGHT HERE.''
20 FOR X = 1 TO 3000
30 NEXT X
40 PRINT ''YOUR TIME IS UP.''
```

Did time go by between the printing of "WAIT RIGHT HERE" and "YOUR TIME IS UP"? How much time went by?

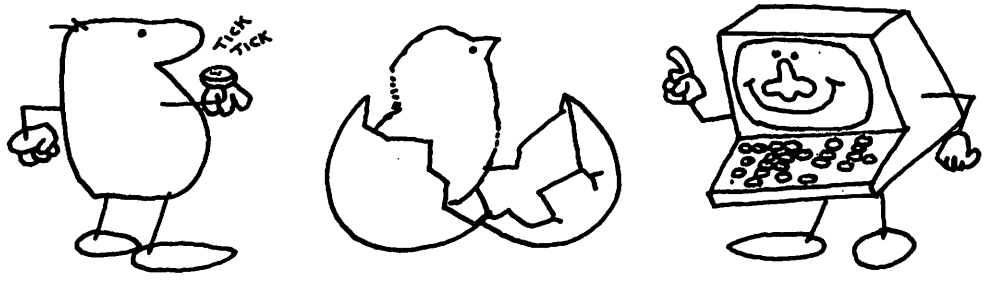
Now change line 20 as follows and RUN it:

```
20 FOR X = 1 TO 10000
```

This time change it to:

```
20 FOR X = 1 TO 2000
```

How are the two times different? Which is longer? How much longer?



6

Boss Around Your Computer

Type the following program:

```
 2 CALL CLEAR
10 FOR X = 1 TO 50
20 PRINT ''NOW I AM PROGRAMMING.''
30 PRINT ''I AM STILL AT IT.''
40 PRINT ''AND I CAN'T STOP.''
50 NEXT X
```

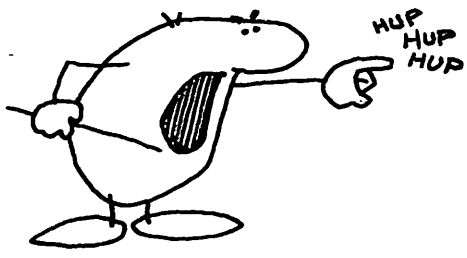
What happens? How many times do you think you told the computer to write lines 20, 30, and 40?

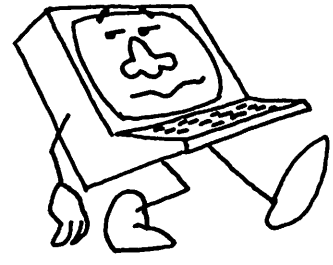
Keep the same program and add:

```
45 PRINT: PRINT
```

Does the program look different?

Did you have to retype the whole program in order to add a line?





7

Line Order

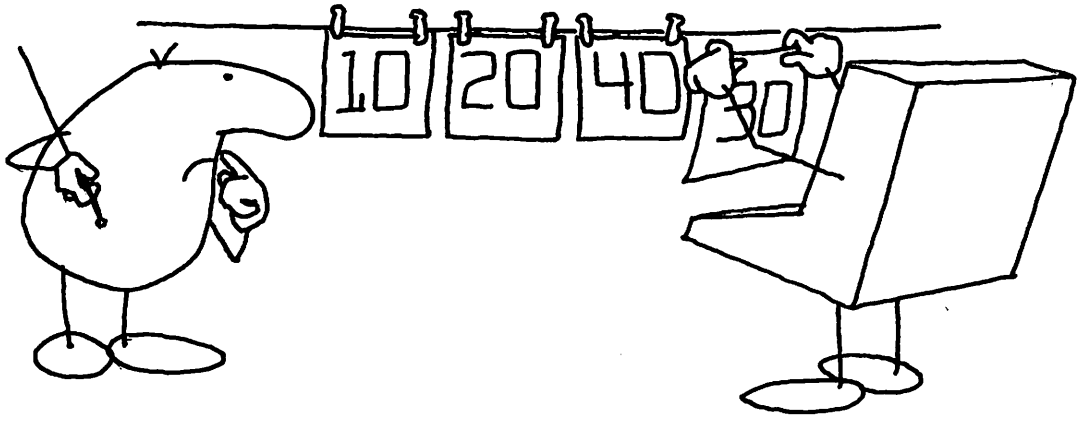
Type the following program:

```
10 PRINT ''I AM USING THE COMPUTER.''
20 PRINT ''AND I DON'T WANT TO STOP.''
 5 PRINT ''NOW I AM PROGRAMMING.''
```

Now LIST the program. What did the computer do to make things easier for you?

Can you figure out a way to add more lines between the beginning and the end?

What do you think would happen if you gave two lines the same number? Try it to find out.



8

Editing Mode

The microcomputer has four modes:

1. **immediate**
2. **programming**
3. **execution**
4. **editing**

In the immediate mode, you give the computer a command and it does it. You do not use **line numbers.** For example, type the following and push ENTER:

```
PRINT 5+7+3
```

Your computer should have answered 15 immediately.

In the programming mode, you also use line numbers. The computer just stores your instructions.

In the execution mode, you tell the comuter to RUN, LIST PRINT, etc., and it does whatever you command.

In the editing mode, you can change a line or more very easily. Suppose you had typed in and tried to RUN these lines:

```
 5 CALL DEAR
10 PLINT ''LOVE''
```

The computer will tell you:

```
BAD NAME IN 5.
```

Type in:

```
EDIT 5
```

This will show you line 5 and allow you to change it. The cursor will blink over the C in CALL. To move it forward, hold down FCTN and D until you get to the D in DEAR. Now you can change DEAR to CLEAR, and then ENTER and RUN.

Next the computer will tell you:

```
INCORRECT STATEMENT IN 10
```

You can now type in EDIT 10 and make your change in line 10.

9

String a Design

A **string**—a **variable** and a $—is one or more **characters.** A string can be interesting. For example, try typing in this:

```
 2 CALL CLEAR
 5 A$ = ''GOOD MORNING''
10 PRINT A$
```

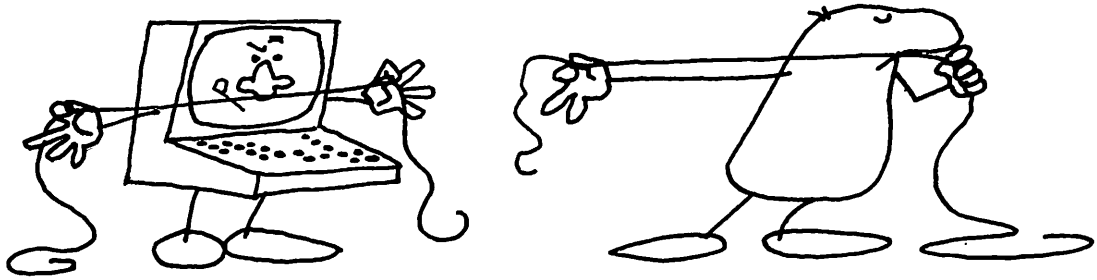
What happens?

You can even get a number value for a string. LEN is a code that will tell you how many characters—in this case, letters and spaces—are in the string.

Try this:

```
10 PRINT LEN (A$), LEN (''YES'')
```

What do the 12 and 3 represent? *(CLUE: LEN is a computer word for "length.")*



Now try this:

```
10 PRINT SEG$ (A$,1,4)
```

What do you get? It's good if you see GOOD. How do you think you could get the computer to print the word MORNING on the screen?

Here is another one for you to try:

```
10 FOR N = 1 TO LEN (A$)
20 PRINT SEG$ (A$,1,N)
30 NEXT N
```

Isn't that cute?

Why does this happen? Since A$ has twelve characters, this loop will be done twelve times, with N = 1,2,3,. . .11,12. The first time, only the first character will be printed (and N = 1). The second time, the first two characters will be printed (and N = 2), and so on.

10

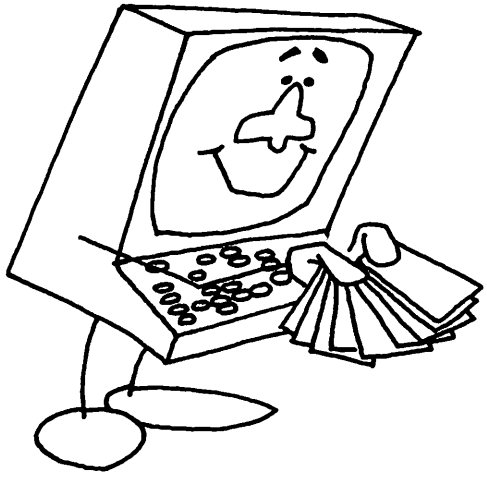
Pick and Choose

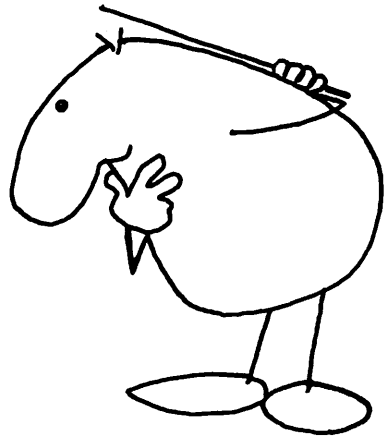
Here is a program to try with a friend:

```
 2 CALL CLEAR
 5 PRINT ''TYPE IN YOUR AGE.''
10 INPUT A
12 CALL CLEAR
15 PRINT ''YOUR AGE IS''
20 PRINT A
   RUN
```

INPUT asks your friend to type in a number while the program is running.

When you are finished with this program, type the word NEW.





Now that you have learned to construct a program with an INPUT, you can take advantage of computer logic to construct games. The computer is able to decide if two numbers are equal or if one number is greater than (>) or less than (<) another. If they are equal, you can tell the computer to do one thing; if they are unequal, you can tell it to do something else.

Try this simple game:

```
 2 CALL CLEAR
 5 PRINT ''PICK A NUMBER. TYPE IT IN. THE CHOICES
   ARE 1,2, OR 3.''
10 INPUT N
20 IF N< >3 THEN 100
30 IF N = 3 THEN 40
40 PRINT ''YOU ARE CORRECT.''
50 END
100 PRINT ''NO, TRY AGAIN.''
110 GOTO 10
```

The statements you used are called **IF-THEN** statements. When two symbols are used together, they have these meanings:

```
< =    is less than or equal to
> =    is greater than or equal to
< >    is not equal to (it is less than or more than)
```

11

Give Me a T!

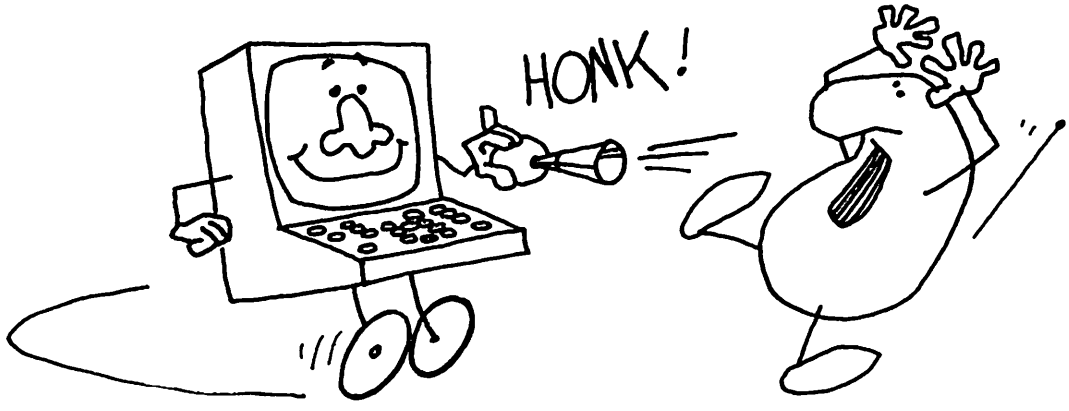
Use the "Pick a Number" program from the last lesson. Change line 30 to:

```
30 IF N = 3 THEN 200
```

Now we are going to add sound. **CALL SOUND** is the command that we will use. Type this program:

```
200 ZAP = 110
210 FOR BEEPS = 1 TO 10
220 CALL SOUND (-500, ZAP, 1)
230 ZAP = ZAP + 110
240 NEXT BEEPS
250 GOTO 40
```

Try typing HONKS instead of BEEPS in lines 210 and 240. What happened?



12

Math Magic

Our computer can do many tricks with numbers. Let's get used to some easy ones first. Try this:

```
PRINT 3 + 4
```

The TI-99/4A can do six different arithmetic operations:

1. addition (+)	`PRINT 5 + 7`
2. subtraction (−)	`PRINT 6 - 2`
3. multiplication (*)	`PRINT 7*8`
4. division (/)	`PRINT 63/7`
fractions (/)	`PRINT 3/2`
5. exponentiation (4^5, for example)	`PRINT 4*4*4*4*4` or `PRINT 4^5`
6. square root ($\sqrt{16}$, for example)	`PRINT SQR(16)`

Make up some math problems of your own.

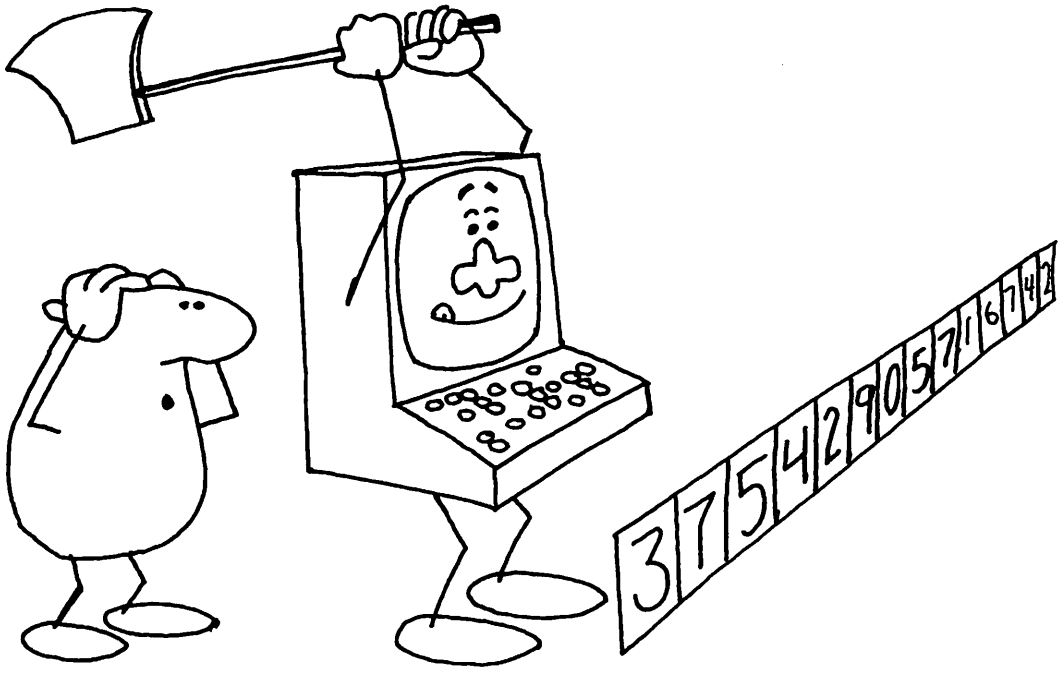
13

Some Strange Numbers

Type the following:

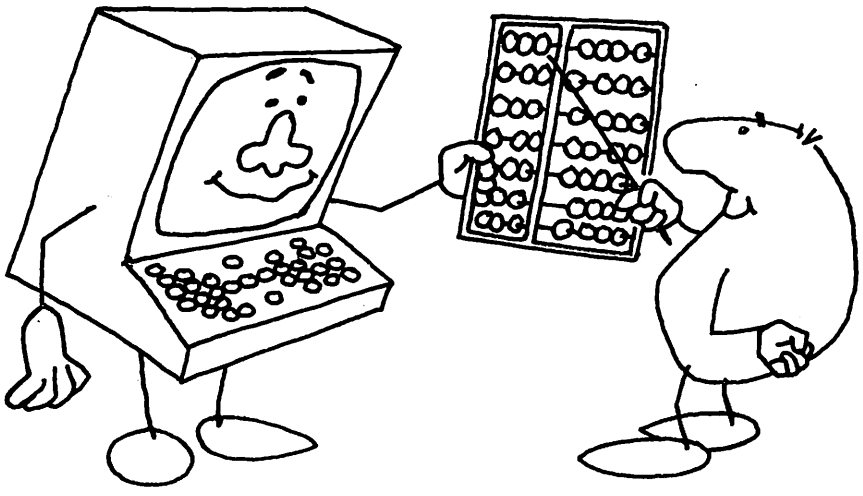
```
PRINT .375429057167 42
```

What do you see? The computer rounds off. To what number of places did it round off? Is that always true? This time try 375.42905716742. What does the computer do to it?



To get an idea of what computers do with numbers in expressions, do these in your head or on paper first, and then do them on the computer.

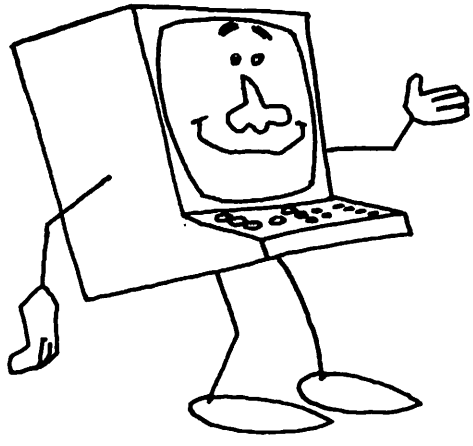
1. 3+1 * 2
2. 2+4/2
3. 3+2+6
4. 7−3+2
5. 3+2^2+4 * 2
6. 3+2^3−2 * 2+5
7. 3+6−2+4^2
8. 8+4/3−3
9. SQR (9)−SQR (5)
10. 5 * 8+SQR (9)
11. 3−2 * 25
12. 3^2+7^2+2^2−3
13. SQR (25)−SQR (16) * 3^2
14. 4+2^2/2^3−1
15. 7/3+4
16. SQR (81) * 11+2

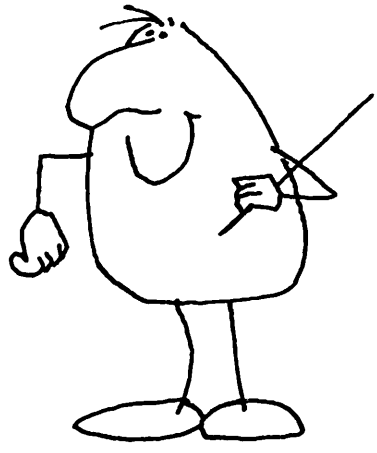


The computer follows this order for carrying out math operations:

1. All operations within parentheses. If one set is inside another, it does the inside set first.
2. Exponents and square roots.
3. Multiplication and division.
4. Addition and subtraction.

Now, using parentheses, write the expressions on page 22 more clearly.





14

Repeat

One advantage of computers is their ability to perform repetitive tasks quickly and without getting bored. Follow these programs involving square roots and see the shortcuts that develop.

Try this program:

```
 10 PRINT 1, SQR (1)
 20 PRINT 2, SQR (2)
 30 PRINT 3, SQR (3)
 40 PRINT 4, SQR (4)
 50 PRINT 5, SQR (5)
 60 PRINT 6, SQR (6)
 70 PRINT 7, SQR (7)
 80 PRINT 8, SQR (8)
 90 PRINT 9, SQR (9)
100 PRINT 10, SQR (10)
```

SQR is computer language for "square root" and SQR(X) is the way you write "square root of a number," with X standing for the number.

This is a shortened form of the same program. Try it. Type NEW and

```
 2 CALL CLEAR
10 N = 1
20 PRINT N, SQR (N)
30 N = N + 1
40 IF N < = 10 THEN 20
```

How does this four-line program compare with the ten-line program you ran first? Using the same loop, we can shorten the program even more. Type NEW and

