BASIC Made Easy for the IBM PCjr

Education Series





IBM Software for IBM Personal Computers **Building blocks to BASIC.**

Learn to write programs, create music and draw colorful pictures ...in eight friendly and fun lessons.



Personal Computer Education Series

BASIC Made Easy for the IBM PCjr

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: International Business Machines provides this manual "as is", without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this manual at any time and without notice.

This product could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of this publication.

Products are not stocked at the address below. Requests for copies of this product and for technical information about the system should be made to your Authorized IBM Personal Computer dealer.

A Reader's Comment Form is provided at the back of this publication. If this form has been removed, address comments to: IBM Corporation, Personal Computer, P.O. Box 1328-C, Boca Raton, Florida 33432. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligations whatever.

© Copyright International Business Machines 1983

About This Book

n

Π

h

=

BASIC Made Easy is for those who have no knowledge of computers. If you have worked with computers before, or if you know something about computer languages, feel free to whiz through the book to get a feel for the unique features of the IBM PC*jr* and its version of BASIC.

The book is broken up into four chapters with two lessons in each: Chapter One starts you off with simple one-line programs; Chapter Two teaches you how to build programs that are several lines long; Chapter Three shows you how to interact with your own programs; and Chapter Four teaches you how to add color and sound to what you have learned.



We assume that you have Cartridge BASIC for your IBM PCjr and that you are using a 40-column color television set. Most examples will work with other 40-and 80-column monitors as well. Colors and intensities used in Lessons 5 and 7, however, may vary depending on your display device.

If you have never used a computer before, you will want to work through the book carefully. As you finish each lesson, hopefully you will feel more confident and more comfortable with the computer.

We suggest you don't read through this book in an armchair or while propped up against a pillow. Study it in front of your computer. Try out every program in as many ways as you can, making notes as you go along. Don't hesitate for fear you might make a mistake. Your mistakes won't hurt the computer. In fact, you'll learn a lot more by making a few mistakes.

We also suggest that you do **not** try to go through all the lessons in one sitting. You will tire and lose interest as you try to remember all the information. Try completing one chapter at a time and review the material thoroughly, then take a break. Come back later and complete the next chapter. This will help keep the text informative and entertaining for you. In the text, programming examples help teach the lessons. At the end of each lesson, projects (and their solutions) help you practice what you've learned. Both types have been kept simple and light-hearted. To solve them, you won't need any advanced math—just a little simple arithmetic. To inspire you, we've also thrown in some ideas here and there for you to build your own useful or entertaining programs.

We want you to enjoy every minute of this book so we've put you in charge of teaching yourself. You learn by doing. You decide how fast or slow to go, or when you need a break; your computer is a very obedient and patient helper. Take as long as you like to learn how to give the computer instructions.



And...oh...by the way, you're also in charge of what the computer does. Don't expect it to tap you on the shoulder and say, "Psst! I had a great idea last night that you simply must try!"

Note: Students younger than 12 years of age may need some guidance from an adult.

Contents

Introduction	. 1
What You Need	. 3
Additional Reading	. 3
Starting the IBM PC <i>jr</i>	. 4
To Start Cartridge BASIC	. 5
To Start Cartridge BASIC While Using	
DOS	. 8
Chanter 1. Cetting Started	0
About This Chapter	.) 0
Lesson 1 Instant Programming	.) 10
The PPINT Command	10
The CLS Statement	12
What Is a String?	12
Correcting Mistakes	13
The BEEP Statement	14
The SOLIND Statement	17
Your Turn	10
Solutions	20
	21
Taking a Break?	21
Lesson 2 Writing Programs	24
Is the Computer Off?	25
Your First Big Program	26
LISTing Your Program	28
The REM Statement	30
Error Messages	30
Editing Your Program	32
The NEW Command	34
Spacing and Print Zones	34
A Direct Statement: GOTO	35
Your Turn	39
Solutions	40

Chapter Review	41
Taking a Break?	42
C C	
Chapter 2: Programs That Count	43
About This Chapter	43
Is Your Computer Switched Off?	44
Error Messages	44
Lesson 3. The Computer Remembers	45
Letters Equal Numbers	47
The INPUT statement	48
Your Own SOUND Track	51
Quick Review	54
Your Turn	55
Solutions	56
Lesson 4. A Variety of Variables	57
String Variables	59
Four String Functions (or: A String	
Quartet)	62
READ and DATA Statements	66
FOR and NEXT Statements	70
Your Turn	73
Solutions	74
Chapter Review	75
Chapter 3: Branches and Loops	. 77
About This Chapter	77
Error Messages	78
Lesson 5. The Great Screen Escapade	. 81
Printing Anywhere On the Screen	. 81
Be a Math Whiz with FOR and NEXT .	. 84
A Spot of Color	88
Your Turn	92
Solutions	93
Lesson 6. The Computer Decides	. 94
A Fork in the Road	94
Setting A Clear Path	. 96
How the Subroutine Works	. 100
Your Turn	102
Solution	103

- -

_

	Chapter Review
Xuladi	Chapter 4: Games and Show Business 105
	About This Chapter
	Error Messages 105
	Lesson 7. Music
	Play Maestro Play 107
-	Once Again, With Feeling 111
	Fast or Slow 112
	High or Low 113
	Three-Part Harmony
	Volume and Countermelody 121
	Your Turn 124
	Solutions 125
	Lesson 8. Art 126
Milita i,	The Artist's Touch
	The PSET Statement
	The LINE Statement
	The CIRCLE Statement
	In Living Color
Feedback	The PAINT Statement
	Hooray for You
finite and	Your Turn 143
	Solutions 144
formali	Chapter Review
	Appondix A Mossages 147
	Appendix A. Messages
-	Appendix B. BASIC: A Quick Reference
~	Commands 153
	Statements
,	Appendix C. Reserved Words in BASIC 157
~	Index Index-1

Notes:

_ -

-

-

.

-

-

-

-

-

4

-

-

Introduction



BASIC Made Easy introduces you to the BASIC computer language on the IBM PC*jr*. The first thing to learn is that *BASIC* stands for Beginner's All-purpose Symbolic Instruction Code and that it was designed with you, the "beginner," in mind.

The second thing to learn is that a computer language, like BASIC, is similar to a foreign language in that you use a different set of words to convey a message. But in this case, you are conveying a message to a computer and not another person. You'll learn also that a computer is not forgiving when given an incorrect "translation" of a message. You have to convey the message in the exact form that the computer knows or it will give you an error message. Remember to be consistently accurate in your messages with the computer, and you and the computer will get along well.

....

Learning a computer language is a lot easier than it sounds. The first big hurdle is computer vocabulary; you have to learn at least some of it to have the computer work for you. It would be like asking a carpenter to build a house without blueprints; if the carpenter isn't given instructions (blueprints) he understands, he can't begin to work.

You can learn the essentials of BASIC in a few hours—much sooner than it takes to learn a foreign language or build a house. In fact, you'll find that learning BASIC is more like playing with pieces of a construction set. Once you know the shapes and how to join pieces, you will discover new uses for them. We believe this book will teach you those "building block" essentials of BASIC. The following is the minimum hardware requirement to perform the *BASIC Made Easy* examples:

- IBM PCjr with 64 KB of memory
- 40- or 80-column color television or color monitor
- BASIC Cartridge

These units should be hooked up according to the Setup Instructions that come with the IBM PC*jr*.

Additional Reading

Before using this book, you should be familiar with Chapter One and Chapter Two of the IBM PC*jr Guide* to Operations. It shows you how to use the IBM PC*jr* keyboard and also how to spot and solve some common problems.

After reading BASIC Made Easy, you may want to look into other advanced topics of the BASIC computer language. If so, the IBM PC*jr* BASIC or the IBM PC*jr* Hands-On BASIC books are for you. These books give complete information on the BASIC language on the IBM PC*jr*.

Starting the IBM PCjr



Before we get into the lessons, let's learn how to start the IBM PC*jr*. Eventually while reading the book, you may to take a break between lessons. You can refer back to this section each time for start up instructions.

For now, pick one of the following headings that best fits your situation: if your computer is off, see the next page, if your computer is on, see page 7, if you've already started with DOS, see page 8.

To Start Cartridge BASIC

If Your Computer Is Off

- 1. Turn on the TV set (or color monitor). Set the volume control on low (if available).
- 2. If you have a diskette drive, remove any diskette that may be in it.
- 3. Slide the open end of the BASIC cartridge into either of the cartridge slots. The label should be facing up.



4. Find the power switch at the left rear of the IBM PC*jr* System Unit and flip it on.

First you see the IBM logo. A few seconds later, you see a screen titled **The IBM PC***jr* **BASIC**. On the second line of the screen, **Version J** lets you know that you are using Cartridge BASIC. The third line tells you how much memory is available. The fourth line says **OK**. OK means that the computer is ready.

5. If you cannot see the words to the left of the TV screen, adjust the picture as follows: press and hold down the Ctrl and Alt keys, then press the Cursor Right (Pg Dn) key once or twice until the picture is centered. In the same way, press the Cursor Left (Pg Up) key to move the picture left on the screen.



6. The line at the bottom of the screen describes certain special keys. Since you don't need this information right now, type **KEY OFF** and press the

Enter key . The IBM PC*jr* returns an OK to tell you it did what you said, and that it's now ready for something else.

7. You are ready to begin Chapter 1: Getting Started. Turn to that section now.

If Your Computer is On

- 1. If you have a diskette drive, remove any diskette that may be in it.
- 2. Slide the open end of the BASIC cartridge into either of the cartridge slots. The label should be facing up. This causes the System Unit to restart itself.

First you see the IBM logo. A few seconds later, you see a screen titled **The IBM PC***jr* **BASIC**. On the second line of the screen, **Version J** lets you know that you are using Cartridge BASIC. The third line tells you how much memory is available. The fourth line says **OK**. OK means that the computer is ready.

- 3. If you cannot see the words to the left of the TV screen, adjust the picture as follows: press and hold down the Ctrl and Alt keys, then press the Cursor Right (Pg Dn) key once or twice until the picture is centered. In the same way, press the Cursor Left key (Pg Up) to move the picture left on the screen. (See previous page for illustration.)
- 4. The line at the bottom of the screen describes certain special keys. Since you don't need this information right now, type **KEY OFF** and press the

About This Chapter

-

This chapter has two lessons. In these lessons, you will learn to give simple commands to the computer and see the results on the screen. When you're done with Lessons 1 and 2, you will be able to:

- Write one-line programs that work instantly.
- Correct your typing mistakes, delete something you don't want, and clear the screen before typing a new program.
- Use two commands to create sounds on the computer.
- Correct one or two errors that may come up on your screen.

Lesson 1. Instant Programming



The PRINT Command

Press a few keys and watch the screen. Works like a typewriter, right? One of the things a computer can do is remember an instruction and carry it out. Let's jump right in and give IBM PC*jr* our first instruction.

Type your name and press the Enter key **Enter** . (Pressing the Enter key tells the computer to read your instruction and then *perform* it.)

Oops! You got a **Syntax error** which means the computer did not understand your instruction. Don't panic.

The computer understands only specific words called either *commands*, *functions*, or *statements*. One of these specific words is the PRINT statement, which tells the computer to print something on the screen. For the computer, the word PRINT always comes first, followed by what you want to print enclosed in quotes. You then press the Enter key **to perform the** instruction.

Let's try to get it right this time. (To get capital letters, hold down the Shift key.)

1. Type the following line exactly as you see it:

PRINT "Hi! Anybody in there?"

2. Watch the screen and press the Enter key



If you make a typing mistake, press the Esc key ^[5s] (top left corner of the keyboard) to erase the entire line, and start over.

Notice that the computer printed only what you typed between the quotation marks.

Before we go on, look at the keyboard to see where all the characters are located and how to get them.

The CLS Statement

The CLS statement clears the screen so you can start typing at the top again. When you use CLS, you also erase your instruction from the computer's memory if you don't use line numbers. (We'll discuss memory and line numbers in a little bit.)

1. To clear the screen, type:

CLS

- 2. Then press the Enter key **true**. Presto! A fresh screen.
- 3. Now type (in small letters):

print "

- 4. Type your name, then type the closing quotation mark.
- 5. Press the Enter key [Inter].

You can type statements in small letters, if you like. We'll show them in capitals throughout this book, anyway. m

m

Just between you and your computer, a *string* is whatever you type inside quotation marks. The computer prints a *string* exactly as you type it. You must always remember to type the PRINT statement first.



A string can be a word or a bunch of words, or it can be a number or many numbers, such as your telephone number or your age.

Have some fun printing different *strings*—a greeting, your address, your phone number, your favorite movie, or the name of a loved one. Put each message inside quotation marks. Don't forget to use the **PRINT** statement.

Correcting Mistakes

Another way to correct a mistake other than the Esc key is to use the four arrow keys: Cursor Left \overline{reup} , Cursor Right \overline{reup} , Cursor Up \overline{reup} , and Cursor Down \overline{reud} (lower right-hand side of the keyboard). Press these keys to move the cursor up, down, left, or right to the mistake. Then type the correct letter or word, and press the Enter key \overline{reup} . Your correction takes effect *only after* you press Enter.

Also, remember from a few moments ago that if you want to erase an entire line, press the Esc key.

Or, if you're having a bad day, you can clear the whole screen by typing CLS.

.

The BEEP Statement

Every computer loves to do something silly now and then. Your computer loves to beep. Let's use the BEEP statement to make it beep.

