I Speak BASIC to My COMMODORE 64™

Aubrey B. Jones, Jr.

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

HAYDEN

I Speak BASIC to My Commodore 64™

Aubrey B. Jones, Jr.

HAYDEN BOOK COMPANY
a division of Hayden Publishing Company, Inc.
Hasbrouck Heights, New Jersey

To Alyce, Aubrey III, and Adrienne

Production Editor: RONNIE GROFF
Production Assistant: LORI WILLIAMS
Developmental Editor: KAREN PASTUZYN
Art Director: JIM BERNARD
Book Design: JOHN M-RÖBLIN
Compositor: VAN GROUW COMPOSITION CO., INC.
Printed and bound by: COMMAND WEB OFFSET INC.

Library of Congress Cataloging in Publication Data

Jones, Aubrey B.
I speak BASIC to my Commodore 64.

1. Commodore 64 (Computer) — Programming. 2. BASIC (Computer program language) I. Title.
QA76.8.C64J66 1983b 001.64'2 83-22529
ISBN 0-8104-6172-2

3 4 5 6 7 8 9 PRINTING

84 85 86 87 88 89 90 91 92 YEAR

CONTENTS

PART 1

Hardware (The Machines)

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 Commodore 64 microcomputer.
5. To define and explain the terms hardware, software, microcomputer, microprocessor, RAM, ROM, processor, input unit, output unit, and memory.
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!

- Let's destroy some myths. First of all, despite what you might have heard, people control computers, people design them, people build them, people sell them, and, most of all, people tell them what to do (which is another way of saying that people "program" them).
- A computer program is a set of instructions that specify what the computer must do. Computer programmers write these instructions.

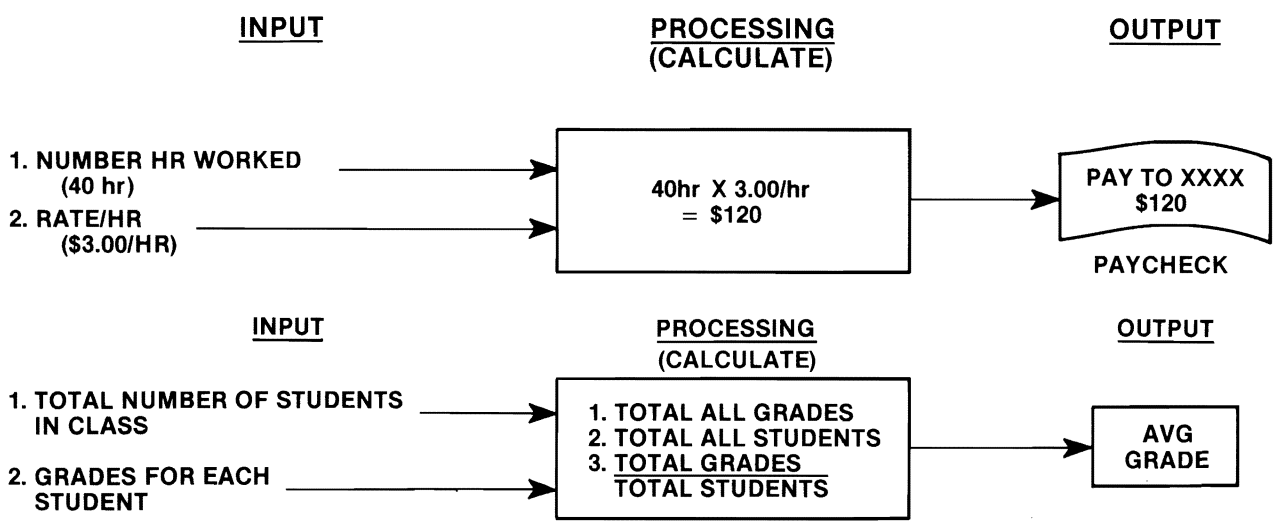
1.1

Box Diagram of Typical Data Processing Operation

1.2

Examples of Data Processing Operation

- Data are collections or representations of facts or instructions.
- Although a computer isn't necessary to perform a data processing operation, a computer system is capable of processing data at tremendous speeds.



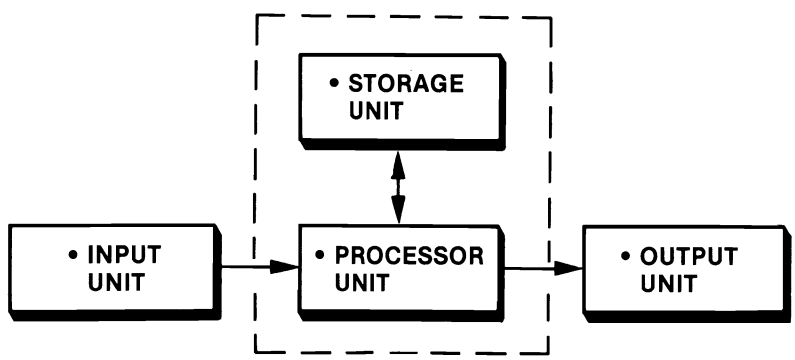
1.3

Terms You Should Know

- **HARDWARE**
  — The computer and computer-related equipment (the machines)
- **SOFTWARE**
  — The instructions for the computer (the program)

1.4

Box Diagram Showing Basic Parts of a Computer



1.5

Stores or Remembers

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

- Computers are controlled by the program which is in the main storage unit.

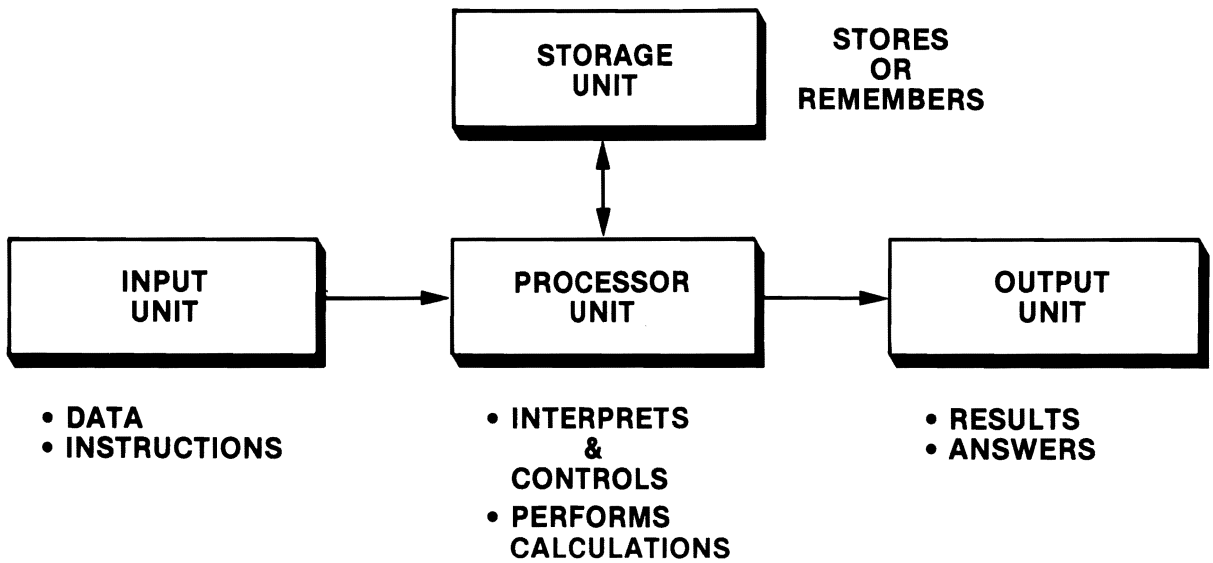
1.6

Interprets, Controls, and Calculates

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

1.7

Box Diagram of a Computer System



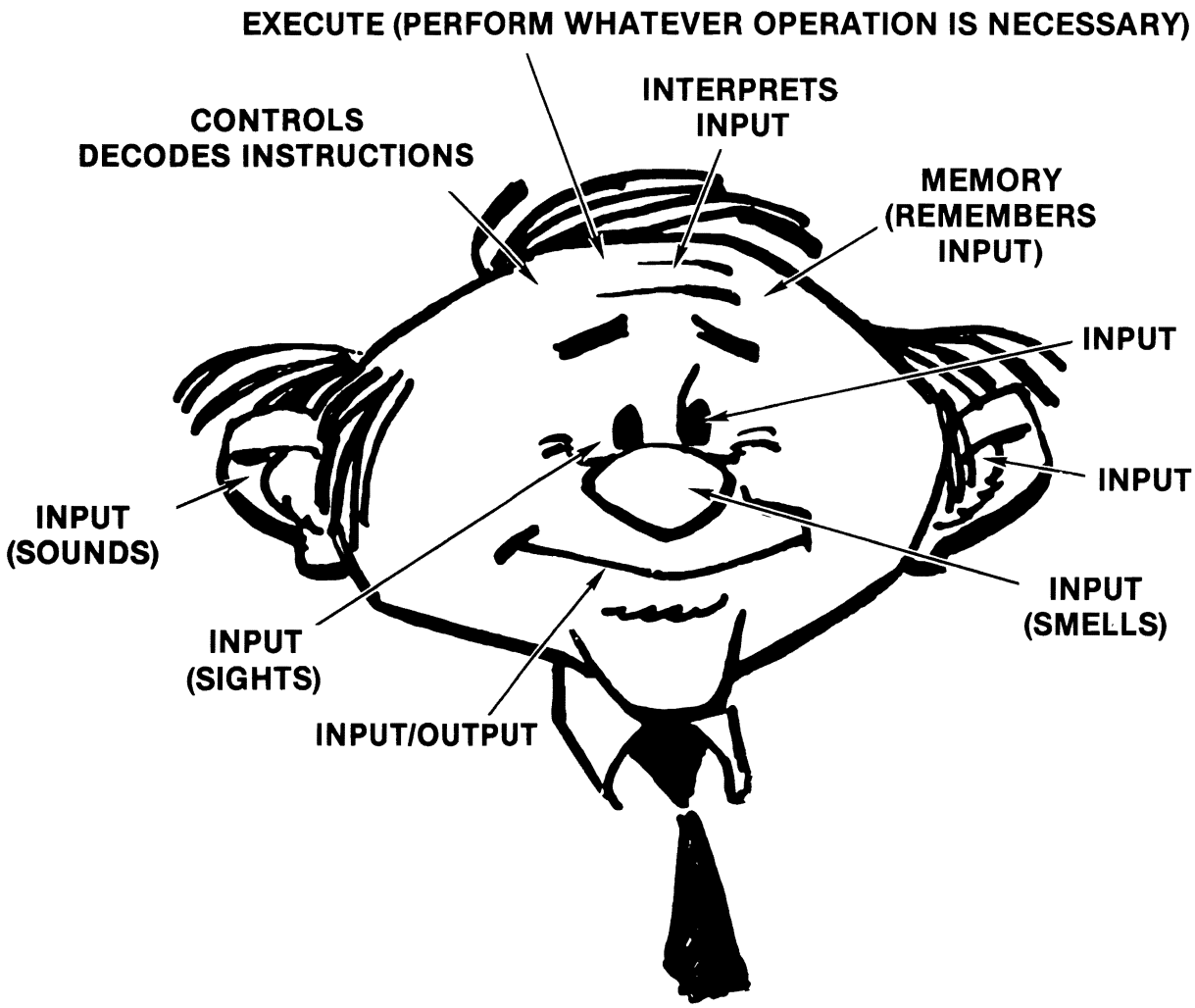
1.8

What We Have Learned

- Input → Provides instructions and data
- Storage → Stores or remembers (memory)
- Processor → Interprets, controls, and calculates
- Output → Provides answers and results

1.9

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



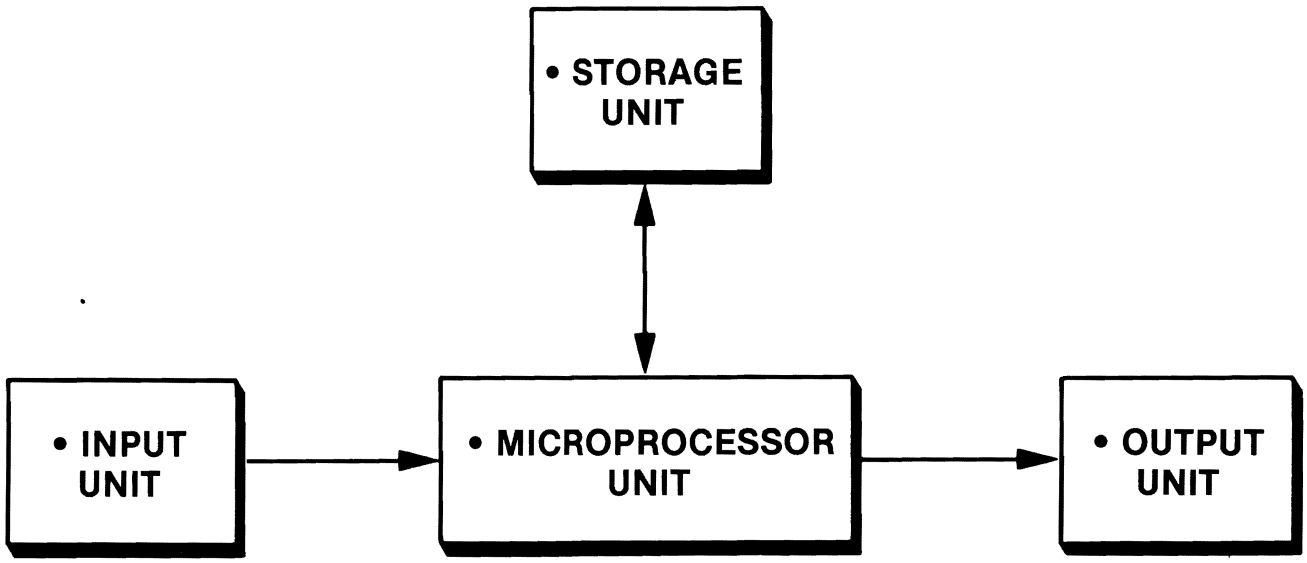
1.10

More Terms You Should Know

- **Microprocessor** = Very small processor.
- **Microcomputer** = Very small computer
- **RAM** = Random Access Memory
  — CAN be changed by the user
  — Information stored in RAM will be destroyed if power fails or is 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 (nonvolatile)
  — Control program (BASIC interpreter) stored here

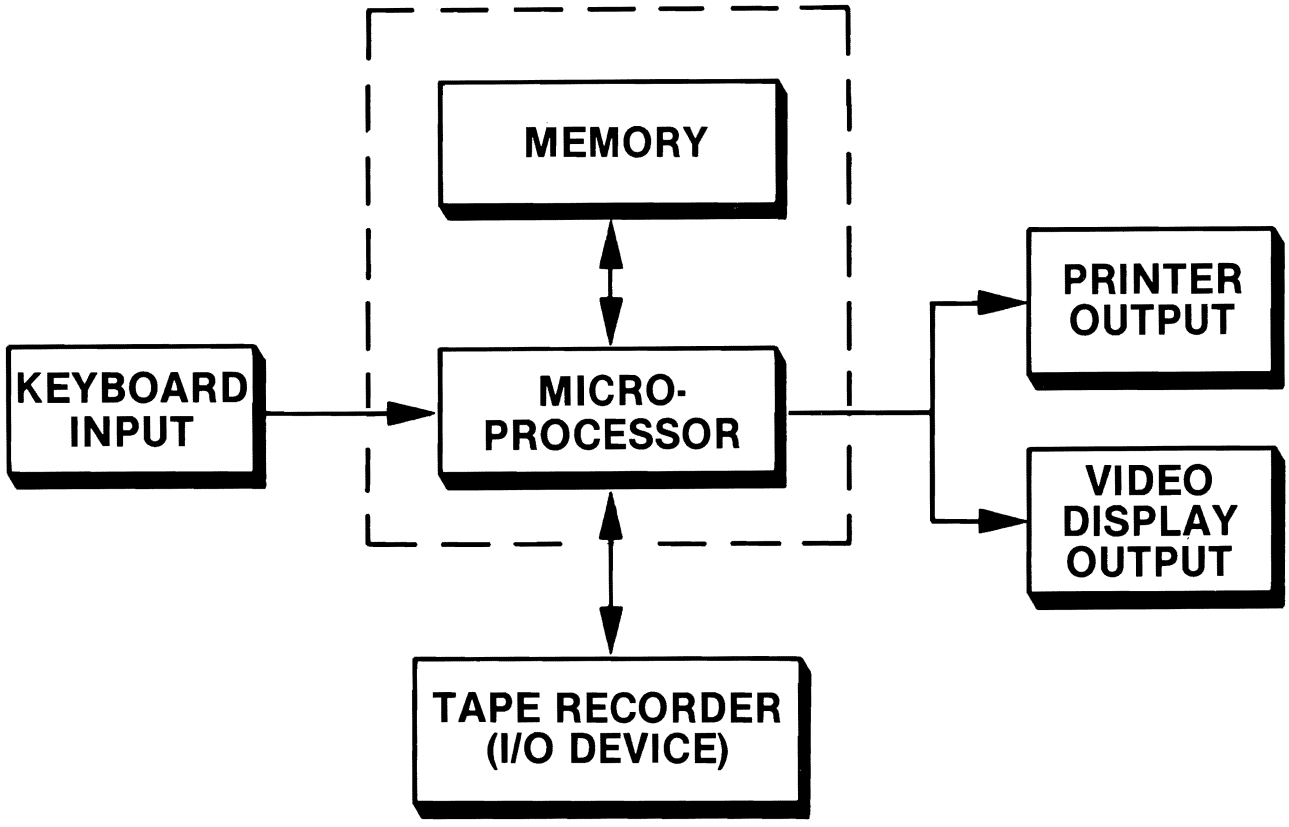
1.11

Box Diagram of a Microcomputer



1.12

Basic Components of the Commodore 64 Computer



1.13

Commodore 64 Computer System

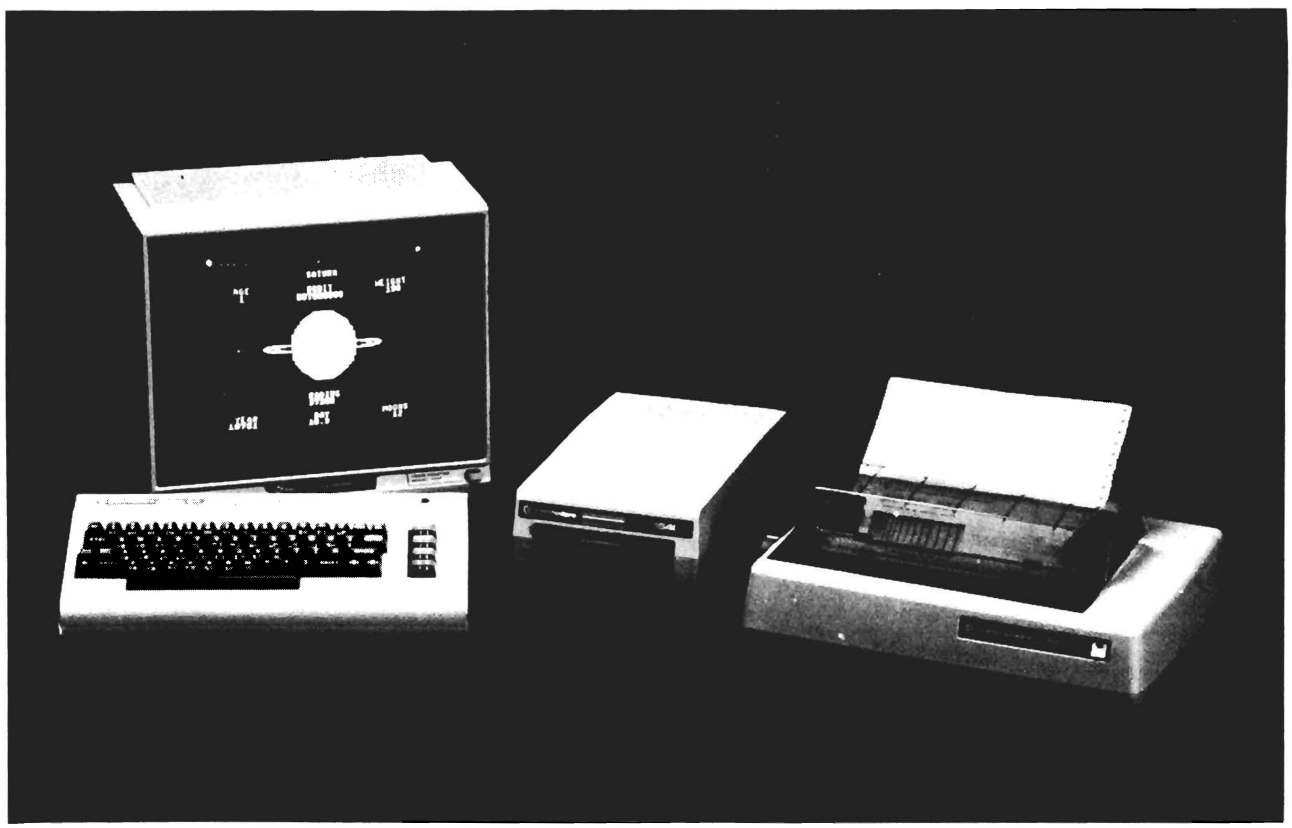


Photo Courtesy of Commodore Business Machines, Inc.

