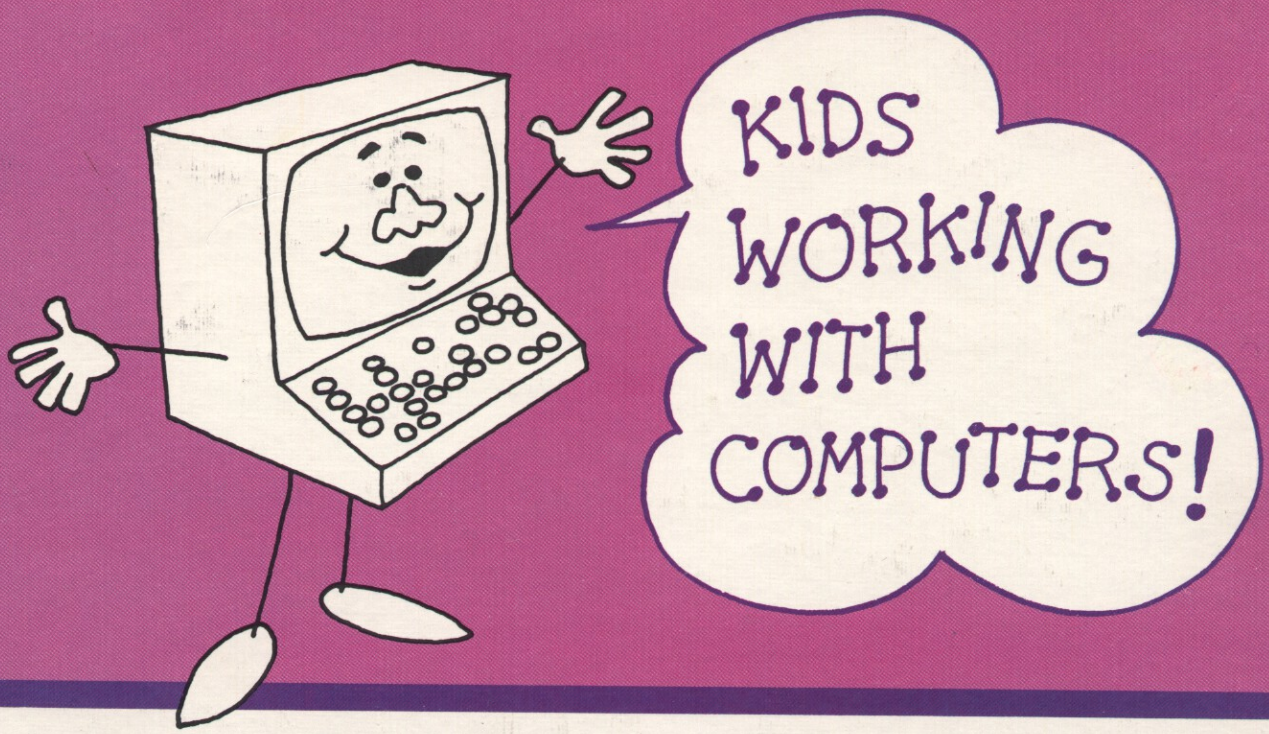
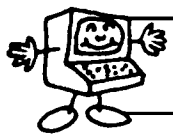


THE ATARI® BASIC MANUAL

Thomas Milton Kemnitz
Lynne Mass



Kids Working with Computers

THE ATARI® BASIC MANUAL

Thomas Milton Kemnitz **Lynne Mass**

CHILDRENS PRESS™
CHICAGO

Library of Congress Cataloging in Publication Data

Kemnitz, Thomas Milton.
The Atari BASIC manual.

(Kids working with computers)
Includes index.
Summary: Introduces some simple programs which can be done using BASIC on an Atari computer.
1. Atari computer—Programming—Juvenile literature. 2. BASIC (Computer program language)—Juvenile literature. [1. Atari computer—Programming. 2. Programming (Computers) 3. BASIC (Computer program language)] I. Mass, Lynne. II. Title. III. Series.

QA76.8.A82K46 1985 001.64'2 85-395
ISBN 0-516-08424-0

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

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

CONTENTS

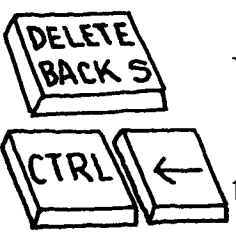
1

Meet the Computer

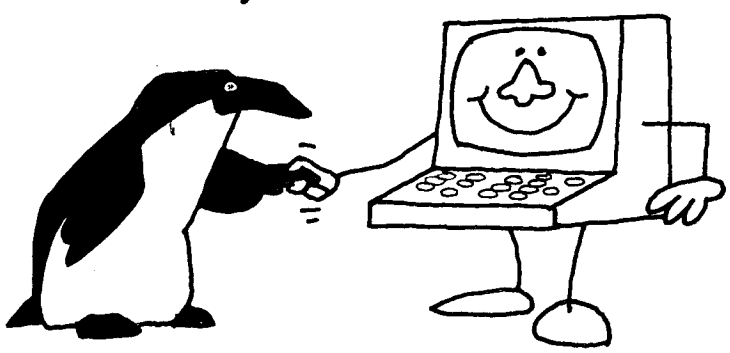
Let's meet the computer. To start:

1. Turn on the monitor.
2. Make sure the BASIC cartridge is loaded in the cartridge slot.
3. Turn on the computer, using the switch on the right side.
4. You will see: □READY

The solid white box is called the **cursor.** You are now ready for your first **program.**



If you type a mistake, press DELETE/BACK S and type over. Or you can hold down CTRL and press ← until you reach your mistake. Then get rid of the error by typing the correct entry.



To get quotation marks, hold down the SHIFT key and press the key with the quotation marks (") at the top.

Type exactly what you see. Press **RETURN** at the end of each line.


```
10 PRINT ''HELLO, MY NAME IS (type in name).''
20 PRINT ''I AM LEARNING TO WORK THE ATARI.''
30 PRINT ''MY TEACHER IS (type in teacher's
   name).''
40 PRINT ''(type something about yourself.)''
```

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

Type **RUN** and push RETURN. What happens?

Hold down SHIFT and push CLEAR. What happens? Has your program been lost? Type LIST and push RETURN to find out.

Type **NEW.** Now try LIST. What happens?

Review

What do these words do? Why are they called **commands?**

LIST

RUN

NEW

2

Do I Have to Type PRINT?

Type the following program. Remember to press RETURN at the end of each line.

```
10 PRINT ''I AM TYPING...''
20 PRINT ''ON AN ATARI COMPUTER.''
30 PRINT ''THIS IS FUN!''
```

Type LIST and hit RETURN. Type RUN and hit RETURN. What is the difference between the commands LIST and RUN?

Now type:

```
20 ? ''ON THE KEYBOARD OF AN ATARI COMPUTER.''
```

Type LIST [RETURN]. What did the computer do?

This time type RUN [RETURN]. What does a question mark mean to the computer?

Try your own ideas and decide whether you want to use ? or PRINT.

3

Do I Need Quotation Marks?

Type the following program:

```
10 ? ''HI, MOM; HI, DAD!''
```

Did you remember to press RETURN? Now type LIST and press RETURN. Then type RUN and press the RETURN key.

This time try the program without quotation marks:

```
10 ?HI, MOM; HI, DAD!
```

What does the computer do? Do you need quotation marks? Why?

