I Speak BASIC to My VIC™

Aubrey B. Jones, Jr.

A field-tested computer literacy course that introduces students to BASIC language programming.

HAYDEN

I Speak BASIC to My VIC™

Aubrey B. Jones, Jr.

HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

To Alyce, Aubrey III, and Adrienne

Acquisitions Editor: GARY MARKMAN
Production Editor: RONNIE GROFF
Art Director: JIM BERNARD
Compositor: VAN GROUW COMPOSITION CO., INC.
Printed and bound by: THE MURRAY PRINTING COMPANY

Library of Congress Cataloging in Publication Data

Jones, Aubrey B.

I Speak BASIC to my VIC.

Summary: An introduction to microcomputers and computer programming.

1. VIC 20 (Computer)—Programming—Juvenile literature.
2. Basic (Computer program language)—Juvenile literature.
[1. VIC 20 (Computer)—Programming. 2. Basic (Computer program language) 3. Computers. 4. Programming (Electronic computers)
I. Title.
[QA76.8.V5J67 1983] 001.64'2 83-52
ISBN 0-8104-6179-X

1 2 3 4 5 6 7 8 9 PRINTING

83 84 85 86 87 88 89 90 91 YEAR

Contents

Part 1 **The Hardware (Or The "Boxes")** **1**
Objectives; Typical Data Processing Operation; Basic Parts of a Computer; Summary; Practice 1

Part 2 **The Software ("The Program")** **17**
Objectives; How Humans Talk to Computers; A BASIC Program; VIC Keyboard; VIC Power-Up Rules; Summary; Practice 2

Part 3 **Your First Computer Program** **39**
Objectives; Writing Your First Computer Program; Executing Your Program; Expanding Your Program; Listing Your Program; Ending Your Program; Summary; Practices 3, 4, 5

Part 4 **More Programming Tools** **58**
Objectives; Mathematical Operations; Programming Mathematical Operations; A BASIC Mathematical Program—Area of a Rectangle; Print Zones; Practices 6, 7

Part 5 **Scientific Notation** **79**
Objectives; Scientific Notation; Review and Feedback; Practice 8

Part 6 **Relational Operators and IF-THEN/GOTO Statements** .. **86**
Objectives; Relational Operators; Using IF-THEN Statements (Conditional Branching); Using GOTO Statements (Unconditional Branching); Practices 9, 10

Part 7 **Input Statements** **99**
Objectives; Input Statements; Area of Rectangle Problem Revisited (Using Input Statements); String Variables; Practices 11, 12, 13

Part 8 **Using the Calculator Mode and Sizing Memory** **113**
Objectives; Bit vs. Byte; How Much Memory Is Used in BASIC Programs; Summary; Practice 14

Part 9 **Using the Cassette Recorder** **122**
Objectives; A Cassette Recorder as an Input/Output Device; Practices 15, 16, 17

Part 10 **Using FOR-NEXT-STEP Statements** **133**
Objectives; FOR-NEXT Statements and Loops; Comparison of GOTO, IF-THEN, and FOR-NEXT Program Loops; Loop Flowcharts; Timer Loops; Practices 18, 19

Part 11 **Reading Data** **149**
Objectives; READ-DATA Statements; Restore Function; Practice 20

Part 12 **Video Display Graphics** **163**
Objectives; Formatting—SPC(n), TAB(n); VIC Graphic Keys—Shift Mode; Drawing with Graphics; Practice 21

Part 13 **Arrays** **191**
Objectives; One-Dimensional Arrays; Two-Dimensional Arrays; DIM Statement; Summary; Practices 22, 23

Part 14 **INT(X), ABS(X), and RND(X) Functions** **206**
Objectives; INT(X) Function; ABS(X) Function; RND(X) Function; Summary; Practices 24, 25

Part 15 **Subroutines** **220**
Objectives; Subroutines; ON-GOTO; ON-GOSUB; Summary; Practices 26, 27

Extra Practices **236**

PART 1

The Hardware (Or The "Boxes")

What You Will Learn

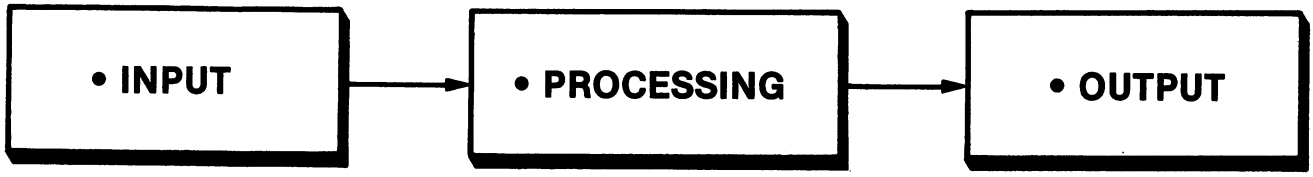
1. That the computer is a valuable tool that can solve problems, print words, draw pictures, store information, retrieve information, compare information, play games, and do many other things to help you in everyday life.
2. That people control computers and that computers cannot think (despite what you might have heard).
3. To identify and explain the basic parts of a computer and relate them to a "box diagram" of a general purpose computer.
4. To identify and explain the function of the basic parts of a VIC microcomputer.
5. To define and explain the terms hardware, software, microcomputer, microprocessor, RAM, ROM, processor, input unit, output unit, memory, and binary.
6. That computers are simple and easy to use; and above all that computers are fun!

Welcome to the World of Computers!

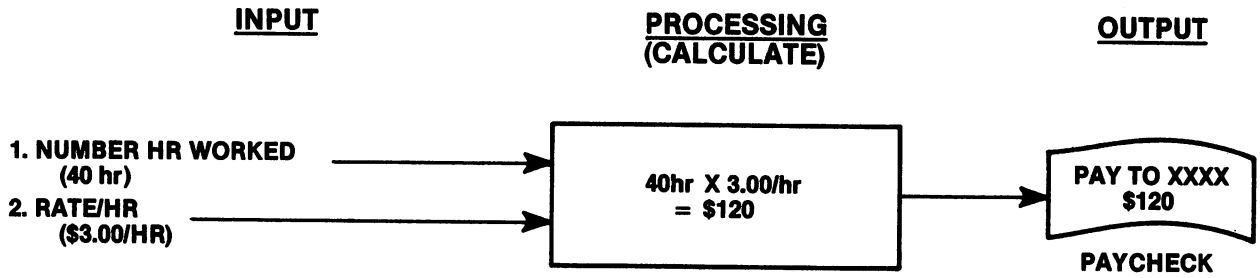
People Control Computers!

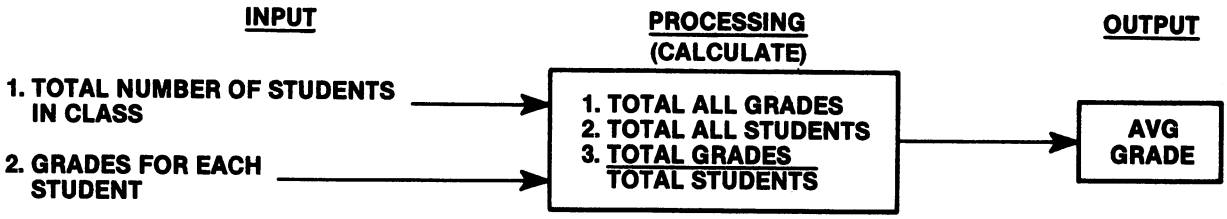
Computers Can't Think!

Typical Data Processing Operation
"Box" Diagram

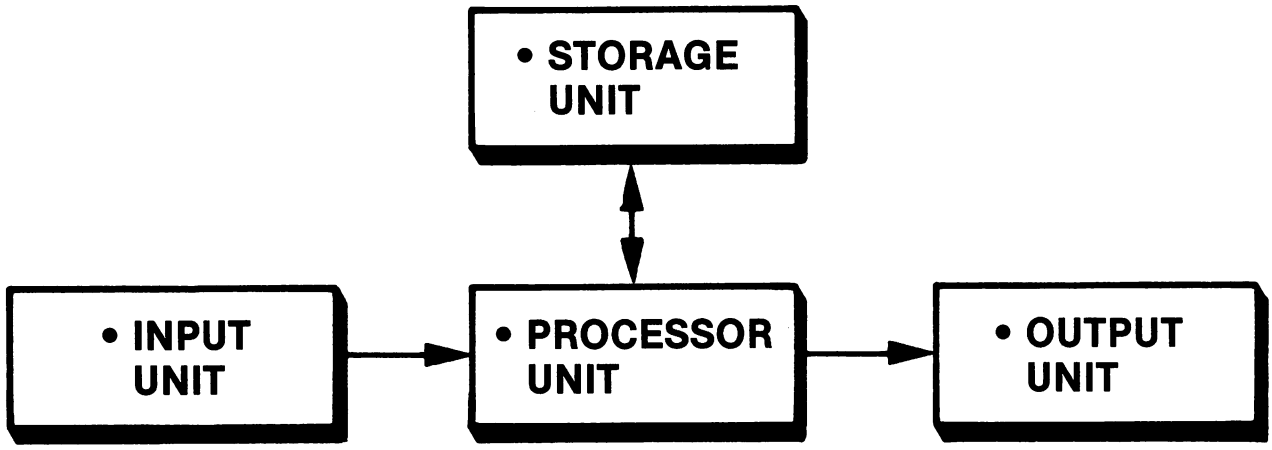


Examples of Data Processing Operation





BOX Diagram
Showing
Basic Parts of a Computer



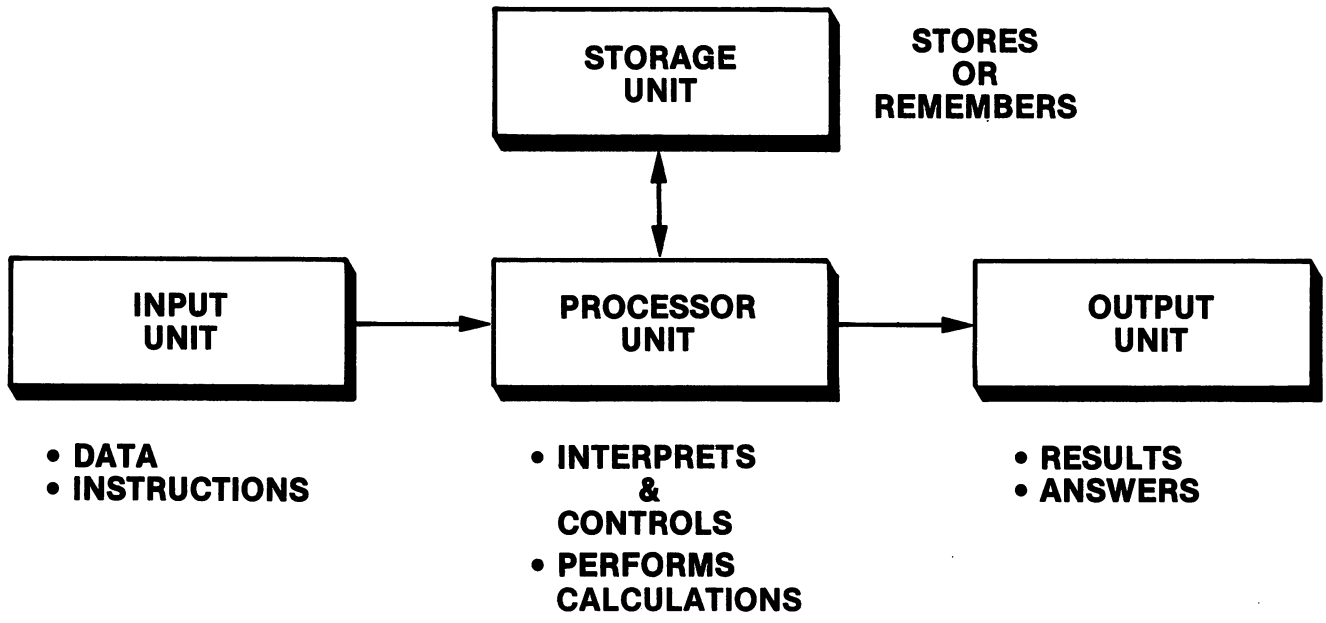
Stores or Remembers

- **Storage unit (memory)**
  - **— Stores both information and instructions until needed (requested)**

Interprets, Controls, & Calculates

- PROCESSOR UNIT
  - — INTERPRETS (DECODES) INSTRUCTIONS AND REGULATES (CONTROLS) THEIR EXECUTION
  - — PERFORMS ALL OF THE CALCULATIONS

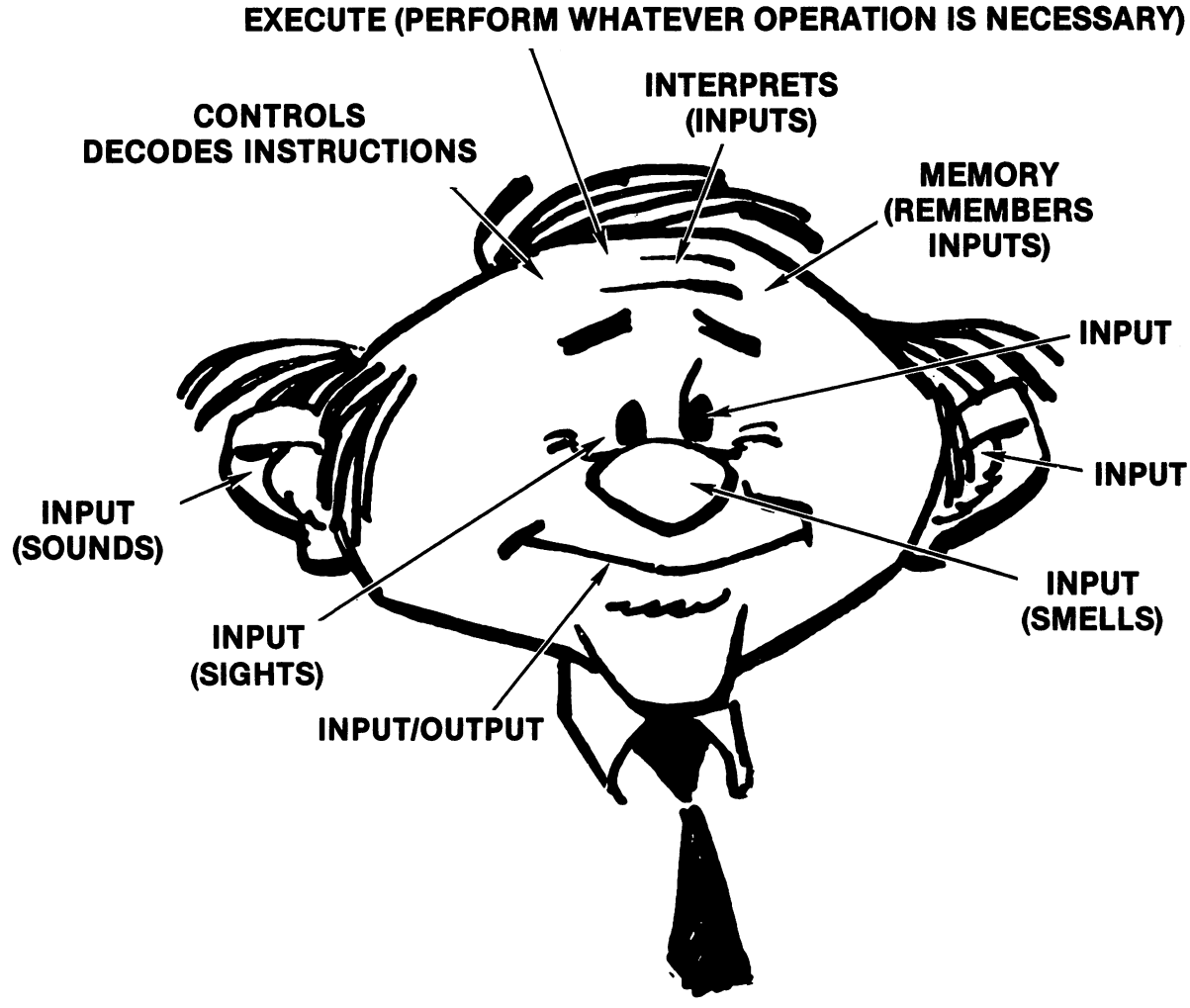
Box Diagram of a Basic Computer System



What We Have Learned

- INPUT ⟶ PROVIDES INSTRUCTIONS AND DATA
- STORAGE ⟶ STORES OR REMEMBERS (MEMORY)
- PROCESSOR ⟶ INTERPRETS, CONTROLS, & CALCULATES
- OUTPUT ⟶ PROVIDES ANSWERS AND RESULTS

"Human Computer"
Man Can Think But Computer Can't!

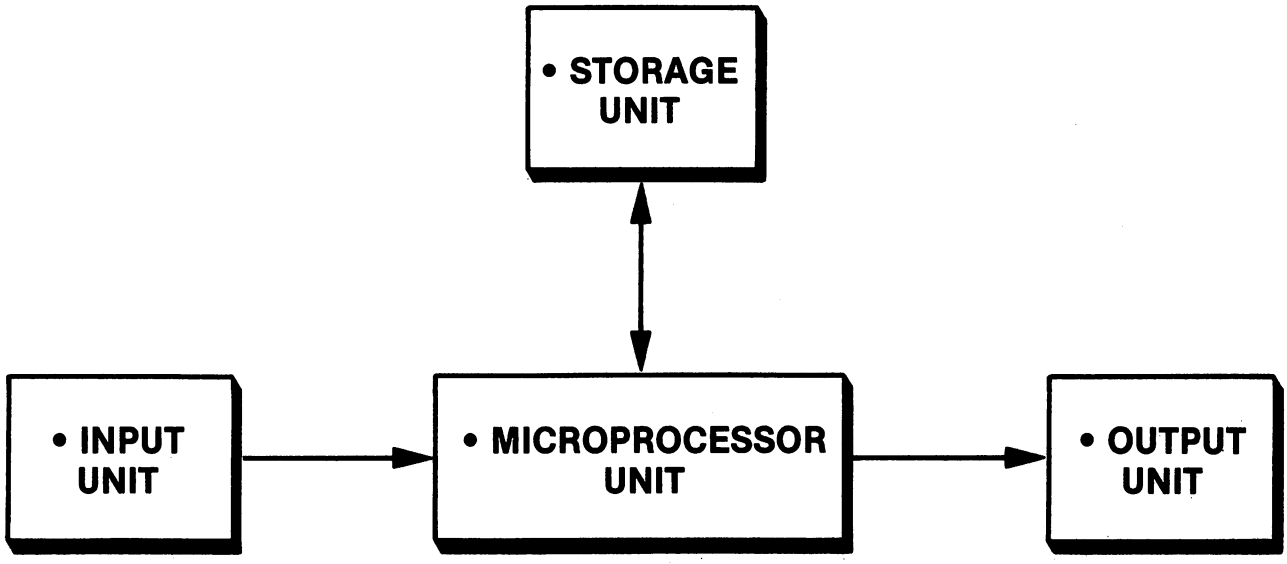


Some Terms You Should Know

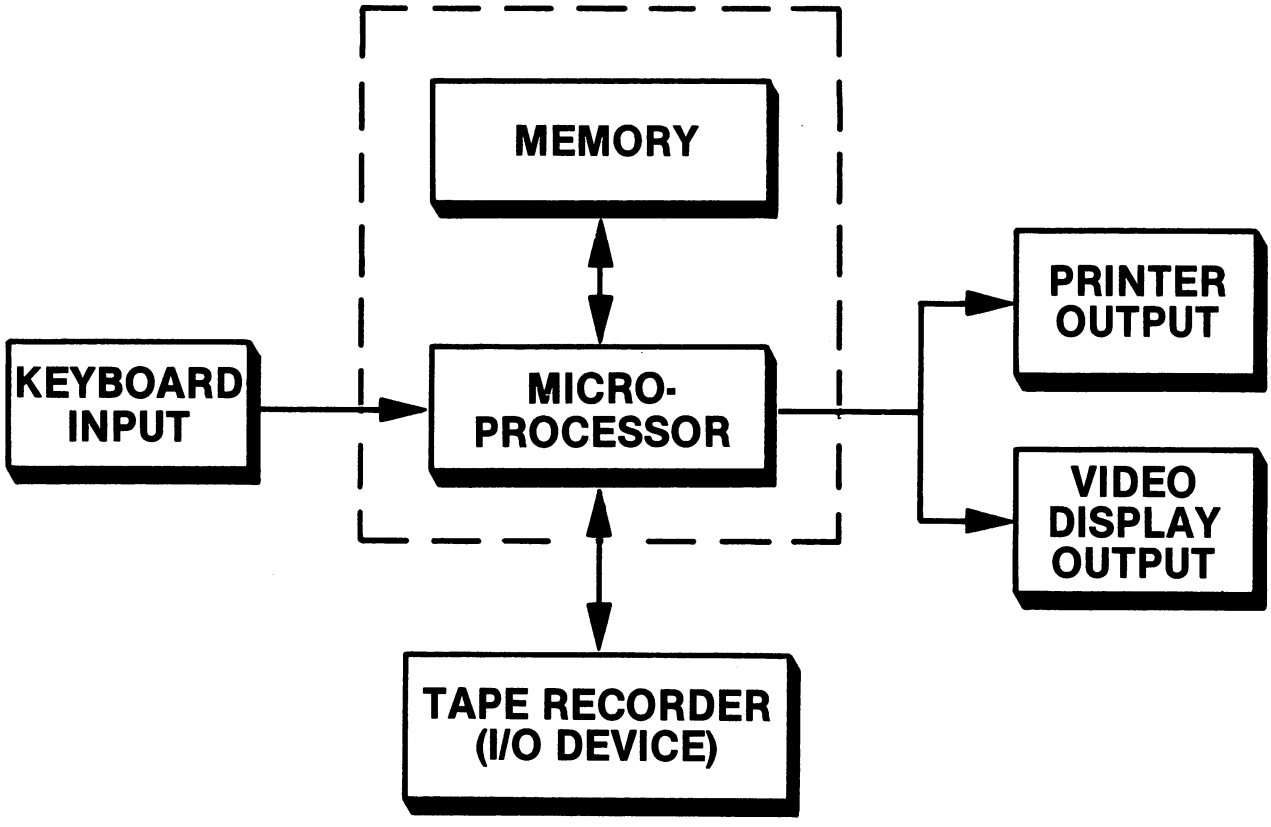
- **MICROPROCESSOR**
- **MICROCOMPUTER**
- **RAM**
- **ROM**

- **MICRO = Very small**
- **MICROPROCESSOR = Very small processor**
- **RAM = Random access memory**
  - **— CAN BE changed by the user**
  - **— Information stored in RAM will be destroyed if power fails or turned-off (volatile)**
- **ROM = Read only memory**
  - **— CANNOT be changed by the user**
  - **— Information stored in ROM is not destroyed if power fails or is turned-off (non-volatile)**
  - **— Control program (BASIC compiler) stored here**

Box Diagram of a Microcomputer



Basic Components of the VIC Computer



*VIC HAS 20K ROM AND 5K RAM
EXPANDABLE TO 32K

Courtesy of Commodore Business Machines, Inc.

Courtesy of Commodore Business Machines, Inc.

What We Have Learned

DATA PROCESSING OPERATION STEPS:	BASIC COMPUTER PARTS:	MICROCOMPUTER PARTS:
• INPUT →	• INPUT UNIT →	• INPUT UNIT
• PROCESSING →	• PROCESSOR UNIT + MEMORY UNIT →	• MICROPROCESSOR + MEMORY
• OUTPUT →	• OUTPUT UNIT →	• OUTPUT UNIT

PRACTICE 1

Box Diagram of a Computer

1. Draw the BOX DIAGRAM of a BASIC computer.
   a. Label each box with the correct name.
   b. List the functions of each box.

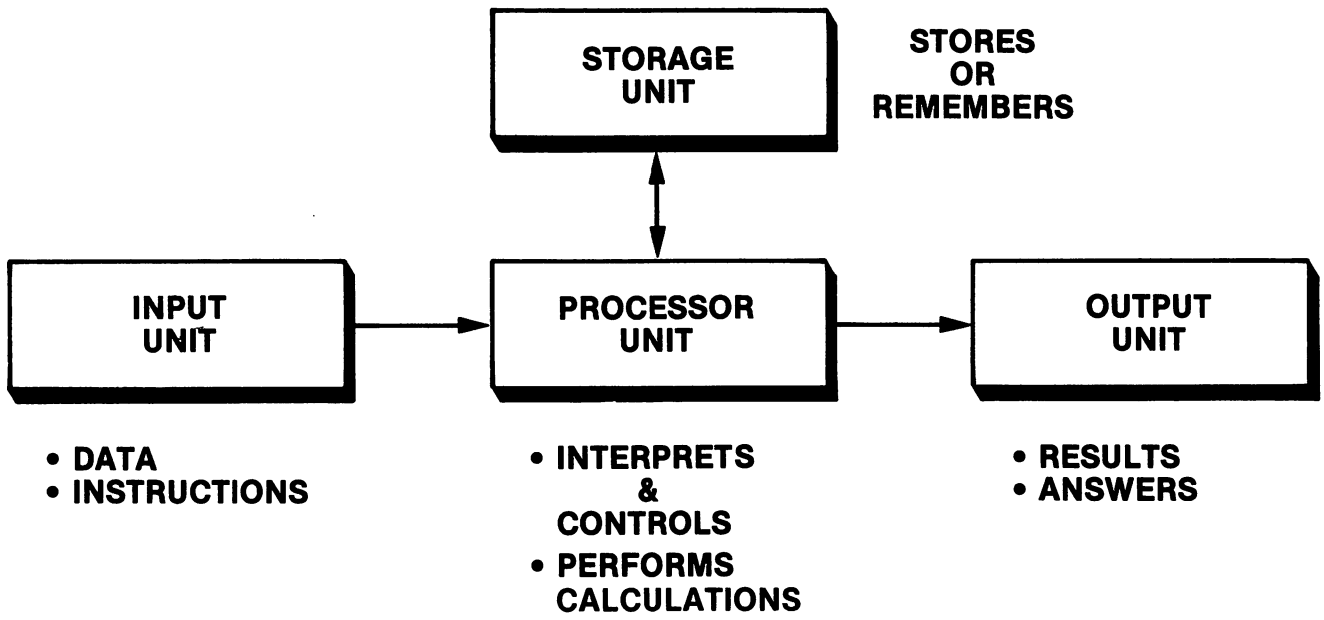
PART 2

The Software (The "Program")

What You Will Learn

1. To define the terms hardware, software, BASIC, binary, and compiler, and to relate them to computers.
2. That computers speak a foreign language: machine language.
3. How humans talk to computers via a programming language called BASIC.
4. To identify the principal parts of a BASIC program.
5. To identify and explain the purpose of all the keys on the VIC keyboard.
6. How to connect and power up a VIC microcomputer.