1.14

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 computer system.
   a. Label each box with the correct name.
   b. List the functions of each box.

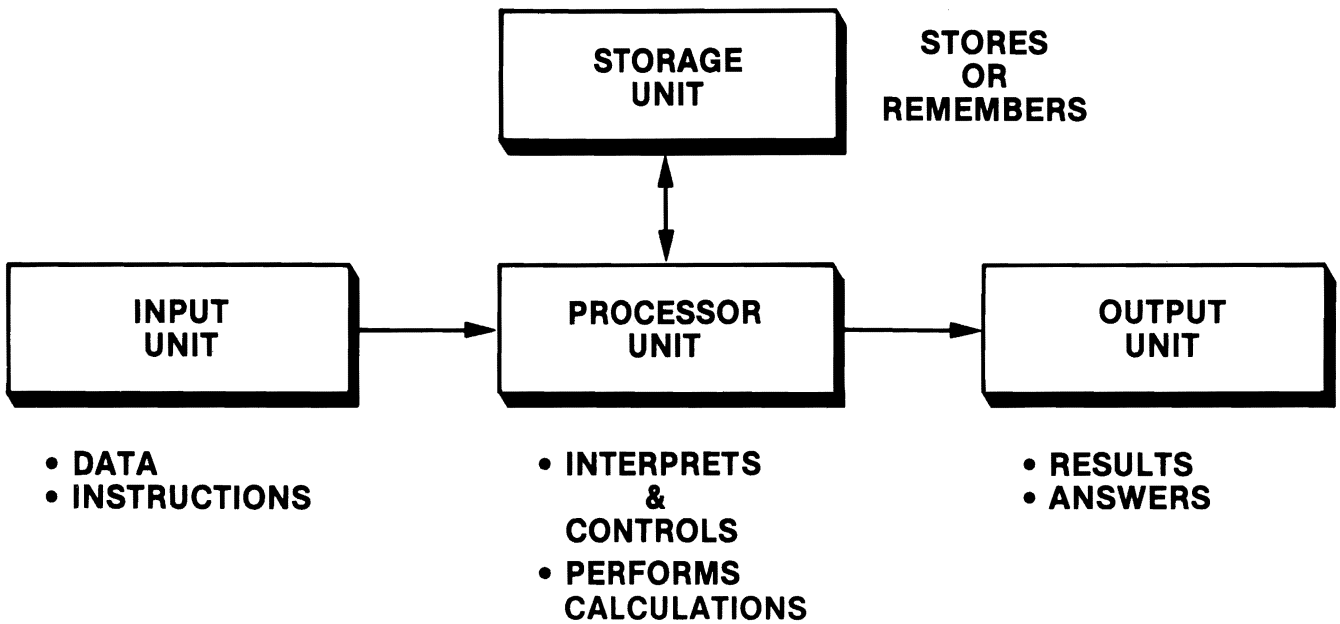
PART 2
Software (The "Program")

What You Will Learn

1. To define the terms hardware, software, BASIC, binary, and interpreter, 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 Commodore 64 keyboard.
6. How to connect and power up a Commodore 64 microcomputer.

REVIEW

Box Diagram of a Computer System



REVIEW

Terms You Should Know

- **HARDWARE**
  — The computer and computer-related equipment (the machines)
- **SOFTWARE**
  — The instructions for the computer (the program)

2.1

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 symbols: "Ø" and "1"
- A computer is capable of executing only machine language instructions.

2.2

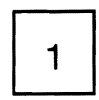
Machine Language: Bit

Bit = binary digit
Bit = smallest memory cell in a computer
Bit = "1" or "Ø"

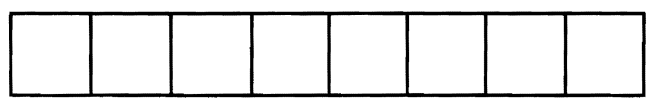
- Machine language instructions are stored in the main computer storage of the processor unit.
- These instructions are coded in bits.

2.3

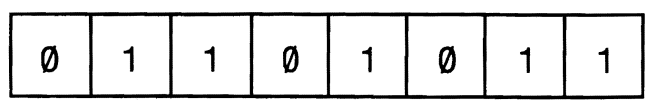
Machine Language: Byte



Memory Cell with 1 Bit



8 Memory Cells



8 Bits = 1 Byte

- A bit was found to be insufficient to store all the letters of the alphabet, special characters, and numbers needed to process data.
- A byte is a series of 8 bits.
- A byte is used to store a single letter, number, or character. For example, a byte might contain the binary equivalent of the letter "A" or the number "7."

2.4

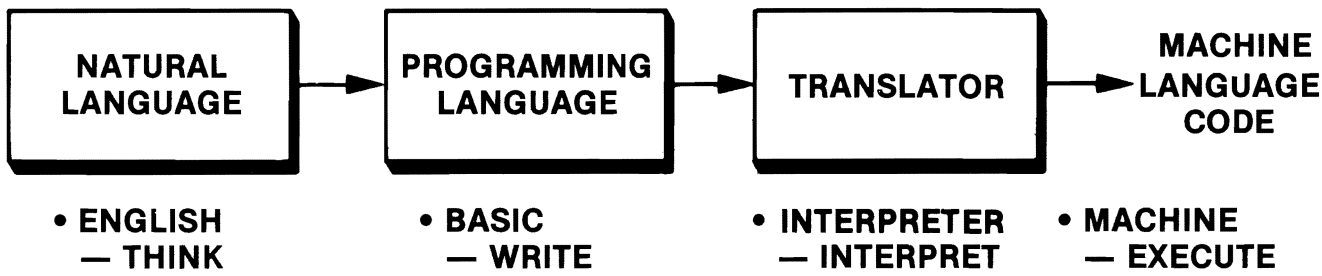
Machine Language: KByte

Byte = 8 Bits
K = 1000
KBytes = 1000 Bytes
KBytes = 8000 Bits

- More exactly, a byte = 2^{10} (1024) bits, but you need know only that K stands for "one thousand."
- Microcomputers use RAM as their main computer storage. A 32K microcomputer is one that has 32000 (or more exactly, 32768) bytes of RAM.

2.5

How Humans Talk to Computers



- Even though a computer can execute only machine language instructions, it is not necessary to learn machine language to communicate with a computer.
- People normally start to speak with their natural language, but we need a *programming language* to talk to a computer.
- A program, located in ROM in a microcomputer and called an *interpreter program*, interprets or translates a programming language into machine language.
- There are many programming languages, some of which are COBOL (**Co**mmon **B**usiness **O**riented **L**anguage), FORTRAN (**For**mula **Tran**slation), RPG (**R**eport **P**rogram **G**enerator), PL/1, Pascal, and BASIC (**B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode).

2.6

Terms You Should Know

- **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 Commodore 64

2.7

A Comparison between English and BASIC

To Program You Must Learn the Language First!

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

2.8

Learning a New Vocabulary

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

Key Words

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

Commands

- NEW
- LIST
- RUN
- CONT

2.9

Commands versus Statements

COMMANDS

— Executed as soon as you type them and press RETURN

STATEMENTS

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

2.10

A Program in BASIC

	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

Note:

Ⓐ 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.

2.11

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
- Normally are multiples of 5's, 1Ø's, or some other multiples to leave space for inserting new program lines between old ones

Note:

○ Computer will start executing at lowest numbered line unless told to start elsewhere.

○ Although it is perfectly legal to number program lines more closely (like 1, 2, 3, 4, etc.), don't do it!

2.12

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

2.13

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

2.14

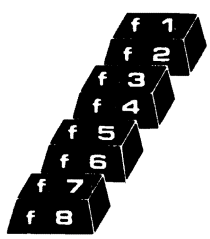
Commodore 64 Keyboard

Photo Courtesy Lou Odor

2.15

Special Function Keys on the Commodore 64 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 the 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 the SHIFT key with the 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 Commodore 64, however, but typically are used if an application program or cartridges containing special programs assign a function to these keys.



2.16

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 line. • [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 left 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. INSERT is obtained when the [SHIFT] key is held down.

2.17

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

KEY	FUNCTION
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 157 for more details.)
CTRL	• Stands for "Control." This key is used with color keys to select the colors that you create on the Commodore 64 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 Commodore 64; — 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 shorthand 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 Commodore 64 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.

2.18

Commodore 64 Power-Up Rules

YOUR ACTION	DISPLAY
1. Make certain the Commodore 64 micro-computer 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 video display and set the RF modulator to "Computer." (Make certain that channel selection on the television knob matches that on the computer.)	
4. Turn on the Commodore 64. The power switch is located on the right side of the Commodore 64.	
5. After a few seconds the message should appear on the screen as shown. • The blinking or flashing ■ is called the cursor. The cursor must be present in order to enter commands from the keyboard.	***COMMODORE 64 BASIC V2*** 64K RAM SYSTEM 38911 BASIC BYTES FREE READY ■ ←CURSOR
6. You are now ready to use Commodore 64 BASIC.	

Note:

○ If your Commodore 64 does not display the message shown above, turn off the power, wait a few seconds, and then turn power back on.

2.19

Two Modes of Commodore 64 (Important to Remember)

The Commodore 64 can operate in two modes, and it is important that you know which mode the computer is in.

1. Upper-Case/Full Graphics mode
2. Lower-Case/Text mode (including Left-Side Graphics)

- Upper-Case/Full Graphics mode
  — When you turn on the Commodore 64, you are automatically in this 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 Commodore 64 into the second mode. This permits you to use the Commodore 64 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 the C= key.

PRACTICE 2

Becoming Familiar with Your Commodore 64

Become familiar with the Commodore 64 microcomputer by doing the following:

1. Turn on the Commodore 64 using the Power-Up Rules (see page 29).
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 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 the CLR HOME key?

   _____
4. Move the cursor to the right by pressing the CRSR ⇦⇨ key several times. Then, press the CLR HOME key. What happened? _____
5. Move the cursor down by pressing CRSR ⇧⇩ key several times. Then press the 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

Your First Computer Program

What You Will Learn

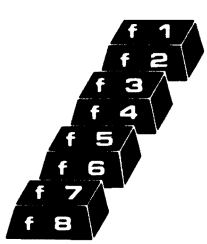
1. To enter and RUN your first BASIC program.
2. To explain the purpose and use of the following BASIC commands: **LIST, NEW, RUN.**
3. To explain the purpose and use of the following key words: **PRINT, REM, END.**
4. To explain the purpose and use of the following special function keys:

   [⇧ CRSR ⇩], [⇦ CRSR ⇨], [RETURN], [SHIFT], [RUN STOP], [CLR HOME], [INST DEL]

5. To explain the purpose and use of the following miscellaneous points:
   ■ cursor, "" (quotes), line numbers, power-up rules.

REVIEW

Special Function Keys on the Commodore 64 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 the 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 the SHIFT key with the 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 Commodore 64, however, but typically are used if an application program or cartridges containing special programs assign a function to these keys.

REVIEW

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 line. • 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 left 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. INSERT is obtained when the SHIFT key is held down.

REVIEW

Commodore 64 Power-Up Rules

YOUR ACTION	DISPLAY

1. Make certain the Commodore 64 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 video display and set the RF modulator to "Computer." (Make certain that channel selection on the television knob matches that on the computer.)
4. Turn on the Commodore 64. The power switch is located on the right side of the Commodore 64.
5. After a few seconds the message should appear on the screen as shown.
   - The blinking or flashing ■ is called the cursor. The cursor must be present in order to enter commands from the keyboard.
6. You are now ready to use Commodore 64 BASIC.

```
***COMMODORE 64 BASIC V2***
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY
■ ←CURSOR
```

Note:

○ If your Commodore 64 does not display the message shown above, turn off the power, wait a few seconds, and then turn power back on.)

3.1
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

Note:

○ Step 5 means turning the computer off when you have finished programming.

3.2
Typical Display Readout

```
1Ø          PRINT          "HELLO THERE"

2Ø          PRINT          "YOUR NAME"

3Ø          END

RUN
```

3.3

Writing Your First Computer Program

YOUR ACTION	DISPLAY
1. Before you start typing your program, always type NEW and press the RETURN key.	
2. Type the line exactly as shown:	10 PRINT "HELLO THERE NAME!" ■ Ⓐ Ⓑ
3. Use the SHIFT key to type the upper characters like the quotation marks (") and the exclamation point (!).	
4. Do *not* press the RETURN key yet!	
5. Go back and examine your typed line *very carefully.* Did you make a mistake? If you did, just press INST/DEL .	
6. Is everything OK? If it is, you can press RETURN . (This tells the computer to "look at" what you just typed in).	10 PRINT "HELLO THERE NAME!" ■
7. When the cursor appears as shown, the computer is saying, "It's your turn... I'm waiting for you."	

Note:

○ NEW is a command that erases any program that may have been in the computer's memory, and typing NEW is an important first step in programming.

Ⓐ Insert student's name.

Ⓑ The Commodore 64 has 40 columns across the screen. If the line you typed has more than 40 characters (including spaces, numbers, and symbols), the characters will "spill over" or "wrap around" to the next line automatically.

3.4

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: 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).	HELLO THERE NAME! READY ■

3.5
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 recognize a zero. Look at your keyboard closely.

3.6

Writing Your First Computer Program — Almost? (Correcting Errors)

PROBLEMS: (You forgot to follow instructions.)

1. **Missing Quotes (")**
   — You forgot to enclose everything after the word PRINT in quotation marks. Don't forget to use quotation marks if you want something printed!
2. **Too Many Quotation Marks**
   — You typed too many. That won't work either!
3. **Forgot the Key Word PRINT**
   — You forgot to type PRINT. How will the computer know to PRINT something if you don't tell it to?
4. **Forgot to Type the Line Number**
   — 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.

SOLUTION:

If you have already pressed RETURN, you must retype the entire line to correct your error.

- Type in the same line number you wish to change.
- Retype the line exactly as shown on the previous page. (But this time, be more careful!)
- Then, check the line over for errors.
- If everything is OK, don't forget to press RETURN! Pressing RETURN tells the computer to "look at" what you just typed and act accordingly.

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

3.7

Expanding Your Program

YOUR ACTION	DISPLAY
1. You now have a program in the computer (unless you turned it off. If you did, retype the line as shown):	
	1Ø PRINT "HELLO THERE NAME!" 2Ø PRINT "I'M GOING TO MAKE YOU A SUPERST AR!"
2. Type in Line 2Ø *exactly* as shown:	
3. Check your new Line (2Ø) *very carefully,* especially the quotation marks.	
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! I'M GOING TO MAKE YOU A SUPERSTAR! READY ■
7. If it did not work, check your program for errors.	

Go to next page.

3.8

Using the PRINT Statement for Spacing

YOUR ACTION	DISPLAY
1. Look at your video display. Would you like more space between the two lines? OK, this is how you do it.	HELLO THERE NAME! I'M GOING TO MAKE YOU A SUPERSTAR! READY ■
2. Type in a new line as shown and then press [RETURN].	15 PRINT
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 10 and 20.	HELLO THERE NAME! I'M GOING TO MAKE YOU A SUPERSTAR! READY ■
5. Observe that the PRINT statement (Line 15) was placed between Lines 10 and 20. Since you were smart enough to number your lines by 10'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.	

Note:

○ You might want to clear the screen from time to time to make it easier to read. To do this hold down the [SHIFT] key and then press [CLR/HOME] key.

3.9

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

Go to next page.

DISPLAY

```
5 REM THIS IS MY FIRST COMPUTER PROGRAM
```

```
HELLO THERE NAME!

I'M GOING TO MAKE YOU A SUPERSTAR!

READY
■
```

3.10

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 SUPERST AR!" 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 want to LIST only one line. Type LIST 20 and press [RETURN] and the screen will display:	20 PRINT "I'M GOING TO MAKE YOU A SUPERST AR!" 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 press [RETURN].	10 PRINT "HELLO THERE NAME!" 15 PRINT 20 PRINT "I'M GOING TO MAKE YOU A SUPERST AR!" READY ■

Go to next page.

Note:

○ So you can start with a clean display, hold down the [SHIFT] key while pressing [CLR/HOME] to clear the screen.

3.11

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 that the computer will know it is finished. However, the Commodore 64 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"
```

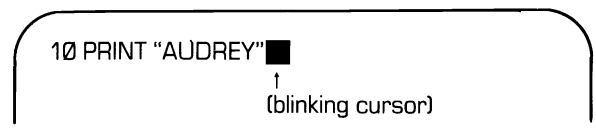
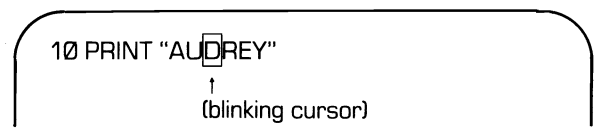
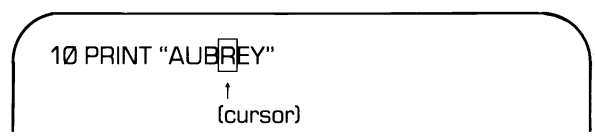
```
HELLO THERE NAME!

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

READY
■
```

3.12

Using the CRSR ⇦ Key to Save Retype Time

YOUR ACTION	DISPLAY
1. You typed Line 1Ø 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.	

Note:

- ◯ Type NEW and clear the screen (CLR/HOME) before you start.
- ◯ Use the CRSR ⇦ key on a line *only after* the quote is closed (i.e., the second set of quotation marks has been typed). If you try to use this key *before* the second set of quotes, you will see the character ⇦, in which case use the INST/DEL key to erase these characters.

3.13

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	LIST
• Backspace and delete	INST DEL	—
• Backspace without deleting characters	SHIFT and ⇦ CRSR ⇨	—
• Retype rest of line after correction	⇦ CRSR ⇨	—
• Reset computer (without destroying program 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).

SUMMARY

COMMANDS*

- CONT
- LIST
  — LIST MM
- NEW
- RUN

*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

Commodore 64 Power-Up Rules

SPECIAL FUNCTION KEYS

RUN STOP

CLR HOME

RETURN

SHIFT

INST DEL

⇧ CRSR ⇩

⇦ CRSR ⇨

Note:

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

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. REM statements to describe your program
   b. Spacing between each of the lines displayed (printed)
   c. An END statement
3. Type your program and enter it.
4. RUN your program.
5. LIST your program.

Note:

○ 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 REM 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 key word: **LET**.
4. To explain the purpose and use of the mathematic operators: multiply (*), divide(/), add (+), subtract (—), exponentiate or raise a number to a power(↑).
5. To explain the function and use of commas, semicolons, and PRINT zones.
6. To identify variables that can be used with Commodore 64 BASIC.

REVIEW

COMMANDS*

- CONT
- LIST
  — LIST MM
- NEW
- RUN

*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

Commodore 64 Power-Up Rules

SPECIAL FUNCTION KEYS

RUN STOP

CLR HOME

RETURN

SHIFT

INST DEL

⇧ CRSR ⇩

⇦ CRSR ⇨

Note:

◯ MM = Any line number (e.g., 10, 20, 30, etc.)

◯ If you don't understand everything on this page, stop! Go back over this section until you understand it thoroughly!

4.1

Math Operators

= (Equals)	* (Multiply)
+ (Add)	/ (Divide)
− (Subtract)	↑ (Exponentiation)

Note:

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

4.2

Order of Arithmetic Operations

- Multiply → Divide → Add → Subtract (left to right)
  — "My Dear Aunt Sally"
- If parentheses are used:
  — the innermost set of parentheses is simplified first, followed by each successive set outward.
  — the M.D.A.S. order is followed inside all sets of parentheses.

4.3

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
```

4.4

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]

Note:

○ The same answer can be worked out first on paper and then on the computer. If you type in

```
PRINT ( (6+4) * 2) /4
```

and press [RETURN], the computer will print out 5.