Just for Fun

Type NEW, then try this:

```
FOR B = 1 TO 798 : ? ''A'' ; : NEXT B : ?''PHEW!''
```

4

Computer Loops

Type the following program and RUN it:

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

When you have seen enough, push BREAK. This is called a **LOOP** and is controlled by the **GOTO** command.

Now try this program:

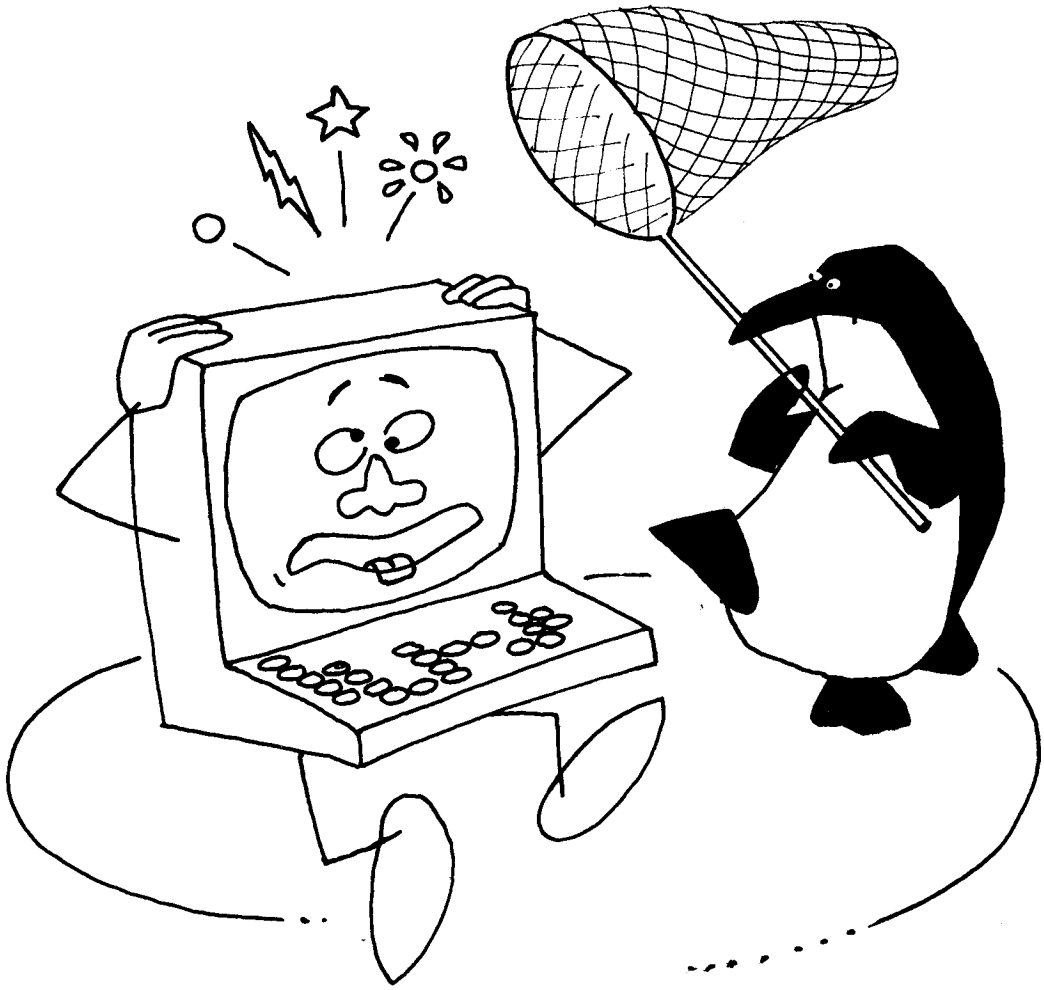
```
10 FOR X = 1 TO 5
20 PRINT ''HELP, MY COMPUTER WENT CRAZY.''
30 NEXT X
40 PRINT ''NO, IT IS UNDER CONTROL.''
```

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

Retype line 10 this way:

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

This will erase what was there before; it is just like recording over a tape that had something on it.



What do you think will happen when you LIST the program? When you RUN it?

 Try putting your name in line 20. Have fun!

5

The Egg Timer

Type this program:

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

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

Change line 20 as follows and RUN the program:

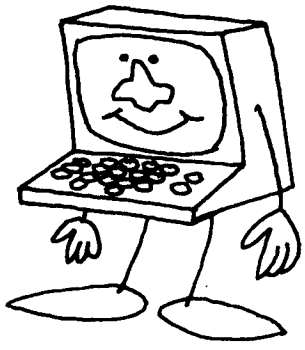
```
20 FOR X = 1 TO 2000
```

Then change it to this and RUN the program:

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

How are the three times different?

How could you change the program to make a thirty-second delay? Work it out and try it.



6

Boss Around Your Computer

Type the following program:

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

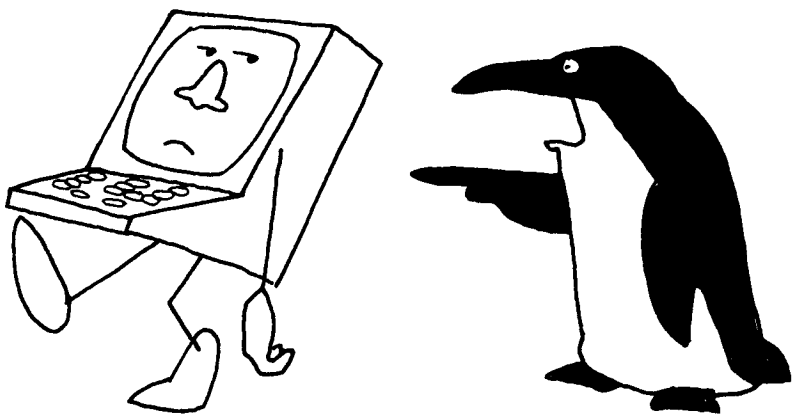
What happens? How many times do you think you told the computer to write lines 20, 30, and 40?

Keep the same program and add:

```
45 PRINT: PRINT
```

Does the program look different?

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



7

Line Order

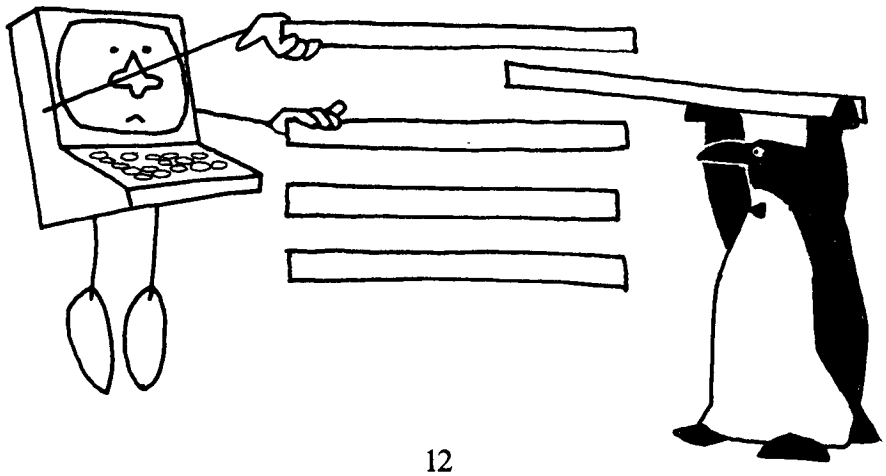
Type the following program:

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

Now RUN the program. What did the computer do to help you?