Use the CLS statement to clear the screen, then type:

BEEP



Did you remember to press the Enter key? Good. Remember to press the Enter key whenever you want the computer to carry out your instruction.

Also remember that you can type statements in small letters if you prefer.

You can type two statements in the same line, too. Let's try it.

Type:

BEEP: PRINT "I love to beep."

Notice that you need a colon to join two statements in the same line.

Now press Enter to hear and see the program.

There you go! That one line is a program—an instruction to tell the IBM PC*jr* what to do. I'll bet you used the PRINT statement without knowing you wrote a program. Let's see what else we can do with one-line programs and the SOUND statement.

Enter key : The IBM PC*jr* returns an **OK** to tell you it did what you said, and that it is ready for something else.

5. You are ready to begin Chapter 1: Getting Started. Turn to that section now.

To Start Cartridge BASIC While Using DOS

DOS stands for Disk Operating System. You may have previously started your IBM PC*jr* from the DOS diskette.

- If the BASIC cartridge is already in the slot and the DOS prompt A> is on the screen, type the word BASICA after the DOS prompt and press the Enter key [inter].
- If the BASIC cartridge is *not* in the slot, slide the open end of the BASIC cartridge into either slot. (The label should be facing up.) This causes the System Unit to restart itself. After typing in the date and time correctly, type BASICA and press the Enter key [Enter].

1. A

You see a screen titled **The IBM PC***jr* **BASIC**. The second line of the screen reads **Version J** to let you know that you are using Cartridge BASIC. The third line tells you how much memory is available. The fourth line says **OK**. OK means that the computer is ready.

3. You are ready to begin Chapter 1: Getting Started. Turn to that section now.

The SOUND Statement

A beeping computer! Hmmmf, you say. But can it sing a high note? Get ready for this. Clear the screen with CLS, then type:

SOUND 500, 50



Press Enter [III]. The SOUND statement gets a musical tone out of the computer.

Do you see the two numbers with a comma between them? The first number (500) tells the IBM PC*jr* what tone to play; 1000 would be a higher one, 40 would be a low scraping sound. IBM PC*jr* can't go lower than 37, or higher than 32767.

The second number (50) tells IBM PCjr how long or short to play the tone. The bigger the number, the longer the tone. IBM PCjr can use a number from 0 to 65535.

It probably wouldn't be a good idea to experiment with the longest playing time until you learn how to stop it. Somebody at home may not appreciate your creativity. Later, we'll learn the trick of how to stop a program that's already working.

Please do try, however, the SOUND statement with different tone numbers. Make a funny low sound like a large bullfrog, or a long high sound like a tea kettle whistling.



Your Turn

Project #1: Can you combine two SOUND statements to make two tones? Try it. How about four tones in a row?

Hint: You need to use a colon (:) to join statements.

Remember: When typing a zero, don't type the letter O by mistake. The zero key is on the top row of the keyboard.

Project #2: Here are the frequencies (known as Hz) for each of five notes on the piano. You may also know these notes as "Do, re, mi, fa, sol."

C 523 Hz
D 587 Hz
E 659 Hz
F 698 Hz
G 784 Hz

Write a one-line program to play any of these three notes for a duration of about 1/2 second each.

Hint: You need to use a colon (:) to join statements.

Solutions

Project #1

SOUND 200,10:SOUND 250,10:SOUND 300,10:SOUND 400,10

(If you are using a 40-column screen, did you notice that part of the SOUND statement was broken when you tried to type the entire program on one line? Don't worry, the program still works. The computer can read a broken line. It all doesn't have to fit on the same line.)

Project #2: We wrote three notes that will remind you of the nursery rhyme "Three Blind Mice":

SOUND 659,10: SOUND 587,10: SOUND 523,10

Quick Review

EEEEEEEEEEEE

Go ahead and do your own thing with the statements you have learned. Here is a list to help you remember these statements.

- CLS Clears the screen.
- **PRINT** Prints strings (messages inside quotation marks).

BEEP Makes the computer beep.

SOUND Produces musical tones. You can make a tone high or low, long or short.



Taking a Break?

If you need to take a short break, leave the computer turned on until you return. If you want to turn off the computer, you can later restart it by following one of the procedures in the "Introduction".

-	Notes:	
-		
-		
-		
_		
-		
-		
, and the second se		
-		
_		
_		
_		
-		
-		
-		
-		
-		
_		
, 		

Lesson 2. Writing Programs



You have been writing short one-line programs that worked as soon as you pressed the Enter key. In Lesson 2, you will learn to combine many instructions in a program and store the program in the computer's memory. You will learn to perform such a program, display it on the screen, change it, and run it again.

Learning how to program is like learning how to ride a bicycle: starting out can be a little awkward, but with practice, it gets easier.
Before you begin, here are some reminders:

- Don't forget to press Enter after typing a line, or after making a correction.
- Use the four arrow keys to move your cursor where you want it.
- Use the CLS command to clear the screen when it gets cluttered or when you want to start over. It does not clear away what you told the computer to remember.
- Use quote marks after a PRINT command when you want to print a *string*.
- If you prefer, type a statement in small letters.

Is the Computer Off?

If your computer is off, refer back to the "Introduction" and follow the procedure. In time, you will know this procedure by heart and may not have to refer to the "Introduction."

Your First Big Program

A program is a set of instructions to the computer. So far, you have been writing instant one-line programs that work when you press the Enter key, and vanish from the computer's memory when you clear the screen.

You now will learn to write a program that stays in the computer's memory after you press the Enter key. Each line in such a program begins with a number (called, naturally, a line number). The difference is that you can perform such a program many times because the computer remembers it.

Let's type a four-line program. First, clear the screen by using the CLS statement. Then type each line exactly as you see it (including the spaces).

```
10 CLS
20 PRINT "Noon whistle at the factory."
30 SOUND 600, 50
40 END
```

Note: Remember to press the Enter key after typing each line. This time, however, nothing will happen immediately.

You just typed a program with four numbered lines. Line 10 tells the computer to clear the screen. Can you guess what Lines 20 and 30 will do? Sure you can! Line 40 tells the computer to stop reading instructions. END is another statement like PRINT or CLS.

Now, let's tell the computer to *perform* this program. Type **RUN**, then press the Enter key [1].



RUN is one of those computer words that was mentioned in "About This Book" in the front of the book. Literally it doesn't make sense to "run" a program. How can a set of instructions run? What RUN means in the computer sense is to operate like a machine. You've often watched a blender or a washing machine run. Like these, you "run" a program. RUN is an example of a word in the BASIC language that you learn to make the computer work for you.

To hear the whistle again, type **RUN** once more, and press the Enter key [Inter].

Another way to run the program which saves typing is to press the FN key (upper right corner of the keyboard) and then press the F2 key. (See the figure for their locations.)



Throughout the rest of the book, we'll use Fn and F2 as shown above to run a program.

LISTing Your Program

Let's look at that program again. Oops, where did it go? It's stored in the computer's memory because you started each line with a line number. To see it again, type **LIST**, and press the Enter key [ner]. This is known as *listing* the program.

As you may have wondered, another way to list (or display) the program is to press the Fn key and then the F1 key. (See the figure on the next page.) When the word **LIST** appears on the screen, press the Enter key. We'll use Fn and F1 keys throughout the book like Fn and F2. There is one more key combination we'll learn later.



m

When you write a program like our "noon whistle" program, line numbers like 1, 2, 3, and 4 would work as well as 10, 20, 30, and 40. The computer reads and does each command in numerical order. Numbering lines in tens (10, 20, 30), however, allows you to add new lines in between.

Let's add a new line. Below Ok, type:

35 PRINT "It's lunch time."

Did you press the Enter key? Good. Now list the program again by pressing Fn [n], and then F1 [n]. Notice that Line 35 slid into place just after

Line 30. Let's go ahead and run the program, but before you press Fn and then F2, make sure the cursor is below the program. You will get an error message if the cursor is not below the last line of the program. When you've checked, press Fn [fn] then F1 [f1] to RUN the program.

The REM Statement

One more thing. Let's give your program a title. Add this line to your program. Type:

5 REM My First Program

Press Enter [1], of course. Now, clear the screen with **CLS** and list the program with Fn [1] and then F1 [1]. Neat, huh? You can also use REM as a reminder or note in any part of the program. (REM stands for "remark.") Run the program again, if you like. Notice that your title did not get printed. It shows up only when you list a program.

Error Messages

Sometimes a program you type may not work because of an error in it. Just because you got an error message does not mean there is a big problem with the computer, the program, or you. The error could be a misspelling, a misplaced word or letter, a missing or wrong line number, or something else that the computer did not understand. So don't panic.

Syntax error is an example you saw in Lesson 1. While using this book, you may see an error message from the list below. If you see a message not in this list, please refer to Appendix A in this book where frequently seen error messages are explained. Error messages you are likely to see while doing Chapter 1 are:

• Illegal function call

The computer cannot do what you ask. LIST your program and check line numbers, commands, and statements. Then rerun the operation.

Missing operand

Your program is missing a part of an instruction. LIST your program and check it over before re-trying the operation.

• Syntax error

_

A program line, command, or statement was typed incorrectly. The incorrect line is displayed for you to correct.

For a complete list of error messages, refer to Appendix A of the *BASIC* reference book.

Editing Your Program



Editing means changing or correcting. An easy way to change a line is to use the EDIT command followed by the line number of the line you want to change.

When editing, remember that your program is stored in the computer's memory. To store your changes properly, you must press Enter after editing and *while* the cursor is still on the line.

Let's edit line 35 by adding a few words. Type:

EDIT 35

Press Enter [Inter] . Line 35 is now on the screen.

 Move your cursor under the I in It's. (Use the Cursor Right arrow key .) 2. Find the Ins (Insert) key into in the lower right part of the keyboard. Press it.

This allows you to add words in a line. (You know the Ins key has been pressed by the square flashing cursor. Pressing Ins again gives you the normal cursor again.)

3. Type:

ή

Your attention, please!

Leave a space before It's.

4. Press Enter (This stores the added words in the computer's memory.)

By pressing the Enter key or any of the four cursor control keys, you turn off the ability to insert words. You would have to press the Ins key to use it again.

- 5. Clear the screen with CLS and LIST the program.
- 6. RUN the program and check it over.

You can rewrite a line the same way you add a line. Simply type the line number, a space, then your new BASIC line. Pressing the Enter key exchanges the new line with the old. If you don't press the Enter key, the change is not made.

Here are some more editing tips. Be sure to try each of them on your program.

- To LIST a single line, type LIST and the line number (separated by a space), then press Enter.
- To erase a line in the computer's memory, simply type the line number and press Enter.

- To print a blank line, type **PRINT** and press Enter.

The NEW Command

Before going any further, let's clear your program from the computer's memory. You do this by using the NEW command.

Type:

NEW

and press the Enter key .

Remember: When you type CLS, the screen is cleared, but the computer still remembers your program. When you type NEW, your program is no longer in the computer's memory.

Now type CLS to clear the screen.

Spacing and Print Zones

Did you type NEW to clear your program? If so, type this program:

```
10 REM WARNING!!!
20 PRINT "The"
30 PRINT "sky"
40 PRINT "is"
50 PRINT "falling!"
60 END
```

Run it. The words appear one under the other. To put the words in a line, type a semicolon (;) at the end of lines 20, 30, and 40. (Use the cursor control keys to move the cursor to the end of each line, and press Enter before leaving the line.) Re-run the program. Oops, the words are jammed together. You need to separate the words.

Here's a clue. On line 20, put a blank space after The ("The "). Run the program.

Hmmmm, it worked. Now do the same for lines 30 and 40. (Use the EDIT command to do your editing.)

What would happen if you put a comma in place of the semicolon? Let's find out. First, clear the screen and list your program. Now type commas instead of the semicolons. (Remember to press the Enter key after you edit each line.) RUN the program.

Notice how the words are spread out. A comma tells the computer to put the next word into a new zone. There are two print zones on your 40-column screen (five if you have an 80-column screen).

A Direct Statement: GOTO

You can make the computer do something over and over again by using the GOTO statement.

- 1. Clear your program from memory by using the NEW command.
- 2. Enter this program:

```
5 CLS
10 PRINT "WARNING! The sky is falling!"
20 SOUND 480, 10: SOUND 520, 10
30 GOTO 5
```

Run the program by pressing Fn ^{Fn}, then F2 ²/_{F2}[®].

Do you want to stop the noise? To stop the noise, calmly:

1. Press the Fn fn key and release it.

2. Press the letter **B** key (Break).

Whew, that was close, huh? From now on, we'll call this sequence of keys Fn-Break [n] - [Break]. (After stopping that noise, it should be called something better, like "great helper.") When you "break" a program, the computer shows the line number where it stopped running.

The GOTO statement in our program told the computer to go back to line 5. Each time it did so, the computer repeated lines 5, 10, and 20. Then it went back and repeated line 5 again. Your program has put the computer into what is referred to as an "endless loop."



Let's have some more fun with GOTO. First use the NEW command.

Enter this program and run it:

5 CLS 10 PRINT "Malayalam" 20 SOUND 440, 10 30 GOTO 10

The word prints endlessly in a vertical line while the noise plays on and on. Use our invaluable Fn-Break fn - Break to stop the endless loop.

Now, list the program and add a semicolon at the end of line 10. Do you know what will happen when you run it? Find out by running the program.