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)) = _____

Note:

○ You should work these out on paper. If you use the computer to check the answers, be sure to use the word PRINT and leave off the equals sign (=). For example, type

```
PRINT 2 ↑ 3 + 4 * 5 — 4/2 * 5
```

and press RETURN.

SUMMARY

Tips on Using Parentheses

- 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:
  — Innermost 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.

4.5

Numeric Variable Names Used with Commodore 64 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 numeric 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 numeric variable names.
  — The complete list of reserved words, which cannot be used in variable names, appears in the *Commodore 64 Programmer's Guide.*

4.6

Program for a Mathematical Operation

Line No.	Key Word	Other Part of Statement	
1Ø	LET	X = 5	RETURN
2Ø	LET	Y = 12	RETURN
3Ø	LET	Z = X*Y	RETURN
4Ø	PRINT	Z	RETURN
99	END		
RUN			

Note:

- LET is an optional key word for Commodore 64 BASIC. Some computers require you to use LET, however. Beware of this if you use another computer.
- RETURN is not part of the program. It is just a reminder to press it after each line.

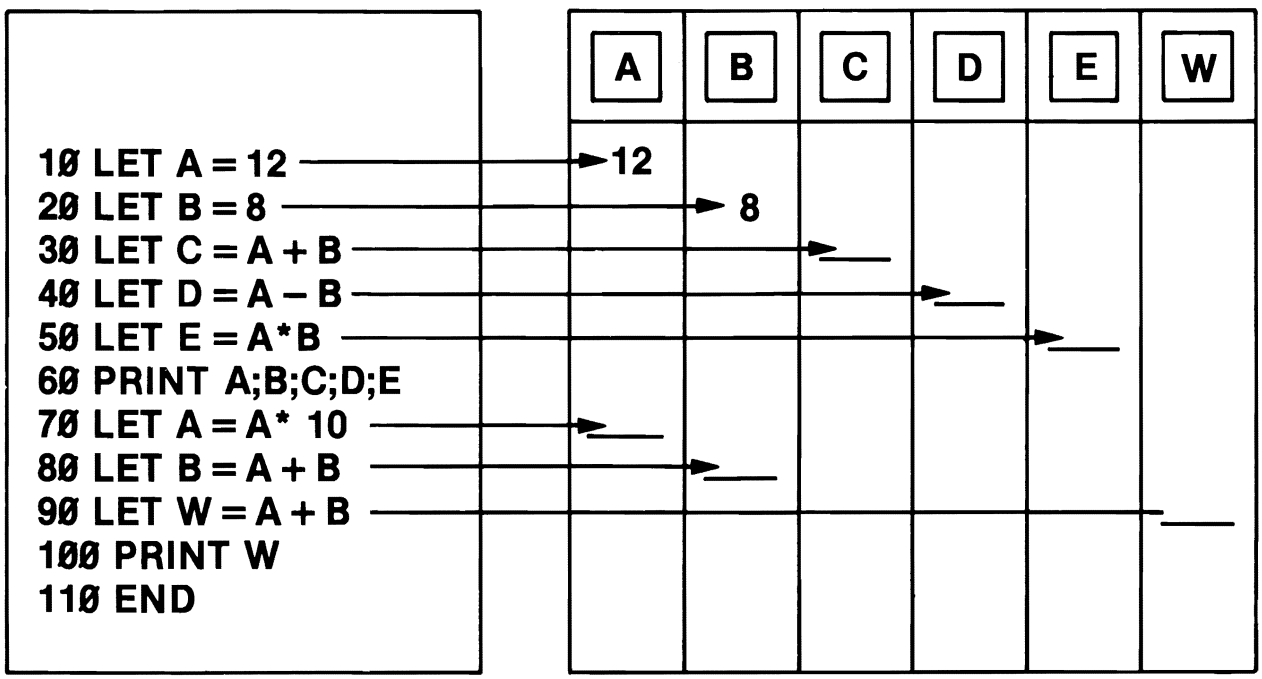
4.7

Analysis of the 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 to 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

EXERCISE 4-2

Assigning Numeric Values to Variables



4.8

A Mathematical Program: Area of Rectangle

YOUR ACTION	DISPLAY
1. Type NEW and then clear the screen (hold down the SHIFT key and then press CLR/HOME).	
2. Type and enter.	5 REM AREA OF A RECTANGLE PROBLEM 20 REM AREA (A) = LENGTH (L) X WIDTH (W) 30 LET L = 10 40 LET W = 5 50 LET A = L*W 60 PRINT A
3. Type RUN and press RETURN.	RUN 50 READY ■

Note:

○ 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.

4.9

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	50 THE AREA = 50 Ⓐ
3. Add Line 80 to read then press RETURN.	80 PRINT"THE AREA IS",A
4. Type RUN and press RETURN.	50 THE AREA = 50 THE AREA IS 50 Ⓑ
5. Add Line 90 to read then press RETURN.	90 PRINT"THE AREA IS"; A;"SQ.INCHES"
6. Type RUN and press RETURN.	50 THE AREA = 50 Ⓒ THE AREA IS 50 Ⓓ THE AREA IS 50 SQ.INCHES
7. LIST the program.	5 REM AREA OF A RECTANGLE PROBLEM 20 REM AREA(A)=LENGTH(L)XWIDTH(W) 30 LET L=10 40 LET W=5 50 LET A=L*W 60 PRINT A 70 PRINT"THE AREA =";A 80 PRINT"THE AREA IS",A 90 PRINT"THE AREA IS";A;"SQ.INCHES"

Note:

Ⓐ The semicolon in Line 70 caused the value of A to be PRINTed next to a label, namely, THE AREA =.

Ⓑ The commas in Line 80 caused the value of A to be PRINTed separated from a label.

Ⓒ The semicolons in Line 90 caused the value of A to be PRINTed between two labels. A space is automatically inserted before and after a numeric variable when PRINTed. This may not be true for other microcomputers.

Ⓓ There are four different outputs in this RUN since the program contained four different PRINT statements (Lines 6Ø, 7Ø, 8Ø, and 9Ø).

ASSIGNMENT 4-1

1. Write a program to find the area of a triangle.
   a. GIVEN: A = 1/2BH where B = 5, H = 1Ø
   b. Include REM statements
   c. Have the program PRINT "THE AREA="; (your answer); "SQ. FT."
2. Write a program to find the volume of a rectangular solid.
   a. GIVEN: V = LWH where L = 5, W = 1Ø, H = 2
   b. Include REM statements
   c. Have the program PRINT "THE VOLUME="; (your answer); "CUBIC IN."
3. Write a program to convert Fahrenheit to Celsius.
   a. GIVEN: C = (F − 32) × (5/9) where F = 75°
   b. Change the value of F to 45° and RUN the program again.
4. Write a program to convert Celsius to Fahrenheit.
   a. GIVEN: F = 9/5 × C + 32 where C = 2Ø°
   b. Change the value of C to 35° and RUN the program again.

Note:

◯ Remember that a formula must be written in proper computer form in your programs, i.e., A = 1/2*B*H for area of a triangle.

SUMMARY

PRINT and LET

- LET is an optional key word in Commodore 64 BASIC.
  — Other computers using BASIC might require the use of LET.
- 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 numeric variable and a "message."

Note:

○ 10 PRINT "A" tells the computer to print the letter A.
10 PRINT A tells the computer to print the value of the variable A (a number).

4.10
PRINT Zones

- The Commodore 64 is divided into four PRINT zones.
  — Each PRINT zone has 10 spaces up to 10 characters.
  — The Commodore 64 can display up to 40 characters per line ($4 \times 10 = 40$).
- 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 10, the computer will continue printing in the next PRINT zone. (Remember the maximum number of characters, symbols, and spaces per line cannot exceed 40.)

Zone 1 10 Spaces	Zone 2 10 Spaces	Zone 3 10 Spaces	Zone 4 10 Spaces
LEEDSPRIME	COMPUTERS*	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.

4.11

PRINT Zones and the Use of Commas

YOUR ACTION	DISPLAY
1. Type NEW and press [RETURN].	
2. Type Line 1Ø to read then press [RETURN].	
3. Type RUN and press [RETURN].	1Ø PRINT "ZONE 1", "ZONE 2", "ZONE 3", "ZONE 4" ZONE 1 ZONE 2 ZONE 3 ZONE 4 Ⓐ
4. Type Line 2Ø to read then press [RETURN].	
5. Type RUN and press [RETURN].	2Ø PRINT "ZONE 1",, "ZONE 3" Ⓑ ZONE 1 ZONE 2 ZONE 3 ZONE 4 ZONE 1 ZONE 3 Ⓒ

Note:

Ⓐ There are four (4) 1Ø-character PRINT zones per line (since 4 × 1Ø = 4Ø, the screen can display up to 4Ø characters per line).

Ⓑ Notice that there are two commas between Zone 1 and Zone 3.

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

4.12

Semicolon versus Comma

YOUR ACTION	DISPLAY
1. Type NEW and press [RETURN].	
2. Type exactly as shown, then press [RETURN].	10 PRINT "A";"SEMICOLON";"PACKS";"ITEMS"; "CLOSE";"TOGETHER"
3. Type exactly as shown, then press [RETURN].	20 PRINT "BUT A","COMMA", "LEAVES","SPACES"
4. Type RUN and press [RETURN].	ASEMICOLONPACKSITEMSCLOSETOGETHER BUT A COMMA LEAVES SPACES
5. Type NEW and press [RETURN].	
6. 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
7. Type RUN and press [RETURN].	5 10 15

Note:

○ On the Commodore 64, when the semicolon is used between two numeric *variables,* the computer automatically inserts one space between them. This might not be true with other computers, so beware!

SUMMARY

Use of the Semicolon and Comma

- The effect of the semicolon varies from computer to computer, but it is always true that a semicolon leaves less space between the output than a 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 closer together, use a semicolon
- EXCEPTION:
  A numeric variable will have a space before and after the number when PRINTed on the Commodore 64. For example, if your program contained the following line:

  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 there is space before and after the number 50 in the output, even though the PRINT statement contains semicolons.

PRACTICE 6

Perimeter of a Rectangle Program

Part I

1. Enter and RUN this program:

```
10 REM PERIMETER OF A RECTANGLE
20 REM PERIMETER=2*LENGTH+2*WIDTH
30 LET L=9
40 LET W=4
50 LET P=2*L+2*W
60 PRINT P
```

2. Add a new program line to include a label on your answer. For example, THE PERIMETER OF THE RECTANGLE IS 26 INCHES.
3. Add new program lines to PRINT the following:
   a. THE LENGTH OF THE RECTANGLE IS 9 INCHES.
   b. THE WIDTH OF THE RECTANGLE IS 4 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 this 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

5.1
Scientific Notation

- Scientists often express large numbers like 186,000 and small numbers like 0.00015 as the product of two numbers.
- 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. $5{,}000{,}000 = 5 \times 10^{6}$

5.2

Scientific Notation (Cont.)

Ordinary Notation		Scientific Notation		Scientific Notation in Commodore 64	Meaning
5,000,000,000	=	5×10^{9}	=	5E + 09	Add 9 zeroes after 5
.000005	=	5×10^{-6}	=	5E − 06	Shift decimal 6 places to left
.00000005	=	5×10^{-8}	=	5E − 08	Shift decimal 8 places left
5 (with 15 zeroes)	=	5×10^{15}	=	5E + 15	Add 15 zeroes after 5
5 (with 16 zeroes)	=	5×10^{16}	=	5E + 15	Add 16 zeroes after 5

- The Commodore 64 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:

○ Numbers with 10 or more digits are automatically converted to scientific notation on the Commodore 64.

○ Numbers that are less then 0.01 are automatically converted to scientific notation.

ASSIGNMENT 5-1

1. Enter and RUN the following program:

```
5 PRINT "♥"
10 LET A = 5000000000
20 LET B = .0000005
30 LET C = .00000005
40 LET D = 5000000000000
50 PRINT A,B,C,D
```

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

1. Quiz — written/lab
2. Open discussion with students on concerns and interest
3. Evaluation of students' performance
4. 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,ØØØ,ØØØ = 5×1Ø^{6}):

   a. 7,12Ø,ØØØ,ØØØ
   b. .ØØØØØ7
   c. .ØØØØØØØ8
   d. 6,1ØØ,ØØØ,ØØØ,ØØØ
   e. 8ØØØØØØØØØØØØØØØØ (16 zeroes)
   f. 8ØØØØØØØØØØØØØØØ (15 zeroes)
   g. 913,ØØØ,ØØØ,ØØØ
   h. 77,ØØØ,ØØØ,ØØØ,ØØØ
   i. 4Ø9,ØØØ,ØØØ,ØØØ,ØØØ
   j. 321ØØØØØØØ

2. Change the above numbers to computer scientific notation used in the Commodore 64. (Example: 5,ØØØ,ØØØ,ØØØ = 5E+ØØ9).

PART

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.

6.1

Relational Operators

- Relational operators allow a computer to compare one value with another.
  — The three relational operators include:

Symbol	Meaning	Examples
=	Equals	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

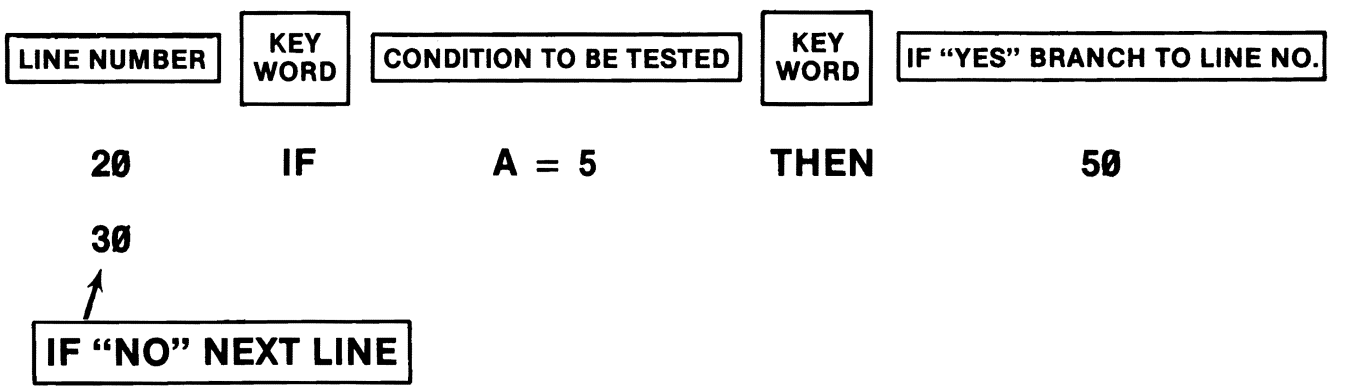
Note:

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

6.2

IF-THEN (Conditional Branching)

- 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:



6.3
Sample Program Using IF-THEN

- 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 a conditional branching statement?
  — What's the condition or test?

EXERCISE 6-1
IF-THEN

Given: A = 1Ø, B = 2Ø, C = 3Ø

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).

6.4
A Counting Program Using IF-THEN

YOUR ACTION	DISPLAY
1. Type in these lines.	10 LET J=0 20 LET J=J+1 30 PRINT J 40 IF J < 10 THEN 20
2. RUN the program.	1 2 3 4 5 6 7 8 9 10

EXERCISE 6-2

Modify above program to count to 50 by 5's.

6.5

IF-THEN Counter Program

```
10 J = 0
20 J = J + 1
30 PRINT J,
40 IF J < 4 THEN 20
50 END
```

Program Analysis

Initialize	Program Execution	"J" Counter Status	Display
	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,		
	40 IF J < 4 THEN 20		
4th Time	20 J = J + 1	[4] = 3 + 1	
	30 PRINT J		
	40 IF J < 4 THEN 20		
END	50 END		1 2 3 4

6.6

IF-THEN Counter Program (Stop-Action)

```
10 J = 0
20 J = J + 1
30 PRINT J
40 STOP
45 REM TYPE CONT TO CONTINUE
50 IF J < 4 THEN 20
60 END
```

Program Analysis

Initialize	Program Execution	"J" Counter Status	Display
	10 J = 0	[0]	
1st Time	20 J = J + 1	[1] = 0 + 1	
	30 PRINT J		1
	40 STOP		BREAK IN 40
	45 REM TYPE CONT TO CONTINUE		READY
	50 IF J < 4 THEN 20		
2nd Time	20 J = J + 1	[2] = 1 + 1	
	30 PRINT J		2
	40 STOP		BREAK IN 40
	45 REM		READY
	50 IF J < 4 THEN 20		
3rd Time	20 J = J + 1	[3] = 2 + 1	
	30 PRINT J		3
	40 STOP		BREAK IN 40
	45 REM		READY
	50 IF J < 4 THEN 20		
4th Time	20 J = J + 1	[4] = 3 + 1	
	30 PRINT J		4
	40 STOP		BREAK IN 40
	45 REM		READY

Initialize	Program Execution	"J" Counter Status	Display
	5Ø IF J < 4 THEN 2Ø		
END	6Ø END		

EXERCISE 6-3

GOTO (Unconditional Branching)

- Type and RUN this program:

```
1Ø PRINT "♥"
2Ø PRINT "YOUR NAME";
3Ø GOTO 2Ø
```

- What happened?
  - — Do you know how to stop the program? (What about the RUN/STOP key?)
  - — What does Line 3Ø tell the computer to do?
  - — Are there any tests or conditions to be satisfied in Line 3Ø before it does what it has to do?
  - — Do you understand now why the GOTO statement is called an unconditional branching statement?

Note:

○ Line 1Ø clears the screen.

EXERCISE 6-4
GOTO/IF-THEN

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

```
10 PRINT "WELCOME TO LEEDS MIDDLE SCHOOL"
20 GOTO 60
25 PRINT
30 PRINT "HELLO SUPERSTAR"
35 END
40 PRINT "COMPUTERS ARE MY THING"
50 GOTO 100
60 IF A = 5 THEN 140
70 PRINT "COMPUTER WORKSHOP"
80 GOTO 40
90 REM A TRICKY PROGRAM
100 LET A = 5
110 GOTO 60
120 PRINT "AND I'M A SUPERSTAR!"
130 GOTO 25
140 PRINT "COMMODORE 64 MICROCOMPUTER"
150 PRINT "I CAN DO IT TOO"
160 PRINT "I SPEAK BASIC"
170 GOTO 120
```

ASSIGNMENT 6-1

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

6.7

What We Have Learned

- 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 EQUAL 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. Write a program that assigns a value to variables A and B and prints either "A IS GREATER THAN B" or "B IS GREATER THAN A."
2. Change the values of A and B 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 four columns.

   Example:

20	21	22	23
24	25	26	27

   and so forth

PART 7

The INPUT Statement

What You Will Learn

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

7.1

Description of the 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.

7.2

Example of the INPUT Statement

YOUR ACTION	DISPLAY
1. Type NEW and press [RETURN].	
2. Type and enter Lines 5 and 10 as shown.	5 PRINT "THE NUMBER I'M THINKING OF IS" 10 INPUT A
3. Type RUN and press [RETURN].	THE NUMBER I'M THINKING OF IS ? ■ Ⓐ
4. Enter a number (e.g., type 5 and enter).	THE NUMBER I'M THINKING OF IS ? 5 READY ■
5. RUN this program several times to get the feel of it.	