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

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



8

Oops! We're Off Course

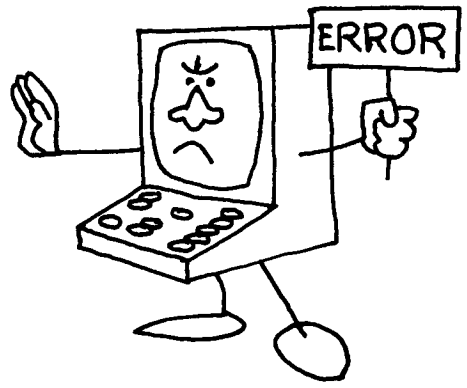
Suppose you typed an error or two in your program. Type this (ignore the error messages that appear after you hit RETURN):

```
10 PRINT ''JULIUS CAESAR WAS A LOMAN.''
20 PRIMT ''GEORGE WASHINGTON WAS AN AMERICAN.''
30 ''KING KONG WAS AN APE.''
```

Your screen will tell you:

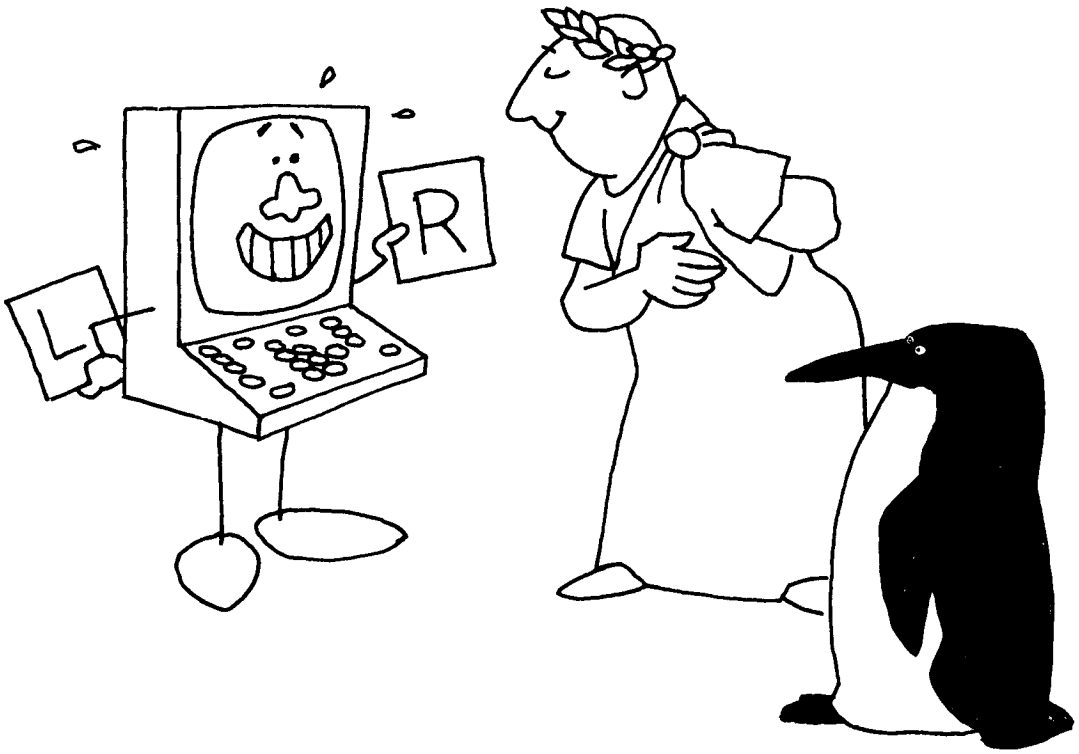
```
JULIUS CAESAR WAS A LOMAN
ERROR—17 AT LINE 20
```

To fix the errors in lines 20 and 30 (or in any other line), simply retype the lines, being very careful to spell PRINT correctly (or, even easier, use a ?).



Now, when you LIST the program, all of the lines will list. But Caesar was a ROMAN, not a LOMAN. We'll have to correct that!

By holding down CTRL and pushing ↑, you can get the cursor up to line 10. Then, press → to position it over the L in LOMAN. Retype the letter, changing the L to an R, then move the cursor to the end of the line and press RETURN. Now LIST.

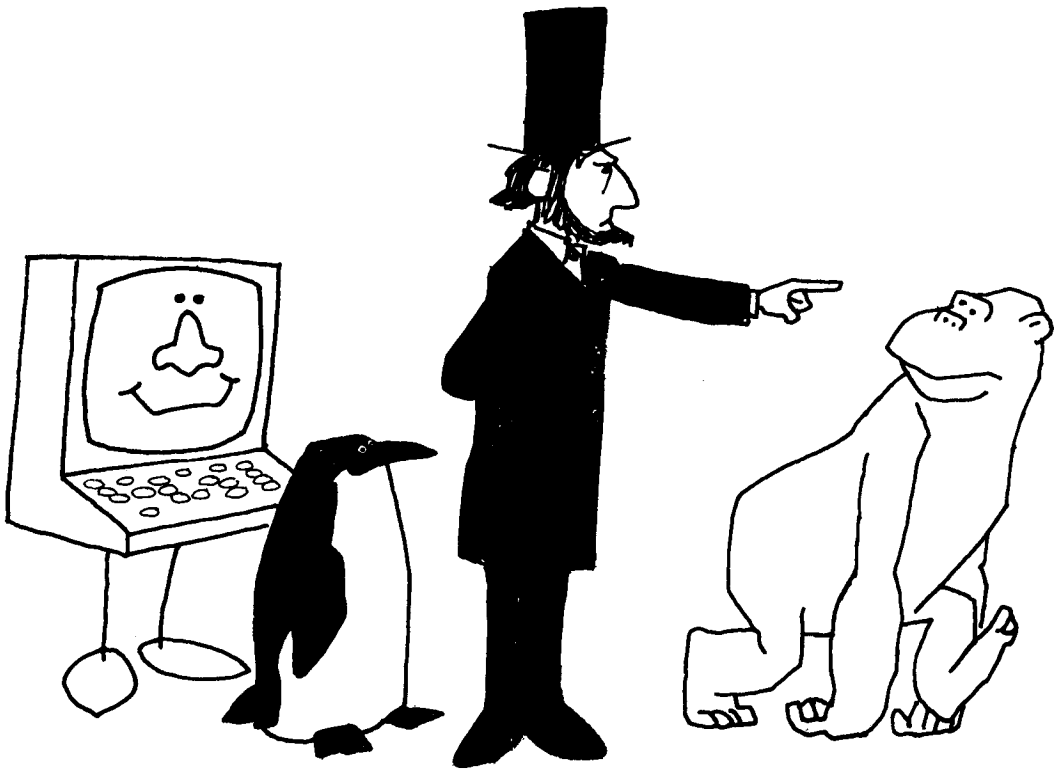


You might think you are finished, but your teacher does not want a word about King Kong in your program. You were supposed to type:

```
30 PRINT ''ABE LINCOLN WAS AN AMERICAN.''
```

All you need to do is type the new line and hit RETURN, and King Kong will be gone!

Another editing trick is to tell the computer to list the line you want to change. Try LIST 20 to see how this works. If you want to see two lines, type LIST 10, 20.



