$9.95

Texas Instruments

PROGRAMMING DISCOVERY IN

TI BASIC

STUDENT GUIDE

QUICK REFERENCE CHART OF TI-99/4A KEYBOARD FUNCTIONS

Press:

ALPHA LOCK

In locked (down) position causes characters on the screen to print in upper case; in unlocked (up) position produces lower case characters.

SPACE BAR

Moves cursor right, producing a space on the screen; also erases characters cursor moves across.

ENTER

Inputs information to the computer.

FCTN

The FCTN key accesses third characters on the front of the keys, as well as the functions identified on the interchangeable overlay strip located on the console above the keyboard. The FCTN key as well as the second key designated must be pressed simultaneously to access the desired function.
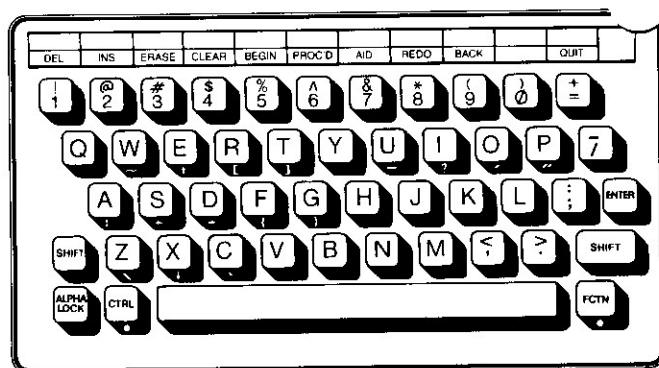
Press:

FCTN =	*(QUIT)*	Returns to TI Master Title Screen from program (or cartridge).
FCTN 1	*(DEL)*	Deletes text.
FCTN 2	*(INS)*	Prepares computer to insert text.
FCTN 3	*(ERASE)*	Clears an entire line from the screen.
FCTN 4	*(CLEAR)*	Causes a program in progress to stop.
FCTN S	*(◀)*	Moves cursor left without erasing text printed on the screen; negates DELete and INSert functions.
FCTN D	*(▶)*	Moves cursor right without erasing text printed on the screen; negates DELete and INSert functions.

IMMEDIATE MODE COMMANDS

The following commands are used frequently in Immediate Mode. CALL CLEAR can be used in both Immediate and Programming Modes.

Type:

BYE	**Press:** *(ENTER)*	Clears computer's memory and returns to TI Master Title Screen, leaving TI BASIC.
NEW	*(ENTER)*	Clears computer's memory and returns to TI BASIC READY screen (remains in TI BASIC).
CALL CLEAR	*(ENTER)*	Clears the screen but does not affect memory (remains in TI BASIC).
EDIT	*(ENTER)*	Brings the line, beginning with whatever number typed (100, 200, etc.) after pressing ENTER, to the screen for editing.
RUN	*(ENTER)*	Executes (runs) a program in memory.
LIST	*(ENTER)*	Lists a program in memory.

TEXAS INSTRUMENTS

PROGRAMMING DISCOVERY IN

TI BASIC

STUDENT GUIDE

Texas Instruments invented the integrated circuit, the microprocessor
and the microcomputer, which have made TI synonymous
with reliability, affordability, and compactness.

TEXAS INSTRUMENTS COMPUTER ADVANTAGE CLUB

TABLE OF CONTENTS

YOUR NAME _____

YOUR INSTRUCTOR _____

CITY _____

STATE _____

DATE _____

TI BASIC
can help you discover a lot about yourself ...

that **YOU** are...
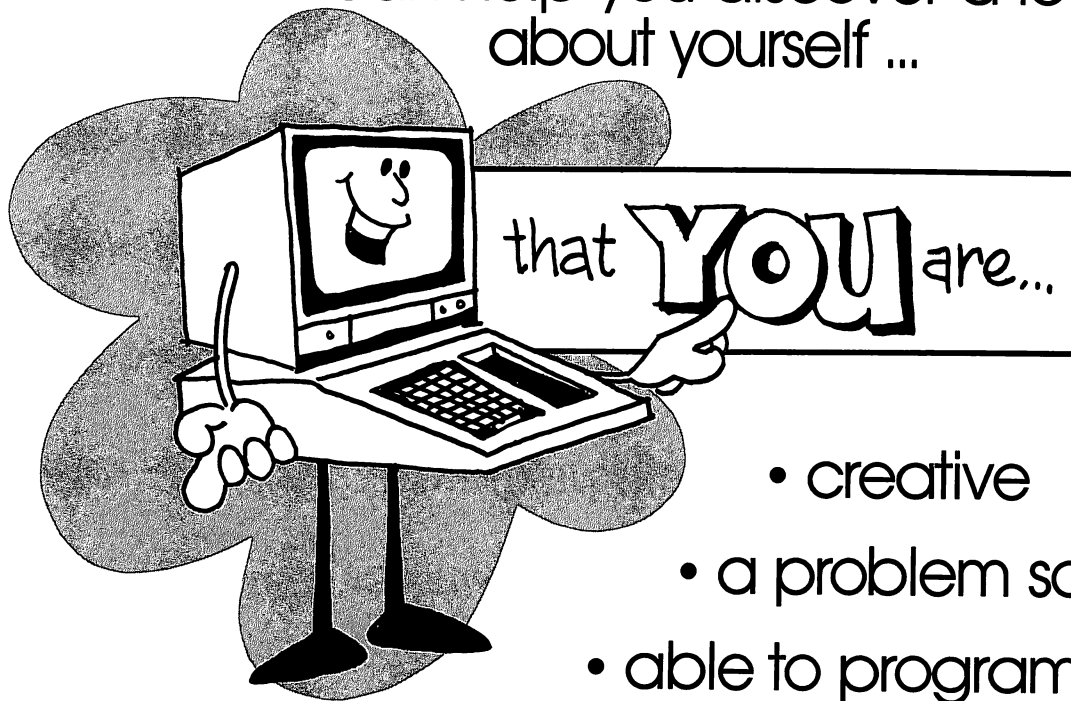
- creative
- a problem solver
- able to program

Have a great time discovering with **TI BASIC!**

What is TI Basic?

BASIC stands for Beginner's All-purpose Symbolic Instruction Code. It is the most popular programming language in use today because it is much like our own English language. Most microcomputers use a version of BASIC as their primary programming language. The TI-99/4A BASIC is called TI BASIC. It gives your computer a full range of programming capabilities for most home and personal applications.

What can my child expect to learn from this book?

This children's workbook introduces beginning level TI BASIC programming skills. Because TI BASIC is a "language," the terminology and concepts take time for assimilation and exploration, but there is much more to learn and explore beyond this level.

We hope your child's enthusiasm for learning will be an incentive for other members of your family to explore the learning environment of Texas Instruments Computer Advantage Club classes, too. For information concerning TICAC classes, for purchasing TI Home Computer software, peripherals, or accessories that you are unable to obtain from your local dealer, or for any questions you may have about your TI products, call Texas Instruments Consumer Hotline at our toll free number **1-800-TI CARES.**
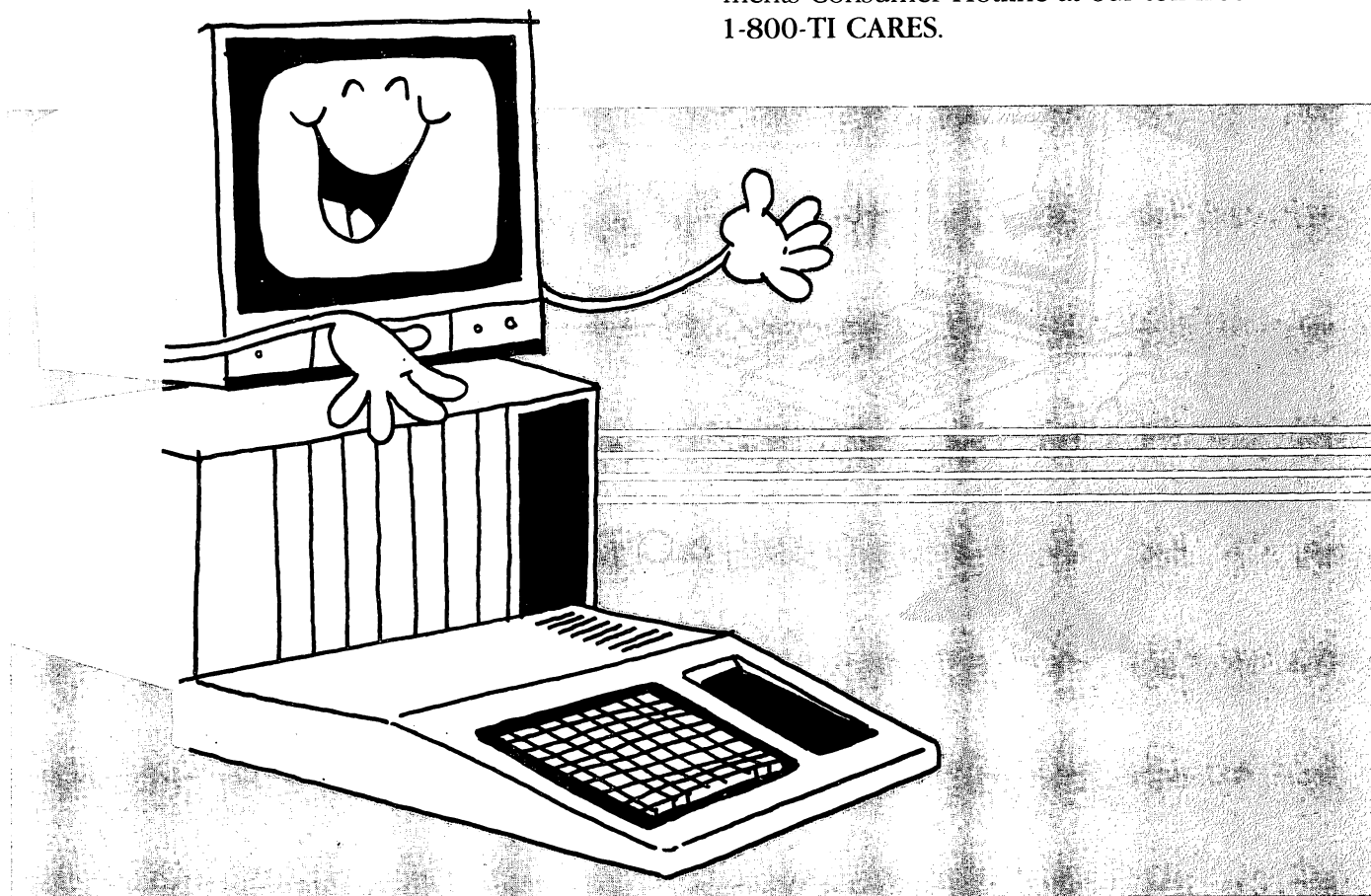
The first computers were very large—the size of one or two rooms—and weighed thousands of pounds. They used large glass vacuum tubes to function as switches, much like a light switch. Even the earliest electronic computers used electronic codes to process information. Depending on whether a switch is on or off, electronic codes that the computer understands can be created. When small, metallic *transistors* were invented, they replaced the vacuum tubes. This was the first step computers took toward growing smaller and faster.

When the integrated circuit was developed, one little flake of silicon, a *chip*, contained tens of thousands of transistors. Computers shrunk again in size. At the same time they became faster and more powerful. When a *microprocessor* (the "brains" of the computer) was put on one chip in 1971, minicomputers and microcomputers were born.

Today, there are three main types of computers: mainframes, minis, and micros. *Mainframes* are large computers, like those used at universities, NASA, and in large businesses. They process many jobs at one time. They have a huge memory storage capacity and are extremely fast. When a mainframe runs a program, even very large ones, it takes only fractions of a second to complete the job. Mainframes cost millions of dollars.

Minicomputers are smaller versions of mainframes. They are very fast and can run more than one program at a time. Minicomputers can cost tens of thousands of dollars.

Microcomputers are the leaders in the computer revolution. Because of their size, portability, and low price, they have become very popular with the general public. Micros have smaller memories compared to minis and mainframes. Computer scientists are building more memory into them all the time. Usually only one person at a time uses a microcomputer. That is why they are sometimes called personal computers. Microcomputers come in large and small sizes. The larger micros are usually used in small businesses. Smaller microcomputers are generally used in the home. They can cost from under one hundred dollars to thousands of dollars.
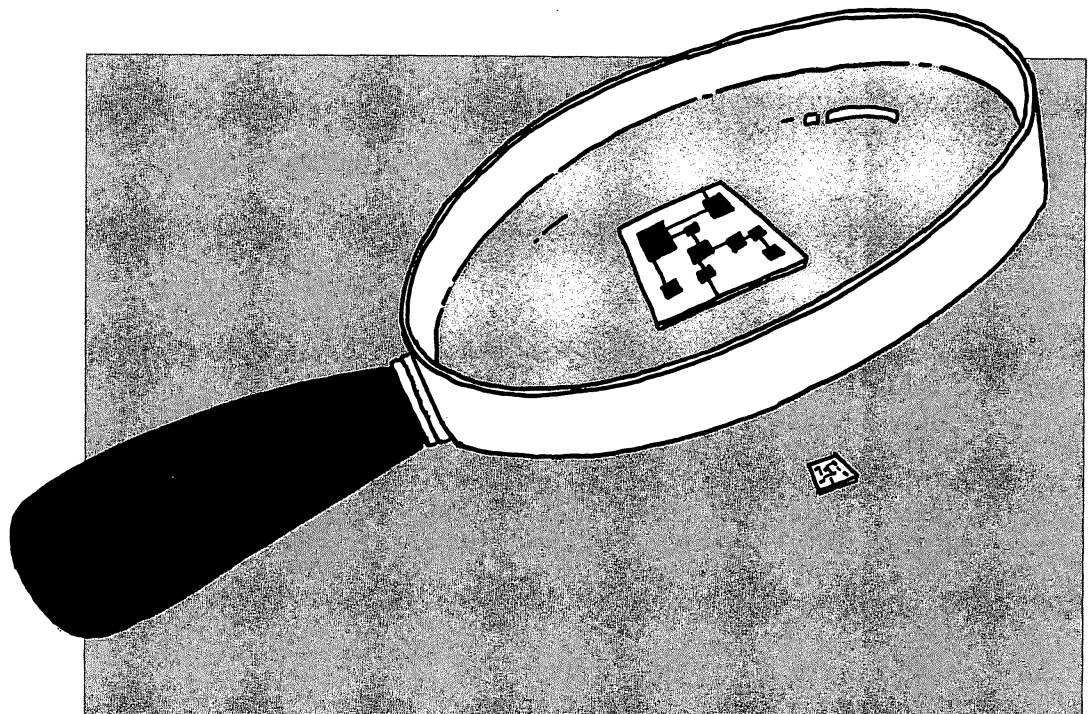
Why do we need computer languages?

Since computers only understand electronic impulses, we need a way to communicate and exchange information with them. A computer must be given specific instructions telling it what to do, when, and where to put information, or *data.* Computer languages allow this necessary communication between people and the machines.

## What are the names of some common computer languages?

BASIC, Extended BASIC, Assembly, Pascal, LOGO, PILOT, etc.

What do the letters in the word *BASIC* represent?

Beginner's **A**ll-Purpose **S**ymbolic **I**nstruction **C**ode.

Why BASIC?

BASIC is one of the programming languages that is easier to learn. BASIC is very much like English. Many of the words used in BASIC have the same meaning in English: PRINT, GOTO, STOP, END, RUN, etc. Most microcomputers use a built-in version of BASIC, making it one of the most widely used computer languages.
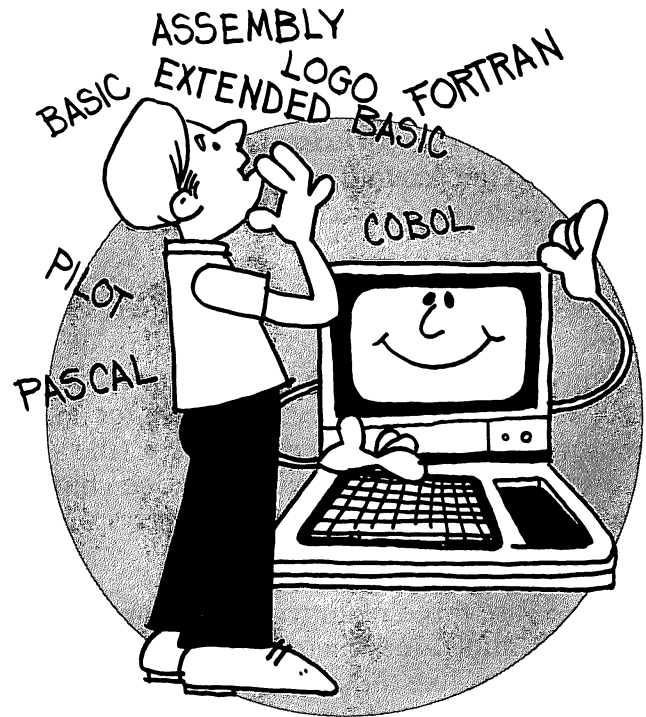
Other programming languages

TI Extended BASIC is a more powerful version of TI BASIC. With TI Extended BASIC, you can make graphics that move on the screen. TI Extended BASIC can also use more memory than TI BASIC, so programs can be longer. It is faster than TI BASIC, too. Because it is more powerful than TI BASIC, TI Extended BASIC is often used for business programs and games.

Assembly is a language very much like the electronic code of zeros and ones the computer uses. Since Assembly language goes through very little translation by the computer, it runs programs much faster than other languages. Assembly commands do not look like TI BASIC commands, which use common English words. This fact makes Assembly more complicated to use in programming.

Pascal is a language that, like BASIC, is easy to read and understand. Pascal is readily translated by com-

puters into their own electronic code. Pascal can run on many different types of computers with very little translation required.

LOGO is designed especially for children who are just learning how to use computers. It is very easy to use and is especially good for drawing pictures on the screen. But it is also powerful enough to solve complex problems.

PILOT stands for Programmed Inquiry, Learning, Or Teaching. It is designed to help teachers develop computer-based learning programs for use in the classroom.

TI Forth is an advanced programming language. Like Assembly, it is very fast but easier to learn than Assembly.

A computer can:

- Help you learn programming languages, which, in turn, can help improve your reading skills, thinking, and ability to follow directions.
- Help you keep track of things (how many and what kind of baseball cards, stamp books, etc. you may have).
- Help you learn to type.
- Help you calculate scoring averages of your favorite sports teams.
- Help you learn more math.
- Help you learn how to spell better.

What do you think a computer could do for:

Your parents? _____

_____

_____

Your grandparents? _____

_____

_____

Your friends? _____

_____

_____

Get Ready . . .

The computer has two levels or *modes* of operation: Immediate Mode and Programming Mode. In *Immediate Mode,* any command you type is performed immediately when you press the ENTER key. In *Programming Mode,* each instruction to the computer has a number in front of it. The computer stores all the numbered instructions and does not perform them until you type RUN and press ENTER. Then the computer finds the instruction with the lowest number and performs or *executes* it. It continues to the next highest-numbered instruction and performs it. The computer continues in this manner until all instructions have been executed.

Get Set . . .

Review the pages about how to use the keyboard. Learn to use the arrow keys (FCTN S, FCTN D) and the space bar to move to any mistakes you make, then correct them.

Turn on the TV or monitor, then the computer console. The colorful TI Master Title Screen appears on the screen. Follow the instructions PRESS ANY KEY TO BEGIN. Then follow the second set of instructions PRESS 1 FOR TI BASIC. A third screen appears and the message TI BASIC READY displays at the bottom of the screen. Below that message is a prompt character (>). The *prompt* character marks the beginning of each line typed. Right beside it is a flashing *cursor* (■) showing the computer is ready for instructions or data to be entered! Information you give to the computer is called *input.*

Go!

Check your keyboard. Is the ALPHA LOCK key in locked (down) position? The computer may not understand some of your commands if the ALPHA LOCK key is in unlocked (up) position.*

Now try this:

Is everything correct? If your name were Jamie, your command would look like this:

The line PRINT "JAMIE" moves upward on the screen. The upward movement is called *scrolling*. When you press ENTER, the name is printed below the command. Information or data the computer gives back to you is called *output*.

Typing PRINT is a shortcut for telling the computer "I want you to print on the screen whatever I type." That is why it is called the PRINT command!
Try another example.

The automatic repeat feature allows four lines to be printed on the screen before stopping. This automatic return to the next line is called *wrapping*.

Type:
PRINT "   "

PRINT "JAMIE"
Now press:
ENTER

TI BASIC READY

>PRINT "JAMIE"
JAMIE

Press FCTN P to make quotation marks. Type your name between the quotation marks.

The computer automatically places a prompt character (>) in front of each command line.

The flashing cursor appears below the name printed.

*If you forgot a quotation mark, or misspelled PRINT, the computer will print an *error message* (*INCORRECT STATEMENT) on the screen after pressing ENTER. This message tells you something needs to be corrected. Simply retype the line correctly and press ENTER. For more information on error messages, see pp. III-8–III-12 in the *User's Reference Guide*.

Type:
PRINT "XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXX "
Press:
ENTER

Helpful Hint: Hold down the X key. It will *automatically repeat* until the cursor comes to the end of the fourth line. Type quotation marks over the last X.

*When you use joysticks, put ALPHA LOCK in the unlocked (up) position.

The PRINT command can also make the computer act like a calculator.

This is what will appear on your screen:

The PRINT command scrolls upward, and the answer is printed on the next line. Notice the number does not print at the left margin as does the PRINT command above it. The comptuer automatically prints a *leading space* in front of numbers and a *trailing space* behind numbers. If a number is negative, a minus sign appears in front of the number instead of a space.

When you want the computer to work math problems:

Type:
PRINT 2 + 8
(Press ENTER)

Use SHIFT = to make the addition sign. Be sure not to press FCTN = (QUIT) by accident!

TI BASIC READY

>PRINT 2 + 8
 10
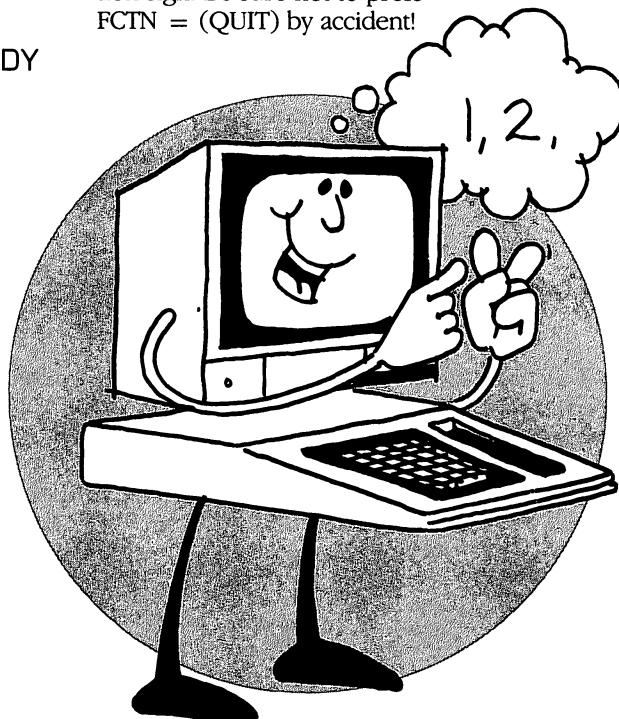
- To subtract . . . . use the − sign (SHIFT /).
- To multiply . . . . use the * sign (SHIFT 8).
- To divide . . . . use the / sign.
- For exponents* . . . . use the ^ sign (SHIFT 6).