Note:

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

7.3

The INPUT Statement with Built-In PRINT

YOUR ACTION	DISPLAY
1. Type NEW and press RETURN.	
2. Type in the program as shown.	5 PRINT "WHAT IS THE NUMBER" 1Ø INPUT A 2Ø PRINT "THE NUMBER WAS";A
3. RUN the program and enter a 5 when the program stops for the INPUT.	WHAT IS THE NUMBER ? ■ THE NUMBER WAS 5 READY
4. Type Line 5 to read:	5 PRINT "WHAT IS THE NUMBER";
5. RUN the program again; enter a 7 when the program stops for the INPUT.	WHAT IS THE NUMBER? ■ THE NUMBER WAS 7 READY Ⓐ
6. Change Line 5 to read:	5 INPUT "WHAT IS THE NUMBER";A
7. Delete Line 1Ø by typing 1Ø and pressing RETURN.	
8. RUN the program and enter a 9 when the program stops for the INPUT.	WHAT IS THE NUMBER? ■ THE NUMBER WAS 9 READY
9. LIST the program and RUN it several times, entering a different number each time.	LIST 5 INPUT "WHAT IS THE NUMBER";A 2Ø PRINT "THE NUMBER WAS";A READY Ⓑ

Note:

◯ An INPUT statement has a built-in PRINT feature, allowing you to combine a PRINT with an INPUT for the message you want!

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

Ⓑ This two-line program produces the same results as the three-line program:

```
5 PRINT "WHAT IS THE NUMBER";
10 INPUT A
20 PRINT "THE NUMBER WAS";A
```

7.4

Area of Rectangle Program (Using INPUT Statements)

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

7.5

Area of Rectangle Problem Revisited (Using INPUT Statements)

YOUR ACTION	DISPLAY
1. Type in program Lines 1Ø through 6Ø as shown.	1Ø REM AREA OF A RECTANGLE PROBLEM 2Ø INPUT "THE LENGTH IS"; L 3Ø INPUT "THE WIDTH IS"; W 4Ø A = L * W 5Ø PRINT "THE AREA IS"; 6Ø PRINT A Ⓐ
2. Type RUN and press RETURN. 3. Type in the length (say 1Ø) and enter. 4. Type in the width and press RETURN. 5. What is your answer?	THE LENGTH IS? 1Ø THE WIDTH IS? ■ Ⓑ

Note:

Ⓐ 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 program to do the following (using INPUT statements):
  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

7.6

What We Have Learned

- Trailing semicolon hooks two lines together.
- An INPUT statement causes the computer to stop and wait for an input from the keyboard.
- An INPUT statement can have a built-in message to tell you what to enter.

  For example:

  ```
  1Ø INPUT "YOUR AGE"; A
  ```

7.7

Numeric versus String Variables

Numeric Variable		Declaration Character*		String Variable
A	+	$	=	A$
A1	+	$	=	A1$
AB	+	$	=	AB$
AZ	+	$	=	AZ$

Note:

⊛ Adding the string declaraton character ($) to the numeric variable allows you to use any numeric variable as a string variable.

7.8

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."

EXERCISE 7-1

YOUR ACTION	DISPLAY
1. Type and enter.	5 PRINT "♥" 1Ø INPUT "YOUR FIRST NAME"; A$ 2Ø INPUT "YOUR MIDDLE NAME"; B$ 3Ø INPUT "YOUR LAST NAME"; C$ 4Ø PRINT A$;" ";B$;" ";C$ 5Ø INPUT "YOUR FULL NAME"; D$ 6Ø PRINT D$ Ⓐ
2. RUN.	YOUR FIRST NAME? AUBREY YOUR MIDDLE NAME? BRIGHT YOUR LAST NAME? JONES AUBREY BRIGHT JONES YOUR FULL NAME? AUBREY BRIGHT JONES AUBREY BRIGHT JONES READY ■

Note:

Ⓐ String variables can be printed together.

◯ To insert a space between string variables, you must print a space enclosed in quotes between the variables.

◯ A semicolon alone will not cause a space to be printed between string variables.

ASSIGNMENT 7-2

1. RUN and analyze the following program:

```
10 INPUT "YOUR NAME IS"; A$
20 INPUT "YOUR HOUSE NUMBER"; A
30 INPUT "YOUR STREET NAME"; 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 didn't Line 60 contain quotes? (60 PRINT A; " " ; B$)
   c. Why didn't we use the $ symbol (or string declaration character) with A and B in Lines 20 and 40?

SUMMARY

String Variables

- String variables can be assigned to indicate letters, words, and/or combinations of letters, numbers, and special characters.
- It is possible to contain up to 255 characters per string variable.
- String variables can be printed together.
- In a PRINT statement, use " " marks containing a space between string variables to separate them.

PRACTICE 11

Perimeter of a Rectangle Program (Using INPUT Statements)

1. Enter and RUN this program:

```
10 REM PERIMETER OF RECTANGLE PROBLEM
20 INPUT "WHAT IS THE LENGTH";L
30 INPUT "WHAT IS THE WIDTH";W
40 P = 2*L+2*W
50 PRINT "THE PERIMETER IS";P
```

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 the year you were born.
   b. INPUT how many brothers you have.
   c. INPUT how many sisters you have.
2. PRINT each with the proper labels. Example:

   I WAS BORN IN _____.
   I HAVE _____ BROTHERS.

PRACTICE 13
String Variables

Part I

1. Enter and RUN the following program:

```
10 INPUT "WHAT IS YOUR NAME";A$
20 INPUT "WHAT IS YOUR HOUSE NUMBER";A
30 INPUT "WHAT IS YOUR STREET NAME";B$
40 INPUT "WHAT IS YOUR ZIP CODE";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 didn't we use the $ 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 favorite subject.
   c. Your favorite food.
   d. Your favorite movie star.
   e. Your favorite color.
   f. Your zodiac sign.

PART 8

Using the Calculator Mode

What You Will Learn

1. To review the mathematical operators.
2. To review the order of operations using the M.D.A.S. rule.
3. To explain the purpose and use of the command: **CLR**.
4. How to use the Commodore 64 in the calculator mode using variables.

REVIEW

Math Operators

= (Equals)	* (Multiply)
+ (Add)	/ (Divide)
— (Subtract)	↑ (Exponentiation)

Note:

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

REVIEW

Order of Arithmetic Operations

- Multiply → Divide → Add → Subtract (left to right)
  — "My Dear Aunt Sally"
- If parentheses are used:
  — the innermost set of parentheses is simplified first, followed by each successive set outward.
  — the M.D.A.S. order is followed inside all sets of parentheses.

REVIEW

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]
```

REVIEW

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]

Note:

○ The same answer can be worked out first on paper and then on the computer. If you type in

```
PRINT ( (6+4) * 2) /4
```

and press [RETURN], the computer will print out 5.

8.1

Calculator Mode ("Immediate Mode")

- Use PRINT followed by a mathematical expression to have the Commodore 64 act like a calculator.
- You do not use a line number to operate the calculator mode.
- You can use variables in the calculator mode.
- You should type CLR before using variables in the calculator mode. CLR "clears" all the values variables may have.

ASSIGNMENT 8-1

- Type in each of the following and record the computer's response after you press RETURN:

YOU TYPE	RESPONSE
PRINT 4+5	_____
PRINT 6*8−3	_____
PRINT 4+5*4	_____
PRINT 7/2*3−1	_____
PRINT 2↑4−5	_____
PRINT (24+3)*(40−20)	_____
PRINT (30−5)−7+(3*8)	_____
PRINT 4↑3−(34−30)↑2	_____

ASSIGNMENT 8-2

- Type in each of the following and record the computer's response after you press [RETURN]:

YOU TYPE	RESPONSE
CLR	_____
PRINT A	_____
PRINT B	_____
A=5	_____
B=12	_____
C=4	_____
PRINT A+B–C	_____
PRINT C*A–B	_____
PRINT C↑A–B	_____
PRINT A*B–B*A	_____
CLR	_____
PRINT A	_____
PRINT B	_____
PRINT C	_____

8.2
What We Have Learned

- The command CLR "clears" the values of all variables. Numeric variables will then have a value of Ø.
- To use the calculator mode, type PRINT followed by a mathematical expression and press RETURN. The answer to the mathematical expression will then be printed.
- Variables can be assigned values in the calculator mode.
- You can evaluate expressions using variables in the calculator mode.
- The Commodore 64 evaluates expressions in the calculator mode using the M.D.A.S. rule.

PRACTICE 14
Calculator Mode

Part I

1. Use the Commodore 64 in calculator mode to solve the following:
   a. 25*8/2
   b. (25+6)−7+(2*4)
   c. 7/2*5*2↑4

Part II

1. Assign the following values in the calculator mode (remember to CLR):
   a. A=6
   b. B=2
   c. C=1Ø
   d. D=3
2. Use the calculator mode to solve the following with the values from 1:
   a. A*B−C+D
   b. D↑B*A−C
   c. (A+D)*C↑B

PART

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.

9.1

The Cassette Recorder as an I/O Device

- 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 (SAVE) it on 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 save only your program on cassette (not the program output).
○ Refer to the *Commodore 64 Programmer's Guide* for tips on using the recorder.

9.2

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.

9.3
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 the next program found on the tape.
VERIFY "TEST"	Search for the program named "TEST" on the tape cassette recorder and VERIFY it.
A$ = "TEST 2" **VERIFY A$**	Search for the program on the tape assigned the string variable A$ and VERIFY it.

9.4
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 the program on the tape named "TEST" and LOAD it.
A$ = "TEST 2" **LOAD A$**	Search for the program on the tape named "TEST 2" and LOAD it.
LOAD "Z"	If the program named Z is not on the tape, this command will produce a list of all the programs on the tape. For example: FOUND PROGRAM "TEST" FOUND PROGRAM "TEST 2" FOUND PROGRAM "GAMES"

9.5

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

YOUR ACTION	DISPLAY
1. Type and enter the program shown.	10 PRINT "THIS IS A TEST PROGRAM" 20 FOR J = 1 TO 25 30 PRINT J, : NEXT ■
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. Simultaneously press RECORD and PLAY levers on tape recorder. 7. Press STOP lever on tape recorder. 8. Rewind tape using REW lever.	OK SAVING PROGRAM 1 READY ■ Ⓑ

Note:

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

Ⓑ Screen will go blank for a few seconds and then you will see the message shown.

9.6

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

Note:

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

Ⓑ The screen will go blank several times before the complete message as shown is displayed.

9.7

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.	
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 TEST PROGRAM" 20 FOR J = 1 TO 25 30 PRINT J, : NEXT READY ■

Note:

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

Ⓑ The screen will go blank several times before the complete message as shown is displayed.

EXERCISE 9-1
Using LOAD "Z" AND SHIFT RUN/STOP KEYS

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 FIRST 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. Simultaneously press RECORD and PLAY levers on the tape recorder.	OK SAVING PROGRAM 1 READY ■
4. VERIFY the program. a. Rewind the tape. b. Type VERIFY command.	VERIFY "PROGRAM 1" ■
c. Press RETURN key.	PRESS PLAY ON TAPE
d. Press PLAY on tape recorder. e. Press STOP on tape recorder.	OK SEARCHING FOR PROGRAM 1 FOUND PROGRAM 1 VERIFYING OK READY ■
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. d. Press STOP on tape recorder.	OK SAVING PROGRAM 2 READY ■
7. VERIFY Program 2. a. Rewind tape. b. Type VERIFY command.	VERIFY "PROGRAM 2" ■

EXERCISE 9-1 (Cont.)

YOUR ACTION	DISPLAY
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 OK READY ■
e. Press STOP on tape recorder.	
8. Type and enter program shown.	300 PRINT "THIS IS THE THIRD TEST PROGRAM" 399 END
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	

Note:

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

EXERCISE 9-1 (Cont.)

YOUR ACTION	DISPLAY
11. LOAD "Z" a. Rewind tape. b. Type LOAD 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 MY FIRST TEST PROGRAM

Note:

Ⓐ 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 the same problem.
   (That is, you should use an INPUT statement for the total enrollment, number of freshmen, number of sophomores, number of juniors.)

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 an INPUT statement for the weight of each 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 1000 bulbs were left?

PART

Using FOR-NEXT...STEP Statements

10

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.

10.1

FOR-NEXT Statement

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

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 READY ■

10.2

FOR-NEXT...STEP Loop

YOUR ACTION	DISPLAY
1. Retype and enter Line 10 of resident Ⓐ program as shown. 2. Type RUN and press RETURN.	10 FOR J = 1 TO 10 STEP 3 Ⓑ AUBREY 1 AUBREY 4 AUBREY 7 AUBREY 10 READY ■

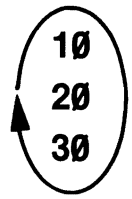
Note:

Ⓐ 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).

10.3

Example of Program Statements Using 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
```

10.4
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).

10.5

Comparison of Program Loops (Listings)

A. GOTO (Unconditional Loop)

```
5 PRINT "♥"
10 LET J = 0
20 LET J = J + 1
30 PRINT "AUBREY" ; J
40 GOTO 20
```

- This program loops forever (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
```

- 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
```

- This program loops 6 times.

10.6

Comparison of Program Loops (Outputs)

A. "DUMB LOOP"*	B. "SMART LOOP"	C. "SMART LOOP"
AUBREY 1	AUBREY 1	AUBREY 1
AUBREY 2	AUBREY 2	AUBREY 2
AUBREY 3	AUBREY 3	AUBREY 3
AUBREY 4	AUBREY 4	AUBREY 4
AUBREY 5	AUBREY 5	AUBREY 5
AUBREY 6	AUBREY 6	AUBREY 6
AUBREY 7		
AUBREY 8		
AUBREY 9		
AUBREY 10		
AUBREY 11		
AUBREY 12		
AUBREY 13		
AUBREY 14		
AUBREY 15		
AUBREY 16		

Note:

(*) Press the RUN/STOP key to get out of this loop.

SUMMARY
FOR-NEXT...STEP

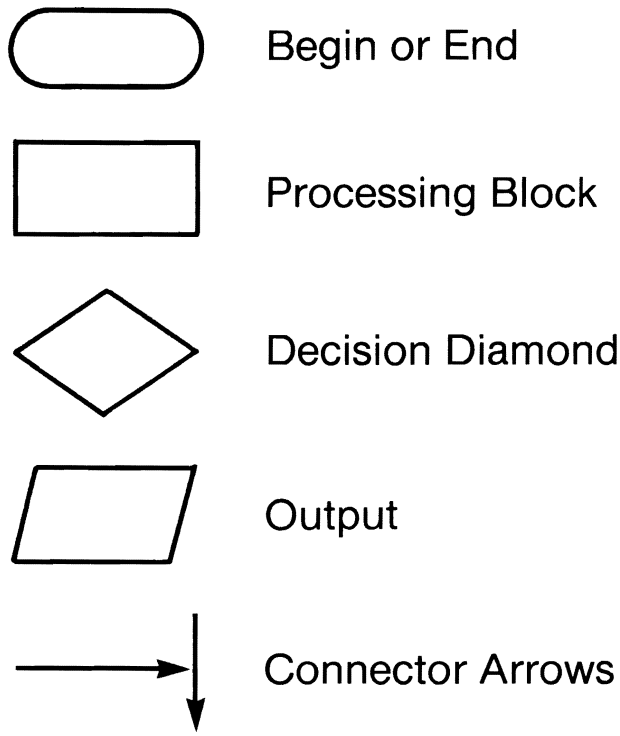
- FOR-NEXT is always used as a pair.
- If the key word "STEP" is not used, the increment of 1 is assumed.
- If the STEP has a negative value, the counter is decremented. For example:

```
20 FOR J = 10 TO 1 STEP -1
```

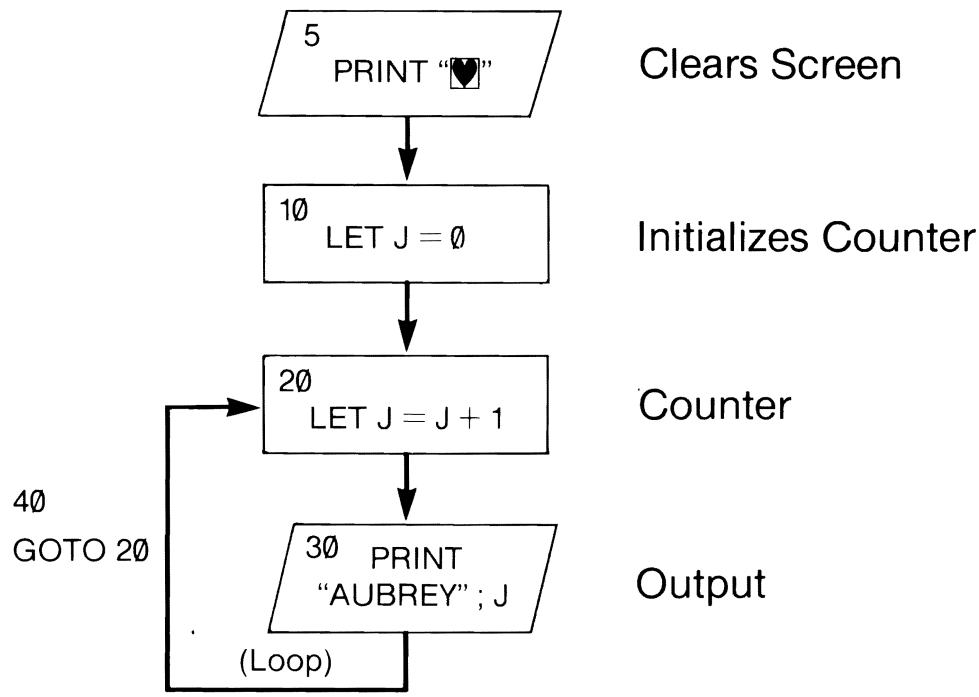
- If the STEP has a positive value, the counter is incremented. For example:

```
20 FOR J = 4 TO 10 STEP 2
```

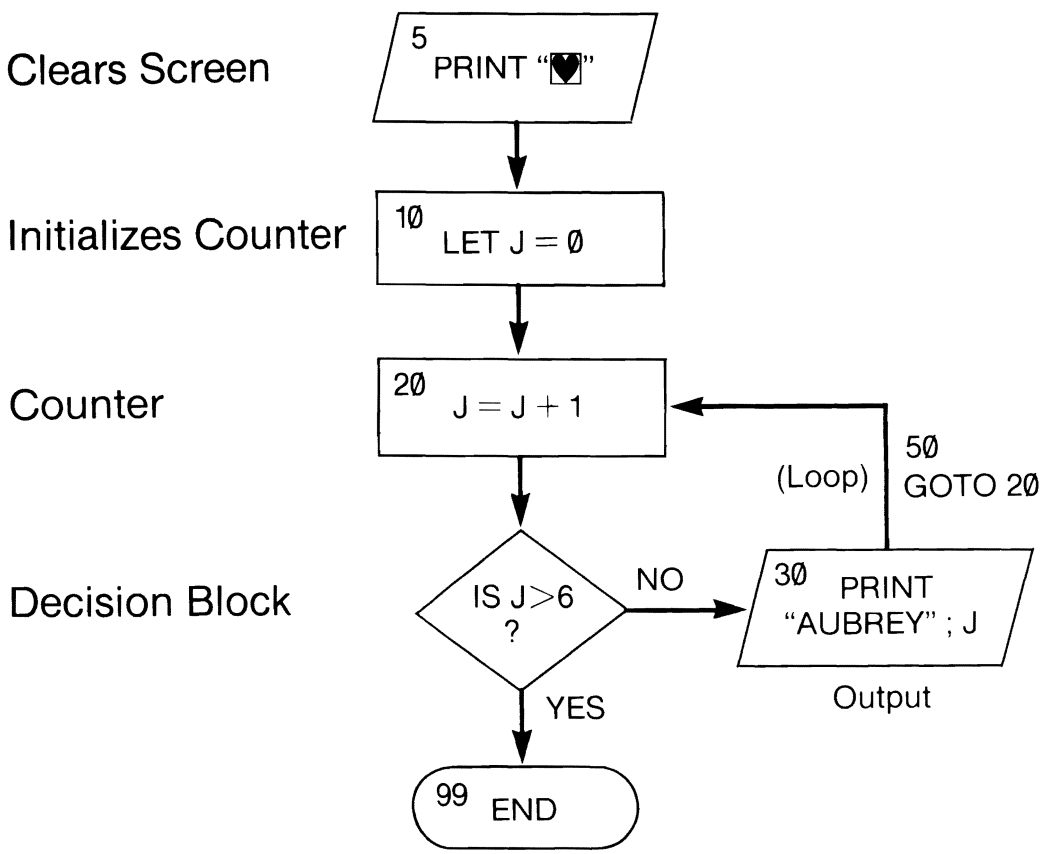
10.7
Flowchart Symbols



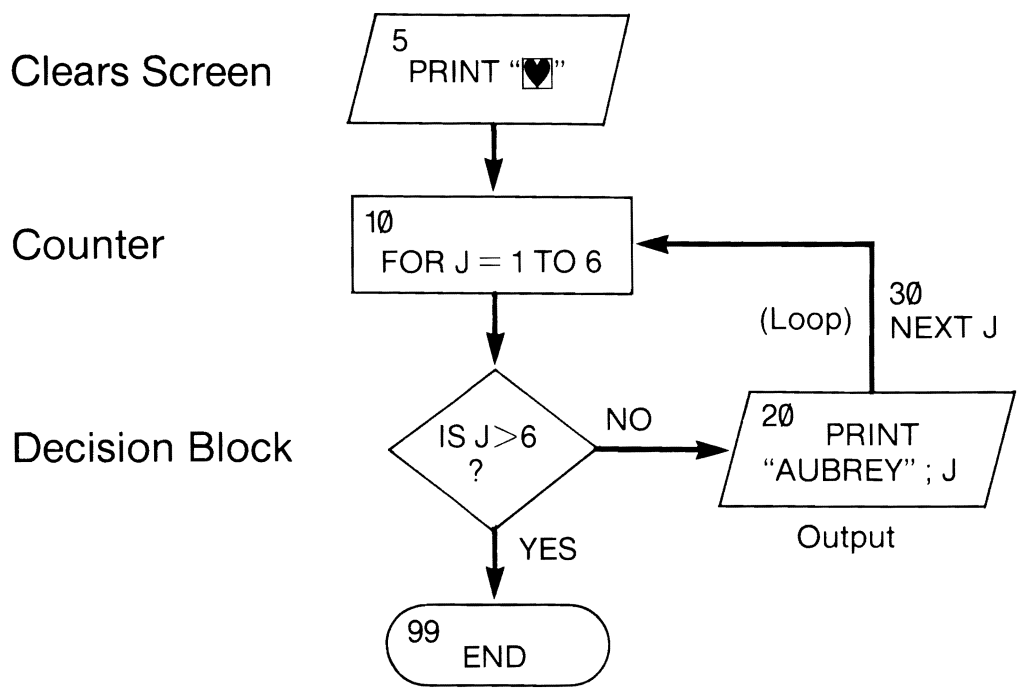
10.8
GOTO Loop



10.9
IF-THEN Loop



10.10
FOR-NEXT Loop



Note:

○ FOR-NEXT work together as a counter.