9

String Along

A **STRING**—a **variable** and a $—is made up of one or more **characters.** A string can be interesting. For example, try this:

```
 5 DIM A$(25)
10 A$ = ''GOOD MORNING''
20 PRINT A$
```

← *DIM tells the computer to save enough **memory** for your string. The number in parentheses tells the computer how much memory to save. It should be larger than the string you anticipate.*

What happens?

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

Try this. Type it exactly as it is here:

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

SPACE — NO SPACE — SPACE — NO SPACE

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

10

INPUT

Here is a program to try with a friend:

```
 5 DIM A$(2)
10 DIM N$(20)
15 PRINT ''TYPE IN YOUR AGE''
20 INPUT A$
25 PRINT ''TYPE IN YOUR NAME''
30 INPUT N$
35 PRINT N$; ''YOUR AGE IS ↑'';A$
                            SPACE
```

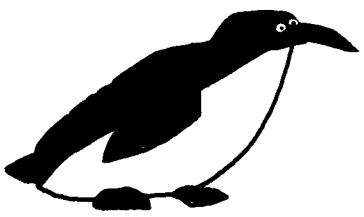
INPUT lets you type in a response while the program is running. It is always used with DIM to reserve enough memory for the input.

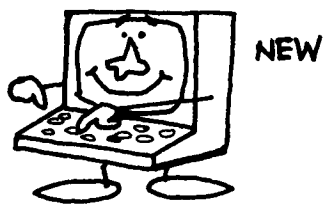
Push SHIFT and CLEAR and then RUN the program.

Let's get a bit fancier. Try:

```
35 PRINT ''YOU ARE ↑'';A$;'' ↑YEARS OLD ↑'';N$
                  SPACE    SPACE      SPACE
```

When you are finished with this program, type NEW.



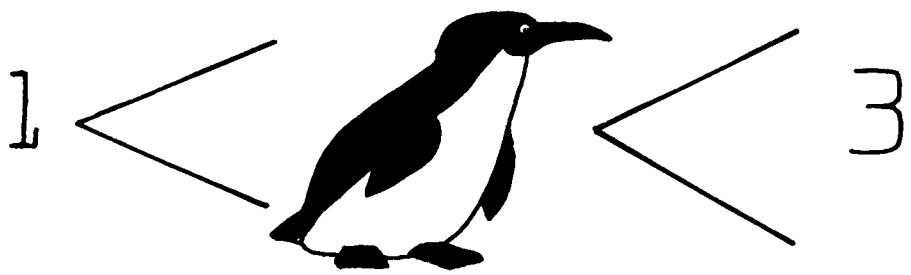


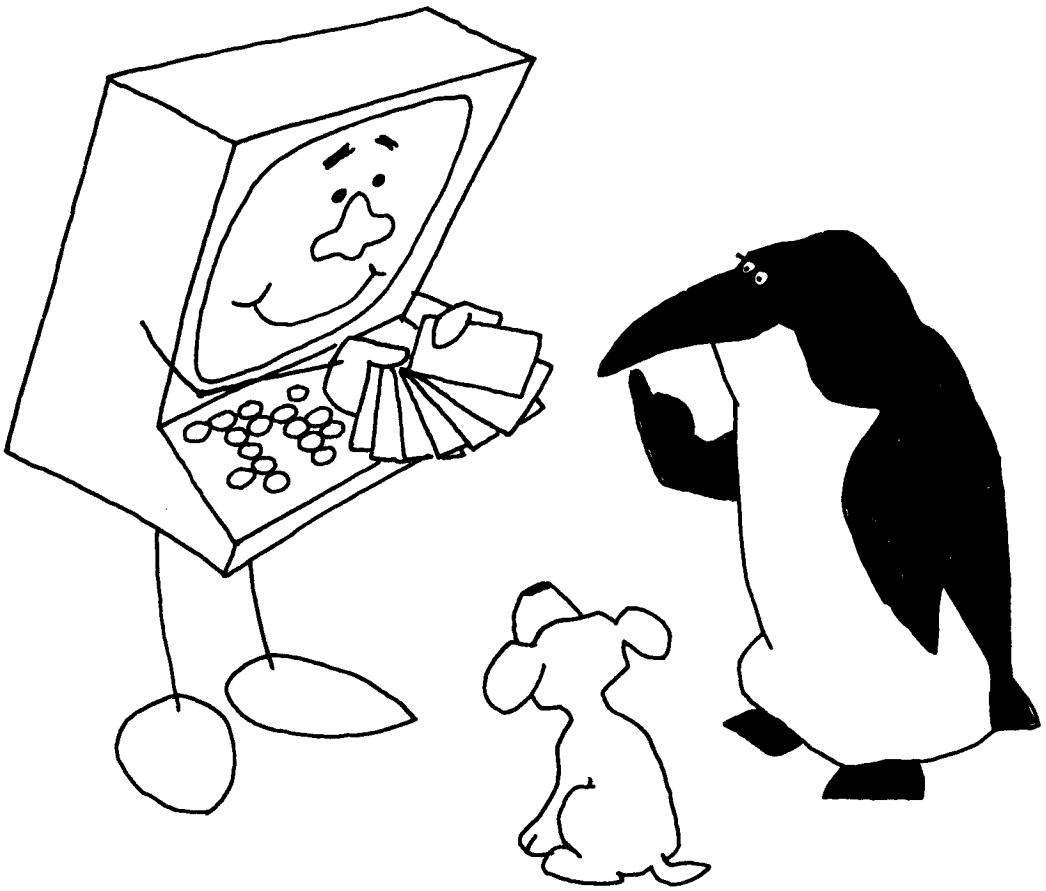
11

Pick and Choose

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

```
 5 DIM N(1)
10 PRINT ''PICK A NUMBER. TYPE IT IN. YOUR CHOICES
   ARE 1, 2, OR 3.''
20 INPUT N
30 IF N < > 2 THEN GOTO 60
40 IF N = 2 THEN PRINT ''YOU ARE CORRECT.''
50 END
60 ? ''NO, TRY AGAIN.''
70 GOTO 20
```





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

< = is less than or equal to

> = is greater than or equal to

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

12

Blast Off!

Use the "Pick and Choose" program from the last lesson. Change line 40 to:

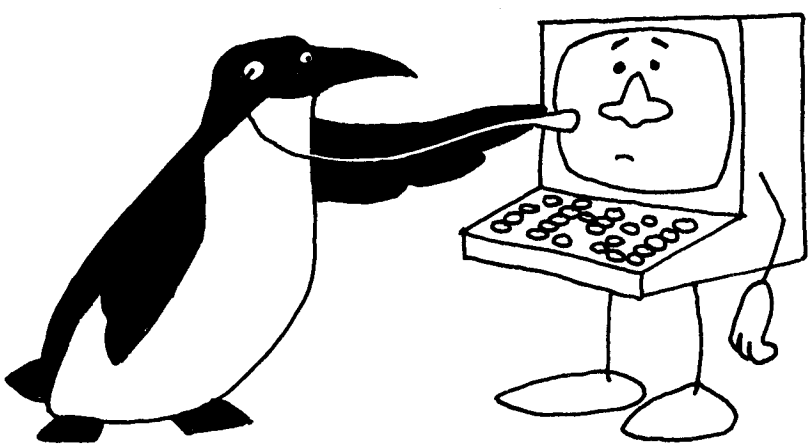
```
40 IF N = 2 THEN GOTO 200
```

Add:

```
45 ? ''YOU ARE CORRECT.''
```

Now we are going to add sound!

```
200 FOR X = 26 TO 69
210 SOUND 0, X, 10, 8
215 FOR T = 1 TO 150
220 NEXT T
225 NEXT X
230 GOTO 45
```



13

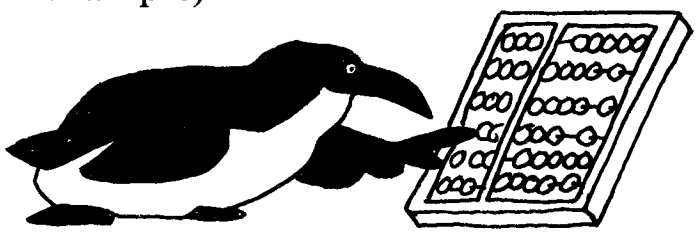
Math Magic

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

```
PRINT 3 + 4
```

The computer can do six different arithmetic operations. Try all of them.