Box Diagram of a Basic Computer System



More Terms You Should Know

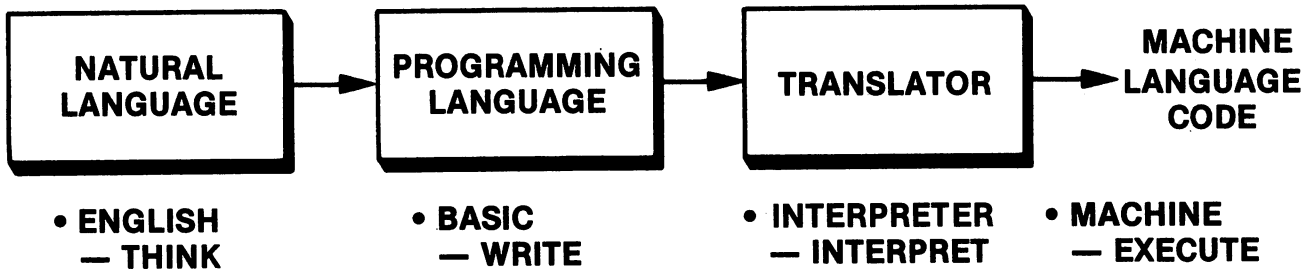
- **HARDWARE**
  - **— THE COMPUTER AND COMPUTER RELATED EQUIPMENT (THE BOXES)**

- **SOFTWARE**
  - **— THE INSTRUCTIONS FOR THE COMPUTER (THE PROGRAM)**

Computers Speak a Foreign Language! (No Speak English, French, German Spanish, or Any Other Natural Language)

- **COMPUTERS SPEAK IN *MACHINE* LANGUAGE**
  - **— MACHINE LANGUAGE IS A FORM OF *BINARY* CODING**
  - **— BINARY IS A WORD DENOTING "TWO"**
  - **— MACHINE LANGUAGE USES TWO BASIC SYMBOLS: "Ø" AND "1"**

How Humans Talk to Computers



- ENGLISH — THINK
- BASIC — WRITE
- INTERPRETER — INTERPRET
- MACHINE — EXECUTE

- **BASIC**

  (Beginner's all-purpose symbolic instruction code)

  — Popular programming language for writing instructions to the computer

- **INTERPRETER**

  — Translates BASIC into machine code
  — (You really don't have to know anything about an interpreter since it is used automatically when you run a BASIC program)
  — Located in the ROM in the VIC

To Program You Must Learn the Language First!

A Comparison between English and BASIC

ENGLISH LANGUAGE

- **Words**
  - — Used to make sentences
- **Sentences**
  - — Used to make paragraphs
- **Paragraphs**
  - — Lengths vary
- **Commands**
  - — Can be one word
  - — e.g., STOP! HALT!
- **Sentence Numbers**
  - — Optional (seldom used)

BASIC PROGRAMMING LANGUAGE

- **Key Words**
  - — Used to make statements
- **Statements**
  - — Used to make programs
- **Programs**
  - — Lengths vary
- **Commands**
  - — Executed immediately
  - — e.g., NEW, LIST, RUN
- **Line Numbers**
  - — Must be used for each statement

Learning a New Vocabulary

Here Are the Key Words and Commands You'll Learn:

KEY WORDS

- PRINT
- END
- LET
- INPUT
- GO TO
- IF --- THEN
- REM
- STOP
- FOR --- NEXT
- READ-DATA

COMMANDS

- NEW
- LIST
- RUN
- CONT

Commands vs. Statements

COMMANDS

— **Executed as soon as you type them and press RETURN**

STATEMENTS

— **Put into programs and are only executed after you type the command RUN and press RETURN**

A BASIC Program

	LINE NUMBER	KEY WORD	OTHER PART OF THE STATEMENT	"LOOK AT" REQUEST*
1st STATEMENT	1Ø	PRINT	"HELLO THERE"	RETURN
2nd STATEMENT	2Ø	PRINT	"YOUR NAME"	RETURN
3rd STATEMENT	3Ø	END		RETURN
COMMAND	RUN			RETURN

*Pressing the RETURN key tells the computer to "LOOK AT" (and store) what you have just typed. You must press this key after each statement or command.

Line Numbers

- Serve as a guide to the computer in running the program.
- Tell the computer in what order it should carry out your instructions.
- Computer will start executing at lowest numbered line unless told to start elsewhere.
- Normally are multiples of 5's, 10's, or some other multiples to leave space for inserting new program lines between old one.
- Although it is perfectly legal to number program lines more closely (like 1, 2, 3, 4, etc.), don't do it!

Key Words

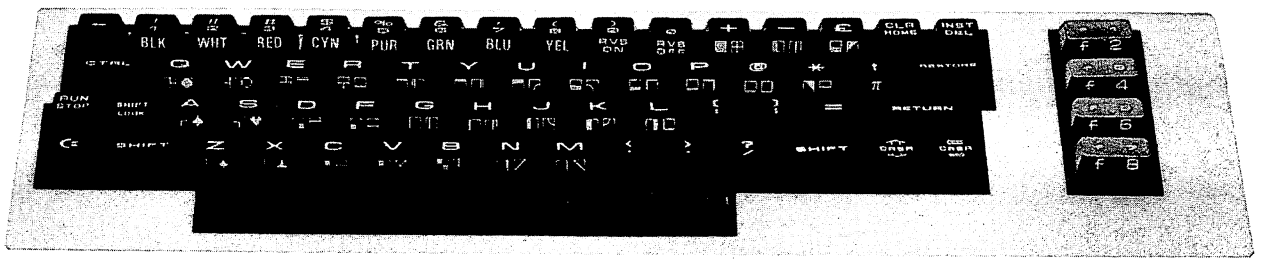
- **Never used alone**
- **Need line number**
- **Always part of a BASIC statement that has some other part to it***
- **Executed only after command RUN is typed and RETURN key is pressed**

***To the purist, we know that key words like END and STOP can be used alone; but you still need line numbers, and you must type RUN and press RETURN to execute.**

What We Have Learned

- **Key words**
  - **— Used to make statements**
- **Statements**
  - **— Must have line numbers and key words**
  - **— Used to make programs**
- **Programs**
  - **— May vary in length**
- **Commands**
  - **— Executed as soon as you type them and press RETURN**

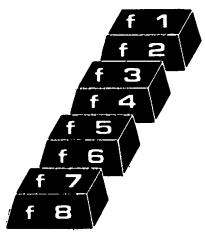
VIC KEYBOARD



Courtesy of Commodore Business Machines, Inc.

Special Function Keys on the VIC Keyboard

KEY	FUNCTION
RETURN	• Causes the computer to "look at" the line you just typed in and to act accordingly. The key must be pressed each time you want to enter a line from the keyboard.
SHIFT	• Some keys have two characters printed on them. Use this key to type CLR, CRSR ⇧, CRSR ⇦, INST, and graphic symbols. The SHIFT key must be held down while pressing any other key to give shifted character of that key.
SHIFT LOCK	• Pressing this key until it "clicks" into place holds SHIFT key down so that both hands are free to type in shifted mode. To release the SHIFT/LOCK key, just press and release the key.
RUN STOP	• STOP stops execution of a program. To continue execution type CONT and press RETURN.
	• RUN causes the next program on the optional tape cassette unit to be located, loaded into memory, and then executed immediately. RUN is obtained by using SHIFT key with RUN STOP key.
RESTORE	• "Resets" the computer with the advantage that any programs you had in memory are retained. To reset the computer, you must hold down the RUN STOP key while pressing the RESTORE key.
PROGRAMMABLE FUNCTION KEYS	• The four tan keys located on the right side of the keyboard can be assigned tasks or functions by the programmer (you). This permits you to assign special functions to these keys. By using these keys with and without the SHIFT key, you can get a total of eight (8) assignable function keys. These function keys are not assigned when you first turn on the VIC, however, but typically are used if an application program or cartridges containing special programs assign a function to these keys.



Cursor Control Keys

KEY	FUNCTION
CLR HOME	• HOME moves the cursor to the upper left-hand corner of the screen (i.e., to its HOME position). Screen remains the same (i.e., *NOT* cleared). • CLR clears the screen and homes the cursor. CLR is obtained by pressing the CLR/HOME key while holding down the SHIFT key.
⇧ CRSR ⇩	• CRSR ⇩ or cursor down moves the cursor down one column. • CRSR ⇧ or cursor up moves the cursor up one line each time the key is pressed in the shifted mode (that is, holding down the SHIFT key while CRSR key is pressed).
⇦ CRSR ⇨	• CRSR ⇨ or cursor right moves the cursor to the right one character position. When the cursor reaches the end of a line, it "wraps around" the screen and moves to the beginning of the next line down. • CRSR ⇦ or cursor left moves the cursor to the left one position. When the cursor reaches the end of a line, it "wraps around" and moves up one row and to the extreme right-hand end of this row. Cursor ⇦ is a shifted character.
INST DEL	• DEL or DELETE backspaces the cursor or moves the cursor to the left one character position and erases the last character typed. • INST or INSERT permits you to insert additional characters in a line by opening a space in the line at the current cursor position. INST is obtained when SHIFT key is held down.

Graphics (C=), Control (CTRL) Keys, and Color (1-8)

C= • This key is called the "Graphics" or "Commodore" key. There are two graphic characters on each of the graphic keys. To get the graphics on the left side of the key, simply hold down the C= key and press the desired graphics key. To get right-side graphics, hold down the SHIFT key while pressing the desired graphics key. (Refer to Part 12, Video Display Graphics, on page 163 for more details.)

CTRL • Stands for "Control." This key is used with color keys to select the colors that you create on the VIC screen. The CTRL key works like the SHIFT key (that is, you must hold it down while pressing the desired color key). It also:

— provides you with the ability to define your own control commands that you can incorporate into applications you might develop for the VIC;

— is used with some plug-in cartridges to perform special functions;

— slows down the program if held down while program is running.

CTRL COLOR • Permits you to change the colors of the characters displayed by holding down the CTRL key and pressing one of the eight (8) color keys located on the top row of the keyboard. A short-hand notation for each color is shown on the face of the keys (as shown below).

CTRL BLK — Black	CTRL PUR — Purple
CTRL WHT — White	CTRL GRN — Green
CTRL RED — Red	CTRL BLU — Blue
CTRL CYN — Cyan (light blue)	CTRL YEL — Yellow

Note: Once you "set" a color, everything you type will be in that color until you change colors again. (You try it!)

CTRL RVS ON • Reverses the images the VIC puts on the screen. Everything you type will be reversed. To reverse the image, hold down the CTRL key while pressing RVS ON . (Try it!)

CTRL RVS OFF • Gets the screen back to normal. Hold down CTRL and press RVS OFF .

VIC Power-Up Rules

ACTION

1. **Make certain the VIC microcomputer is connected properly (refer to User Manual if you have questions).**
2. **If the tape recorder is connected, it should be in the *STOP* mode. (This procedure assumes that you are not using a disk.)**
3. **Turn on the VIC. The power switch is located on the right side of the VIC.**
4. **After a few seconds the message should appear on the screen as shown.**
   - **NNNN BYTES FREE line shows how much memory is available to you. For example:**
     NNN = 3,583* for a 4K VIC system
     NNNN = 14,847* for a 16K VIC system
     NNNN = 31,231* for a 32K VIC system

     ***(May vary slightly on some machines)**
   - **The blinking or flashing ■ is called the cursor. The cursor must be present in order to enter commands from the keyboard.**
5. **You are now ready to use VIC BASIC. (Note! If your VIC does not display the message shown above, then turn off the power, wait a few seconds, and then turn power back on.)**

DISPLAY

```
***CBM BASIC V2***
NNNN BYTES FREE
READY
■ ←CURSOR
```

Two Modes of VIC (Important to Remember)

- **The VIC can operate in two modes, and it is important that you know which mode the computer is in. The two modes are:**
  - **— Upper-Case/Full Graphics Mode**
  - **— Lower-Case/Text Mode (Including Left-Side Graphics)**
- **Upper-Case/Full Graphics Mode**
  - **— When you turn on the VIC, you are automatically in the "Full Graphics" mode, which means you can type upper-case letters and 62 graphic characters.**
- **Lower-Case/Text Mode**
  - **— If you press the SHIFT and C= keys at the same time, you put the VIC into "text" mode. This permits you to use the VIC as an ordinary typewriter, with full upper- and lower-case letters plus all of the graphics on the left side of the graphics key.**

NOTE: To get back to Upper-Case/Full Graphics Mode, hold down the SHIFT key and press C= key.

Getting It Together

- STEP 1 — WRITE YOUR PROGRAM
- STEP 2 — GET THE COMPUTER READY
- STEP 3 — ENTER YOUR BASIC PROGRAM
- STEP 4 — RUN YOUR PROGRAM
- STEP 5 — SIGN OFF

PRACTICE 2

Becoming Familiar with Your VIC

Become familiar with the VIC microcomputer by doing the following:

1. Turn on the VIC using the Power-up Rules (see page 35).
2. How many buttons did you have to press? a) _____
   and where was the button located? b) _____
3. Locate the [SHIFT] key.
   a. How many [SHIFT] keys are there on the keyboard? _____
   b. Hold down the [SHIFT] key and press every key that has a second symbol on it (e.g., pressing [1] and [2]). What happened? _____
   (Note! If you see some symbols appear on the screen, don't worry about what they are used for because you will learn more about them later.)
   c. What happens if you hold down the [SHIFT] key and press the [↑ CRSR ↓] and [← CRSR →] keys? _____
   d. What happens if you hold down the [SHIFT] key and press [CLR HOME] key? _____
4. Move the cursor to the right by pressing the [← CRSR →] key several times. Then, press [CLR HOME] key. What happened? _____
5. Move the cursor down by pressing the [↑ CRSR ↓] key several times. Then press [CLR HOME] key. What happened? _____
6. Become familiar with the cursor control keys by using them to move the cursor all over the screen. Where is the HOME position for the cursor? _____
7. Press every key on the keyboard to see what appears on the screen. Do this in both the shifted (i.e., holding down [SHIFT] key while pressing another key) and the unshifted modes.
8. The cursor keys, [← CRSR →] and [↑ CRSR ↓], have a repeat feature that keeps the cursor moving until you release the key. (You try it!)

PART 3

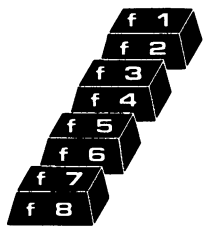
Your First Computer Program

What You Will Learn

1. To enter and run your first BASIC program.
2. To explain the purpose and use of the following BASIC commands: LIST, NEW, RUN.
3. To explain the purpose and use of the following key words: PRINT, PRINT (for spacing), REM, END.
4. To explain the purpose and use of the following special function keys: [↑ CRSR ↓], [← CRSR →], [RETURN], [SHIFT], [RUN STOP], [CLR HOME], [INST DEL], [SHIFT LOCK].
5. To explain the purpose and use of the following miscellaneous points: ■ cursor, " " (quotes), line numbers, reset button, power-up rules.

Special Function Keys on the VIC Keyboard (Review)

KEY	FUNCTION
RETURN	• Causes the computer to "look at" the line you just typed in and to act accordingly. The key must be pressed each time you want to enter a line from the keyboard.
SHIFT	• Some keys have two characters printed on them. Use this key to type CLR, CRSR ⇧, CRSR ⇦, INST, and graphic symbols. The SHIFT key must be held down while pressing any other key to give shifted character of that key.
SHIFT LOCK	• Pressing this key until it "clicks" into place holds SHIFT key down so that both hands are free to type in shifted mode. To release the SHIFT/LOCK key, just press and release the key.
RUN STOP	• STOP stops execution of a program. To continue execution type CONT and press RETURN.
	• RUN causes the next program on the optional tape cassette unit to be located, loaded into memory, and then executed immediately. RUN is obtained by using SHIFT key with RUN STOP key.
RESTORE	• "Resets" the computer with the advantage that any programs you had in memory are retained. To reset the computer, you must hold down the RUN STOP key while pressing the RESTORE key.
PROGRAMMABLE FUNCTION KEYS	• The four tan keys located on the right side of the keyboard can be assigned tasks or functions by the programmer (you). This permits you to assign special functions to these keys. By using these keys with and without the SHIFT key, you can get a total of eight (8) assignable function keys. These function keys are not assigned when you first turn on the VIC, however, but typically are used if an application program or cartridges containing special programs assign a function to these keys.



Cursor Control Keys (Review)

KEY	FUNCTION
CLR HOME	• HOME moves the cursor to the upper left-hand corner of the screen (i.e., to its HOME position). Screen remains the same (i.e., *NOT* cleared). • CLR clears the screen and homes the cursor. CLR is obtained by pressing the CLR/HOME key while holding down the SHIFT key.
⇧ CRSR ⇩	• CRSR ⇩ or cursor down moves the cursor down one column. • CRSR ⇧ or cursor up moves the cursor up one line each time the key is pressed in the shifted mode (that is, holding down the SHIFT key while CRSR key is pressed).
⇦ CRSR ⇨	• CRSR ⇨ or cursor right moves the cursor to the right one character position. When the cursor reaches the end of a line, it "wraps around" the screen and moves to the beginning of the next line down. • CRSR ⇦ or cursor left moves the cursor to the left one position. When the cursor reaches the end of a line, it "wraps around" and moves up one row and to the extreme right-hand end of this row. Cursor ⇦ is a shifted character.
INST DEL	• DEL or DELETE backspaces the cursor or moves the cursor to the left one character position and erases the last character typed. • INST or INSERT permits you to insert additional characters in a line by opening a space in the line at the current cursor position. INST is obtained when SHIFT key is held down.

VIC Power-Up Rules (Review)

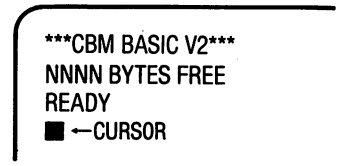
ACTION

1. **Make certain the VIC microcomputer is connected properly (refer to User Manual if you have questions).**
2. **If the tape recorder is connected, it should be in the *STOP* mode. (This procedure assumes that you are not using a disk.)**
3. **Turn on the VIC. The power switch is located on the right side of the VIC.**
4. **After a few seconds the message should appear on the screen as shown.**
   - **NNNN BYTES FREE line shows how much memory is available to you. For example:**

     NNN = 3,583* for a 4K VIC system
     NNNN = 14,847* for a 16K VIC system
     NNNN = 31,231* for a 32K VIC system

     ***(May vary slightly on some machines)**
   - **The blinking or flashing ■ is called the cursor. The cursor must be present in order to enter commands from the keyboard.**
5. **You are now ready to use VIC BASIC. (Note! If your VIC does not display the message shown above, then turn off the power, wait a few seconds, and then turn power back on.)**

DISPLAY



Typical Display Readout

```
1Ø          PRINT          "HELLO THERE"

2Ø          PRINT          "YOUR NAME"

3Ø          END

RUN
```

Writing Your First Computer Program

YOUR ACTION

1. **Before you start typing your program, always type NEW and press the RETURN key.**
2. **Type the line exactly as shown: ⟶**
3. **Use SHIFT key to type the upper characters like the quotation marks (") and the exclamation point(!).**
4. **Do *not* press RETURN key yet!**
5. **Go back and examine your typed line *very carefully*. Did you make a mistake? If you did, just use the backspace key DEL to erase a character.**
   (Note: If you made a mistake at the beginning of the line, you will have to erase your way back to that point and then retype the rest of the line.)
6. **Is everything OK? If it is, you can press RETURN. (This tells the computer to "look at" what you just typed in.)**
7. **The prompt] should appear. The computer is saying, "It's your turn ...I'm waiting for you."**

DISPLAY

```
10 PRINT "HELLO THERE
NAME!" ■
```

Ⓐ Ⓑ

```
10 PRINT "HELLO THERE
NAME!"
READY
■
```

NOTE

Ⓐ **Insert student's first name**

Ⓑ **The VIC has 22 columns across the screen. If the line you typed has more than 22 characters (including spaces, numbers, and symbols), the characters will "spill over" or "wrap around" to the next line automatically.**

Common Errors

- Missing quotes (")
- Too many quotes
- Forgot the key word PRINT
- Forgot the line number
- Forgot to press RETURN
- Used the character "O" for the number "ZERO" (Ø).

(Note: A slash is used to help you to recognize a zero. Look at your keyboard closely.)

Writing Your First Computer Program — Almost? (Errors)

PROBLEM (You Forgot to Follow Instructions)	SOLUTION
1. MISSING QUOTES (") — You forgot to enclose everything after the word PRINT in quotation marks. (If you want something printed, don't forget the quotation marks!)	• If you have already pressed RETURN, you must retype the entire line to correct your error. Here is how you do it:
2. TOO MANY QUOTATION MARKS — You typed too many. (That won't work either!)	• Type in the same line number you wish to change (1Ø in this case). That is, if you want the computer to replace that line with the corrected line.
3. FORGOT THE KEY WORD PRINT — You forgot to type PRINT. (How will the computer know you want to print if you don't tell it to print?)	• Next, retype the line exactly as shown on previous page. (But follow directions this time, Dummy!)
	• Then, check line over for errors.
4. FORGOT TO TYPE THE LINE NUMBER (1Ø) — Line numbers tell the computer where to start. The computer always starts executing from the lowest numbered line unless you tell it to start elsewhere. (We will show you how to tell the computer to start at another line later. Keep the faith!)	• If everything is OK, don't forget to press RETURN! When you press RETURN it tells the computer to "look at" what you just typed and to act accordingly.

Read this page if you had any errors! Then correct your errors before going to the next page!

Executing Your Program

YOUR ACTION	DISPLAY
1. Tell the computer to execute or run your program. The command for this is simple: RUN.	
2. So type *RUN* and press [RETURN].	
3. If you made no mistakes, the display will read: ⟶	HELLO THERE NAME! READY ■
4. If it did not work, try again (i.e., check your program for errors).	
5. If it did work, let out a yell, "HEY, I CAN DO IT TOO!"	

Go to next page (if you completed this one OK)

Expanding Your Program

YOUR ACTION	DISPLAY
1. You now have a program in the computer. (Unless you turned it off. If you did, retype line as shown): →	1Ø PRINT "HELLO THERE NAME"
2. Type in line 2Ø *exactly* as shown: →	2Ø PRINT "I'M GOING TO MAKE YOU A SUPERSTAR!"
3. Check your new line (2Ø) *very carefully*, especially the quotation marks.	READY ■
4. Everything OK? Press RETURN. (Remember, always press RETURN if you want the computer to look at what you typed.)	
5. Let's run your program. Type RUN and press RETURN.	
6. If you did it right, the screen will read: →	HELLO THERE NAME!
7. If it did not work, check your program for errors.	I'M GOING TO MAKE YOU A SUPERSTAR!
	READY ■

Go to next page

Using the Print Statement for Spacing

YOUR ACTION

1. **Look at your video display. Would you like more space between lines 1Ø and 2Ø? OK, this is how you do it.**

2. **Type in a new line as shown ⟶ and then press RETURN.**

3. **Now type *RUN* and press RETURN**

4. **WOW! A *PRINT* "nothing" puts a space between what you told the computer to print in Lines 1Ø and 2Ø.**

5. **Observe that the PRINT statement (Line 15) was placed between Lines 1Ø and 2Ø. Since you were smart enough to number your lines by 1Ø's, it was much easier to modify your program. (That's because you left room to insert new lines between the old ones.) Although it is perfectly legal to number program lines more closely (like 1, 2, 3, 4), don't do it.**

Go to next page

DISPLAY

```
HELLO THERE NAME!
I'M GOING TO MAKE YOU
A SUPERSTAR!
READY
■
```

```
15 PRINT
```

```
HELLO THERE NAME!

I'M GOING TO MAKE YOU
A SUPERSTAR!
READY
■
```

Inserting Remarks into a Program (But Not Printing Them Out)

YOUR ACTION

1. **Another important statement is REM, which stands for remark. It is often convenient to insert remarks into a program. The main reason for inserting remarks is so you or someone else can refer to them later and know what the program is for and how it is used.**
2. **When you tell the computer to execute the program by typing RUN and pressing RETURN, it will skip right over any number line that begins with the statement REM. The REM statement will have no effect on the program. (Let's see about that!)**
3. **Type Line 5 exactly as shown and then press RETURN (*'s are just for decoration).**

DISPLAY

```
5 REM *THIS IS MY FIRST
COMPUTER PROGRAM*
READY
■
```

4. **Type RUN and press RETURN.**
5. **It is the same as before (REM statement was not printed).**

```
HELLO THERE NAME!

I'M GOING TO MAKE YOU
A SUPERSTAR!
READY
■
```

Go to next page

Listing Your Program
(Looking At Your Program to See What It Contains)

YOUR ACTION*

DISPLAY

1. **To list your program is easy. The command is LIST.**
2. **Now you type LIST and press RETURN:**

```
 5 REM THIS IS MY FIRST
COMPUTER PROGRAM
10 PRINT "HELLO THERE
NAME!"
15 PRINT
20 PRINT "I'M GOING TO
MAKE YOU A SUPERSTAR"
READY
■
```

3. **You can call for a listing of your program any time the cursor ■ appears on a line by itself.**
4. **Also, you might only want to list one line. Type LIST 20 and press RETURN and the screen will display:**

```
20 PRINT "I'M GOING TO
MAKE YOU A SUPERSTAR"
READY
■
```

5. **You might also want to list several program lines, starting at one line and ending at another. For example, type List 10 - 20 and RETURN.**

```
10 PRINT "HELLO THERE
NAME!"
15 PRINT
20 PRINT "I'M GOING TO
MAKE YOU A SUPERSTAR!"
READY
■
```

***Note: So you can start with a clean display, hold down SHIFT key while pressing CLR HOME to clear screen.**

Go to next page

Ending Your Program

YOUR ACTION

1. **The end of a program is the last statement you want the computer to execute. Most computers require you to place an END statement after this point, so the computer will know it is finished. However, the VIC does *not* require an END statement. (Other computers might require it though.)**
2. **Let's add an END statement to your program. Type and enter: ⟶**
3. **Now type RUN and press RETURN.**
4. **No change from before! The program ended, but it did not print "END."**
5. **Let's make it print the END. (How do we do that?)**
6. **Oh, I remember! We need a PRINT statement. So let's try it. Type and enter: ⟶**
7. **Now *RUN* your program.**
8. **IT WORKED AGAIN! (If not, check the program.)**
9. **Note that there is no space between THE END and the line above it. Why? (Because you did not tell the computer to put a space between them!)**

DISPLAY

```
99 END

HELLO THERE NAME!

I'M GOING TO MAKE YOU
A SUPERSTAR!
READY
■
```

```
98 PRINT "THE END"
```

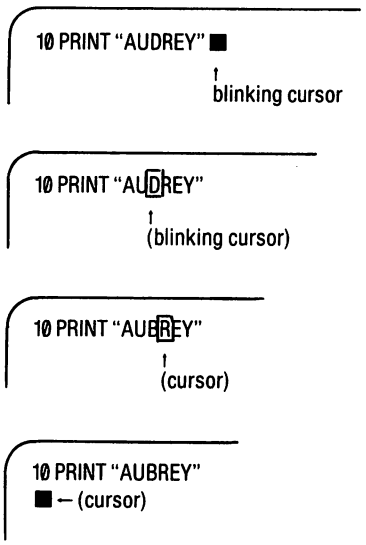
```
HELLO THERE NAME!

I'M GOING TO MAKE YOU
A SUPERSTAR!
THE END
READY
■
```

Using the CRSR ⇦ Key to Save Retype Time

YOUR ACTION*

1. You typed Line 10 as shown but have *not* pressed [RETURN] (blinking cursor at the end of that line indicates you have not pressed [RETURN]).
2. You wish to change the "D" to a "B" or to PRINT AUBREY. So you use the [CRSR ⇦] key to move the cursor to the left one space at a time. (Don't forget to use the [SHIFT] key.)
3. Now type "B" but *don't* press [RETURN] yet. (Note that the cursor has moved to the next letter "R.")
4. If you have finished typing the line and everything is correct, press [RETURN]. (Note that after you press [RETURN] the blinking cursor moved to the beginning of the next line.)
5. Remember you can always retype the entire line but the [CRSR ⇦] key saves you time.

DISPLAY



*Type NEW and clear the screen ([CLR HOME]) before you start.

Some Helpful Keys and Commands to Remember

ACTION	KEY(S) TO PRESS	COMMAND OR PROGRAM STATEMENT
• Home the cursor	CLR HOME	—
• Clear screen and home cursor	SHIFT and CLR HOME	Print "♥" (see note)
• Enter data	RETURN	—
• Execute a program	R U N and RETURN	RUN
• STOP program execution	RUN STOP	STOP
• Continue program	C O N T and RETURN	CONT
• List the program	L I S T and RETURN	
• Backspace and delete	INST DEL	—
• Backspace without deleting characters	SHIFT and ⇦ CRSR ⇨	—
• Retype rest of line after correction	⇦ CRSR ⇨	—
• Reset computer (without destroying programs in memory)	RUN STOP and RESTORE	
• END or STOP program (during an input statement)	RUN STOP and RESTORE	

Note: ♥ is the symbol you get for clear screen (shift of CLR HOME key).