10.11
Timer Loop

- The Commodore 64 can do approximately 7ØØ FOR-NEXT loops per second.
- Example:

```
5 REM 1Ø SECOND TIMER PROGRAM
1Ø PRINT "TIMER PROGRAM COUNTING"
2Ø FOR X = 1 TO 7ØØØ
3Ø NEXT X
4Ø PRINT "TIMER PROGRAM ENDED"
```

- You don't believe the Commodore 64 can count? Well, try it! Type in the above program and RUN. Don't forget to use your watch!

ASSIGNMENT 10-1

1. Type in and RUN the following program:

```
5 PRINT "♥"
10 PRINT "INPUT A VALUE N" : PRINT:PRINT
15 INPUT "ENTER 1500, 2500, 3500, OR 7500"; N
20 PRINT "♥"
25 PRINT "THIS IS A DEMONSTRATION OF"
30 PRINT:PRINT
35 FOR J=1 TO N: NEXT J
40 PRINT "USING A FOR-NEXT TIMER LOOP"
45 PRINT:PRINT:PRINT:PRINT
50 FOR J=1 TO N: NEXT J
60 PRINT "IF YOU WISH TO CHANGE THE DISPLAY'S SPEED"
65 PRINT:PRINT
70 FOR J=1 TO N: NEXT J
80 PRINT "CHANGE THE VALUES OF N IN THE FOR-NEXT LOOP"
85 PRINT:PRINT:PRINT:PRINT
90 FOR J=1 TO N: NEXT J
100 PRINT "IF YOU WISH TO STOP THE DISPLAY"
105 PRINT:PRINT
110 FOR J=1 TO N: NEXT J
120 PRINT "PRESS THE RUN/STOP KEY"
130 FOR J=1 TO N: NEXT J
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 by 5's from 5Ø to 5 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 the 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.)
4. Type NEW and enter.
5. Using IF-THEN, write a program to generate all even numbers between 11 and 51 from the 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 enter data into the Commodore 64.
3. To write, enter, and RUN programs using READ-DATA and RESTORE.

11.1

READ-DATA Statements

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

11.2

Ways of Entering Data into the Computer

Built-In	From Keyboard	READ-DATA Combination
10 LET A = 5	10 INPUT A	10 DATA 5 20 READ A
• Builds value into the program	• Allows you to enter data through the keyboard	• DATA statement contains the value • READ statement assigns the value to the variable named

Note:

◯ Data lines can be read only by READ statements.

◯ READ-DATA work together to enter data into the computer.

11.3
READ-DATA Example

```
                5 REM READ—DATA EXAMPLE
DATA Statement  10 DATA  (1) , (2) , (3) , (4) , (5)
                          ↓     ↓     ↓     ↓     ↓
READ Statement  20 READ   A   ,  B  ,  C  ,  D  ,  E
PRINT Statement 30 PRINT  A   ,  B  ,  C  ,  D  ,  E
```

Note:

- ◯ Each variable in a READ statement must have a corresponding value in a DATA statement.
- ◯ Each READ statement can read a number of pieces of data if each variable is separated by a comma.
- ◯ Data lines can be used only by READ statements.

EXERCISE 11-1
READ-DATA

YOUR ACTION	DISPLAY
1. Type and enter.	10 DATA 1,2,3,4,5 20 READ A,B,C,D,E 30 PRINT A,B,C,D,E
2. Type RUN and press RETURN.	1 2 3 4 5

Note:

- ○ 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.

SUMMARY

DATA Statement

- 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 values.
  - — 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."
20 DATA LEEDS MIDDLE SCHOOL,COMPUTERS
30 DATA 125, 250, 750, 1000
```

Note:

○ Quotes are used in Line 10 because data contain commas.

SUMMARY

READ Statement

- 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 occurs if there are more attempts to READ than there are data items.
- Example:

```
10 DATA "JONES, A.B.","SMITH, R.J."
20 DATA LEEDS MIDDLE SCHOOL,COMPUTERS
30 DATA 125,250,750,1000
40 READ A$,B$,C$,D$,A,B,C,D
```

Note:

○ There are eight variables in the READ statement and eight items in the DATA statements.

ASSIGNMENT 11-1

1. Type and enter the following program:

```
10 PRINT "NAME",,"GRADE"
20 READ A$
30 IF A$ = "END" THEN PRINT "END OF LIST": END
40 READ G
50 IF G < 75 THEN PRINT 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?

11.4
RESTORE

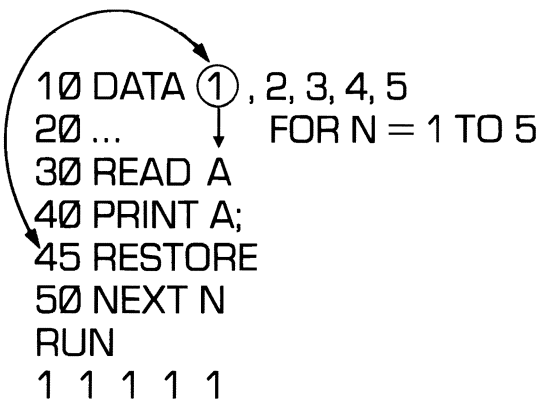
- Key word that causes the next READ statement executed to start over with the first DATA statement.
  - — This lets your program reuse the same data lines.
  - — Sometimes it is necessary to READ the same data more than once without having to RUN the complete program again; therefore, RESTORE is used.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

11.5

Illustration of the RESTORE Statement



Note:

- RESTORE caused data Line 10 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

RESTORE and READ-DATA Statements 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 45 (type and enter).	45 RESTORE Ⓐ
4. LIST the program.	10 DATA 1,2,3,4,5 20 FOR N = 1 TO 5 30 READ A 40 PRINT A ; 45 RESTORE 50 NEXT N
5. Type RUN and press RETURN.	1 1 1 1 1 Ⓑ

Note:

Ⓐ This restores data line to its original unread condition.

Ⓑ Computer reads first data item over and over.

SUMMARY
READ-DATA, RESTORE

- READ-DATA
  — These key words are used to enter lots of data into the computer.
  — READ-DATA statements are common in programs.
- READ
  — Each variable in a READ statement must have a corresponding value in a DATA statement or an OUT OF DATA ERROR will occur.
- DATA
  — DATA statements can be placed anywhere in a program.
  — DATA statements can be used only by READ statements.
  — If more than one piece of data is placed in a DATA statement, they must be separated by commas.
  — DATA statements are read from left to right by READ statements.
- RESTORE
  — This key word is used to return DATA statements to their original, unread state so they can be used again.

PRACTICE 20
READ-DATA

1. Type and enter the following program:

```
5 PRINT "♥"
10 PRINT "NAME",, "GRADE"
20 READ A$
30 IF A$ = "END" THEN PRINT "END OF LIST":END
40 READ G
50 IF G > 75 THEN PRINT 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

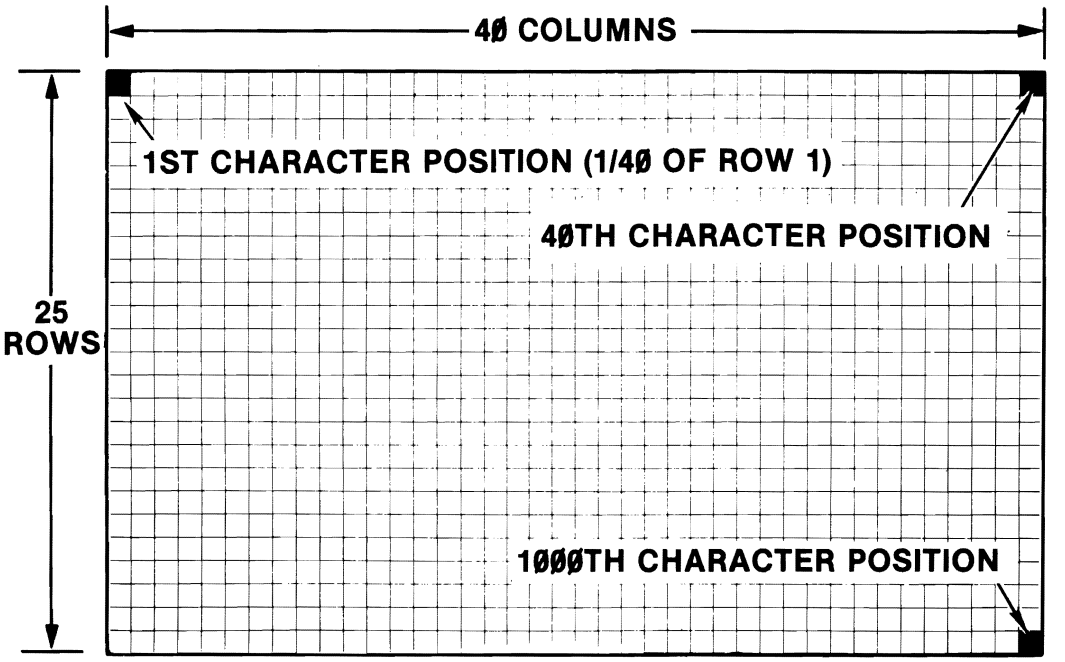
What You Will Learn

1. To become familiar with the layout of the graphic display of the Commodore 64 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 Commodore 64 graphics might appear to be very complicated initially, but if you follow the examples and do the exercises in this section, you will have the necessary fundamentals to do graphics on your computer. *Practice* is the key to success with graphics — experiment with the keys to see what graphic symbols appear on the screen. Try to sketch some simple figures with your Commodore 64.

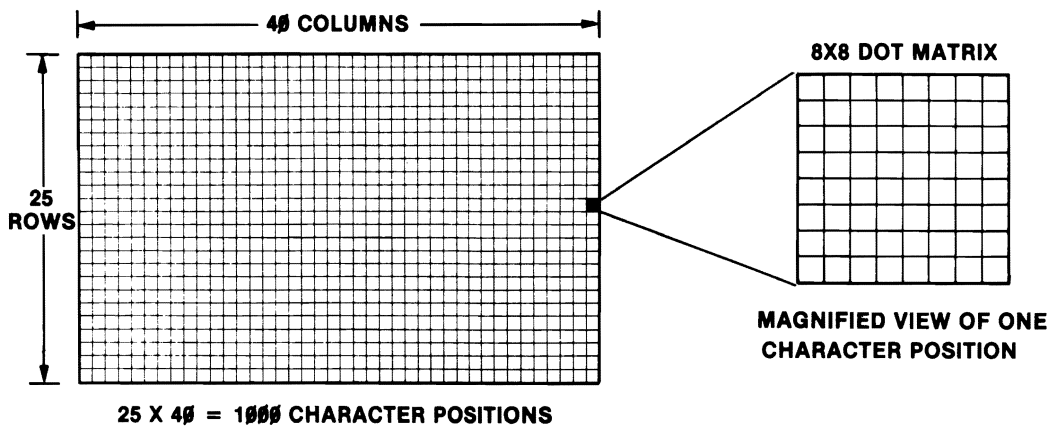
12.1
Video Display Layout



- The display has 1000 character positions arranged as 25 rows, with 40 characters per row (25 × 40 = 1000 characters).
  - — One-fortieth 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 page).
  - — Although one line on the display is only 40 characters long, up to 80 characters can be handled logically. The Commodore 64 uses the "wraparound" feature or uses two rows to display one line. (This, of course, assumes you did not press the RETURN key at the end of the line. But if you attempt to type an 81st character, the Commodore 64 will automatically go to the next line.)

12.2

Illustration of Dot Matrix for One Character Position



Note:

○ 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.

12.3
Commodore 64 Keyboard

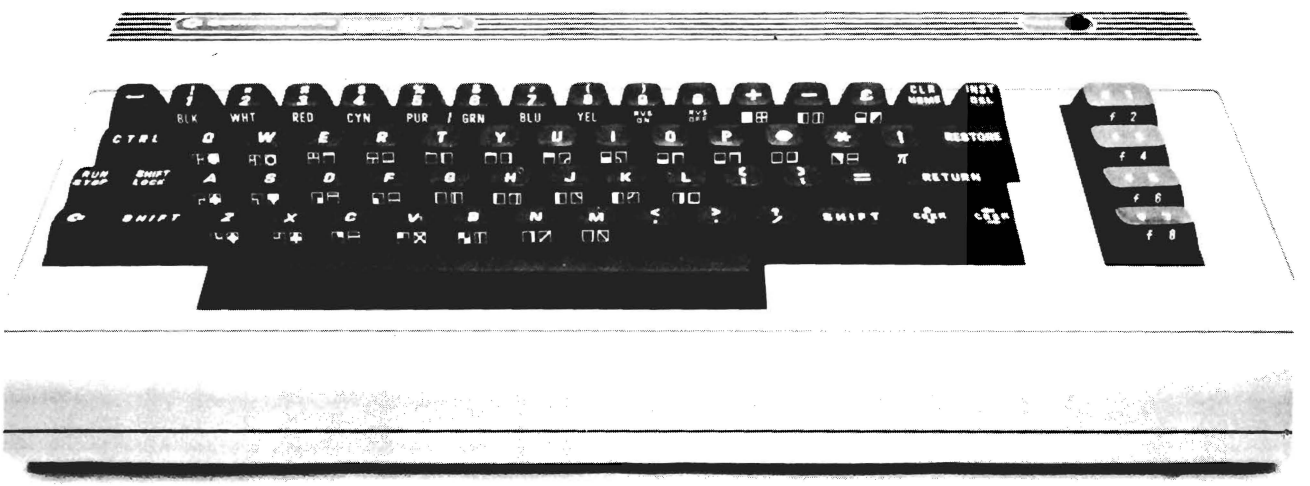


Photo Courtesy Lou Odor

12.4
Graphic Keys

- On page 160 is a picture of the Commodore 64 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 graphic key.
- To display the graphic characters on the right side of the keys, you must hold down the SHIFT key while pressing the desired graphic key.
- Here are some examples:

LEFT-SIDE GRAPHICS			RIGHT-SIDE GRAPHICS		
HOLD DOWN	THEN PRESS	DISPLAY	HOLD DOWN	THEN PRESS	DISPLAY
C=	E		SHIFT	E	
C=	R		SHIFT	R	
C=	W		SHIFT	W	
C=	Q		SHIFT	Q	

- 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 Commodore 64.
  - — 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.

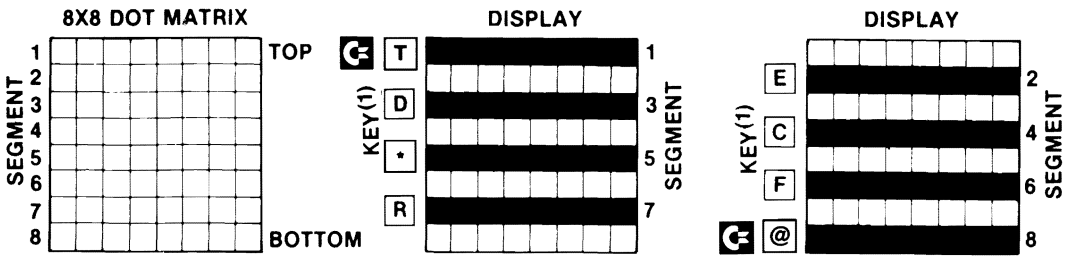
12.5
Graphic Characters Reference Charts

- 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 Commodore 64 display. Also note that:
  - — Symbols are grouped by similarity of line segments; the charts are labeled A through G. (Refer to pages 163-169.)
  - — 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 the shifted [D] key the Commodore 64 will display a horizontal line in one character position. Then, if you press the shifted [E] key the Commodore 64 will display the 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 8x8 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 graphic 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  mode.

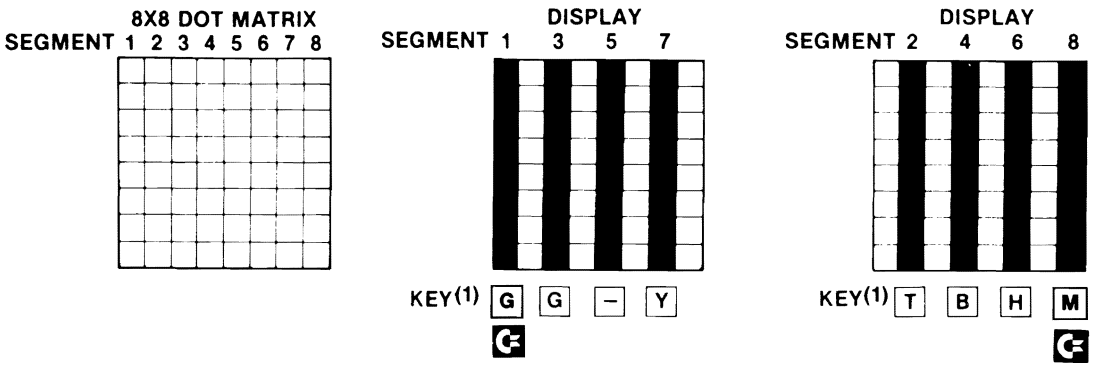
12.6

Graphic Characters Reference Chart A

Horizontal Line Segments



Vertical Line Segments



(1)Use the SHIFT key with the desired key except where the key is shown.