---

*An example of an *exponent* is 2^3. This is the same as saying $2 \times 2 \times 2$. Both 2^3 and $2 \times 2 \times 2$ equal 8. In computer print, this "exponentiation" is written 2^3 because the computer cannot print the small numbers.

Try this problem. Make sure you leave a space between PRINT and 3.

Type:
PRINT 3*10-4/2 + 2^2
(Press ENTER)

 32

The computer performs mathematical operations in this order:

3*10 − 4/2 + 2^2
3*10 − 4/2 + 4
30 − 4/2 + 4
30 − 2 + 4
28 + 4
32

Exponents first.
Multiplication and division, from left to right.

Addition and subtraction, from left to right.
The answer is 32.

Using parentheses can change the order in which the comptuer performs mathematical operations. Any operation between parentheses is performed before all others. Try this example:

The answer is 13. What would the answer be if the parentheses weren't there? The computer can remember numbers for you.

Using the LET statement is like telling the comptuer "I want you to remember that the letter A stands for 2, and the letter B stands for 3."

Now tell the computer to add, subtract, multiply, or divide these numbers for you.

Try this one.

Nothing seems to happen when you press ENTER after typing this line. The computer has made C equal to the sum of A + B. You know the answer is 5.
But to see the answer on the screen, you must tell the computer to print it!

The answer prints on the screen!

Type:
PRINT 3*(10-4)/2 + 2^2
(Press ENTER)

   13

Type:
LET A = 2
(Press ENTER)

Type:
LET B = 3
(Press ENTER)

Type:
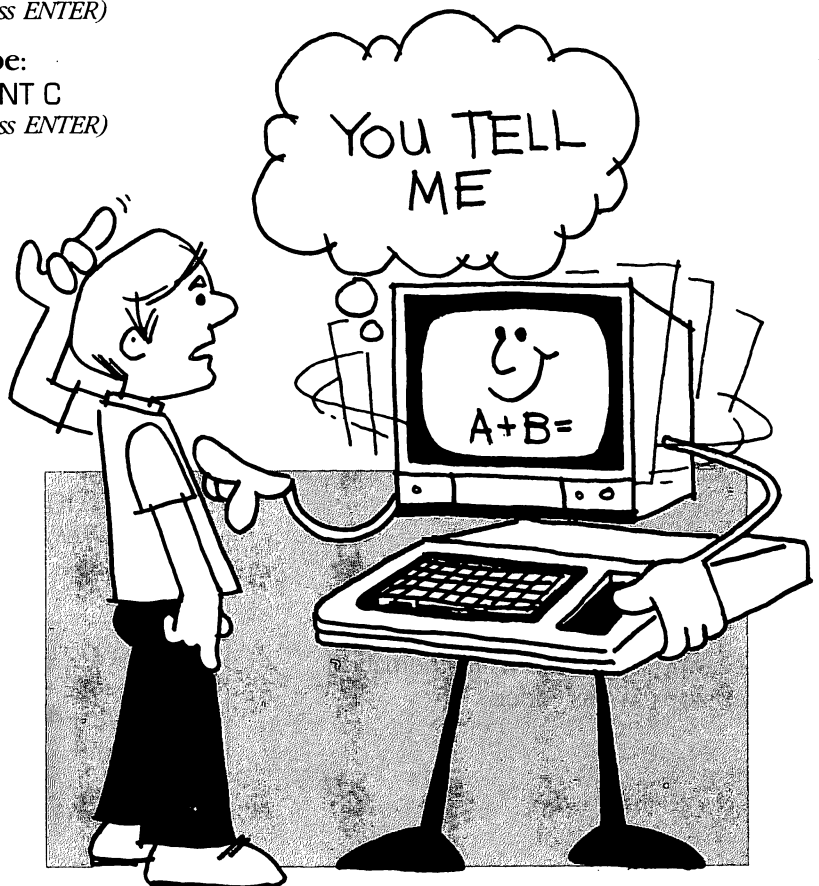PRINT A + B
(Press ENTER)

   5

Type:
LET C = A + B
(Press ENTER)

Type:
PRINT C
(Press ENTER)

   5

Use SHIFT 9 to make the left parentheses and SHIFT 0 to make the right parentheses. Be sure not to use brackets [ ] or braces { } by accident! Can you predict the answer?

The answer is displayed below the PRINT command.

Variables

In previous examples, the letters A, B, and C have represented numbers. The letters A, AB, BBB, CCCC, or any other combination (up to fifteen letters) can represent any numbers you choose. These letters are called *variables** because their value can vary or change. The variables you have seen so far are called *numeric variables* because their values are equal to numbers (LET A = 2).

The LET statement can also be used to make a variable represent a word, a name, a phrase, or a sentence.

It is like telling the computer "I want you to remember that A$ stands for my name." When you press ENTER, nothing happens. Remember—to see something printed on the screen, you must tell the computer to print it!

When you press ENTER, your name appears on the screen!

When variables represent letters or words, they are called *string variables*. When you want the computer to remember a value with letters in it, such as a name or a phrase, you must always:

This is how the computer tells the difference between a numeric variable (A) and a string variable (A$). A variable with a $ can contain letters and numbers, but the numbers cannot be added, subtracted, multiplied, or divided. Numeric variables can contain only numbers, and they can be used in mathematical operations.

Both string and numeric variables can be made of up to 15 characters, for example: LET RUMPELSTILTSKIN = 1

The value you give to either a string or a numeric variable is called a *constant*. In the example above, 1 is the constant.

You probably have a lot of clutter on the screen. You can "clear" the screen, like you would erase a chalkboard.
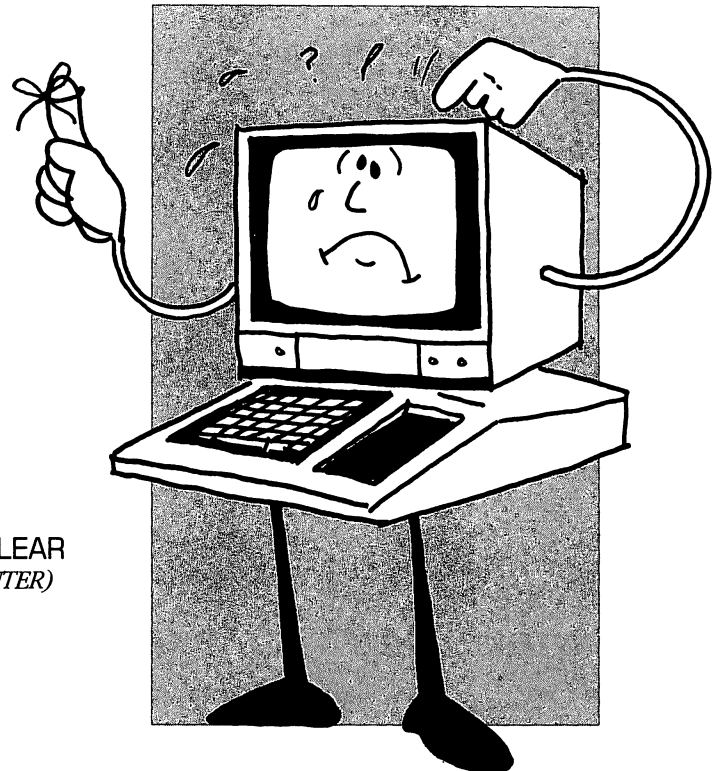
The screen is cleared!

*See the *User's Reference Guide* page II-11 for more information on variables.

Type:
LET A$ = "   "
(Press ENTER)

Type your name between the quotation marks.

Type:
PRINT A$
(Press ENTER)

1) use a dollar sign ($) at the end of the variable
and
2) put what the variable stands for between quotation marks

Type:
CALL CLEAR
(Press ENTER)

The computer can space messages on the screen. This is done using colons, commas, and semicolons. These characters are called *print separators*. They separate or space items printed on the screen!

These examples demonstrate how print separators work.

Type: PRINT 1:2:3:4
(Press ENTER)

Use a colon to print data (numbers or words) on separate lines. The numbers will print in a column one under the other.

1

2

3

4

Type: PRINT 1::2
(Press ENTER)

Use more than one colon if you want to skip more than one line between printed items.

1

2

Type: PRINT 1,2,3,4
(Press ENTER)

Remember—positive numbers are printed with a leading space. That's why the numbers in this example print in columns 2 and 16.

1 2
3 4

Type: PRINT "HELLO";"HELLO"
(Press ENTER)

A semicolon causes characters to print side-by-side with no extra spaces between them.

HELLOHELLO

Type: PRINT "HELLO";" HELLO"
or
Type: PRINT "HELLO ";"HELLO"
or
Type: PRINT "HELLO HELLO"

String variables print without spaces leading or following them. To print items side-by-side with spaces between, leave a space before (or after) the items to be separated. After entering each of these commands, the computer will print HELLO HELLO on the screen (one space between the two words).

Type: PRINT 1;2;3;4
(Press ENTER)

1 2 3 4

Remember—numbers print with trailing spaces following them. Positive numbers print with leading spaces in front of them instead of a plus sign. Negative numbers print with a minus sign in front of them instead of a leading space.

The TAB function can also be used to position printing on the screen.

Type: PRINT TAB (5);"HELLO"
(Press ENTER)

HELLO

The TAB function names the column where characters are to begin printing on the screen. The column number in parentheses follows the word TAB. TAB is very useful for printing two or more columns on the screen.

You can make the computer perform several commands all at once but only when you are ready for it to execute. Place numbers in front of each program line. That tells the computer to store all data until given an execute (RUN) command.

The computer must "hold" all data to be stored until you are ready for it to execute your instructions. To make the computer store data in its memory, place *line numbers* in front of each instruction. When a command or instruction to the computer has a line number in front of it, it is called a statement or a *program line.* When the computer stores all program lines in its memory until you tell it to execute (RUN), it is operating in Programming Mode.

In this example, program lines are numbered by tens starting with ten. When you tell the computer to execute or RUN the program lines, it will look for the smallest number first and perform it. Then it will find the next larger number, perform it, and so on. You can number program lines starting with 1 or with 100. It makes no difference to the computer. Most programmers number their program lines by tens. This leaves room between lines to add more lines later if necessary.

```
10 LET A = 5
20 LET B = 8
30 LET C = A + B
40 PRINT C
50 END
```

When you *key in* this program, use the arrow keys and space bar to correct errors as you go along. (See Shortcuts and Editing Features, pp. 52-55, for more information about correcting mistakes.) Leave a space between the line number and the word following it. Remember to press ENTER at the end of each line. Using an END statement with TI BASIC to end programs is optional (line 50).

You be the computer. What will each PRINT command print on the screen when the program runs?

```
10 LET PHRASE$ = "I KNOW
THAT"
20 LET A = 10
30 LET B = 5
40 LET C = A*B
50 LET A$ = "TIMES"
60 LET B$ = "EQUALS"
70 PRINT PHRASE$
80 PRINT A
90 PRINT A$
100 PRINT B
110 PRINT B$
120 PRINT C
130 END
```

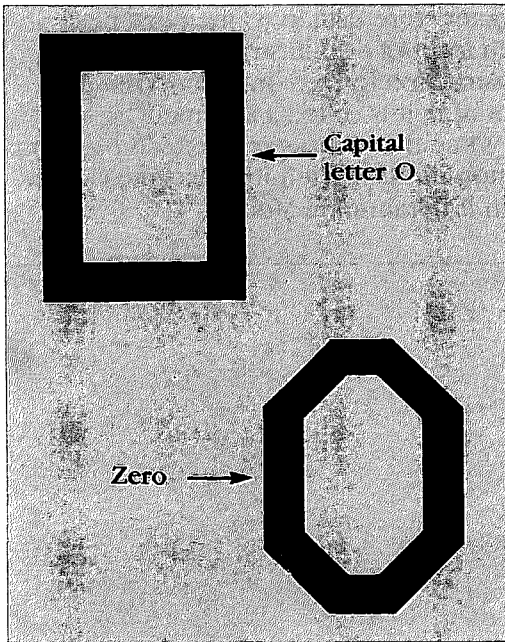Now it's time to tell the computer to perform your instructions!

Type:
RUN
(Press ENTER)

The screen turns green, and your instructions are performed! When the computer has finished, the screen turns to normal color and **DONE** appears at the bottom of the screen. A set of instructions like the one you just entered and ran is called a *program.*

Capital letter O

Zero ➔

To make the computer clear the screen before printing, add a CALL CLEAR statement.

LIST the program again to see how this new line is added automatically.

Check each line to make sure there are no mistakes. Remember to always use a 0 when you mean zero, not the capital O. Also, always use the numeral 1 when you mean the number one, not the small letter 1.

If you find a mistake in a line you have already entered:

When you press ENTER, line 10 will display at the bottom of the screen. Use the arrow keys (FCTN S, FCTN D) to move the cursor to the error. Then type over the error to correct it. When you are finished correcting the line, press ENTER. Now run the program again. Everyone makes mistakes, including people who write programs for a living! Part of writing good programs depends on how good you are at finding mistakes! This process of getting the *bugs* out of a program is called *debugging* a program. Study the section on Shortcuts and Editing Features, pp. 52-55, for more information on how to find and correct mistakes.

If your screen did not turn green, but displayed an error message like *CAN'T DO THAT, *STRING-NUMBER MISMATCH IN 100 or *INCORRECT STATEMENT IN 120, the computer is letting you know that there is a mistake somewhere in the program! If this happens:

Type:
LIST
(Press ENTER)
The program you entered will display on the screen. Study each line and line number carefully against the program printed in this book. Are PRINT and LET spelled correctly? Are all the quotation marks included? Do all string variables have a $ following them? Are your line numbers all numbers and not letters like O and l? To correct errors, type EDIT followed by the line number you wish to correct, then press ENTER.

Type:
EDIT 10
(Press ENTER)

Use the arrow keys (FCTN S, FCTN D) to move the cursor to the error. Type the correction over the error. When you are finished correcting the line, press ENTER.

Type:
5 CALL CLEAR
(Press ENTER)

Type:
LIST
(Press ENTER)

Type:
EDIT 10
(Press ENTER)

Type whatever line number you wish to change or edit.

Type:
RUN
(Press ENTER)

Match-Up

Have you studied the keyboard and the section on Shortcuts and Editing Features on pages 52-55? If not, review those pages, then come back to this quiz and test yourself. Answers are in the Appendix on page 56.

___ 1. LIST

___ 2. CALL CLEAR

___ 3. NEW

___ 4. EDIT 10

___ 5. FCTN 4 (CLEAR)

___ 6. FCTN = (QUIT)

___ 7. FCTN 3 (ERASE)

___ 8. FCTN S (◀)

___ 9. FCTN D (▶)

___ 10. BYE

A. Moves the cursor to the left without erasing anything in its path.

B. Clears the screen and any program in memory, but remains in TI BASIC.

C. Returns to the TI Master Title Screen from TI BASIC and clears the memory.

D. Tells the computer to print the program in memory on the screen.

E. Returns to the TI Master Title Screen and clears the memory.

F. Clears the screen.

G. Moves the cursor to the right without erasing anything in its path.

H. Stops a program that is running.

I. Erases a program line from the screen.

J. Brings up line 10 of your program on the screen so you can change or edit it if you choose.
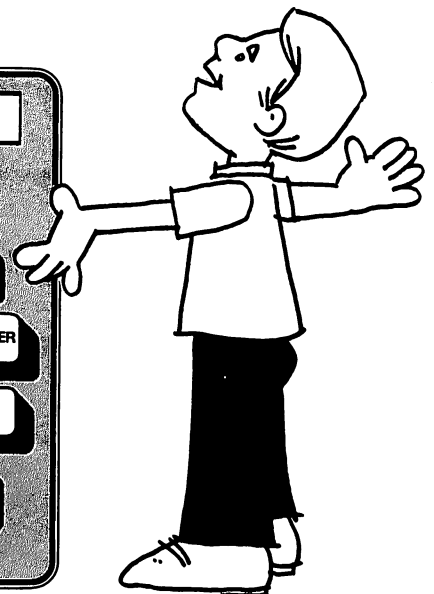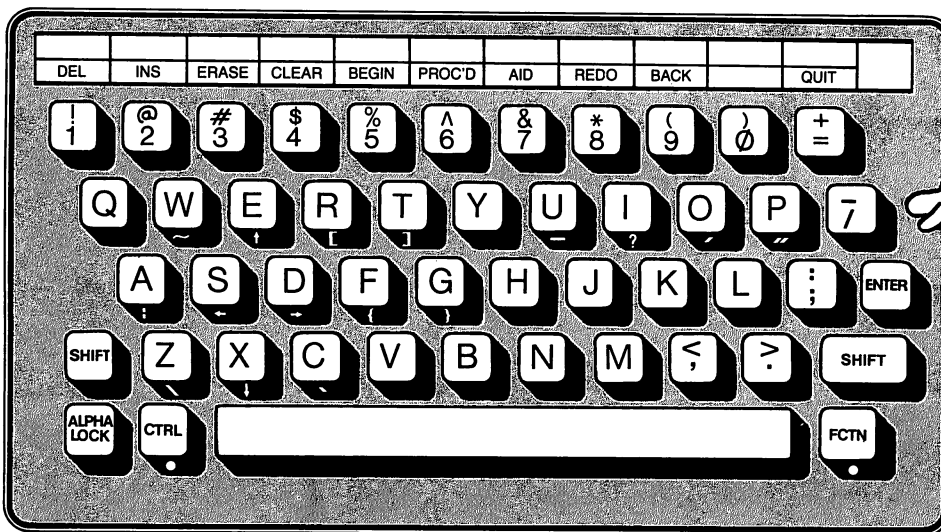
As you know, the computer performs program lines in numerical order, from smallest to largest. Using the GOTO statement, you can make the computer "jump" to any program line. When the computer "goes to" a program line out of order, it is called *branching*.

Here is an example of how the GOTO statement works:

NEW
(ENTER)

Always type NEW and press ENTER to clear the computer's memory before starting a new program.

10 PRINT "LET ME OUT OF HERE!"
(ENTER)

Line 10 will print a message.

20 GOTO 10
(ENTER)

Line 20 is a GOTO statement. It tells the computer to go back to line 10, so the message on line 10 is printed again. The computer performs the next instruction (line 20), which sends it back to line 10. This happens over and over, until you stop the program.

RUN
(ENTER)

To stop the program, use FCTN 4 (CLEAR). Hold down the FCTN key and the 4 key at the same time until the computer beeps. The message *BREAKPOINT AT 20 prints on the screen. That tells you the program stopped running at line 20.

To run the program again, type RUN and press ENTER.

The GOTO statement creates *loops* in programs. Program lines can "go to" themselves (60 GOTO 60). This is often used to hold graphics on the screen (see the program in the CALL SCREEN section, p. 20). It can also be used if you do not want the program to end and print **DONE** on the screen. As you learn more about TI BASIC, you will find many uses for GOTO.
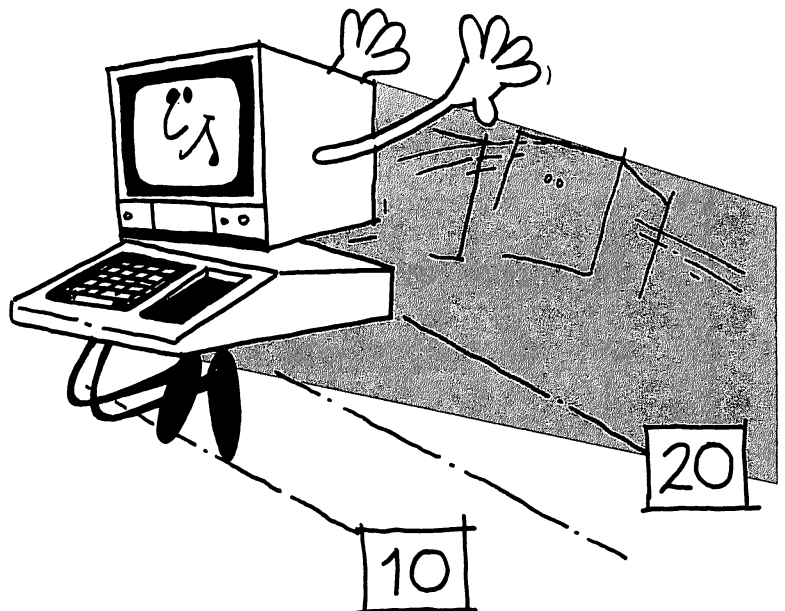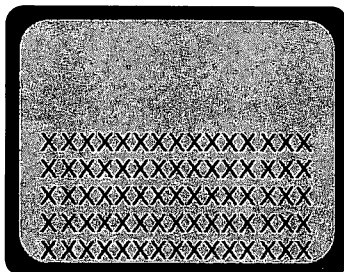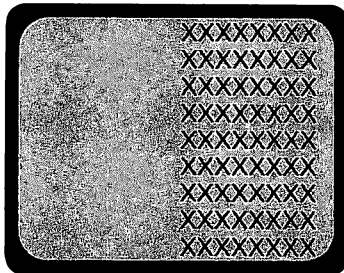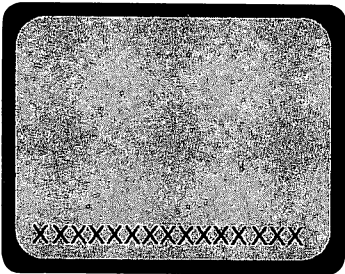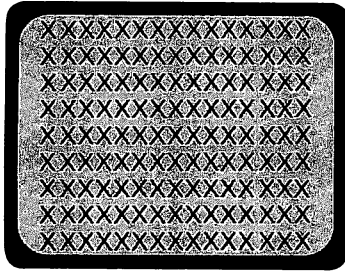
1. Write a program that will fill the entire screen with printing. Hint: This should be a two-line program!

_____

_____

_____

2. Write a program that prints one line that moves up to the top of the screen, disappears, and is followed by another from the bottom of the screen. Hint: This is a "loop" program and requires special spacing.

_____

_____

_____

3. Write a program that covers the entire right half of the screen with print. Hint: Use the TAB function.

_____

_____

_____

4. Write a program that covers the bottom or top half of the screen with print, then stops, but does not print **DONE**. Hint: You will need to make one program line "go to" itself. (Review the explanations on page 16.)

_____

_____

_____

_____

_____

Answers to "Practice with GOTO" can be found in the Appendix on page 56.

Using a mathematical formula, you can write a program that will make the computer calculate the answer. This program calculates the area of a rectangle (length of the rectangle multiplied by its width).

When you run this program, the computer prints a message (line 20). It stops at line 30 and displays a question mark on the screen. It is waiting for you to enter a number! Type the length of the rectangle and press ENTER. Another question mark appears when the program reaches line 50. Enter the width of the rectangle and press ENTER. The computer prints the answer!

The same program can be shortened by combining the message line with the IN-PUT statement. The message must still have quotation marks around it. A colon must separate the message and the variable. Leave a space between the end of the message and the ending quotation marks. This separates the last word of the message from the number input when the program prints. Your program will be easier to read when it prints on the screen.

When the program runs now, the cursor flashes to let you know the computer is waiting for INPUT. The previous program displayed a question mark to show you it was waiting for input.