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



14

Some Strange Numbers

Type:

```
PRINT 04.340
```

What do you see? The computer does not print the end zero—or the beginning one, either.

Now type:

```
PRINT 498575.6898
```

What do you see this time? The computer rounds off. To what number did it round off?

Now type:

```
PRINT 212.50514401441
```

What does the computer print?

This time type:

```
PRINT .50514401441
```

What does the computer print? Why? How many digits does each number have?

To get an idea of what computers do with numbers

in expressions, do these in your head or on paper first, and then do them on the computer.

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

The computer follows this order for carrying out all math operations:

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

Now, using parentheses, write these expressions more clearly.

15

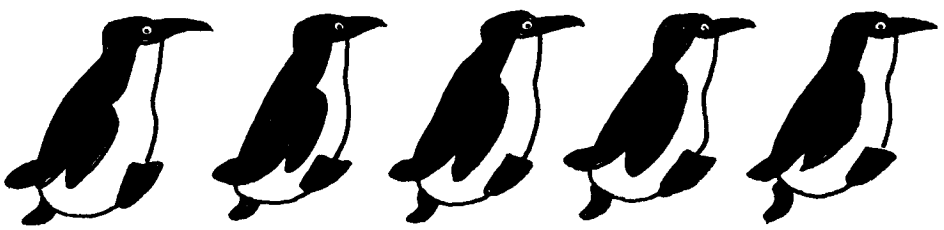
Repeat

One advantage of computers is their ability to perform repetitive tasks quickly and without getting bored.

Follow these programs involving square roots and see the shortcuts that develop. SQR is computer language for "square root" and SQR(X) is the way to write "square root of a number." The letter X stands for whatever number is being used.

Try this program:

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



Try this shortened form of the same program:

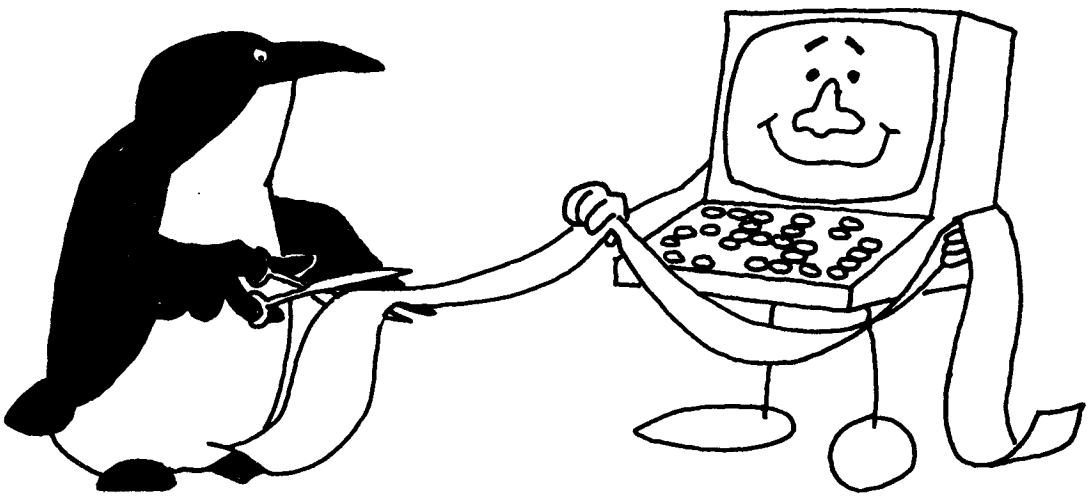
```
10 N = 1
20 PRINT N, SQR(N)
30 N = N + 1
40 IF N < = 10 THEN GOTO 20
```

How does this four-line program compare with the ten-line program you ran first?

Using the loop from the last program we can shorten the program even more:

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

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



16

At Random

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

To see what RND(X) does, try this:

```
PRINT RND(0)
```

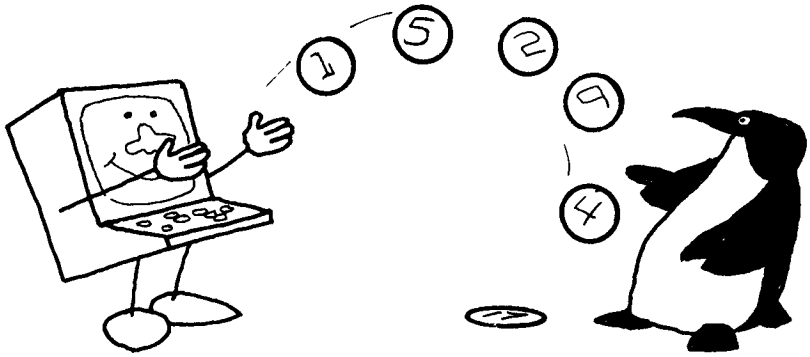
What did you get? Try it again. Did you get the same number this time?

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

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

To get rid of the numbers after the decimal, try this:

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



INT is a code for **integer.**

What happens if you multiply by 100? Experiment! Try changing the RND(0) to RND (other things).

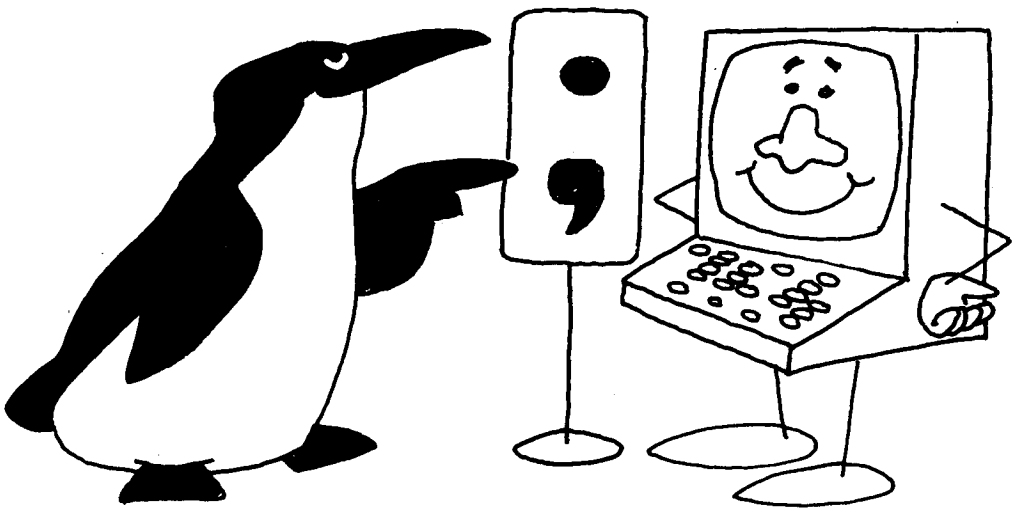
Try this program. Predict what will happen before you RUN it.

```
10 FOR N = 1 TO 10
20 PRINT INT (10 * RND(0))
30 NEXT N
```