Aha! The words are put one after the other until the screen is filled. Stop the loop by pressing Fn-Break $\boxed{Fn} - \boxed{Break}$.

You can use your own name in line 10, if you like. (By the way, the word "Malayalam" can be read forwards and backwards. It's actually the name of a South Indian language.)

You can make the message print slow or fast by changing the second number (10) in line 20. Try it.

Another question. What will a comma instead of the semicolon do in line 10? To get a clue, go back to where we first talked about the comma. Give up? Well, type in the comma and run the program.

How did you do? How have you done so far? Except for the Projects and Chapter Review, you are done with

the first chapter. You may not want to go on to the next chapter until you understand this one. Give yourself a chance to learn.

Your Turn

Project #1: Write a spooky sounding program that also fills the screen with the word "Nevermore." Add a SOUND statement to make the ghostly tone each time the word appears on the screen.

Project #2: Write a program that prints your name, your age, the color of your hair and eyes, your hobbies, and your favorite food. Write a separate statement for each item on the list.

Solutions

Project #1

```
10 CLS
20 PRINT "Nevermore"
30 SOUND 100, 10
40 GOTO 10
```

Use the Fn-Break \boxed{Fn} - \boxed{Break} keys to stop the program.

Project #2: (Since we didn't know your name, we wrote about a special little person.)

```
5 REM Special Little Person
10 PRINT "Name: Sarah Louise"
20 PRINT "Age: 8"
30 PRINT "Hair: Dark Brown"
40 PRINT "Eyes: Light Brown"
50 PRINT "Hobby: Dodging Piano Practice"
60 PRINT "Favorite Food: Crispy Fried Anything"
70 END
```

Here's what you learned in Lessons 1 and 2:

- You can use the PRINT statement to print a *string* (letters, numbers, or spaces between quote marks).
- The BEEP statement makes a beeping sound for 1/4 second.
- The END statement tell the computer that this is the last line of the program.
- The CLS statement clears the screen.
- The NEW command clears your program from the computer's memory.
- The SOUND statement gives you a tone. You can make this tone high or low, long or short. (More about this in the next chapter.)
- You use a colon to join two commands or statements in the same line.
- When you write a statement without a line number, the computer does it right away (when you press Enter). When you add line numbers, you RUN the program by typing RUN or by pressing Fn, then F2.
- You can list a program from memory only if it has numbered lines by typing LIST or by pressing Fn, then F1.
- You can edit a program by moving your cursor and making corrections. The EDIT command prints the line you want to correct.

- A semicolon prints strings together. A comma separates strings into print zones.
- The GOTO statement makes the computer repeat a program from the line you want. You jump out of this "endless loop" by pressing Fn, then Break (B).

Taking a Break?

Leaving the computer on while you take a break is quite all right. However, if you turn it off and need to restart it, use the startup procedure from the "Introduction."

About This Chapter

In this chapter, you will learn a few new statements that allow you to pack more fun and power into your programs. Remember, you're still in charge. So what ever happens, keep that smile of confidence and success on your face.

In Lesson 3, you will learn to:

- Use the PRINT statement with numbers.
- Give values to numeric variables in a program.
- Use the INPUT statement with numeric variables.
- Use the PLAY statement to play a musical tune.

In Lesson 4, you will learn to:

- Use LEN, LEFT\$, RIGHT\$, and MID\$ on strings.
- Use the INPUT statement with character variables.
- Use the READ and DATA statements to simplify your programs.
- Invent simple games, using variables.

Is Your Computer Switched Off?

If the computer is off, follow the startup procedure described in the Introduction.

Error Messages

In Chapter 2, you may see one of these error messages:

• Illegal function call

The computer cannot do what you ask. LIST your program and check the line numbers and statements. Then re-run the program.

• Missing operand

Your program is missing part of an instruction. LIST the program and check it over before re-trying the operation.

• Syntax error

LIST your program and check the spelling and punctuation. Then rerun the program.

• Undefined line number

You have referred to a line number that's not in your program. LIST and check your program before you rerun it.

• Type mismatch

You gave a string value instead of a numeric value, or vice versa. LIST and check the variables and values in your program.

Lesson 3. The Computer Remembers



Anyone out there have trouble with math problems? If you do, the next program may help. Watch this.

Type this one-line program (don't type any quotation marks):

PRINT 5 + 5

Press the Enter key. There. Was that fast enough for you? Try adding a couple of big numbers.

Without quotation marks, the PRINT statement allows you to add, subtract, multiply, and divide numbers. To

multiply, use the asterisk (*); to divide, use the slash (/). Here is a program with four separate problems. We'll use numbered lines this time.

Use the NEW command; then type the following program (don't forget the line numbers or to press the Enter key at the end of each line):

```
5 CLS
10 PRINT 62 + 28
20 PRINT 12 - 5
30 PRINT 4 * 5
40 PRINT 25 / 5
50 END
```



Run the program and check the answers. (Question: Can you print the answers in one line?) Use the NEW command, then clear the screen with the CLS statement.

Now here's a way to print both the problem and the answer.

```
10 PRINT "4 multiplied by 5 = " 4 * 5
20 PRINT "25 divided by 5 = " 25 / 5
30 END
```

The computer prints the problem (between quotation marks) and then attaches the answer. By the way, if you type words in a PRINT statement without quotation marks, the computer prints a zero.

Letters Equal Numbers

The computer loves to equate things. For example, let's tell the computer that A = 42, B = 37, and C = 21. Then we'll ask it to add A, B, and C.

Use NEW, then CLS. Then type and run this program:

5 CLS 10 LET A = 42 20 LET B = 37 30 LET C = 21 40 PRINT A + B + C 50 END

The answer is 100. You can use any letter from A through Z, even whole words (no spaces) to stand for a number. If you've got nothing to do for a while, try a long word like "supercalifragilisticexpialidocious." No more than 40 letters though, please.

Edit line 40 to say LET D = A + B + C. Then in line 45 tell the computer to **PRINT** D.

40 LET D = A + B + C 45 PRINT D

The letter symbols you used are called *numeric variables* because they stand for numbers. The computer remembers the number value you give to the variables A, B, C, and D. The variable A has the value of 42. If you later change A's value to 103, the value of 42 disappears from the computer's memory, and the new value becomes 103.

The INPUT statement

The INPUT statement tells the computer to ask you for some information. For example, the following program will ask you for your age and then tell it to you.

Use the NEW command. Then enter these lines:

```
5 CLS
10 PRINT "What is your age?"
20 INPUT A
30 PRINT "You are" A "years old."
40 END
```

Before you run the program, read the next few paragraphs first. We've just done something different.

The program above is a new type of program from those that we've been writing; this one asks you to answer a question *before* the computer finishes running the program.

Here's how it works. The INPUT statement in line 20 puts a question mark on the screen. Line 30 assigns the age you will type into the variable "A" which is sandwiched between the two strings in quotation marks.

Let's see how it works. RUN the program with Fn-F2 $[r_{p_2}]^{r_{p_1}}$ and type your age after the question mark, then press the Enter key $[r_{p_2}]^{r_{p_2}}$.



What happens if you type a word instead of your age? Try it.

You get a message saying: **Redo from start** followed by the question mark. Type a number this time after the question mark. It works. The message means you typed the wrong "type" of answer; it was looking for a number, not a word. If you like, rewrite the program to give a different answer. Here's an example:

```
5 CLS
10 PRINT "What is your age?"
20 INPUT A
30 LET B = A + 1
35 PRINT "Next year you will be " B " years old."
40 END
```



Line 30 gives a value to the variable B. When asked to print B, the computer adds 1 to the value of A. (See again that the program works, even though a line is split.) Then, line 35 prints the message.

Run the program.

You can do without the PRINT statement in line 10, because the INPUT statement can also print your message. Edit line 10 to read like this:

10 INPUT "What is your age?";A

Run the program again. Isn't it easier to type one line instead of two? We'll include strings with the INPUT statement wherever possible.

When you're ready to go on, use NEW, then CLS to clear the memory and the screen.

Your Own SOUND Track



You can use the INPUT statement and a variable to make musical tones. Remember that a SOUND statement needs two numbers, one for frequency (high or low note), the other for duration (long or short note). Let's write a program that asks for a frequency number (between 37 and 1000), then plays the tone for about one half-second (duration = 10). If you want to be precise, a second equals 18.

We'll cover each line separately:

Line 5 clears the screen.

5 CLS

Line 10 prints a message asking for a frequency number. We'll use INPUT to print the message.

10 INPUT "Give a number between 37 and 1000 "; N

This prints the message and then a question mark. (You may use any numeric variable, of course. We've used N for note.)

Line 20 has the SOUND statement followed by the variable N and the small duration number, 10.

20 SOUND N, 10

Type lines 5 through 20 and RUN the program.

Now add a GOTO statement at line 30 to make the program repeat from line 5. RUN the program for as long as you care to. Then press Fn-Break to jump out of the loop.

Can you add a second INPUT statement to ask for a duration number between 1 and 50? Of course you can! Try it yourself before going any further.

Here's one way to add a second INPUT statement. Add line 15 below and edit line 20 as shown.

```
15 INPUT "Now another number between 1 and 50."; D
20 SOUND N, D
30 GOTO 5
```

You can make the screen easier to read by putting a blank line or two after line 10. A PRINT statement with nothing after it gives you a blank line. Insert line 12 with an empty PRINT statement in it.

If you like to experiment, try asking for two or three tones in a row before playing them with SOUND statements.

Quick Review



- PRINT allows you to do arithmetic by writing statements without quotation marks.
- *Numeric variables* are letter symbols that stand for numbers.
- INPUT asks for information to continue running a program.
- SOUND requires a frequency and duration number.

Your Turn

Project #1: Write a program that asks for three numbers, then prints the total. Use one INPUT statement for each number you ask for, and a different numeric variable for each.

Project #2: Write a program that asks for the total grocery bill for each of four weeks. Then have the computer calculate the total grocery expenses for a month.

Project #3: Write a program that fills the screen with numbers from 1 to 1000 and beyond. (Set the value of your variable to 1, then increase it by 1. Use a GOTO statement to repeat and a semicolon to fill the screen.)

Solutions

Project #1

```
5 REM Project #1
10 CLS
20 INPUT "Give me the first number ";A
30 INPUT "Give me the second number ";B
40 INPUT "Give me the third number ";C
50 PRINT A "+" B "+" C "=" A+B+C
60 END
```

Project #2

```
1 REM Monthly Grocery Expenses
10 CLS
20 INPUT "Grocery expenses: first week ";A
30 INPUT "Grocery expenses: second week ";B
40 INPUT "Grocery expenses: third week ";C
50 INPUT "Grocery expenses: fourth week ";D
60 PRINT "Total for the month =" A+B+C+D "dollars"
70 END
```

Project #3

```
1 REM Columns of Numbers
5 CLS
10 A = 1
20 PRINT A;
30 A = A + 1
40 GOTO 20
50 END
```

Remember to use Fn-Break to stop the program.

Lesson 4. A Variety of Variables



You have seen how easy numeric variables are to use. Now let's have fun with another kind of variable, the *string variable*. We'll measure the string variable, chop it up, and put it together again. We also will learn a way to "loop" through a program with the FOR and NEXT statements.

In Lesson 4, you will learn to:

- Use string variables in a program to make it easy to print strings without typing them.
- Use the LEN, LEFT\$, RIGHT\$, and MID\$ functions to play with strings.

- Use the READ and DATA statements with any set of variables.
- Use the FOR and NEXT statements to make program loop back to run a certain part of itself again.

String Variables

Okay, let's get one thing straight. Anyone who still thinks we are talking about a ball of string or a yoyo gets to sit on a cactus patch until dawn. No, no. A *string* is a group of letters, or numbers, or keyboard symbols that are always enclosed in quotation marks.



Here is a short program with *string variables*. You'll know a string variable by the dollar sign at the end. (The LET statement is optional, so we left it out this time.)

Use the NEW command. Then type:

```
5 CLS
10 A$ = "Sunday "
20 B$ = "Monday "
30 C$ = "Tuesday "
40 D$ = "Wednesday "
50 E$ = "Thursday "
60 F$ = "Friday "
70 G$ = "Saturday "
80 PRINT A$, B$, C$, D$, E$, F$, G$
```

We put each word in quotes so the computer will print each word as it is. Each word, like "Sunday," is a string value. Now RUN the program and see how the words are printed in 2 columns (5 columns on an 80-column screen).

Experiment a little. Replace the commas in line 80 with semicolons and RUN the program.

String variables can stand for more than one word, too, as in the next program. Type NEW, then press Enter to clear the old program from the computer's memory.

Enter this new program:

```
5 CLS
10 ABC$ = "What did the dirt say when it rained?"
20 XYZ$ = "If this keeps up, my name will be mud."
30 PRINT ABC$, XYZ$
```

RUN the program. Do you see how you can assign entire sentences to string variables?

Also, did you notice the string variables are three letters long. You can make the variable names as descriptive as you want. You could replace ABC\$ and XYZ\$ with different string variables like JOKE\$ and PUNCHLINE\$, or whatever word with a "\$" at the end. However, avoid using any of the *reserved words* listed in Appendix C of this book.


Use the NEW command, then enter this program:

5 CLS 10 INPUT "What is your name, Oh honorable one ";N\$ 20 PRINT "Aloha, most honorable " N\$ 30 END

RUN this program for yourself before trying it out on your friends. Make sure there is a space between the "e" in "honorable" in line 20 and the quotation mark.