Learned in This Session

COMMANDS*

- CONT
- LIST
  - LIST MM
- NEW
- RUN
  - RUN MM

* Executed as soon as you type them and press RETURN

KEY WORDS**

PRINT "MESSAGE"
PRINT (SPACE)
REM
END

** Used to make statements. Statements are executed after you type RUN and press RETURN

MISCELLANEOUS

■ CURSOR

" " QUOTATION MARKS

LINE NUMBERING

- KEYBOARD LAYOUT
- VIC POWER-UP RULES

SPECIAL FUNCTION KEYS

NOTE: If you don't understand everything on this page, stop!
Go back over this session until you understand it thoroughly!
MM = Any line number (e. g., 10, 20, 30, etc.)

Assignment* 3-1

1. WRITE* A PROGRAM TO PRINT ON SEPARATE LINES
   A. Your Name
   B. Your Entire Address
   C. Your Telephone Number
2. EXPAND* YOUR PROGRAM TO INCLUDE THE FOLLOWING:
   A. Remark Statement to Describe Your Program
   B. Spacing between Each of the Lines Displayed (Printed)
   C. Include an End Statement
3. TYPE YOUR PROGRAM AND ENTER IT
4. RUN YOUR PROGRAM
5. LIST YOUR PROGRAM

* WRITE YOUR PROGRAM ON PAPER AND GET IT CHECKED BY YOUR TEACHER FIRST.

PRACTICE 3

Writing and Running Your First Program

1. Write a program to PRINT the following:
   a. Your name (first and last)
   b. Your school's name
   c. Your teacher's name
2. Enter and RUN it.

PRACTICE 4

Inserting Remarks and Spacing into Your Program

1. If you have erased the program from Practice 3, rewrite the program and do the following: (If you still have the program from Practice 3 in the computer, you do not have to rewrite the program.)
   a. Add a new program line with a remark statement to your program (any remarks you want to make).
   b. Have the computer insert one space between your name and your school's name in the output on the display (that is, you add the necessary program line).
   c. Have the computer insert two spaces between your school's name and your teacher's name in the output on the display.

PRACTICE 5

Listing and Ending Your Program

1. Rewrite the program from Practice 4 and do the following (Again, if you have the program in the computer, you don't have to rewrite it. But in case you don't know what is in the computer, just type NEW and rewrite the program.):
   a. Add an END statement to tell the computer it is the end of your program.
   b. Add a statement to have your computer PRINT "The END."
   c. RUN your program.
2. List your program.
   a. How large is your program now? (How many lines?)
   b. Copy the program in your notebook.

PART 4

More Programming Tools

What You Will Learn

1. To enter and run more BASIC programs: mathematical programs, area of rectangle program.
2. To explain the order of mathematical operations using the M.D.A.S. rule.
3. To explain the purpose and use of the keyword: LET.
4. To explain the purpose and use of the BASIC mathematic operators: multiply (*), divide (/), add (+), subtract (—), exponentiate or raising a number to a power (↑).
5. To explain the function and use of commas, semicolons, and print zones.
6. To list and identify variables that can be used with VIC BASIC.

Review of Part 3

COMMANDS*

- CONT
- LIST
  - LIST MM
- NEW
- RUN
  - RUN MM

* Executed as soon as you type them and press RETURN

KEY WORDS**

PRINT "MESSAGE"
PRINT (SPACE)
REM
END

** Used to make statements. Statements are executed after you type RUN and press RETURN

MISCELLANEOUS

■ CURSOR

" " QUOTATION MARKS

LINE NUMBERING

- KEYBOARD LAYOUT
- VIC POWER-UP RULES

SPECIAL FUNCTION KEYS

RUN STOP

CLR HOME

RETURN

SHIFT

SHIFT LOCK

INST DEL

CRSR ⇧⇩

CRSR ⇦⇨

REPEAT

NOTE: If you don't understand everything on this page, stop!
Go back over this session until you understand it thoroughly!
MM = Any line number (e. g., 1Ø, 2Ø, 3Ø, etc.)

Math Operators

= (Equal)	* (Multiply)
+ (Add)	/ (Divide)
– (Subtract)	↑ (Exponentiation)

(↑) means raising a number
to a power like 2^2, 2^3, or 2^4

Order of Arithmetic Operations

- Multiply ⟶ Divide ⟶ Add ⟶ Subtract
  (Left to Right)
  — "My Dear Aunt Sally"

- If Parentheses are used
  — Innermost level operations first
  — Then next level out
  — M.D.A.S. order inside parentheses

Order of Operations Example — (Without Parentheses)

- If there are no parentheses, the computer performs operations by going from left to right doing exponentiation operations (↑) first. Then (*) and (/) are done in order from left to right and finally (+) and (−) are done in order from left to right. (Remember M.D.A.S.!)

Example:

```
4 + 5 * 4 ↑ 3   − 4/2 =
4 + 5 * [64]    − 4/2 =
4 +   [320]     − 4/2 =
4 +   320       − [2] =
 [324]          − 2   = [322]
```

Order of Operations Example — (With Parentheses)

- **If there are parentheses, the computer starts at the inner pair of parentheses and converts everything to a single number. Then the computer repeats the process with the next pair of parentheses working "inside" out.**

Example:

((6 + 4) * 2) /4 =

([10] * 2) /4 =

[20] /4 = [5]

In-Class Exercise 4-1

You Try Some Now (Without Parentheses)

1) 2 ↑ 3 + 4 * 5 − 4/2 * 5 = _____
2) 14 − 2 * 2 + 6 − 2 * 3 * 2 = _____
3) 14/2 * 3 − 2 ↑ 3 + 4 = _____

Now try some with parentheses

1) 6 + (9 * 2) = _____
2) (6 + (9 * 2)) * 5 = _____
3) 3 * ((4 + (6 * 2)) * (9/3 − 1)) = _____

Tips on Using Parentheses — Summary

- When in doubt, use parentheses. They can't do any harm!
  — Use parentheses around operations you want performed first
- Make sure that every left parenthesis has a matching right parenthesis
  — Count them to be sure!
- Order of Operations
  — Inner most pair of parentheses first (M.D.A.S. rule inside parentheses)
  — Then work "inside" out
  — In case of a "tie," computer starts to the left and works right doing exponentiation (↑) and the M.D.A.S. rule.

Variable Names Used with VIC BASIC

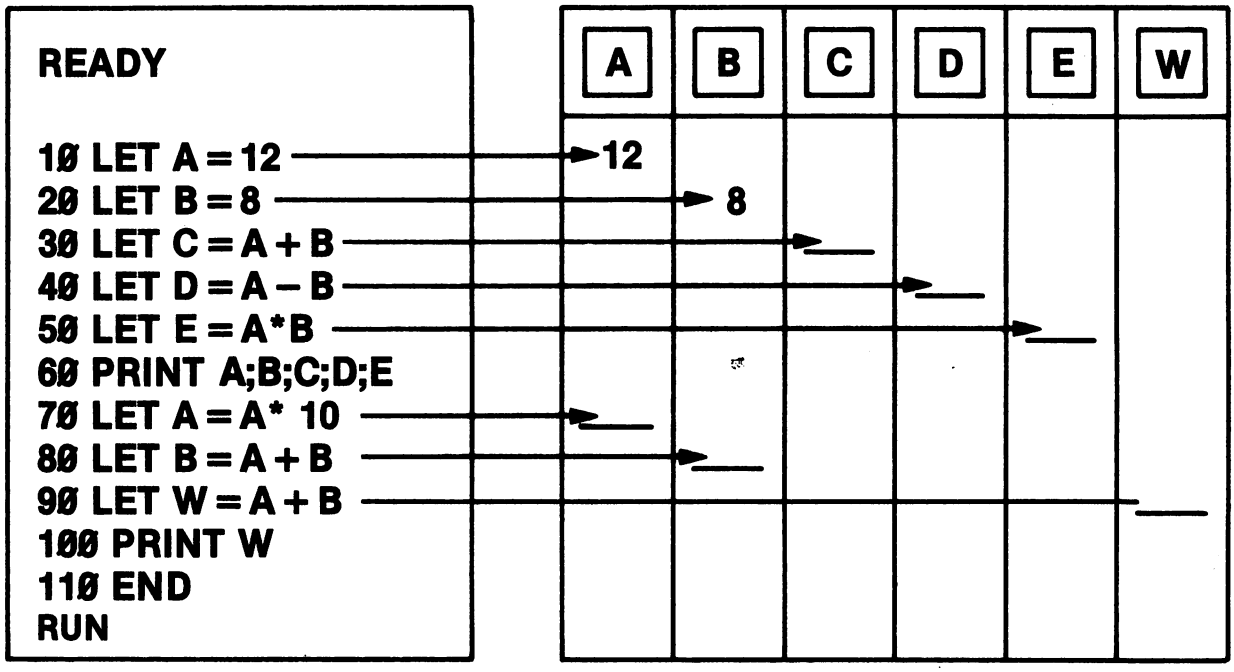
- **Must begin with a letter (A-Z)**
  - **— May be followed by another letter**

    or

  - **—May be followed by a digit (Ø-9)**
- **Some examples of variable names include:**
  - **— A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.**
  - **— A1, A2, B1, B2, C3, C5, D9, N9, P4, Q1, R6, Y7**
  - **— AA, AZ, GP, MU, ZZ, BB, XY, LL, FG, LE, RE**

    (You get the picture! Using the above combinations, you can use approximately 9ØØ variable names.)
- **There are some words with special meaning in the BASIC language and they *cannot* be used as variable names.**
  - **— The complete list of reserved words, which cannot be used in variable names, appears in the VIC Reference Manual.**

In-Class Exercise 4-2
(Assigning Numeric Values to Variables)



Basic Program for a Mathematical Operation

Line No.	Key Word[1]	Other Part of Statement	
10	LET	X = 5	RETURN
20	LET	Y = 12	RETURN
30	LET	Z = X*Y	RETURN
40	PRINT	Z	RETURN
99	END		
RUN			

(1) LET is an optional key word for VIC BASIC. Some computers require you to use LET however. Beware of this if you use another computer.

Analysis of the BASIC Program for a Mathematical Operation

Line No.	Statement	Meaning to Computer
1Ø	LET X = 5	Assign a value of 5 to variable X
2Ø	LET Y = 12	Assign a value of 12 to variable Y
3Ø	LET Z = X*Y	Take the values of X and Y, multiply them together, and assign the resulting value to the variable Z
4Ø	PRINT Z	Print the value of Z (which is 6Ø in the example)
99	END	END PROGRAM
RUN		EXECUTE PROGRAM

A BASIC Mathematical Program — Area of Rectangle

YOUR ACTION **DISPLAY**

1. **Type NEW and then clear the screen (hold down SHIFT key and then press CLR HOME).**
2. **Type and enter.** →

```
10 REM AREA OF A RECT
ANGLE PROBLEM
20 REM AREA (A) = LENG
TH  (L) X WIDTH (W)
30 LET L = 10
40 LET W = 5
50 LET A = L*W
60 PRINT A
RUN
  50
READY
■
```

3. **Type RUN and press RETURN.**

NOTE THAT WE SAID IN LINE 60 PRINT A. There were no quotes around the letter A because we wanted the computer to PRINT the *value* of A.
If we wanted the computer to PRINT the exact word or letter, we would put quotes around the word or variable.

Area of Rectangle Program Modified

YOUR ACTION	DISPLAY	
1. Add Line 70 to read then press RETURN.	70 PRINT "THE AREA =", A	
2. Type RUN and press RETURN.	THE AREA = 50	Ⓐ
3. Add Line 80 to read then press RETURN.	80 PRINT "THE AREA IS ", A	
4. Type RUN and press RETURN.	THE AREA IS 50	Ⓑ
5. Add Line 90 to read then press RETURN.	90 PRINT "THE AREA IS "; A; "SQ. IN."	
6. Type RUN and press RETURN.	THE AREA IS 50 SQ. IN. READY ■	Ⓒ & Ⓓ

Notes:

Ⓐ **Comma in Line 70 told the computer to print the answer (A) and its label (THE AREA =) on the same line. Note that the label includes 10 characters with the spaces (count them).**

Ⓑ **Commas in Line 80 told the computer to print the answer (A) and the label (THE AREA IS) on separate lines because the label exceeds 10 characters.**

Ⓒ **A semicolon automatically inserts one space between two items it is separating if the two items include a variable and a "message." (This is unique to VIC. In standard BASIC programming, a semicolon tells the computer to PRINT output close together with no spacing. So beware!)**

Ⓓ **LIST your program when you finish. RUN your program several times and note that you have printed your answer four different ways.**

Assignment* 4-1

1. **Write a Program to Find Area of a Triangle**
   A. **GIVEN: A = ½ bh WHERE b = 5, h = 1Ø**
   B. **Include Remarks Statement**
   C. **Have Program PRINT "THE AREA = " (Your Answer) "SQ. FT."**

2. **Write a Program to Find the Volume of a Rectangular Solid**
   A. **GIVEN: V = L*W*H, L = 5, W = 1Ø, H = 2**
   B. **Include Remarks Statement**
   C. **Have Program PRINT "THE VOLUME = " (Your Answer) "CUBIC IN."**

Summary — Mathematical Operations

- **LET is an optional key word when using VIC BASIC.**
  — Other computers using BASIC might require use of LET, so beware!
- **1Ø PRINT A: Tells computer to print the value of A**
  — Whereas 1Ø PRINT "A": Tells computer to print letter A (because the computer will print anything within quotes).
- **A comma in a PRINT statement tells the computer to leave several spaces between items separated by the commas.**
- **A semicolon inserts one space between two items it is separating on the same line if the two items include a variable and a "message."**

Print Zones

ZONE 1 **11 Spaces**	**ZONE 2** **11 Spaces**
***LEEDSPRIME**	***COMPUTERS***

Note:

- **Try typing in the words shown above. Count each character or symbol as you type it. Also note that there are no spaces between characters.**

Print Zones

- The VIC is divided into two PRINT zones.
  - — Each PRINT zone has 11 spaces for up to 11 characters.
  - — The VIC can display up to 22 characters per line (2 x 11 = 22).
- Commas are used to tell the computer to move to the next PRINT zone.
  - — The cursor moves to the next PRINT zone each time a comma is encountered.
  - — If the number of characters, symbols, or spaces preceding a comma on a line is greater than 11, the computer will continue printing in the next print zone. (Remember the maximum number of characters, symbols, and spaces per line cannot exceed 22.)

Print Zones and the Use of Commas

YOUR ACTION	DISPLAY	NOTES
1. Type NEW and press [RETURN].		
2. Type Line 10 to read then press [RETURN]. →	10 PRINT "ZONE 1", "ZO NE 2" ZONE 1 ZONE 2	Ⓐ Ⓑ Ⓒ
3. Type RUN and press [RETURN].		

NOTES

Ⓐ **There are two (2) 11-character PRINT zones per line (since 2 x 11 = 22, the screen can display up to 22 characters per line).**

Ⓑ **Note that there is one comma between ZONE 1 and ZONE 2.**

Ⓒ **The comma tells the computer to move to the next PRINT zone each time a comma is encountered.**

Semicolon vs. Comma

YOUR ACTION	THE DISPLAY READS:
1. Type NEW and press RETURN.	
2. Type exactly as shown then RETURN.	10 PRINT "A"; "SEMICOL ON "; "PACKS"; "ITEMS"; "CLOSE"; "TOGETHER" 20 PRINT "BUT A", "COM MA"; "LEAVES", "SPACES"
3. Type exactly as shown then RETURN.	
4. Type RUN and press RETURN.	ASEMICOLONPACKSITEMSCL OSETOGETHER BUT A COMMA LEAVES SPACES
5. Type Lines 30, 40, 50, and 60 as shown then press RETURN.	30 LET A = 5 40 LET B = 10 50 LET C = 15 60 PRINT A; B; C
6. Type RUN 30 and press RETURN.	5 10 15

Note: There is one exception to this rule for the VIC. When the semicolon is used between two variables A, B, the computer automatically inserts one space between them. This might not be true with other computers, so beware!

Use of the Semicolon — Summary

- The effect of the semicolon from computer to computer varies, but it is always true that a semicolon leaves less space between the answers or results printed than the COMMA.
- GENERAL RULE: when you want more than one item on the same line and
  - — If you want your results or output spread out, use a comma.
  - — If you want your results or output close together, use a semicolon.
- Exception

  With VIC, the semicolon normally tells the computer to print your output close together except when there is a printed message and a variable. For example, in the statement: 60 PRINT "THE AREA IS"; A; "SQ. INCHES" the output will look like this (If A = 50):

  The area is 50 sq. inches

  Notice that the computer automatically inserted one space between two items when it encountered a semicolon. This is unique to VIC.

PRACTICE 6

Area of a Rectangle Program

Part I

1. Enter and RUN this progam:

```
10 REM AREA OF A RECTA
NGLE PROGRAM
20 REM AREA (A) = LENG
TH(L)*WIDTH(W)
30 LET L = 10
40 LET W = 5
50 LET A = L*W
60 PRINT A
```

2. Add a new program line to include a label on your answer. For example, the area of the rectangle is 50 square inches.
3. Add new program lines to PRINT the following:
   a. The length of the rectangle is 10 inches.
   b. The width of the rectangle is 5 inches.

Part II

1. *Do not* type NEW.
2. Change the values of L and W in the program. (Think before you change the lines! How many lines do you have to change? Change only those lines!)

PRACTICE 7

Program Using Mathematical Operators

1. Enter and RUN the following program:

```
10 REM MATH PROBLEMS
20 LET A = 75
30 LET B = 50
40 LET C = A+B
50 PRINT C
```

2. Change the values of A and B in the program and RUN it. Fill in the results: A = _____, B = _____, C = _____.
3. Add a program line to label the answer. Example: "The sum is (your answer)."
4. Write a program to multiply (*) two numbers (any two).
5. Add the program line to PRINT: "The product of (your no.) "*" (your no.) "is" (your answer). Example: The product of 5 * 5 is 25.
6. Write a program to divide (/) two numbers (any two).
7. Add the program line to PRINT: "The quotient of" (your #) "/" (your #) is (your answer). Example: The quotient of 10/2 is 5.
8. Write a program to subtract (–) two numbers (any two).
9. Add the program line to PRINT: "The difference between "(your #) "–" (your #) is (your answer). Example: The difference between 10–5 is 5.

Additional practices for this Part will be found in the back of the book.

PART 5

Scientific Notation

What You Will Learn

To understand and use scientific notation.

Review and Feedback

The purpose of this part of the program is to evaluate students' overall performance and determine which students are having problems. The students who are having problems will be given the opportunity to review concepts they have not mastered. The review and feedback phase is divided into the following parts:

1. Exam — written/lab
2. Open discussion with students about their concerns and interests
3. Evaluation of student's performance
4. Recommendations

Scientific Notation

- Scientists often express large numbers like 186,000 and small numbers like 0.00015 as the product of two numbers. For example:

a) $186{,}000 = 1.86 \times 10^{5}$
b) $0.00015 = 1.5 \times 10^{-4}$
c) $764{,}000 = 7.64 \times 10^{5}$
d) $0.0347 = 3.47 \times 10^{-2}$
e) $5{,}000{,}000 = 5 \times 10^{6}$

Scientific Notation

Ordinary Notation		Scientific Notation		Scientific Notation in PET	Meaning
5,000,000,000	=	5 X 10^{9}	=	5E + 09	ADD 9 zeroes after 5
.000005	=	5 X 10^{-6}	=	5E − 06	Shift decimal 6 places to left
.00000005	=	5 X 10^{-8}	=	5E − 08	Shift decimal 8 places left
5 (with 15 zeroes)	=	5 X 10^{15}	=	5E + 15	ADD 15 zeroes after 5
5 (with 16 zeroes)	=	5 X 10^{16}	=	5E + 16	ADD 16 zeroes after 5

- The VIC uses scientific notation for very large and very small numbers.
- Rule 1: E + 09 means move the decimal point 9 places to the right.
- Rule 2: E — 09 means move the decimal point 9 places to the left.

Note 1: When numbers with 10 or more digits are entered, the VIC automatically converts them to scientific notation.

Note 2: When numbers are less then 0.01, the VIC automatically converts them to scientific notation.

Assignment 5-1 — (Scientific Notation)

1. Type, enter, and RUN the following program:

```
 5 PRINT "♥" (see note)
10 PRINT 5000 000 000, 0.0000005, .00000005, 50 000 000 000
15 PRINT
20 PRINT 5 000 000 000 000 000, 5 000 000 000 000 0000
          (15 zeroes)            (16 zeroes)
```

2. Experiment with scientific notation until you feel comfortable with it.

Note: Line 5 clears the screen. To clear the screen, you must hold down the [SHIFT] key and press the [CLR/HOME] key. "♥" is the symbol for clear screen (don't forget the quotes).

Review and Feedback

A. Quiz — Written/Lab

B. Open discussion with students on concerns and interest

C. Evaluation of student's performance

D. Recommendations

FEEDBACK QUESTIONNAIRE

1. Do you like working with computers? _____ (yes, no) If not, why not? _____

   _____

2. What things do you like most about computers? _____

   _____

3. What do you dislike most about computers? _____

   _____

4. If you were a design engineer and could design the computer to do anything you wanted it to, what kinds of things would you include in your design? (Use your imagination!)

   _____

   _____

5. What was the hardest thing for you to understand about the computer so far? _____
6. What was the easiest thing for you to understand? _____

   _____

7. Were you afraid or nervous when you first used the computer? _____ (yes, no)
8. Do you feel comfortable using the computer now? _____ (yes, no)
9. Would you prefer to be doing something else rather than learning about computers? _____ (yes, no) If yes, what would you like to do? _____

   _____

10. Is the teacher going too fast, too slow, or just right for you? _____
11. Do you find the lessons interesting, boring, or so-so? _____
12. If you could teach this course, what would you do to make the lessons more interesting? _____

    _____

13. Have you decided what you want to do for a vocation? _____ (yes, no)
    If yes, what? _____
14. Would you like to take additional courses to learn more about computers and programming? _____ (yes, no)
15. Do you have any additional comments? _____

    _____

    _____

    _____

    _____

    _____

    _____

PRACTICE 8

Scientific Notation

1. Convert the following to standard scientific notation (example: 5,000,000 = 5×10^6):
   a. 5,165,123
   b. .000007
   c. .00000008
   d. 6,001,255
   e. 80 000 000 000 000 000 000 (16 zeros)
   f. 8000 000 000 000 000 000 (15 zeros)
   g. 9,000,156
   h. 7,701,777
   i. 77,701,777
   j. 5612345
2. Change the above numbers to computer scientific notation used in the VIC (example: 5,000,000,000 = 5E+09).

PART 6

Relational Operators and IF-THEN/GOTO Statements

What You Will Learn

1. How computers compare (or relate) one value with another.
2. To explain the purpose and use of the six relational operators: =, >, <, <=, >=, <>.
3. To explain the purpose and use of the key words IF-THEN, GOTO.
4. To write, enter, and run programs that use IF-THEN and GOTO statements.
5. To understand and use the counting program.

Relational Operators

- Allow computer to compare one value with another.
  - — The three relational operators include

Symbol	Meaning	Examples
=	Equal	A = B
>	Greater than	A > B
<	Less than	A < B

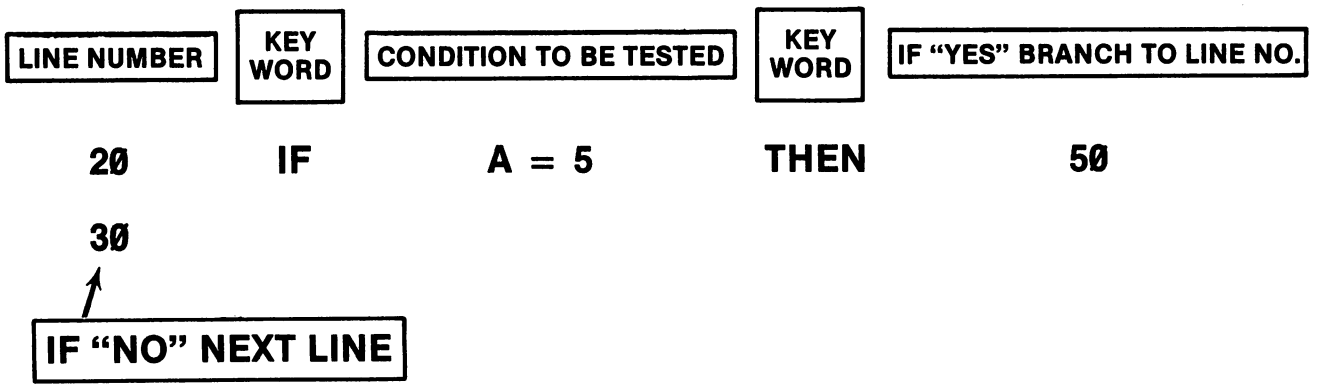
  - — Combining the three operators above we have

<>	Is not equal to	A <> B
<=	Less than or equal to	A <= B
>=	Greater than or equal to	A >= B

NOTE: To distinguish between < and >, just remember that the smaller part of the < symbol points to the smaller of two quantities being compared.

IF-THEN

- **IF-THEN is used in conditional branching.**
  - **— That is, the program will "branch" to another part of the program on the condition that it passes the test it contains.**
  - **— If the test fails, the program simply continues to the next line.**
- **Example:**



Sample Program Using IF-THEN (Conditional Branching)

- Program

```
10 LET A = 5
20 IF A = 5 THEN 50
30 PRINT "A DOES NOT EQUAL 5."
40 END
50 PRINT "A EQUALS 5."
RUN
```

- The screen should display

```
A EQUALS 5
```

- Why is Line 20 above a conditional branching statement?
  — What's the condition or test?

In-Class Exercise 6-1 (IF-THEN)

Given: A = 1Ø, B = 2Ø, C = 3Ø

Exercises:

Exercise No.	Statement	Condition is (T or F)	Branch to (Line N) Ⓐ
1.	1Ø IF A = B THEN 4Ø	F	2Ø
2.	1Ø IF A <> B THEN 5Ø		
3.	1Ø IF A > B THEN 6Ø		
4.	1Ø IF A < B THEN 7Ø		
5.	1Ø IF C < = A + B THEN 8Ø		
6.	1Ø IF C > = A + B THEN 9Ø		
7.	1Ø IF B > A THEN 1ØØ		
8.	1Ø IF B/A > = C/A THEN 11Ø		
9.	1Ø IF A * B < = A * C THEN 12Ø		
1Ø.	1Ø IF C/A < = A * B THEN 13Ø		

Ⓐ **Note: If condition is false (F), the computer will execute the next line (i.e., 2Ø).**

A Counting Program — Using IF-THEN

- **Program**

```
10 LET J = 0
20 LET J = J + 1
30 PRINT J
40 IF J < 10 THEN 20
RUN
OUTPUT IS*
```

* OUTPUT
1
2
3
4
5
6
7
8
9
10

- **In-Class Exercise 6-2**

  Modify above program to count to 50 by 5's

IF-THEN Counter Program Analysis

	PROGRAM EXECUTION	"J" COUNTER STATUS	DISPLAY
INITIALIZE	10 J = 0	[0]	
1ST TIME	20 J = J + 1	[1] = 0 + 1	
	30 PRINT J,		
	40 IF J < 4 THEN 20		
2ND TIME	20 J = J + 1	[2] = 1 + 1	
	30 PRINT J,		
	40 IF J < 4 THEN 20		
3RD TIME	20 J = J + 1	[3] = 2 + 1	
	30 PRINT J,		
4TH TIME	40 IF J < 4 THEN 20	[4] = 3 + 1	
	20 J = J + 1		
	30 PRINT J		
	40 IF J < 4 THEN 20		
END	50 END		1 2 3 4

IF-THEN COUNTER — Program Analysis (Stop-Action)

	PROGRAM EXECUTION	"J" COUNTER STATUS	DISPLAY
INITIALIZE	10 J = 0	10 [0]	
1ST TIME	20 J = J + 1	20 [1] = 0 + 1	
	30 PRINT J	30	[1]
	40 STOP		
	45 REM TYPE CONT TO CONTINUE		
	50 IF J < 4 THEN 20		
2ND TIME	20 J = J + 1	20 [2] = 1 + 1	
	30 PRINT J	30	[2]
	40 STOP		
	45 REM		
	50 IF J < 4 THEN 20		
3RD TIME	20 J = J + 1	20 [3] = 2 + 1	
	30 PRINT J	30	[3]
	40 STOP		
	45 REM		
	50 IF J < 4 THEN 20		
4TH TIME	20 J = J + 1	20 [4] = 3 + 1	
	30 PRINT J	30	[4]
	40 STOP		
	45 REM		
	50 IF J < 4 THEN 20		
END	60 END		

In-Class Exercise 6-3
(GOTO — Unconditional Branching)

- **Type and RUN this program:**

  10 PRINT "[♥]" (see note)
  20 PRINT "YOUR NAME";
  30 GOTO 20

- **What happened?**
  - **Do you know how to stop the program? (What about the [RUN/STOP] key?)**
    Explain this simple program (Line 10 merely clears the screen).
    But what does Line 30 tell the computer to do?
  - **Were there any tests or conditions to be satisfied in Line 30 before it does what it has to do?**
  - **Do you understand now why the GOTO statement is called an unconditional branching statement?**

- **Don't leave this page until you understand everything!**

Note: Line 10 clears the screen.