*Notice the LET statement is missing. Using a LET statement to assign values to variables is optional.

```
NEW
10 CALL CLEAR
20 PRINT "ENTER THE
LENGTH OF THE RECTANGLE,
THEN PRESS ENTER."

30 INPUT L

40 PRINT "ENTER THE WIDTH
OF THE RECTANGLE, THEN
PRESS ENTER."

50 INPUT W

60 AREA = L*W

70 PRINT "AREA = ";AREA

80 END
RUN
```

Line 10 clears the screen.
Line 20 prints a message.

Line 30 stops the program and lets you enter a number.

Line 40 prints a message.

Line 50 lets you enter another number.

Line 60 is the formula for finding the area of a rectangle.*

Line 70 prints the answer.
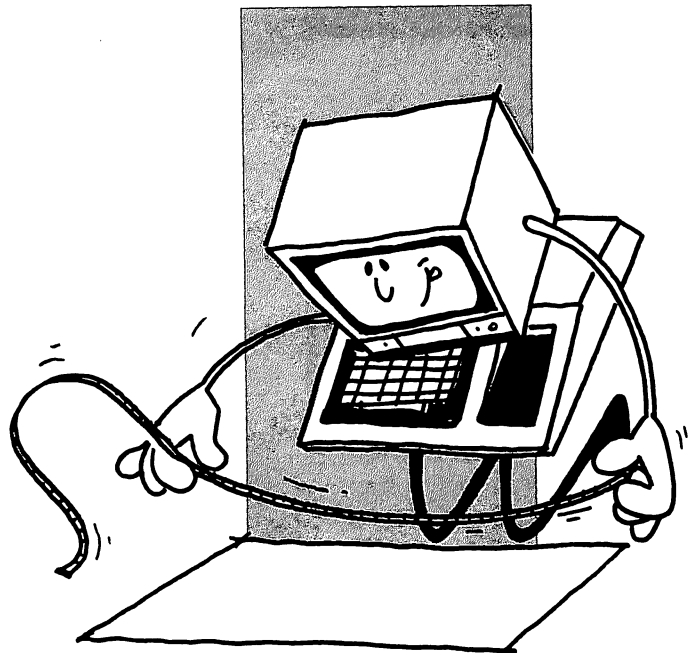
Line 80 ends the program.

```
NEW
10 CALL CLEAR
20 INPUT "ENTER THE
LENGTH OF THE RECTANGLE,
THEN PRESS ENTER ":L
30 INPUT "ENTER THE WIDTH
OF THE RECTANGLE, THEN
PRESS ENTER ":W
40 AREA = L*W
50 PRINT "AREA = " ;AREA
60 END
RUN
```

You can use an INPUT statement to enter string variables, too. This program lets you enter your name or your friend's name.

```
NEW
10 CALL CLEAR
20 PRINT "PLEASE TYPE
YOUR FIRST NAME, THEN
PRESS ENTER:"
```

```
30 INPUT N$
```
Line 30 waits for input.

```
40 PRINT "THANK YOU FOR
YOUR INPUT,  ";N$;" . "
```
Line 40 prints a message and the name that was entered in line 30.

```
50 END
RUN
```

A message and INPUT statement can be combined in programs using string variables, as well as in programs using numeric variables.

To edit line 20, type:

```
EDIT 20
(ENTER)
```

Change it to:

```
20 INPUT "PLEASE TYPE
YOUR FIRST NAME, THEN
PRESS ENTER  ":N$
(ENTER)
```

Messages used with INPUT statements explain what kind of information is needed. When directions in a program are clear and the program is easy to operate, the program is called *friendly*.

To erase line 30, type:

```
EDIT 30
(ENTER)
```

Press:

```
FCTN 3 (CLEAR)
```

or

Type:

```
30
(ENTER)
```

LIST the program to see how it has changed, then RUN the program. For more information about editing programs, see the section on Shortcuts and Editing Features on pages 52-55.

The TI-99/4A computer has sixteen colors you can use in programs. Each color has a code number. This chart lists the color code numbers.

COLOR CODE CHART

Color	Code #	Color	Code #
Transparent	1	Medium Red	9
Black	2	Light Red	10
Medium Green	3	Dark Yellow	11
Light Green	4	Light Yellow	12
Dark Blue	5	Dark Green	13
Light Blue	6	Magenta	14
Dark Red	7	Gray	15
Cyan	8	White	16

This program will change the screen color to dark red (color code 7).

```
NEW
10 CALL CLEAR
20 CALL SCREEN(7)
RUN
```

```
NEW
```

This program lets you change the color of the screen when the program is running. The CALL SCREEN statement "calls" the color you choose to the screen.

```
10 INPUT "CODE NUMBER?":N
```
Line 10 asks you to input the code number for the color you want to see on the screen.

```
20 CALL SCREEN(N)
```
Line 20 calls to the screen the color of the code number input in line 10.

```
30 GOTO 30
RUN
```
Line 30 is a GOTO statement that makes the color stay on the screen until you stop the program. Line 30 keeps "going to" itself!

To stop the program, press FCTN 4 (CLEAR).

The following program changes screen color whenever you input different numbers. The only difference between the program above and this program is in line 30. Instead of retyping the whole program again, just change line 30. To change line 30, retype the new line 30, press ENTER, then RUN the program.

```
NEW
10 INPUT "CODE NUMBER?":N
20 CALL SCREEN(N)
30 GOTO 10
RUN
```

If you input number 1 or 2, the screen will not display the message in line 10, but the program is still running! Try entering another number.

If you enter a number larger than 16 or smaller than 1, the program will stop and print an error message (*BAD VALUE IN 20). If this happens, type RUN again and press ENTER. The program will ask for input again. Use a number between 3 and 16.

The computer sees the TV or color monitor screen as a grid of rows and columns. Remember that rows go across. Columns go up and down. The screen is made of 24 rows and 32 columns. Each position on the screen can be identified by its row number and column number. Row 12, column 15 identifies the position of the star (asterisk) below.

You can make drawings called *graphics* using CALL VCHAR (for "vertical character") and CALL HCHAR (for "horizontal character"). To place one character on the screen in the position marked on the grid above, type and ENTER this program:

```
NEW
10 CALL CLEAR

20 CALL VCHAR(12,15,42)

RUN
```

Line 10 clears the screen.

Line 20 "calls" a character to the screen. The first number (12) is the row number. The second number (15) is the column number. The third number (42) is the code number for the asterisk. (See p. 24 for character code numbers.)

When this program runs, the screen turns green and an asterisk appears in the center. Then the screen returns to its normal color, and **DONE** prints at the bottom.

The CALL HCHAR statement can be used to produce the same character.

```
NEW
10 CALL CLEAR

20 CALL HCHAR(12,15,42)

RUN
```

Line 20 "calls" a character to the screen. The first number (12) is the row number. The second number (15) is the column number. The third number (42) is the code number for the asterisk.

The difference between the two statements, CALL VCHAR and CALL HCHAR, is shown in the next program. When this program is run, vertical (up and down) columns and horizontal (across) rows of characters print on the screen. The fourth number in each statement (24 and 32) is the number of times the character will repeat.

To stop the program, press FCTN 4 (CLEAR).

```
NEW
10 CALL CLEAR
20 CALL VCHAR(1,16,42,24)



30 CALL HCHAR(1,1,42,32)




40 GOTO 40
RUN
```

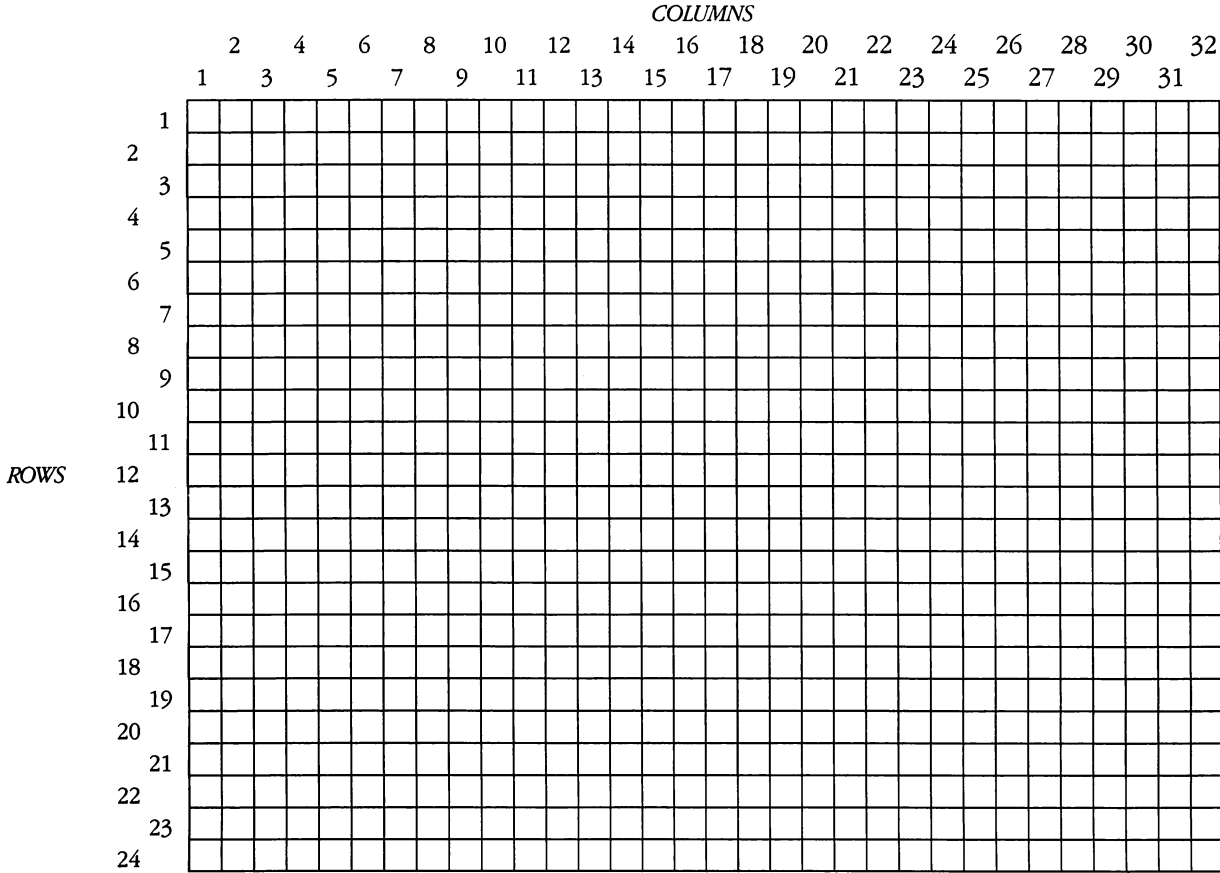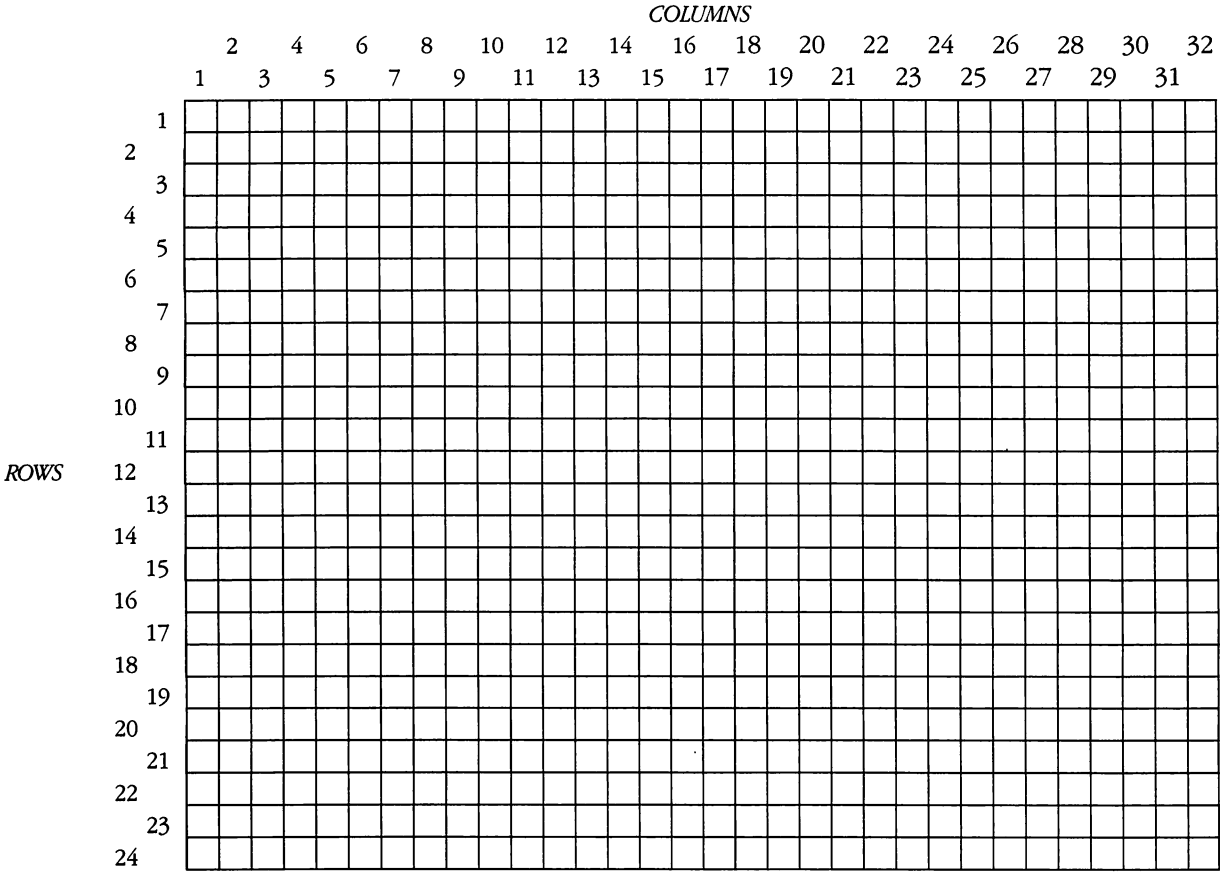**Line 20** places an asterisk in row 1 (top of the screen) at column 16 (center of the screen). Then 23 more asterisks print near the bottom of the screen.

Line 30 places an asterisk in row 1 (top of the screen) at column 1 (left-hand side of the screen). It also prints the asterisk 31 more times.

Line 40 causes a loop that holds the picture on the screen.

COLUMNS

ROWS

COLUMNS

ROWS

COLUMNS

ROWS

SET #1		SET #2		SET #3	
Code	Character	Code	Character	Code	Character
32	(space)	40	(	48	0
33	!	41	)	49	1
34	"	42	*	50	2
35	#	43	+	51	3
36	$	44	,	52	4
37	%	45	-	53	5
38	&	46	.	54	6
39	'	47	/	55	7

SET #4		SET #5		SET #6	
Code	Character	Code	Character	Code	Character
56	8	64	@	72	H
57	9	65	A	73	I
58	:	66	B	74	J
59	;	67	C	75	K
60	<	68	D	76	L
61	=	69	E	77	M
62	>	70	F	78	N
63	?	71	G	79	O

SET #7		SET #8		SET #9	
Code	Character	Code	Character	Code	Character
80	P	88	X	96	`
81	Q	89	Y	97	a
82	R	90	Z	98	b
83	S	91	[	99	c
84	T	92	\	100	d
85	U	93	]	101	e
86	V	94	^	102	f
87	W	95	—	103	g

SET #10		SET #11		SET #12	
Code	Character	Code	Character	Code	Character
104	h	112	p	120	x
105	i	113	q	121	y
106	j	114	r	122	z
107	k	115	s	123	{
108	l	116	t	124	¦
109	m	117	u	125	}
110	n	118	v	126	~
111	o	119	w	127	DEL

You have already learned how to change the screen color using the CALL SCREEN statement. The CALL COLOR statement can be used in programs, with CALL HCHAR and CALL VCHAR, to give color to the character and its background. Look at the screen grid on page 21. Each little square or block on the grid is made up of 64 smaller squares. The picture shows the square with the star (asterisk) from page 21 if it were magnified many times.

This is the asterisk from the screen grid on page 21. The color of the asterisk is called the *foreground color*. The color of the rest of the little block around it is called the *background color*.

To give color to characters on the screen, use a CALL COLOR statement. Color codes are on Page 20.

NEW

```
10 CALL CLEAR
20 CALL COLOR(2,5,14)
```

Line 10 clears the screen.
Line 20 gives color to screen characters. The first number (2) is a character set code. See the chart on page 24 to find the set number for character 42. The second number (5) sets the foreground color. The third number (14) sets the background color.

```
30 CALL VCHAR(1,16,42,24)
40 CALL HCHAR(1,1,42,32)
50 GOTO 50
```

RUN

Lines 30 and 40 produce a design on the screen.
Line 50 holds the design on the screen.

Press FCTN 4 (CLEAR) to stop this program. If characters from more than one character set are used, one CALL COLOR statement must be used for each set.

You have already seen some programs using FOR and NEXT statements. FOR-NEXT statements are similar to GOTO statements. In a FOR-NEXT statement the loop is *controlled.* The number of times a loop repeats is already decided. Branching in FOR-NEXT statements is called *conditional branching.* GOTO statements go directly to another line over and over until you stop it. This is called *unconditional branching.*

This program counts from 1 to 10 on the screen. Read the explanation for each line to see how FOR and NEXT statements work.

```
NEW
10 CALL CLEAR
20 FOR X = 1 TO 10
```

Line 20 assigns values to the variable. The first value assigned to X is 1. The last or maximum value assigned to X is 10.

```
30 PRINT X
```

Line 30 prints the current value of X.

```
40 NEXT X
```

Line 40 uses NEXT X. This means the value of X is increased by one. If the value of X was 1, it now becomes 2. The new value of X is tested against the maximum value of 10. As long as the current value of X is not greater than 10, the program branches to line 20. After several more loops, X reaches a value greater than 10. At that point the program goes to line 50 and ends.

```
50 END
RUN
```

Now try this program and see if you can follow how FOR-NEXT works here.

```
NEW

10 FOR X = 1 TO 10
```

Line 10 assigns values to X. The lowest value assigned to X will be 1. The last value assigned to X will be 10.

```
20 PRINT X
30 NEXT X
```

Line 30 makes the program branch back to line 10 until the maximum value of 10 is reached.

```
40 END
RUN
```

A good use of FOR-NEXT is to create a delay or pause in a program. Try the program below to see how the pause works.

```
NEW
10 CALL CLEAR
20 CALL COLOR(2,5,16)
30 CALL HCHAR(12,3,42,28)

40 FOR B = 1 TO 1000
50 NEXT B
60 END
```

Line 40 sets a maximum value for B at 1000. The computer must count to 1000 before ending the program.

```
RUN
```

1) Each FOR statement must be paired with a NEXT statement.
2) Each nested loop must use a different variable (X for one, Y for the other).
3) The inside loop (Y) must be contained within the outside loop (X).

Remember—each time you use a FOR statement in a program, you must use a NEXT statement!

More than one FOR-NEXT statement can be used in a program. These are called *nested loops* because one loop is contained inside the other. When using nested loops follow these rules:

This program has two loops. Can you find them? One is made up of lines 10 and 50, the other is made up of lines 20 and 40. RUN this program to see how the values of X and Y print on the screen.

```
NEW
10 FOR X = 1 TO 3
20 FOR Y = 1 TO 3
30 PRINT X;Y
40 NEXT Y
50 NEXT X
60 END
RUN
```

A FOR-NEXT statement has a built-in *counter.* FOR-NEXT loops are automatically set to count forward by ones.

This program displays the sixteen different screen colors. Use of DELAY holds each color briefly on the screen.

```
NEW
100 CALL CLEAR
```

```
110 FOR N = 1 TO 16
```
Line 110 names the color code numbers of the colors to be displayed on the screen.

```
120 CALL SCREEN(N)
```
Line 120 calls the colors to the screen.

```
130 FOR DELAY = 1 TO 200
```
Line 130 sets the counting range from 1 to 200.

```
140 NEXT DELAY
```
Line 140 creates a loop making the computer count to 200 before going on to line 150.

```
150 NEXT N
```
Line 150 starts the process to select the next screen color.

Use STEP if you want to make the computer count faster (for example, to count by 5s, 10s, or 20s), or count backwards.

This program counts from 0 to 100 by 10s and prints the numbers on the screen.

```
100 CALL CLEAR
```

```
110 FOR X = 0 TO 100 STEP 10
```
Line 110 sets the counting range from 0 to 100 in increments of 10.

```
120 PRINT X
```
Line 120 prints each number.

```
130 PRINT
```
Line 130 double-spaces the numbers printed by "printing" a blank line.

```
140 NEXT X
```
Line 140 begins the process of printing the next number on the screen.

STEP must have a space in front of it and behind it. Try different values with STEP to see how it can make a difference in the program.

The TI-99/4A computer can make two kinds of sound—musical tones and noise. The computer has a musical range of five octaves. A single program statement can play up to three tones at one time. This allows you to play chords.

Before entering a musical or noise program, make sure the sound on your TV or monitor is loud enough to hear! Also make sure the ALPHA LOCK key is in locked (down) position.

Try this tone:

There are three numbers in a CALL SOUND statement. They must be separated by commas and enclosed in parentheses. The first number sets the length or *duration* of the tone (how long it plays). The duration can range from 1 to 4250. The second number sets the pitch or the *frequency* of the tone (how high or low it plays). The frequency can range from 110 to 44733. The third number sets the loudness or *volume* of the tone. The volume can range from 0—the loudest, to 30—the quietest.

All the tones in the computer that are also available on the piano are listed on the chart on page 30. The computer can play many more notes than those listed.

To produce more than one tone per statement, study line 20 in the program. Notice the duration must be the same for all tones in one statement.

When this two-line program runs, one note (line 10) plays first, followed by a chord (line 20).

```
NEW
10 CALL SOUND (1000,440,2)
```



```
20 CALL SOUND (1000,
440,2,659,2,880,2)
RUN
```

The first number sets the duration for all three tones. The second number (440) is a tone followed by its volume (2). The fourth number (659) is another tone followed by its volume (2). The sixth number (880) is a third tone followed by its volume (2).

To make the computer produce noise, place a negative number from -1 to -8 in the frequency position of the CALL SOUND statement. The frequency position is the second number following CALL SOUND.

Try this example:

The CALL SOUND statement in line 30 makes the sound of a crash!

Here are more sound programs to try.*

```
30 CALL SOUND (500, -7,0)
RUN
```

Going Up

```
NEW
10 FOR X = 110 TO 1000 STEP 50
20 CALL SOUND( -100,X,0)
30 NEXT X
40 END
RUN
```

Hills and Valleys

```
NEW
10 FOR X = 110 TO 2000 STEP 50
20 CALL SOUND ( -100,X,0)
30 NEXT X
40 FOR X = 2000 TO 110 STEP -50
60 NEXT X
70 GOTO 10
RUN
```

Hills, Valleys, and Plains

```
NEW
10 FOR X = 110 TO 2000 STEP 50
20 CALL SOUND ( -100,X,0)
30 NEXT X
40 FOR X = 2000 TO 110 STEP -50
50 CALL SOUND ( -100,X,0)
60 NEXT X
70 CALL SOUND (1000, -7,0)
80 END
RUN
```

*For an explanation of FOR-NEXT and STEP, see pages 26-27.