If you like, add some more INPUT statements to ask for a person's favorite color, or best birthday present, or the scariest thing.

Four String Functions (or: A String Quartet)

Now that you know a little about strings, let's look at some fun things to do with them, namely, chop them up.

First of all, there is something that you may have figured out for yourself: strings can be joined by using a plus sign.

Use the NEW command, then enter this example without a line number:

PRINT "The " + "Mad " + "Hatter"

That's easy enough. (We left a space after each word and before the quotation mark so the words wouldn't be squeezed together.) Now for something a little trickier.

The LEN Function

You can have the computer count the characters in a string by using the LEN function. (LEN is short for "length.")

Use the CLS command, type this program, and RUN it:

```
10 W$ = "alligator"
20 PRINT LEN(W$)
```



Let's turn this into a game. Let's write a program that asks for a string then lets the computer count the letters, numbers, and spaces in the string. Use NEW and CLS, then enter this program:

```
10 PRINT "You type in any string and"
20 PRINT "I'll tell you how long it is."
30 PRINT
40 INPUT A$
50 S = LEN(A$)
60 PRINT "Your string is "; A$
70 PRINT "and has ";S;"characters in it."
80 END
```

Make sure there is a blank space after "is" in line 60. RUN the program and type in your first name. If you type your first and last name, the space between the two names counts as a character, too.

Try it out on a friend. Change the wording, maybe. Make line 80 say **GOTO 10**, if you like, but remember to use Fn-Break when you're finished.

The LEFT\$ Function

Obviously, the computer can spot each character in a string. If so, it could probably separate them as well. Hmmm! Is that really possible?

Use the NEW command, then enter and RUN this program:

```
10 W$ = "alligator"
20 PRINT LEFT$(W$,4)
```

Aha! To snip off the left portion of a string, we use LEFT\$. (W\$,4) tells the computer to take the string called W\$, then snip off and print the first four characters from the left ("alli"). The rest of the string vanishes.

Try using LEFT\$ to break up big, difficult words into their parts. Or use this long word: Turtlesareslowbutsure.

The RIGHT\$ Function

As you might expect, RIGHT\$ allows you to snip and print any right-hand portion of a string. Here's how you do it:

```
10 W$ = "alligator"
20 PRINT RIGHT$(W$,5)
```

The computer picks off the five rightmost letters ("gator") and prints them. We say (in our fancy computer way) that the substring "gator" was printed.

Once again, try using RIGHT\$ on a long word or a row of numbers or the 26 letters of the alphabet.

Put some zip into your program by using FOR and NEXT to snip off one or two letters at a time. Here's an example:

Use the NEW command, then enter and RUN this program:

```
10 W$ = "12345678901234567890"
20 FOR K = 1 TO 20
```

Line 20 tells the computer to do something 20 times, starting from one. Do what? We'll show you an example of how you can snip off letters from both ends of the word at the same time.

First, let's assign a string variable to the "number" string, W\$, and a numeric variable to the "snipping" number, K, then assign the whole thing to two variable names, L\$ and R\$.

```
30 L$ = LEFT$(W$,K) : R$ = RIGHT$(W$,K)
```

Now it's time to add a PRINT and SOUND statement:

```
40 PRINT
50 PRINT K;L$
60 PRINT K;R$
70 SOUND 440,1: SOUND 520,1
80
```

Question: What should line 80 be?

Answer: 80 NEXT K

*

RUN the program. Can you see from the last few lines on the screen how the first line adds a number on the right, and the second line adds a number on the left, until they both have the same last number. And, for those of you who like starting from the middle, BASIC proudly presents: MID\$.

The MID\$ Function

MID\$ lets you snip and print the middle or inside portions of a string. The MID\$ function tells the computer three things: which string, where to start snipping, and how many characters to snip.

Use the NEW and CLS commands, then enter and RUN this program:

```
10 W$ = "alligator"
20 PRINT MID$(W$,4,3)
```

The letters "iga" are printed (cute name for a space creature, huh?). The computer starts at the fourth character and snips three characters.

There! You are now an expert in computer karate! Use it well!

READ and DATA Statements

To continue with variables, sometimes you may want the computer to read many items from a list and print them in order. You can do this by assigning a list of items to a variable.

The READ and DATA statements help the computer read items from a list, one at a time. Let's look at an easy example.

We're going to ask the computer to read the numbers 100, 200, and 300 from a data list and then print them.

Now, since these are numbers, we must use numeric variables (not string variables). Type NEW and enter this program:

5 CLS 10 READ H 20 PRINT H 30 GOTO 10 40 DATA 100, 200, 300

Line 10 tells the computer to look for a number in line 40 because READ looks for DATA; they're a couple. Line 20 says: "Print that number." At line 30, the computer begins repeating this operation and reads the next number.

RUN this program. Once the three numbers are printed in order, the computer tells you it is **Out of DATA in 10**. READ wanted more numbers from DATA but couldn't get them. Got it?

We can assign a variable to each of the three numbers and add them up. In that case, we won't need a GOTO statement.

Use the NEW command, then enter this program:

```
5 CLS
10 READ H, I, J
20 PRINT H + I + J
30 DATA 100, 200, 300
```

Run it. Now, let's use string variables with the READ and DATA statements. Clear your program from memory and enter this one:

5 CLS 10 READ G\$ 20 PRINT G\$ 30 GOTO 10 40 DATA Papa Bear, Mama Bear, Baby Bear





Run it. First, the computer assigns the value "Papa Bear" to the variable G\$. The second time, it assigns "Mama Bear" to G\$, the third time, "Baby Bear."

You can place the DATA statement anywhere in the program. The important thing to remember is that your items will be printed in order. If you write more than one DATA statement, the computer will begin with the lowest line number, line 30 before line 40 for example. Inside each DATA statement, the computer will go from left to right.

Note that the three strings in line 40 do not have quotation marks. You do not need quotation marks around strings in a DATA statement, unless a string includes a comma, semicolon, or a leading or trailing space.

Amuse yourself by dreaming up interesting uses for what you have learned so far. We'll get you started with this little show business gag.

Use the NEW command, then enter this program:

```
5 CLS

10 INPUT "What is your name"; N$

15 PRINT

20 PRINT "PRESENTING..."

30 SOUND 440,1:SOUND 520,1:SOUND 560,1

40 PRINT

50 PRINT "THE COMPUTER CAPERS OF"

60 SOUND 440,1:SOUND 880,1:SOUND 780,2

70 PRINT

80 PRINT "THE INCREDIBLE..."

90 PRINT

100 PRINT N$;

110 SOUND 44,1:SOUND 520,1:SOUND 560,1
```

Run this program for a chuckle. If you decide to add line 120 to say **GOTO 100**, remember to use Fn-Break to jump out.



FOR and NEXT Statements

While we are having fun with variables, let's quickly look at another pair of statements that make use of numeric variables. The FOR and NEXT statements tell the computer to follow commands while counting through a series of numbers.

Here's a simple example. We'll make the computer print numbers from 1 to 10 by using the FOR and NEXT statements.

Use the NEW command, then enter and run this program.

```
5 CLS
10 FOR N = 1 TO 10
20 PRINT N
30 NEXT N
```

Run it. In line 10, N is a numeric variable that stands for a series of numbers from 1 through 10. The first time through, the computer reaches line 20 and prints 1. At line 30, the computer gives N the next higher number, then goes back to line 20 and prints 2. This is repeated until the computer counts past 10. Then it stops printing.

You can use a FOR/NEXT loop to slow down or speed up a part of any program. For example, see how slowly the following program puts a word on the screen.

Use the NEW command, then enter and run this program:

```
5 CLS
10 INPUT "Where were you born" ; W$
20 PRINT W$;
30 FOR Y = 1 TO 2500
40 NEXT Y
50 GOTO 20
```

Remember to stop the loop by Fn-Break.

To get a better idea of what we mean about slowing down or speeding up the timing of a program, speed up the program by changing 2500 to 250, then to 25, and run the program. Try it.

Get the idea? Good! We'll say more about the FOR/NEXT loop in the next chapter. For now, think about it as a *delay tactic* in a program. Try it out with sound effects and see what it does.



OK. Hold it. We've seen a lot of these functions. Before we end the chapter, a word about what a function is.

1. A function (like RIGHT\$) is not a command (like PRINT). It cannot stand alone or begin a statement.

10 PRINT RIGHT\$(W\$,3)is correct.10 RIGHT\$(W\$,3)is not correct.

2. A function can hold a value, like a variable does. This value can be put in the computer's memory and printed later.

LEN(W\$) is a function. It tells the computer to count and remember the length of a string called W\$.

3. A function is immediately followed by parentheses (). Inside these () are "mini-instructions" for the computer. For example:

LEN function has one instruction:	LEN(W\$)
LEFT\$ function has two:	LEFT\$(W\$,4)
MID\$ function has three:	MID\$(W\$,1,5)

As you can see, the mini-instructions in parentheses () can be either strings or numbers, or both.

Your Turn

- . e

-

-

· 6

.....

~

÷

~

- _ .

Project #1: Write a program that asks for the name of a month, asks how many letters to abbreviate in the name of the month, then prints a message saying how many letters the month will be abbreviated, the name of the month to be abbreviated, and the abbreviation of the month.

Project #2: When you finish this book, you may want to throw a grand party. Use the READ and DATA statements to print a short list of your guests.

Project #3: Play any series of musical tones, using the READ and DATA statements. Print a title or message for each tone. Use the FOR/NEXT statements to make a delay loop.

Solutions

Project #1

```
1 REM Abbreviating the month
5 CLS
10 INPUT "Type the name of a month.";M$
20 PRINT
30 INPUT "How many abbreviated letters";N
40 PRINT
50 L$ = LEFT$(M$,N)
60 PRINT "The ";N;"-letter abbreviation"
70 PRINT "of ";M$; "is ";L$
80 END
```

Project #2

```
1 REM Birthday Party Guests
5 CLS
10 PRINT "My party guests will be:"
20 FOR I=1 TO 5
30 READ G$
40 PRINT G$
50 NEXT I
60 DATA Barb, Heidi, Marge, Donna, Teresa
70 END
```

Project #3

```
1 REM Musical Tones
5 CLS
10 READ T,D,N$
20 PRINT N$
30 SOUND T,D
40 FOR I=1 TO 1000
50 NEXT I
60 GOTO 10
70 DATA 200,10,Low,1400,10,High,400,20,Long,900,5,Short
80 END
```

An **Out of data** message occurs at the end of the Project #2 example. See if you can figure out how to add a FOR NEXT loop to the program to eliminate the **Out of data** message.

. . e

. .

- -

~

~~

· ·

-

- -

e - 21

~ . .

c ...

-. **-**

e .

Here's what we covered in Lessons 3 and 4:

- Without quotation marks, the PRINT statement allows you to calculate with numbers.
- For the computer,
 - + means add numbers.
 - means subtract.
 - * means multiply.
 - / means divide.
- Numeric variables stand for numbers. For example, X = 10. The equals sign means "in place of."
- String variables stand for words (or symbols or numbers) that you put between quotation marks.
 For example, A\$ = "Wizard of Oz."
- The LEN measures the length of a string. LEFT\$, RIGHT\$, and MID\$ functions chop off different parts of a string.
- How to distinguish functions from commands and statements by what they do in a program.
- The INPUT statement tells the computer to ask for information and to store the information in a numeric or string variable.
- The READ and DATA statements tell the computer to read items from a list. The READ statement must come before the PRINT statement line. The DATA statement can be anywhere in the program.

Notes:

e,

E,

N,

a,

N,

<u>, en e</u>

q

ę

2

R,

a,

Ţ

Chapter 3: Branches and Loops

About This Chapter

Nobody can call you a mere beginner on the computer any more. You have learned quite a few commands and statements in BASIC as well as some practical and amusing ways in which to apply them. By now, you are probably breathing like a hungry dragon, saying: "More, more, I want to learn more!" So we'll step aside and let you go roaring off into Chapter 3.



There are two lessons in this chapter.

In Lesson 5, you learn to:

- Print messages in different parts of the screen.
- Use the FOR and NEXT statements to count and to repeat instructions to the computer.
- Mix sound effects with background colors by using the COLOR statement.

In Lesson 6, you learn to:

- Use the IF and THEN statements to help the computer choose between two alternatives.
- Use different symbols to set clear conditions for such a choice.
- Make smaller loops in your programs by using the GOSUB and RETURN statements.

Error Messages

Here is a list of possible error messages you may encounter in Chapter 3. If you get one, check through the program again. If that doesn't work, just type the program over again.

• FOR without NEXT or NEXT without FOR

You typed a FOR statement without a matching NEXT statement, or vice versa. LIST and correct your program before you rerun it.

• Illegal function call

The computer cannot do what you ask. LIST your program and check line numbers, commands, and statements. Correct the program, then rerun it.

• Missing operand

÷.6

ب

• •

^

^

-

د به

æ .e

er •

e - .-

e . .

• ~

÷ ,

æ.,•

- -

–

Your program is missing an instruction. LIST and correct the program before you rerun it.

• Syntax error

Correct the spelling and punctuation in your program, then rerun it.

• Undefined line number

Check the line numbers in your program, then rerun it.

Notes:

-

2

P,

Lesson 5. The Great Screen Escapade



Printing Anywhere On the Screen

So far, everytime you've used a PRINT statement, the computer has been printing things on the left side of your screen. That gets a bit boring after awhile, don't you agree? Well, how would you like to print something in the center of your 40-column screen?