Exercise 6-4 (GOTO/IF-THEN)

Exercise:

- **Study the program below and write the message that would be printed if the program were executed.**

```
10 PRINT "WELCOME TO LE
EDS MIDDLE SCHOOL"
20 GOTO 70
25 PRINT
30 PRINT "HELLO SUPERSTA
R"
35 PRINT
40 PRINT "COMPUTERS ARE
MY THING"
50 GOTO 100
60 IF A = 5 THEN 90
70 PRINT "COMPUTER WOR
K SHOP"
80 GOTO 40
90 GOTO 120
100 LET A = 5
110 GOTO 60
120 PRINT "AND I'M A SU
 PERSTAR!"
130 END
140 PRINT "VIC MICROCO
 MPUTER"
150 PRINT "I CAN DO IT
 TOO"
160 PRINT "I SPEAK BASIC"
```

Assignment 6-1

1. **Write a program of your choice using conditional (IF-THEN) and unconditional (GOTO) statements.**
2. **Write a counting program.**
   — Count to 100 by 10's.

What We Have Learned — Summary

- Relational operators: =, >, <, <>, < =, > =
- IF-THEN
- GOTO (No space between GO and TO)
- Conditional Branching
  - — If condition is met, (i.e., TRUE), branch to designated line in program.
  - — If condition is not met, (i.e., FALSE), go to next line number in program.
- Unconditional branching
  - — GOTO line XX (no conditions or tests required)
  - — A GOTO statement, as the name implies, forces the computer to go to a specific statement anywhere in the program.

PRACTICE 9

Using IF-THEN

Part I.

1. Enter and RUN the following program:

```
10 LET A = 10
20 IF A = 10 THEN 50
30 PRINT "A DOES NOT E
QUAL 10"
40 END
50 PRINT "A EQUALS 10"
```

2. Change Line 10 to Let A = 5 and then RUN it.
3. Change Line 10 to Let A = 3 and then RUN it.

Part II.

1. Using this program as an example, write a new program to PRINT A EQUALS 3 and RUN it.
2. Change the values of A in Line 10 and RUN the program several times.

PRACTICE 10

Counting Program Using IF-THEN

1. Enter and RUN this program:

```
10 Let J = 0
20 Let J = J+1
30 Print J
40 If J < 10 then 20
```

2. Write a program to count from 1 to 15.
3. Write a program to count to 50 by 5's.
4. Write a program to count to 100 by 10's
5. Write a program to count from 15 to 30 and PRINT the answers in one column (vertically).
   Example: 15
   16
   17
   18
   and so forth
6. Write a program to count from 20 to 40. PRINT answers horizontally in three columns.
   Example:

```
20    21    22
23    24    25
and so forth
```

PART 7

Input Statements

What You Will Learn

1. To explain the purpose and use of key words input, input with built-in print.
2. To explain the purpose and use of a trailing semicolon on a program line.
3. To identify and use string variables A$, B$, C$, and so forth.
4. To explain the difference between numeric and string variables.
5. To write, enter, and run programs that use the concepts of this lesson.

Input Statement

STATEMENT	FUNCTION
1Ø INPUT A	• Causes the computer to stop, PRINT a ?, and wait for you to type in a number. • After you type in a value for A, the computer continues the program when you press the RETURN key.

Input Statements

YOUR ACTION

1. **Type NEW and press RETURN.**
2. **Type and enter Lines 5 & 10 as shown.**
3. **Type RUN and press RETURN.**
4. **Enter a number (e.g., type 5 and enter).**
5. **RUN this program several times to get the feel of it.**

DISPLAY

```
5 PRINT "THE NUMBER IS"
10 INPUT A
```

```
THE NUMBER IS
? ■
```

Ⓐ

```
THE NUMBER IS
? 5
READY
■
```

Ⓐ **The question mark on the screen means, "It's your turn and I'm waiting."**

Input Statements with Built-In Print

YOUR ACTION	DISPLAY
1. Add a semicolon to Line 5 of the resident program (i.e., the program now residing in the computer).	5 PRINT "THE NUMBER IS " ;
2. RUN the program again.	THE NUMBER IS? ■
3. Change Line 5 to read:	5 INPUT "THE NUMBER IS "; A READY ■
4. Delete Line 10 by typing 10 and then press RETURN.	
5. RUN the program.	THE NUMBER IS? ■
6. Hold down RUN STOP key and press RESTORE key.	READY ■

Ⓐ Note that the semicolon puts the question mark on the same line.

Ⓑ The results are exactly the same as before. But here is what was changed:
- — PRINT TO INPUT (Line 5)
- — Eliminated Line 10

Ⓒ The only way to end a program during an input statement is to hold down the RUN STOP key and then press the RESTORE key.

Input Statements —
(Area of Rectangle Program)

```
10  REM  AREA OF A RECTA
NGLE PROBLEM
20  REM  A = L * W
30  PRINT "THE LENGTH I
S"
40  INPUT L
50  PRINT "THE WIDTH IS
"
60  INPUT W
70  A = L * W
80  PRINT "THE AREA IS"
90  PRINT A
```

Area of Rectangle Problem Revisited (Using Input Statements)

YOUR ACTION

1. Type in program Lines 1Ø through 6Ø as shown.

2. Type RUN then press RETURN.
3. Type in the length (say 1Ø) and enter.
4. Type in the width and press RETURN.
5. What is your answer?

DISPLAY

```
10 REM AREA OF A RECTA
 NGLE PROBLEM
20 INPUT "THE LENGTH IS
"; L
30 INPUT "THE WIDTH IS"
; W
40 A = L * W
50 PRINT "THE AREA IS";
60 PRINT A
```

Ⓐ

```
THE LENGTH IS ■
THE LENGTH IS? 10
THE WIDTH IS ■
```

Ⓑ

Ⓐ **Note the trailing semicolon. It is used to hook Lines 5Ø and 6Ø together.**

Ⓑ **Note that the program waits for an input from the keyboard. If you don't enter a number or press RETURN, it will just stay at that line until the machine is turned off.**

Assignment 7-1

Write a simple program to do the following:
(using input statement)
a) Input your age
b) Input your zip code
c) Input your weight
d) Input your height in inches
e) PRINT each of the above with the proper labels (for example: My age is 15 or I am 15 years old).

What We Have Learned

- Trailing semicolon hooks two lines together.
- Input statements cause the computer to stop and wait for an input from the keyboard. For example:
- Input statements can have a built-in message to tell you what to input.

  — 1Ø Input "your age"; A

Numeric vs. String Variables

Numeric Variable		Declaration Character' (1)		String Variable
A	+	$	=	A$
A1	+	$	=	A1$
AB	+	$	=	AB$
AZ	+	$	=	AZ$

(1) NOTE: Simply by adding the string declaration character ($) to the numeric variable allows you to use any numeric variable as a string variable.

Example of Use of String Variables

YOUR ACTION	DISPLAY
1. Type and enter.	10 PRINT "♥" 20 INPUT "YOUR NAME IS "; A$ 30 PRINT "HELLO THERE, "; A$
2. RUN.	YOUR NAME IS ■ HELLO THERE, BILL READY ■ Ⓐ

Ⓐ **NOTE:**
It will print your name and not "BILL," unless your name is "BILL."

Caution!

If input statement does not work properly when you execute it, check to see if the following has occurred:

- **There are too many characters in the prompt (i.e., message within the quotes). This causes the cursor to drop down to next line (error condition).**

To overcome the above problem, use the print statement for the prompt on one line and then use a shorter input statement.

EXAMPLE:

VIC Error {**10 INPUT "PLEASE ENTER YOUR NAME (LAST, FIRST)"; A$**

Recommended

```
10 PRINT "PLEASE ENTER
YOUR NAME (LAST, FIRST
)"
20 INPUT "ENTER NAME";
A$
```

In-Class Exercise 7-1 (String Variables)

YOUR ACTION **DISPLAY**

1. Type and enter.

```
 5 PRINT "♥"
10 INPUT "FIRST NAME";
A $
20 INPUT "MIDDLE NAME"
; B$
30 INPUT "LAST NAME";
C$
40 PRINT A$ " " B$ " "
C$
50 INPUT "FULL NAME";
D$
60 PRINT D$
```

Ⓐ

2. RUN.
(Sample)

```
FIRST NAME? AUBREY
MIDDLE NAME? BRIGHT
LAST NAME? JONES
AUBREY BRIGHT JONES
FULL NAME? AUBREY BRIG
HT JONES
AUBREY BRIGHT JONES
```

Ⓐ **NOTES**

You can add string variables together.
You must insert a space between string variables using " " marks.
A semicolon will not cause a space to be printed.

Assignment 7-2 (String Variables)

1. Run and analyze the following program:

```
10 INPUT "YOUR NAME IS"
; A$
20 INPUT "YOUR HOUSE NU
MBER" ; A
30 INPUT "YOUR STREET N
AME" ; B$
40 INPUT "YOUR ZIP CODE
" ; B
50 PRINT A$
60 PRINT A; "  " ; B$
70 PRINT "ZIP CODE" ; B
```

2. Answer the following questions:
   a) Why were A$ and B$ (string variables) required in Lines 10 and 30?
   b) Why were quotes (" ") inserted in Line 60?
   c) Why did we use $ symbol (or string declaration character) with A and B in Lines 20 and 40?

String Variables — Summary

- String variables can be assigned to indicate letters, words, and/or combinations of letters.
- It is possible to string up to 85 characters per string variable from keyboard.
- String variables can be added together.
- Use " " marks to insert a space between string variables.

Caution!

If input statement does not work properly when you execute it, check to see if the following has occurred:

- There are too many characters in the prompt (i.e., message within the quotes). This causes the cursor to drop down to next line (error condition).

To overcome the above problem, use the print statement for the prompt on one line and then use a shorter input statement.

EXAMPLE:

VIC Error {10 INPUT "PLEASE ENTER YOUR NAME (LAST, FIRST)"; A$

Recommended

```
10 PRINT "PLEASE ENTER
YOUR NAME (LAST, FIRST
)"
20 INPUT "ENTER NAME";
A$
```

PRACTICE 11

Area of Rectangle Problem (Using INPUT Statement)

1. Enter and RUN this program:

```
10 REM  AREA OF RECTA
NGLE PROBLEM
20 INPUT "THE LENGTH I
S"; L
30 INPUT "THE WIDTH IS"
; W
40 LET A = L*W
50 PRINT "THE AREA IS"
; A
```

2. Write a new program using INPUT statements to find volume (volume = length × width × height).
3. Include a statement: The volume is _____.

PRACTICE 12

More INPUT Statement Programs

Part I.

1. Write a program using INPUT statements to change meters to centimeters (centimeters = 100 × meters).
2. Include a statement: _____ meters equals _____centimeters.

Part II.

1. Write a new program using INPUT statements to do the following:
   a. Input your age.
   b. Input your zip code.
   c. Input your weight.
   d. Input your height.
2. PRINT each with the proper labels.
   Example: My age is _____.

PRACTICE 13

String Variables

Part I.

1. Enter and RUN the following program:

```
10 INPUT "YOUR NAME IS
"; A$
20 INPUT "YOUR HOUSE N
UMBER"; A
30 INPUT "YOUR STREET
NAME"; B$
40 INPUT "YOUR ZIP COD
E" ; B
50 PRINT A$
60 PRINT A; "  "; B$
70 PRINT "ZIP CODE"; B
```

2. Answer the following questions:
   a. Why are A$ and B$ (string variables) required in Lines 10 and 30?
   b. Why were quotes (" ") inserted in Line 60?
   c. Why didn't we use $ symbol (or string declaration character) with A and B in Lines 20 and 40?

Part II.

1. Write a new program using INPUT statements, string variables, and a space between each line. PRINT all information (example: My best friend is _____) to give the following information:
   a. Your best friend.
   b. Your best subject.
   c. Your favorite food.
   d. Your favorite actor.
   e. Your favorite color.
   f. Your zodiac sign.

PART 8

Using the Calculator Mode and Sizing Memory

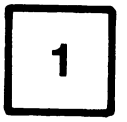
What You Will Learn

1. To define and use the terms bit, byte, k, kbytes.
2. To determine how much memory is used in a BASIC program.
3. To explain the purpose and use of the command FRE (Ø).
4. To use the VIC in calculator mode (i.e., without having to write a program).

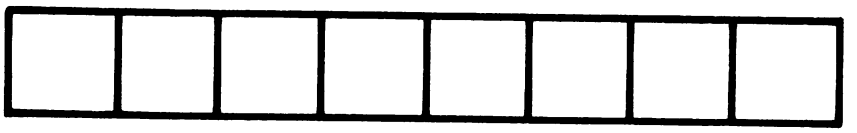
BIT = BINARY—DIGIT

BIT = SMALLEST MEMORY CELL IN A COMPUTER

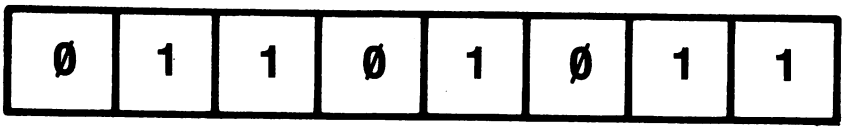
BIT = "1" OR "Ø"



MEMORY CELL WITH 1 BIT



8 MEMORY CELLS



8 BITS = 1 BYTE

BYTE = 8 BITS

K = 1ØØØ

KBYTES = 1ØØØ BYTES

KBYTES = 8ØØØ BITS

How Much Memory Is Used in BASIC Programs

WHAT'S STORED	HOW MUCH MEMORY
ALPHA CHARACTERS (A-Z)	1 BYTE
SPECIAL CHARACTERS (e.g., ", !, +, —, etc.)	1 BYTE
NUMERIC CHARACTERS (Ø-9)	1 BYTE
SPACE	1 BYTE
RETURN	1 BYTE
RESERVED WORDS (e.g., GOTO, PRINT)	1 BYTE
MEMORY OVERHEAD*	3 BYTES

EXAMPLE:

*Memory overhead refers to the way in which the computer stores information. Three additional bytes of memory are used in addition to the line number and the information written on the line.

NOTE!

The above is just an exercise to help you understand memory allocation. You don't have to count bytes to determine how much memory was used. (The next page will show you an easy way to find out how much memory is available for your use.)

The Memory Command

- **PRINT FRE (Ø)**
  - **This command is used to let you know how much memory is available to you.**
  - **Sometimes it may be important to know how much memory you are using for a given program.**
  - **If the amount of memory available in the VIC you are using is 16K, this means that there are about 16,ØØØ different memory locations to store and process your programs (actually 16,384).**
- **Note!**
  - **With no program loaded, there are less than 16,384 memory locations available for use. The difference in memory space, between actual space and 16,384, is set aside for processing programs and overall management and monitoring of what the computer is doing.**
  - **Also, your VIC might have more than 16K of memory. So make certain you know how much memory you have in your computer. (The next page will show you how to determine the amount of memory available to you.)**

Assignment 8-1

1. Determining available memory:
   a) Type NEW and press RETURN.
   b) Type PRINT FRE(Ø) and press RETURN.
   c) Display reads: _____.
   d) Now type the following and press RETURN
      1Ø PRINT "LEEDS MIDDLE
      SCHOOL."
   e) Type PRINT FRE(Ø) and press RETURN.
   f) How much space is left in memory? _____

2. Use VIC in calculator mode to solve the following:
   a) 25 * 4/2
   b) (25 + 6) − 7 + (2 * 5)
   c) 7/2 * 5 * 2 ↑ 3
   d) Any other problems you want to try

What We Have Learned

- COMPUTERS SPEAK IN MACHINE LANGUAGE
- MACHINE LANGUAGE IS A FORM OF BINARY CODING
- BINARY CODE CAN BE EITHER "Ø" OR "1" BITS
- BIT = BINARY DIGIT
- BYTE = 8 BITS
- YOU DO NOT HAVE TO KNOW MACHINE LANGUAGE TO USE COMPUTERS!

PRACTICE 14

Sizing Memory and Calculator Mode

Part I.

1. To determine available memory:
   a. Type NEW and press RETURN.
   b. Type PRINT FRE (Ø) and press RETURN.
   c. Display reads: _____.
   d. Now type the following and enter 1Ø PRINT "LEEDS MIDDLE SCHOOL."
   e. Type PRINT FRE (Ø) and press RETURN.
   f. How much space is left in memory? _____.

Part II.

1. Use VIC in calculator or immediate mode to solve the following:
   a. 25 * 4/2
   b. (25 + 6) − 7 + (2 * 5)
   c. 7/2 * 5 * 2 ↑ 3
   d. Any other problems you want to try.

PART 9

Using the Cassette Recorder

What You Will Learn

1. How to use the cassette as an output device to save information stored in memory.
2. How to use the cassette as an input device to load information from tape to memory.
3. To explain and use the commands SAVE, LOAD.
4. To make critical settings on the tape recorder and to practice using the recorder.

A Cassette Recorder Is an I/O Device

Using Cassette Tape Recorder

- The cassette tape recorder is an input/output (I/O) device that allows you to "save" information on cassette or "load" information from cassette.
  - — When you have typed a long program and wish to save it, you can save it on (SAVE) cassette.
  - — When you are ready to use it again, you can load (LOAD) it from the cassette.
  - — After you have saved your program, you should check it for recording errors. You can do this with a "VERIFY" command.
- Note! You can only save your program on cassette (not the program output).
- Refer to the VIC Reference Manual for tips on using the recorder.

SAVE Command

- Writes (outputs) a copy of the current program from memory to the tape cassette recorder.
- Format: SAVE "NAME"
  - Where "NAME" is the name given to the program by the user to distinguish the current program from other programs or data on the same tape.
  - If no name is used an unnamed program will be saved.
- Examples:

Command	*Meaning*
SAVE	Write current program on tape.
SAVE "TEST"	Write current program onto tape and assign the name TEST.
A$ = "TEST 2" SAVE A$	Write current program onto tape and assign the name given to the string variable A$, which is "TEST 2" in this example.

VERIFY Command

- Verifies that the recording made by the SAVE command is accurate. This command reads and compares the program on the tape to the program in memory without actually loading the program into memory.
  - — If the program is correct, the message OK will be displayed.
  - — If the program was recorded incorrectly the message ? VERIFY ERROR will be displayed.
  - — Always verify a program after you save it.
- Format: Verify "NAME"
  - — Verify should have same name as used with SAVE.
- Examples:

Command	*Meaning*
VERIFY	Verify next program found on tape.
Verify "TEST"	Search for the program named "TEST" on the tape cassette recorder and verify it.
A$ = "TEST 2" Verify A$	Search for a program or tape assigned the string variable A$ and verify it.

LOAD Command

- Loads (inputs) a program from tape to memory.
- Format: Load "NAME"
  - — Where "NAME" is the same name given to the program when it was saved on tape.
  - — If no name is used the first program on the tape will be loaded.
- Examples:

Command	*Meaning*
LOAD	Load the first program found on the tape.
LOAD "TEST"	Search for a program on the tape named "TEST" and load it.
A$ = "TEST 2" LOAD A$	Write Search for a program on the tape named "TEST 2" and load it.
LOAD "Z"	If a program named Z is not on tape, this command will produce a list of all the programs on tape. For example: FOUND PROGRAM "TEST" FOUND PROGRAM "TEST 2" FOUND PROGRAM "GAMES"

Saving a Program from Memory on the Tape Recorder (Using the Tape Recorder as an Output Device)

YOUR ACTION	DISPLAY
1. Type and enter program shown.	10 PRINT "THIS IS A TES T PROGRAM" 20 FOR J = 1 TO 25 30 PRINT J, : NEXT READY ■
2. Place a blank tape in recorder.	
3. Rewind tape to beginning by pressing REW lever on tape recorder.	
4. Type SAVE command as shown.	SAVE "PROGRAM 1" ■
5. Press RETURN key.	PRESS RECORD & PLAY ON TAPE *
6. Press record & PLAY levers on tape recorder simultaneously.	
7. Press STOP lever on tape recorder.	
8. Rewind tape using REW lever.	OK SAVING PROGRAM 1 READY ■

* This message will be shown only if *no* tape control keys on the tape recorder are depressed (e.g., REW, PLAY, STOP).

Verifying That a Program Saved on Tape Is Correct (Checks for Recording Errors after a Program Is Saved)

YOUR ACTION	DISPLAY
1. Make certain tape is rewound to beginning.	
2. Type the following command:	VERIFY "PROGRAM 1" ■
3. Press RETURN on keyboard.	PRESS PLAY ON TAPE *
4. Press PLAY on tape recorder.	OK SEARCHING FOR PROGRAM 1 FOUND PROGRAM 1 VERIFYING OK READY ■
5. Press STOP on tape recorder.	
6. Rewind tape and then press STOP on tape recorder.	
7. If program saved on tape is not correct the message shown will be displayed. Note! If this message is displayed, repeat the above procedure and if you still get an error message SAVE the program again.	? VERIFY ERROR

*This message will be shown only if *no* tape control keys on the tape recorder are depressed (e.g., REW, PLAY, STOP).

Loading a Program from Tape Recorder to Memory (Using the Tape Recorder as an Input Device)

YOUR ACTION	DISPLAY
1. Make certain tape is rewound to beginning, and also that you type NEW.	
2. Type the command shown.	LOAD "PROGRAM 1"
3. Press RETURN key on keyboard.	PRESS PLAY ON TAPE *
4. Press PLAY on tape recorder.	OK SEARCHING FOR PROGRAM 1 FOUND PROGRAM 1 LOADING READY ■
5. Rewind tape and then press STOP.	
6. Type the command shown.	LIST
7. Press RETURN.	10 PRINT "THIS IS A TE ST PROGRAM" ** 20 FOR J = 1 TO 25 30 PRINT J, : NEXT READY ■

* This message will be shown only if *no* tape control keys on the tape recorder are depressed (e.g., REW, PLAY, STOP).

**You can put more than one instruction on each numbered line in your program by separating them by a colon. The maximum number of characters per numbered line cannot exceed 88. (Note! Since the screen can display up to 22 characters per line, VIC will store up to 4 X 22 = 88 characters or four lines of display per each line number.)

Exercise 9-1

(Using LOAD "Z" AND SHIFT RUN/STOP KEY)

YOUR ACTION	DISPLAY
1. Insert a blank tape and then rewind it.	
2. Type NEW and then enter program shown. (Don't forget to press RETURN key to enter data.)	100 PRINT "THIS IS MY F IRST TEST PROGRAM" 199 END
3. Save the program.	
a) Type SAVE command as shown.	SAVE "PROGRAM 1" ■
b) Press RETURN key.	PRESS RECORD & PLAY ON TAPE *
c) Press RECORD and PLAY levers simultaneously on the tape recorder.	OK SAVING PROGRAM 1 READY ■
4. Verify the program.	
a) Rewind the tape.	
b) Type VERIFY command.	4b VERIFY "PROGRAM 1" ■
c) Press RETURN key.	4c PRESS PLAY ON TAPE *
d) Press PLAY on tape recorder.	4d OK SEARCHING FOR PROGRAM 1 FOUND PROGRAM 1 VERIFYING OK READY ■
e) Press STOP on tape recorder.	

***This message will be shown only if *no* tape control keys on the tape recorder are depressed (e.g., REW, PLAY, STOP).**

5. Type and enter program shown.	200 PRINT "THIS IS THE SECOND TEST PROGRAM" 299 END
6. Save Program 2.	
a) Type SAVE command as shown.	SAVE "PROGRAM 2" ■
b) Press RETURN key.	PRESS RECORD & PLAY ON TAPE *
c) Press RECORD and PLAY levers simultaneously on tape recorder.	OK SAVING PROGRAM 2 READY ■
d) Press STOP on tape recorder.	
7. Verify Program 2.	
a) Rewind tape.	
b) Type VERIFY command.	VERIFY "PROGAM 2" ■
c) Press RETURN key.	PRESS PLAY ON TAPE *
d) Press PLAY on tape recorder.	OK SEARCHING FOR PROGRAM 2 FOUND PROGRAM 1 FOUND PROGRAM 2 VERIFYING READY ■
e) Press STOP on tape recorder.	
8. Type and enter program shown.	300 PRINT "THIS IS THE THIRD TEST PROGRAM" 399 END

Exercise 9-1 (continued)

YOUR ACTION	DISPLAY
9. Save Program 3.	
a) Type SAVE command as shown.	SAVE "PROGRAM 3" ■
b) Press [RETURN] key.	PRESS RECORD & PLAY ON TAPE
c) Press RECORD and PLAY levers.	OK SAVING PROGRAM 3 READY ■
d) Press STOP.	
10. Verify Program 3.	
a) Rewind tape.	
b) Type VERIFY command.	VERIFY "PROGRAM 3" ■
c) Press [RETURN] key.	PRESS PLAY ON TAPE
d) Press PLAY on tape recorder.	OK SEARCHING FOR PROGRAM 3 FOUND PROGRAM 1 FOUND PROGRAM 2 FOUND PROGRAM 3 VERIFYING OK READY ■
e) Press STOP on tape recorder.	
11. Load "Z"	
a) Rewind tape.	
b) Type command as shown.	LOAD "Z" ■
c) Press [RETURN] key.	PRESS PLAY ON TAPE
d) Press PLAY on tape recorder.	OK SEARCHING FOR Z FOUND PROGRAM 1 FOUND PROGRAM 2 FOUND PROGRAM 3 BREAK READY ■
e) Press STOP on tape recorder and [RUN/STOP] key on keyboard (after program is found).	
12. Load and Run.	
a) Rewind tape and then press STOP.	
b) Hold down [SHIFT] key and press [RUN/STOP] key.	LOAD
c) Press PLAY on tape recorder.	PRESS PLAY ON TAPE
d) Press STOP on tape recorder.	OK SEARCHING FOUND PROGRAM 1 LOADING THIS IS TEST PROGRAM 1

***This message will be shown only if *no* tape control keys on the tape recorder are depressed (e.g., REW, PLAY, STOP).**

PRACTICE 15

Using the Computer to Solve Problems

1. Write a program to solve the following problem. Include a PRINT statement in your program to describe your answer (output).
   The total enrollment at Armstrong High School is 1,264. There are 367 freshmen, 322 sophomores, and 298 juniors. How many seniors are there?
2. Write a new program using INPUT statements to solve one of the problems.

PRACTICE 16

Finding the Average Problems

1. Write a program to solve the following problem. Include a PRINT statement in your program to describe your answer.
   The weights of three boys are 140 lb, 150 lb, and 130 lb. What is their average weight?
2. Write a new program using INPUT statements to solve the same problem. (That is, you should use the INPUT statement for the weight of the three boys.)