The notes listed below represent all the keys on a piano. The computer can play many more.

FREQUENCY	NOTE	FREQUENCY	NOTE
110	A	740	F#, Gb
117	A#, Bb	784	G
123	B	831	G#, Ab
131	C (low C)	880	A (above high C)
139	C#, Db	932	A#, Bb
147	D	988	B
156	D#, Eb	1047	C
165	E	1109	C#, Db
175	F	1175	D
185	F#, Gb	1245	D#, Eb
196	G	1319	E
208	G#, Ab	1397	F
220	A (below middle C)	1480	F#, Gb
233	A#, Bb	1568	G
247	B	1661	G#, Ab
262	C (middle C)	1760	A
277	C#, Db		
294	D		
311	D#, Eb		
330	E		
349	F		
370	F#, Gb		
392	G		
415	G#, Ab		
440	A (above middle C)		
466	A#, Bb		
494	B		
523	C (high C)		
554	C#, Db		
587	D		
622	D#, Eb		
659	E		
698	F		

Texas Instruments *Beginner's Basic* manual can tell you even more about musical tones and noises.

Speech

Would you like to make your computer talk? You can! Connect the Speech Synthesizer unit and the Terminal Emulator II cartridge to your system.

Try this program to practice with speech. Make sure the ALPHA LOCK key is in locked (down) position.

When you RUN this program, each time you type something and press ENTER, the computer will say whatever you typed!

To change the pitch of the voice, type //20 100 (before you input the words you want the computer to say) and press ENTER. Then type a word, phrase, or sentence. If you stop the program and run it again, the voice returns to normal. The notation for the normal computer voice is //43 128. It is *preprogrammed* into the computer.

Try these different voices:

The first number sets the *pitch* of the voice (whether it is high or low). The range for pitch is 0 to 63. The second number sets the intonation or *slope* (how the voice rises and falls in a sentence). The range for slope is 0 to 255. For the best voice quality, the slope (second number) should be about three times the pitch (first number).

You can put speech into your own programs and use different voices, too! To use speech in a program, always start with an OPEN statement.

This tells the computer to "retrieve" speech from the speech file. Next, use a PRINT #1 statement followed by a colon. Then type in quotation marks what you want the computer to say.

PRINT #1 acts like a PRINT statement, except that the #1 tells the computer to speak the message, instead of printing it.

```
NEW


10 OPEN #1: "SPEECH",OUTPUT
20 INPUT "TYPE WHAT YOU WANT ME
TO SAY, THEN PRESS ENTER ":A$
30 PRINT #1: A$
40 GOTO 20
RUN
```



```
//00 100
//30 200
//60 150
//10 100
```
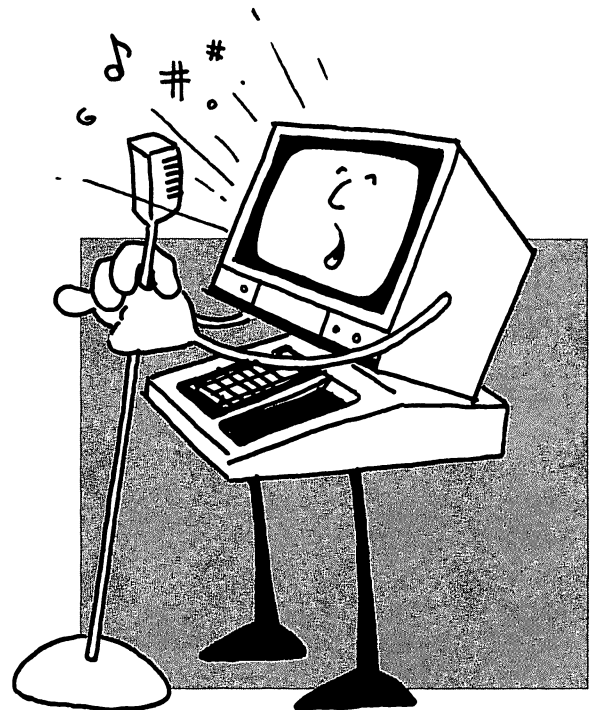
```
NEW
10 OPEN #1:"SPEECH", OUTPUT


20 PRINT #1:"DO YOU WANT ME
TO SAY SOMETHING?"
```

If you want to produce a different voice, put the voice notation in another PRINT #1 statement.

Then add a line to say something with the second voice:

It is a good idea to use a CLOSE statement in a program that contains an OPEN statement.

Experiment with different intonations in sentences using periods, commas, and question marks. Some very interesting sound effects can be created using speech in your programs! See the Terminal Emulator II manual for more about speech and intonation marks.

The TI-99/4A's "GOT THE BEAT!"

Musicians often use a small mechanical object that ticks. It is called a "metronome." It helps musicians keep correct time or rhythm to their music. This program will make a sound like a metronome.

You can alter the tempo of the beat. Change the value of DELAY in line 120 to "1 to 20." A value of 1 to 20 produces a very fast beat. A value of 1 to 200 produces a very slow beat. To stop this program, press FCTN 4 (CLEAR).

The sound of the click can also be changed. In place of QH, experiment with TT, or CK, or other combinations in line 110. Some combinations sound like percussion.

Now your computer's "got the beat!"

Experiment with other sounds formed by pressing the same two keys over and over again. Use the program above, but replace the letters in line 110 with QQQQQQ or RFRFRFRF.

```
30 PRINT #1:"//30 200"


40 PRINT #1:"SOMETHING"


50 CLOSE #1
60 END
RUN
```

```
NEW
100 OPEN #1:"SPEECH", OUTPUT
110 PRINT #1:"QH"
120 FOR DELAY = 1 TO 90
130 NEXT DELAY
140 GOTO 110
RUN
```

Sometimes a decision needs to be made in a program, for example, "Is A greater than B? Is B less than C?" and so on. With the IF-THEN statement, you can set up a test to answer these kinds of questions. Depending on the answer, the program may continue normally or branch to another line in the program.

This program demonstrates how IF-THEN works. The program counts by fives to fifty and prints the numbers on the screen. When it reaches 50, the program stops and prints a message: How does it know to stop at number 50?

Can you think of another way to write this program and produce the same results? Hint: You must use an ELSE statement. First read the following section, then try it! The answer is below.

```
NEW
10 CALL CLEAR
20 LET M=5
30 PRINT "M=";M
40 LET M=M+5
50 IF M<55 THEN 30
60 PRINT "I'M ALL LOOPED
OUT!"
70 END
RUN
```

Line 50 is an IF-THEN statement. It tests whether the value of M is less than 55. If so, the program branches to line 30 and prints the value of M+5. If the value of M is greater than or equal to 55, the program goes to line 60 and prints a message.

This program shows how ELSE can be used with IF-THEN. The computer can tell whether the number entered is a number from 1 to 10. To test this program, try entering negative numbers, decimal fractions, and numbers greater than ten.

```
NEW
10 INPUT "ENTER A NUMBER
FROM 1 to 10:":N
20 IF N>10 THEN 40
30 IF N<1 THEN 40 ELSE 60
```

Lines 20 and 30 set up the tests. If N is greater than 10, then the program goes to line 40 and prints a message. If not, the program proceeds to line 30. Line 30 is another test! If N is less than 1, the program goes to line 40. If not, it goes to line 60. As you can see, using ELSE can help your programs become more detailed.

```
40 PRINT "BAD NUMBER. TRY
AGAIN."
50 GOTO 10
60 PRINT "VERY GOOD!"
70 END
RUN
```

Change line 70 to 70 GOTO 10. This will let you play again, even when you get the correct answer!

Answer: Use an IF-THEN-ELSE statement in line 50 that reads IF M = 55 THEN 60 ELSE 30. There is usually more than one way to write a program.

The computer follows instructions in the form of numbered program lines. It looks for the lowest line number, executes it, then looks for the next consecutive number, performs it, and so on.

Can you unscramble the lines of this program? Put them in correct order the same way the computer would. The answers are in the Appendix on page 56.

170 CALL SOUND (500, −4,2)   _____

200 GOTO 200   _____

110 LET A$ = "995ABC3C3C3C2424"   _____

190 NEXT X   _____

100 CALL CLEAR   _____

130 CALL SCREEN(X)   _____

185 NEXT DELAY   _____

120 FOR X = 3 TO 16   _____

160 CALL VCHAR (1,X + 6,128,24)   _____

180 FOR DELAY = 1 TO 200   _____

140 CALL CHAR (128,A$)   _____

150 CALL HCHAR (X + 4,1,128,32)   _____

Can you guess what this program does?* Key it in and RUN it! Don't forget to use NEW! Helpful Hint: You can type the program line numbers in the order as they appear above. The computer will put them in the right order for you!

*See the section on CALL CHAR (p. 39) for more information about line 140.

Some programs need information stored within the program. You can assign a single value to each variable per LET statement. Using a READ and a DATA statement lets you assign values to several variables using fewer program lines.

This program shows how string variables are assigned values using the LET statement.

This program assigns values to the same variables using READ and DATA statements.

```
NEW
10 LET A$="JOE"
20 LET B$="MARY"
30 LET C$="BILL"

40 PRINT A$
50 PRINT B$
60 PRINT C$
70 END
RUN
```

Lines 10-30 assign values to string variables.

Lines 40-60 print the values of each string variable on the screen.

```
NEW
10 READ A$, B$, C$
```

Line 10 names the variables assigned to the values in the DATA statement in line 20: A$=JOE, B$=MARY, C$=BILL

```
20 DATA JOE, MARY, BILL
```

Line 20 identifies JOE, MARY, and BILL as the values (data) of each variable in the READ statement in line 10.

```
30 PRINT A$:B$:C$
RUN
```

Line 30 prints the data in a column. Remember—the colon prints one value per line.

String variables can contain numeric data, too.

```
NEW
10 READ A$,B$,C$
20 DATA JOE 10.50, MARY 12.75,
BILL 13.06
30 PRINT A$:B$:C$
40 END
RUN
```

READ and DATA statements can also be used to assign values to numeric variables. Remember—commas must separate numeric data items.

```
NEW
10 READ A,B,C
20 DATA 10.50,12.75,13.06
30 PRINT A:B:C
40 END
RUN
```

One variable name may be assigned to an entire line of data, as in the following example. Note that commas are not necessary to separate string data items.

```
NEW
10 READ A$
20 DATA JOE 10.50 MARY 12.75
BILL 13.05
30 PRINT A$
40 END
RUN
```

Numeric and string data can be contained in string variables, but only numeric data may be contained in numeric variables. Numeric data within a string variable cannot be used to perform mathematical operations.

For more about READ and DATA, see the *User's Reference Guide,* pages II-61–II-64.

Sometimes you may want the computer to choose a random number and use it in a program. To see how RND works, type PRINT RND and press ENTER. Do this several times. A random number is printed on the screen each time ENTER is pressed. The same numbers will always print unless you set a range for the numbers. If you do not set a range, the computer chooses only numbers between 0 and 1. If you want only whole numbers, use the INTeger function, as shown in the program below.

This is a "guessing game" program. The computer chooses a random number and you must guess what it is. In line 10, RND*10 sets the range for the random numbers from 0 to 9. In this program, the numbers 1 to 10 are desired. To use numbers from 1 to 10, the value of 1 is added to the integer produced in line 10.

```
NEW
NUM 10

10 X = INT (RND*10) + 1

20 INPUT "ENTER A NUMBER
BETWEEN 1 AND 10: ":N

30 IF N>X THEN 70
40 IF N<X THEN 90

50 PRINT "YOU GOT IT!"
60 GOTO 10
70 PRINT "YOUR GUESS IS
TOO HIGH"
80 GOTO 20
90 PRINT "YOUR GUESS IS TOO LOW"
100 GOTO 20

110 END
120

RUN
```

Type **NUM 10** and press ENTER. This tells the computer to print line numbers for you. They will begin with line 10 and increase by ten each time.

Line 10 generates a random number between 1 and 10.

Line 20 prints a message and a question mark, then waits for a number to be entered.

Lines 30 and 40 test the guess and branch to other parts of the program if the guess entered in line 20 is not correct.

Lines 50-100 display the results of the test in lines 30 and 40, and create loops in the program.

Press ENTER after typing line 110, and again after line 120 appears, to stop automatic numbering.

Run this program several times. Note that the *sequence* of numbers is always the same. To create random sequencing, add the statement RANDOMIZE before the RND statement.

Type: 5 RANDOMIZE.
(Press ENTER)

Now LIST, then RUN the program again.
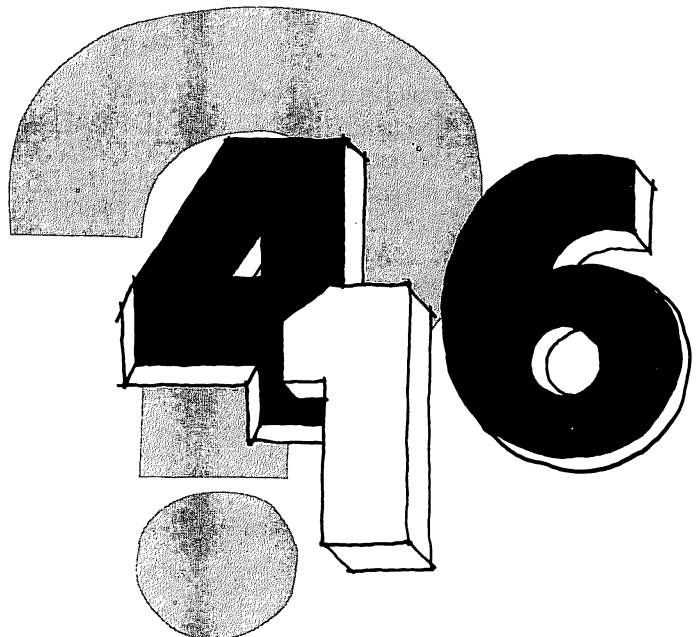
This graphics program uses RND and RANDOMIZE
to place stars on the screen.

```
NEW
NUM

100 CALL CLEAR

110 CALL SCREEN (2)

120 CALL COLOR (2,16,1)

130 RANDOMIZE


140 ROW = INT (RND*24) + 1

150 COL = INT (RND*32) + 1

160 CALL HCHAR (ROW,COL,42)

170 GOTO 140

180
(ENTER)

RUN

FCTN 4 (CLEAR)
```

NUM starts automatic line numbering with line 100 and increases by tens.

Line 110 calls up a black screen.

Line 120 produces white asterisks (character set #2).

Line 130 causes different sequences each time the program is executed.

Line 140 generates ROW numbers (1-24).

Line 150 generates COLUMN numbers (1-32).

Line 160 places a star on the screen.

Line 170 loops for another star.

After 180 appears, press ENTER to stop automatic numbering.

Executes the program.

Stops program execution.

With a GOSUB statement, you can make the computer branch to another section of a program. When you use GOSUB, a RETURN statement must always be included. When the computer finds a GOSUB statement, it branches to the line listed in the GOSUB statement. There it performs a routine, called a *routine*. When it comes to the RETURN statement, the program returns to the line following the GOSUB statement. Study the program below to understand how GOSUB and RETURN work, then RUN the program. Remember to use NUM if you want automatic line numbering.

```
NEW
100 GOSUB 400
```

Line 100 sends the program to a subroutine at line 400. The subroutine clears the screen. When the computer reaches the RETURN statement (line 410), the program returns to the line following the GOSUB statement (line 110).

```
110 PRINT "TO BEGIN A PROGRAMMING
SESSION IN TI BASIC, FOLLOW THESE
RULES:"
```

Line 110 prints a message.

```
120 GOSUB 300
```

Line 120 sends the program to the subroutine in line 300 (a FOR-NEXT loop causing a pause so that the message in line 100 remains on the screen for a few seconds). The RETURN statement in line 320 sends the program to line 130.

```
130 GOSUB 400
```

Line 130 sends the program to the CALL CLEAR subroutine in line 400.

```
140 PRINT "1) TURN ON ALL
PERIPHERALS." :::::
```

Line 140 prints a message.

```
150 GOSUB 300
160 GOSUB 400
170 PRINT "2) TURN ON THE
CONSOLE." :::::
180 GOSUB 300
190 GOSUB 400
200 PRINT "3) PRESS ANY
KEY TO BEGIN." :::::
210 GOSUB 300
220 GOSUB 400
230 PRINT "4) PRESS 1 FOR
TI BASIC." :::::
240 GOSUB 300
250 GOSUB 400
260 END
300 FOR DELAY = 1 TO 700
310 NEXT DELAY
320 RETURN
400 CALL CLEAR
410 RETURN

RUN
```

Line 150 sends the program to the subroutine in line 300 again, and the program continues in this order: lines 300, 310, 160, 400, 410, 170, 180, 300, 310, 190, 400, 410, 200, 210, 300, 310, 220, 400, 410, 230, 240, 300, 310, 250, 400, 410, and ends with line 260.
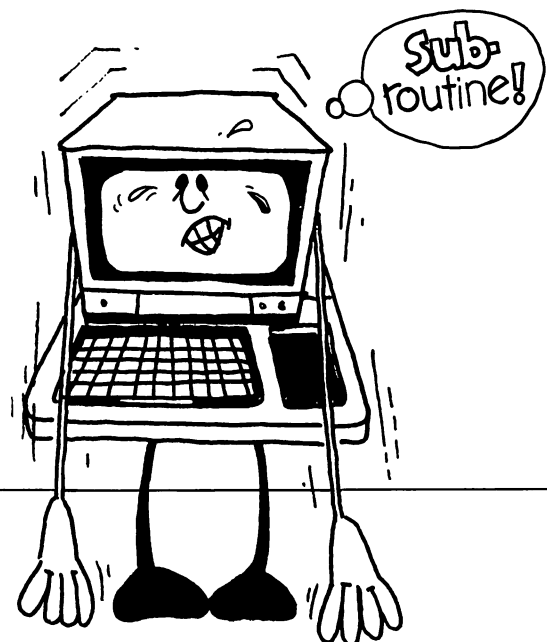
Subroutines are useful for saving space in a program. Instead of repeating routines over and over, you can put the routines in one section of the program. Then you can have the computer branch to it, and back from it, using GOSUB and RETURN.

With the CALL CHAR statement, you can create your own shapes!

Each letter or character on the screen is actually made up of dots 8 rows long and 8 rows wide. That means each letter or character you see on the screen is really made of 64 tiny dots, like the enlarged letter "A" below. (Also see the asterisk character on page 25.)

Each dot in the square can be turned on or off. To learn more about CALL CHAR, first study *Beginner's BASIC* (pages 108-117) or the *User's Reference Guide* (pages II-76-II-79). There is an exercise on page 43 of this book that will help you create your own character.

Try this program to see how CALL CHAR works in a program. The character produced looks like a small square inside another square.

```
NEW
10 CALL CLEAR
20 CALL VCHAR(3,16,96)
30 CALL CHAR
(96,"FFFFC3C3C3C3FFFF")
40 GOTO 40
RUN
```
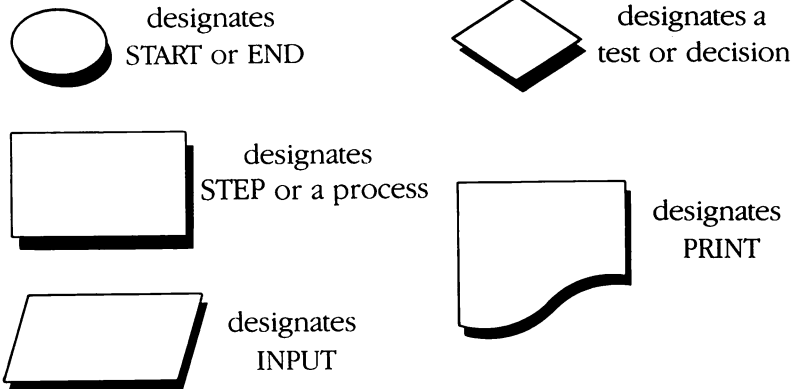
The CALL CHAR statement in line 30 changes the shape of character number 96 (a grave accent). It becomes a small square appearing in the upper center of your screen when the program is run.
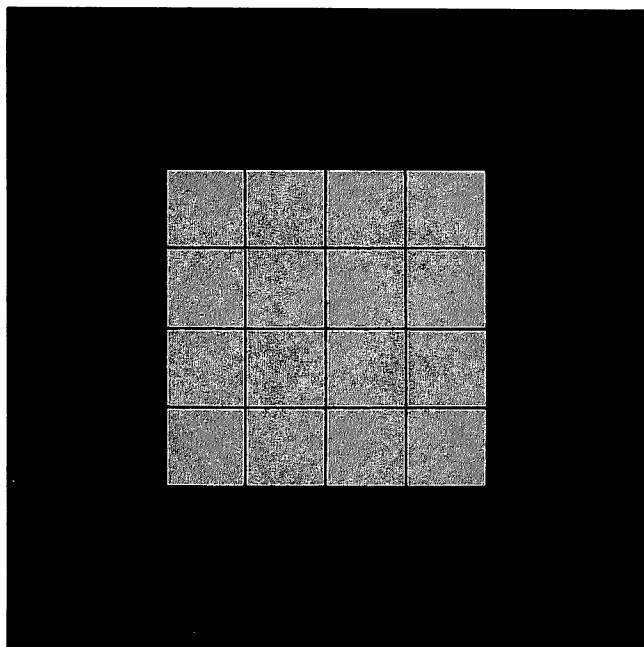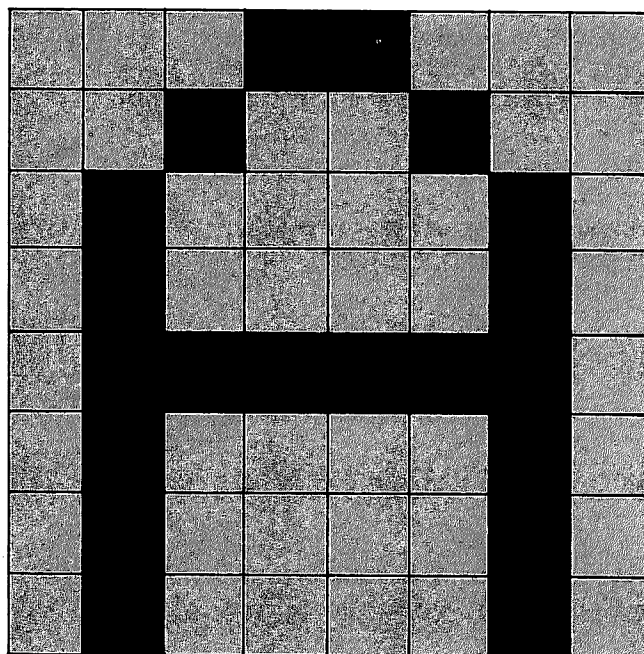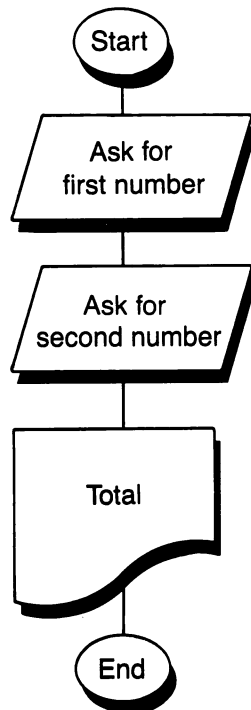
Programs you develop may grow in length. It may become more difficult to write them directly into the computer. It is helpful to analyze the problem first, then draw a master plan, or flowchart, to lay out each step in the program. A *flowchart* is a graphic representation of the steps necessary to solve a problem.