Change lines 10 and 20 to:

```
10 FOR N = 1 TO 70
20 PRINT INT (10 * RND(0));
```

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



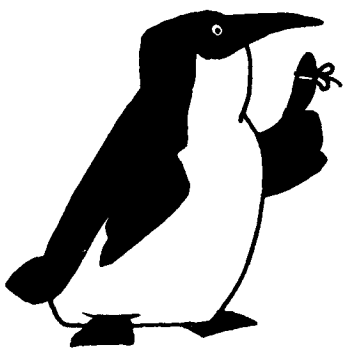
17

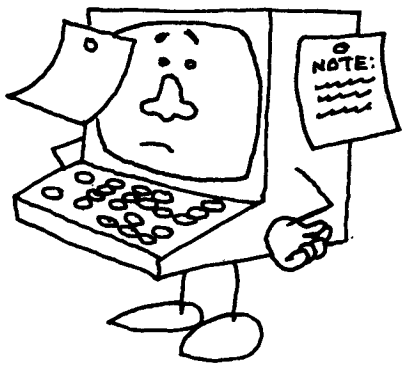
REMark

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

Programmers use REM to:

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





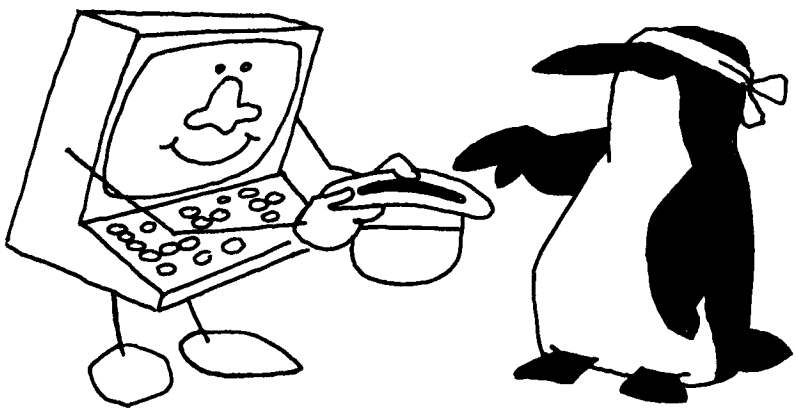
18

Guess a Number

Here is a number game using RND, INPUT, and other things you have learned.

```
  2 REM GUESS A NUMBER
 10 G = 1 + INT (100 * RND(0))
 15 PRINT ''GUESS A NUMBER FROM 1 TO 100.''
 20 DIM R(4)
 30 INPUT R
 40 IF 4 = G THEN GOTO 100
 50 IF R < G THEN GOTO 80
 60 PRINT ''TOO HIGH.''
 70 GOTO 30
 80 PRINT ''TOO LOW.''
 90 GOTO 30
100 PRINT ''VERY GOOD, YOU GUESSED IT!''
```

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



19

Modes

The microcomputer has four modes:

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

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

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

Your computer should have answered 15 immediately.

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

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

The editing mode, as you learned earlier, lets you make changes and correct mistakes.

20

READ/DATA

The purpose of this program is to play a game in which a friend tries to guess one of the numbers in lines 110 and 120 (the statements are called **DATA** statements).

See if you can predict the purpose of **READ:**

```
  2 REM GUESSING GAME
  5 DIM G(3)
 10 PRINT ''PICK A NUMBER.''
 20 INPUT G
 30 READ D
 40 IF D = 99999 THEN GOTO 80
 50 IF D < > G THEN GOTO 30
 60 PRINT ''YOU ARE CORRECT.''
 70 END
 80 PRINT ''WRONG, TRY AGAIN.''
 90 RESTORE
100 GOTO 10
110 DATA 9, 15, 18, -60, 242, 0, 80
120 DATA 78, 4, 15, 25, -22
130 DATA -99999
```

After you have entered your program, use SHIFT and CLEAR to clear the screen so that your friend does not see the numbers in the data statements.

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

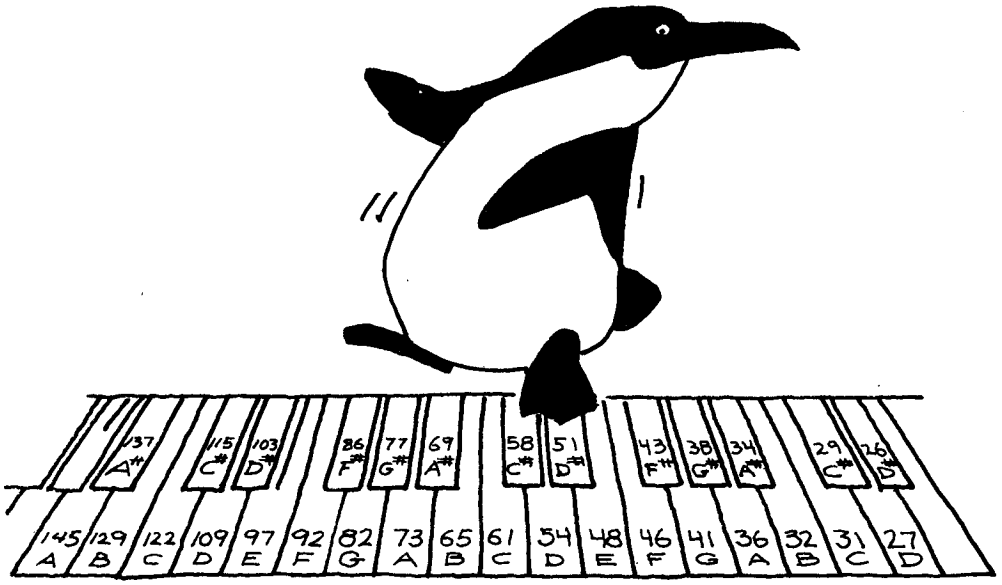
21

Make Music

The wonderful part of READ/DATA is that it enables you to write music on the computer. If you know the numbers of the notes, you can put them into the program as data.

Here are the notes and their numbers. The lower numbers are the higher notes:

26 D#	48 E	92 F
27 D	51 D#	97 E
29 C#	54 D	103 D#
31 C	58 C#	109 D
32 B	61 C	115 C#
34 A#	65 B	122 C
36 A	69 A#	129 B
38 G#	73 A	137 A#
41 G	77 G#	145 A
43 F#	82 G	
46 F	86 F#	



Now let's use the notes to make computer music. Here's a simple song to get you started:

```
  2 REM MAKE MUSIC
 10 READ D
 20 IF D = -99999 THEN GOTO 80
 30 SOUND 0, D 10, 8
 40 FOR T = 1 TO 50 NEXT T
 50 IF D < > -99999 THEN GOTO 10
 80 END
100 DATA 48, 54, 61, 54, 48, 48, 48
110 DATA 54, 54, 54, 48, 41, 41
120 DATA 48, 54, 61, 54, 48, 48, 48
130 DATA 48, 54, 54, 48, 54, 61
140 DATA -99999
```

Write some music on your own. Work with a friend if you wish.