PRACTICE 17

Using the Computer to Solve Problems

1. Write two programs to solve the following problems. Label your answers.
2. Over a period of six years Mr. Smith drove his car 53,862 miles. What was the average distance each year?
3. After 12 dozen bulbs were sold, how many of the 1,000 bulbs were left?

PART 10

Using FOR-NEXT-STEP Statements

What You Will Learn

1. To explain the purpose and use of key words FOR-NEXT, STEP.
2. To explain the purpose and use of the terms increment, decrement, initialize.
3. To compare key words GOTO, IF-THEN, FOR-NEXT and explain how they relate to one another.
4. To explain the purpose and use of timer loops.

For-Next Statement

- Allows the computer to do the same thing over and over a large number of times (and do it very fast!)

FOR - NEXT Loop

YOUR ACTION	DISPLAY
1. Type and enter program as shown.	5 PRINT "♥" 10 FOR J = 1 TO 10 20 PRINT " AUBREY" ; J 30 NEXT J
2. Type RUN and press RETURN.	AUBREY 1 AUBREY 2 AUBREY 3 AUBREY 4 AUBREY 5 AUBREY 6 AUBREY 7 AUBREY 8 AUBREY 9 AUBREY 10

FOR-NEXT-STEP Loop

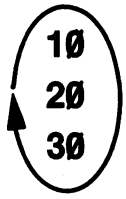
YOUR ACTION	DISPLAY
1. Retype and enter Line 10 of resident* program as shown. →	10 FOR J = 1 TO 10 STE P 3 Ⓐ
2. Type RUN and press RETURN.	AUBREY 1 AUBREY 4 AUBREY 7 AUBREY 10

*Resident means program currently in memory.

Ⓐ If step is not included in the statement, an increment of 1 is assigned by the computer (i.e., step 1).

Example of Program Statements Using Key Words

FOR-NEXT-STEP



```
10 FOR J = 10 TO 1 STEP -1
20 PRINT J ;
30 NEXT J
```

```
RUN
DISPLAY READS:
10 9 8 7 6 5 4 3 2 1
```

Analysis of FOR-NEXT-STEP Statements

LINE NO.	KEY WORD	COUNTER VARIABLE		INITIAL VALUE		FINAL VALUE	INCREMENT/ DECREMENT
1Ø	FOR	J	=	1Ø	TO	1	STEP -1
2Ø	PRINT	J					
3Ø	NEXT	J					

The FOR-NEXT-STEP loop works as follows: The first time the FOR statement is executed, the counter is set for the initial value "1Ø." Then it executes Line 2Ø (PRINT J). When the program reaches Line 3Ø (NEXT J), the counter is decremented by the amount specified (Step-1). If this step has a positive value, the counter is incremented by the amount specified (e.g., Step 2 means increment by 2's).

Comparison of GOTO, IF-THEN, and FOR-NEXT Program Loops

A.

GOTO

(Unconditional Loop)

```
 5 PRINT "♥"
10 PRINT "AUBREY"
20 GOTO 10
RUN
```

- Program loops one zillion times!
  (or until you stop it)

B.

IF-THEN

(Conditional Loop)

```
 5 PRINT "♥"
10 LET J = 0
20 J = J + 1
30 IF J>6 THEN 99
40 PRINT "AUBREY" ; J
50 GOTO 20
99 END
RUN
```

- This program loops 6 times!

C.

FOR-NEXT

(Conditional Loop)

```
 5 PRINT "♥"
10 FOR J = 1 TO 6
20 PRINT = "AUBREY" ; J
30 NEXT J
99 END
RUN
```

- This program loops 6 times!

Comparison of GOTO, IF-THEN, and FOR-NEXT Program Loops

A. "DUMB LOOP"

```
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
```

B. "SMART LOOP"

```
AUBREY 1
AUBREY 2
AUBREY 3
AUBREY 4
AUBREY 5
AUBREY 6
```

C. "SMART LOOP"

```
AUBREY 1
AUBREY 2
AUBREY 3
AUBREY 4
AUBREY 5
AUBREY 6
```

NOTE: Press RUN/STOP Key to Get Out of Loop.

FOR-NEXT SUMMARY

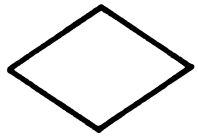
- FOR– NEXT – STEP

— FOR – NEXT is always used as a pair.
— If the key word "step" is not used, the increment of 1 is assumed.
— If the step has a negative value, the counter is decremented (e.g., for J = 1Ø to 1 step -1).
— If the step has a positive value, the counter is incremented (e.g., for J = 4 to 1Ø step 2).

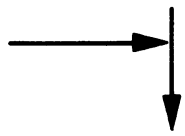
Flowchart Symbols

- Begin or End

- Processing Block



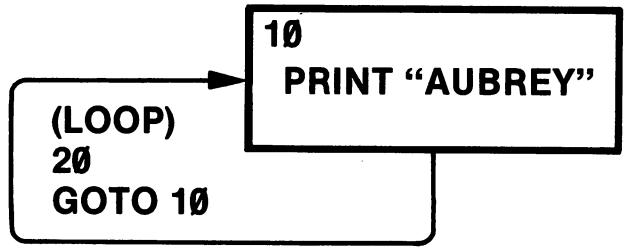
- Decision Diamond



- Connector Arrows

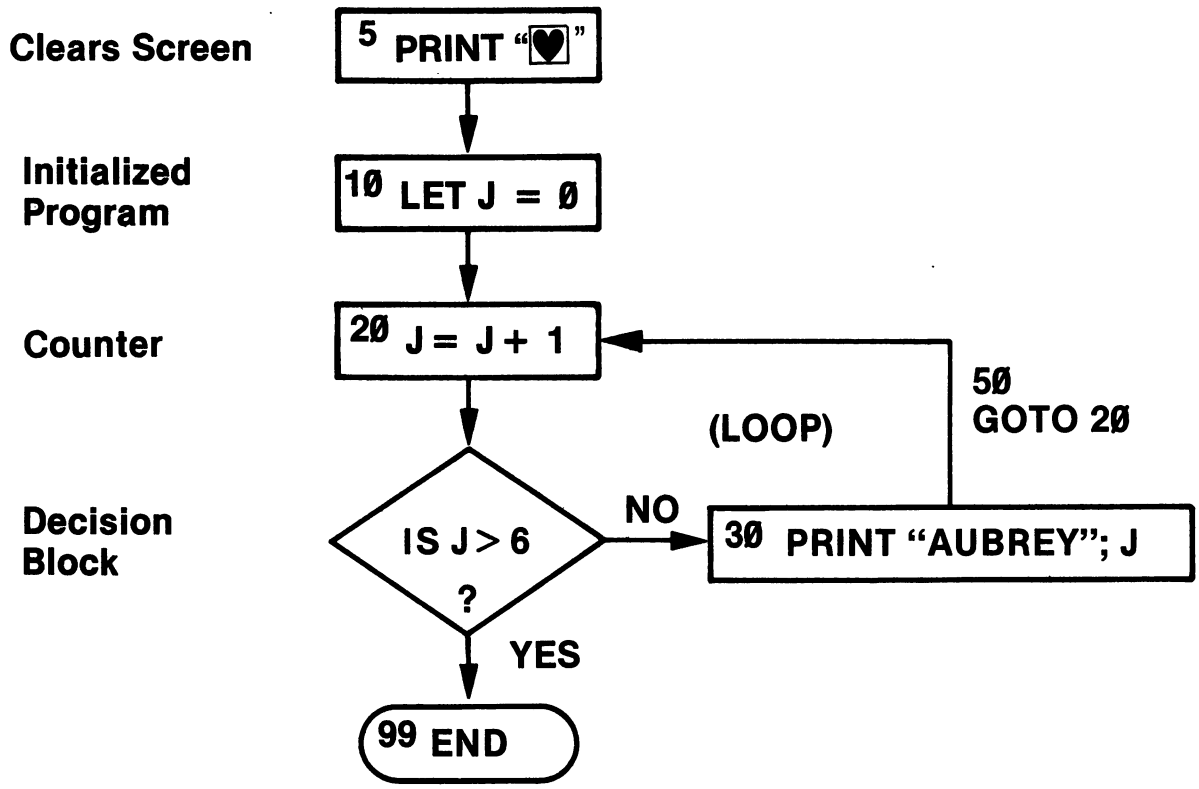
GOTO-LOOP

(Unconditional)

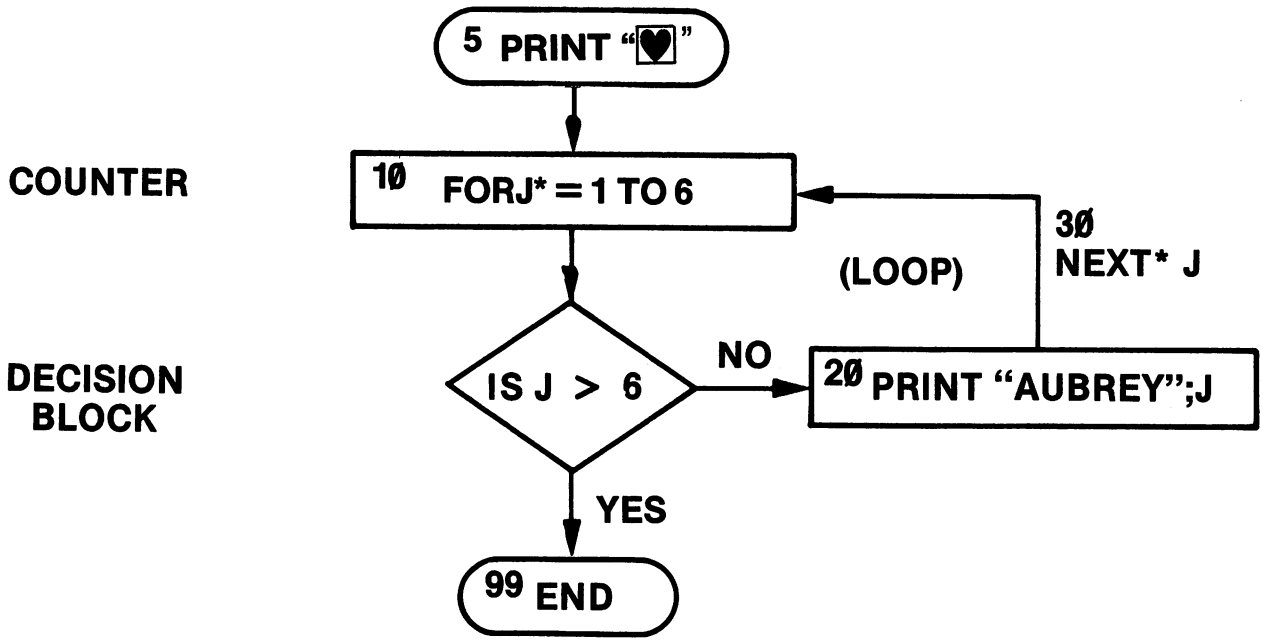


Looping with IF-THEN

FUNCTION



Looping with FOR-NEXT



*FOR-NEXT *Work together as a counter

Timer Loop

- The VIC can do approximately 700 FOR-NEXT loops per second.
- Example

```
  5 REM 10 Second Timer
Program
10 PRINT "Timer Progra
m Counting"
20 FOR X = 1 TO 7000
30 NEXT X
40 PRINT "Timer Progra
m Ended"
```

- You don't believe the VIC can count? Well, try it! (Type in the above program and RUN.) — Don't forget to use your watch!

Assignment 10-1

1. Type, enter, and RUN the following program.

```
 5 PRINT "♥"
10 PRINT "INPUT A VALUE
N" : PRINT:PRINT
15 INPUT "ENTER 1500 or
7500"; N
20 PRINT "♥"
25 PRINT "THIS IS A DEM
ONSTRATION OF"
30 PRINT:PRINT
35 FOR J=1 TO N: NEXT
40 PRINT "USING A FOR-N
EXT TIMER LOOP"
45 PRINT:PRINT:PRINT:P
RINT
50 FOR J=1 TO N: NEXT
60 PRINT "IF YOU WISH T
O CHANGE THE DISPLAY'S
SPEED"
65 PRINT:PRINT
70 FOR J=1 TO N: NEXT
80 PRINT "CHANGE THE VA
LUES OF N IN THE FOR-N
EXT LOOP"
85 PRINT:PRINT:PRINT:P
RINT
90 FOR J=1 TO N: NEXT
100 PRINT "IF YOU WISH
 TO STOP THIS DISPLAY"
105 PRINT:PRINT
110 FOR J=1 TO N: NEXT
120 PRINT "PRESS THE 'ST
 OP' KEY"
130 FOR J=1 TO N: NEXT
140 GOTO 20
```

2. Make certain that you understand this program and can explain it to your teacher.

PRACTICE 18

Counting Programs Using IF-THEN and FOR-NEXT

1. Using IF-THEN, write a program to count by 5's from 5Ø to 5.
   a. Written vertically
   b. Written horizontally
2. *Do not* type NEW (that is, save the program above).
3. Using FOR-NEXT, write a program to count to 5Ø by 5's written horizontally.
   Note: Start your second program at Line 1ØØ. That is, type Line 1ØØ as follows: 1ØØ PRINT : PRINT (Of course, this is to insert two spaces between your outputs.)
4. How many program lines (excluding Line 1ØØ) did it take using FOR-NEXT? _____
   How many using IF-THEN? _____
5. What can you conclude from this task?

PRACTICE 19

Using IF-THEN and FOR-NEXT Statements

1. Using IF-THEN, write a program to generate all the even numbers between 11 and 51 from smallest to the largest (that is, 12, 14, 16, and so forth).
2. *Do not* type NEW.
3. Using FOR-NEXT, write a program that generates the same numbers and PRINT them horizontally. (*Note:* Start at Line 1ØØ. Type Line 1ØØ as → 1ØØ PRINT : PRINT and your next line should be 11Ø.)
4. Type NEW and enter.
5. Using IF-THEN, write a program to generate all even numbers between 11 and 51 from largest to the smallest.
6. Do the same using FOR-NEXT.

PART 11

Reading Data

What You Will Learn

1. To explain the purpose and use of the key words READ, DATA, RESTORE.
2. To compare the three different ways you have learned to input data into the VIC.
3. To write, enter, and run programs using READ-DATA and READ-RESTORE key words.

READ-DATA

READ-DATA statements are much more efficient than INPUT or LET statements when you have lots of data to input.

Ways of Inputting Data to the Computer (i.e., Ways We've Learned So Far)

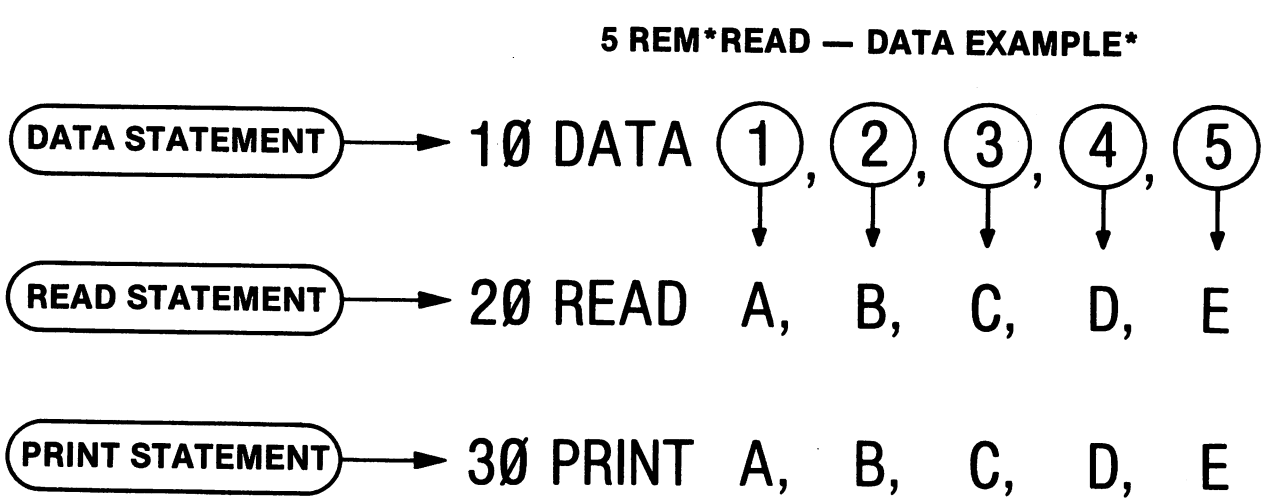
10 LET A = 5	10 INPUT A	10 DATA (5) 20 READ A
BUILT-IN	FROM KEYBOARD	READ-DATA COMBINATION

Ways of Inputting Data to the Computer

STATEMENT	FUNCTION
• 1Ø LET A = 5	• LET statement builds value into the program.
OR	
• 1Ø INPUT A	• INPUT statement allows you to enter data through the keyboard.
OR	
• 1Ø DATA(5)	• DATA statement contains the value (5), which will be stored in a specified variable.
2Ø READ A	• READ statement names the variables in which the values are to be stored.

NOTES: Data lines can be read only by READ statements.
The READ-DATA work together to input data to the computer.

Read-Data Example



NOTES:

- **Each piece of data must be read by a READ statement.**
- **Each READ statement can read a number of pieces of data if each variable is separated by a comma.**
- **Data lines can only be used by READ statements.**

Exercise 11-1 (Reading Data)

Type and enter.

```
10 DATA 1, 2, 3, 4, 5
20 READ A, B, C, D, E
30 PRINT A, B, C, D, E
```

Type RUN and press RETURN.

```
1       2
3       4
5
```

NOTES:

- **The display shows that all five pieces of data in Line 10 were read by Line 20, assigned letters A through E, and printed by Line 30.**
- **Data lines are always read left to right by READ statements.**

Read-Data Summary (Key Words)

DATA

- **Key word that lets you store data inside your program to be accessed (read) by READ statements.**
  - **— Data items will be read sequentially starting with the first item in the first DATA statement and ending with the last item in the last DATA statement.**
  - **— Items in data list may be string or numeric constants.**
  - **— If string values include leading blanks, colons, or commas, you must enclose these values in quotes.**
  - **— DATA statements must match up with the variable types in the corresponding READ statement.**
  - **— DATA statements may appear anywhere it is convenient in a program.**

- **EXAMPLE:**

```
10 DATA "JONES, A.B.",
"SMITH, R.J." (Quotes used here because of commas)
20 DATA LEEDS MIDDLE S
CHOOL, COMPUTERS
30 DATA 125, 250, 750,
1000
```

Read-Data Summary (Key Words)

READ

- **Key word that instructs the computer to read a value from a DATA statement and assign that value to the specified variable.**
  - **— The first time a READ statement is executed, the first value in the first DATA statement is used; the second time, the second value in the DATA statement is used. When all the items in the first DATA statement are used (READ), the next READ will use the first value in the second DATA statement, and so on.**
  - **— An out-of-data error (OD) occurs if there are more attempts to READ than there are data items.**

- **EXAMPLE:**

  40 READ A$, B$, C$, D$, A, B, C, D

  (Note that there are eight read variables and eight data items on previous slide for program lines 10, 20, and 30)

Assignment 11-1

1. Type and enter the following program:

```
10 PRINT "NAME", "GRAD
E
20 READ A$
30 IF A$ = "END" THEN
PRINT "END OF LIST": E
ND
40 READ G
50 IF G < 75 THEN PRIN
T A$, G
60 GOTO 20
70 DATA "GRAY, BILL",
65, "JONES, A.B.", 95
80 DATA "JONES, A.C.",
100, "SMITH, R.L.", 70
90 DATA "EPPS, S.W.",
60, "WELLS, DAVE", 100
, END
```

2. Predict the output of the program.
3. Why were quotes used in the DATA statements?
4. RUN the program and record the results.
5. Why were two commas used in Line 10?

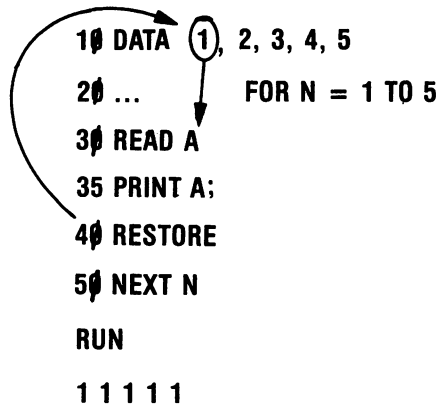
Restore

- Key word that causes the next READ statement executed to start over with the first DATA statement.
  - — This lets your program reuse the same data lines.
  - — Sometimes it is necessary to READ the same data more than once without having to run the complete program again; therefore, RESTORE is used.
  - — Whenever the program comes to RESTORE, all data lines are restored to their original unread condition, both those lines that have been READ and those that have not been READ. This allows all data to be available for reading again, starting with the first data item in the first data line.

NOTE! Remember that each piece of data in a data line can only be read once each time the program is RUN. The next time a READ statement requests a piece of data, it will READ the next piece of data in the data line, or, if data on that line are all used up, it will go to the next data line and start reading it. Therefore, the RESTORE statement is needed if the same data is to be READ more than once in the same program.

CAUTION: Do not confuse the keyword "RESTORE", as used here, with the RESTORE key. Each performs an entirely different function.

Illustration of the READ-RESTORE Feature



NOTE:

- **RESTORE caused data Line 1Ø to be restored to its original unread condition, making all data available for reading again.**
- **Since there is only one read variable, A, it starts with the first piece of data, 1, in this case.**

Exercise 11-2 (READ-RESTORE Data in a FOR-NEXT Loop)

YOUR ACTION	DISPLAY	
1. Type and enter.	10 DATA 1, 2, 3, 4, 5 20 FOR N = 1 TO 5 30 READ A 40 PRINT A ; 50 NEXT N	
2. Type RUN and press RETURN.	1 2 3 4 5	
3. Insert Line 35. (Type and enter)	35 RESTORE	Restores Data Line to Its Original Unread Condition
4. Type RUN and press RETURN.	1 1 1 1 1	Therefore Computer Reads First Data Item Over and Over

READ-DATA SUMMARY

- **READ-DATA**
  - **— Key words used to input lots of data to the computer.**
- **RESTORE**
  - **— Key word used to restore (put back) data so it can be used again.**
- **Data lines can be read only by READ statements.**
  - **— If more than one piece of data is placed on a data line, they must be separated by commas. Each piece of data must be read by a READ statement.**
- **Data lines are read from left to right by READ statements.**
  - **— Data lines can be placed anywhere in a program.**
- **READ-DATA statements are extremely common.**
  - **— RESTORE is used less often.**

PRACTICE 20

READ-DATA

1. Type and enter the following program:

```
 5 PRINT "♥"
10 PRINT "NAME", "GRAD
E"
20 READ A$
30 IF A$ = "END" THEN
PRINT "END OF LIST":E
ND
40 READ G
50 IF G > 75 THEN PRIN
T A$, G
60 GOTO 20
70 DATA "GRAY, BILL",
65, "JONES, A.B.", 95
80 DATA "JONES, A.C.",
100,"SMITH, R.L.", 70
90 DATA "EPPS, S.W.",
60, "WELLS, DAVE", 100
, END
```

2. Predict the output of the program.
3. Why were quotes used in the data statements?
4. RUN the program and record the results.

PART 12

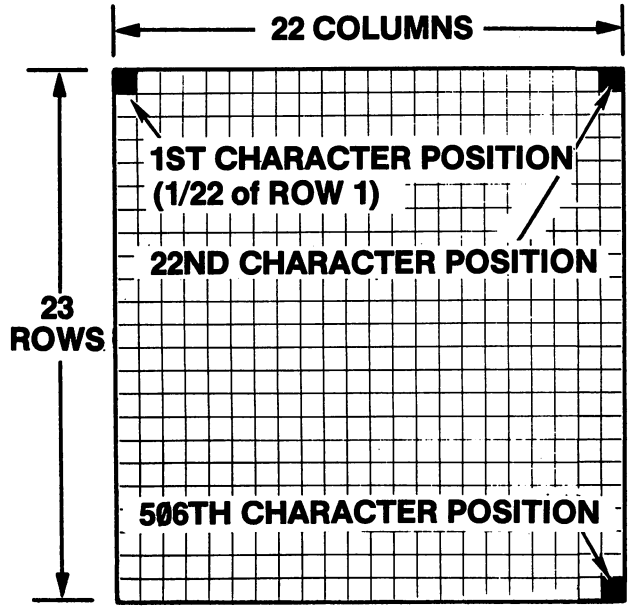
Video Display Graphics

What You Will Learn

1. To become familiar with the layout of VIC's graphic display and to learn how to use the graphic keys on the keyboard to draw pictures and letters on the screen.
2. To understand and use SPC (N) and TAB (N) to format outputs.
3. To write and run programs using all the concepts learned in this lesson.

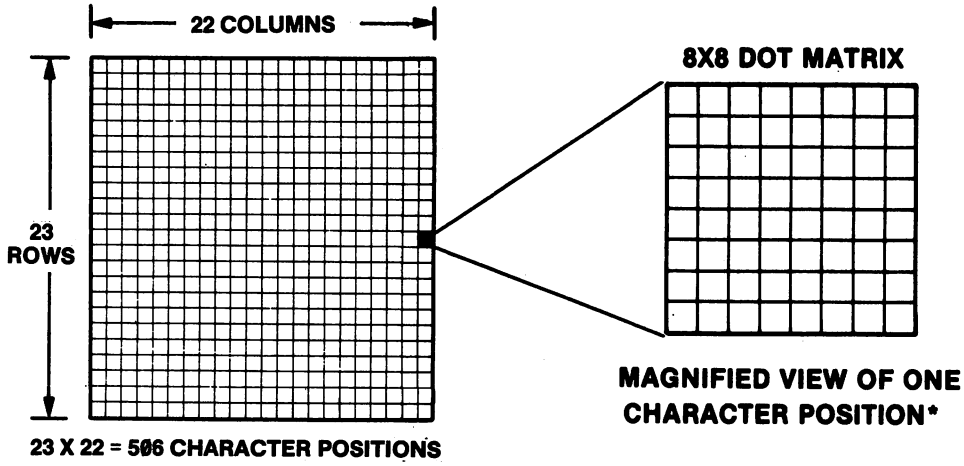
Note! The VIC graphics might appear to be very complicated initially. If you follow the examples and do the exercises in this section, you will have the necessary fundamentals to do graphics on your VIC. Practice is the key to success with graphics on your VIC. Experiment with the keys to see what graphic symbols appear on the screen. Try to sketch some simple figures with your VIC.

Video Display Layout



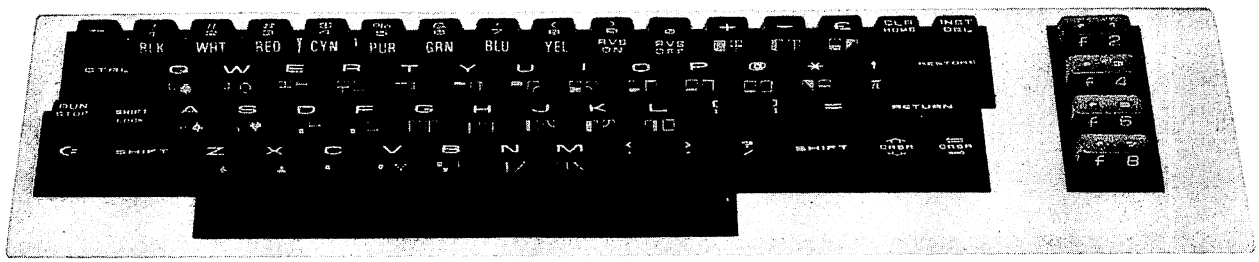
- The display has 5Ø6 character positions arranged as 23 rows with 22 characters per row (23 X 22 = 5Ø6 characters)
  - — One-twenty-second of each display line is a character position
  - — Each character position is an 8X8 dot matrix (dot block) which is used to make characters (see next slide)
  - —Although one line on the display is only 22 characters long, up to 88 characters can be handled logically. The VIC uses the "wrap around" feature or uses four rows to display one line. (This, of course, assumes you did not press the RETURN key at the end of the line.)