Follow all the possible paths in the flowcharts mapping the two simple programs listed.*

The common symbols used in flowcharting are:

designates START or END

designates a test or decision

designates STEP or a process

designates PRINT

designates INPUT

FLOWCHART 1:

Start

Ask for first number

Ask for second number

Total

End

FLOWCHART 2:

Start

Input A

Input B

A>B?

Yes → A is larger

No → B is larger
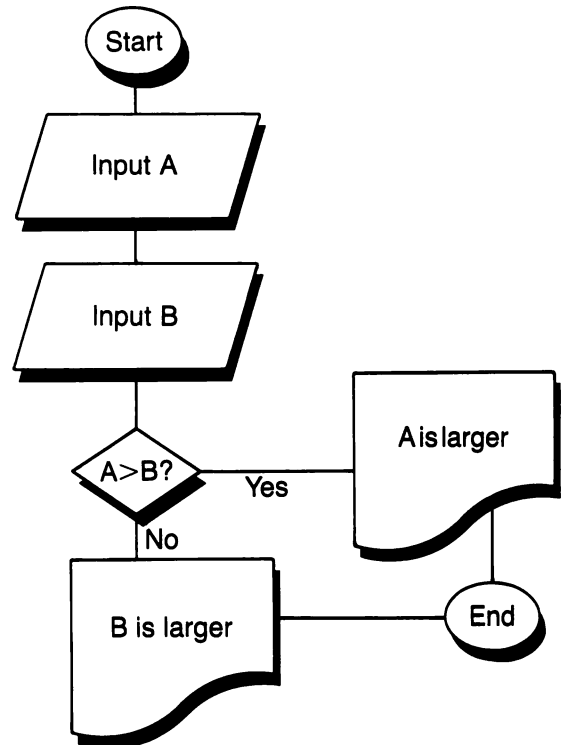
End

```
NEW
10 CALL CLEAR
20 "FIRST NUMBER ":A
30 INPUT "SECOND NUMBER?
":B
40 PRINT A+B
50 END
RUN
```

```
NEW
10 INPUT "ENTER A NUMBER
FOR A: ":A
20 INPUT "ENTER A NUMBER
FOR B: ":B
30 IF A>B THEN 60
40 PRINT "B IS LARGER"
50 GOTO 70
60 PRINT "A IS LARGER"
70 END
RUN
```

*For the purpose of these programs, A will not equal B.

Research Project:
Computer Applications

We know computers are used in airports, hospitals, factories, offices, stores, banks, schools, homes, and outer space. But what jobs do computers perform in these places? Interview your parents, friends, and teachers to learn how computers are used in these areas. You can ask questions like these, or make up your own:

Here are some sample programs in TI BASIC.

All you do is...

1. What kind of information do you give your computer?
2. How do you input your information to the computer?
3. What does the computer do with the information you give it?
4. How does the computer display the processed information?
5. How does a computer help you in your job?
6. When did you get your computer?
7. Who would do the work your computer does if you did not have a computer?

• Turn on the TV or monitor.
• Turn on the computer. There is a switch at the lower right front corner of the console. A red light shows when the computer is on.
• Press any key to begin.
• Press 1 for TI BASIC.
• Type the programs one line at a time. Don't forget to leave a space after line numbers.
• Press ENTER at the end of each line.
• When you finish entering the program lines, type RUN and press ENTER.*

Print Your Name!

```
NEW
10 CALL CLEAR
20 INPUT "TYPE YOUR NAME, THEN PRESS ENTER: ":A$
30 CALL CLEAR
40 PRINT "YOUR NAME IS"
50 PRINT A$::::::::::
60 FOR D = 1 TO 1000
70 NEXT D
80 END
RUN
```

Something with a Little TAB!

```
NEW
10 CALL CLEAR
20 INPUT "FIRST NUMBER PLEASE ":N1
30 INPUT "SECOND NUMBER PLEASE ":N2
40 PRINT TAB(8); N1
50 PRINT TAB(14); N2
60 END
RUN
```

Change line 40 to:

```
40 PRINT TAB(8);N1;TAB(14);N2
```

*If you receive an error message on the screen (for example, *ERROR IN LINE 10), type LIST and press ENTER. The program will appear on the screen. Check each line very carefully for mistakes. Turn to pages 52-55 for more information on correcting errors.

Explore Colors!

```
100 CALL CLEAR
110 FOR X = 1 TO 16
120 Y = Y + 8
130 CALL COLOR(X,X,X)
140 CALL HCHAR(16,X + 7,Y + 30,2)
150 FOR D = 1 TO 300
160 NEXT D
170 NEXT X
180 GOTO 180
```

Explore Division!

```
10 CALL CLEAR
20 PRINT "WATCH ME DIVIDE!"
30 LET X = X + 1
40 LET Y = X/4
50 PRINT X;" DIVIDED BY 4 = ";Y
60 GOTO 30
```

Explore Multiplication!

```
NEW
10 CALL CLEAR
20 PRINT "365 DAYS IN A YEAR"::::::::::
30 INPUT "HOW OLD ARE YOU?":A
40 IF A<3 THEN 120
50 IF A>16 THEN 120
60 CALL SCREEN(A)
70 PRINT "YOU ARE MORE THAN":A*365;"DAYS OLD!"
80 PRINT "YOU ARE MORE THAN":A*12; "MONTHS OLD!"
90 FOR DELAY = 1 TO 1000
100 NEXT DELAY
110 GOTO 10
120 PRINT "USE A NUMBER BETWEEN 3 AND 16.
TRY AGAIN!"
130 GOTO 30
RUN
```

Explore Shapes!

Create a jet plane:

```
NEW
100 CALL CLEAR
110 CALL CHAR(128,"000000010F3FFFFF")
120 CALL CHAR(129, "000003FFFFFFFFFF")
130 CALL CHAR(130,077FFFFFCF8FFFFF8")
140 CALL HCHAR(10,21,128,1)
150 CALL HCHAR(10,22,129,1)
160 CALL HCHAR(10,23,130,1)
RUN
```

Make it move:

Add:	115 FOR X = 1 TO 20
Edit:	140 CALL HCHAR(10,21 – X,128,1)
	150 CALL HCHAR(10,22 – X,129,1)
	160 CALL HCHAR(10,23 – X,130,1)
Add:	170 FOR DELAY = 1 TO 25
	180 NEXT DELAY
	190 CALL CLEAR
	200 NEXT X
	210 END

Change the color of the jet:

Add:	105 CALL COLOR(13,16,1)

Explore Speech!

Remember to use the Speech Synthesizer unit and the Terminal Emulator II cartridge.

```
NEW
10 OPEN #1: "SPEECH", OUTPUT
20 INPUT A$
30 PRINT #1:A$
40 GOTO 20
RUN
```

Explore Music!

```
NEW                        100 NOTE = C
10 CALL CLEAR              110 GOTO 800
15 LET C = 262            200 NOTE = D
20 LET D = 294            210 GOTO 800
25 LET E = 330           300 NOTE = E
30 LET F = 349           310 GOTO 800
35 LET G = 392           400 NOTE = F
40 LET A = 440           410 GOTO 800
45 LET B = 494           500 NOTE = G
50 INPUT "NOTE":A$       510 GOTO 800
55 IF A$ = "C" THEN 100  600 NOTE = A
60 IF A$ = "D" THEN 200  610 GOTO 800
65 IF A$ = "E" THEN 300  700 NOTE = B
70 IF A$ = "F" THEN 400  710 GOTO 800
75 IF A$ = "G" THEN 500  800 CALL SOUND
80 IF A$ = "A" THEN 600  (100, NOTE, 1)
85 IF A$ = "B" THEN 700  RUN
90 GOTO 50
```

Make your own shape using CALL CHAR*:
- Write a simple program to get the shape onto the screen.
- Make the shape move on the screen.
- Give the shape color.

CHAR Worksheet

	LEFT BLOCK	RIGHT BLOCK	CODE
ROW 1			—
ROW 2			—
ROW 3			—
ROW 4			—
ROW 5			—
ROW 6			—
ROW 7			—
ROW 8			—

Any Row

left block   right block

BLOCKS	DOT CODE (0 = off; 1 = on)	SHORT-HAND CODE
	0000	0
	0001	1
	0010	2
	0011	3
	0100	4
	0101	5
	0110	6
	0111	7
	1000	8
	1001	9
	1010	A
	1011	B
	1100	C
	1101	D
	1110	E
	1111	F

CALL CHAR   (⌐ , " ⌐ The shorthand code goes here. ")
⌐ The character code number goes here.

High/Low Game

The computer will generate a random number between 1 and 100. You guess the number!

```
NEW
10 RANDOMIZE
20 CALL CLEAR
30 LET A = INT (RND*100) + 1
40 PRINT "I AM THINKING OF A NUMBER"
45 PRINT "BETWEEN 1 AND 100."
50 C = 0
60 PRINT::
70 INPUT "YOUR GUESS? ":G
80 C = C + 1
90 IF G>A THEN 200
100 IF G<A THEN 220
110 PRINT "YOU GOT IT"
120 PRINT "IT TOOK YOU"; C;" GUESSES."
130 GOTO 30
200 PRINT "TRY A SMALLER NUMBER"
210 GOTO 60
220 PRINT "TRY A LARGER NUMBER"
230 GOTO 60
RUN
```

Options and modifications:

• Add music and color if the answer is correct.
• Add text-to-speech.
• Add an INPUT statement that allows you to choose the maximum range for RND.
• Change the "try again" messages (lines 200 and 220).

Matchstick game. (Just try to win!!)

```
10 PRINT "WE START WITH 21 MATCHES. WE WILL TAKE TURNS"
20 PRINT "REMOVING MATCHES: UP TO FOUR PER TURN."
30 PRINT "IF YOU HAVE TO PICK UP THE LAST MATCH—YOU LOSE."
40 N = 21
50 INPUT "HOW MANY WILL YOU TAKE? ":I
60 IF I<1 THEN 500
70 IF I>4 THEN 500
80 N = N - 5
90 PRINT "I TAKE"; 5 - I;" THAT LEAVES";N
100 IF N>1 THEN 50
110 PRINT "YOU TAKE THE LAST ONE. YOU LOSE."
120 END
500 PRINT "DON'T CHEAT!! TAKE 1, 2, 3 OR 4 ONLY."
510 GOTO 50
```

Options and modifications:

• Add a loop that asks if you would like to play again.
• Add text-to-speech.
• Add color, music, or a frame around the screen.

Graphics

In this program, the computer asks for three numbers. The first two numbers are the dimensions of a block of asterisks. The program surrounds the block with a frame of 0's. The third number is the width of the frame.

```
10 INPUT "WIDTH ":W
20 INPUT "HEIGHT ":H
30 INPUT "FRAME ":F
40 CALL CLEAR
50 FOR A = 1 TO F
60 FOR B = 1 TO (F + W + F)
70 PRINT "O";
80 NEXT B
90 PRINT
100 NEXT A
110 FOR C = 1 TO H
120 FOR D = 1 TO F
130 PRINT "O";
140 NEXT D
150 FOR E = 1 TO W
160 PRINT "*";
170 NEXT E
180 FOR G = 1 TO F
190 PRINT "O";
200 NEXT G
210 PRINT
220 NEXT C
230 FOR I = 1 TO F
240 FOR J = 1 TO (F + W + F)
250 PRINT "O";
260 NEXT J
270 PRINT
280 NEXT I
290 END
```

For width = 5, height = 7, and frame = 2, the output would be:

```
000000000
000000000
00*****00
00*****00
00*****00
00*****00
00*****00
00*****00
00*****00
000000000
000000000
```

Options:

• Add color to characters and screen.

To run this program, a Terminal Emulator II cartridge and a Speech Synthesizer unit must be attached to the computer system. When this program runs, three number inputs are requested (lines 120, 170, 220). The color codes input will appear graphically on the screen and are described with speech. Input is requested again. If the color code for black (2) or transparent (1) is entered for screen color, the prompts for the next color code inputs will not display at the bottom of the screen. Inputing three color codes will cause the program to continue.

```
100 OPEN #1: "SPEECH", OUTPUT
110 CALL CLEAR
120 INPUT "SCREEN COLOR?: ":S
130 PRINT #1:S
140 T = S
150 GOSUB 400
160 S$ = T$
170 INPUT "FOREGROUND?: ":F
180 PRINT #1:F
190 T = F
200 GOSUB 400
210 F$ = T$
220 INPUT "BACKGROUND?: ":B
230 PRINT #1:B
240 T = B
250 GOSUB 400
260 B$ = T$
270 CALL CLEAR
280 CALL SCREEN(S)
290 CALL COLOR(2,F, B)
300 CALL HCHAR(12,3,42,28)
310 PRINT #1: "THE SCREEN COLOR IS "
320 PRINT #1: S$
330 PRINT #1: "THE FOURGROUND COLOR IS "
340 PRINT #1:F$
350 PRINT #1: "AND THE BACKGROUND COLOR IS "
360 PRINT #1:B$
370 CALL SOUND(75,262,0,330,0,392,0)
380 CALL SOUND(1000,262,0,330,0,392,0)
390 GOTO 120
400 ON T GOTO 410,430,450,470,490,510,530,550,570,590,
    610,630,650,670,690,710
410 T$ = "TRANSPARENT"
420 RETURN
430 T$ = "BLACK"
440 RETURN
450 T$ = "MEEDEEUM GREEN"
460 RETURN
470 T$ = "LIGHT GREEN"
480 RETURN
490 T$ = "DARK BLUE"
500 RETURN
510 T$ = "LIGHT BLUE"
520 RETURN
530 T$ = "DARK RED"
540 RETURN
550 T$ = "SIGH + ANN"
560 RETURN
570 T$ = "MEEDEEUM RED"
580 RETURN
590 T$ = "LIGHT RED"
600 RETURN
610 T$ = "DARK YELLOW"
620 RETURN
630 T$ = "LIGHT YELLOW"
640 RETURN
650 T$ = "DARK GREEN"
660 RETURN
670 T$ = "MUHGENTA"
680 RETURN
690 T$ = "GRAY"
700 RETURN
710 T$ = "WHITE"
720 RETURN
```

To run this program, the Terminal Emulator II cartridge and Speech Synthesizer Unit must be attached. This program produces a graphic simulation of a space shuttle lift-off, much like an artist's drawing you might see on a television broadcast. It includes music and speech. Here is a breakdown of the program.

```
10 CALL CLEAR
20 ROW=20
30 COL=3
```

Lines 10-30 clear the screen and set up the shape of the space shuttle.

```
100 REM LAUNCH PAD
110 CALL COLOR (3,15,15)
120 CALL COLOR (10,7,15)
130 CALL COLOR (8,7,15)
135 CALL COLOR (11,7,15)
140 CALL HCHAR (19,6,110)
150 CALL HCHAR (20,6,94)
160 CALL HCHAR (21,6,115)
170 CALL HCHAR (22,6,94)
180 CALL HCHAR (19,5,55)
190 CALL HCHAR (23,3,55,4)
```

Lines 100-190 set up the color and shape of launch pad.

```
200 REM ATLANTIC OCEAN
210 CALL COLOR (4,6,5)
220 CALL VCHAR (4,32,60,20)
230 CALL VCHAR (6,31,60,18)
240 CALL VCHAR (6,30,60,18)
250 CALL VCHAR (7,29,60,17)
260 CALL VCHAR (10,28,60,14)
270 CALL VCHAR (12,27,60,10)
280 CALL VCHAR (13,26,60,7)
290 CALL VCHAR (15,25,60,5)
300 CALL VCHAR (17,24,60,5)
310 CALL VCHAR (20,23,60,3)
320 CALL VCHAR (22,22,60,2)
```

Lines 200-320 set up the Atlantic Ocean.

```
330 GOSUB 7000
```

Line 330 "calls" a subroutine (lines 7000-7170) which places the name of the ocean on the screen.

```
340 GOSUB 2000
```

Line 340 "calls" a subroutine (lines 2000-2150) which sets up the shape of the shuttlecraft on the launch pad.

```
400 REM COUNTDOWN
410 OPEN #1: "SPEECH", OUTPUT
420 PRINT #1: "FINAL COUNT DOWN
FOR SPACE SHUTTLE LIFT OFF
COMMENCING."
430 PRINT #1: "TEN SECONDS AND
COUNTING."
440 PRINT #1: "9"
450 GOSUB 1200
460 PRINT #1: "8"
470 GOSUB 1200
480 PRINT #1: "7. ALL SYSTUHMS ARE GO."
490 PRINT #1: "6"
500 GOSUB 1200
510 PRINT #1: "5. UH WE HAVE BOOSTER
FIRE."
520 PRINT #1: "4"
```

Lines 400-520 set up a spoken countdown from 10 to 4, with comments.

```
530 GOSUB 1000
```

Line 530 "calls" a subroutine (lines 1000-1230) that makes music.

```
550 REM BLAST OFF
560 CALL SOUND (4000, - 7,0)
570 CALL SOUND (4000, - 7,0)
580 ROW = 14
590 COL = 3
600 GOSUB 2000
610 CALL SOUND (4000, - 7,0)
620 ROW = 7
630 COL = 4
640 GOSUB 3000
650 CALL SOUND (2500, - 7,5)
660 ROW = 3
670 COL = 10
680 GOSUB 3000
690 CALL SOUND (2000, - 7,10)
700 ROW = 2
710 COL = 17
720 GOSUB 4000
730 CALL SOUND (2000, - 7,15)
740 ROW = 1
750 COL = 24
760 GOSUB 4000
770 CALL SOUND (2000, - 7,18)
780 ROW = 1
790 COL = 30
800 GOSUB 4000
810 CALL SOUND (3000, - 7,25)

820 GOSUB 5000


830 GOTO 830


1000 REM MUSIC TO LIFT OFF BY
1010 CALL SOUND (1500,131,10)
1020 CALL SOUND (1500,196,10)
1030 CALL SOUND (1500,262,10)
1040 CALL SOUND (250,311,5,262,5,196,5)
1050 CALL SOUND (1500,330,5,262,5,196,5)
1060 RETURN
1200 REM DELAY SUBROUTINE
1210 FOR D = 1 TO 300
1220 NEXT D
1230 RETURN
2000 REM SPACE SHUTTLE
2020 CALL COLOR (13,16,1)
2030 CALL HCHAR (ROW,COL,128)
2040 CALL CHAR (128, "01030307070707")
2050 CALL HCHAR (ROW, COL + 1,129)
2060 CALL CHAR (129, "80C0C0E0E0E0E0E0")
2070 CALL HCHAR (ROW = 1, COL, 130)
2080 CALL CHAR (130, "0F0F1F1F3F3F7F7F")
2090 CALL HCHAR (ROW + 1, COL + 1, 131)
2100 CALL CHAR (131, "F0F0F8F8FCFCFEFE")
2110 CALL HCHAR (ROW + 2, COL, 132)
2120 CALL CHAR (132, "FFFFFFFFFF070707")
2130 CALL HCHAR (ROW + 2, COL + 1, 133)
```

Lines 550-830 begin the blast-off including noise and shuttlecraft shapes. A total of five or more shapes appear on the screen. They chart the path of the shuttlecraft. Three different shapes simulate rotation of the craft as it climbs.

Line 820 calls a subroutine (lines 5000-7170) which produces spoken comments on the lift-off.

Line 830 is the last line of the program. It holds the completed graphic images on the screen.

```
2140 CALL CHAR (133, "FFFFFFFFFFE0E0E0")
2150 RETURN
3000 REM SHUTTLE #2
3010 CALL COLOR (14,16,1)
3020 CALL HCHAR (ROW, COL + 1, 136)
3030 CALL CHAR (136, "0000000000010307")
3040 CALL HCHAR (ROW, COL + 2, 137)
3050 CALL CHAR (137, "00000030F8F8F0E0")
3060 CALL HCHAR (ROW + 1, COL, 138)
3070 CALL CHAR (138, "00000001071F3F7F")
3080 CALL HCHAR (ROW + 1, COL + 1, 139)
3090 CALL CHAR (139, "0F1F7FFFFFFEFCFC")
3100 CALL HCHAR (ROW + 1,COL + 2,140)
3110 CALL CHAR (140, "E0C08000000000000")
3120 CALL HCHAR (ROW + 2,COL,141)
3130 CALL CHAR (141, "7F1F0F0707070301")
3140 CALL HCHAR (ROW + 2,COL + 1,142)
3150 CALL CHAR (142, "F8F8F0F0E0E0E000")
3160 RETURN
4000 REM SHUTTLE #3
4010 CALL COLOR (15,16,1)
4020 CALL HCHAR (ROW,COL,144)
4030 CALL CHAR (144, "E0FE7F3F1FFFFF00")
4040 CALL HCHAR (ROW,COL + 1,145)
4050 CALL CHAR (145, "0000C0FFFFFFFF00")
4060 CALL HCHAR (ROW,COL + 2,146)
4070 CALL CHAR (146, "00000080F0FCFFFF")
4080 RETURN
5000 REM CONVERSATION
5010 PRINT #1: "SHUTTLECRAFT IS NOW SIXTY
MILES OUT OVER THEE ATLANTIC OH SHUN."
5020 PRINT #1: "HEW STON CONTROL CON-
GRATULATIONS ON THAT FINE LIFT OFF.
TURNING CONTROL OVER TO YOU, OVER."
5030 CLOSE #1
5040 RETURN
7000 REM NAME THE ATLANTIC
7010 CALL COLOR (5,3,5)
7020 CALL COLOR (6,3,5)
7030 CALL COLOR (7,3,5)
7040 CALL HCHAR (8,32,65)
7050 CALL HCHAR (9,32,84)
7060 CALL HCHAR (10,32,76)
7070 CALL HCHAR (11,32,65)
7080 CALL HCHAR (12,32,78)
7090 CALL HCHAR (13,32,84)
7100 CALL HCHAR (14,32,73)
7110 CALL HCHAR (15,32,67)
7120 CALL HCHAR (17,32,79)
7130 CALL HCHAR (18,32,67)
7140 CALL HCHAR (19,32,69)
7150 CALL HCHAR (20,32,65)
7160 CALL HCHAR (21,32,78)
7170 RETURN
```

Programming Challenge #1

Do you have a job or get an allowance from your parents for doing odd jobs? Write a program that will allow you to input how much an hour you make, how many hours you work, and during a certain period of time. Your program will consider this input and tell you how much money you can earn in a month, year, or longer.

Programming Challenge #2
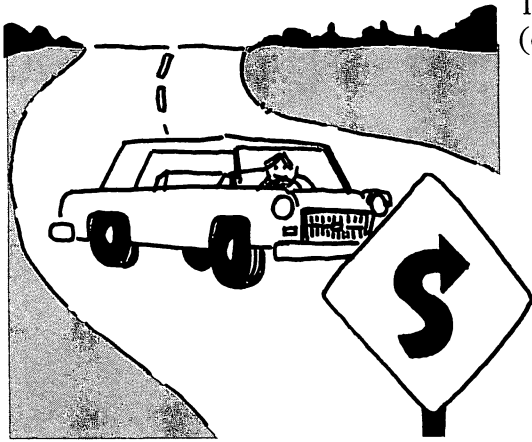
Are you old enough to think about getting your driver's license? To prepare for the written test, write a program that will quiz you (or your friends) on the information in the driver's manual.