Use NEW, then enter this short program:

5 CLS 10 LOCATE 12, 16 20 PRINT "Bullseye" Now RUN the program. Aha, it worked! As you may have guessed, the secret is in line 10; we located the word "bullseye" at row 12 and column 16. Picture the screen with 25 rows running left and right across the screen like benches in a church, and 40 columns running top to bottom like cans stacked on supermarket shelves.

The computer can pick a spot anywhere on the screen "grid" if you tell it which row and column you want. You need two numbers to describe any position on the screen. For example:

Position 1, 1 is the top left-hand corner of the screen. This is also called the *home* position.

Position 25, 40 is the bottom right-hand corner of the screen.

Question: Can you guess the numbers for the bottom left-hand and top right-hand corners of the screen?

Answer: 25, 1 and 1, 40



Note: The lowest line on the screen you can use is line 24, because the computer needs line 25 for displaying the function keys that we see when we start IBM PC*jr* BASIC.

Now, to get back to our little program. The LOCATE statement in line 10 told the computer to start printing at position (12, 16) which is roughly the mid-point on the 40-column screen for an eight letter word.

Use the LOCATE statement to print your name or a number in different positions. If you use a long string or many numbers, choose your starting position carefully, or there won't be enough room. (Hint: Divide the number of letters in the word by two and subtract that number from 20.)

Be a Math Whiz with FOR and NEXT

In Lesson 4, you used FOR and NEXT to delay a part of your program. If you don't remember how, go back and refresh your memory. The FOR and NEXT statements make the computer count through a series of numbers. (You would have to use a PRINT statement, for example, to print the numbers.) Here are some correct examples:

FOR B = 1 TO 1000 : NEXT B FOR X = 13 TO 73 : NEXT X FOR W = 2.4 TO 7.8 : NEXT W

As you can see, it is also correct to pair FOR and NEXT on the same line with a colon in between. This can save you space.

Just for review, let's make the computer print CHEESE four times.

Use the NEW command, then run this program:

```
5 CLS
10 FOR K = 1 TO 4
20 PRINT "CHEESE"
30 NEXT K
40 END
```



Very good.

Now watch this. You say you want to see the multiplication table for the number 12? All right, here goes!

Use the NEW command, then enter this program:

```
5 CLS
10 FOR M = 1 TO 10
20 PRINT "Twelve times" M "=" 12 * M
30 NEXT M
```

The computer prints line 20 ten times with a different answer and then stops. You might want to use your own numbers to get a little practice. So far, the computer has been counting by ones. By adding a STEP command after the FOR statement, you can make it count differently. For example, STEP 2 makes it count by twos, STEP 3 by threes, and so on.

Here's an example:

```
5 CLS
10 FOR D = 2 TO 24 STEP 2
20 PRINT D
30 NEXT D
```

Of course, the computer can count backwards as well. STEP -2 makes it count down by twos, STEP -3 by threes, and so on.



Doesn't this make all sorts of ideas pop up in your head? If it doesn't, it will in a moment. First, change line 10 in the previous program to line 10 below and run it:

10 FOR D = 24 TO 2 STEP -2

So much for that. We now can do something interesting with the STEP command. For example, you can make the computer slide up a range of tones and then down again. Like an elevator, perhaps?

Use NEW and CLS, then enter this program:

```
5 REM "Elevator Ride"

10 FOR U = 50 TO 500 STEP 5

20 PRINT "Going up..."

30 SOUND U, 1

40 NEXT U

50 FOR D = 500 TO 50 STEP -5

60 PRINT "Coming down..."

70 SOUND D, 1

80 NEXT D

90 END
```



After you RUN this program once or twice, change the STEP numbers to suit your fancy.

A Spot of Color



We can't wait to show you how your computer handles colors. For now, just as a taste, let's use the COLOR statement to liven up your changing backgrounds.

Use the NEW command, then enter the following statement:

SCREEN 1,0

When you press Enter, your computer gets into the *graphics mode*. (Before this happened, it was in *text*

mode.) You will learn more about the graphics mode in another chapter. All you need to know for now is that you can use the COLOR statement to select one of 16 background colors.

Here is a list of these colors:

0	Black	8	Gray
1	Blue	9	Light Blue
2	Green	10	Light Green
3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Brown	14	Yellow
7	White	15	Bright White

Here is a COLOR statement that changes the background to blue (remember that colors vary depending on your display device). Enter the statement:

COLOR 1

1

0

Try this:

COLOR 4

Amuse yourself for awhile by switching between background colors.



Now go back into text mode by typing the following:

SCREEN 0,1

In text mode you have a choice of colors. For your background, you can select colors 0 to 7. For your foreground, you can select from among all 16 colors, and the same holds true for the border around the screen.

The following statement:

5 CLS 10 COLOR 14, 1, 4

produces a yellow foreground (the letters you type), a blue background (area surrounding the letters), and a red border around the screen. Try it.



Fun, huh? Looks like you're right on target.

Your Turn

Project #1: Write a program that asks for a tone frequency and a background color from 0 to 7. Then have the computer sound the tone against the color background that is selected. You can include a question for a foreground color as well.

Project #2: Write the multiplication table for the number 8, from 8 times 1, to 8 times 10.

Solutions

```
1 REM Colorful Music

10 SCREEN 1

20 CLS

30 INPUT "Enter tone frequency ";F

40 INPUT "Enter background color (0 to 7) ";B

45 INPUT "Enter foreground color (0 to 15) ";C

50 SOUND F, 18

60 COLOR C,B

70 END
```

Project #2

```
5 CLS
10 FOR M = 1 TO 10
20 PRINT "Eight times" M "=" 8*M
30 NEXT M
```

Lesson 6. The Computer Decides

A Fork in the Road



In handling information, you may want the computer to take one of two paths. The computer follows the signposts in your program. You help the computer take the right path by using the IF/THEN statement in your programs. Each IF/THEN statement acts like a signpost that tells the computer which path to take.

Here's an example of two paths in a program. Use the NEW command, then enter this program:

94

```
5 REM Good Taste in Food
10 INPUT "Do you like peas (y or n)"; A$
20 IF A$ = "y" THEN PRINT "So do I."
30 IF A$ = "n" THEN PRINT "Me neither."
```

Run this program.

1

A

0

A

0

Do you see what is happening in lines 20 and 30? The computer checks your answer.

If you answer "y" for yes, it prints one message; if "n" for no, then it prints a different message.

After line 10, the program "branches out" in one of two paths.



Setting A Clear Path

To help the computer take the right path, you must set down clear conditions. In BASIC, we set clear conditions by using certain signs (like the = sign you just used).

The signs that the computer understands are:

MEANING	EXAMPLE
Equal to	A = YES ;X = 10
Less than	X < 2
Greater than	Y > 5
Not equal to	B\$ <> "YES"
Less than or equal to	X <= 25
Greater than or	
equal to	X >= 100
	MEANING Equal to Less than Greater than Not equal to Less than or equal to Greater than or equal to

Let's write a simple program that uses some of these signs.

We'll ask the computer to print a series of numbers from 1 to 5 and then stop. Instead of using FOR and NEXT, we'll use IF/THEN.

Use the NEW command, then enter this program. Don't RUN it until you finish entering line 60.

```
5 CLS
10 N = 1
20 PRINT N
30 N = N + 1
```

Got it so far? Line 30 increases the value in variable N by one. We now need a line that tells the computer to stop at 5. Hmmm! Let's try the sign that means greater than or equal to (>=).

40 IF N >= 6 THEN 60
When counting goes past five, the computer stops printing because N, which becomes 6, would be equal to 6. We then want the computer to go to line 60 (which we haven't written yet).

Meanwhile, we need a GOTO statement before line 60 (whatever that will be).

50 GOTO 20

8

Ô

000000

m

Aha! The computer will print 1, 2, 3, 4, 5, and then make a mad dash for line 60. What should we say in line 60? How about:

60 PRINT "Counting Made Easy, Too!"

There now. RUN the program and see if it works. Try another set of numbers. Change "6" in line 40 to "66" or "366" and watch the numbers on the screen.

Better still, change the program to make the computer count backwards from a high number to a low one. Then use the $\leq =$ sign (less than or equal to) in the IF statement.

Is the hungry dragon panting for more? Good. Here goes.



Let's introduce another famous boomerang couple, GOSUB and RETURN! When you want a program to branch to a particular line, do a few things there, and then boomerang back, you use GOSUB and RETURN. GOSUB means "Go to the Subroutine and do a few things there, then come back" to where you branched off and do more things. You might say it's like a detour.

In other words, GOTO has no way to directly return you where you left from. With GOTO, if line 100 says to go back to line 10, line 20 through 90 must be run again. With GOSUB and RETURN, if line 100 says GOSUB to 10, line 20 can say RETURN and the program will continue to run after line 100.

Let's type an example using a well-known camp-fire song to see what we mean.

Use NEW, then enter the following lines:

1

Ó

h

D

Ū

n

```
1 REM First Verse

5 CLS

10 PRINT "Old MacDonald had a farm"

15 GOSUB 65

20 PRINT "And on his farm he had some ducks"

25 GOSUB 65

30 PRINT "With a quack, quack here"

35 PRINT "And a quack, quack there"

40 PRINT "Here a quack, there a quack"

45 PRINT "Everywhere a quack, quack."

50 PRINT "Old MacDonald had a farm"

55 GOSUB 65

60 END

65 PRINT "Ee-ay-ee-ay-oh."

70 RETURN
```



RUN the program, then read our explanation below and RUN it again.

How the Subroutine Works

Follow along carefully. At line 15, the program skips to line 65, prints the refrain, then goes back to line 20. At line 25, the same thing happens. The program returns to line 30 and goes all the way down to line 55. Then it branches again to 65 and returns to line 60 where it ends. The main program is from lines 5 to 60. The little subroutine is in lines 65 and 70.

Subroutines are very important and useful to people who write programs. Among many other uses, they are often used to hold math formulas (when repeated calculations have to be done). You also can use a GOSUB to produce a delay or produce effects with sound and colors in a repeating pattern.

Probably the biggest advantage to using a GOSUB and RETURN is that you may save time and some program space. Writing the fewest amount of program lines, which therefore saves time, should be your objective when you write your program. Writing the fewest number of lines helps you in two ways: one, the program runs faster because the computer does not have to read so many lines; and two, you need less memory to run your program which you may need if you write a large program. Enough of this "loopy" lesson. After completing the Projects and reading the Chapter Review, you will begin the most entertaining chapter in the book, Chapter Four. Chapter Four, which is also the last chapter, has the longest but the most fun examples. You'll draw and paint circles and boxes, and compose songs with harmonies and countermelodies. You may even want to combine the two.



Your Turn

Project #1: Write a program that asks for a number, then plays a musical tone. Keep the range of numbers between 50 and 1500. Also try adding a request for a number that specifies the duration of the tone.

Project #2 (Advanced project): Write a program in which Player #1 is asked to give a letter of the alphabet and Player #2 must guess the letter. Include a message to say "Too high, try again" and "Too low, try again." If the letter is guessed correctly, print a message saying "Hooray! You did it!" followed by a musical note.

Solution

```
1 REM Number Game
10 CLS
20 INPUT "Enter a number between 50 and 1500 ";N
30 INPUT "Enter the duration of the tone ";D
40 SOUND N,D
50 END
```

Project #1

```
5 CLS
10 PRINT "Player #1, type any letter between a and z."
20 PRINT
30 INPUT "Your letter";L$
40 CLS
50 INPUT "Player #2, what's your guess";G$
60 IF G$ > L$ THEN PRINT "Too high, try again"
70 IF G$ < L$ THEN PRINT "Too low, try again"
80 IF G$ = L$ THEN GOTO 100
90 GOTO 50
100 PRINT
110 PRINT "Hooray! You did it!"
120 FOR M=50 TO 1000 STEP 20
130 SOUND M,1
140 NEXT M
150 END
```

Chapter Review

Quick, before you call your friends about your exciting discoveries, think awhile about what you learned in Chapter 3. You learned how to:

- Use the LOCATE statement to print a message in different positions of the screen.
- Use the FOR and NEXT statements to count and to repeat instructions to the computer.
- Change the screen background to one of eight colors by using the COLOR statement in text mode.
- Use the IF and THEN statements to help the computer choose between alternatives.
- Make smaller loops in your programs by using the GOSUB and RETURN statements.

About This Chapter

Chapter 4 has something for everyone. It shows how you can write and play songs on the computer. It gives you a chance to do some drawing on the screen. And, of course, you'll play some games with a new twist.

In Lesson 7, you will learn how to:

• Write and play a simple, well-known melody on the computer.

Note: Lesson 7 does not try to teach the principles of music, only a few of the ways to produce computer-generated music.

In Lesson 8, you will learn how to:

• Draw figures on the screen in the graphics mode.

Error Messages

The error messages you may see in Chapter 4 are the same as those we listed at the start of Chapter 3. If you see an unexpected error message, refer to Appendix A in the *BASIC* Reference book.

Lesson 7. Music



This lesson is a reward for your hard work. You are going to be a star in your own show. (The computer will be your secret partner, or your producer, if you like.) You will play sweet music melodies that charm the crowds and bring the house down with wild applause.

You will use the sensational PLAY statement to write and perform a few easy tunes. (Something some of you never dreamed you would do, right?)

Play Maestro Play



You say you don't know the first thing about music? Well, read on and you might just amaze yourself.