Illustration Of Dot Matrix For One Character Position



*There are 64 dots (maximum) available in a single character position.
That is, one character position = 8X8 = 64 dots (maximum) are available for generating a single character.

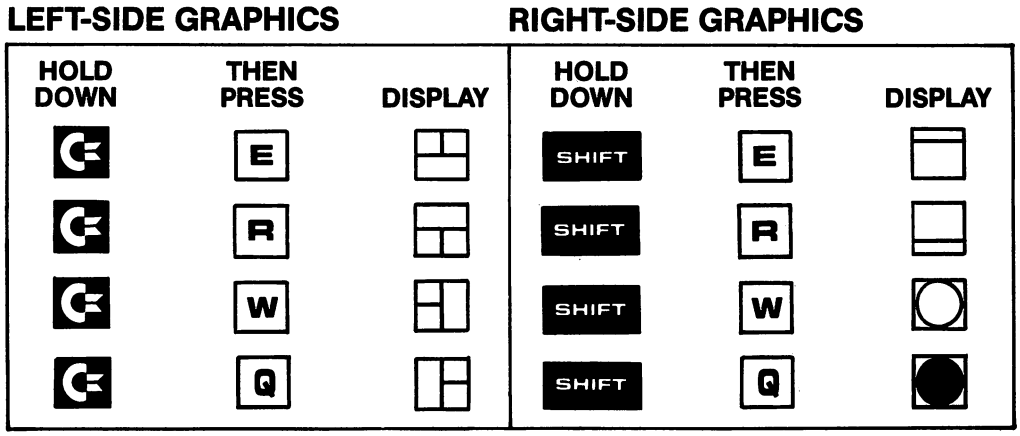
VIC KEYBOARD



Courtesy of Commodore Business Machines, Inc.

Graphic Keys

- On page 166 is a picture of the VIC keyboard showing all of the keys. The graphic characters are shown on the front of the keys. Most of the graphic keys have two graphic characters printed on the front. The text key characters (i.e., standard typewriter characters) are shown on top of the graphic keys.
- There are 62 graphic characters located on 31 keys.
- To display the graphic characters on the left side of the keys, you must hold down the Commodore key C= while pressing the desired graphics key.
- To display the graphic characters on the right side of the key, you must hold down the SHIFT key while pressing the desired graphics key.
- Here are some examples:



- Since there are so many graphic keys, you must understand the function of each key to draw pictures or to create sophisticated graphic displays.
  - — An illustration of each of the graphic characters is given on the following pages to assist you in understanding how to use graphics on the VIC.
  - — For ease of identification each of the graphic characters is listed by the alphanumeric or special symbols that appear on top of the graphic keys.
  - — The graphic keys are grouped by similarity of line segments so that you can get a feel for how the graphic symbols are developed.

Graphic Characters Reference Charts

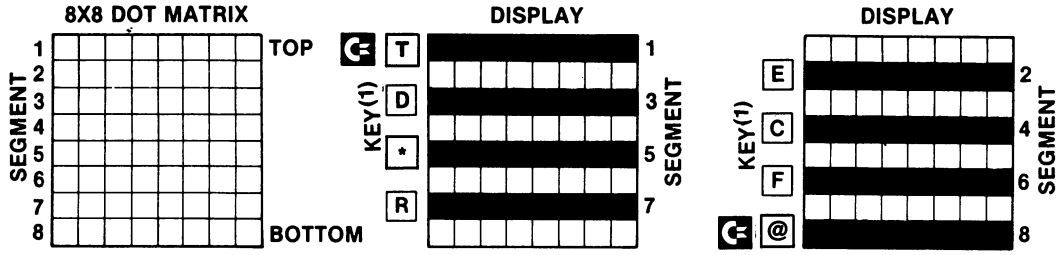
- These charts provide magnified versions of the graphic symbols for ease of use and understanding. It is important to note that the lines as displayed on these charts are much thicker than they would appear on the VIC display Also note that:
  - — Symbols are grouped by similarity of line segments; the charts are labeled A through G. (Refer to pages 170-176.)
  - — Only one (1) of the eight (8) possible horizontal line segments can be displayed in a single character position at a time. For example, if you press shifted [D] key the VIC will display a horizontal line in one character position. Then, if you press shifted [E] key the VIC will display this horizontal line segment in the adjacent character position. (Try this yourself. Press the shifted [D] key and then press the shifted [E] key.)
  - — The same is true for vertical line segments.
  - — The 8 × 8 dot matrix is shown on some charts with the line segments numbered from top to bottom or right to left for reference purposes only.
  - — The square enclosing the graphics symbol is not part of the symbol; it is used to show the boundaries of the symbol.
  - — All graphic keys shown must be used in the shifted mode or Commodore [C] mode.

Graphic Characters Reference Charts

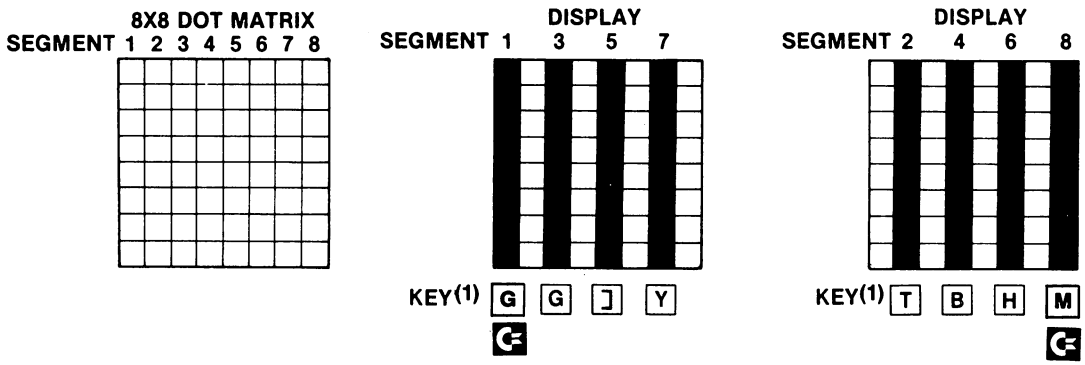
Chart	*Graphic Character Group*
A	Horizontal and vertical line segments
B	Horizontal and vertical bars
C	Square corners and rounded corners
D	T symbols and grids
E	Quarter blocks — solid and open
F	Triangles, diagonals, circles, cross, and "X"
G	Card suit

Graphic Characters Reference Chart A

Horizontal Line Segments



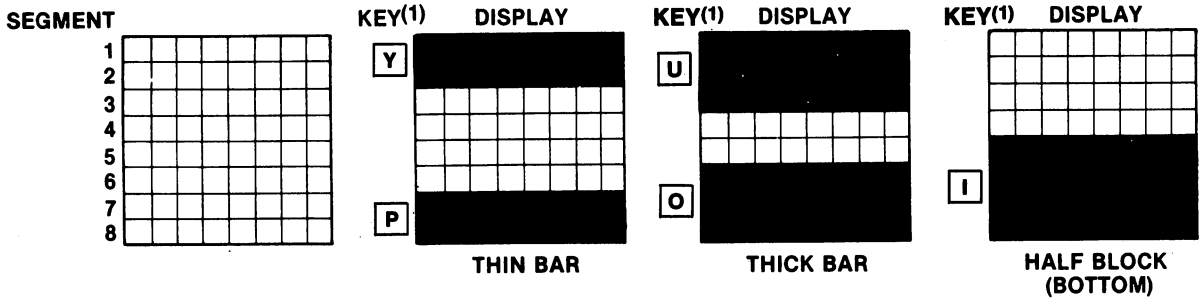
Vertical Line Segments



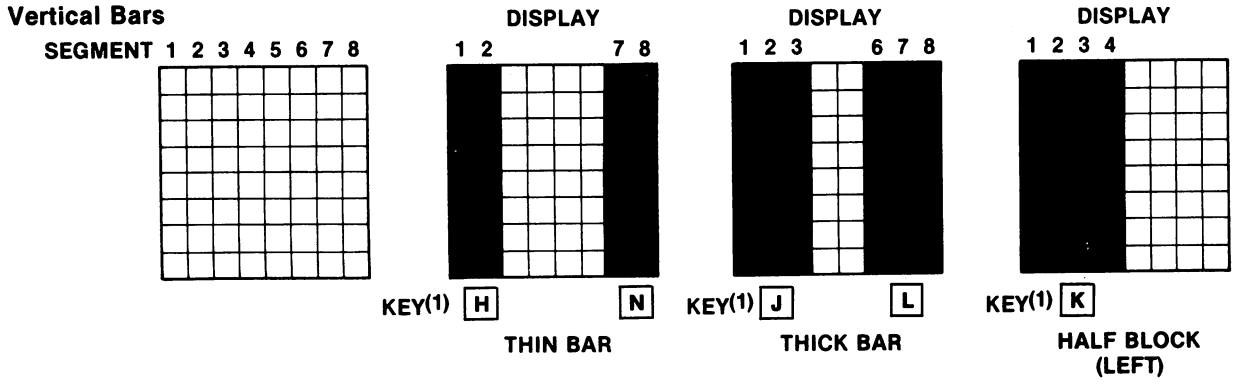
(1) Use SHIFT key with the desired key except where G key is shown. Note! All of the graphic characters shown above will not be displayed in the same character position at the same time. You can select one key at a time. For example, if you press T key together with G key the computer will display a horizontal line in one character position; then if you press E together with SHIFT key the line will be in the adjacent character position.

Graphic Characters Reference Chart B

Horizontal Bars



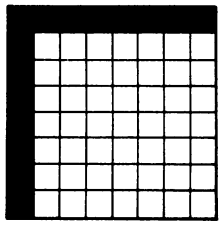
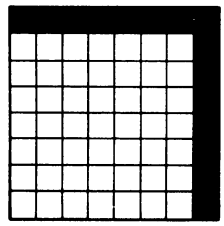
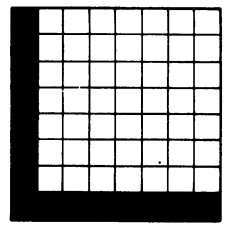
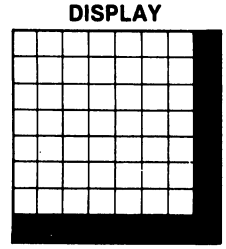
Vertical Bars



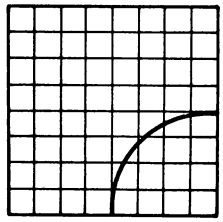
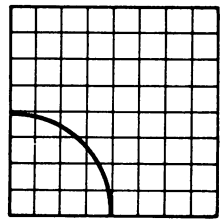
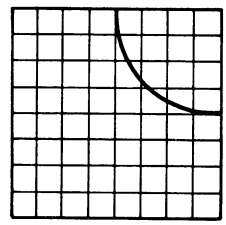
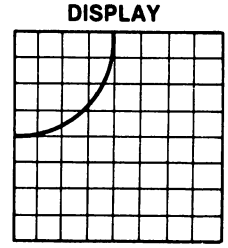
(1) LEFT-SIDE GRAPHICS (HOLD DOWN C= KEY WHILE PRESSING THE KEY SHOWN TO GET THE DESIRED GRAPHIC CHARACTER).

Graphic Characters Reference Chart C

Square Corners

DISPLAY	DISPLAY	DISPLAY	DISPLAY
			
KEY(1) O	P	L	@

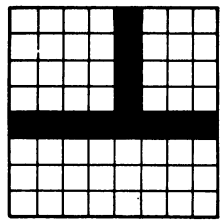
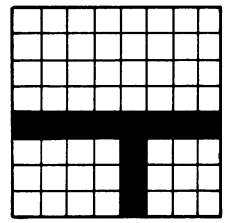
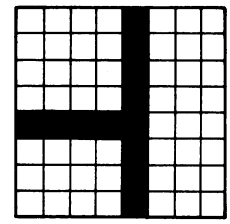
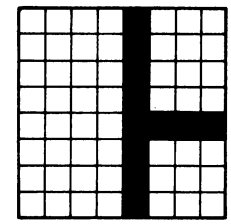
Rounded Corners

DISPLAY	DISPLAY	DISPLAY	DISPLAY
			
KEY(1) U	I	J	K

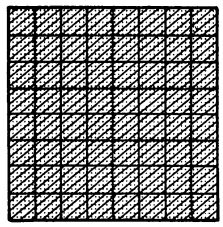
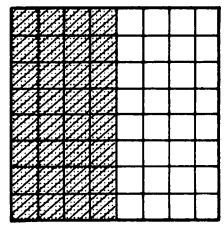
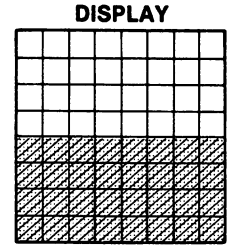
(1) RIGHT-SIDE GRAPHICS (HOLD DOWN SHIFT KEY WHILE PRESSING THE KEY SHOWN).

Graphic Characters Reference Chart D

T Symbols

DISPLAY	DISPLAY	DISPLAY	DISPLAY
			
KEY(1) [E]	[R]	[W]	[Q]

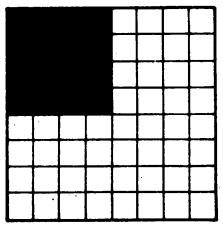
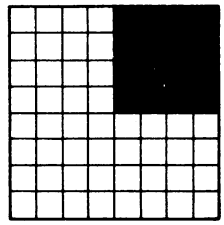
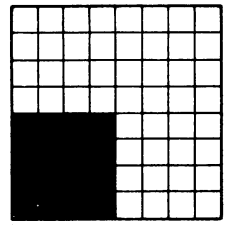
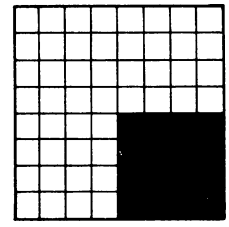
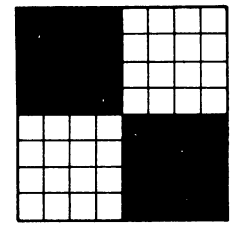
Grids

DISPLAY	DISPLAY	DISPLAY
		
KEY(1) [+]	[−]	[£]

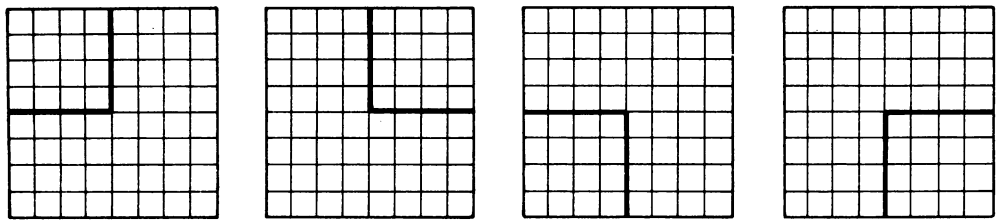
(1) LEFT-SIDE GRAPHICS (USE C= KEY).

Graphic Reference Charts E

Quarter Blocks (Solid)

DISPLAY	DISPLAY	DISPLAY	DISPLAY	DISPLAY
				
KEY(1) [V]	[C]	[F]	[D]	[B]

Quarter Blocks (Open)



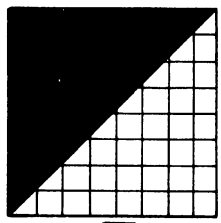
KEY(1) [X] [Z] [S] [A]

(1) LEFT-SIDE GRAPHICS (USE C= KEY).

Graphic Characters Reference Chart F

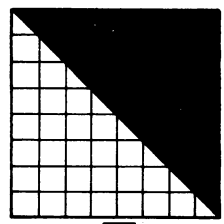
Triangles and Diagonals

DISPLAY



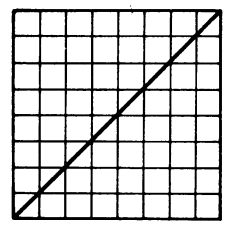
KEY(1) [£]

DISPLAY



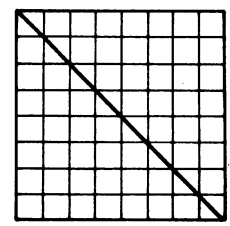
KEY(2) [*]

DISPLAY



KEY(1) [N]

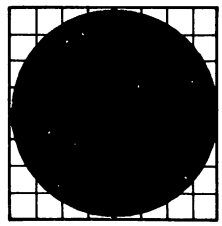
DISPLAY



KEY(1) [M]

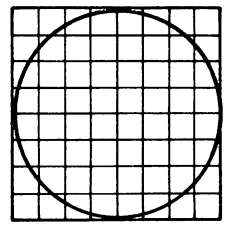
Circles, X and Cross

DISPLAY



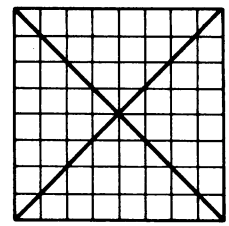
KEY(1) [Q]

DISPLAY



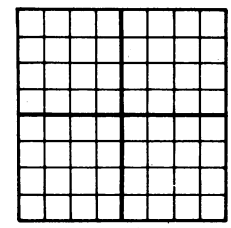
KEY(1) [W]

DISPLAY



KEY(1) [V]

DISPLAY



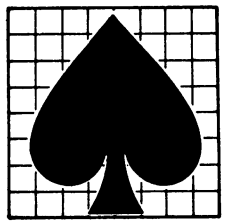
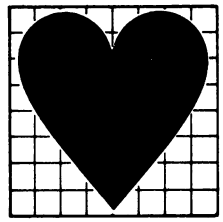
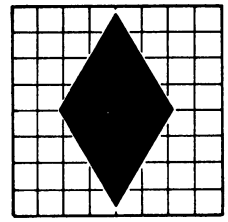
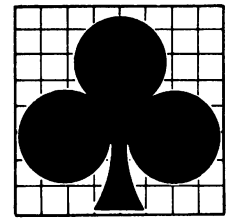
KEY(1) [+]

(1) RIGHT-SIDE GRAPHICS (USE [SHIFT] KEY).

(2) LEFT-SIDE GRAPHICS (USE [C=] KEY)

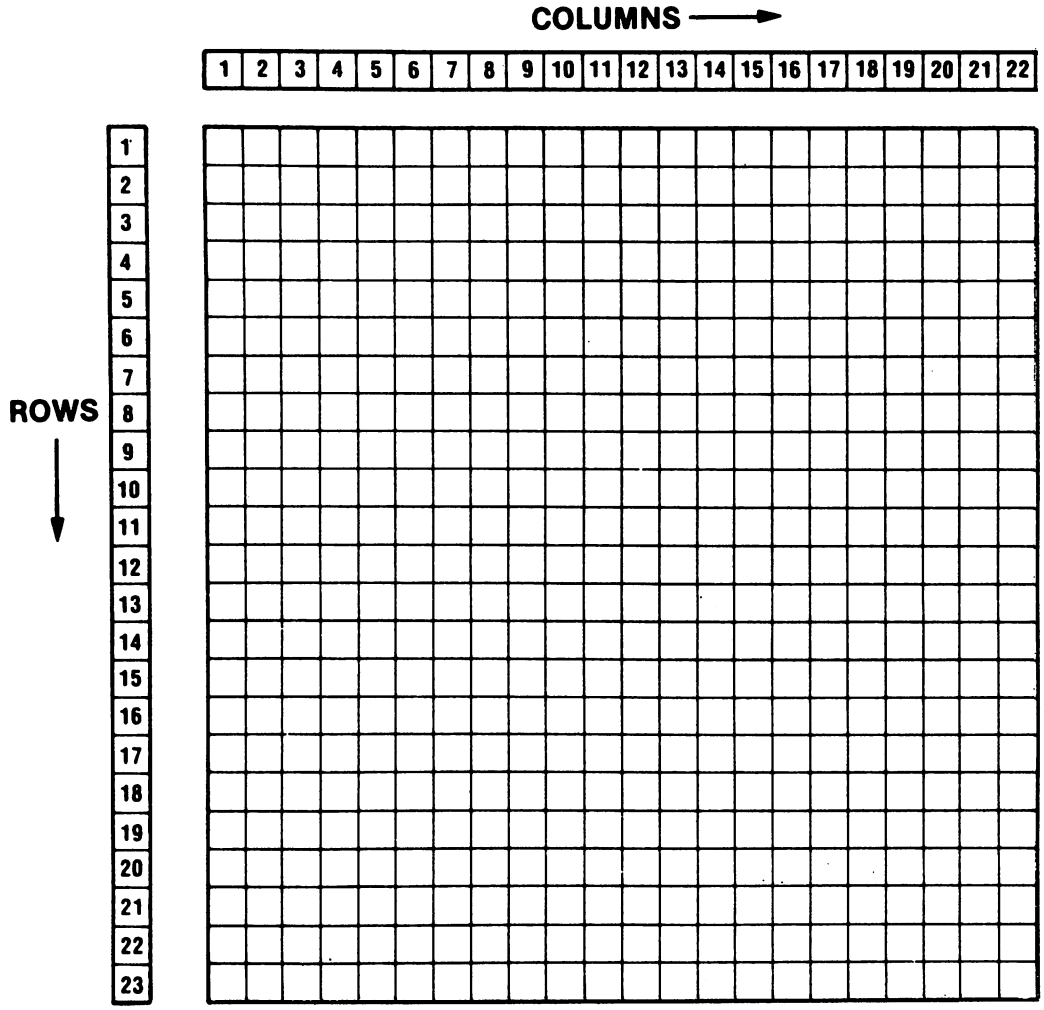
Graphic Characters Reference Chart G

Card Suit

DISPLAY	DISPLAY	DISPLAY	DISPLAY
			
KEY(1) A	S	Z	X

(1) RIGHT-SIDE GRAPHICS (USE SHIFT KEY).

VIC
Video Display Worksheet for Graphics

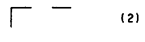


Tips on Using Graphics

Drawing pictures on the VIC is somewhat like putting together a puzzle that has 62 possible pieces. Each piece of the puzzle is represented by one of the 62 graphic characters. The major differences between a puzzle and a picture on VIC, however, is that you will not use all 62 pieces all of the time but you might use the same pieces several times to complete your picture. For example, if you were going to draw a 3 X 3 square you would need the following (refer to pages 170 and 172): (Draw segments on paper. Do not use the computer yet.)

1. An upper-left corner (┌), which is a shifted [O] key. If you press the shifted [O] key, you should see this symbol on display:

2. A straight piece or horizontal line segment that matches the line segment of the corner (—). You have to be careful here because there are eight possibilities (that is, you could select one of these keys: [T], [D], [*], [R], [E], [C], [F], or [@]). But a close examination of the pieces (keys) will show you that only one key will produce the horizontal line segment that matches the top-left corner and that is the left-side [T] key (—). Therefore, you now have this (without the space):



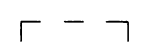
3. A right-top corner (┐), which is a right-side [P] key. Your picture would look like this now (without the spaces):

4. A vertical line segment (|) that matches the right-top corner. Again you have eight possibilities (keys [G], [G], [-], [Y], [T], [B], [H], or [M]). But the only piece or vertical line segment that matches the line segment of the top right corner is the left-side [M] key. We now have this (without the spaces):

5. A bottom-right corner (_|), which is a shifted [@] key. Our square is really beginning to take shape now and looks like this (without the spaces):

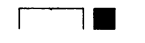
6. A horizontal segment (—) that matches the bottom-right corner. Again, there are eight possibilities but only the left-side [@] key matches the bottom corner piece. The figure now looks like this:

7. A bottom-left corner (└), which is a shifted [L] key. We are almost there because the picture now looks like this:

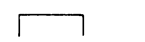


8. Finally, a vertical line segment (|) for the left side. Of the eight possibilities only the left-side [G] key matches. We now have all the pieces to the square which would look like this (if we remove the spaces between the line segments):

The above procedure describes the basic approach to graphics. There are other things to consider if you are going to draw this picture on the VIC, however. For example, what about spacing? You would have to move the picture to the desired part of the screen. Otherwise, all pictures will start in the upper-left-hand corner of the screen because that is where the cursor starts. (We will show you how to do this later.) Also, after you type the top-right corner the cursor will not be in the proper position to draw the vertical line segment. The cursor would be to the right of your picture. Therefore, you would have to move the cursor down and to the left to get it into the proper position for drawing the vertical line segment. You would use [CRSR ⇩] and [CRSR ⇦] keys to get it in the proper position. (We will use the cursor control keys in another example.)

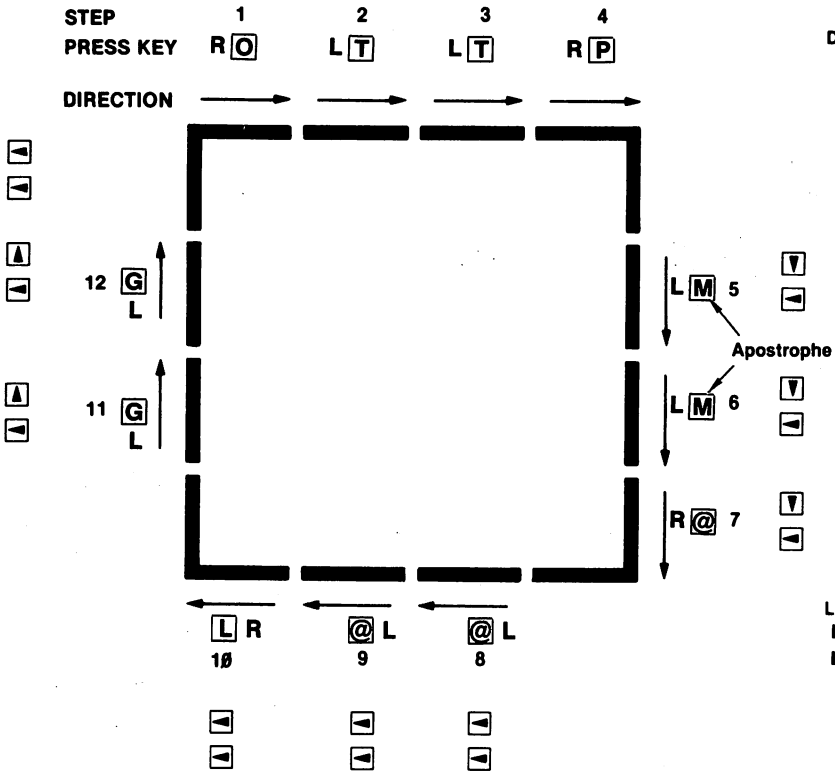


Finally, always draw your picture first on the video display worksheet or on a piece of graph paper that is 22 blocks wide and 23 blocks long. Then match the line segments of your picture with the appropriate graphic symbol assigned to one of the 62 graphic keys. Write the desired key letter next to the line segment on your worksheet.



Note! You should experiment with the graphic keys until you feel comfortable using them. You should also use graphic characters reference charts (pages 170-176), which provide magnified versions of each graphic symbol together with its respective key.

Exercise 12-1 Drawing A Square (In The Calculator Mode)



DIRECTIONS:

1. Follow procedure step by step
2. Hold down shift key and press key shown □
3. Use the cursor control key shown before pressing graphic keys for right, bottom and left sides of square. (You need to do this so that the sides will line up properly.)

LEGEND:

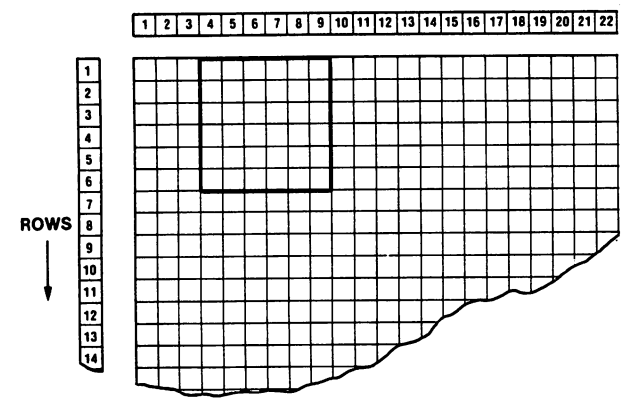
R O, P etc. — Right-side graphic keys
L M, T — Left-side graphic keys.
— Cursor down
— Cursor left
— Cursor right
— Cursor up

Exercise 12-2 Drawing A Square (In Program Mode)

DIRECTIONS:

Just follow the steps below to draw the picture shown. Read from left to right and press the graphic keys once for each time listed. (A □ represents the space bar.)