Programming Challenge #3
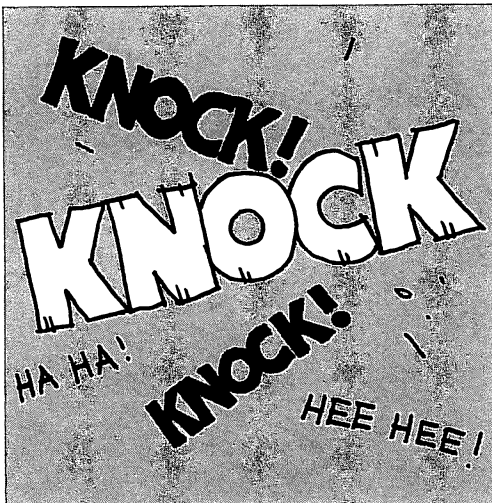
Do you like "knock-knock" jokes? Try writing programs that will tell jokes. Extra special challenge: Try using two or more different voices. You will find the Terminal Emulator II cartridge helpful for this.

*Answers to Programming Challenges are in the Appendix on page 56.

Program Suggestions:

1. Give a title screen to your program using graphics.

2. For each set of scores to be entered, give directions on the screen telling which numbers to input!

Can you think of other team averages you would like to calculate with the help of your computer? Can you adapt this program to another use? Go ahead, try it!
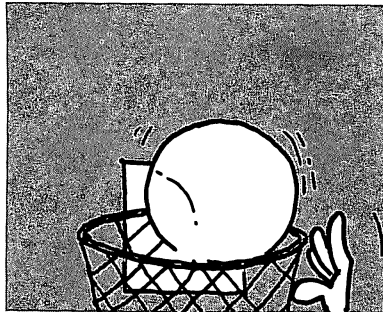
Programming Challenge #4: Computers in Sports

If rushing, pitching, batting, and scoring averages are your "name," computers can help you with your "game!"

These are the basketball scores for Tech High's girls' and boys' basketball teams. Your job is to write a program that will produce a scoring average for each player.

TEAM:	RAVING RAVENS GIRLS' TEAM											

| NAME | GAMES | | | | | | | | | | | |
	1	2	3	4	5	6	7	8	9	10	11	12
Gail Gibley	10	6	6	7	6	8	8	10	12	8	8	8
May Smith	2	3	5	2	2	4	0	1	0	4	4	6
Robbyn Birnbaum	2	4	6	7	6	5	3	4	9	6	5	4
Jenny Medford	5	0	0	6	4	2	2	8	4	4	3	4
"Squawk" Gilmore	10	14	7	16	18	18	16	15	19	24	12	26
Ellen McPeek	7	12	14	11	10	9	12	14	14	16	14	14
Liz "The Fin" Lafin	3	8	10	8	7	8	4	4	4	2	6	4
"Mean" Jeanie Feeney	18	20	17	22	19	12	18	18	21	23	26	27

TEAM:	TECH TROTTERS BOYS' TEAM											

| NAME | GAMES | | | | | | | | | | | |
	1	2	3	4	5	6	7	8	9	10	11	12
John "The Voice" Tapley	4	10	7	5	7	8	5	2	5	6	8	2
"Scooter" C. Brannon	4	12	14	13	8	10	14	16	14	12	11	10
Harry Matawaski	18	8	22	19	14	16	25	16	12	0	8	2
"Shrugs" Gilmore	10	7	9	12	10	8	14	3	18	16	12	22
"Boston" Newman	3	7	2	4	5	7	8	3	4	9	10	8
Ron Kelso	6	3	1	2	2	4	6	2	9	2	3	5
Bobby "Mac" McKinley	7	0	3	4	4	6	2	2	4	2	6	2
Pat Forbes	6	5	4	7	10	4	2	1	7	6	4	1
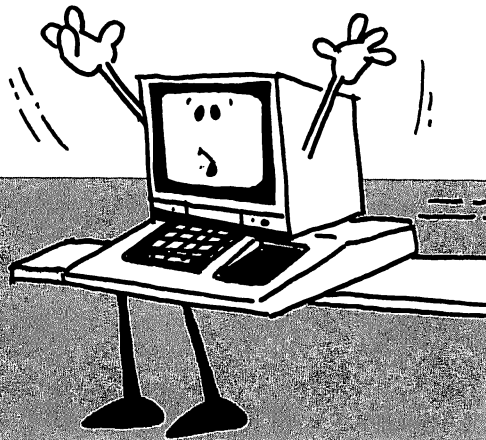
Programs being entered into the computer or programs already written and loaded into the computer can be edited for corrections or modified to change the output of a program.

The TI-99/4A computer has several functions which can shortcut program writing and editing time. In becoming familiar with these functions, first review the keyboard functions listed on the inside front cover of this book.

To clear the screen of unwanted clutter:

CALL CLEAR *(ENTER)*	Clears the screen, but not the memory.

To recall a program from memory:

LIST *(ENTER)*	Displays program in memory on the screen.

To execute a program in memory:

RUN *(ENTER)*	Causes the program in memory to execute.

To stop program execution:

FCTN 4 (CLEAR) *BREAKPOINT AT 5	Program will cease running and print a BREAKPOINT message, communicating the line at which the program was stopped.

Before starting a new program, and before leaving a programming session, the memory should be cleared. To erase a current program from memory:

NEW *(ENTER)*	NEW will clear the computer's memory and return to the TI BASIC READY screen.
BYE *(ENTER)*	BYE will clear the computer's memory and return to the TI Master Title Screen, leaving TI BASIC. Using BYE is the preferred method of ending a programming session before turning off the computer system.

EDITING BY LINE

If an error is made while typing a program line, use the left arrow key (FCTN S) to backspace to the error. Correct the error by typing over it, then press ENTER, or press FCTN 3 (ERASE) to erase the entire line, then type the line (including the line number) correctly, and press ENTER. Use the space bar, FCTN 1 (DEL), FCTN 2 (INS), FCTN S (◀), FCTN D (▶), to make corrections. Refer to the following sections for more information on how to use these features.

If an error is detected in a program line already in memory, follow the directions given below.

To delete or change the contents of a program line:

EDIT 10
(ENTER)

Type EDIT 10 (this example is from the first program line in the INPUT section) and press ENTER. Line 10 and its contents appear on the screen.

OR

10
(FCTN X)

Optionally, type the desired line number and press FCTN X. Line 10 and its contents appear on the screen.

Using either of these methods brings the required line to the screen for alterations. Use the space bar, FCTN 1 (DEL), FCTN 2 (INS), FCTN S (◀), FCTN D (▶), to make corrections. Refer to the following sections for more information on how to use these features.

10 PRINT "THANK YOU FOR YOUR INPUT, JAMIE."

The flashing cursor appears on the first character of the statement following the line number.

If no change is desired in this line:

Press ENTER.

To change the contents of this line:

Use the arrow keys (FCTN S, FCTN D) to move the cursor to the point where change is desired and type over whatever is to be changed. Use the space bar to erase excess characters. Press ENTER to enter the line into the computer's memory.

To erase the contents of this line and enter other information:

Press FCTN 3 (ERASE), then type the information desired. Optionally, the DELete and INSert functions can also be employed (see the section below on DELete and INSert).

To erase this entire line from the program:

Press FCTN 3 (ERASE), then press ENTER. Optionally, instead of typing EDIT 10 and pressing ENTER to call line 10 to the screen, type 10, press ENTER, and the entire line will be erased from the computer's memory.

To remain in Edit Mode and view or edit other lines:

Press FCTN X (�) to view lines in ascending order, or FCTN E (▲) to view lines in descending order.

To leave Edit Mode and effect changes made while in Edit Mode:

Press ENTER.

To leave Edit Mode and not effect changes made while in Edit mode:

Press FCTN 4 (CLEAR).

DELETE AND INSERT

To delete one or more characters within a line:

Move the cursor using the arrow keys (FCTN S, FCTN D). Place the cursor on the character to be deleted. Press FCTN 1 (DEL) once for each character to be removed. The character(s) will disappear and adjustment will be made automatically for the space remaining after the character(s) are deleted. Use an arrow key or press ENTER to exit the DELete function.

To insert one or more characters within a line:

Move the cursor using the arrow keys (FCTN S, FCTN D) on the character appearing to the right of the point where insertion is to occur. Press FCTN 2 (INS), release both keys, and type the desired character(s) to be added. Adjustment will automatically be made to provide space for the insertion of the character(s). Use an arrow key or press ENTER to exit the INSert function.

AUTOMATIC NUMBERING

The computer can be instructed to automatically provide numbers for program lines by use of Number Mode.

For automatic line numbering:

NUM
(ENTER)

NUM should be input before beginning a program. NUM will start line numbering at 100 and increase by increments of 10.

NUM 10
(ENTER)

NUM X will start line numbering at X and increase by increments of 10. In this case, numbering will begin with line 10 and continue in increments of 10.

NUM 5,5
(ENTER)

NUM X,Y will start line numbering at line X and increase by increments of Y. In this case, numbering will begin at line 5 and increase by increments of 5.

To insert automatically-numbered lines in an existing program:

Use NUM X,Y, X represents the point at which numbering should begin, and Y represents the incrementation.

To terminate automatic line numbering:

Press ENTER twice, first to enter your last valid program line, then again immediately after the next generated line number is displayed. The empty program line will not appear in the program.

RESEQUENCING LINE NUMBERS

If program line numbers are not evenly incremented after editing, the program line numbers can be automatically renumbered.

To resequence program line numbers by 10's starting with 100:

RES
(ENTER)
LIST
(ENTER)

Type RES, press ENTER, then LIST the program. The program lines will be renumbered automatically by 10's beginning with 100, including line numbers referenced in any GOTO statement.

To resequence program line numbers by 10's starting with 10:

RES 10
(ENTER)
LIST
(ENTER)

Type RES 10, press ENTER, then LIST the program. Resequencing begins with line 10. Line numbering in increments of 10 is standard for programs. It allows later insertion of program steps at specific points in a program.

To resequence program line numbers by X with increments of Y:

RES 5,5
(ENTER)
LIST
(ENTER)

Resequencing will begin with X and increase by Y, or any numbers inserted into this format. In the example, resequencing will begin with line number 5, followed by line 10, 15, 20 and so on.

LIST, RUN, AND TRACE

It is often desirable to view program lines in memory, especially after several edits have been performed.

To view a program in memory:	**LIST** *(ENTER)*	Displays the program in memory.
If a program is more than 24 lines in length (a full screen), it is sometimes desirable to view only certain portions of the program at one time.	**LIST 100-200** *(ENTER)*	Lists lines 100-200.
	LIST 200- *(ENTER)*	Lists the program beginning with line 200.
	LIST -300 *(ENTER)*	Lists the program from the beginning through line 300.
The LIST command can also access peripherals.	**LIST RS232** *(ENTER)*	Causes a program in memory to print out on a printer.
Typing RUN and pressing ENTER causes a program to execute beginning with the first program line. When editing a program, it is often desirable to stop a program at a certain point during execution and restart it from that point, or from another point, to access certain areas.	*FCTN 4 (CLEAR)*	Stops program execution.
	CONTINUE *(ENTER)*	Restarts program execution from the line at which the program was stopped.
	RUN 300 *(ENTER)*	Causes a program to begin running from the line specified.
The TRACE command is another feature that can aid in the editing process. When TRACE is entered, and a program is run or continued, program line numbers are printed on the screen just before they are executed in the program.	**TRACE** *(ENTER)*	Causes the TRACE feature to be activated.
	RUN *(ENTER)*	Executes the program. CONTINUE may also be used when applicable.
	UNTRACE *(ENTER)*	Causes the TRACE feature to be deactivated. NEW or BYE will also exit the TRACE command.
	RUN	

Match-Up (p.16)

1. d	4. j	7. i	9. g
2. f	5. h	8. a	10. c, e
3. b	6. e, c		

Practice with GOTO (p. 18)

```
1. 10 PRINT "XXXXXXXXXXXXXXXXXXXXXX
      XXXXXXXXXX "
   20 GOTO 10
2. 10 PRINT "XXXXXXXXXXXXXXXXXXXXXX
      XXX :::::::::::::::::::::::"
   20 GOTO 10
3. 10 PRINT TAB (16);"XXXXXXXXXXXXXXX"
   20 GOTO 10
4. 10 CALL CLEAR
   20 PRINT "XXX..."  (Type 84 X's for line 20.)
   30 PRINT "XXX..."  (Type 84 X's for line 30.)
   40 PRINT "XXX..."  (Type 84 X's for line 40.)
   50 PRINT "XXX..."  (Type 84 X's for line 50.)
   60 GOTO 60
```

Line Numbers (p. 36)

```
100 CALL CLEAR
110 LET A$ = "995ABC3C3C3C2424"
120 FOR X = 3 TO 16
130 CALL SCREEN (X)
140 CALL CHAR (128,A$)
150 CALL HCHAR (X + 5,1,12878,32)
160 CALL VCHAR (1, X + 3,128,24)
170 CALL SOUND (500, - 4,2)
180 FOR DELAY = 1 TO 200
185 NEXT DELAY
190 NEXT X
200 GOTO 100
```

Programming Challenge #1 (p. 53)

```
10 CALL CLEAR
20 INPUT "HOW MUCH AN HOUR DO YOU MAKE?":D
30 INPUT "HOW MANY HOURS DO YOU PLAN TO WORK
IN A WEEK?":HR
40 INPUT "HOW MANY MONTHS DO YOU PLAN TO
WORK AT THIS RATE?":MO
50 CALL CLEAR
60 LET TOTAL = D*HR*4*MO
70 PRINT "YOU WILL EARN $";TOTAL;"DOLLARS
IN";MO;"MONTHS"
80 END
```

Programming Challenge #2 (p. 53)

```
100 CALL CLEAR
110 PRINT "WHAT ROAD SIGN IS RED AND"
120 PRINT "IS OCTAGONAL IN SHAPE?":::
130 PRINT "1) A YIELD SIGN"
140 PRINT "2) A RAILROAD SIGN"
150 PRINT "3) A STOP SIGN"
160 INPUT N
```

```
170 IF N = 3 THEN 180 ELSE 200
180 PRINT "YOU GOT IT!"
190 END
200 PRINT "TRY AGAIN"
210 GOTO 160
```

Programming Challenge #3 (p. 53)

```
10 REM KNOCK KNOCK
20 OPEN #1:"SPEECH",OUTPUT
30 CALL CLEAR
40 PRINT "KNOCK KNOCK":::::::::
50 PRINT #1:"KNOCK KNOCK"
60 FOR DELAY = 1 TO 500
70 NEXT DELAY
80 CALL CLEAR
90 PRINT "WHO'S THERE?":::::::::
100 PRINT #1:"WHO'S THERE?":::::::::
110 FOR DELAY = 1 TO 500
120 NEXT DELAY
130 CALL CLEAR
140 PRINT "BOO.":::::::::
150 PRINT #1 :"BOO"
160 FOR DELAY = 1 TO 500
170 NEXT DELAY
180 CALL CLEAR
190 PRINT "BOO WHO?":::::::::
200 PRINT #1:"BOO WHO?"
210 FOR DELAY = 1 TO 500
220 NEXT DELAY
230 CALL CLEAR
240 PRINT "WHY ARE YOU CRYING?":::::::::
250 PRINT #1:"WHY ARE YOU CRYING?"
260 PRINT #1:"HA HA HA HA HA"
270 END
```

Programming Challenge #4 (p. 54)

```
100 CALL CLEAR
110 PRINT "BASKETBALL POINT AVERAGES":::::::::::
120 FOR DELAY = 1 TO 500
130 NEXT DELAY
140 CALL CLEAR
150 INPUT "PRINT PLAYER'S LAST NAME: ":NAME$
160 INPUT "HOW MANY GAMES WERE PLAYED IN THE
    SEASON? ":N
170 CALL CLEAR
180 PRINT "ENTER PLAYER'S POINT SCORE FOR EACH
GAME ONE AT A TIME. PRESS 'ENTER' AFTER EACH
SCORE" :::::::::::
190 FOR X = 1 TO N
200 INPUT SCORE
210 A = SCORE + A
220 NEXT X
230 AVE = A/N
240 PRINT ::
250 PRINT "THE AVERAGE NUMBER OF POINTS SCORED
BY ";NAME$;" PER GAME WAS ";AVE
```

B.C.

One of the most ancient counting machines is the abacus, which was invented in Asia. The Chinese abacus, called a "suan-pan," was developed around 3000 B.C. The abacus was modified and improved in the centuries that followed, and it is still used by many people in the world today.

1600

In the 1640s, the French philosopher and mathematician, Blaise Pascal, invented and built the first adding machine that could carry sums.

In the late 1600s, the German philosopher, Gottfried Wilhelm Leibniz, invented a machine which multiplied by rapid, repeated additions.

1800

The French inventor, Joseph M. Jacquard, designed a loom for weaving patterned fabric in 1801. The loom selected colored threads for sections of cloth as directed by punched cards. Jacquard's punched cards enabled him to direct a machine in specific tasks and to change the directions simply by changing the card.

In the 1820s, Charles Babbage, an English mathematician, designed a calculating device that contained all the components of a modern computer. However, steam was the main source of power at that time, and finding steam too crude for his design, Babbage was unable to complete the device.

In 1888, in the United States, William S. Burroughs made the first adding machine which successfully recorded data.

Hermann Hollerith and John Shaw Billings of the United States invented a machine that used punched cards for counting census data. Hollerith's machine was used in the 1890 U.S. census.

1900

Vannevar Bush, an American electrical engineer, designed and built the first analog computer in the 1930s.

Between 1937 and 1944, Howard H. Aiken of Harvard University designed and built the first digital computer in conjunction with IBM. The Harvard Mark I, or the Automatic Sequence Controlled Calculator, could perform three additions every second. It was eight feet high and fifty-one feet long.

In the 1940s, John Mauchly and John Presper Eckert, electrical engineers at the University of Pennsylvania, built ENIAC (Electronic Numerical Integrator and Computer). ENIAC used 18,000 vacuum tubes, and was much faster than earlier machines.

In 1948, Bell Labs invented the transistor, which eventually replaced the bulky vacuum tubes previously used in computers.

The Universal Automatic Computer, or UNIVAC, was the first computer produced for sale. In 1951, it was installed at the U.S. Bureau of the Census.

In 1954, Texas Instruments designed the world's first commercially produced transistor radio. As a result of the project, TI became the first company to design and mass-produce low-cost germanium transistors.

In 1958, Jack S. Kilby of TI invented the integrated circuit (IC), or "chip," a single piece of silicon containing complete electronic circuits.

In 1967, TI produced the first electronic handheld calculator. This miniature electronic calculator is in the permanent collection of the Smithsonian Institute in Washington, D.C.

The single-chip microprocessor, an integrated circuit containing all the elements of a computer's central processing unit (CPU), was invented by Gary Boone of TI in 1970. Today, similar devices are the "brains" of a wide range of electronic products, such as the TI-99/4A Home Computer.

In 1971, TI introduced the single-chip microcomputer, invented by Gary Boone and Michael Cochran of TI. The device packed all the elements of a computer (CPU, data and instructions memory, input/output circuitry and clock) into a silicon chip the size of a baby's fingernail. The device later evolved into the "miracle chip" that has found wide application in calculators, watches, appliances, automobiles, office equipment, electronic toys and games, and hundreds of other products.

Solid State Software™ technology was invented at TI in 1976. The invention allowed electronic devices— calculators, learning aids, home computers—to be "preprogrammed" through the use of interchangeable, plug-in cartridges.

In 1978, TI introduced the Speak & Spell™ electronic learning aid, a device which incorporated TI's invention, the single-chip speech synthesizer. This speech synthesizer —the first integrated circuit to duplicate the human vocal tract electronically—led to the introduction of TI's Solid State Speech™ technology which is also used in the TI-99/4A.

In 1981, TI introduced TI LOGO, the first microcomputer language allowing children to create a personal learning environment. Using TI LOGO, children "teach" the TI-99/4A Home Computer to draw lines and create colorful moving shapes.

There are two types of memory in a computer system: Random Access Memory (RAM) and Read Only Memory (ROM). Information stored in RAM is not permanent. It can be transferred to storage media (disk, cassette), retrieved from storage media, changed, reordered, and erased. When power to the system is turned off, whatever information was stored in the RAM disappears. For example, a program entered from the keyboard or loaded from disk or cassette is stored in RAM. ROM is built-in, permanently stored memory which cannot be altered in any way. An example of ROM is a pre-programmed cartridge.

Computers "remember" and "calculate" through a series of switches. Each switch codes a zero (off) or a one (on). This coding provides the computer with information. Each switch is called a *bit.* Eight bits are called a *byte.* It takes one byte of information to encode a single character from the keyboard, for example, the letter A. The TI-99/4A alone has the capacity to hold approximately 16,000 bytes of information in memory. This is normally referred to as 16K RAM. TI-99/4A peripherals can expand the system to 48K RAM. With addition of the Mini-Memory cartridge, the system can grow to 52K RAM.

INPUT

The flow of information to the computer is called *INPUT.* Input is supplied mainly through the keyboard, although disk, tape, etc. may also serve to input.

The input goes to the Central Processing Unit (CPU). The *CPU* calculates, sorts, and compares information. It can also store data in the memory for later use.

CPU **MEMORY**

MEMORY exists not only internally in the CPU, but also externally, in the form of storage devices (cassettes, diskettes, and cartridges) discussed on the following page.

OUTPUT

The flow of information from the computer is called *OUTPUT.* The main types of output are the screen display (words and pictures appearing on the video monitor or TV), and hard copy (printouts) from a printer.

SOLID STATE CARTRIDGES

Equipment needed:
• Computer console
• TI Color Monitor or TV with adapter (Video Modulator)
• Cartridge (contains a printed circuit board with chips)

Cartridge software is an excellent way to take advantage of the wide assortment of pre-programmed software currently available. Most cartridges require no extra peripherals because they plug directly into the console. Cartridges, none of which are erasable with the exception of the Mini-Memory, are very easy to use. You can store personal data, temporarily or permanently, on the specialized Mini-Memory cartridge.
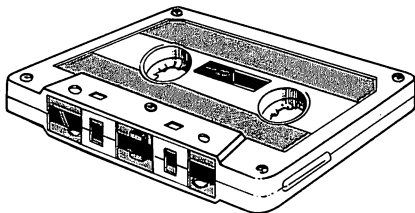
CASSETTES

Equipment needed:
• Computer console
• TI Color Monitor or TV with adapter (Video Modulator)
• Compatible cassette recorder
• Cassette tape (high quality)
• Cassette cable

Using cassettes is an inexpensive way to store personal programs and data files, and to enjoy pre-programmed software. The small amount of equipment needed (recorder and tape) is not very expensive and, of course, can be used for other purposes. One or two cassette recorders can be controlled from the console. Sometimes using cassettes involves waiting and repeating the procedure to successfully load data into the computer. It is the preferred method of data storage if you choose to avoid the higher cost of a disk system.