22

Graphics

The ATARI® offers you three different screens on which to do graphics. You can do graphics using large, medium, or small screens. To see what all three look like, type the following:

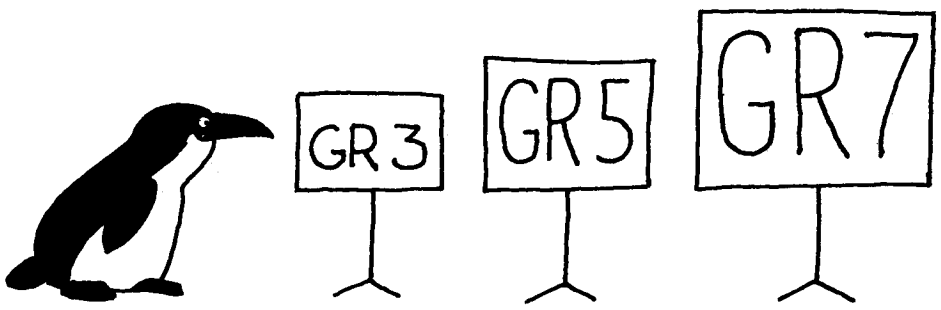
```
GR.3
```

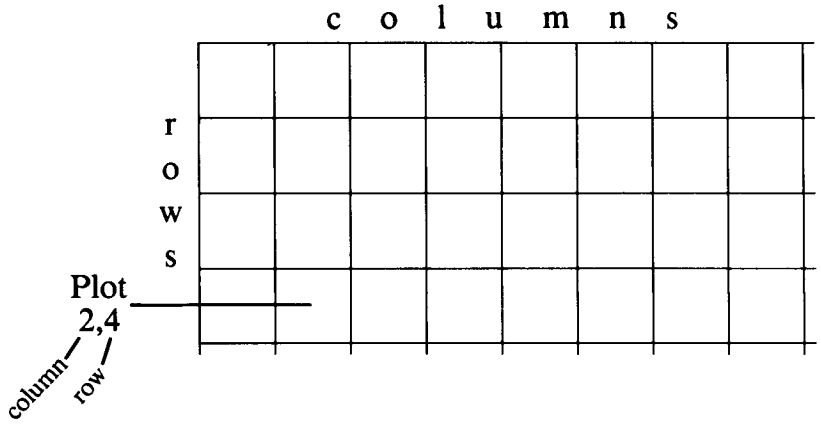
To go back to the original screen, type:

```
GR.0
```

The difference in the three screens is that each has a different number of columns and rows.

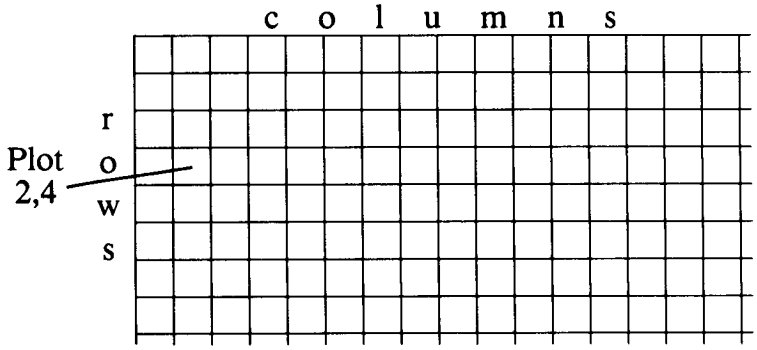
GR.3 has 39 columns + 20 rows.
GR.5 has 709 columns + 40 rows.
GR.7 has 158 columns + 80 rows.





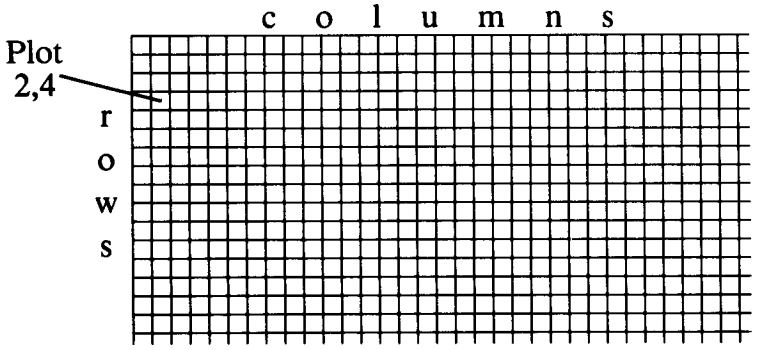
screen GR.3

The squares in this screen are large.



screen GR.5

The squares in this screen are smaller.



screen GR.7

The squares in this screen are tiny.

23

GOSUB

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

The computer will run the subroutine and then return to the next line after the one in which it was told to go to the subroutine—whatever the line number may be. Usually it saves time to build a subroutine only if it involves a number of steps.

We'll demonstrate GOSUB and the difference in the three graphics screens all at the same time. Try typing in this program:

```
 10 COLOR 1          ← Changes the color of the square to gold
 20 FOR G = 3 TO 7
 30 GR.G
 40 GOSUB 100
 50 G = G + 1
 60 NEXT G           ← Sends the program back to line 20
 70 GR.0             ← Restores the screen to its regular configuration
 80 END
100 PLOT 10,5
110 FOR T = 1 TO 400 ← Counts to 400 to delay
120 NEXT T
130 RETURN           ← Sends the program back to line 50
```

24

Going Straight

Now that we know how to find a point on the screen, let's draw a straight line on each of the three screens. Here's how on screen GR.3:

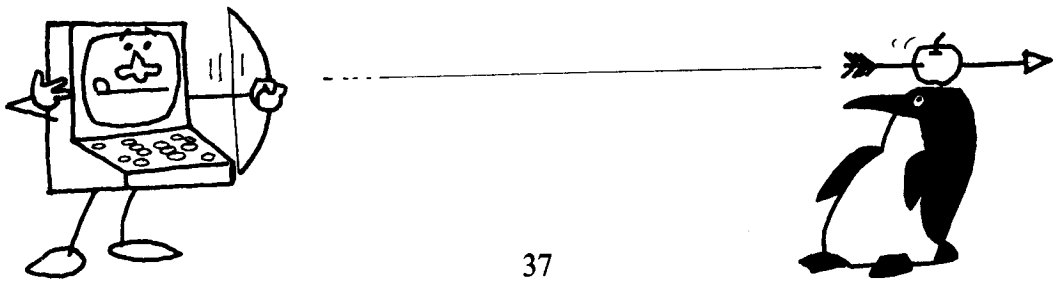
```
10 GR.3
15 COLOR 1
20 FOR T = 0 TO 19
30 PLOT 5, T
40 NEXT T
```

On screen GR.t do this:

```
10 GR.5
15 COLOR 1
20 FOR T = 0 TO 40
30 PLOT 5, T
40 NEXT T
```

Do it this way on screen GR.7:

```
10 GR.7
15 COLOR 1
20 FOR T = 0 TO 79
30 PLOT 5, T
40 NEXT T
```