Music is made of notes. There are seven notes, from A to G, no more. The computer knows these notes. Let's test it:

PLAY "C D E F G A B"

After typing the line, press the Enter key to hear the seven notes. Run it a few times. Sounds good, huh?

The sounds go up like rungs in a ladder, a little higher each time.

On the piano keyboard, these notes look like this:



You see seven white keys and five black keys. Going from left to right, the black keys stand for: C sharp, D sharp, F sharp, G sharp, and A sharp. Going from right to left, these same black keys stand for: B flat, A flat, G flat, E flat, and D flat. Simple? Maybe. Let's go on.

On the computer, the sign for sharp is either # or +. The sign for flat is a minus sign (-). From left to right, let's now play the notes associated with the white and black keys. Look at our keyboard picture while the program runs.

PLAY "C C# D D# E F F# G G# A A# B"

Run it. Now play the same notes in the same order, but use flat signs instead. Type:

One more thing. A note can be timed to the beat you want. The first line of the song "Yankee Doodle" goes like this. Type the line and RUN it:

PLAY "C C D E C E D"

Something wrong? Make the last note D2 and run it again.

Aha! D2 played twice as long as D. Hmmmm!

Just out of curiosity, try these timings for the following notes and compare how they sound. Let's forget about "Yankee Doodle" for now. We'll come back to it.

10 PLAY "C2 D2 E2 F2 G2" 20 PLAY "C1 D1 E1 F1 G1" 30 PLAY "C8 D8 E8 F8 G8" 40 PLAY "C16 D16 E16 F16 G16"

Run the musical program a couple of times and try to pick out the different notes and their timing.

Now add these lines:

```
5 PLAY "C4 D4 E4 F4 G4"
6 PLAY "C D E F G"
```

Lines 5 and 6 sound exactly the same. The computer plays any note without a number to the beat or "timing" of 4, unless you put in a different number. C1 is twice as long as C2, and C2 is twice as long as C4. C4 is twice as long as C8, and so on. Numbering of notes will make more sense as you experiment with it. Let's try one. Before we can listen to the music, we have to write the notes and lyrics in a program. We'll practice with another familiar tune.



Use the NEW command, then enter this program:

```
5 REM "Twinkle, twinkle, little star"
10 PLAY "C C G G A A G2"
15 REM "How I wonder what you are."
20 PLAY "F F E E D D C2"
25 REM "Up above the world so high"
30 PLAY "G G F F E E D2"
35 REM "Like a diamond in the sky."
40 PLAY "G G F F E E D2"
45 REM "Twinkle, twinkle, little star"
50 PLAY "C C G G A A G2"
55 REM "How I wonder what you are."
60 PLAY "-----"
```

Line 60 is yours to fill. Should be a piece of cake by now.

Answer: Line 60 is the same as line 20.

The curtain opens. Play your first song. RUN the program, close your eyes, and smile from ear to ear. (Eat your heart out, Beethoven!)

You can play it again, uh, Sam! Add GOTO 10 in line 70, if you like. Remember to use Fn-Break to stop the song.

Once Again, With Feeling

n

You can change the "flow" of any computer-generated song on the IBM PC*jr* by adding one of the following after a PLAY statement:

PLAY "ML"	means <i>music legato</i> . ML will sound the notes smoothly, with two breaks between the notes.
PLAY "MS"	means <i>music staccato</i> . MS will sound each note distinctly for three-fourths of its duration. You will hear a slight break between notes.
PLAY "MN"	means <i>music normal</i> . MN will sound each note for seven-eighths of its duration. Unless you specify otherwise, the computer will play notes in <i>music normal</i> .
Add line Q hale	www.to. "Twininglylo_twinglylo_little_stard"

Add line 8 below to "Twinkle, twinkle, little star":

8 PLAY "ML"

Rerun the program and listen carefully for the change.

Then replace "ML" in line 8 with "MS" and run the program. Finally, replace "MS" with "MN" and hear what *music normal* sounds like.

To make listening even more interesting, a piece of music may be played in more than one style. Depending on the melody you're working with, you may wish to switch from "MN" to "ML" and back again, or from "ML" to "MS" and then to "MN" again.

Fast or Slow

You can play a song faster or slower. We call this "varying the tempo." By itself, the computer plays any song at T120, or 120 C4 notes per minute, or 60 C2 notes per minute. You can change the value of T, provided you don't go below 32 or above 255.

We'll repeat this information for clarity's sake.

Standard timing for music is T120. You can change this tempo number to play a song fast or slow. T60 would be half the standard tempo. T240 would be double time.

The slowest tempo on the IBM PC_{jr} is T32 and the fastest is T255.

Add line 8 to your program:

8 PLAY "T60"

RUN the program. The song plays at half the tempo. Try changing T to 150, or 80, or 240 and hear the difference.



High or Low

Finally, to play lower C's or higher B's, you must know how notes are arranged on the piano. The piano keys are divided into seven equal groups called *octaves*. Each *octave* has seven notes, starting with C and ending with a higher B. The leftmost octave on the piano sounds very low indeed, the rightmost octave has those tinkling, high-pitched notes in it. Now, how does the computer know which of the seven *octaves* you mean?

Picture the middle octave on the piano keyboard, the series of seven notes starting with the middle C, on up to the higher B, CDEFGAB. The computer knows this as the octave 3 on the piano, or O3 ("O" stands for octave).



Below O3 (to the left of it on the piano) are three octaves known to the computer as O0, O1, and O2.

Above O3 (to the right of octave 4 on the piano), are three more octaves: O4, O5, and O6. That makes a total of seven octaves, O0 TO O6. Got it?



The computer will play notes in O4 (fifth octave from the left), unless you specify a different octave.

Let's play a C note in four different octaves.

Use NEW and CLS, then enter this without line numbers:

PLAY "03 C 04 C 05 C 02 C"

Let's play a descending sequence of eight notes, from an O4 G down to a lower G (in octave 3).

Use CLS, then enter this program:

```
10 PLAY "G F# E D C"
20 PLAY "03 B A G"
```

Notice when you don't specify the octave, the computer understands O4. Line 20 has notes that go below fourth octave C, so we specify O3. Try out a different series of notes that dip into a lower octave or climb into a higher one. The computer stays in the octave you specify until you change it.

Another way to shift octaves is to use the ">" (greater than) or the "<" (less than) sign. For example, ">C" means you will hear the C in octave 5 (remember the usually-played octave is 4). Similarly, "<C" gives you the C in octave 3. Let's try it. Type NEW, CLS and enter this program:

10 PLAY "<C >C >C" 20 PLAY "03 C 04 C 05 C"

When you run the program, both lines should sound the same notes.

Oops, curtain call. By popular request, "Yankee Doodle."



Type the following song:

n

Ē

10 PLAY "03 C C D E C E D2" 20 PLAY "03 C C D E C2 02 B2"

The last note there was B in octave 2.

30 PLAY "03 C C D E F E D C" 40 PLAY "02 B G A B 03 C2 C2"

If you go any further, be prepared to use dotted notes and the note value of 8. A dot (or period) after a note increases its length by half. For example, $C_{-} = C4 + C8$, making it one and one-half times an ordinary C.

Hmmmm! Sounds dreadfully complicated. Not really. Listen. Add these computer music lines to your 4-line program.

```
50 PLAY "02 A. B8 A G A B 03 C2"
60 PLAY "02 G. A8 G F E2 G2"
70 PLAY "02 A. B8 A G A B 03 C"
80 PLAY "02 A G 03 C 02 B 03 D 03 C2 C2"
90 END
```





There. Keep running the program and you'll get the hang of it.

If you need a blank pause anywhere in a song (musicians call it a "rest"), use P and a time value (1, 2, 4, 8, 16 and so on) wherever needed. By the way, the spaces between notes are not necessary. We have used them to help you learn more easily.

Three-Part Harmony

m

Ē

h

ţ,

Ē

17

ņ

Perhaps the choicest musical feature of your IBM PCjr is that it will serenade you in three-part harmony. In plain language, this means that you can run a program on the IBM PCjr to sound two or three notes together, or up to three melodies that can blend with each other (how well they blend is up to you).

You may not know a great deal about musical harmony, but in a few short minutes you will be amazed at the talents hidden in this cuddly little computer of yours.

The first thing you must put in your program is a statement to turn off the speaker (beeper) in your IBM PC*jr* System Unit and another to turn on the speaker in your TV set. (Harmonies will not play on the System Unit speaker.)

Use CLS, and NEW, then enter these lines:

1 REM Some Classy Chords 5 BEEP OFF: SOUND ON

Line 5 will turn off the system's speaker and begin channeling all the sounds through the TV set. Enter this line:

10 PLAY "<C", "<E"

Now run this three-line program. You hear two notes sounding briefly together. Both notes are in Octave 3. As you can see, the two strings are separated by a comma, but the computer read and performed both together.

Let's add a third note to produce a pleasing chord. But before doing that, let us lengthen the duration of each note by specifying a length of one for each. 8 PLAY "L1", "L1"

Let's add that third note in line 10. Edit line 10 to read:

10 PLAY "<C", "<E", "<G"

Run the program. Oops, that G note is too short. Notice that each string must be defined separately. Edit line 8 to specify "L1" for the third string. While you're at it, specify the octave you want in line 8 instead of line 10 (this allows the notes to be plain to see):

```
8 PLAY "L1<", "L1<", "L1<"
10 PLAY "C", "E", "G"
```

Run the program. Can you hear the combined notes? Run it a few times, then add the following lines to your program:

20	PLAY	"D",	"F+",	, "A"
30	PLAY	"F",	"A",	">C"
40	PLAY	"D",	"G",	"B"
50	PLAY	"E",	"G",	"C"

Run the program a few times. You are hearing a series of chords used frquently in popular jazz music. To improve the effect, make all "first string" notes sound in Octave 1, "second string" notes in Octave 2, and "third string" notes in Octave 3. Edit line 8 as follows:

```
8 PLAY "01 L1", "02 L1", "03 L1"
```

Rerun the program. Experiment some more with chords, if you care to.

Volume and Countermelody

Ē

n n

Ē

n

m

h

n n

If any note sounds too loud or soft for your liking, you might want to reduce or raise its volume. In a PLAY statement, V stands for volume. V can have a value from 0 to 15. Unless you specify differently (like octaves), V is always set at 8 (which is moderate volume). 0 is low volume, 15 is high.

Let's make one last change to our jazzy music. Let's set the volume of the third string at 12. Edit line 8 to look like this:

8 PLAY "01 L1", "02 L1", "03 L1 V 12"

Fascinating though the possibilities are, you need a fair amount of musical knowledge to compose harmonies on the IBM PC_{jr} .

In music, along with beautiful harmonies, you may hear songs in which there is not only a main melody but a secondary melody, called a countermelody. Combining a main melody and countermelody can also create beautiful or interesting music.

To help you see how a countermelody can be added to a song, we offer you this version of "Twinkle, Twinkle Little Star" (lines 30, 40, and 60 appear here as they would if you have a 40-column screen):

```
1 REM "Twinkle, twinkle" in harmony
3 BEEP OFF:SOUND ON
5 PLAY "MN 03", "ML 02 V6", "MS 01 V7"
10 PLAY "CCGGAAG2", "C<G>ECF2E2"
20 PLAY "FFEEDDC2", "D<B>C<AFG8L16AB>C2"
30 PLAY "GGFFEED2", "L8EE<G>EDD<G>DCC<AB
-B2"
40 PLAY "GGFFEED2", "L8<EEL16CDL8EDDL16<
B>CL16<<ABL8>C<82"
50 PLAY "CCGGAAG2", "C<G>ECF2E2"
60 PLAY "FFEED8L16BAGFEDC2", "CCCC<BB>C2
", "AAGGL8GFEDL2C"
```

Run the program. Close your eyes and listen closely for the two melodies. Can you pick out the main melody and the secondary melody? Do you see how the secondary melody was created in the program?

Now you can try your hand at writing other songs you know. Try some easy ones first without countermelodies, like "Row, row, row your boat" or "Three Blind Mice" or "Happy Birthday to You." Then see if you can write a countermelody for them. This is quite a challenge.

If you need inspiration to start, here is a tune that will CHEER you on:

```
5 PLAY "T160"
10 PLAY "02 G8 03 C8 E8 G E8 G2 P2"
20 PLAY "02 G#8 03 C#8 F8 G# F8 G#2 P2"
30 PLAY "02 A8 03 D8 F#8 A F#8 A2 P2"
```



Ē

Ē

Π

Your Turn

Project #1: Write a program that plays a series of notes in a single octave. Use a FOR and NEXT loop to play this series in each of the seven octaves, from O0 to O6.

Project #2: Write a program that asks for a series of six letters, one at a time, between A and G. Then have it ask for an octave number. Then have the computer play a silly tune made of the six notes.

Solutions

Project #1

```
1 REM Sailing the Seven C's
5 CLS
10 FOR I = 0 TO 6
20 PLAY "0 = I; CEFG1"
30 NEXT I
40 END
```

Project #2

```
1 REM Make up a silly tune
5 CLS
10 PRINT "Type a note each time I ask for one."
20 PRINT
30 PRINT "Each note should be between A and G."
40 PRINT
50 PRINT "Then type an octave between 0 and 6."
60 PRINT
70 INPUT "First note ";A$
80 INPUT "First note ";B$
90 INPUT "First note ";C$
100 INPUT "Fourth note ";D$
110 INPUT "Fourth note ";D$
110 INPUT "Fifth note ";F$
130 INPUT "Sixth note ";F$
130 INPUT "Enter octave number (0 to 6) ";0
140 PLAY "0 = 0;" +A$+B$+C$+D$+E$+F$
150 END
```

Lesson 8. Art

The Artist's Touch



Time to change from aspiring musician to creative artist. Just as PLAY was our tool to recreate music, DRAW will be our tool to design colorful pictures.