DISKETTES
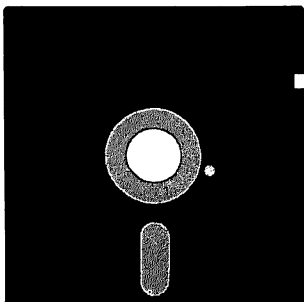
Equipment needed:
• Computer console
• TI Color Monitor or TV with adapter (Video Modulator)
• Disk Controller card with the Peripheral Expansion System (or TI Disk Controller peripheral)
• One or more TI Disk Memory Drives
• Diskettes (5¼ inch floppy disk)

Using the TI Disk Memory System is the fastest method of storing and retrieving personal data. Pre-programmed software is also available on disk. The same programs are usually available on both diskette and cassette. Though the equipment involved costs more than cassette, the speed and efficiency of using diskettes is well worth it. About 87,000 keystrokes of information can be stored on a single disk.

SOLID STATE CARTRIDGES

To Insert Solid State Cartridges
• Turn on your monitor or TV, then the console. The TI Master Title Screen should automatically display on your screen.
• Insert a solid state cartridge firmly into the slot to the right of the keyboard. The screen will go blank momentarily, then the TI Master Title Screen should reappear.
• Press any key to move to the next screen.
• Select the name of the cartridge you inserted usually by pressing the number 2 (the number 1 will always take you into TI BASIC).

To Remove Solid State Cartridges
• Return to the TI Master Title Screen by pressing FCTN and the " = " key, or by exiting according to instructions included in the cartridge's program.
• Remove the cartridge from its slot.

In Case of Difficulty
• If a cartridge does not seem to operate properly, reinsert the cartridge. If the screen locks or produces unusual displays, turn the console off and wait a few seconds before turning it back on.

CASSETTE SOFTWARE

Load and Save with Cassette
• Use only high-quality cassette tapes, sixty minutes or less in length.
• Connect a compatible cassette recorder to your computer with cassette cables. Insert the single plug at one end of the cable into the back of the console. The red-wired plug at the opposite end of the cable connects to the cassette microphone jack. The white-wired plug adjacent to it connects to the earphone or auxiliary cassette jack. It is not necessary to insert the black-wired remote plug into the cassette recorder (it can cause some cassette recorders to malfunction). If you are using dual cassette cables, insert the end marked CS1 into the cassette recorder.
• Make sure the ALPHA LOCK key is in locked (down) position.

Loading a Progam
• Make sure the cassette recorder is either plugged in or contains good batteries.
• Insert a cassette containing a program into the cassette deck.
• Turn on the monitor or TV, then the console.
• Select TI BASIC from the menu.
• When TI BASIC READY appears, type OLD CS1, then press ENTER.
• Follow the directions on the screen to "walk" through the loading procedure.
• When the DATA OK statement appears on the screen, wait for the flashing cursor to reappear.
• Type RUN and press ENTER. (It may take several seconds for the program to display its first screen on your video.)

Saving a Program
• Prepare a program to save by either writing or loading it from another source.
• Insert a blank cassette tape into the cassette recorder.
• Turn on the monitor or TV, then the console.
• Select TI BASIC.

• Type SAVE CS1, then press ENTER.
• Follow the instructions on the screen to "walk" through the SAVE procedure.
• When the SAVE procedure is complete, the question CHECK TAPE(Y OR N)? appears on the screen. Respond with "Y" to check the recording.

If You Have Difficulties

An error message reading "NO DATA FOUND" may be displayed. If this happens, check for the following problems and correct them if necessary:

• an unplugged cassette recorder or weak batteries
• improperly placed or loose connections
• cassette volume level (it should be around 8)
• cassette tone level (it should be around 6)
• an excessively long leader on your cassette tape (after you press the PLAY button on your recorder, wait a few seconds, then press ENTER)

If these common problems have been corrected but error messages continue, the cassette recording may be damaged, the cassette recorder may be incompatible with the TI-99/4A system, or improperly functioning equipment may be found.

DISKETTE SOFTWARE

To Load and Save from Diskette

• Follow the set-up procedures outlined in the Disk Memory System manual.
• Use standard 5¼-inch, soft-sectored, 40-track floppy diskettes.
• Make sure ALPHA LOCK is in the locked (down) position.

Loading a Program

• Turn on the Peripheral Expansion Box (or disk drive and controller), the monitor or TV, and finally the computer.
• Insert a diskette containing stored programs into the disk drive.
• Close the disk drive door.
• Select TI BASIC.
• When TI BASIC READY appears on the screen, type OLD DSK1.-------- (where -------- represents the name of the program) and press ENTER.
• When the flashing cursor reappears on the screen (this may take several seconds to occur), type RUN and press ENTER.

Saving a Program

• Initialize a blank diskette using the Disk Manager cartridge. Refer to the Disk Manager manual for instructions on initialization.
• Prepare a program to save by either writing or loading it from another source.
• Insert the initialized diskette into the disk drive unit.
• Close the disk drive unit door. Never open a disk drive door when the red light is on.
• Type SAVE DSK1.-------- (where -------- represents the name of the program) and press ENTER.
• Remove the diskette when the SAVE procedure is completed. For future reference, label the diskette with the name of the program(s) saved.

SOME FINAL TIPS

As a strict rule, keep all software (especially diskettes and cassettes) away from heat, magnetic fields (metal detectors, TVs, monitors, magnets), and static electricity.

Handle disks with care. Be careful not to soil the magnetic surface or bend or warp the disks.

Reading the software and hardware instruction booklets, and keeping them nearby for quick reference, can save a great deal of time and aggravation.

Software Packages Available

INFORMATION MANAGEMENT

CARTRIDGES

Home Financial Decisions
Household Budget Management (Cassette or Disk Data storage system required.)
Securities Analysis
Personal Record Keeping (Cassette or Disk Data storage system recommended.)
Tax/Investment Record Keeping (Disk system required.)
Personal Real Estate (Cassette or Disk Data storage system recommended.)
Personal Report Generator (Disk drive/controller or Cassette recorder/ cable, TI Impact Printer/RS-232 Interface required.)
TI Writer* (Disk drive/controller, 32K Memory Expansion Unit. TI Impact Printer/RS-232 Interface required.)
Microsoft™ Multiplan[1]* (Disk drive/controller, 32K Memory Expansion Unit required. TI Impact Printer/RS-232 Interface recommended.)
Terminal Emulator II (Telephone Coupler and RS-232 Interface required for telecommunications. Speech Synthesizer required for text-to-speech.)

DISKS AND CASSETTES

Mailing List (Disk only)
Personal Finance Aids (Disk and Cassette)
Checkbook Manager (Disk only)
Business Aids Library
  • Financial Management (TI Extended BASIC cartridge required. Disk only)
  • Inventory Management (Disk only)
  • Invoice Management (Personal Record Keeping or Statistics required. Disk only)
  • Cash Management (TI Extended BASIC cartridge required. Disk only)
  • Lease/Purchase Decisions (Disk and Cassette)
Personal Tax Plan[2] (Disk drive/controller, P-Code Card, and 32K Memory Expansion Unit required. TI Impact Printer/RS-232 Interface recommended. Disk only)
TI-Count Business Packages[3] (TI Extended BASIC cartridge, TI Impact Printer/RS-232 Interface required. 32K Memory Expansion and second Disk drive recommended. Disk only)

EDUCATION

CARTRIDGES

Early Learning Fun (For ages 3-6)
Beginning Grammar (For grade levels 2-5)
Number Magic (For ages 6 and up)
Video Graphs (For all ages)
Physical Fitness (For ages 13 and up)
Early Reading[4] (For beginning readers. Speech Synthesizer required.)
Reading Fun[4] (For grade level 2. Speech Synthesizer optional.)
Reading On[4] (For grade level 3.)
Reading Roundup[4] (For grade level 4.)
Reading Rally[4] (For grade level 5.)
Reading Flight[4] (For grade level 6.)
Addition/Subtraction 1[4] (For grade level 1. Speech Synthesizer recommended.)
Addition/Subtraction 2[4] (For grade levels 1-2. Speech Synthesizer recommended.)
Multiplication 1[4] (For grade levels 3-4. Speech Synthesizer recommended.)
Division 1[4] (For grades 3-5. Speech Synthesizer optional.)
Computer Math Games II & VI[5] (For grade levels 1-9.)
Developmental Learning Materials (DLM) Series:
  Alien Addition (for addition skills)
  Minus Mission (for subtraction skills)
  Alligator Mix (for discrimination skills)
  Meteor Multiplication (for multiplication skills)
  Demolition Division (for division skills)
  Dragon Mix (for discrimination skills)
  Music Maker (Cassette or Disk Data storage system recommended.)
Weight Control and Nutrition (Cassette or Disk Data storage system recommended.)
TI LOGO II (32K Memory Expansion Unit required. TI Impact Printer/ RS-232 Interface, Disk drive/controller, Cassette recorder/cable, all optional.)
Scholastic Spelling[6] (For grade levels 3-6. Speech Synthesizer required.)
Milliken Math Series (For grade levels K-8.)
  • Addition
  • Multiplication
  • Integers
  • Decimals
  • Laws of Arithmetic
  • Measurement Formulas
  • Subtraction
  • Division
  • Fractions
  • Percents
  • Equations

DISKS AND CASSETTES

Music Skills Trainer (For ages 10 and up. Disk and Cassette)
Computer Music Box (For ages 10 and up. Disk and Cassette)
Market Simulation (Disk and Cassette)
Music Maker Demonstration (Music Maker cartridge required. Disk only)
Basketball Statistician (TI Extended BASIC cartridge required. Disk only)
Bridge Bidding I (Disk and Cassette)
Bridge Bidding II (Disk and Cassette)
Bridge Bidding III (Disk and Cassette)
Speak & Spell™ Program (Disk only)
Speak & Math™ Program (Disk and Cassette)
Spell Writer (Terminal Emulator II cartridge and Speech Synthesizer required. Disk and Cassette)
TI PILOT (Disk drive/controller, 32K Memory Expansion Unit, and P-Code Card required. Editor/Filer/Utilities disk required for UCSD P-System program development. Speech Synthesizer optional for speaking programs. Disk only)
Course Designer Authoring System* (TI Extended BASIC cartridge required. Speech Synthesizer, TI Impact Printer/RS-232 Interface, Video controller, all optional. Disk only)
SMU Electrical Engineering Library (Cartridge with Disk or Cassette available)
Text-to-Speech (English) (Disk drive/controller, Speech Synthesizer, 32K Memory Expansion Unit, TI Extended BASIC cartridge required. Disk only)
Touch Typing Tutor*

Software Packages Available

PLATO® LEARNING CENTER

Now, the power of the TI Home Computer makes PLATO courses on disks available to everyone. Choose from more than 450 programs in the PLATO curriculum. Here's all that's required to use PLATO courseware:
- TI-99/4A Home Computer
- TI Disk Memory System
- TI Memory Expansion
- PLATO Interpreter Solid State Cartridge
- PLATO Program Packages (your choice)

Parents interested in Basic Skills programs for their children receive additional help when they buy the PLATO Interpreter Solid State Cartridge, because the package includes:
- The Survey Disks, which ask your child questions to determine their strengths and weaknesses in reading, grammar, and math.
- The Parent's Questionnaire, which asks you questions that help you assess your child's academic skills.

PLATO is a trademark of Control Data Corporation, U.S.A.
Copyright © 1982 Control Data Corporation. All rights reserved.
PLATO Courseware is manufactured under license by Texas Instruments Incorporated.

BASIC SKILLS (GRADES 3-8)

Mathematics
- Basic Number Ideas
- Addition
- Subtraction
- Multiplication
- Division
- Fractions
- Decimals
- Ratio, Proportion, and Percent
- Geometry and Measurement

Reading
- Making New Words
- Understanding New Words
- Understanding What You Read
- Thinking About What You Read
- Judging What You Read

Grammar
- Parts of Speech
- Building and Using Sentences
- Spelling and Usage
- Capital Letters and Punctuation
- Writing Letters

HIGH SCHOOL SKILLS (GRADES 9-12)

Mathematics
- Basic Number Ideas
- Math Sentences in One Variable
- Math Sentences in Two Variables
- Geometry
- Measurement
- Special Topics

Reading
- Reading
- General Reading
- Prose Literature
- Poetry
- Drama

Writing
- Spelling and Punctuation
- Grammar
- Diction
- Sentence Structure
- Logic and Organization

Science
- Physics
- Chemistry
- Earth Science
- Biology

Social Studies
- Geography
- Economics
- Behavioral Science
- Political Science
- History

ENTERTAINMENT

CARTRIDGES

Parsec* (Speech Synthesizer and Wired Remote Controllers optional.)
Tombstone City: 21st Century (Wired Remote Controllers optional.)
TI Invaders (Wired Remote Controllers optional.)
Car Wars (Wired Remote Controllers optional.)
Alpiner* (Speech Synthesizer, Wired Remote Controllers optional.)
Othello[7]
Chisholm Trail (Wired Remote Controllers optional.)
Football
Video Games I (Wired Remote Controllers optional.)
Hunt the Wumpus (Wired Remote Controllers optional.)
Indoor Soccer (Wired Remote Controllers optional.)
Mind Challengers
A-Maze-Ing (Wired Remote Controllers optional.)
The Attack[8] (Wired Remote Controllers optional.)
Blasto[8] (Wired Remote Controllers optional.)
Blackjack and Poker (Wired Remote Controllers optional.)
Hustle[8] (Wired Remote Controllers optional.)
ZeroZap[8]
Hangman[8]
Connect Four[8]
Yahtzee[8]
Video Chess

E.T.[9], The Extra-Terrestrial (Speech Synthesizer recommended.)
Munch Man

DISKS AND CASSETTES

Tunnels of Doom (Cartridge with Disk or Cassette available)
Adventure International Series (Cartridge with The Pirate Adventure disk or cassette available.)

- Adventureland
- Voodoo Castle
- Strange Odyssey
- Pyramid of Doom
- Savage Island I & II
- Mission Impossible
- The Count
- Mystery Fun House
- Ghost Town
- The Golden Voyage

Mystery Melody (Disk and Cassette)
Oldies But Goodies–Games I (Disk and Cassette)
Oldies But Goodies–Games II (Disk and Cassette)
Saturday Night Bingo (Speech Synthesizer required. Disk and Cassette)
Draw Poker (TI Extended BASIC cartridge required. Disk and Cassette)

Software Packages Available

COMPUTER PROGRAMMING

CARTRIDGES

Speech Editor (Speech Synthesizer required.)

Editor/Assembler (32K Memory Expansion Unit and Disk drive/controller required.)

Mini-Memory (Cassette recorder/cable recommended. May also be used with all other peripherals.)

TI Extended BASIC

DISKS AND CASSETTES

Pascal Development System (Peripheral Expansion System, 32K Memory Expansion, Disk drive/controller required. TI Impact Printer/RS-232 Interface optional. Card Plus Disks)

Programming Aids I (TI Impact Printer/RS-232 Interface optional. Disk and Cassette)

Programming Aids II (TI Impact Printer/RS-232 Interface optional. Disk only)

Programming Aids III (TI Extended BASIC cartridge required. Disk only)

UCSD Pascal[10] Compiler (32K Memory Expansion, P-Code Card required. Disk only)

UCSD P-System[10] Assembler/Linker (32K Memory Expansion, P-Code Card required. TI Impact Printer/RS-232 Interface optional. Disk only)

UCSD P-System[10] Editor/Filer/Utilities (32K Memory Expansion, P-Code Card required. TI Impact Printer/RS232 Interface optional. Disk only)

Teach Yourself BASIC[11] (Separate packages for TI-99/4A and TI-99/4. Disk and Cassette)

Teach Yourself Extended BASIC (TI Extended BASIC cartridge required. For TI-99/4. Disk and Cassette)

TI FORTH (Editor/Assembler cartridge and 32K Memory Expansion required. Disk only)

TI Advanced Assembly Debugger (Editor/Assembler cartridge and 32K Memory Expansion required. Disk only)

MATH ENGINEERING

Statistics (Disk and Cassette Data storage system recommended. Cartridge only)

Math Routines Library (Disk and Cassette)

Electrical Engineering Library (Disk and Cassette)

Graphing Package (Disk and Cassette)

Structural Engineering Library (Disk and Cassette)

AC Circuit Analysis Library (TI Impact Printer/RS-232 Interface optional. Disk only)

---

* For TI-99/4A only

1 Developed for Texas Instruments by Microsoft™, Inc. Multiplan is a Trademark of Microsoft, Inc.

2 Developed for Texas Instruments by Aardvark Software, Inc.

3 Developed for Texas Instruments by Pike Creek Computer Company, Inc.

4 Developed by Texas Instruments in conjunction with Scott, Foresman and Company

5 Developed for Texas Instruments by Addison Wesley Publishing Company

6 Developed in conjunction with Scholastic Publishing Company, Inc.

7 Othello is a Trademark of Gabriel Industries, a division of CBS, Inc.

8 A Trademark of Milton Bradley

9 A Trademark of and licensed by Universal City Studios, Inc.

10 UCSD P-Systems is a Trademark of the University of California

11 Developed by Texas Instruments in conjunction with Wolfdata Corporation

TI-99/4 and /4A Users' Groups are groups consisting primarily of amateur computer programmers who gather to exchange information, programs and programming ideas.

Below is a list of both national and international Users' Groups locations. If you or a friend are interested in becoming a part of a Users' Group, contact the group in your area by mail, using the addresses supplied.

INTERNATIONAL USERS' GROUPS

International 99/4 Users' Group, Inc.
P.O. Box 67
Bethany, OK 73008

International Home Computer Users' Association
P.O. Box 371
Rancho Sante Fe, CA 92067

99/4 Users of America
5028 Merit Drive
Flint, MI 48506

AUSTRALIA

National Coordinator:
Shane Anderson
P.O. Box 101
Kings Cross, Sydney N.S.W. 2011

Sydney Interim Coordinator:
Brian Lewis
P.O. Box 149
Pennant Hills
N.S.W. 2110

Melbourne Interim Coordinator:
Doug Thomas
59 Lanstrom Quad
Kilsyth, Vict
Australia 3137

Brisbane Interim Coordinator:
Alwyn Smith
42 Palmtree Avenue
Scarborough, Old
Australia 4020

Perth Interim Coordinator:
Kim Schlunke
P.O. Box 246
Mt. Lawley
Western Australia 6014

Tasmania Interim Coordinator:
Andrew Zagni
161 Carrelast Street
Howrah, Tasmania
Australia 7018

Adelaide Interim Coordinator:
Gerald Tan
Pamela Street
Happy Valley
S.A. 5159

CANADA

Paul Langlois
706-10883 Saskatchewan Drive
Edmonton, Alberta
Canada, T6E 4S6

Carleton Home Computer Users' Group
John Street R.R.#2
Stittsville, Ontario
Canada K0A 3G0

Toronto Home Computer Users' Group
3175 Kirwin Avenue
Townhouse #159
Mississauga, Ontario
Canada L5A 3M4

Victoria 99'er Group
402-1471 Fort Street
Victoria B.C.
Canada V8S 1Z4

COLUMBIA

Asociacion Colombiana de Usarios 99/4
Av. Nutivara #C 3-6
Medellin Colombia S.A.

ENGLAND

T.I. HOME
Paul Michael Dicks
157 Bishopsford Road
Morden
Surry SM4 6BH

GERMANY

Frankfurt
American Express International
Dept. 204
Attn: Mr. C. Quigtar
APO N.Y. 09757

U.S. USERS' GROUPS

ALABAMA
T.I. B.U.G.
709 Nytol Circle
Birmingham, AL 35210

ARIZONA
Arizona 99 Users' Group
4328 E. LaPuenta Avenue
Phoenix, AZ 85044

CALIFORNIA
Orange County 99/4 Users' Group
1673 Chateau
Anaheim, CA 92802

L.A./South Bay 99er Users' Group
5128 Merrill Street
Torrance, CA 90503

SF/South Bay 99er Users' Group
16380 E. LaChiquita
Los Gatos, CA 95030

COLORADO
Colorado 99/4 Users' Group
15177C East Louisiana Drive
Aurora, CO 80012

FLORIDA
Tampa Bay 99er Users' Group
13097 Lois Avenue
Seminole, FL 33542

ILLINOIS
Chicago 99/4 Users' Group
353 Park Drive
Palatine, IL 60067

KANSAS
Mid-America 99/4 Users' Group
P.O. Box 2505
Shawnee Mission, KS 66201

MASSACHUSETTS
M.U.N.C.H
1241 Main Street
Worchester, MA 01603

Pioneer Valley TI-99/4 Users' Group
3 Market Street
Northampton, MA 01060

MINNESOTA
Greater Minneapolis-St. Paul Home Computer Users' Group
P.O. Box 12351
St. Paul, MN 55112

MISSOURI
Kansas City 99/4A Computer Users
4511 N. Troost
Kansas City, MO 64116

99/4 Users' Group of St. Louis
4127 Quincy
St. Louis, MO 63116

NEW YORK
Jerald Greenberg
34 Maple Ave. Box 8
Armonk, NY 10504

Upstate New York 99/4 Users' Group
7 Steve Lane
Albany, NY 12205

OHIO
Cin-Day Users' Group
11987 Cedarcreek Drive
Cincinnatti, OH 45240

OREGON
Pacific Northwest TI-99/4 Users' Group
P.O. Box 5537
Eugene, OR 97405

Portland Users of Ninety-Nines
421 Northwest 69th Street
Vancouver, WA 98665

PENNSYLVANIA
Daniel Cooper
P.O. Box 285
Hazelton, PA 18201

Pittsburgh Users' Group
P.O. Box 18124
Pittsburgh, PA 15236

RHODE ISLAND
Tri-State Users' Group
P.O. Box 457
Lincoln, RI 02864-0457

SOUTH CAROLINA
Carolina Computer Club
225 Wynchwood Drive
Irmo, SC 29063

TENNESSEE
Athens 99/4 Computer Users' Group
c/o Bob Lamb
McMinn County High School.
2215 Congress Parkway
Athens, TN 37303

TEXAS
Dallas TI Home Computer Group
P.O. Box 672
Wylie, TX 75098

Andy Belivacqua
Route 2, Box 75-U
Mansfield, TX 76063

JSC Users' Group (JUG)
15727 El Camino Real
Houston, TX 77062

Houston Users' Group
10107 Westview #112
Houston, TX 77043

Lubbock Computer Club 99/4 Users' Group
5730 67th Street
Lubbock, TX 79424

West Texas 99/4 Users' Group
P.O. Box 6448 M/S 3030
Midland, TX 79701

WASHINGTON D.C.
Washington D.C. 99/4 Users' Group
P.O. Box 267
Leesburgh, VA 22075

WASHINGTON (STATE)
Pudget Sound 99'ers
P.O. Box 6073
Lynnwood, WA 98036

WISCONSIN
Program Innovators
2007 North 71st Street
Waauwatosa, WI 53213

LOGO USERS' GROUP

Young Peoples' LOGO Association
1208 Hillsdale Drive
Richardson, TX 75081

Many different people—men and women with small businesses, teachers, hobbyists, and professionals in many fields—have begun to use home computers. Popular computing magazines now include articles to match their particular interests. Materials about computing are also becoming part of the coverage of many magazines and journals. Teachers' journals now include suggestions for computer use in the classroom, and general magazines also have articles on computing.

Because of the rapid increase in the number of people who see the value and fun in personal computing, no list of suggested sources can cover all information available.

Use this list as an introduction to information on computing. A visit to your community's library will show you the wealth of information available on this exciting new field.

Helpful Sources on BASIC

Davis, William S. *BASIC: Getting Started.* Reading, Mass.: Addison-Wesley Publishing Company, 1981.