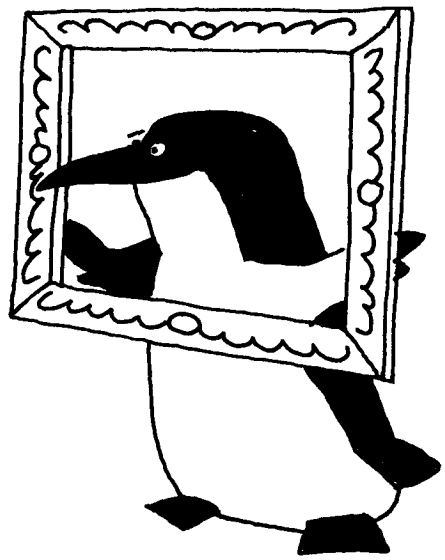


25

Getting Framed

Now let's try drawing more than one line. In fact, let's make a frame. Try this on screen GR.3:

```
10 COLOR 1
20 GR.3
30 FOR T = 0 TO 19
40 PLOT 0, T
45 PLOT 39, T
50 NEXT T
60 FOR P = 0 TO 39
65 PLOT P, 0
70 PLOT P, 19
75 NEXT P
```



On screen GR.5:

```
10 COLOR 1
20 GR.5
30 FOR T = 0 TO 39
40 PLOT 0, T
45 PLOT 79, T
50 NEXT T
60 FOR P = 0 TO 79
65 PLOT P, 0
70 PLOT P, 39
75 NEXT P
```

On screen GR.7:

```
10 COLOR 1
20 GR.7
30 FOR T = 0 TO 79
40 PLOT 0, T
```

```
45 PLOT 158, T
50 NEXT T
60 FOR P = 0 TO 157
65 PLOT P, 0
70 PLOT P, 79
75 NEXT P
```

Try the following on the GR.3 program you just did. Do you want to see the jailhouse window? Try adding the bars this way:

```
41 PLOT 10, T
42 PLOT 20, T
43 PLOT 30, T
```

Now let's build up the side walls of the prison:

```
44 PLOT 1, T
46 PLOT 2, T
47 PLOT 38, T
48 PLOT 37, T
```

We can add the top and bottom walls:

```
66 PLOT P, 1
67 PLOT P, 2
71 PLOT P, 18
72 PLOT P, 17
```

And now add this:

```
80 PRINT ''HELP! I'M BEING HELD PRISONER.''
85 FOR T = 1 TO 100000
86 NEXT T
90 GOTO 80
```

26

Colorful Colors

Gold is not the only color that the Atari® produces. Try this:

```
10 COLOR 0
```

What color do you get?

Now try:

```
10 COLOR 2
```

What color do you get?

This time try:

```
10 COLOR 3
```

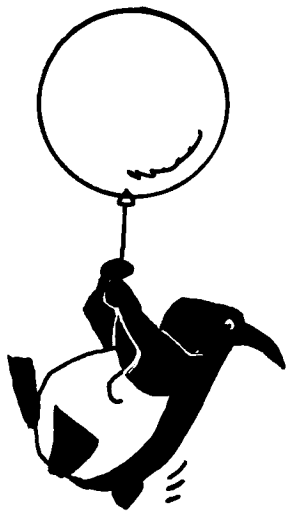
What color do you get?

27

Weightless Name

Sometimes it is handy to be able to print what you want where you want it. Try this program:

```
  2 REM WEIGHTLESS NAME
  5 GR.0
 10 DIM A$(30)
 20 ? ''TYPE IN YOUR NAME''
 30 INPUT A$
 40 PLOT 12, 5
 50 PRINT A$
```



Now, try this:

```
  2 REM 3 FLOATING AROUND
  5 GRAPHICS 0
 10 DIM A$(20)
 20 DIM B$(20)
 25 DIM C$(20)
 30 DIM D$(5)
 35 PRINT ''TYPE YOUR FIRST NAME''
 40 INPUT A$
 45 PRINT ''TYPE YOUR MIDDLE NAME''
 50 INPUT B$
 55 PRINT ''TYPE YOUR LAST NAME''
 60 INPUT C$
 65 PRINT ''PRESS CLEAR, THEN PRESS RETURN''
 70 INPUT D$
 75 PLOT 7,5
 80 PRINT A$
 85 PLOT 12, 10
 90 PRINT B$
 95 PLOT 17, 15
100 PRINT C$
```

GLOSSARY

BREAK A command that tells the computer to break the program at the end of the current line and to await a new command.

CATALOG Shows a list of all the names of programs on a disk.

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

COMMAND Tells the computer specific things to do. Examples include PRINT, LIST, and RUN.

CURSOR The white square that moves on the screen. It is where the next character will be.

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

DIM Used to reserve memory to go with an INPUT statement.

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

EXECUTION MODE Mode in which computer executes the commands you have given in the program.

FOR/NEXT The FOR statement tells the computer to do something a certain number of times. The NEXT statement comes after the FOR statement and tells the computer to go back and do the action stated. The FOR/NEXT statement is a type of loop.

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

FUNCTION Any one of a number of things the computer does, such as print out random numbers, integers, etc.

GOSUB Command used to send the computer to a subroutine; it is always used with the RETURN command.

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

Example:
```
GOTO 100
```

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

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

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

INPUT Enables the program user to type in a response while the program is running. Always used with DIM to reserve enough memory for the input.

Example:
```
5 DIM A$(20)
10 PRINT ''YOUR NAME'';
15 INPUT A$
20 PRINT A$
```

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

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

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

LIST Shows everything that is in the computer's memory, listing it in line-number order.

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

MEMORY Where information is stored inside the computer. There are two kinds of memory: RAM and ROM. RAM means "Random Access Memory" and is used for the material you enter. ROM is "Read Only Memory" and is used by the computer manufacturer to store languages such as BASIC—and other necessary material.

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

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

PLOT A command that tells the computer where it is to print a character on the screen.

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

Example:
```
PRINT 2 * 5
10
```

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

PROGRAMMING MODE Mode in which computer stores in its memory instructions and their order of execution.

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

RANDOM (RND(0)) A command that allows the computer to choose from a range of expressions between 0 and 1.

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

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

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

RETURN A command at the end of a subroutine that brings the computer back to the next line of the program after that which told it to go to the subroutine.

RETURN KEY The key that must be pushed to enter whatever has been typed into the computer's memory and central processing unit.

RUN A command that starts the program executing.

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

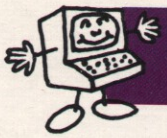
TIMER Delays a program. It leaves a certain amount of time between two parts of a program.

VARIABLE A quantity in a string that can take on any of a set of given values.

ERROR MESSAGES

2 MEMORY FULL
3 VALUE ERROR
4 TOO MANY VARIABLES
5 STRING LENGTH ERROR
6 READ OUT DATA
8 INPUT STATEMENT ERROR
9 ARRAY/STRING DIM ERROR
12 LINE NOT FOUND (GOSUB, GOTO)
13 NO FOR TO GO WITH NEXT
14 LINE TOO LONG
15 GOSUB/FOR LINE DELETED
16 BAD RETURNS

INDEX



Kids Working with Computers

THE **ACORN® BASIC** MANUAL

THE **APPLE® BASIC** MANUAL

THE **APPLE® LOGO®** MANUAL

THE **ATARI® BASIC** MANUAL

THE **COMMODORE® BASIC** MANUAL

THE **COMMODORE® LOGO** MANUAL

THE **IBM® BASIC** MANUAL

THE **IBM® LOGO** MANUAL

THE **TEXAS INSTRUMENTS BASIC** MANUAL

THE **TEXAS INSTRUMENTS LOGO** MANUAL

THE **TRS-80® BASIC** MANUAL

THE **TRS-80® COLOR LOGO** MANUAL

ISBN 0-516-08424-0

CHILDRENS PRESS
REINFORCED BINDING