```
10 FOR N = 1 TO 10
20 PRINT N, SQR (N)
30 NEXT N
```

Try to figure out a program for printing a table of square roots for only even integers from 10 to 20.

15

At Random

The computer can print out numbers that are not predictable. The numbers are formed randomly and have the code **RND**, which stands for "random numbers." Random numbers form the basis for many kinds of computer games.

To see what (RND + 0) does, try this (to get random numbers on the TI-99/4A, you have to have the word **RANDOMIZE** in the program):

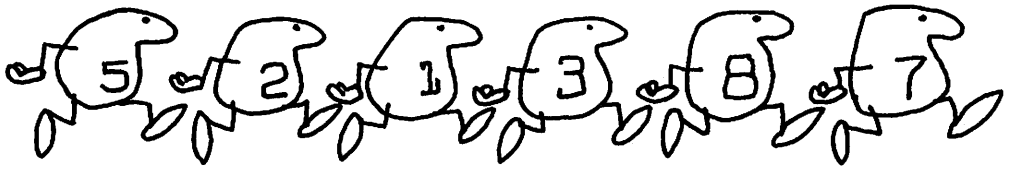
```
10 RANDOMIZE
20 PRINT (RND + 0)
```

What did you get?

Try it again. Did you get the same number? Are the numbers whole numbers or decimals?

How can you get a number >1 (greater than one)? Try this:

```
20 PRINT 10 * (RND + 0)
```



To get rid of the numbers after the decimal, try this (**INT** is a code for INTEGER):

```
20 PRINT INT (10* (RND+0))
```

What happens if you multiply by 100? Experiment! Change (RND + 0) to (RND + other things).

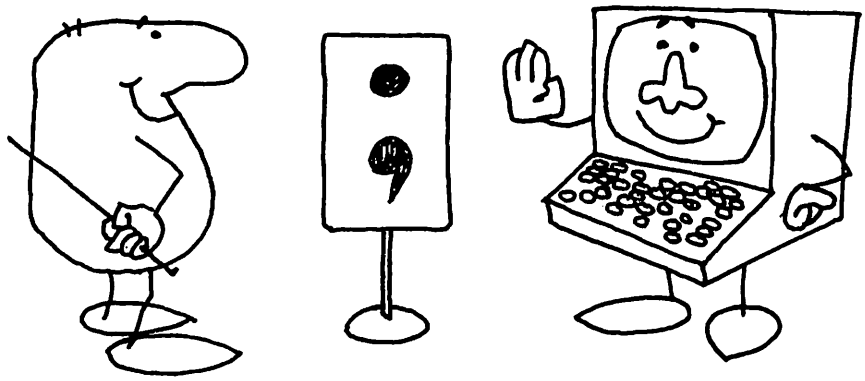
Try adding these lines. LIST and predict what will happen before you RUN it.

```
 2 CALL CLEAR
15 FOR N= 1 TO 10
30 NEXT N
```

Change lines 15 and 20 to:

```
15 FOR N= 1 TO 90
20 PRINT INT (10 * (RND + 0));
```

What happens? What does the semicolon (;) do?



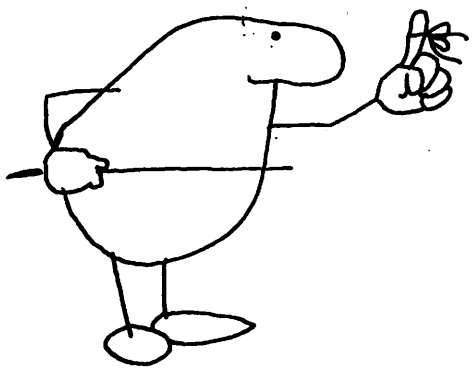
16

REMark

Now that you are becoming a real programmer, you should know about **REM.** REM is short for **REMARK.** Any line in a program that is headed by REM will not RUN, but it will LIST. So, a REMARK is like a note right in the middle of the program. Programmers use REM to:

1. Name programs.
2. Remind themselves of bugs or changes.
3. Date programs.
4. Note programming language.
5. Identify variables.
6. Remind themselves or tell others what they expect a part of the program to do.

From now on we'll use REM in some of our programs.



17

Make a Guessing Game

Here is a number game using RND and other things you have learned:

```
  2 CALL CLEAR
  5 REM GUESSING GAME
  6 REM ENTERED BY (your name)
  7 REM FROM KIDS WORKING WITH COMPUTERS
  8 REM COPYRIGHT 1983 TRILLIUM PRESS
  9 RANDOMIZE
 10 G = 1 + INT (100 * (RND + 0))
 15 PRINT ''GUESS A NUMBER FROM 1 TO 100.''
 20 LET C = 0
 30 INPUT A
 50 C = C + 1
 60 IF A = G THEN 130
 70 IF A < G THEN 100
 80 PRINT ''TOO HIGH''
 90 GOTO 110
100 PRINT ''TOO LOW''
110 GOTO 30
130 ZAP = 110
131 FOR BEEPS = 1 TO 10
132 CALL SOUND (-500, ZAP, 1)
133 ZAP = ZAP + 110
134 NEXT BEEPS
135 PRINT ''VERY GOOD, YOU GUESSED IT!''
140 PRINT ''YOU TOOK'';C;''GUESSES.''
```

Make up your own guessing game. Save it on a disk or on the printer.

18

READ/DATA

In this program you try to guess one of the numbers in lines 110 and 120 (these lines are called **DATA** statements). Try to figure out the purpose of **READ.**

```
 2 CALL CLEAR
 5 REM PICK A NUMBER
 6 REM ENTERED BY (your name)
 7 REM DATE (type today's date)
10 PRINT ''PICK A NUMBER''
20 INPUT G
30 READ D
40 IF D = -9999 THEN 80
50 IF D < > G THEN 30
55 ZAP = 110
56 FOR BEEPS = 1 TO 10
57 CALL SOUND (-500,ZAP,1)
58 ZAP = ZAP + 110
59 NEXT BEEPS
60 PRINT ''YOU ARE CORRECT''
70 END
80 PRINT ''WRONG, TRY AGAIN''
90 RESTORE
100 GOTO 10
110 DATA 9,15,18,-60,242,0,80
120 DATA 78,4,45,25,-22,-9999
```

DATA statements may be placed anywhere in the program. DATA is information read by the computer when a READ statement appears.

Now make up another guessing game.

19

Colorful Colors

To get color and graphics (pictures made up of dots of color) on the screen, use **CALL COLOR.** Each color has a code number. For instance, dark red is 7.

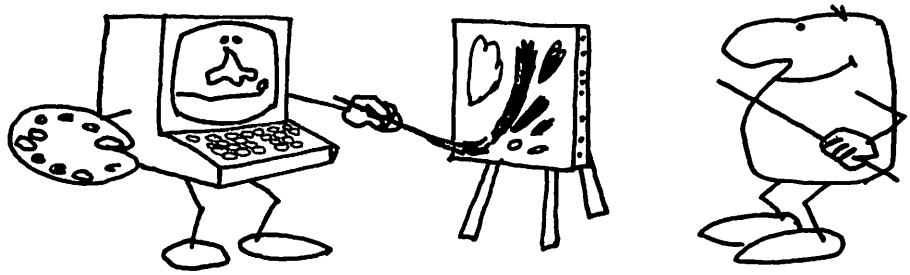
You also have to tell the computer where you want the color. For instance, **CALL SCREEN** turns the background a color without changing the letters. Try:

```
10 CALL SCREEN (7)
20 GOTO 10
```

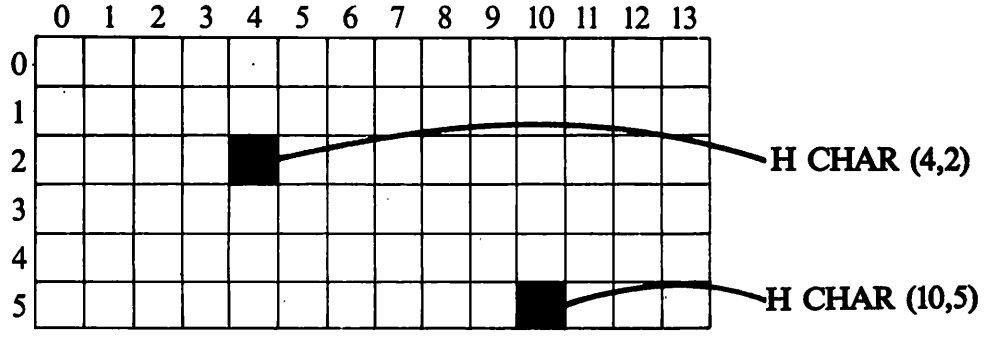
Would you like a blinking screen that changes color? Try adding this:

```
 5 CALL SCREEN (5)
 6 FOR X = 1 TO 1000
 7 NEXT X
11 FOR X = 1 TO 1000
12 NEXT X
20 GOTO 5
```

When you've seen enough, hold down FCTN and press 4.



You can locate specific spots on a screen by using a grid like this:



To command the computer where to put a character, you look at the address—the place where the square is located—and use **HCHAR** or **VCHAR.** If you are going to put in only one character, it does not matter which you use. But, if you want to put a lot of characters across in a line, use HCHAR (which stands for "horizontal characters"). If you want a lot of characters up and down, use VCHAR (which stands for "vertical characters").

Try this:

```
10 CALL CLEAR
20 CALL VCHAR (10,15,42)
```

10 means 10 down on the Y-axis;
15 means 15 across on the X-axis.
*42 is the number for an **

Do you want to see it blink? Add:

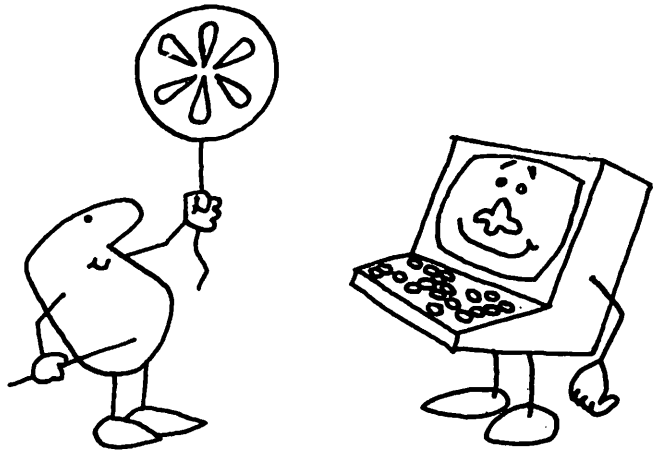
```
30 GOTO 10
```

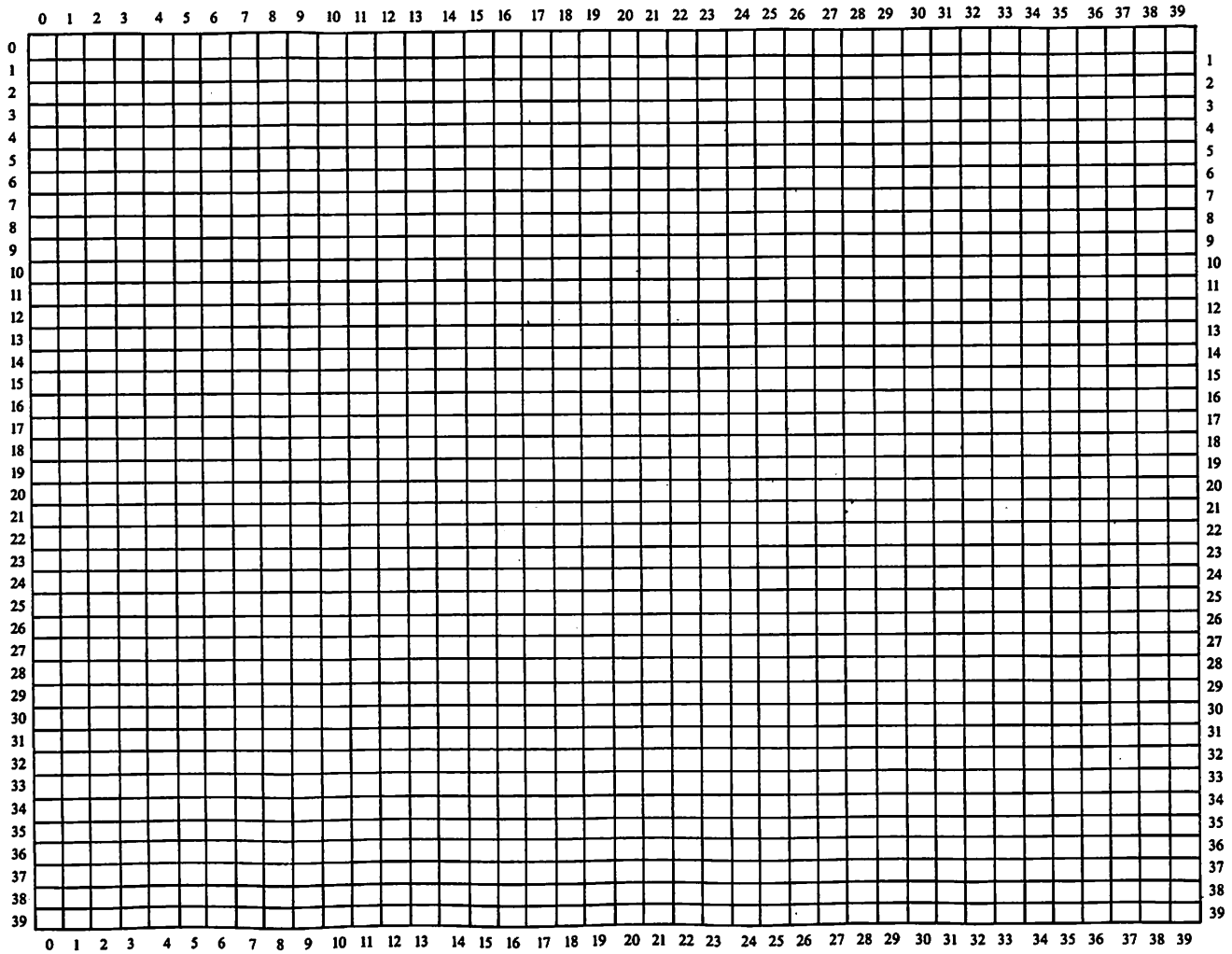
Look at the full grid on page 34. Can you write a program that would put an asterisk (*) in the middle of the screen and make it blink once per second? Try it.

Here are the colors and their numbers:

1. transparent	7. dark red	12. light yellow
2. black	8. cyan	13. dark green
3. medium green	9. medium red	14. magenta
4. light green	10. light red	15. gray
5. dark blue	11. dark yellow	16. white
6. light blue		

Practice plotting designs using the full grid.




0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

20

Going Straight

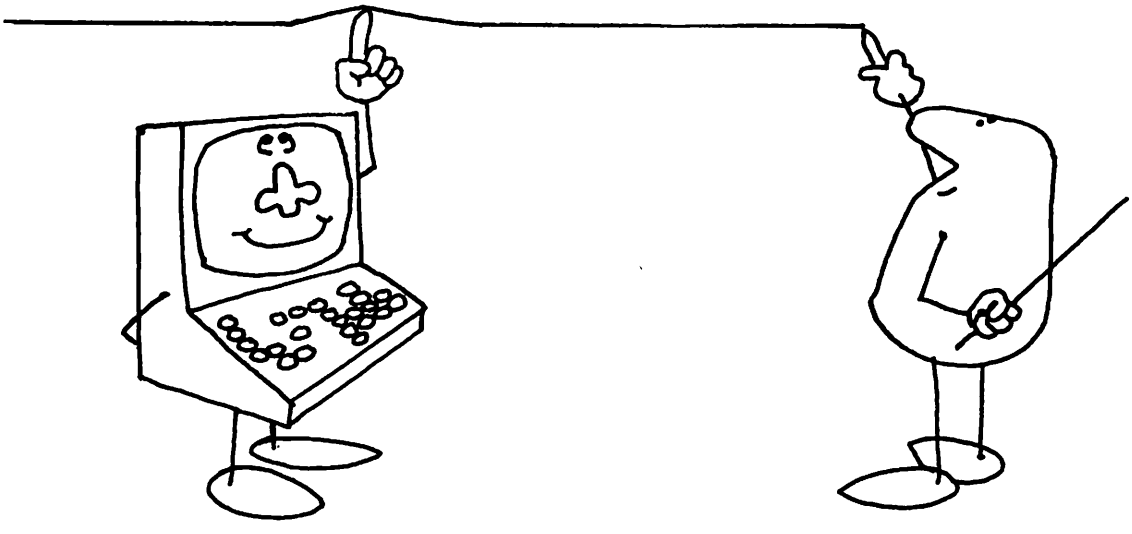
Suppose you want a straight line. There has to be an easier way than putting one little asterisk after another. And there is.

If you want a horizontal line, use CALL HCHAR:

```
 2 REM GOING STRAIGHT
 5 CALL CLEAR
10 CALL HCHAR (8,3,42,17)
```

8 means 8 down on the Y-axis. 3 means begin at 3 on the X-axis. 42 means asterisk. 17 means end at 17 on the X-axis

To draw a vertical line, edit line 10, changing the H to V.



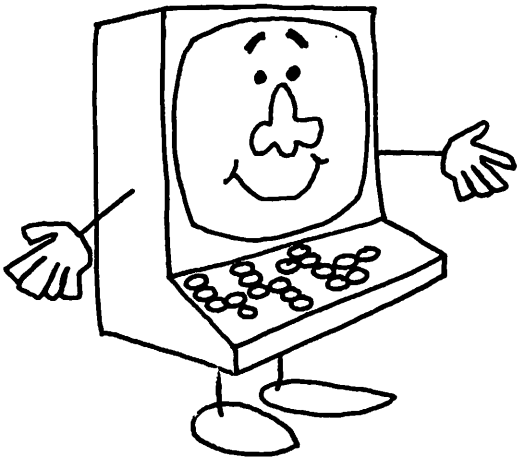
21

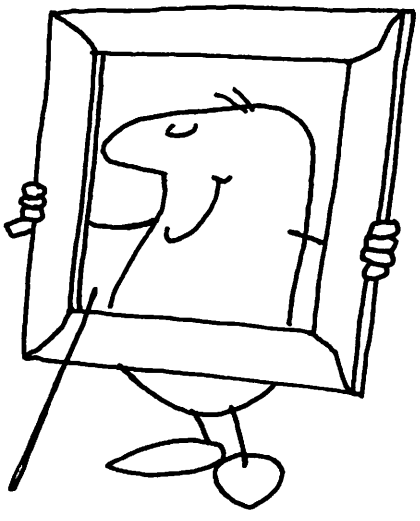
Getting Framed

Now let's try drawing more than one line. In fact, let's make a frame. Here's how:

```
  5 CALL CLEAR
 10 CALL VCHAR (1,3,42,22)
 20 GOTO 100
100 CALL HCHAR (1,3,42,28)
110 GOTO 200
200 CALL VCHAR (1,30,42,22)
210 GOTO 300
300 CALL HCHAR (23,3,42,28)
310 GOTO 10
```

Now, can you figure out how to write your name in the frame? Try it!





22

GOSUB

GOSUB is a command that allows us to establish a subroutine that we can use many times in a program. GOSUB must always be used with RETURN.

Suppose we wanted to add sound at the end of each line in drawing our frames. GOSUB will help us do it. Here's how. Type in this program and RUN it:

```
 15 GOSUB 800
105 GOSUB 800
205 GOSUB 800
305 GOSUB 800
800 REM SOUND SUBROUTINE
810 ZAP = 110
820 FOR BEEPS = 1 TO 10
830 CALL SOUND (-500, ZAP,1)
840 ZAP = ZAP + 110
850 NEXT BEEPS
860 RETURN
```

What happens if you change line 310 to:

```
310 GOTO 5
```

Try changing it this way:

```
310 GOTO 310
```

What happens?

23

TRACE

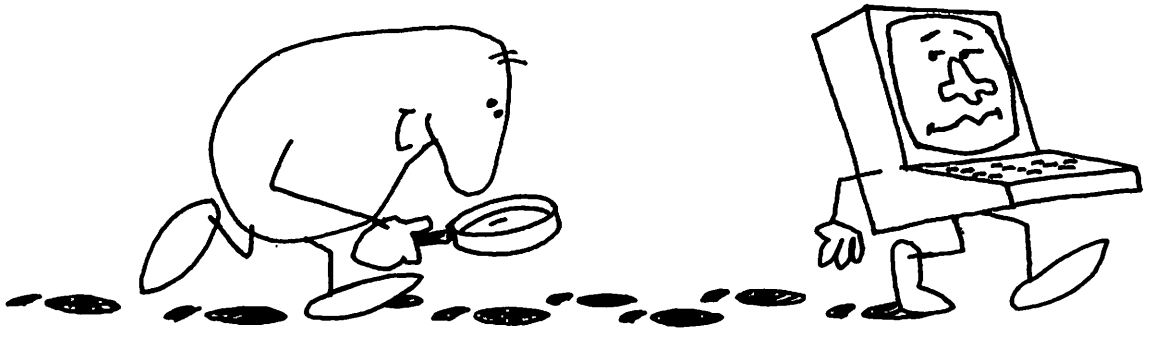
When you have subroutines, your programs can get very complicated. However, there is a way to follow all the steps that you have directed the computer to take, in the order that it takes them. To see this, change the program we used in the last lesson by typing in:

```
310 END
```

Now type **TRACE.** Press ENTER and then RUN.

All the numbers below the line are necessary for the computer to draw the last line and produce the music. Why do you suppose we see 830, 840, and 850 ten times?

When you want to get rid of TRACE, just type the command **UNTRACE.**



24

Graphics Characters

So far in our graphics programs, we have used the asterisk. But the TI-99/4A has many other available characters. Let's see some.

Try entering the program we used in Lesson 21, "Getting Framed." This time instead of 42, the ASCII code for *, use the following and RUN the program:

in line 10, use 30 (the cursor)
in line 100, use 45 (the minus sign)
in line 200, use 33 (the exclamation point)

What is happening in the corners? You can fix that. Can you figure out how? If you said by shortening the lines, you're right. Here is one way:

```
 10 CALL VCHAR (2,3,30,21)
100 CALL HCHAR (1,3,45,28)
200 ALL VCHAR (2,30,33,21)
300 CALL HCHAR (23,3,95,28)
```

The chart on page 40 shows the full set of ASCII characters and numbers.

ASCII Code

32		(space)	55	7	
33	!	(exclamation point)	56	8	
34	"	(quote)	57	9	
35	#	(number sign)	58	:	(colon)
36	$	(dollar)	59	;	(semicolon)
37	%	(percent)	60	<	(less than)
38	&	(ampersand)	61	=	(equals)
39	'	(apostrophe)	62	>	(greater than)
40	(	(open parenthesis)	63	?	(question mark)
41	)	(close parenthesis)	64	@	(at sign)
42	*	(asterisk)	65-90	A-Z	(taller letters)
43	+	(plus)	91	[	(open bracket)
44	,	(comma)	92	\	(reverse slant)
45	-	(minus)	93	]	(close bracket)
46	.	(period)	94	^	(exponentiation)
47	/	(slash)	95	_	(underline)
48	0		96	`	(grave)
49	1		97-122	A-Z	(shorter letters)
50	2		123	{	(left brace)
51	3		124	|	
52	4		125	}	(right brace)
53	5		126	~	(tilde)
54	6		127	DEL	(appears on the screen as a blank)

25

CALL COLOR

CALL COLOR is a command that gives you control of three sets of colors:

screen

foreground

background

For example, CALL COLOR (1,2,6) would give you a transparent screen with a black foreground (letters) and a blue background.

Try it so a friend can input a name:

```
 2 REM NAME IN LIGHTS
 5 CALL CLEAR
10 INPUT ''YOUR NAME? '':A$
20 CALL CLEAR
30 PRINT A$
40 CALL COLOR (1,2,6)
50 GOTO 50
```

26

TAB

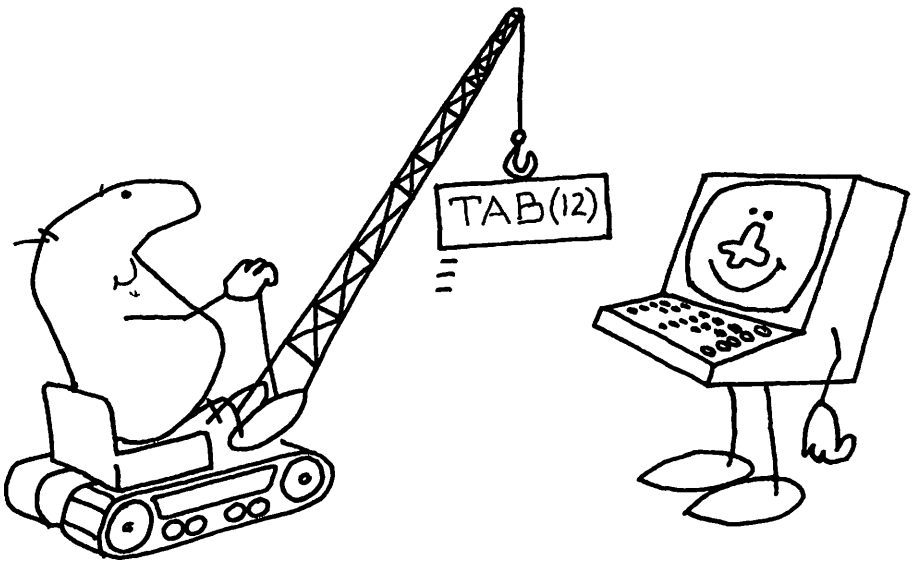
TAB is a command that allows you to position words on the screen. You activate TAB by typing TAB with a number in parentheses, like this:

```
PRINT TAB(12); A$
```

Let's see how it works. Using the last lesson, try changing line 30 to:

```
30 PRINT TAB(12); A$
```

What happened?



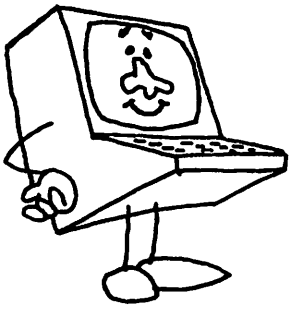
27

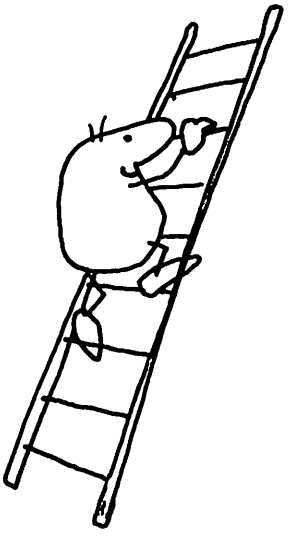
Just for Review

Try making your name climb up the center of a dark red screen with a light blue background, giving it a few second's delay after each line. See if you can do it without looking at the program below.

Here's one way of making it happen. Compare your way with this program:

```
 2 REM CLIMBING NAME
 5 CALL CLEAR
10 INPUT ''YOUR NAME? '':A$
20 CALL CLEAR
30 PRINT TAB(12); A$
35 CALL SCREEN (7)
40 CALL COLOR (1,2,6)
50 FOR X = 1 TO 1000
60 NEXT X
70 GOTO 30
```





GLOSSARY

CALL CHAR A command to use numbers to define characters; 32 to 127 are ASCII, and 128 to 159 are available for additional characters.

CALL CLEAR A command to clear the screen.

CALL COLOR A command to control the colors in a program.

CALL HCHAR A command to control horizontal characters.

CALL SCREEN A command that enables the user to change screen color.

CALL SOUND A command that enables the user to use the sound capabilities of the computer.

CALL VCHAR A command to control vertical characters.

CHARACTER Any one letter, number (0-9), space, or symbol.

COMMAND A specific order to the computer, such as RUN, PRINT, LIST, CONTINUE, etc.

CONTINUE If you stop a program by holding down the FUNCTION key and hitting 4, you can begin it again by typing CONTINUE and pressing the ENTER key.

CURSOR The blinking black rectangle that moves on the screen. It is where the next character will be.

DATA A list of values to be given to the READ statements.

EDIT A command used with a line number to call up a line in a program to be edited or changed.

EDITING MODE A mode in which you are able to alter, delete, add to, or otherwise edit the statements in a program.

ENTER A key that enters whatever you have typed on the screen into the computer's memory.

EXECUTION MODE A mode in which the computer executes the command you have given in the program.

FOR/NEXT The FOR statement tells the computer to do something a certain number of times. The NEXT statement comes after the FOR

statement and tells the computer to go back and do the action stated. The FOR/NEXT statement is a type of loop.

Example:
```
10 FOR A = 1 to 15
15 PRINT A
20 NEXT A
```

FUNCTION KEY (FCTN) When held down, this allows other keys to function as marked on their sides, as follows:

FCTN	A		
	C		
	D	→	moves the cursor forward one space without changing what was in that space
	E	↑	in editing, moves cursor to line above
	F	{	(left brace)
	G	}	(right brace)
	I	?	(question mark)
	O	'	(apostrophe)
	P	"	(quotation marks)
	R	[	(left bracket)
	S	←	backspace without removing what was there before
	T	]	(right bracket)
	U	_	(underline)
	W	~	(tilde)
	X	↓	in editing, moves cursor to line below
	Z	\	

FCTN 1	DELETE	allows you to delete material to the right of the cursor one space at a time.
2	INSERT	allows you to insert material into an already entered line.
3	ERASE	allows you to erase an entire line.
4	CLEAR	will break a running program. The same key may be used when entering a program; in this case it will scroll the screen up one line without entering that line.
FCTN-	± QUIT	will take the computer out of BASIC.

GOSUB (or GO SUB) A command used to send the computer to a subroutine—it always used with the RETURN command.

GOTO Tells the program to go to the line number stated; the program continues from that line.

Example:
```
GOTO 100.
```

IF-THEN A program device for the computer to go to another part of the program if a relationship is true.

Example:
```
20 IF A< >B THEN GOTO 70
30 IF A = B THEN 40
40 PRINT B: END
70 PRINT
```

IMMEDIATE MODE The computer is in the immediate mode when it has a command or a list of commands that are to be executed as soon as RETURN is pushed. In the immediate mode, the commands do not have line numbers, which are necessary for the program mode.

INPUT Lets the user type in a response while the program is running.

Example:
```
10 INPUT A
RUN
?5
```

INT (or INTEGER) A command to tell the computer to round off to a whole number or integer.

LET Statement setting a value, as in LET A = 13.

LINE NUMBER A number typed at the beginning of a memory line.

LIST Shows everything in the computer's memory in line-number order.

LOOP A series of instructions that are repeated over and over again until an ending condition is reached.

MEMORY Where information is stored inside the computer. There are two kinds of memory: RAM and ROM. RAM means "RANDOM ACCESS MEMORY" and is used for the material you enter. ROM is "READ ONLY MEMORY" and is used by the computer manufacturer to store languages such as BASIC and other material that you need to know.

MENU A list of numbered choices. It asks the user to select one.

NEW A command that erases everything in the random access memory.

OUTPUT What you see on the screen, printer, or other hardware.

PRINT A command that tells the computer to write on the screen whatever follows PRINT.

Example:
```
PRINT 2* t
10
```

PROGRAM Instructions to the computer telling it what to do and how.

PROGRAMMING MODE The mode in which the computer stores in its memory instructions and their order of execution.

Example:
```
10 INPUT A$
20 PRINT A$
```

RANDOM (RND(1)) A command that tells the computer to choose a random number. It must appear after the word RANDOMIZE.

RANDOMIZE Tells the computer to turn on the random-number function.

READ Puts value from DATA statements to variable(s).

Example:
```
100 READ A, B$
```

REM (REMARK) REM allows the programmer to make notations in the program without having those notations become part of the program.

RETURN A command at the end of a subroutine that brings the computer back to the line following the one that told it to go to the subroutine.

RUN A command that starts the program executing.

STRING A group of characters represented by a two-or-more-digit symbol ending in a $—as in A$.

TAB A command that lets you position words on the screen.

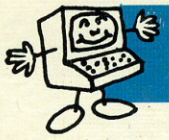
TIMER Delays a program. It leaves a certain amount of time between one part of a program and another point.

TRACE A command that, when used with RUN, will have the computer print on the screen all the steps it goes through in a program in the order that it does them.

UNTRACE A command that turns off TRACE.

VARIABLE A character that can take on any of a set of given values.

INDEX



Kids Working with Computers

THE **ACORN® BASIC** MANUAL

THE **APPLE® BASIC** MANUAL

THE **APPLE® LOGO®** MANUAL

THE **ATARI® BASIC** MANUAL

THE **COMMODORE® BASIC** MANUAL

THE **COMMODORE® LOGO** MANUAL

THE **IBM® BASIC** MANUAL

THE **IBM® LOGO** MANUAL

THE **TEXAS INSTRUMENTS BASIC** MANUAL

THE **TEXAS INSTRUMENTS LOGO** MANUAL

THE **TRS-80® BASIC** MANUAL

THE **TRS-80® COLOR LOGO** MANUAL

0 33038 08428

ISBN 0-516-08428-3

CHILDRENS PRESS
REINFORCED BINDING