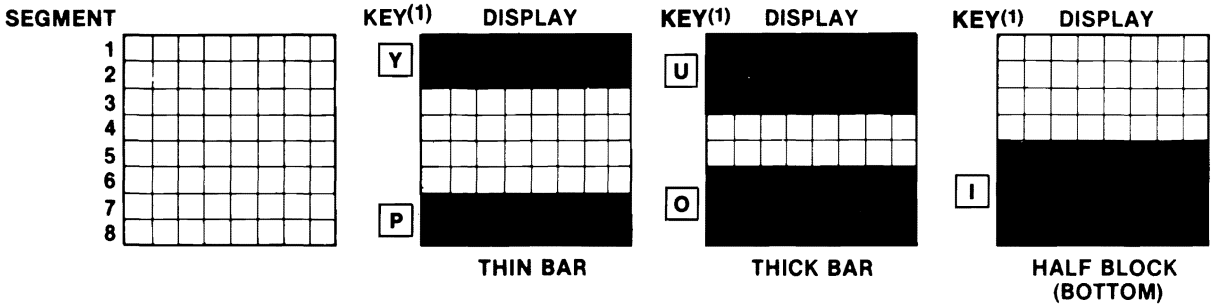
Note:

- 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 the T key together with the key the computer will display a horizontal line in one character position; then if you press E together with the SHIFT key the line will be in the adjacent character position.

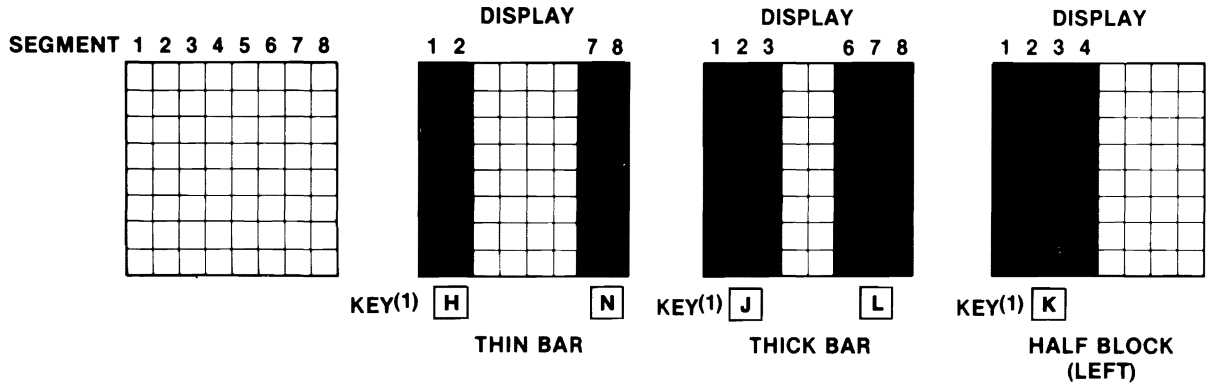
12.7

Graphic Characters Reference Chart B

Horizontal Bars



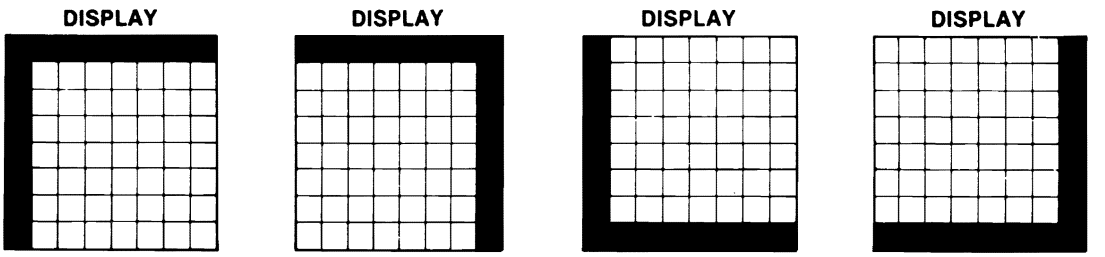
Vertical Bars



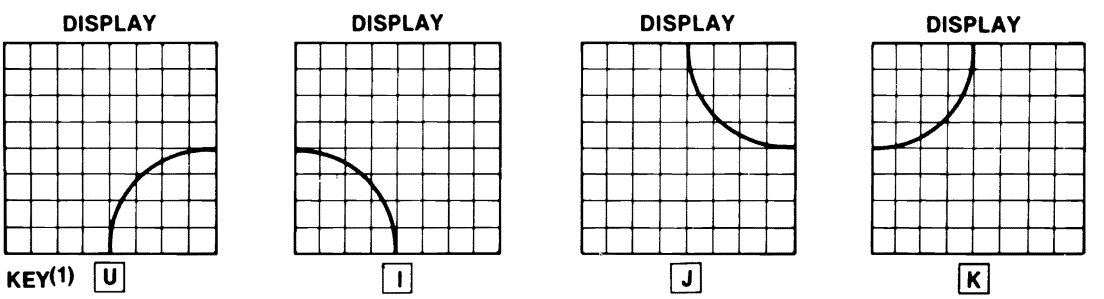
(1)Left-side graphics (hold down the key while pressing the key shown to get the desired graphic character).

12.8
Graphic Characters Reference Chart C

Square Corners



Rounded Corners

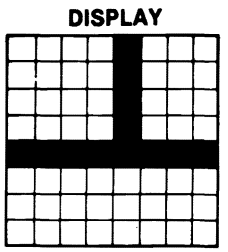


(1)Right-side graphics (hold down the SHIFT key while pressing the key shown).

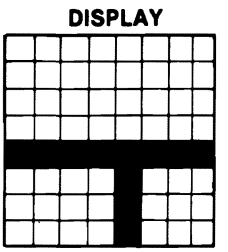
12.9

Graphic Characters Reference Chart D

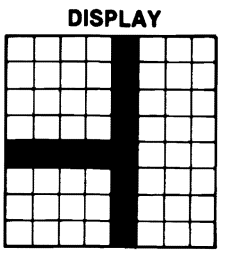
T Symbols



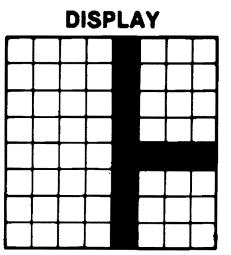
KEY(1) E



R

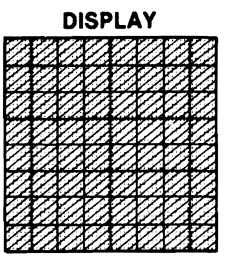


W

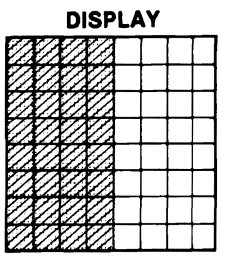


Q

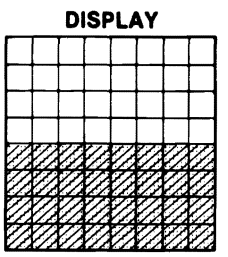
Grids



KEY(1) +



−

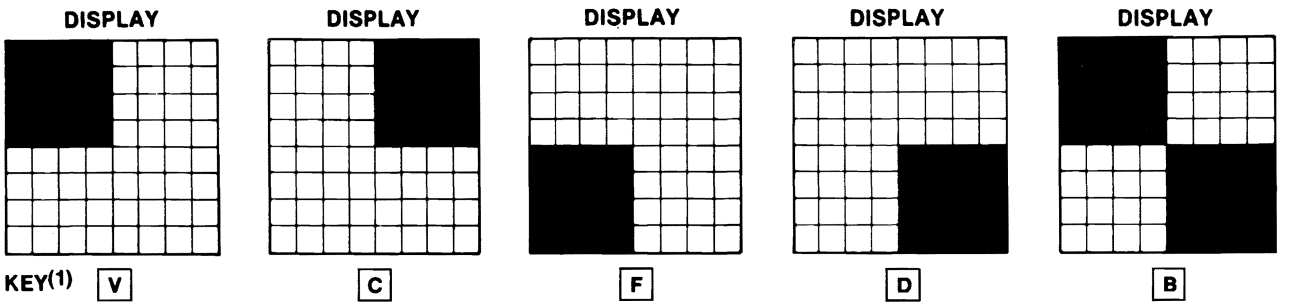


£

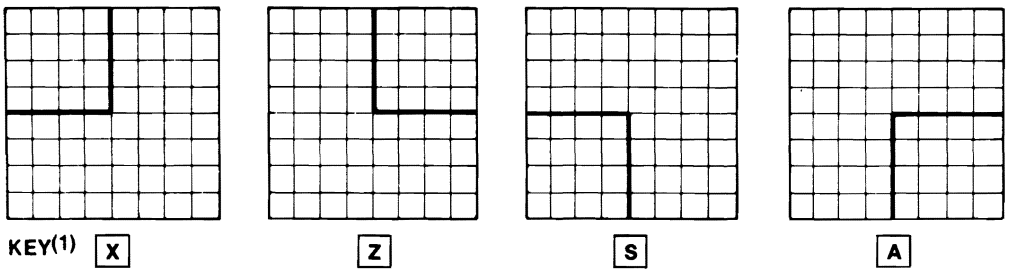
(1)Left-side graphics (use the C= key).

12.10
Graphic Characters Reference Chart E

Quarter Blocks (Solid)



Quarter Blocks (Open)



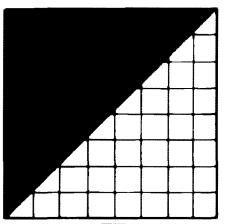
(1)Left-side graphics (use the G key).

12.11

Graphic Characters Reference Chart F

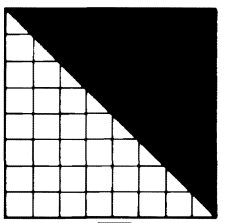
Triangles and Diagonals

DISPLAY



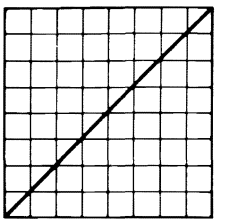
KEY(1) £

DISPLAY



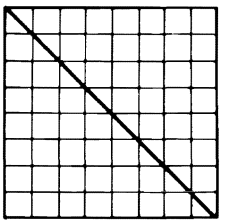
KEY(2) *

DISPLAY



KEY(1) N

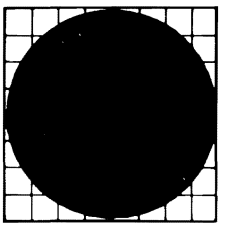
DISPLAY



KEY(1) M

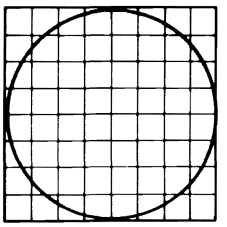
Circles, X, and Cross

DISPLAY



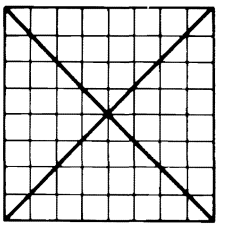
KEY(1) Q

DISPLAY



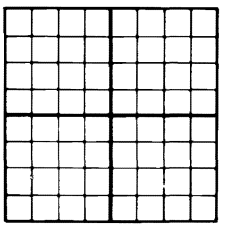
KEY(1) W

DISPLAY



KEY(1) V

DISPLAY



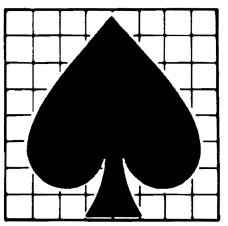
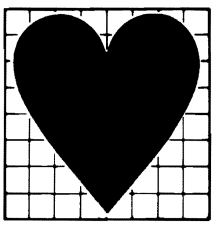
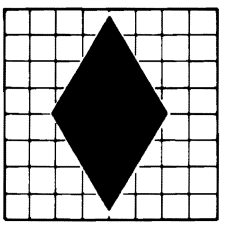
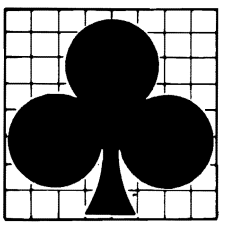
KEY(1) +

(1)Right-side graphics (use the SHIFT key).

(2)Left-side graphics (use the key).

12.12
Graphic Characters Reference Chart G

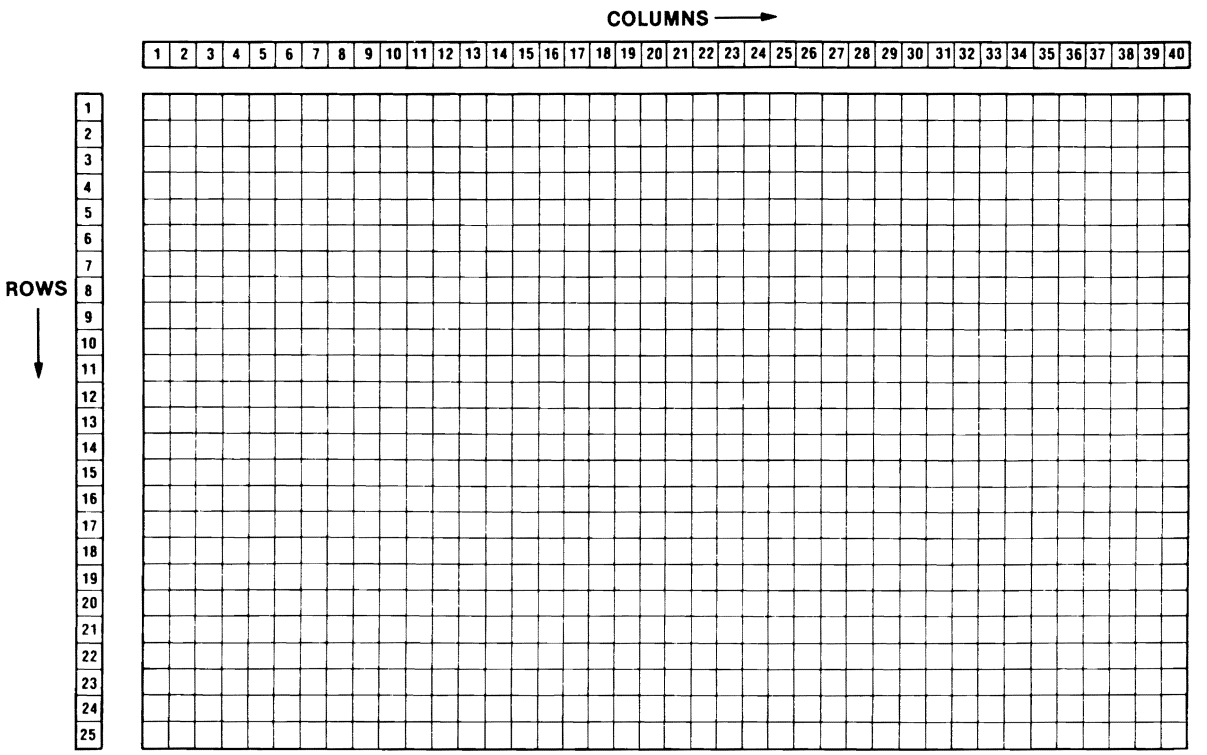
Card Suit

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

(1)Right-side graphics (use the SHIFT key).

12.13

Video Display Worksheet for Graphics



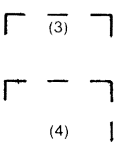
12.14

Tips on Using Graphics

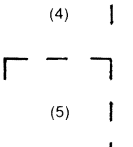
Drawing pictures on the Commodore 64 is somewhat like putting together a puzzle that has 62 possible pieces. Each piece of the puzzle is represented by one of the 62 graphic characters. The major difference between a puzzle and a picture on the Commodore 64, however, is that you will not use all 62 pieces 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 3x3 square you would need the following (refer to pages 163 and 165). Draw segments on paper. Do not use the computer yet.)

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

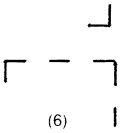
2. A straight piece or horizontal line segment that matches the line segment of the corner (⌜¯). You have to be careful here because there are eight possibilities (that is, you could select one of these keys: [T], [D], [*], [R], [E], [C], [F], or [@]). But a close examination of the pieces (keys) will 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):



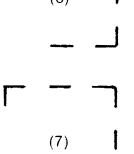
3. A top-right corner (¬), which is a right-side [P] key. Your picture would look like this now (without the spaces):



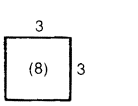
4. A vertical line segment (|) that matches the top-right corner. Again you have eight possibilities (keys [G], [G], [-], [Y], [T], [B], [H], or [M]). But the only piece or vertical line segment that matches the line segment of the top-right corner is the left-side [M] key. We now have this (without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



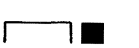
5. A bottom-right corner (¯|), which is a shifted [@] key. Our square is really beginning to take shape now and looks like this (without the spaces):



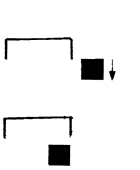
6. A horizontal segment (¯) that matches the bottom-right corner. Again, there are eight possibilities but only the left-side [@] key matches the bottom-right corner. The figure now looks like this:



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):



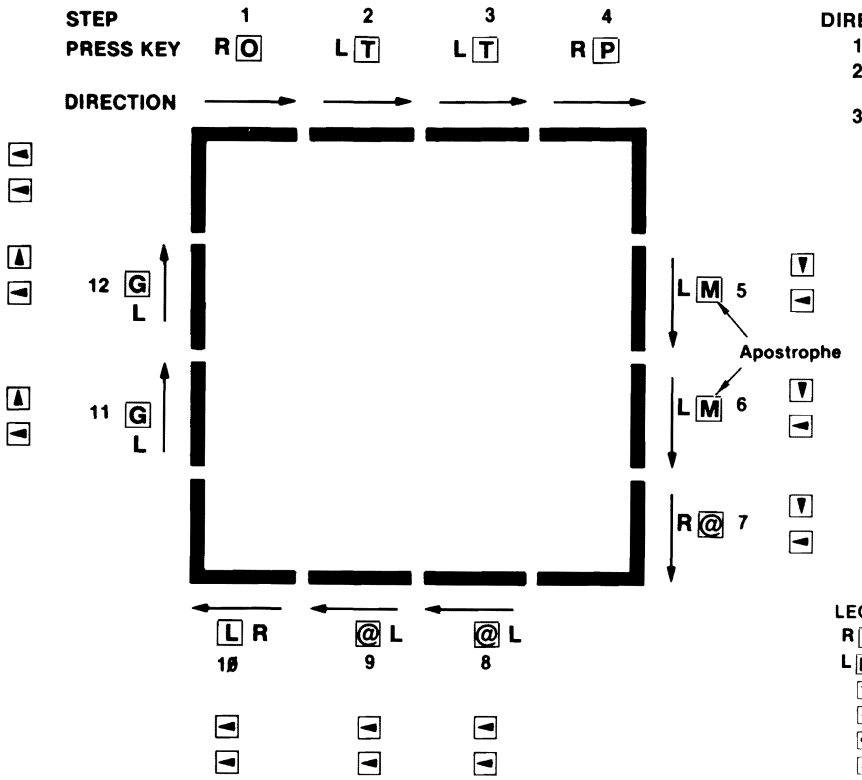
This procedure describes the basic approach to graphics. There are other things to consider if you are going to draw this picture on the Commodore 64, however. For example, what about spacing? You would have to move the picture to the desired part of the screen. Otherwise, all pictures will start in 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 the [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 40 blocks wide and 25 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 163-169), which provide magnified versions of each graphic symbol together with its respective key.

EXERCISE 12-1

Drawing a Square (In 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

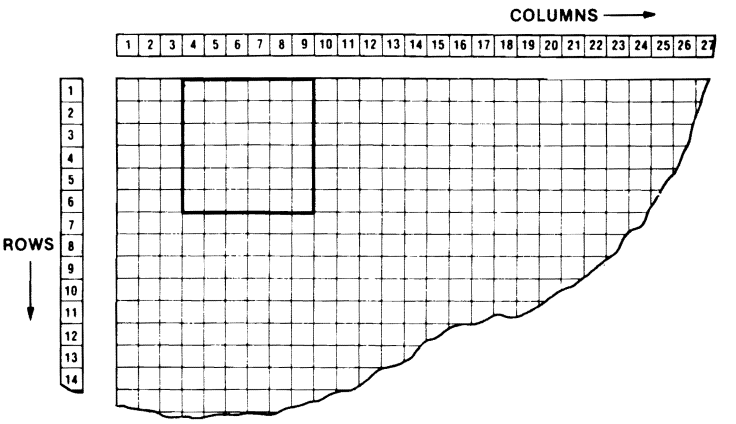
EXERCISE 12-2

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 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:

- ◯ Left-side graphics: T , M , @ , * , G (use with C key).
- ◯ Right-side graphics: O , P , @ , * , L (use with SHIFT key).
- ◯ Notice that both the left and right sides of the @ key are used.