Do you remember in Lesson 5, we learned how to shift from text mode (SCREEN 0,1) to graphics mode (SCREEN 1,0)? Type SCREEN 1,0 now to get into graphics mode again. Although you can't immediately tell, your screen is like a grid full of tiny squares. The squares are called points. In graphics mode, there are 320 points from left to right (columns), and 200 points from top to bottom (rows). The following lines list the four corner points on the screen by column and row position:

Position 0,0 is at the top left-hand corner.

n n

n n

M

Π

Position 319,0 is at the top right-hand corner.

Position 0,199 is at the bottom left-hand corner.

Position 319,199 is at the bottom right-hand corner.



The starting position for SCREEN 1,0 is in the middle of the screen. It is as though your pencil is in the middle of a piece of paper ready to draw. You can move your "pencil" from the starting position to another position by giving the following directions:

U means "move up"	U50 means "move up 50 points"
R means "move right"	R50 means "move right 50 points"
D means "move down"	D50 means "move down 50 points"
L means "move left"	L50 means "move left 50 points"

These four prefixes, or *substatements*, (U, R, D, L) will let us draw a box on the screen.

Let's try it. Use NEW, then enter this program:

5 CLS 10 B = 50 20 DRAW "U = B; R = B; D = B; L = B;"

RUN the program. (To make the program easier to enter, we used a numeric variable in line 10.) Do you see how the lower left corner of the box is in the middle of the screen. This is the automatic starting position. We'll learn how to change the starting position in a moment, but first let's use the middle of the screen to start.

Change the value of B to make the box smaller or bigger, try 100 and 150.

We can shape this box to look like a diamond (diagonal box) by using the following instructions:

E means "move diagonally up and right"

F means "move diagonally down and right"

G means "move diagonally down and left"

H means "move diagonally up and left"

Let's do it.

Ĩ

n

Use NEW and CLS, then enter this program:

5 CLS 10 D = 50 20 DRAW "E = D; F = D; G = D; H = D;"

RUN it.

Because of the speed of the computer, you can't see how each line is drawn separately. If you could, you would see a line drawn from the center of the screen "diagonally up and right" 50 positions. The next line would be drawn "diagonally down and right" 50 positions to form an up-side-down "V." The two other lines would connect like dots to form the bottom of the diamond.

Once again, change the proportions as you like and rerun the program.

You can also get lines to begin from the *same* starting point by typing the substatement N before each directional substatement. Relax! That's not as tricky as it sounds. An example will help.

Use NEW and CLS, then enter this program which traces a line and returns the cursor to where it began:

5 (CLS	
10	DRAW	"NU40"
20	DRAW	"NE40"
30	DRAW	"NR40"
40	DRAW	"NF40"
50	DRAW	"ND40"
60	DRAW	"NG40"
70	DRAW	"NL40"
80	DRAW	"NH40"



Run it.

Get the idea? Each line originates from the center of the screen instead of from the end of the previous line. Repeat the program if you're still confused. There also is a way to move the starting point from a previous position without drawing a line and that is by using the substatement, B. To see how B works, use NEW and CLS, then enter this program:

10 CLS 20 DRAW "U50" 30 DRAW "BF50" 40 DRAW "D50"

Run it.

n

h

m

h

n n Here's what you just did. Line 20 draws a line 50 positions up from the center as before. Line 30 moves the starting point "diagonally down and right" 50 positions from the previous point, and line 40 draws a line down 50 positions from there.

Where line 30 begins to draw its line still depends on the location of the ended point drawn by line 20.

The next letter substatement we'll learn does not rely on any previous end point or automatic starting point.

To move your "pencil" to any starting point, use the M (stands for move) substatement. Try this one.

Use NEW and CLS, then enter this program:

5 CLS 10 DRAW "BM160,0 M160,199"

Run it. It draws a vertical line down the center of the screen. "BM160,0" moves the starting point to the top center of the screen. "M160" draws the line from the top to the bottom of the screen.

(For more information about the N, B, or M substatements, look them up under "DRAW" in the IBM PC*jr BASIC* book.)

Experiment with these substatements until you feel quite familiar with what they can do for you.

Remember:

N means "move, but return to my starting point"

B means "move, but don't make a line"

M means "go to a new starting point"

The PSET Statement

O.K. What else can we do? Plenty. A handy way to master screen positions is to use the PSET statement; it puts dots on the screen. Let's put a dot in the center of the screen.

Use CLS, then enter without line numbers:

PSET (160, 100)

Run it.

Vary the two numbers (we call them "coordinates"). Place a dot in the corners of the screen and wherever else you wish to see one. You can do detailed things with dots, but they take time to do.

Try using dots to draw buttons with dots for holes, pretend the dots are stars in space, or create a small face with dots for eyes, nose, and a mouth.




The LINE Statement

Let's go one step further with the PSET statement and use it as another way to draw lines.

The IBM PC*jr* can draw a straight line by connecting two dots on the screen made with PSET. Let's place two dots on the screen, one below the other. Use NEW, CLS, and then enter these lines:

5 CLS 10 PSET (160,60) 20 PSET (160,140)

Run the program and look for the dots. Then enter:

```
30 LINE (160,60) - (160,140)
```

Notice that the coordinates in line 30 are the same as those in lines 10 and 20. Run the program and watch the screen. See the line?

Experiment with different sets of coordinates until you become familiar with how the LINE statement works.

After you've tried drawing different geometric shapes, you may want to reward yourself with this star:

```
5 CLS
10 LINE (160,49) - (120,170)
20 LINE (120,170) - (225,99)
30 LINE (225,99) - (95,99)
40 LINE (95,99) - (200,170)
50 LINE (200,170) - (160,49)
60 END
```

The CIRCLE Statement

Circles are probably one of the most fun things to draw. The CIRCLE statement tells the computer where to put the circle and how big it should be.

The CIRCLE statement has two parts:

1. Center of circle (position x, position y)

2. Radius (distance from the center to the edge)

Use CLS, then enter this without line numbers:

CIRCLE (250, 35), 20

Press Enter. This program draws a circle of radius 20 near the upper right corner of the screen.

We'll get back to circles in a moment, but first let's learn more about how to play with color.

In Living Color

Π

n n

Π

Π

T

Î

n n



In Lesson 5, we dabbled with the COLOR statement in graphics mode to change our screen background. If a refresher course is necessary, go back to Lesson 5, "A Spot of Color," and work through that section once more.

To prevent possible confusion, here are notes on the COLOR statement for text mode and graphics mode.

Text Mode (SCREEN 0.1) The COLOR statement in text mode has three parts to COLOR foreground, background, border But before you use the color statement, you must remember to turn on the "color burst" feature by

The following is an example of a COLOR statement in text mode:

COLOR 14.2.1

SCREEN 0.1

including the "1":

it

This sets a yellow foreground, a green background, and a blue border.





Π

n n

Ħ

With a color monitor in text mode, foreground has 16 possible colors (0 - 15), background has 8 possible colors (0 - 7), and border has 16 possible colors (same as foreground).

If you are using a color TV set, the colors you see may be weak or blurred. Try adjusting the color control to improve the picture.

Graphics Mode (SCREEN 1,0)

Both the color television set and the color monitor use graphics mode.

The COLOR statement in graphics mode has two parts:

COLOR background, palette

The background has 16 possible colors (0 - 15).

The palette gives you two selections of three colors each. Let's step through this slowly.

You can select palette 0 or palette 1.

Color	Palette 0	Palette 1
1	Green	Cyan
2	Red	Magenta
3	Brown	White

In both palettes, 0 gives the background color already on the screen.

An example might help. First, use CLS, then turn on the graphics with color burst (different from color burst in text mode):

SCREEN 1,0

Now enter this line without a line number:

COLOR 9, 1

Press Enter. This program sets a light blue background, and selects palette 1 (which, as you can see, has three possible colors). Now what? You need a figure of some sort to apply a color from palette 1.

Very well, let's use what we've learned and draw a circle.

Draw a circle in the center of the screen having a radius of 30. Try this:

CIRCLE (160, 100), 30

Add a number from 1 to 3 for the color you want in palette 1. Magenta? All right, if you insist. Here's the complete statement.

Use CLS, then enter this without line numbers:

CIRCLE (160, 100), 30, 2

If you forget to add the last number, the computer picks number 3 (white in palette 1, brown in palette 0).

Understanding how palettes work is not easy at first. Later, you can come back and look at this section again. For now, let's go on.

The PAINT Statement

m

Ē

Ħ

Ħ

m

Now that we know how to make colorful circles, let's learn how to fill them with color.

You use the PAINT statement to fill a shape or figure with a color from palette 0 or palette 1.

The PAINT statement gives the computer a starting point anywhere inside the area to be "painted." Then it gives the color to be used inside the figure. Finally, it gives the color that serves as the boundary for the painted area.

Here are the parts of the PAINT statement:

- 1. PAINT (position x, position y)
- 2. Inside color
- 3. Figure border color

Use CLS, then enter this example. Don't RUN it until after entering line 30 below:

10 COLOR 1, 1

sets a blue background and selects palette 1.

```
20 DRAW "U20; R20; D20; L20;"
```

draws a rectangle on the screen.

```
30 PAINT (165, 95), 2, 3
```

fills the box with magenta (color 2, palette 1), leaving the edges of the box white (color 3, palette 1).

RUN the program.

Hey, that's a lot more fun than crayons.

Be sure to practice working with these graphics statements. And be nice to yourself if you see something you didn't expect. Surprising yourself can be fun!

Let's try one last example. As a final test of your understanding of graphics, think of what this program will do as you enter each line. (You may also like what you see.)

```
5 CLS

10 CIRCLE (130,59), 55, 1

20 CIRCLE (210,59), 55, 1

30 CIRCLE (170,100), 55, 1

40 PAINT (130,59), 55, 1

50 PAINT (210,59), 55, 1

60 PAINT (170,100), 55, 1

70 PALETTE 1, 1: PALETTE 2, 14: PALETTE 3, 13
```

Were you surprised?



Hooray for You

You deserve a tremendous round of applause for plowing through the material in this book. We hope you're happy for having made the effort.

Are you a programmer now? Well, you're on the way to being one. You have learned to approach tasks on the small computer like a programmer does. You know a lot about the BASIC language on the IBM PC*jr*. And you have sampled the joys of building a program and seeing it work. You're in great shape, friend.

We'll say 'bye for now. We hope your appetite for learning about computers keeps growing. We wish you luck.



Your Turn

) All and a

)

) and the s

,

-

Project #1: Write a program that draws the outline of a table with legs. Then center a triangle on top of the table.

Project #2: Draw five circles (radius 10 each), one in the center of the screen, one in each of the four corners of the screen. Fill the center circle with red, and make its edge green.

Solutions

Project #1

```
5 CLS
10 SCREEN 1,0:CLS
20 DRAW "L30D4R15D30R4U30R40D30R4U30R15U4L47"
30 DRAW "BU1BL8E16F16L31"
```

Project #2

```
10 SCREEN 1,0: CLS: COLOR 8,0
20 CIRCLE (160, 100), 10, 1
30 CIRCLE (30, 9), 10, 1
40 CIRCLE (289, 9), 10, 1
50 CIRCLE (30, 190), 10, 1
60 CIRCLE (289, 190), 10, 1
70 PAINT (160, 100), 2, 1
```

Chapter Review

-

Take a moment to review what you learned in Chapter 4. You learned to:

- Use the PLAY statement to write and perform melodies on the computer.
- Use the DRAW statement to draw simple figures on the screen.
- Use the COLOR statement in graphics mode to change the background, foreground, and border colors.
- Use the PSET statement to put a point in any position on the screen.
- Use the CIRCLE statement to draw a circle of any radius.
- Use the PAINT statement to fill a shape or figure with a color from palette 0 or palette 1.

Notes:

. . .

. .

-

....

-

-

.

- -

4

. ...

Appendix A. Messages

When the computer finds something going wrong, it flashes an error message on the screen. You have already seen many instances of error messages explained in this book. This appendix lists a few more error messages. For a complete list of errors and their explanation, refer to the *BASIC* reference manual.

Number Message

73 Advanced Feature

Your program used an Advanced BASIC feature while you were using Disk BASIC.

Start Advanced BASIC and rerun your program.

17 Can't continue

-

-

You tried to use CONT to continue a program that:

- Halted because of an error
- Was changed during a break in running it
- Is not there

Check your program, and use RUN to run it.

71 Disk not Ready

The diskette drive door is open or a diskette is not in the drive.

Place the correct diskette in the drive and continue the program.

70 Disk Write Protect

You tried to write to a diskette that is write-protected.

Make sure you are using the right diskette. If so, remove the write protection, then retry the operation.

This error may also occur because of a hardware failure.

11 Division by zero

In an expression, you tried to divide by zero, or you tried to raise zero to a negative power.

It is not necessary to fix this condition, because the program continues running.

26 FOR without NEXT

A FOR was encountered without a matching NEXT. That is, a FOR loop was active when the physical end of the program was reached.

Correct the program so it includes a NEXT statement.