PARTIAL LAYOUT OF DISPLAY



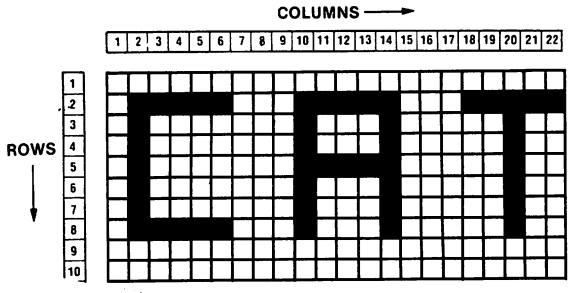
YOUR ACTION	LINE #	ENTER PROGRAM KEYWORD	QUOTES	PRESS GRAPHIC KEYS (NOTE 1) TO DRAW PICTURE ABOVE	QUOTES & RETURN
STEP 1	10	PRINT	"	□□□OTTTTP	" RETURN
STEP 2	20	PRINT	"	□□□G□□□□M	" RETURN
STEP 3	30	PRINT	"	□□□G□□□□M	" RETURN
STEP 4	40	PRINT	"	□□□G□□□□M	" RETURN
STEP 5	50	PRINT	"	□□□G□□□□M	" RETURN
STEP 6	60	PRINT	"	□□□L@@@@@@	" RETURN
STEP 7	99	END			RETURN
STEP 8	5	PRINT	"	▼	" RETURN (CLEARS SCREEN)
STEP 9	RUN				

NOTE 1: LEFT-SIDE GRAPHICS: T, M, @*, G (USE WITH C KEY)
RIGHT-SIDE GRAPHICS: O, P, @*, L (USE WITH SHIFT KEY)
***NOTE THAT BOTH THE LEFT AND RIGHT SIDES OF @ KEY ARE USED.**

Exercise 12-2 Drawing Pictures

YOUR ACTION	DISPLAY
1. Type program lines shown (Do not type NEW) 2. Before you run the program with these lines added, write what you expect to see. _____ _____ _____ 3. Run the program. 4. Explain what happened and why. 5. List your program and make certain you understand it.	15 FOR J = 1 TO 2500 : NEXT 25 FOR J = 1 TO 2500 : NEXT 35 FOR J = 1 TO 2500 : NEXT 45 FOR J = 1 TO 2500 : NEXT 55 FOR J = 1 TO 2500 : NEXT

Exercise 12-3 Drawing Large Sized Letters with VIC



DIRECTION:
Type and enter the program line by line as shown. (Don't forget the quotes and the RETURN key.)

LEGEND:
- □ — Space Bar
- ⊞ — Graphic key in shifted mode
- ▼ — Clear screen symbol

LINE NO. / KEYWORD / GRAPHIC KEYS

```
100 PRINT " ▼ "
110 PRINT " □ ⊞⊞⊞⊞⊞ □ □ ⊞⊞⊞⊞⊞ □ □ ⊞⊞⊞⊞⊞ " RETURN
120 PRINT " □ ⊞ □ □ □ □ □ ⊞ □ □ □ ⊞ □ □ □ □ ⊞ □ □ " RETURN
130 PRINT " □ ⊞ □ □ □ □ □ ⊞ □ □ □ ⊞ □ □ □ □ ⊞ □ □ " RETURN
140 PRINT " □ ⊞ □ □ □ □ □ ⊞⊞⊞⊞⊞ □ □ □ □ ⊞ □ □ " RETURN
150 PRINT " □ ⊞ □ □ □ □ □ ⊞ □ □ □ ⊞ □ □ □ □ ⊞ □ □ " RETURN
160 PRINT " □ ⊞ □ □ □ □ □ ⊞ □ □ □ ⊞ □ □ □ □ ⊞ □ □ " RETURN
170 PRINT " □ ⊞⊞⊞⊞⊞ □ □ ⊞ □ □ □ ⊞ □ □ □ □ ⊞ □ □ " RETURN
```

TAB and SPC Functions

- **TAB**
  - — **Moves the cursor to the right to the specified column position.**
  - — **Format:**

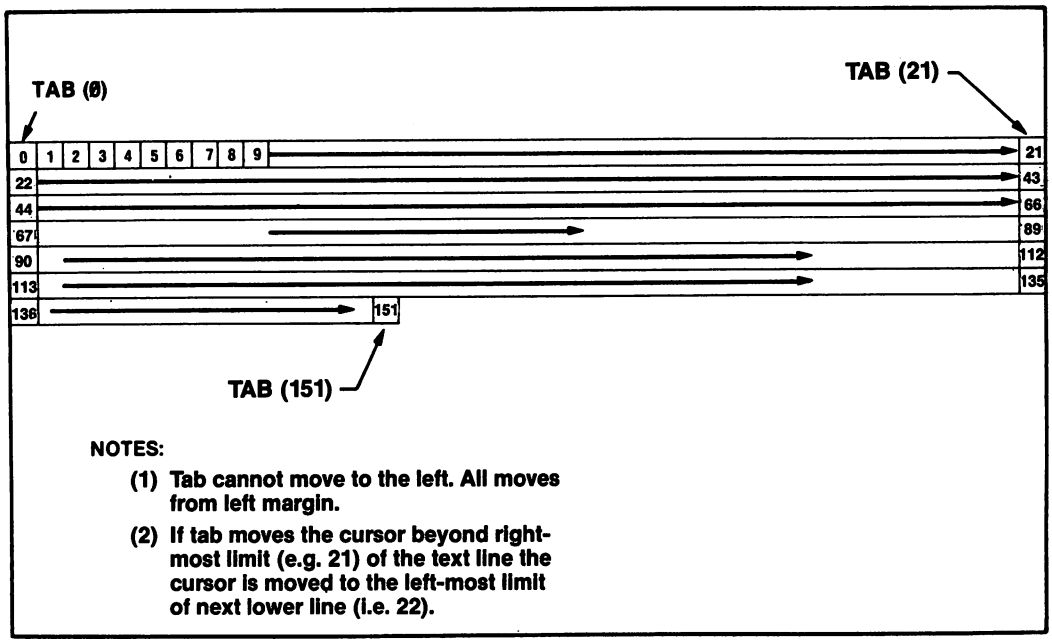
    TAB (N) where N is a number in the range from Ø to 255.
  - — **Example:**

    1Ø PRINT TAB (1Ø) "TABBED 1Ø".
- **SPC**
  - — **Moves cursor to the right a specified number of positions starting at the current cursor position.**
  - — **Format:**

    SPC (N) where N is a number in the range from Ø to 255.
  - — **Example:**

    1Ø PRINT SPC (1Ø) "SPACED 1Ø".

Illustration Showing Location of Tab Printing Positions (Ø to 151)



NOTES:

(1) Tab cannot move to the left. All moves from left margin.

(2) If tab moves the cursor beyond right-most limit (e.g. 21) of the text line the cursor is moved to the left-most limit of next lower line (i.e. 22).

TAB Example

YOUR ACTION

1. **Type and enter program shown.**
2. **RUN the program. Note the following:**
   - **TAB moves the cursor right to a specified column position.**
   - **TAB (Ø) is the first column on the left and TAB (21) is the last column on the right.***
   - **TAB (22) starts a new line but is same column as TAB (Ø).**
   - **TAB (255) is the last tab position.**

*See page 184.

DISPLAY

```
 5 Print "♥"
10 PRINT TAB (Ø) "TAB
Ø"
20 PRINT TAB (1) "TAB
1"
30 PRINT TAB (5) "TAB
5"
40 PRINT TAB (10) "TAB
10"
50 PRINT TAB (22) "TAB
22"
60 PRINT TAB (30) "TAB
30"
70 PRINT TAB (255) "T
255"
```

```
TAB Ø
 TAB 1
     TAB 5
          TAB 10
TAB 22
          TAB 30
                    T   255
```

SPC Example

YOUR ACTION

1. Type and enter program shown.

DISPLAY

```
 5 PRINT "♥"
10 PRINT SPC (0) "SPC
0"
20 PRINT SPC (1) "SPC
1"
30 PRINT SPC (5) "SPC
5"
40 PRINT SPC (10) "SPC
10"
50 PRINT SPC (22) "SPC
22"
60 PRINT SPC (30) "SPC
30"
```

2. RUN the program. *Note:* This example works the same as TAB because the starting position of the cursor in this example is the left side of the display, which is the same as the starting position for TAB.

```
SPC 0
 SPC 1
      SPC 5
            SPC 10
SPC 22
          SPC 30
```

Difference between TAB (N) and SPC (N) — Example

Example:

```
10 PRINT "ADRIENNE"; T
AB (10); "JONES"
20 PRINT "ADRIENNE"; S
PC (10); "JONES"
RUN
ADRIENNE      JONES
■ ——————→ ■          (cursor)
```

(TAB (10) moves cursor 10 spaces right from first column to start printing "JONES")

```
ADRIENNE                JON
ES
          ■ ——————→ ■          (cursor)
```

(SPC (10) moves cursor 10 spaces right from the last cursor position to start printing "JONES")

In summary, TAB (N) moves cursor right to a specified column (N+1) whereas SPC (10) moves cursor to a specified position (N) spaces to the right of the current cursor position.

Exercise 12-4 — Difference between SPC and TAB

YOUR ACTION

1. Type and enter this program: →
***Note:* Line 20 (TAB) causes printing to start 5 spaces to the right. Line 30 does same as line 20 and SPC (5) inserts 5 spaces between "STUDENTS" and "GRADE"**

DISPLAY

```
 5 PRINT "♥"
10 PRINT "# STUDENTS";
"GRADE"
15 PRINT
20 PRINT TAB (5) "# ST
UDENTS"; "GRADE"
25 PRINT
30 PRINT TAB (5) "# ST
UDENTS" SPC (5); "GRAD
E"
99 END
```

2. RUN the program and explain what happened.

```
# STUDENTSGRADE

      # STUDENTSGRADE

      # STUDENTS          GRA
DE

READY
■
```

3. Type and enter this program.

```
100 PRINT "ADRIENNE" T
 AB (10) "JONES"
110 PRINT "ADRIENNE" S
 PC (10) "JONES"
```

4. RUN the program (RUN 100).

```
ADRIENNE  JONES

ADRIENNE            JON
ES
```

5. Explain what happened.

TAB and SPC Functions — Summary

- These functions are useful in setting up your output print format.
  - — TAB (N) moves the cursor right to a specified column (N+1).
  - — SPC (N) moves the cursor to a specified position (N) spaces to the right of the current cursor position.
  - — TAB and SPC are especially useful when the output is a column of numbers with headings.
  - — For SPC (N) and TAB (N), (N) can be a number from Ø to 255.

PRACTICE 21

Graphics

1. Write a program that will do the following:
   a. Draw an 8 × 8 square three blocks or squares from the left side of the screen.
   b. Three blocks from the left side of the screen draw a letter C that is 7 blocks long and 5 blocks wide.
   c. Try other pictures or letters (if you have the time).

PART 13

Arrays

What You Will Learn

1. To explain the purpose of using arrays.
2. To set up one- and two-dimensional numeric arrays.
3. To explain the purpose and use of the terms DIM, A(3), A(2,3), DIM A(10), DIM DB(7,5).
4. To develop, enter, and run programs using numeric arrays.

Arrays

A. What is an array?

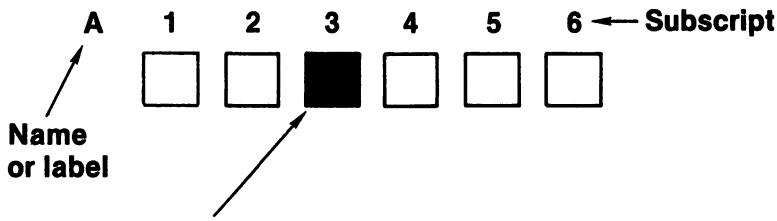
- **An array is a lineup, an arrangement, or an orderly grouping of things.**

B. Why use an array?

- **Use it when we wish to have more variables available in a program.**
  - **— Although the VIC BASIC permits the use of approximately 900 variables for numerics, sometimes thousands of variables are required for storing and retrieving many pieces of data.**
  - **— The array allows you to arrange your data so that it can be stored and retrieved easily.**

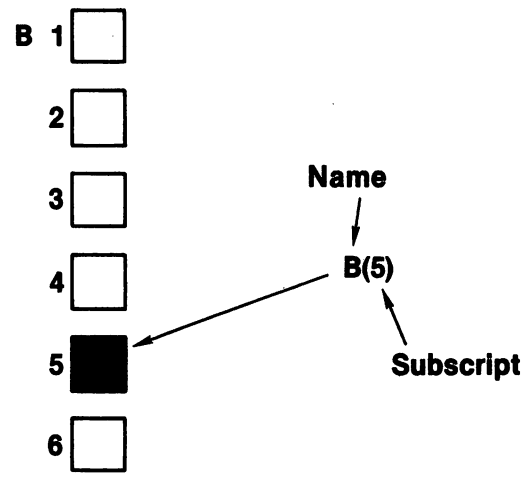
One-Dimensional Array — Illustration

SIX-ELEMENT ARRAY — NAMED A*



- A(3) is pronounced A SUB 3.
  - — A(3) represents the third cell or box in the array (lineup).
  - — Data stored in this cell would be addressed by the label A(3).
  - — Suppose data were stored in the sixth cell: A(6)? (You got it!)

SIX-ELEMENT ARRAY — NAMED B*



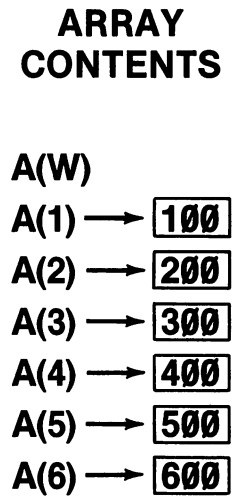
- B(5) represents the fifth cell in the array where data can be stored and retrieved.

*A and B are optional names. Any valid variable name can be used to name an array in VIC BASIC.

One-Dimensional Array — Program Example

PROGRAM	DISPLAY	REMARKS
1Ø DATA 1ØØ, 2ØØ, 3ØØ, 4ØØ, 5ØØ, 6ØØ 2Ø FOR W = 1 to 6 3Ø READ A(W) 4Ø NEXT W		• Lines 2Ø-4Ø store data in array A(W)
5Ø FOR W = 1 to 6 6Ø PRINT W, A(W) 7Ø NEXT W RUN	1 1ØØ 2 2ØØ 3 3ØØ 4 4ØØ 5 5ØØ 6 6ØØ	• Lines 5Ø-7Ø retrieve data from array A(W)

One-Dimensional Array — Program Example (Con't)



Above is an illustration of what happens after data are stored in array A(W). Note that in location A(1), the first data element (100) is stored. In location A(2), the second data element (200) is stored, and so on until the sixth data element (600) is stored in location A(6). Remember that line 10 of the program contained the data elements that were read using lines 20 through 40.

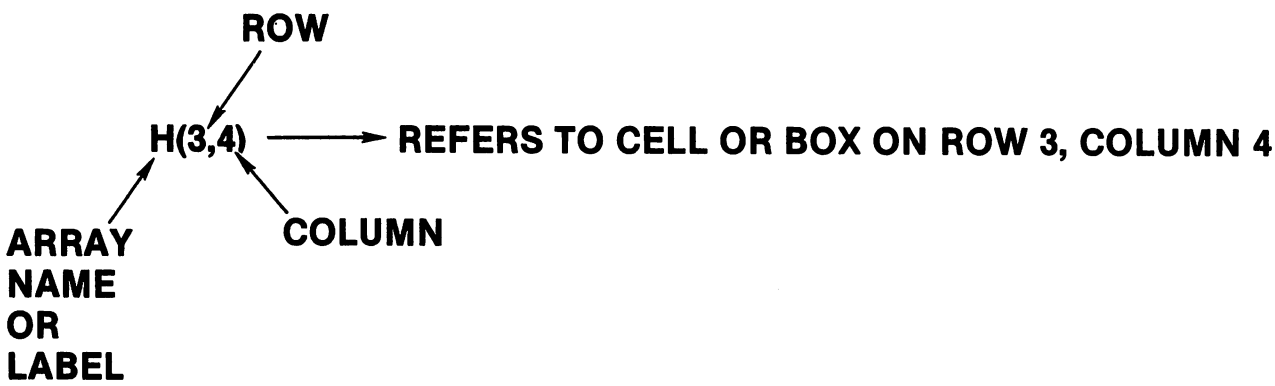
Two-Dimensional Array — Illustration

COLUMN

ROW

H	1	2	3	4	5	6
1	11	12	13	14	15	16
2	21	22	23	24	25	26
3	31	32	33	34	35	36
4	41	42	43	44	45	46
5	51	52	53	54	55	56
6	61	62	63	64	65	66

36 ELEMENT ARRAY (MATRIX)
(NAMED H)



In-Class Exercise 13-1

(Fill in the Blanks Using the Matrix on Page 194)

LABEL	ROW	COLUMN	CONTENTS
H(1,1)	_____	_____	_____
H(4,5)	_____	_____	_____
H(3,3)	_____	_____	_____
H(2,3)	_____	_____	_____
H(6,6)	_____	_____	_____
H(1,6)	_____	_____	_____
H(2,4)	_____	_____	_____
H(4,4)	_____	_____	_____

DIM Statement

- **DIM - Lets you set the depth (number of elements allowed per dimension)**
  - **— If no DIM statement is used, a depth of 11 (subscripts Ø-1Ø) is allowed for each dimension of each array used.**
  - **— DIM statements may be placed anywhere in your program.**
  - **— To redimension an array, you must first use a CLR statement. Otherwise, an error will result!**
- **EXAMPLE**

```
1Ø  DIM  A(6),        B(2,3),        C(21)
```

Sets a one-dimension array A with 6 elements A(Ø) — A(5) or A(1) — A(6)*

Sets a two-dimension array B with 3 ROWS (numbered Ø-2) and 4 COLUMNS (numbered 0-3)

Sets a one-dimension array with 21 elements A(Ø) — A(2Ø) or A(1) — A(21)*

***If A(Ø) is not used**

Checkbook Array Example

- Consider the following table of checkbook information:

Check #	Date Written	Amount
1ØØ	6/5/81	$ 15.5Ø
1Ø1	6/7/81	25.ØØ
1Ø2	6/15/81	145.ØØ
1Ø3	6/22/81	65.ØØ
1Ø4	6/3Ø/81	211.ØØ
1Ø5	6/3Ø/81	79.5Ø

- Note that every item in the table may be specified by reference to two numbers: the row number and the column number. For example, (Row 3, Column 3) refers to the amount $145.ØØ.
- The above table can be set up in a 6 × 3 array or matrix (see next page).

Checkbook Array Example (Con't)

CK	1	2	3
1	100	60581	15.50
2	101	60781	25.00
3	102	61581	145.00
4	103	62281	65.00
5	104	63081	211.00
6	105	63081	79.50

6 × 3 MATRIX (ARRAY) — NAMED CK

NOTES:

1. Data recorded in form mm ddyy where mm = month number, dd = day, and yy = last two digits of year.
2. Since CK is a numeric array, alpha-numerical characters such as dashes cannot be stored.

Checkbook Array Example (Con't)

YOUR ACTION

1. Setting Up the Array (Lines 1Ø through 11Ø)

A. Let's type and enter Lines 1Ø through 11Ø as shown:

(NOTE: Line 1Ø sets up dimension of array. Lines 2Ø-11Ø read the values into array CK.)

NOTE: DIM CK (6, 3) Sets up a 6 × 3 array (excluding zero subscripts) with 6 rows (numbered 1 to 6) and 3 columns (numbered 1 to 3)

2. Manipulating the Array (Finding the Sum)

A. Add lines 12Ø through 16Ø to the program as shown:

(NOTE: Lines 12Ø-16Ø add up all the checks written.)

B. Type RUN and press RETURN.

DISPLAY

```
1Ø DIM CK (6,3)
2Ø FOR ROW = 1 to 6
3Ø FOR COL = 1 to 3
4Ø READ CK(ROW, COL)
5Ø NEXT COL, ROW
6Ø DATA 1ØØ, 6Ø581, 1
5.5Ø
7Ø DATA 1Ø1, 6Ø781, 2
5.ØØ
8Ø DATA 1Ø2, 61581, 1
45.ØØ
9Ø DATA 1Ø3, 62281,
65.ØØ
1ØØ DATA 1Ø4, 63Ø81, 2
11.ØØ
11Ø DATA 1Ø5, 63Ø81, 7
9.5Ø
12Ø FOR ROW = 1 to 6
13Ø SUM = SUM + Ck (RO
W, 3)
14Ø NEXT ROW
15Ø PRINT "TOTAL OF CHE
CKS WRITTEN";
16Ø PRINT " $"; SUM
TOTAL OF CHECKS WRITTE
N $541.
```

NOTE:

ROW and COL are used for convenience. Remember, however, the computer will only use the first two characters, RO and CO in this example.

Checkbook Array Example (Con't)

YOUR ACTION | **DISPLAY**

3. Manipulating the Array (Print out all checks written on a given day)

A. Do not type NEW.

B. Add the following steps to your program: →

```
200 PRINT "LIST CHECKS
 WRITTEN ON (MM DD YY)"
205 INPUT "ENTER (MM DD
 YY)"; DT
210 PRINT: PRINT "CHECKS
 WRITTEN ON"; DT; "ARE L
 ISTED BELOW"
215 PRINT
220 PRINT "CHECK #", "AMO
 UNT": PRINT
230 FOR ROW = 1 TO 6
240 IF CK (ROW,2) = DT TH
 EN PRINT CK (ROW,1),
 CK ( ROW,3)
250 NEXT
```

C. Type RUN and press RETURN.

D. Enter a date (e.g., 63081 which is 6/30/81). →

```
TOTAL OF CHECKS WRITTE
N: $541.
LIST CHECKS WRITTEN ON
(MM DD YY)?
ENTER (MM DD YY)?
CHECKS WRITTEN ON 63081
ARE LISTED BELOW;
     CHECK #          AMOUNT
       104             211.00
       105              79.50
READY
■
```

Assignment 13-1

Read pages 119 and 120 in *VIC Personal Computer Guide.*

Summary

- A2 ≠ A(2)
  - — A2 is an ordinary variable
  - — A(2) is a subscripted variable
- Any time you have a subscript larger than 10 (depth of 11), you must use a DIM statement.
  - — Example:

    1Ø DIMA (25), B(17, 18)
- One-Dimensional Array

  SUBSCRIPT

  — A(3) is pronounced A SUB 3

  NAME
- Two-Dimensional Array (Matrix)

  ROW

  — H(3,4) refers to cell or box on row 3, column 4

  NAME COLUMN

PRACTICE 22

Arrays

1. Write a program to read the following numbers into an array and then PRINT them out:
676 15Ø 175 188 19Ø 277 876 976 912 544
2. Change program to find the sum and average of the 1Ø numbers given.
3. Label the answer: The sum is _____, and the average is _____.

PRACTICE 23

One-Dimensional Array

1. Suppose we had the following results of a quiz given to a class of 1Ø students:

Student #	1	2	3	4	5	6	7	8	9	1Ø
Student's Grade	75	85	95	87	1ØØ	77	83	69	98	88

   a. Using a one-dimensional Array, write a program to find the class average.
   b. Add the necessary program lines to find the highest grade and the lowest grade.
   c. Have the program PRINT : Class Average is _____, Highest Grade is _____ and Lowest Grade is _____.
   d. Enter and RUN each of these programs several times.

PART 14

INT(X), ABS(X) & RND(X) Functions

What You Will Learn

1. To explain the purpose and use of INT(X), ABS(X), and RND(X) functions.
2. To explain the purpose and use of the term RANDOM.
3. To write, run, and analyze programs using the INT(X), ABS(X), and RND(X) functions.

INT(X) Function

- INT(X) or integer function allows you to round off any number, large or small, positive or negative, into a whole number (or integer).
- INT(X) means
  - — If X is a positive number, then the largest whole number can be found by chopping off the decimal part.

    Example:

    INT (5.7) = 5

    INT (Ø.7) = Ø
  - — If X is a negative number, the largest whole number can be found by moving down to the next lowest whole number (that is, make a negative number more negative).

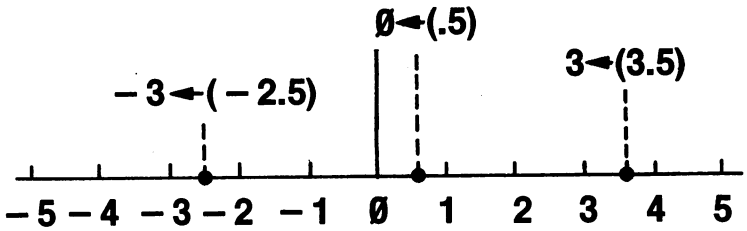
    Examples:

    INT (−. 6) = −1 INT (−3.14) = −4

    INT (−.2) = −1 INT (−7.28) = −8

Exercise 14-1 INT(X)

Graphical Representation



For negative numbers:
"Move to next lowest whole number"

For positive numbers:
"Chop off decimal part"

X	INT(X)
Ø.5	_____
–1.7	_____
2.345	_____
–Ø.8	_____
Ø	_____
3.1415	_____
76.14	_____
–1Ø.35	_____

INT(X) FUNCTION — ROUNDING $$

YOUR ACTION

1. Type and enter this program. →

2. Now RUN.

3. Add Line 15 to program as shown.
(Note: In Line 15 we multiply by 100, add .5, take the INT, which is now 667, and then divide 667 by 100. 667/100 is 6.67, which is what we want, two decimal places.)

4. Now RUN program.
(I told you so!)

DISPLAY

```
10 LET A = 20/3
20 PRINT "$"; A
$ 6.66666667
15 A = INT (100*6.6666
6667 + .5) /100
$6.67
```

Assignment 14-1 INT(X)

1. Type NEW and enter this program for finding the area of a circle:

```
10 REM AREA OF A CIRCL
E 3.14159* R ↑ 2
20 INPUT "THE RADIUS I
S "; R
30 P = 3.14159
40 A = P*R ↑ 2
50 PRINT "THE AREA IS"
; A
```

2. RUN the program several times to make sure it works.
3. Change the program to suppress (chop off) all of the numbers to the right of the decimal point. (RUN the program to make sure it works.)
4. Change the program to make the answer accurate to one decimal place. (For example, if R = 1, then Area (A) = 3.1.)

ABS(X) Function

- **ABS(X) = Abbreviation for absolute value of X**
- **Examples:**

ABS (12) = 12	**ABS (− 1Ø) = 1Ø**
ABS (Ø) = Ø	**ABS (−357) = 357**

- **Note! ABS (25 − 1Ø) = ABS (1Ø − 25) = 15**

Assignment 14-2 ABS(X)

YOUR ACTION	DISPLAY
1. Type and enter the program shown.	5 PRINT "♥" 1Ø INPUT "TYPE ANY + 0 R − #"; N 2Ø X = ABS (N) 25 PRINT:PRINT 3Ø PRINT "N", "X" 35 PRINT:PRINT 4Ø PRINT N,X
2. RUN the program several times using both positive and negative numbers. **(Note that regardless of the number you input as N, the absolute value of X is the same number without the sign.)**	

RND(X) Function

- RND(X) or random number function causes the computer to give you a "surprise" number.
  - — It's as though the computer spins a wheel of chance.
  - — It's like pulling a number out of a hat.
  - — It's unpredictable!
- The random number function – general form

  Let N = INT (X * RND(1) + 1)

  Where N = The random number

  RND = Abbreviation for random

  X = Any number between 1 and 32767
- The general form for finding random numbers may seem a little complicated at first but it's not once you understand how to use it. All you need to do is just give "X" the value or number you wish to be the highest random number. When you run the program, you will have a number between 1 and X.