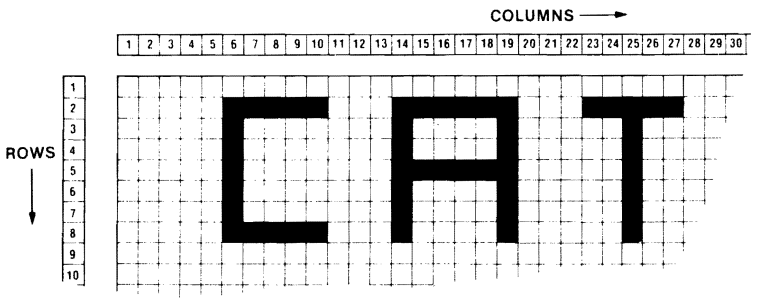
EXERCISE 12-3

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 12-4

Drawing Large-Size Letters with Commodore 64



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
```

12.15
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:

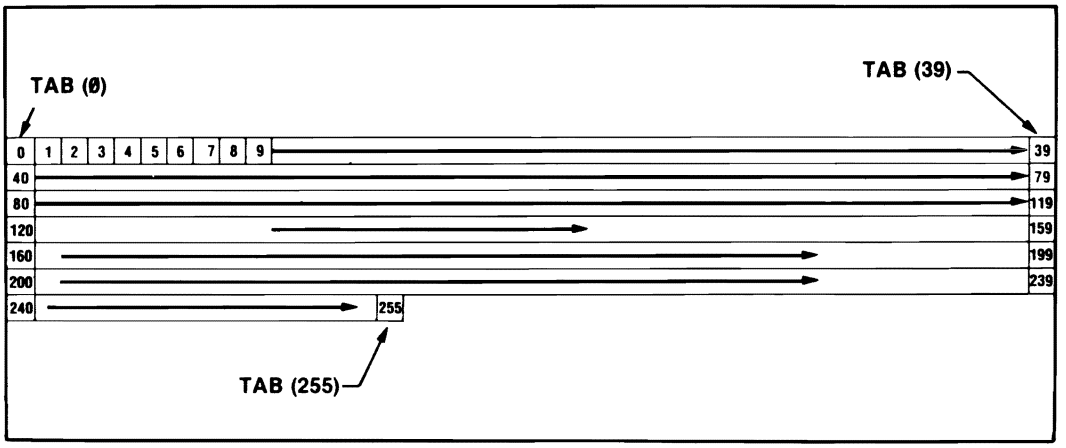
```
10 PRINT TAB (10) "TABBED 10"
```

- SPC
  — Moves the 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:

```
10 PRINT SPC (10) "SPACED 10"
```

12.16

Illustration Showing Location of TAB Printing Positions (Ø to 255)



Note:

- TAB cannot move to the left. All moves are from the left margin.
- If TAB moves the cursor beyond the rightmost limit (e.g., 39) of the text line, the cursor is moved to the leftmost limit of the next lower line (i.e. 4Ø).

12.17

TAB Example

YOUR ACTION	DISPLAY
1. Type and enter the program shown.	5 PRINT "♥" 10 PRINT TAB (Ø) "TABBED Ø" 20 PRINT TAB (1) "TABBED 1" 30 PRINT TAB (5) "TABBED 5" 40 PRINT TAB (10) "TABBED 10" 50 PRINT TAB (20) "TABBED 20" 60 PRINT TAB (30) "TABBED 30" 70 PRINT TAB (40) "TABBED 40" 80 PRINT TAB (255) "TABBED 255"
2. RUN the program.	TABBED Ø TABBED 1 TABBED 5 TABBED 10 TABBED 20 TABBED 30 TABBED 40 TABBED 255 READY

Note:

- ◯ TAB moves the cursor right to a specified column position.
- ◯ TAB (Ø) is the first column on the left and TAB (39) is the last column on the right. (See previous page.)
- ◯ TAB (40) starts a new line but is in the same column as TAB (Ø).
- ◯ TAB (255) is the last TAB position.

12.18

SPC Example

YOUR ACTION	DISPLAY
1. Type and enter the program shown.	5 PRINT "▼" 10 PRINT SPC (0) "SPACED 0" 20 PRINT SPC (1) "SPACED 1" 30 PRINT SPC (5) "SPACED 5" 40 PRINT SPC (10) "SPACED 10" 50 PRINT SPC (20) "SPACED 20" 60 PRINT SPC (30) "SPACED 30" 70 PRINT SPC (40) "SPACED 40" 80 PRINT SPC (255) "SPACED 255"
2. RUN the program.	SPACED 0 SPACED 1 SPACED 5 SPACED 10 SPACED 20 SPACED 30 SPACED 40 SPACED 255 READY

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.

12.19
Difference between TAB (N) and SPC (N)

Example:

```
10 PRINT "ADRIENNE"; TAB (10); "JONES"
20 PRINT "ADRIENNE"; SPC (10); "JONES"
RUN
ADRIENNE     JONES
■ ——————→ ■ (cursor)
```

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

```
ADRIENNE                JONES
          ■ ——————→ ■ (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 (N) moves cursor to a specified position (N) spaces to the right of the current cursor position.

EXERCISE 12-5A

Difference between SPC and TAB

YOUR ACTION	DISPLAY
1. Type and enter this program.	5 PRINT "♥" 10 PRINT "# STUDENTS"; "GRADE" 15 PRINT 20 PRINT TAB (5) "# STUDENTS"; "GRADE" 25 PRINT 30 PRINT TAB (5) "# STUDENTS" SPC (5) "GRADE" Ⓐ
2. RUN the program and explain what happened.	# STUDENTSGRADE # STUDENTSGRADE # STUDENTS GRADE
3. Type NEW and enter this program.	5 PRINT "♥" 100 PRINT "ADRIENNE" TAB (10) "JONES" 110 PRINT "ADRIENNE" SPC (10) "JONES"
4. RUN the program. 5. Explain what happened.	ADRIENNE JONES ADRIENNE JONES READY

Note:

Ⓐ Line 20 (TAB) causes printing to start 5 spaces to the right. Line 30 does the same as Line 20, and SPC (5) inserts 5 spaces between "STUDENTS" and "GRADE."

EXERCISE 12-5B

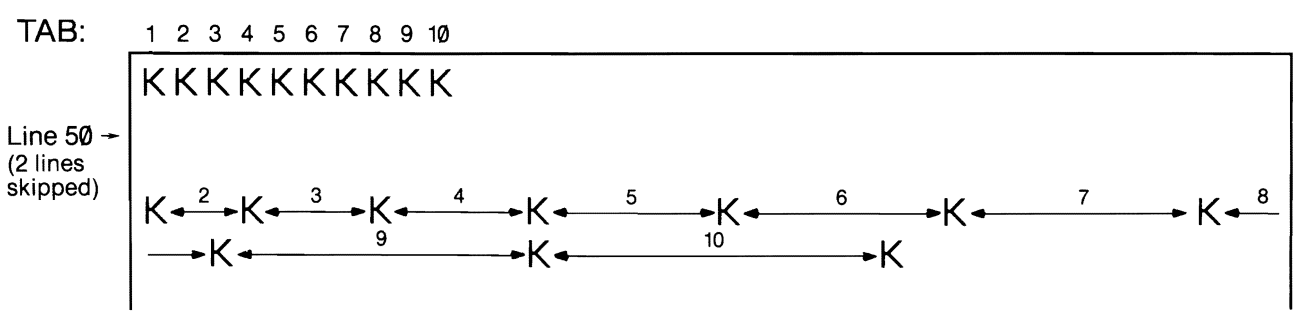
Difference between SPC and TAB (Cont.)

YOUR ACTION	DISPLAY
6. Type NEW and enter this program.	10 PRINT "▼" 20 FOR J = 1 TO 10 30 PRINT TAB(J)"K"; 40 NEXT J 50 PRINT:PRINT 60 FOR J = 1 TO 10 70 PRINT SPC(J)"K"; 80 NEXT J
7. RUN the program	KKKKKKKKKK K K K K K K K K K K

Note:

- ◯ Line 10 clears the screen.
- ◯ Semicolon at end of Line 30 allows the next character printed to be tabbed on the same line.
- ◯ Line 50 skips two lines.
- ◯ Semicolon at end of Line 70 allows the next character printed to be spaced from the last character printed, on the same line.

ANALYSIS



SUMMARY

TAB and SPC Functions

- 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 8x8 square three blocks or squares from the left 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.
2. 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.

13.1
An Array

What is an array?

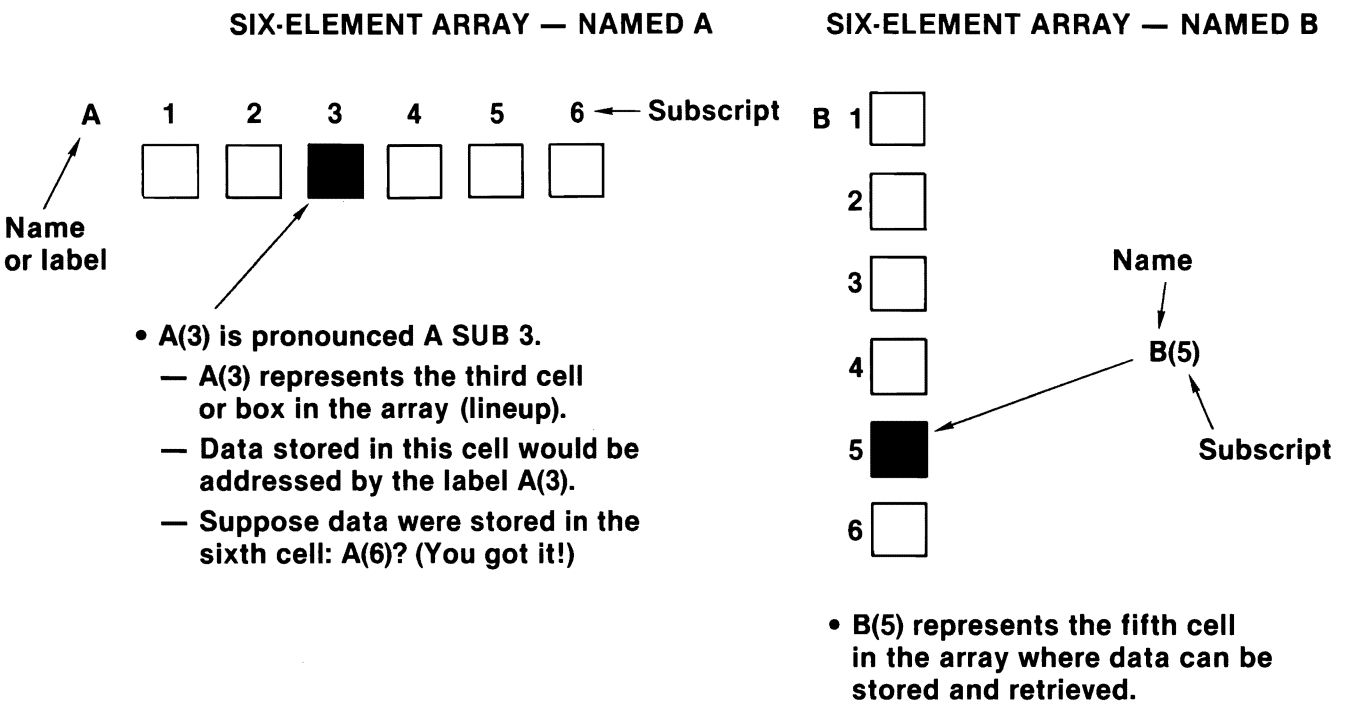
- An array is a lineup, an arrangement, or an orderly grouping of things.

Why use an array?

- We use it when we wish to have more variables available in a program.
  - — Although the Commodore 64 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.

13.2

One-Dimensional Array Illustration



Note:

○ A and B are optional names. Any valid variable name can be used to name an array in Commodore 64 BASIC.

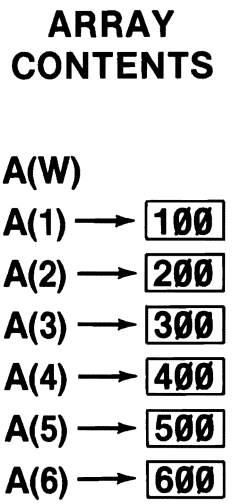
13.3
One-Dimensional Array Program

YOUR ACTION	DISPLAY
1. Type in these program lines.	10 DATA 100,200,300,400,500,600 20 FOR W=1 TO 6 30 READ A(W) 40 NEXT W 50 FOR W=1 TO 6 60 PRINT W,A(W) 70 NEXT W
2. RUN the program.	1 100 2 200 3 300 4 400 5 500 6 600

Note:

- ◯ Lines 20-40 store the data in array A(W).
- ◯ Lines 50-70 retrieve the data from array A(W) and PRINT the data.

13.4
One-Dimensional Array Program Analysis



- 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.

13.5

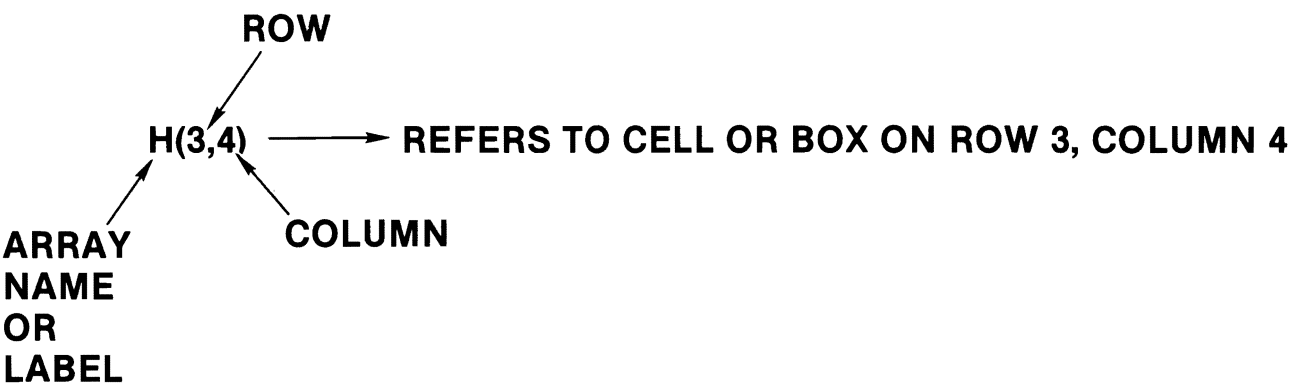
Two-Dimensional Array Illustration

COLUMN

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

ROW

36 ELEMENT ARRAY (MATRIX)
(NAMED H)



EXERCISE 13-1

Fill in the blanks using the matrix on page 190.

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)	_____	_____	_____

13.6
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-dimensional array A with 6 elements A(Ø)—A(5) or A(1)—A(6)*

Sets a two-dimensional array B with 3 rows (numbered Ø-2) and 4 columns (numbered Ø-3)

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

*If A(Ø) is not used.

13.7

Checkbook Array Example

- Consider the following table of checkbook information:

Check #	Date Written	Amount
100	6/5/83	$ 15.50
101	6/7/83	25.00
102	6/15/83	145.00
103	6/22/83	65.00
104	6/30/83	211.00
105	6/30/83	79.50

- 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 $145.00.
- The above table can be set up in a 6x3 array or matrix.

CK	1	2	3
1	100	60583	15.50
2	101	60783	25.00
3	102	61583	145.00
4	103	62283	65.00
5	104	63083	211.00
6	105	63083	79.50

Note:

- ◯ The date is recorded in the form mmddyy, where mm = month number, dd = day, and yy = the last two digits of year.
- ◯ Since CK is a numeric array, alphanumeric characters such as dashes cannot be stored.

13.8

Checkbook Array Program — Setting Up the Array

YOUR ACTION	DISPLAY
1. Type in these lines:	5 PRINT "♥" 10 DIM CK(6,3) 20 FOR ROW=1 TO 6 30 FOR COL=1 TO 3 40 READ CK(ROW,COL) 50 NEXT COL,ROW 60 DATA 100,60583,15.50 70 DATA 101,60783,25.00 80 DATA 102,61583,145.00 90 DATA 103,62283,65.00 100 DATA 104,63083,211.00 110 DATA 105,63083,79.50 120 FOR ROW=1 TO 6 130 SUM=SUM+CK(ROW,3) 140 NEXT ROW 150 PRINT "TOTAL OF CHECKS"; 160 PRINT ":$";SUM
2. Type RUN.	TOTAL OF CHECKS:$ 541

13.9

Checkbook Array Program — Manipulating the Array

YOUR ACTION

3. **Do not type in NEW.**

4. Add these steps to your program:

5. Type RUN.
   (Enter 63083 for INPUT.)

DISPLAY

```
200 INPUT "LIST CHECKS WRITTEN ON (MM DD YY)"; DT
210 PRINT: PRINT "CHECKS WRITTEN ON"; DT; "ARE LISTED BELOW:"
215 PRINT
220 PRINT "CHECK #", "AMOUNT": PRINT
230 FOR ROW = 1 TO 6
240 IF CK (ROW,2) = DT THEN PRINT CK (ROW,1), CK (ROW,3)
250 NEXT ROW
```

```
TOTAL OF CHECKS:$ 541
LIST CHECKS WRITTEN ON (MM DD YY)?
CHECKS WRITTEN ON 63083 ARE LISTED BELOW:

CHECK #          AMOUNT
  104              211
  105              79.5
```

ASSIGNMENT 13-1

Read pages 95-98 in *Commodore 64 User's Guides.**

**Commodore 64 User's Guide,* Commodore Business Machines, Inc., Wayne, Pennsylvania, 1982.

SUMMARY

Arrays

- A2 is not the same as A(2).
  — A2 is an ordinary variable.
  — A(2) is a subscripted variable.
- Any time you have a subscript larger than 1Ø (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 the 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 the program and RUN it several times.

PART

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

14

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 Number.
3. To write, RUN, and analyze programs using the INT(X), ABS(X), and RND(X) functions.

14.1
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, by making a negative number more negative).

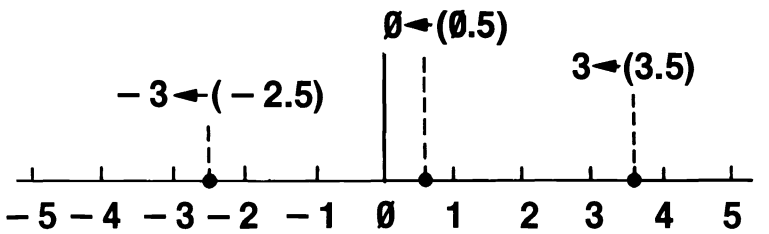
  Example:

  INT (−Ø.6) = −1 INT (−3.14) = −4
  INT (−Ø.2) = −1 INT (−7.28) = −8

EXERCISE 14-1

INT(X)

Graphic 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	_____

14.2

INT(X) Function — Rounding $$

YOUR ACTION	DISPLAY
1. Type and enter this program.	10 A =20/3 20 PRINT"$";A
2. Now RUN.	$6.66666667 READY
3. Add Line 15 to program as shown.	15 A =INT(100*A+.5)/100
4. Now RUN the program. (I told you so!)	$6.67 READY

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.

ASSIGNMENT 14-1
INT(X)

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

```
10 REM AREA OF A CIRCLE = 3.14159 * R ↑ 2
20 INPUT "THE RADIUS IS"; 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.)

14.3
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.	1Ø INPUT "TYPE ANY POSITIVE OR NEGATIVE #"; N 2Ø X = ABS (N) 3Ø PRINT "N","X" 4Ø PRINT N,X
2. RUN the program several times using both positive and negative numbers.	

Note:

○ Regardless of the number you INPUT as N, the absolute value of X is the same number without the sign.

14.4
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 PRINT some random number
                                between 1 and 6 inclusive.
20 PRINT INT(4*RND(1)+1)      — will PRINT some random number
                                between 1 and 4 inclusive.
30 PRINT INT(10*RND(1)+1)     — will PRINT some random number
                                between 1 and 10 inclusive.
```

- Type in the program and RUN it several times. Your output for each run will be three numbers — the first, some number between 1 and 6; the second, some number between 1 and 4; and the third, some number between 1 and 10.