12 Illegal direct

You tried to enter a statement in direct mode (without line numbers) that the computer does not allow.

The statement should be entered as part of a program line.

5 Illegal function call

The computer cannot do what you asked. Check your program and line numbers before you rerun it.

22 Missing operand

An expression contains an operator, such as * or OR, with no operand following it.

Make sure you include all the required operands in the expression.

1 NEXT without FOR

The NEXT statement doesn't have a corresponding FOR statement. A variable in the NEXT statement may not match any previous FOR statement variable.

Fix the program so the NEXT has a matching FOR.

4 Out of data

A READ statement is trying to read more data than is in the DATA statements.

Correct the program so that there are enough constants in the DATA statements for all the READ statements in the program.

7 Out of memory

A program is too large, has too many FOR loops or GOSUBs, too many variables, expressions that are too complicated, or complex painting.

You may want to clear the screen with CLS at the beginning of your program to set aside more stack space or memory area.

27 Out of Paper

The printer is out of paper, or the printer is not switched on.

You should insert paper (if necessary), verify that the printer is properly connected, and make sure that the power is on; then, continue the program.

6 Overflow

The number you entered is too large for the BASIC number format.

3 RETURN without GOSUB

A RETURN statement needs a previous GOSUB statement.

Correct the program. You probably need to put a STOP or END statement before the subroutine so the program doesn't "fall" into the subroutine.

15 String too long

You wrote a string more than 255 characters long. Try to break the string into smaller strings.

2 Syntax error

A line contains an incorrect sequence of characters, such as an unmatched parenthesis, a misspelled command or statement, or incorrect punctuation. Or, the data in a DATA statement doesn't match the type (numeric or string) of the variable in a READ statement.

When this error occurs, the line to be corrected is displayed. Edit the line or the program.

13 Type mismatch

You gave a string value where a numeric value was expected, or you had a numeric value in place of a string value.

8 Undefined line number

A line reference in a statement or command refers to a line which doesn't exist in the program.

Check the line numbers in your program, and use the correct line number.

Notes:

Appendix B. BASIC: A Quick Reference

This appendix lists the BASIC commands and statements you have been using in this book. These lists can serve you as a quick reference. For detailed information about each item, look up Chapter 4, "Commands, Statements, Functions, and Variables," in the *BASIC* Reference book.

Commands

-

The following is a list of BASIC commands you have used in this book. The parts of each command are shown, but not always in complete form. The purpose of each command is briefly explained.

Command	Action
DELETE line1-li	ne2
	Deletes specified program lines.
EDIT line	Displays a program line for changing.
LIST line1-line2	
	Displays program lines on the screen.
NEW	Erases the current program and variables.
RUN	Performs a program. The R option may be used to keep files open.

RUN lin	e
----------------	---

Runs the program in memory starting at the specified line.

Statements

This section lists the BASIC statements you have used in this book. The list tells what each statement does and shows its parts. For the more complex statements, the parts shown may not be complete. You can find detailed information about each statement in the *BASIC* Reference book.

Statement	Action
END	Stops the program, closes all files, and returns to command level.
FOR variable=	x TO y STEP z
	Repeats program lines a number of times. The NEXT statement closes the loop.
GOSUB line	Calls a subroutine by branching to the specified line. The RETURN statement returns from the subroutine.
GOTO line	Branches to the specified line.
IF expression T	HEN clause ELSE clause
	Performs the statement(s) in the
	THEN clause if the condition is met.
	Otherwise, performs the ELSE
	clause or goes to the next line.

LET variable=expression	
-	Sets the value for a variable.
NEXT variable	Closes a FORNEXT loop (see FOR).
REM remark	Includes remark in program.
BEEP	Makes the speaker sound a short note.
CIRCLE (x,y),r	Draws a circle with center (x,y) and radius r.
CLS	Clears the screen.
COLOR foreground, background, border	
	In text mode, sets colors for
	border screen.
COLOR backgrou	nd.palette
	In graphics mode, sets background
	color and palette of foreground colors.
DATA list of cons	tants
	Creates a data table to be used by READ statements.
DRAW string	Draws a figure as specified by string.
INPUT "prompt";variable list	
	Reads data from the keyboard.
LOCATE row,col	Positions the cursor. Other
	parameters allow you to define the
	size of the cursor and whether it is visible or not.

)

, Change and the second

PAINT (x,y),paint,boundary	
	Fills in an area on the screen defined by boundary with the paint color.
PLAY string	Plays music as specified by string.
PRINT list of exp	r essions Displays data on the screen.
PSET (x,y),color	Draws a point on the screen, in the foreground color if color is not specified.
READ variable list	Retrieves information from the data table created by DATA statements.
SCREEN mode	Sets screen mode, color on or off.
SOUND freq,dura	ition Generates sound through the speaker.
WIDTH size	Sets screen width. Other options allow you to specify the width of a printer or a communications file.

Appendix C. Reserved Words in BASIC

Certain words have special meaning to BASIC. These words are called *reserved words*. Reserved words include all BASIC commands, statements, function names, and operator names. Reserved words cannot be used as variable names.

You should always separate reserved words from data or other parts of a BASIC statement by using spaces or other special characters as allowed by the rules for BASIC.

Following is a list of all the reserved words in BASIC:

ABS	COMMON
AND	CONT
ASC	COS
ATN	CSNG
AUTO	CSRLIN
BEEP	CVD
BLOAD	CVI
BSAVE	CVS
CALL	DATA
CDBL	DATE\$
CHAIN	DEF
CHDIR	DEFDBL
CHR\$	DEFINT
CINT	DEFSNG
CIRCLE	DEFSTR
CLEAR	DELETE
CLOSE	DIM
CLS	DRAW
COLOR	
СОМ	

157

EDIT	
ELSE	
END	LINE
ENVIRON	LIST
ENVIRON\$	LLIST
EOF	LOAD
EQV	LOC
ERASE	LOCATE
ERDEV	LOF
ERDEV\$	LOG
ERL	LPOS
ERR	LPRINT
ERROR	LSET
EXP	MERGE
FIELD	MID\$
FILES	MKDIR
FIX	MKD\$
FNxxxxxxx	MKI\$
FOR	MKS\$
FRE	MOD
GET	MOTOR
GOSUB	NAME
GOTO	NEW
HEX\$	NEXT
IF	NOT
IMP	OCT\$
INKEY\$	OFF
INP	ON
INPUT	OPEN
INPUT#	OPTION
INPUT\$	OR
INSTR	OUT
INT	PAINT
INTER\$	PEEK
IOCTL	PEN
IOCTL\$	PLAY
KEY	PMAP
КПІ	POINT
I FFT\$	POKE
LLI I V	

POS	STOP
PRESET	STR\$
PRINT	STRIG
PRINT#	STRING\$
PSET	SWAP
PUT	SYSTEM
RANDOMIZE	TAB(
READ	TAN
REM	THEN
RENUM	TIME\$
RESET	TIMER
RESTORE	ТО
RESUME	TROFF
RETURN	TRON
RIGHT\$	USING
RMDIR	USR
RND	VAL
RSET	VARPTR
RUN	VARPTR\$
SAVE	VIEW
SCREEN	WAIT
SGN	WEND
SHELL	WHILE
SIN	WIDTH
SOUND	WINDOW
SPACE\$	WRITE
SPC(WRITE#
SQR	XOR
STEP	
STICK	

Notes:

Index

Special Characters B

< sign 96
<> sign 96
<= sign 96
+ sign 75
\$ sign 59
* key 45
* sign 75
- sign 75
/ key 45
/ sign 75
> sign 96
>= sign 96
= sign 96

A

about Chapter 1 9 about this book iii adding a line 28 addition 45 additional reading 3 arrow keys 14 Backspace key 14 BASIC commands 153 BASIC statements 21, 153, 154 non-I/O 154 BEEP statement 15, 21, 26 blank line 33, 52 blanks 157 breaking an endless loop 36

C

calculator functions 45, 46 chapter reviews Chapter 1 41 Chapter 2 75 Chapter 3 104 Chapter 4 145 CIRCLE statement 134 clearing the screen 13, 26 CLS statement 13, 21, 26 colon 15 COLOR statement 88, 90, 136 colors 88 background colors 88, 135, 139

foreground colors 88, 135, 139 comma 35, 37 commands in BASIC 10 BEEP statement 15, 21 CLS statement 13, 21 EDIT command 32 LIST command 28 NEW command 34, 35, 46, 60 **PRINT command** 10 PRINT statement 21, 33 SOUND statement 15, 17, 21 **STEP command 86** computer's memory 32 coordinates, screen 132 correcting a line 14 cursor 10, 14, 32

D

delay tactic 70 diagonal lines 128 division 45 division by zero 148 DRAW statement 128 drawing on the screen 127, 128, 132 duration 15, 52

EDIT command 32 editing a program 28, 32 editing your program 33 End key 14 END statement 26 endless loop 36 Enter key 10 erasing lines 12 error codes Appendix A error messages 30, Appendix Α FOR without NEXT 79 illegal function call 30, 44, 79 missing operand 30, 44, 79 NEXT without FOR 79 out of DATA 67 redo from start 49 syntax error 10, 30, 44, 79 type mismatch 44 undefined line number 44, 79 Esc key 12

F

E

flashing square cursor 32 Fn key 26 Fn-Break keys 36 FOR/NEXT statements 70, 84 frequency 15, 52 function (Fn) key 14 functions in BASIC 72 LEFT\$ function 64 LEN function 62 MID\$ function 66 RIGHT\$ function 64 F1 key 28 F2 key 26

G

getting a blank line 33, 52 GOSUB/RETURN statements 98 GOTO statement 35, 52 graphics mode 88, 128, 137

Η

hardware requirements 3

I

IF/THEN statements 94 INPUT Statement 48, 52, 60 Ins key 32 inserting words 32 introduction 1

J

joining strings 62

K

keys Backspace key 14 Ctrl key 14 cursor keys 14 End key 14 Enter key 10 Esc key 14 Fn key 28 F1 key 28 F2 key 28 Ins key 32 Num Lock key 14 Shift key 10

L

LEFT\$ function 64 LEN function 62

Index-4

LET statement 50 line numbers 26, 28 LINE statement 133 LIST command 28, 33 listing a program 28, 33 listing one line 33 LOCATE statement 81, 83 loops 70

M

making corrections 14, 28, 32 math problems 46 melody, writing a 107 memory of computer 32 messages 44, Appendix A MID\$ function 66 mini-instructions 72 multiplication 45 musical notes 107, 109 musical pause 118 musical timing 109, 112 musical tones 15, 52

N

NEW command 34, 35, 46, 60 numeric variables 47, 52, 66

0

octaves 113 overflow 150

P

PAINT statement 139 palette colors 137, 139 piano keyboard 107, 109 PLAY statement 107, 109 playing a melody 107 practice exercises for Lesson 1 20 for Lesson 2 39 for Lesson 3 55 for Lesson 4 73 for Lesson 5 92 for Lesson 6 102 for Lesson 7 124 for Lesson 8 143 **PRINT** command 10 PRINT statement 21, 26, 33, 52 print zones 35 program 26 program loops 70 program statement 26 **PSET statement** 132

Q

quotation marks 13

R

READ/DATA statements 66 REM statement 30, 34 reserved words in BASIC 60, 157 review of Chapter 1 41 review of Chapter 2 75 rewriting a line 32 RIGHT\$ function 64 RUN command 26 running a program 26, 32

S

screen columns 81 screen coordinates 132 screen graphics 128 screen positions 81, 127, 132 screen rows 81 SCREEN statement 88, 136 semicolon 34, 35, 60 setting conditions 96 shapes, drawing 128 slowing down a program 70 snipping characters 65 software requirements 3 SOUND statement 15, 17, 21, 52 spaces 157 spacing your output 60 comma 35, 37 print zones 34 semicolon 35 square flashing cursor 32 starting the computer 5 statements **CIRCLE statement** 134 COLOR statement 88, 90, 136 **DRAW** statement 128 FOR/NEXT statements 70, 84 **GOSUB/RETURN** statements 98 GOTO statement 35, 52 **IF/THEN statements** 94 INPUT statement 48, 60 LET statement 50 LOCATE statement 81, 83 PAINT statement 139 PLAY statement 107, 109 **PSET statement** 132 **READ/DATA** statements 66 REM statement 30, 34 **SCREEN** statement 88 statements in BASIC 21 **STEP command 86** string 59

string functions 62 string variables 57, 59 strings 13 subroutine 98, 100 subtraction 45 syntax error 10

T

tempo 109, 112 text mode 88, 90, 136 title for your program 30 tones 15

U

using colors 135 using the BEEP statement 15, 21 using the CLS statement 13, 21 using the keys * key 45 / key 45 arrow keys 14 Backspace key 14 Crtl-Break keys 36 End key 14 Enter key 10 F1 key 28 F2 key 26 Ins key 32 Shift key 13 using the LIST command 28 using the NEW command 34 using the PRINT command 10 using the PRINT statement 21 using the RUN command 26 using the SOUND statement 15, 17, 21

V

value 60 values 48 variables numeric variables 47, 52, 66 string variables 57, 59, 94

W

writing a program 26 writing music 107, 109





System requirements:



IBM Color Display or a color TV



64KB of memory Cartridge BASIC

©IBM Corp. 1983 All rights reserved

International Business Machines Corporation P.O. Box 1328-S Boca Raton, Florida 33432

6024116 Printed in USA