Dwyer, Thomas A., and Critchfield, Margot. *BASIC and the Personal Computer.* Reading, Mass., Addison-Wesley Publishing Company, 1978.

Inman, Don; Zamora, Ramon; Albrecht, Bob; Quiram, Jacquelyn; O'Dell, Bob. *Beginner's BASIC.* Dallas: Texas Instruments, 1981.

Peckham, Herbert D. *Programming BASIC with the TI Home Computer.* New York: McGraw-Hill Book Company, 1979.

Shelley, John. *Addison-Wesley Pocket Guide to Programming.* Reading, Mass.: Addison-Wesley Publishing Company, 1982.

Texas Instruments. *User's Reference Guide* (for the TI-99/4A Home Computer). Dallas: Texas Instruments, 1981.

General Interest Magazines

BYTE: The Small Systems Journal
POB 590
Martinsville, NJ 08836

Creative Computing
POB 5214
Boulder, CO 80321

99'er Magazine
POB 5537
Eugene, OR 97405

Computers & Electronics
(formerly *Popular Electronics)*
P.O. Box 2774
Boulder, CO

Personal Computing
4 Disk Drive
Box 1408
Riverton, NJ 08077

Popular Computing
POB 307
Martinsville, NJ 08836

Recreational Computing
People's Computer Company
1263 El Camino, POB E
Menlo Park, CA 94025

Journals of Special Interest to Teachers

Classroom Computer News
Subscription Dept.
51 Spring Street
Watertown, MA 02172

The Computing Teacher Journal
The Computer Science Dept.
University of Oregon
Eugene, OR 97403

Computers and Education
Pergamon Press, Inc.
Elmsford, NY 10523

Technological Horizons in Education
(T.H.E. Journal)
Information Synergy, Inc.
POB 992
Acton, MA 01720

TI BASIC Reference Chart

The following information provides a guide to TI BASIC commands (C), statements (S), and functions (F), with examples provided where appropriate.

Some items appearing in this list may not be covered in this manual. For further information, refer to the *User's Reference Guide.*

TERM	EXAMPLE	EXPLANATION
BREAK		Causes program to halt when encountered or optionally when lines listed are encountered. (C,S)
BYE	BYE	Leaves TI BASIC and returns to Master Title Screen. (C)
CALL CHAR	CALL CHAR(36,FFF)	Allows definition of a character to a key. In the example, the computer is instructed to redefine the dollar sign ($) character (ASCII code 36) to a substitute pattern. (C,S)
CALL CLEAR	CALL CLEAR	Clears the screen but doesn't alter the program in computer memory. (C)
CALL COLOR	CALL COLOR(1,16,13)	Allows the change of character color and the corresponding character background block. In the example, the computer is instructed to use white (color code 16) as the foreground color and dark green (color code 13) as the background color for all the characters in character set 1 (refer to Character Set Codes Chart p. 34). (C,S)
CALL HCHAR	CALL HCHAR(1,4,42,10)	Tells the computer to place a horizontal line of characters on the screen. In this case, ten asterisks (character code 42) print on the screen starting at row 1, column 4.(C,S)
CALL KEY	10 CALL KEY(0,KEY, STATUS)	Assigns ASCII code of key pressed on specified key-unit (0-5) to return-variable. (C,S)
CALL SCREEN	CALL SCREEN(9)	Tells the computer to change the screen color, in this case, to medium red (9). (Refer to Color Code Chart p. 34.) (C,S)
CALL SOUND	CALL SOUND (1000,440,1)	Instructs the computer to produce music and noise. In the example, the computer is instructed to play a tone to last one second (1000 milliseconds) at a frequency of 440 cycles per second (middle C on the piano) at a loud volume (refer to Musical Tones Frequencies Chart p. 34). (C,S)
CALL VCHAR	CALL VCHAR (2,5,42,10)	Tells the computer to place a vertical line of characters on the screen. In this case, ten asterisks (character code 42) print on the screen (starting at row 2, column 5). (C,S)
CHR$		Returns the string character corresponding to the ASCII code. (F)
DATA	DATA 34,23,0	Allows data storage within a program. In the example, the numbers following DATA are stored as data to be accessed by a READ statement. (C,S)
DIM	10 DIM X(15)	Dimensions the listed arrays as specified by integers. (C,S)
EDIT	EDIT 20	Displays a line for editing. In the example, line 20 will be displayed. (C)
END	90 END	Stops a program. In a program, END is always the last program statement and must be preceded by a line number, as in the example. (C,S)
FOR-NEXT	10 FOR A = 1 TO 10 20 NEXT A	Creates a program loop determined by the equation following FOR. In the example, the value of A is increased by one with each successive loop until the value of A exceeds 10. (S)
FOR . . . TO	10 FOR X = 1 TO 100 20 NEXT X	Repeats execution of statements between FOR and NEXT until the control variable exceeds the limit. With each loop between FOR and NEXT, incrementation is by one unless otherwise specified by STEP. (S)
GOSUB	100 GOSUB 500 110 PRINT "OK" 120 END 500 CALL CLEAR 510 RETURN	Transfers control to a subroutine at a specified line number (in this case, line 500) until RETURN is encountered (in this case line 510, which returns the program to line 110). (S)

GOTO	200 GOTO 10	Directs the program to transfer to a line number. In this case, the program will loop back to line 10. (S)
IF-THEN	10 IF A = 5 THEN 40	Tests the value of the variable following IF. In the example, if A is equal to 5, the program transfers to line number 40. If A is not equal to 5, the instruction is ignored and the program goes on to the next program line. (S)
INPUT	10 INPUT A	Stops a program and waits for information to be entered. The value entered is assigned to the variable A. (S)
INT	INT(34.6)	Tells the computer to find the greatest integer (whole number) which is less than or equal to the number in parentheses. (F)
LEN	10 LET A$ = "HELLO THERE" 20 PRINT LEN(A$)	Counts the number of characters in a specific string expression (in this case, 11) and displays the number on the screen. (F)
LET	LET A = 5	Assigns a value to a variable. In the example, the value 5 is assigned to the variable A. In TI BASIC, the word LET is optional, so this statement could be written A = 5. (C,S)
LIST	LIST	Tells the computer to display the program in its memory. (C)
NEW	NEW	Erases the current program in the computer's memory and prepares to store a new program. (C)
NUM	NUM NUM 5,5	Automatically generates sequenced line numbers beginning at line 100 in increments of 10. Optionally, an initial line and/or increment may be specified. The second example begins numbering with line 5 and increments by five. (C)
OPEN	OPEN #1:"DSK1.MYFILE"	Opens a file, a memory segment on an accessory device, allowing the computer to output data to the device, in this case, Disk Drive 1 (DSK1). (C,S)
PRINT	PRINT "HELLO"	Instructs the computer to print on the screen whatever is enclosed in quotation marks. In the example, the word HELLO is printed on the screen. (C,S)
RANDOMIZE	10 RANDOMIZE	Resets the random number generator to an unpredictable sequence. (C,S)
READ	10 READ X,Y,Z	Tells the computer to find a DATA statement in the program and assign values respectively to the variables which follow the READ statement. (S)
REM	10 REM PAYROLL PROGRAM	Describes or REMarks about program operations at certain points in a program. (C,S)
RESEQUENCE **RES**	RESEQUENCE RES	Renumbers program statements starting at 100 in increments of 10. Optionally, an initial line number and/or increment may be specified. (C)
RETURN	100 GOSUB 500 110 PRINT "OK" 120 END 500 CALL CLEAR 510 RETURN	Transfers program control from subroutine to statement following corresponding GOSUB (or ON-GOSUB) statement, in this case, line 110. (S)
RND	PRINT RND	Tells the computer to print a pseudo-random number greater than or equal to zero and less than one. (F)
RUN	RUN	Tells the computer to run, or execute, the program in its memory. (C)
SAVE	SAVE CS1 SAVE DSK1.MYFILE	Tells the computer to store the program in its memory onto a cassette tape (SAVE CS1) or diskette (SAVE DSK1.MYFILE). (C)
SEG$	10 LET A$ = "HELLO THERE" 20 PRINT SEG$(A$,1,5)	Lifts a substring, beginning at a specified position with specified length, from a string expression and prints that segment on the screen, in this case, HELLO. (F)
STOP	500 STOP	Terminates program execution. (C,S)
TAB	TAB(15)	Tells the computer to begin printing at a designated position, in this case, position 15. (F)
TRACE	TRACE	Lists line numbers of statements before they are executed. (C,S)
UNTRACE	UNTRACE	Negates the TRACE feature. (C,S)

Allophone—A minimal unit of human speech.

Array—A collection of numeric or string variables arranged in a list or matrix for processing by the computer. Each element in an array is referenced by a subscript describing its position in the list.

ASCII—The American Standard Code for Information Interchange; the code structure used in most personal computers to represent letters, numbers, and special characters.

BASIC—(Beginners All-purpose Symbolic Instruction Code)—A very successful and popular computer language developed at Dartmouth College in 1963-64.

Baud—The signaling speed of information in a computer (typically relating to input and output). It is the number of bits of information per second that a computer can process. Baud rates are a factor in selecting a printer for a computer.

Binary—The two-digit (bit) number system based on 0 and 1. Computers recognize the binary bits 0 and 1 by using gates. Gates are electronic circuits which, when either off or on, represent 0 or 1.

Bit—A binary digit (0 or 1).

Branch—A departure from the sequential performance of program statements. An unconditional branch causes the computer to jump to a specified program line each time the branching statement is encountered. A conditional branch transfers program control based on the result of some arithmetic or logical operation.

Breakpoint—A point in a program specified by the BREAK command where program execution can be suspended. During a breakpoint, operations can be performed in the Immediate Mode (Command Mode) to help locate program errors. Program execution can be resumed with a CONTINUE command, unless editing occurred while the program was stopped.

Buffer—An area of computer memory for temporary storage of an input or output record.

Bug—An error in the hardware or software of a computer.

Byte—A string of eight binary bits. The computer uses approximately one byte of information to encode the letter "A".

Cassette—A form of computer storage for programs and other data.

Central Processing Unit—(CPU)—The nerve center of a computer; the network of electronic circuits that interprets programs and tells a computer how to execute them.

Character—A letter, number, punctuation symbol, or special graphics symbol, usually equivalent to one byte.

Chip—Tiny silicon slices used to produce electronic memories and other circuits. A single chip may have as many as 30,000 electronic parts.

Circuit Board—A rigid fiberglass or phenolic card upon which various electronic parts are mounted. Printer or etched copper tracks connect the various parts to one another.

Command—A word or pair of words that tells the computer to do something in the Immediate Mode. Examples: NEW, LIST, RUN, CALL CLEAR.

Command Cartridge—Pre-programmed ROM cartridges which are easily inserted in the TI computer to extend its capabilities. See also: Solid State Cartridge

Computer—A network of electronic switches and memories that processes data.

Concatenation—Linking two or more strings to make a longer string. The "&" is the concatenation operator.

Console—Main part of the computer containing the keyboard and the CPU.

Constant—A specific numeric real number (such as 1.2 or −9054) or a string constant (any combination of up to 112 characters enclosed in quotes, such as "HELLO THERE" or "275 FIRST STREET").

CPU—See Central Processing Unit.

Cursor—A small flashing square showing where a typed character will appear.

Data—Information, often numerical in form.

Default—A standard characteristic or value which the computer assumes, if certain specifications are omitted within a statement or program.

Disk—See Floppy Disk.

Diskette—See Floppy Disk.

Display—The video screen on the monitor.

Exponent—A number indicating the power to which a number or expression is to be raised, usually written at the right and above the number. For example: $2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$. In TI BASIC, the exponent is entered following the ^ symbol or following the letter "E" in scientific notation. For example: $2^8 = 2\char`^8$; $1.3 \times 10^{25} = 1.3E25$.

File—A collection of related data records stored on a device; also used interchangeably with device for input/output equipment which cannot use multiple files, such as a line printer.

Floppy Disk—A flexible plastic disk coated with the same magnetic material used to make recording tape.

Flow Chart—A diagram of geometric shapes connected by arrows that show the progression of a computer program. Flow charts are handy for developing complicated computer programs and illustrating how programs work.

Gate—A very simple electronic circuit that is always either on or off. Clusters of gates can manipulate binary numbers (0 = off, 1 = on). They can also count, do arithmetic, make decisions, and store binary numbers. Gates are the basic building block of computers.

Graphics—Visual construction on the screen, such as graphs, patterns, and drawings, both stationary and animated.

Graphics Line—A 32-character line used by TI BASIC graphics subprograms.

Hard Copy—The permanent print-out of a program or its results produced by a printer connected to a computer.

Hardware—The circuit boards and electronic parts which compose a computer.

Hexadecimal—A base 16 number system using 16 symbols, 0-9 and A-F. It is used as a convenient "shorthand" way to express binary code. For example, 1010 in binary = A in hexadecimal, 11111111 = FF. Hexadecimal is used for constructing graphics characters in the CALL CHAR subprogram.

Immediate Mode—A computer mode in which commands are entered directly into the computer without a line number. Such commands are executed immediately.

Input—The means by which data is entered into a computer—often a keyboard.

Input Line—The amount of data which can be entered at one time. In TI BASIC, this is 112 characters.

Instruction—A statement or command that tells a computer what to do.

Integer—A whole number, either positive, negative, or zero.

Interpreter—The program stored inside a computer that converts or translates BASIC statements into the computer's machine language.

Iteration—One repetition of the technique of repeating a group of program statements. See "Loop."

K—Short for kilo meaning thousand, and used to designate memory capacity—thus a 4K memory has approximately 4,000 storage elements.

Keyboard—A typewriter-like panel of keys used to enter programs and data into a computer.

Line Number—A number identifying a statement in a program. Line numbers determine the order in which a computer executes commands in a program.

Loop—A group of consecutive program lines which are performed repeatedly, usually a specified number of times.

Mantissa—The basic numeric portion of a number expressed in scientific notation. In 3.264E + 4, the mantissa is 3.264.

Memory—Any of the many devices (ROMs, RAMs, floppy disks, magnetic tapes, etc.) that store computer programs and data.

Microcomputer—A computer made by combining a microprocessor with some memory. Microcomputers are small in size, not performance.

Microprocessor—The central processing unit of a computer assembled on a single silicon chip.

Monitor—Television-like device used to display programs as they run or are being written.

Operator—A symbol used in calculations (numeric operators) or in relationship comparisons (related operations). The numeric operators are $+, -, *, /, \char`^$. The relational operators are $>, <, =, >=, <=, <>$.

Output—Information sent from the computer, e.g. graphics on the monitor screen, a report being printed. Also, the means by which data leaves a computer—often a television monitor or printer.

Paper Tape—A narrow ribbon of paper containing computer data in the form of punched holes. A hole indicates the bit 1; no hole indicates the bit 0. Paper tape is sometimes used to enter programs into a computer.

Peripheral—An accessory which can be added to a computer to increase its capability and usefulness (floppy disk, paper tape unit, etc.).

Personal Computer—An economical microcomputer designed for use by small businesses, schools, and computer hobbyists.

Printer—A computer output mechanism that delivers hard copy data.

Print Line—A 28-character line which can be referenced by PRINT and DISPLAY statements.

Program—The list of instructions that tells a computer what to do to perform a task.

Program Line—A line containing a single statement, the maximum length of which is 112 characters.

Programmer—A person who writes computer programs.

Programming Language—Numeric or alphabetic commands which a computer can assimilate, understand, and execute.

Prompt—A symbol (>) which marks the beginning of each command or program line entered; a symbol or phrase that requests input from the user.

RAM (Random Access Memory)—A temporary memory, i.e. one in which data is stored so long as electrical power is applied. Data in RAM can be accessed or changed and is lost if electrical power is cut off.

ROM (Read Only Memory)—Certain instructions for the computer are permanently stored in ROM and can be accessed but cannot be changed. Data in ROM is not lost if electrical power is cut off.

Run Mode—A computer mode in which the computer is executing a program. Run Mode is terminated when program execution ends normally or abnormally. You can cause the computer to leave Run Mode by pressing CLEAR during program execution. (See Breakpoint)

Scientific Notation—A method of expressing very large or very small numbers by using a base number (mantissa) times ten raised to some power (exponent).

Scroll—Movement of text on the screen so that additional information can be displayed.

Software—Computer programs written on paper or stored on magnetic tape or a floppy disk.

Solid State Cartridge—Pre-programmed ROM cartridges which are easily inserted in the TI computer to extend its capabilities. See also: Command Cartridge

Speech Synthesizer—A peripheral that enables the computer to talk.

Sprite—A character to which you can give shape, color, speed, screen position, and direction in TI LOGO and in TI Extended BASIC.

Statement—A single line of a computer program containing a single instruction such as PRINT, LET, RUN, etc.

String—A series of letters, numbers, and symbols treated as a unit.

Subprogram—A predefined general-purpose procedure accessible to the user through the CALL statement in TI BASIC. Subprograms extend the capability of BASIC and cannot be easily programmed in BASIC.

Subroutine—A program segment which can be used more than once during the execution of a program, such as a complex set of calculations or a print routine. In TI BASIC, a subroutine is entered by a GOSUB statement and ends with a RETURN statement.

Subscript—A numeric expression which specifies a particular item in an array. In TI BASIC the subscript is written in parentheses immediately following the array name.

Terminal—An input device such as a keyboard, or an output device such as a printer or a TV monitor.

Trace—Listing the order in which the computer performs program statements. Tracing the line numbers can help you find errors in a program flow.

Turtle—In TI LOGO, the small triangle with which designs are drawn on the screen.

Users' Group—An informal or formal association of persons who own or operate similar or identical computing equipment. Users' groups are usually formed to exchange programs and other helpful information.

Variable—A name given to a value which may vary during program execution. A variable is a memory location where values can be replaced by new values during program execution.

Wired Remote Controllers—Small, handheld controls, sometimes called joysticks, used to move graphics in desired directions on the screen.
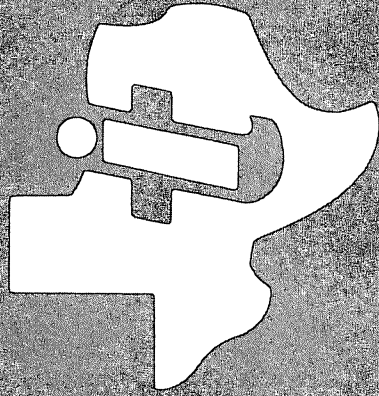
This certifies that

has successfully completed the
course of instruction in

PROGRAMMING DISCOVERY IN TI BASIC

sponsored by the

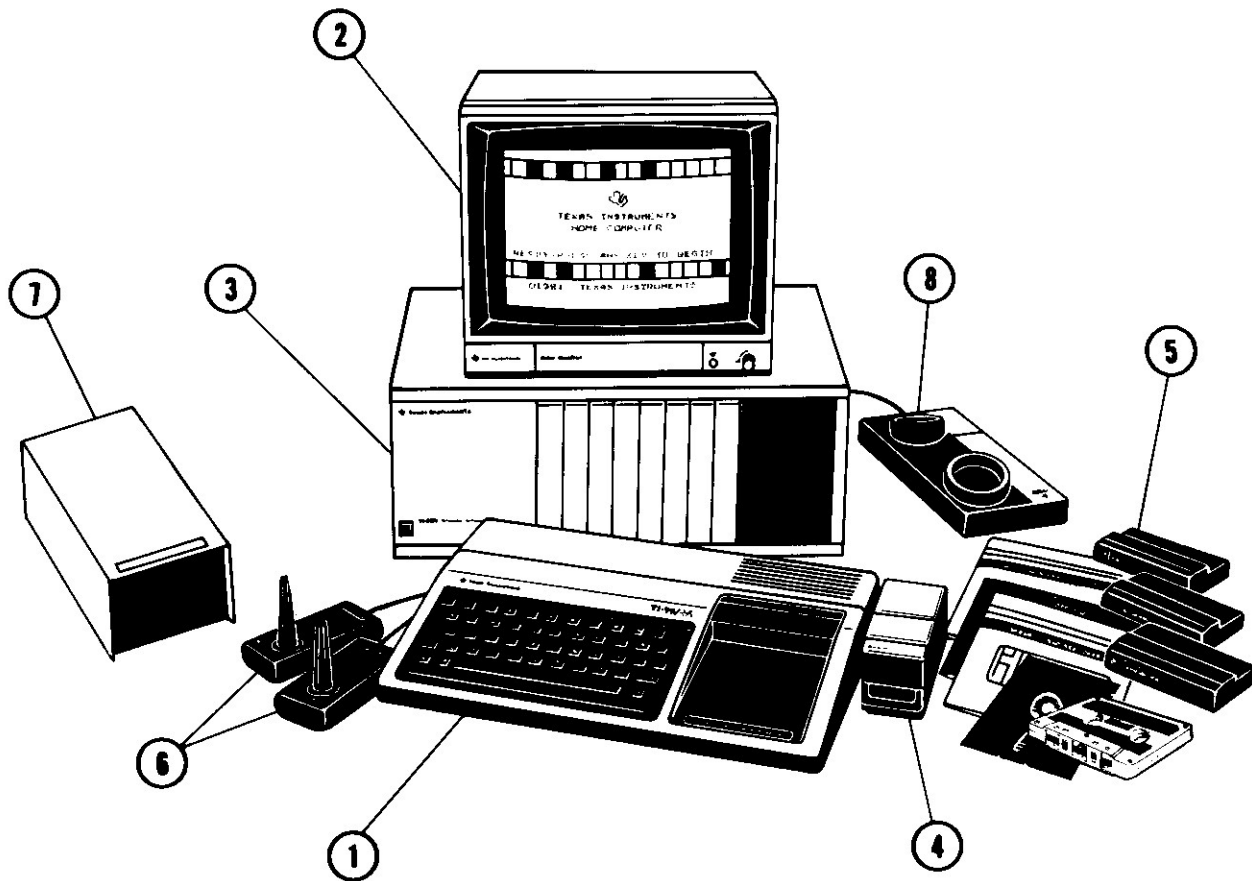TEXAS INSTRUMENTS COMPUTER ADVANTAGE CLUB

_____
Program Instructor

_____
Date

TEXAS
INSTRUMENTS

Parts of a Computer

1. **TI-99/4A HOME COMPUTER CONSOLE**: A typewriter-like console that allows you to enter, store, and manipulate data.

2. **VIDEO MONITOR**: A ten-inch color screen with a display format for 24 lines of 32 characters and audio capabilities.

3. **PERIPHERAL EXPANSION SYSTEM**: A compact system designed to centralize the Disk Memory System, the RS-232 Interface, the Memory Expansion unit, and other accessories.

4. **SPEECH SYNTHESIZER**: A device which reproduces human speech electronically and accurately, allowing the computer to communicate verbally.

5. **HOME COMPUTER SOFTWARE**: A large library of pre-programmed cassettes, diskettes, and Solid State Cartridges designed to help you learn, keep household records, or play stimulating games.

6. **WIRED REMOTE CONTROLLERS**: Eight-position remote controls with top-mounted action button which allow you to move objects on the screen.

7. **DISK MEMORY SYSTEM**: Stores data or programs for later use.

8. **TI TELEPHONE COUPLER (MODEM)**: Allows the Home Computer to send or receive information through a telephone.

TEXAS INSTRUMENTS

Printed in U.S.A.

1053665-0001