  Example:

  10 PRINT INT (6 * RND (1) + 1) (will give you a random number from 1 to 6 inclusive)

  20 PRINT INT (4 * RND (1) + 1) (will give you a random number from 1 to 4 inclusive)

  30 PRINT INT (10 * RND (1) + 1) (will give you a random number from 1 to 10 inclusive)
- Type, enter, and RUN the above program several times or until you understand how random numbers work.

Random Number — Program Example

YOUR ACTION	DISPLAY
1. Type and enter. (Line 5 allows you to enter "X" or the highest random number you want.)	5 PRINT "ENTER A NO. B ETWEEN 1 AND 100" 7 INPUT "ENTER NO."; X 10 FOR J = 1 TO 10 20 PRINT INT (X * RND (1)+1); 30 NEXT J
2. RUN. (Observe that there are ten random numbers between 1 and X where X = your input).	(SCREEN SHOULD HAVE TEN RANDOM NUMBERS BETWEEN 1 AND X.)
3. RUN program again to get the idea.	
4. Change Line 10 to read:	10 FOR J = 1 TO 100
5. RUN. (Get the idea?)	(SCREEN SHOULD HAVE ONE HUNDRED RANDOM NUMBERS BETWEEN 1 AND X.)

Coin Toss Program

ACTION AND REMARKS	DISPLAY
1. Type and enter program as shown:	5 REM COIN TOSS PROGRA M 10 REM H = HEADS, T = TAILS 20 H = 0: T = 0: PRINT 25 PRINT "♥" 30 PRINT "HOW MANY TIM ES SHALL I FLIP THE CO IN" 35 INPUT "ENTER NO."; N 40 PRINT "♥" 50 PRINT "I'M FLIPPING THE COIN... STANDBY" 60 FOR K = 1 TO N 70 X = INT (2 * RND (1)+1) 80 ON X GOTO 90, 100 90 H = H + 1: GOTO 110 100 T = T + 1 110 NEXT K 120 PRINT "♥"
(Line 20 initializes counters, sets H = T = 0.)	
(Line 40 starts next line at top of screen.)	
(Line 60 begins FOR-NEXT statement and runs it "N" times.)	
(Line 70 generates integers between 1 and 2.)	
(Line 80 tells the program to go to Line 90 if X = 1 = heads and to Line 100 if X = 2 = tails.)	
(Line 90, "heads" are counted.)	
(Line 100, "tails" are counted.)	
(Line 110 sends control back to Line 60 for "N" passes.)	

Coin Toss Program (Con't)

ACTION AND REMARKS	DISPLAY
(Line 13Ø prints the headings.)	13Ø PRINT "HEADS", "TA ILS",
(Line 14Ø prints the values of H and T.)	14Ø PRINT H, T: PRINT
(Line 15Ø calculates and prints the percentage of heads, percentage of tails.)	15Ø PRINT 1ØØ*H/N; "% ", 1ØØ*T/N; "%"
(Line 155 provides spacing for better appearance.)	155 PRINT
	16Ø PRINT "TOTAL FLIPS ="; N

Assignment 14-3 RND(X)

YOUR ACTION	DISPLAY
1. Type and enter the program as shown. ——→	5 REM PICK A NUMBER GA ME 10 PRINT "♥" 20 X = INT (10 * RND (1) + 1) 25 PRINT "♥" 30 PRINT "ENTER A NUMB ER BETWEEN 1 & 10" 35 PRINT : INPUT "ENTER THE NO."; N 40 IF X = N GOTO 100 50 IF X < N GOTO 110 60 IF X > N GOTO 120 100 PRINT "♥" : PRINT " RIGHT ON" 105 FOR J = 1 TO 2500 : NEXT GOTO 10 110 PRINT : PRINT "LOW ER" : GOTO 25 120 PRINT : PRINT "HIG HER" : GOTO 25
2. RUN the program.	

NOTE! Hold down [RUN STOP] key and then press [RESTORE] key to stop program from running.

Assignment 14-3 RND(X)

3. Analyze the program.

   Line 1Ø _____ the display.

   Line 2Ø is the _____ generator.

   Line 35 allows the user to _____ a number.

   Lines 4Ø, 5Ø, and 6Ø are _____ statements that compare
   conditional, unconditional

   the random number _____ with the input number _____.
   X,N X,N

   Lines 1ØØ, 11Ø, and 12Ø are PRINT statements that guide the player. Why does Line 1Ø5 GOTO Line 1Ø and why do Lines 11Ø and 12Ø GOTO Line 25? Explain the function of Line 1Ø5.

4. Modify (change) the program to pick a number between 1 and 1ØØ, and RUN this program several times.

5. For more details on generating random numbers, refer to pages 43 and 13Ø in the *VIC Personal Computer Guide.*

Summary

- **ABS(X) — Provides the absolute value of X regardless of the number you input (i.e., X is that same number without the sign).**
- **INT(X) — Provides integer or whole number value of X.**
  - **— If X is a positive (+) number, it chops off the decimal part.**
  - **— If X is a negative number, it rounds down to the next lowest whole number (e.g., INT (−Ø.6) = −1).**
- **RND(X) — Causes the computer to give you a random number.**
  - **— INT (X * RND (1)+ 1) gives you a random number from 1 to X inclusive.**

PRACTICE 24

INT(X) and ABS(X)

1. Fill in the banks with the appropriate INT(X):

X	INT(X)
Ø.7	_____
−2.5	_____
6.365	_____
−Ø.8	_____
−1Ø.65	_____
Ø	_____
3.2425	_____
−7.61	_____
−Ø.3	_____
Ø.3	_____

2. The following program can be used for finding the area of a circle:

```
1Ø REM AREA OF A CIRCL
E = 3.14159 * R ↑ 2
2Ø INPUT "THE RADIUS IS
"; R
25 PRINT "THE RADIUS IS
IN (in., ft., or yd.,)"
27 PRINT : INPUT "ENTER
IN, FT, OR YD"; A$
3Ø A = 3.14159 * R ↑ 2
4Ø PRINT "THE AREA IS"
; A; "SQ"; A$
```

   a. Enter and RUN the program several times to make certain it works.
   b. Change the program to suppress (chop off) all the numbers to the right of the decimal point (RUN the program to make sure it works).
   c. Change the program to make the answer accurate to one decimal place. (For example if R = 1, then area (A) = 3.1).

PRACTICE 25

Random Number

1. Write a program that will let you pick a random number between 1 and 1ØØ. The program should let you input a number from the keyboard and provide the following clues on your guess.
   a. If the number you pick matches the number the computer picks, have the computer PRINT "Right On."
   b. If the number from the keyboard is too high, have the program print "Lower."
   c. If the number from the keyboard is too low, have the program print "Higher."
   d. Enter and RUN the program several times.

PART 15

Subroutines

What You Will Learn

1. To explain the purpose for using subroutines.
2. To explain the purpose and use of terms ON-GOTO, GOSUB, RETURN, ON GO-SUB.
3. To develop, enter, and run programs using subroutines and ON-GOTO statements.

Subroutine

What Is It?

- **A subroutine is a short program or routine that is built into a large program to do specific calculations or perform repetitive functions.**

Why Use It?

- **There are times when you need the same type of calculation at various points in your program, but instead of retyping the statements needed for this calculation each time, you can write a subroutine to perform the needed calculations.**

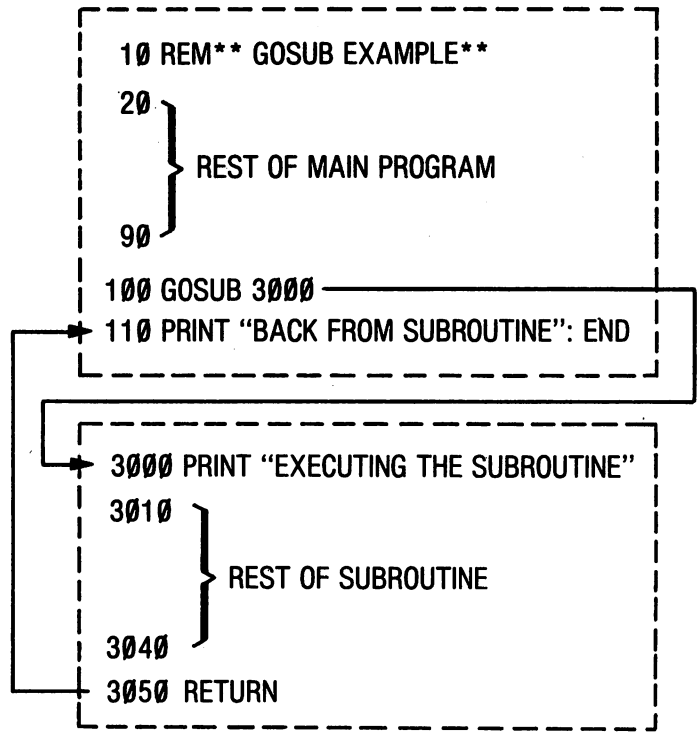
How Do You Call a Subroutine?

- **To call or branch to a subroutine, use the GOSUB statement.**
  - **— The GOSUB XXXXX statement directs the computer to go to that line number and execute the program steps until it reaches the key word RETURN, which ends the subroutine.**
  - **— RETURN is always built into a subroutine and is used to tell the computer that the subroutine is finished. When finished, the control of the program is returned to the statement in the main program immediately following the most recently executed GOSUB.**

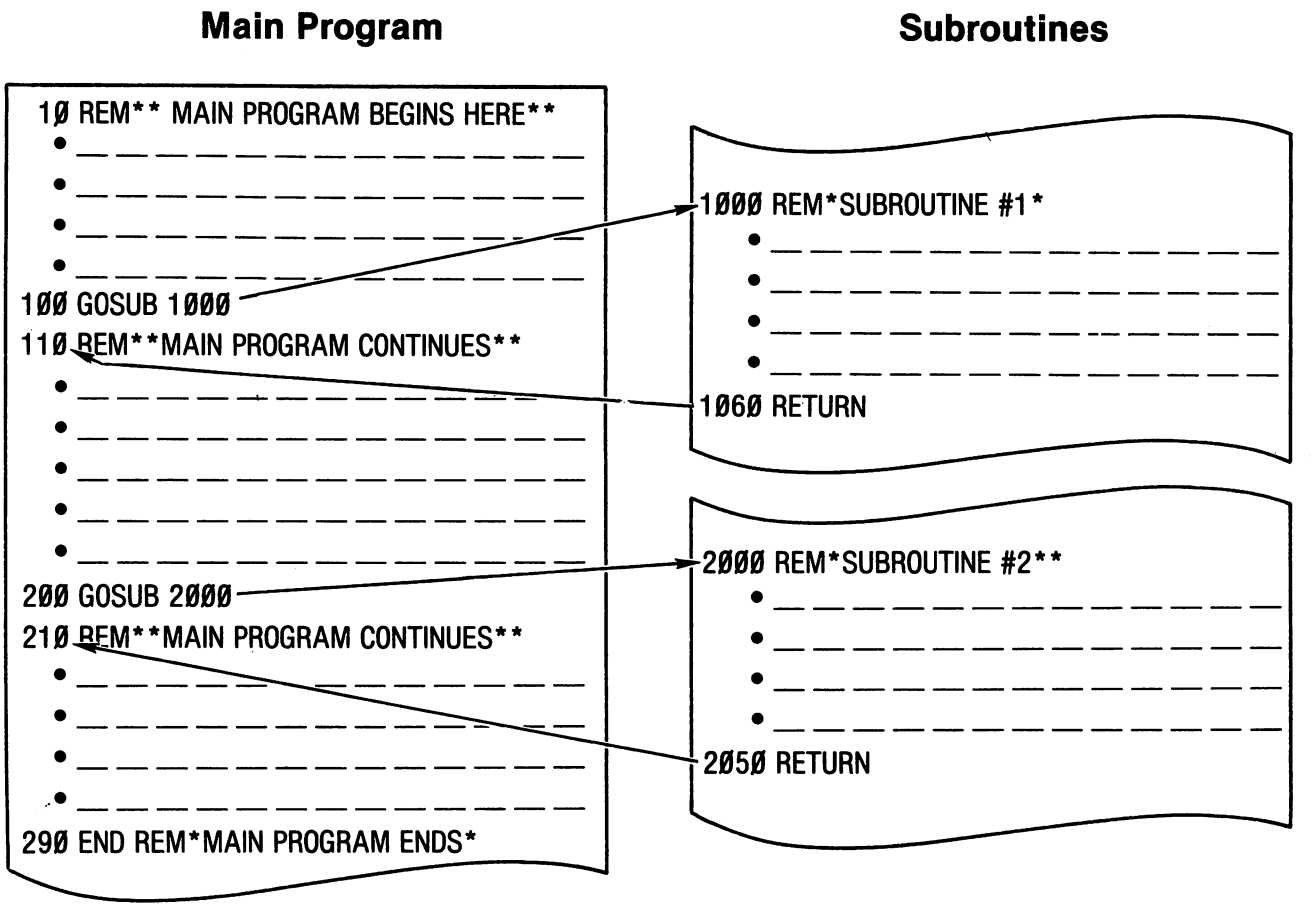
Subroutine Example

Main Program:

Subroutine:



Subroutine Illustration



Subroutine Illustration (Con't)

1. When the computer reaches the GOSUB in Line 1ØØ, the program will branch (GOTO) Line 1ØØØ, which is the beginning of Subroutine #1.
2. After Subroutine #1 is executed and the RETURN (Line 1Ø6Ø) is reached, control is passed back to the main program (Line 11Ø). Note that Line 11Ø is the next higher number after the GOSUB that put it in the subroutine (Line 1ØØ).
3. The computer continues through the main program to the GOSUB in Line 2ØØ, which branches control to Subroutine #2 in Line 2ØØØ.
4. After the subroutine is executed, the RETURN (Line 2Ø5Ø) passes the control back to Line 21Ø in the main program. (Note again that this is the next higher line number after the GOSUB in Line 2ØØ.)
5. An END statement is included in the program (Line 29Ø) after the main program is finished to keep it from accidentally falling into the subroutine. We only want the subroutines to be executed when we call for them by a GOSUB.

Sample Program Using Subroutines (Temperature Conversion)

Main Program

```
10 REM TEMP CONVERSION
PROGRAM
15 PRINT "♥"
20 PRINT "DO YOU WISH
TO CONVERT C TO F (Y O
R N)"
25 PRINT
27 INPUT "ENTER Y OR N
" ; A$
30 IF A$ = "Y" THEN 80
40 PRINT:INPUT "DEGREE
S F" ; F
50 GOSUB 2000
60 PRINT:PRINT "HAVE Y
OU FINISHED (Y OR N)"
65 PRINT
67 INPUT "ENTER Y OR N
" : B$
70 IF B$ = "N" THEN 40
75 END
80 PRINT:INPUT "DEGREE
S C"; C
90 GOSUB 1000
100 PRINT:PRINT "HAVE
YOU FINISHED (Y OR N)"
105 PRINT
107 INPUT "ENTER Y OR
N "; C$
110 IF C$ = "N" THEN 8
0
120 END
```

Subroutine #1

```
1000 REM C TO F CONVER
  SION
1010 F = (9/5)*C+32 : P
  RINT
1020 PRINT C; "DEGREES
  C="; F; "DEGREES F"
1030 RETURN
```

Subroutine #2

```
2000 REM F TO C CONVER
  SION
2010 C = (F –32)*(5/9)
  : PRINT
2020 PRINT F; "DEGREES
  F="; C; "DEGREES C"
2030 RETURN
```

Analysis of Sample Program Using Subroutines

1. Lines 1Ø through 11Ø comprise the main program.
2. Line 27 is an input statement to ask the user if he wants to convert from C to F or from F to C. Yes (Y) means C to F and No (N) means F to C.
3. Line 3Ø is a conditional branch statement. If the user wants to convert Centigrade C to Fahrenheit, then branch to Line 8Ø; otherwise, go to Line 4Ø.
4. Line 4Ø allows the user to input the °F to be converted to °C.
5. Lines 5Ø and 9Ø call the subroutines.
6. Line 67 asks the user if he is finished. In Line 7Ø the program will branch to Line 4Ø (if B$ = N) or the program will END (if B$ ≠ N).
7. Line 8Ø is similar to Line 4Ø, except that it allows the user to input the °C to be converted to °F.
8. Lines 1Ø5 and 11Ø are the same as Lines 65 and 7Ø.
9. The first subroutine begins at Line 1ØØØ and ENDS at Line 1Ø3Ø. It RETURNS control to Line 1ØØ in the main program.
10. The second subroutine begins at Line 2ØØØ and ENDS at Line 2Ø3Ø. It RETURNS control to Line 6Ø in the main program.

Subroutine Exercise

```
   5 PRINT "♥"
  10 PRINT "THIS IS"; "  ";
  20 GOSUB 1000
  30 PRINT "OF HOW"; "  ";
  40 GOSUB 2000
  50 PRINT "WORKS"
  60 END
1000 PRINT "AN EXAMPLE
  "; "  ";
1010 RETURN
2000 PRINT "A SUBROUTI
  NE"; "  ";
2010 RETURN
```

- **NOTE! Make sure you leave a space between quotes in Lines 10, 30, 1000, and 2000.**

1. Analyze the program and write the message. _____
2. Now type and enter the program.
3. RUN the program. Does it agree with your message?

Assignment 15-1

1. Analyze the program below and write the message:

```
  1Ø LET B = 1Ø
  2Ø GOSUB 2ØØØ
  3Ø B = B + 5
  4Ø GOSUB 2ØØØ
  5Ø B = B + 1Ø
  6Ø GOSUB 2ØØØ
  99 END
2ØØØ REM SUBROUTINE
2Ø1Ø IF B<12 THEN 2Ø5Ø
2Ø2Ø IF B = 25 THEN 2Ø
  7Ø
2Ø3Ø PRINT "PRIME"
2Ø4Ø GOTO 2Ø8Ø
2Ø5Ø PRINT "♥" : PRINT
  "LEEDS"
2Ø6Ø GOTO 2Ø8Ø
2Ø7Ø PRINT "COMPUTERS"
2Ø8Ø RETURN
```

Message_____

ON-GOTO Example

YOUR ACTION

1. Type NEW and enter this program.

2. Before you RUN the program, analyze it. Can you predict what will happen when you RUN it? (I sure hope you can by now!)

3. RUN the program several times until you feel comfortable with it.

DISPLAY

```
 5 PRINT "♥"
10 PRINT "TYPE A NUMBE
R FROM 1 TO 3"
15 PRINT: INPUT "ENTER
NO."; N
20 IF N = 1 GOTO 110
30 IF N = 2 GOTO 130
40 IF N = 3 GOTO 150
45 PRINT:PRINT
50 PRINT "HEY DUMMY, I
WANT A NUMBER FROM 1 T
O 3!"
60 PRINT:PRINT: GOTO 10
99 END
110 PRINT "N = 1"
120 END
130 PRINT "N = 2"
140 END
150 PRINT "N = 3"
160 END
```

ON-GOTO Example (Con't)

YOUR ACTION	DISPLAY
4. Erase Lines 2Ø, 3Ø, and 4Ø. Simply type in each line number separately and then press RETURN.	
5. Type and enter this line:	2Ø ON N GOTO 11Ø, 13Ø, 15Ø
6. List your program.	(SHOULD HAVE NEW LINE 2Ø + LINES 5, 1Ø, AND 45 THROUGH 16Ø FROM PREVIOUS PAGE. IF YOU DON'T HAVE THESE LINES, FIX IT!)
7. RUN the program a few times.	(WORKS JUST THE SAME AS BEFORE, DOESN'T IT?)
8. RUN the program again. Use the following inputs:	
1.5	N = 1
1.8	N = 1
2.8	N = 2
Ø.8	HEY DUMMY, I WANT A NU
3.99	MBER BETWEEN 1 & 3!
(Now do you understand that N = INT (N) or whole number?)	N = 3

ON-GOTO Example Analysis

1. Line 2Ø tells the computer to do the following:
   - If, the integer (whole number) value of N is 1, GOTO Line 11Ø.
   - If the integer value of N is 2, GOTO Line 13Ø.
   - If the integer value of N is 3, GOTO Line 15Ø.
   - If the integer value of N is not one of the numbers listed above, then move on to the next line.
2. The ON-GOTO statement has a built-in INT statement, which really acts like this:
   2Ø ON INT (N) GOTO----ETC.

Assignment 15-2 ON-GOTO

1. Type and enter the following program:

```
 5 PRINT "♥"
10 PRINT "ENTER A NUMBE
R FROM 1 TO 5"
15 PRINT: INPUT "ENTER N
O."; N
20 ON N GOTO 100, 200, 3
00, 400, 500
25 PRINT: PRINT
30 PRINT "HEY I WANT A
NUMBER FROM 1 TO 5!" : PR
INT : GOTO 10
40 END
100 PRINT "N = 1" : END
200 PRINT "N = 2" : END
300 PRINT "N = 3" : END
400 PRINT "N = 4" : END
500 PRINT "N = 5" : END
```

2. Answer the following questions before running the program
   a. What happens (output) if the input is 1.8 (Line 15)?_____
   b. What happens (output) if the input is 3.99? _____
   c. What happens (output) if the input is 2.89? _____
   d. What happens if the input is 0.5? _____

3. RUN the program several times and record the following:

INPUT	OUTPUT

ON-GOSUB

- Works like ON-GOTO, except control branches to one of the subroutines specified by the line numbers in the line number list.
- Example:

```
10 INPUT "Choose 1, 2,
OR 3"; K
20 ON K GOSUB 1000, 20
00, 3000
99 END
1000 PRINT "SUBROUTINE
#1" : RETURN
2000 PRINT "SUBROUTINE
#2" : RETURN
3000 PRINT "SUBROUTINE
#3" : RETURN
```

- K may be a numerical constant, variable, or expression.
  — It must have a positive value, however, or an error will occur.
- If K ≠ 1, 2, or 3, the program will go to the next line (99 END).

Summary

- GOSUB XXXX, causes the computer to:
  - — Go to the subroutine beginning at line XXXX (the specified line number).
  - — Work through the subroutine until it finds a RETURN statement.
  - — Return control to the statement that follows the GOSUB statement in the main program.
- ON n GOSUB XXXX, ------, YYYY
  - — Multi-way branching statement that is controlled by a test variable (n), which sends control of the program to one of the subroutines specified by line numbers in the line number list (i. e., XXXX,----, YYYY).
  - — The test variable n must be a numerical constant, variable, or expression that has a non-negative value or else an error will occur.
- ON n GOTO XXXX, ----, YYYY
  - — Works like ON n GOSUB except control branches to one of the line numbers specified (XXXX, ----, YYYY).
  - — ON n GOTO 1st line number, 2nd line number ———— nth line number expression must be between Ø and 255 inclusive.
  - — If n<Ø, an error will occur.

PRACTICE 26

Program to Convert Centigrade to Fahrenheit and Vice Versa

1. Write a program that will do the following:
   a. Convert Centigrade to Fahrenheit.
   b. Convert Fahrenheit to Centigrade.
   c. Allow you to select either A or B above.
   d. Allow you to input from keyboard.
   e. PRINT the answer as follows:

   ____*____ degrees Celsius = ____**____ degrees Fahrenheit

   or

   ____*____ degrees Fahrenheit = ____**____ degrees Celsius

   * Keyboard input value
   ** Calculated output value

PRACTICE 27

Program for Sample Profit/Loss Statement

1. When a product is sold for more than it costs, the seller receives a profit. When a product is sold for less than it costs, the seller takes a loss.

   Therefore: sell price − cost = profit or loss

   If we let: S = Sell price
   C = Cost
   U = No. of units
   P = Profit
   L = Loss

   Then: P (or L) = S*U − C*U

   a. Write a program that will compute the profit or loss for a business if the sell price and cost are known. (*Note:* Program should permit you to enter cost and sell price from the keyboard.)
   b. Have the computer PRINT the following:

   NO. OF UNITS _____
   UNIT PRICE ($) _____
   UNIT COST ($) _____
   TOTAL SALES ($) _____
   TOTAL COST ($) _____
   PROFIT/LOSS ($) _____
   % OF SALES _____

   c. RUN the program several times and record your answer.

EXTRA PRACTICE 1

Programming Mathematical Operators

1. Given two numbers A=25 and B=5:
   a. Write one program that will add, subtract, divide (A/B), multiply, and square the two numbers (A and B).
   b. The answer should PRINT as shown here:
      The sum of A and B is _____ (your answer).
      The difference of A and B is _____ (your answer).
      The quotient of (A/B) is _____ (your answer).
      The product of A*B is _____ (your answer).
      The square of A is _____ (your answer).
      The square of B is _____ (your answer).

EXTRA PRACTICE 2

Finding the Average

1. Write a progam to find the average of three numbers.
2. Have the program PRINT: The average is _____.
3. Add a program line to have the program PRINT the average of your # _____, your # _____, and your # _____ is your answer _____. Example: The average of 3, 4, and 8 is 5.

EXTRA PRACTICE 3

More Mathematical Operations

Write five separate programs to PRINT the answer to these problems (the answer should read 25 * 2 + 4 = 54, and so on.):

1. 25*2+4
2. $3^2 + 4 - 2$
3. $36 \div 4 * 5$
4. $28 + 4 * 6 \div 8$
5. $(18-2) \div 3 + 4\ (6*3) + 2^3$

EXTRA PRACTICE 4

Print Zones

Part I.

Write a program to PRINT the word "Leeds" in the following ways:

	ZONE 1	ZONE 2
1.	LEEDS	LEEDS
2.	LEEDS	
3.		LEEDS

Part II.

Using page 73, type in the information as shown (LEEDSPRIME)...and so on.

1. Count the number of characters in both zones. How many?
2. How many in zone 1 _____, zone 2 _____.

EXTRA PRACTICE 5

Area of Square and Volume of Cube

1. Write a program to solve the following problems. Label your answers.
   a. The side of a square is 27 inches. Find its area (area (A) = s^2).
   b. If the side of a cube is also 27 inches, find its volume (volume (V) = s^3).
2. Using INPUT statements, write a program to find the area of a square and volume of a cube.
   a. Solve the problems above (assume sides of square and cube are equal).
   b. Using different lengths for the side, RUN the program again (assume that the sides of the square and the cube are equal).

EXTRA PRACTICE 6

Printing Tables of Numbers, Squares, and Cubes

1. Write a program to generate the first 25 numbers and PRINT their squares on the same line.
   Example:
   ```
   1    1
   2    4
   3    9
   4   16
   and so forth
   ```
2. Write a program to generate the first 25 numbers and PRINT their cubes on the same line.
   Example:
   ```
   1    1
   2    8
   3   27
   4   64
   and so forth
   ```
3. Write a program to generate all the numbers from 20 to 1 and PRINT the numbers, and their squares and cubes, on the same line and in three columns.
   Example:
   ```
   20   400   8000
   19   361   6859
   18   324   5832
   and so forth
   ```

EXTRA PRACTICE 7

Printing Three Times and Nine Times Tables

1. Write a program to generate the three times table from $3 \times 1 = 3$ to $3 \times 12 = 36$. The printout should look exactly like this:
   ```
   3 * 1 = 3
   3 * 2 = 6
   3 * 3 = 9
   3 * 4 = 12
   and so forth
   ```
2. Write a program to generate the nine times table from $9 \times 1 = 9$ to $9 \times 12 = 108$.

EXTRA PRACTICE 8

Two-Dimensional Array

1. Suppose we have a class of ten students. The course grade is based upon three quizzes, and the results for the class are as follows:

Student #	1	2	3	4	5	6	7	8	9	10
Quiz #										
1	88	41	100	88	79	76	86	90	85	100
2	75	52	65	57	98	86	96	91	86	92
3	71	47	75	77	86	96	85	92	97	82

a. Write a program to PRINT the following information:

Student #	*Course Avg./Student*
1	?
2	?
3	?
4	?

Computer calculate and PRINT average

and so forth

Quiz #	*Class Avg./Quiz*
1	?
2	?
3	?

Computer calculate and PRINT average

HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

ISBN 0-8104-6179-X