14.5

RND(X) Function — Program Example

YOUR ACTION	DISPLAY
1. Type and enter.	5 INPUT "ENTER A NO. BETWEEN 1 AND 100"; X Ⓐ 10 FOR J = 1 TO 10 20 PRINT INT (X * RND (1)+1), 30 NEXT J
2. RUN.	(Screen should have ten random numbers between 1 and X.)
3. RUN the program again to get the idea.	
4. Change Line 10 to read:	10 FOR J = 1 TO 50
5. RUN the program. (Get the idea?)	(Screen should have fifty random numbers between 1 and X.)

Note:

Ⓐ Line 5 allows you to enter "X" or the highest random number you want.

ASSIGNMENT 14-3

Coin Toss Program

1. Type in the following program:

```
5 PRINT "♥"
6 REM COIN TOSS PROGRAM
10 REM T=TAILS, H=HEADS
15 T=0
20 H=0
30 INPUT "HOW MANY TIMES SHALL I FLIP THE COIN ";N
40 PRINT "♥"
50 PRINT "I'M FLIPPING THE COIN...STAND BY"
60 FOR K=1 TO N
70 X=INT(2*RND(1)+1)
80 IF X=1 THEN 100
90 H=H+1
95 GOTO 110
100 T=T+1
110 NEXT K
120 PRINT "♥"
130 PRINT "HEADS","TAILS"
140 PRINT
145 PRINT H,T
150 PRINT 100*H/N;"%",100*T/N;"%"
155 PRINT
160 PRINT "TOTAL FLIPS =";N.
```

2. RUN this program several times and discuss the results.

Note:

○ Lines 15 and 20 initialize the counters (H and T) to zero.
Line 60 begins the FOR-NEXT statement and executes "N" times.
Line 70 generates a random number (either a 1 or a 2).
In Line 90, "heads" are counted.
In Line 100, "tails" are counted.
Line 130 prints the headings.
Line 140 prints the values of H and T.
Line 150 calculates and prints the percentage of heads, percentage of tails.
Line 155 provides a space for better appearance.

ASSIGNMENT 14-4

Guess the Number Program

1. Type in the following program:

```
10 REM GUESS THE NUMBER GAME
20 PRINT "♥"
30 X=INT(10*RND(1)+1)
40 INPUT "GUESS A NUMBER BETWEEN 1 & 10:";N
50 IF X=N THEN 110
60 IF X < N THEN 90
70 PRINT "HIGHER"
80 GOTO 40
90 PRINT "LOWER"
100 GOTO 40
110 PRINT "THAT'S RIGHT!"
120 FOR J=1 TO 2000
130 NEXT J
140 GOTO 10
```

2. RUN the program. (To stop the program, use the RUN/STOP and RESTORE keys.)
3. Analyze the program.

   Line 20 _____ the display.

   Line 30 is the _____ generator.

   Line 40 allows the user to _____ a number.

   Lines 50 and 60 are _____ statements that compare
   (conditional, unconditional)

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

   Lines 70 and 90 are PRINT statements that guide the player.
   Why does Line 140 GOTO Line 10, and why do Lines 80 and 100 GOTO Line 40? Explain the function of Lines 120 and 130.
4. Modify (change) the program to pick a number between 1 and 100, and RUN this program several times.

5. For more details on generating random numbers, refer to pages 48 through 53 in the *Commodore 64 User's Guide.*

SUMMARY

INT(X), ABS(X), RND(X)

- INT(X) — Provides the 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 it down to the next lowest whole number (e.g., INT (−Ø.6) = −1).
- ABS(X) — Provides the absolute value of X regardless of the number you INPUT (i.e., X is that same number without the sign).
- 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)

1. Fill in the blanks 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 CIRCLE = 3.14159 * R ↑ 2
2Ø INPUT "THE RADIUS IS"; R
25 INPUT "THE RADIUS IS IN (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

RND(X)

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 the program and RUN it 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 the key words **GOSUB, RETURN, ON-GOTO, ON-GOSUB.**
3. To develop, enter, and RUN programs using subroutines.

15.1
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.

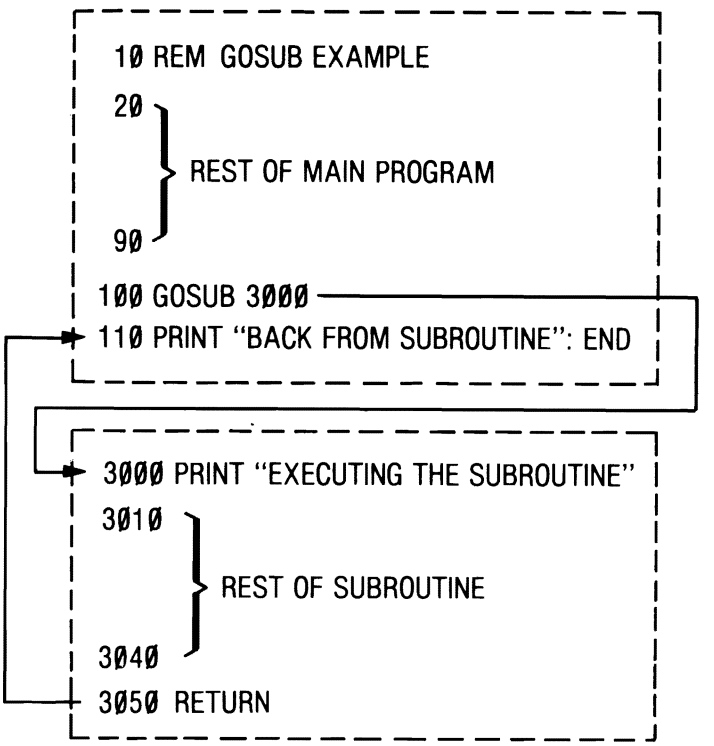
How Do You Call a Subroutine?

- 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 it is finished, the control of the program is returned to the statement in the main program immediately following the most recently executed GOSUB.

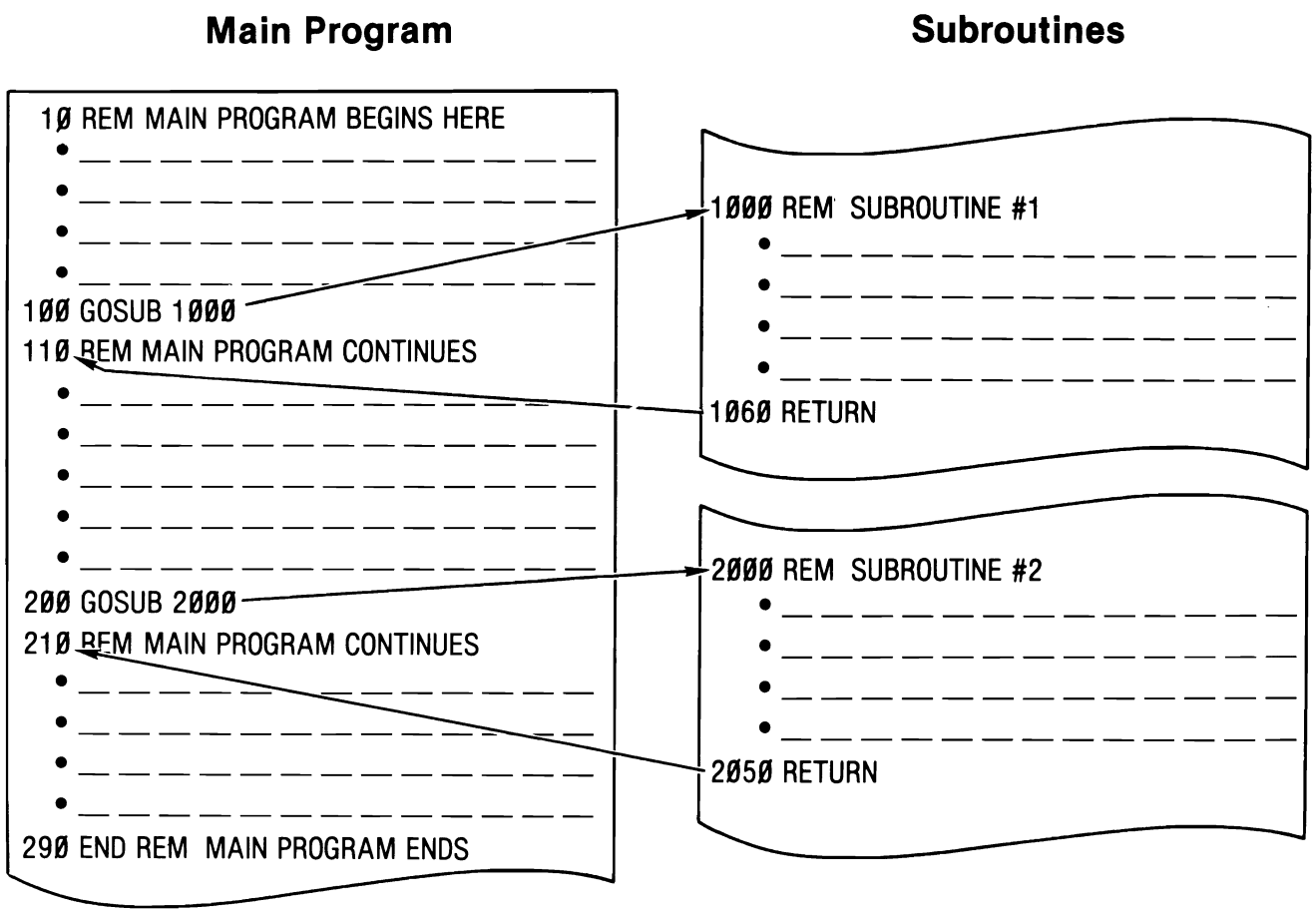
15.2
Subroutine Example

Main Program:

Subroutine:



15.3
Subroutine Illustration



15.4
Analysis of Subroutine Illustration

1. When the computer reaches the GOSUB in Line 1ØØ, the program will branch to 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 want the subroutines to be executed only when we call for them by a GOSUB.

15.5

Sample Program Using Subroutines (Temperature Conversion)

Main Program

```
10 REM TEMPERATURE CONVERSION PROGRAM
15 PRINT "♥"
20 INPUT "DO YOU WISH TO CONVERT C TO F (Y OR N)";A$
30 IF A$ = "Y" THEN 80
40 PRINT: INPUT "DEGREES FAHRENHEIT";F
50 GOSUB 2000
60 PRINT: INPUT "HAVE YOU FINISHED (Y OR N)";B$
70 IF B$ = "N" THEN 15
75 END
80 PRINT: INPUT "DEGREES CELSIUS";C
90 GOSUB 1000
100 GOTO 60
110 IF C$ = "N" THEN 15
120 END
```

Subroutine #1

```
1000 REM CELSIUS TO FAHRENHEIT CONVERSION
1010 F = (9/5)*C + 32 : PRINT
1020 PRINT C;"DEGREES CELSIUS =";F;"DEGREES FAHRENHEIT"
1030 RETURN
```

Subroutine #2

```
2000 REM FAHRENHEIT TO CELSIUS CONVERSION
2010 C = (F-32) * (5/9):PRINT
2020 PRINT F;"DEGREES FAHRENHEIT =";C;"DEGREES CELSIUS"
2030 RETURN
```

15.6

Analysis of Sample Program Using Subroutines

1. Lines 1Ø through 12Ø comprise the main program.
2. Line 2Ø is an INPUT statement asking the user if he wants to convert from C to F. A yes answer (Y) means the user wants to convert from Celsius to Fahrenheit.
3. Line 3Ø tests the answer to the INPUT and branches either to Line 8Ø or Line 4Ø.
4. Line 4Ø allows the user to INPUT the degrees Fahrenheit.
5. Line 5Ø branches to the subroutine on converting F to C (Line 2ØØØ).
6. Line 6Ø asks the user if he is finished. If the user answers "N" then the program will clear the screen (Line 15) and start the main program over again. If the user answers "Y" then the program will end (Line 75).
7. Line 8Ø is similar to Line 4Ø. It allows the user to INPUT the degrees Celsius.
8. Line 9Ø branches to the subroutine on converting C to F (Line 1ØØØ).
9. Line 1ØØ routes the program to Line 6Ø where a line feed is executed for better output appearance and the user is asked if he is finished.
10. Line 1ØØØ through Line 1Ø3Ø contain the subroutine for converting Celsius to Fahrenheit and PRINTing the answer. Line 1Ø3Ø RETURNs control to Line 1ØØ in the main program.
11. Line 2ØØØ through Line 2Ø3Ø contain the subroutine for converting Fahrenheit to Celsius and PRINTing the answer. Line 2Ø3Ø RETURNs control to Line 6Ø in the main program.

15.7
Subroutine Exercise

```
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 SUBROUTINE";" ";
2010 RETURN
```

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

Analyze the program below and write the message:

```
10 LET B = 10
20 GOSUB 2000
30 B = B + 5
40 GOSUB 2000
50 B = B + 10
60 GOSUB 2000
99 END
2000 REM SUBROUTINE
2010 IF B < 12 THEN 2050
2020 IF B = 25 THEN 2070
2030 PRINT "PRIME"
2040 GOTO 2080
2050 PRINT "♥" : PRINT "LEEDS"
2060 GOTO 2080
2070 PRINT "COMPUTERS"
2080 RETURN
```

Message _____

15.8

Conditional Branching (The Long Way)

YOUR ACTION	DISPLAY

1. Type NEW and enter this program.

```
5 PRINT "▼"
10 INPUT "TYPE A NUMBER FROM 1 TO 3";N
20 IF N = 1 THEN 110
30 IF N = 2 THEN 130
40 IF N = 3 THEN 150
50 PRINT "HEY, I WANT A NUMBER FROM 1 TO 3!"
60 GOTO 10
110 PRINT "N = 1"
120 END
130 PRINT "N = 2"
140 END
150 PRINT "N = 3"
160 END
```

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.

15.9

Conditional Branching (The Short Way) with ON-GOTO

YOUR ACTION	DISPLAY
4. Erase Lines 20, 30, and 40. Simply type in each line number and press RETURN.	
5. Type this line:	20 ON N GOTO 110,130,150
6. LIST your program. (If your program does not look like this, fix it!)	5 PRINT "▼" 10 INPUT"TYPE A NUMBER FROM 1 TO 3";N 20 ON N GOTO 110,130,150 50 PRINT "HEY, I WANT A NUMBER FROM 1 TO 3!" 60 GOTO 10 110 PRINT"N = 1" 120 END 130 PRINT"N = 2" 140 END 150 PRINT"N = 3" 160 END
7. RUN the program a few times. (Works the same as before, doesn't it.)	
8. RUN the program again. Use the following INPUTs: 1.5 1.8 2.8 0.8 3.99	N = 1 N = 1 N = 2 HEY, I WANT A NUMBER FROM 1 TO 3! N = 3

Note:

◯ The ON-GOTO statement has a built-in INT function!

15.10
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 function, which really acts like this:
   2Ø ON INT (N) GOTO...etc.

ASSIGNMENT 15-2

1. Type and enter the following program:

```
5 PRINT "♥"
10 INPUT "ENTER A NUMBER FROM 1 TO 5";N
20 ON N GOTO 100, 200, 300, 400, 500
30 PRINT "HEY, I WANT A NUMBER FROM 1 TO 5!" : 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:
   - What happens (output) if the INPUT is 1.8 (Line 10)? _____
   - What happens (output) if the INPUT is 3.99? _____
   - What happens (output) if the INPUT is 2.89? _____
   - What happens if the INPUT is 0.5? _____
3. RUN the program several times and record the following:

INPUT	OUTPUT

15.11
ON-GOSUB

- Works like ON-GOTO, except that 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, 2000, 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).

ASSIGNMENT 15-3

1. Type in the following program:

```
5 REM A QUIZ PROGRAM
10 PRINT "♥"
20 PRINT "THIS PROGRAM GIVES A QUIZ."
30 PRINT "YOU HAVE A CHOICE:"
40 PRINT "1. ADD","2. SUBTRACT"
50 PRINT "3. MULTIPLY","4. DIVIDE"
60 INPUT "WHAT IS YOUR CHOICE ";N
80 IF N < 1 THEN 30
90 IF N > 4 THEN 30
100 A=INT(10*RND(1)+1)
110 B=INT(10*RND(1)+1)
120 ON N GOSUB 500,600,700,800
130 INPUT "YOUR ANSWER ";S
140 IF S=A1 THEN 180
150 PRINT "NO. THE ANSWER WAS";A1
160 INPUT "ANOTHER PROBLEM ";A$
170 IF A$="YES" THEN 10
175 END
180 PRINT "VERY GOOD!"
190 GOTO 160
500 A1=A+B
510 PRINT A;"+";B;" = "
520 RETURN
600 A1=A-B
610 PRINT A;"-";B;" = "
620 RETURN
700 A1=A*B
710 PRINT A;"*";B;" = "
720 RETURN
800 A1=A/B
810 PRINT A;"/";B;" = "
820 RETURN
```

2. Change the program so that:
   - the problems generate larger numbers.
   - the answer to the subtraction problem is never negative.
   - the answer to the division problem is always an integer.
   - the person always gets 5 problems at a time.
   - the person receives a score of the number of correct answers.

SUMMARY

GOSUB, ON-GOSUB, ON-GOTO

- GOSUB XXXX
  - — Here the computer branches to the subroutine beginning at XXXX (the specified line number).
  - — The subroutine is executed until a RETURN statement is encountered.
  - — Program control returns to the statement that follows the GOSUB statement in the main program.
- ON n GOSUB XXXX,----,YYYY
  - — This is a conditional branching statement that sends control of the program to one of the subroutines specified in the line number list (XXXX,----,YYYY).
  - — The value of the test variable (n) determines which subroutine is executed. For example, if n=1, control is passed to the subroutine in line XXXX.
  - — Once a subroutine is being executed, the rules governing GOSUB are followed, i.e., a RETURN passes control to the statement following the ON n GOSUB.

- ON n GOTO XXXX,----,YYYY
  - — This statement works similar to ON n GOSUB except control does not pass to a subroutine but to another part of the main program.
  - — The value of n determines which line number is executed next.

Note:

○ n can be a constant, a variable, or a variable expression.

○ n must not be negative.

○ n must be a value from Ø to 255 inclusive.

PRACTICE 26

Program to Convert Celsius to Fahrenheit and Vice Versa

1. Write a program that will do the following:
   a. Convert Celsius to Fahrenheit.
   b. Convert Fahrenheit to Celsius.
   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 BETWEEN A AND B IS _____ (your answer).
   THE QUOTIENT OF A AND B IS _____ (your answer).
   THE PRODUCT OF A AND 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 program 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. 312+4−2
3. 36/4*5
4. 28+4*6/8
5. (18−2)/3+4(6*3)+213

EXTRA PRACTICE 4

Print Zones

Part 1

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

	ZONE 1	ZONE 2	ZONE 3	ZONE 4
1.	LEEDS	LEEDS	LEEDS	LEEDS
2.	LEEDS		LEEDS	
3.		LEEDS		LEEDS
4.		LEEDS	LEEDS	LEEDS
5.			LEEDS	LEEDS
6.				LEEDS

Part II

Using page 68, type in the information as shown (LEEDSPRIME)...and so on.

1. Count the number of characters in all four zones. How many?
2. How many in Zone 1 _____; Zone 3 _____?

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 a 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 2Ø to 1 and PRINT the numbers, their squares, their cubes, and fourth powers on the same line.

   Example: 2Ø 4ØØ 8ØØØ 16ØØØØ
   19 361 6859 13Ø321
   18 324 5832 1Ø4976
   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 $9 \times 12 = 1Ø8$.

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	1Ø
Quiz #										
1	88	41	1ØØ	88	79	76	86	9Ø	85	1ØØ
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	?	Computer to calculate and PRINT average
3	?	
4	?	

and so forth

Quiz #	*Course Avg./Quiz*	
1	?	
2	?	Computer to calculate and PRINT average
3	?	

HAYDEN BOOK COMPANY
a division of Hayden Publishing Company, Inc.
Hasbrouck Heights, New Jersey
ISBN 0-8104-6172-2