



NATIONAL OPEN UNIVERISITY OF NIGERIA

SCHOOL OF SCIENCE AND TECHNOLOGY

COURSE CODE: CIT 132

COURSE TITLE: PROGRAMMING IN BASIC

**COURSE
GUIDE**

**CIT 132
PROGRAMMING IN BASIC**

Course Team Course Writer Prof. R. O. Ayeni
 Programme Leader
 Course Coordinator



NATIONAL OPEN UNIVERSITY OF NIGERIA

National Open University of Nigeria
Headquarters
14/16 Ahmadu Bello Way
Victoria Island, Lagos

Abuja Office
5 Dar es Salaam Street
Off Aminu Kano Crescent
Wuse II, Abuja

e-mail: centralinfo@nou.edu.ng

URL: www.nou.edu.ng

Published by
National Open University of Nigeria

Printed 2014

ISBN: 978-058-906-6

All Rights Reserved

CONTENT

PAGES

Introduction	iv
What you will Learn in this Course	iv
Working through this Course	iv
Assessment	iv
Course Overview	iv
How to get most from this Course	v
Summary	vi

INTRODUCTION

CST - Programming in BASIC is an introductory course in programming in BASIC. The second part will cover all other essential topics. The BASIC language was originally designed as a simplified version of FORTRAN for use in teaching programming. From the simple beginnings, the language has grown to become a very popular multi-purpose language available on a wide variety of machines.

WHAT YOU LEARN IN THIS COURSE

An electronic computer may be called an electronic “brain” but its functions and problem-solving ability depend on the intelligence of a human being who directs and controls the machine. This person is called a programmer and is responsible for giving the computer a set of instructions consisting of the necessary steps required to derive a solution to a given problem. The set of instructions that control the computer is called a program. The purpose of this course is to teach you how to program in BASIC.

The course will:

- i. Introduce you to basic hardware and software.
- ii. Teach you how to program in BASIC to a certain level.
- iii. Teach you how to use computer to process information.

WORKING THROUGH THIS COURSE

The major components of the course are:

1. Course Guide
2. Modules
3. Study Units
4. Further Reading (which is recommended)

ASSESSMENT

There are two aspects to the assessment of the course: continuous assessment and final examination. The tutor-marked assignments will form the continuous assessment. So you must do all and submit all to your tutor for grading. The date and place of final examination will be communicated to you by NOU.

COURSE OVERVIEW

Module 1

- Unit 1 Computers: Hardware and Software
- Unit 2 Computing: Arithmetic and Logical Operation
- Unit 3 Basic Statements
- Unit 4 Basic Statements (Continued)
- Unit 5 Listing and Debugging Programmes

Module 2

- Unit 1 Flowcharting and Basic Statements
- Unit 2 The Decision Statement
- Unit 3 The “IF” Statement; the ‘REM’
- Unit 4 Logical Expressions and the AND/OR Logical Operators
- Unit 5 The Counting Process and Basic Statements

Module 3

- Unit 1 The Accumulation Process and Statement FOR/NEXT
- Unit 2 The Accumulation Process
- Unit 3 Nested Loops
- Unit 4 Application of Nested Loops in Statistics
- Unit 5 Project

Module 4

- Unit 1 One – dimensional Arrays
- Unit 2 Use of Arrays
- Unit 3 DIM Statement
- Unit 4 Array Manipulation
- Unit 5 End – of – File Conditions

HOW TO GET MOST FROM THIS COURSE

In distance learning study units replace the University lecture. This is one of the great advantages of distance learning; you can read and work through specially designed study materials at your own pace, and at a time and place that suit you best. Think of it as reading the lecture instead of listening to the lecture. Just as a lecturer might give you in-class exercise, your study units provide exercises for you to do at appropriate points.

The following hints are useful:

1. Read this course guide schedule
2. Organize a study schedule
3. Once you have created you own study schedule, do everything you can to stick to it. The major reason that students fail is that they get behind their course work.
4. Turn to a unit and read the introduction and the objectives of the unit.
5. Use the order of the units.
6. Practice the exercises on a computer.
7. Debug your programs
8. Review the objectives for each unit at the end of the unit.
9. Do the Tutor-Marked Assignments
10. Start the next unit

SUMMARY

CST - is the first two semester course in BASIC programming. At the end of the first course you will be able to use the computer to do some exercises in the course and hope that you will find the course interesting and useful.

**MAIN
COURSE**

CONTENT	PAGES
Module 1	1
Unit 1 Computers: Hardware and Software	1
Unit 2 Computing: Arithmetic and Logical Operation..	4
Unit 3 Basic Statements	7
Unit 4 Basic Statements (Continued)	12
Unit 5 Listing and Debugging Programmes	18
Module 2	22
Unit 1 Flowcharting and Basic Statements.....	22
Unit 2 The Decision Statement.....	31
Unit 3 The “IF” Statement; the ‘REM’.....	36
Unit 4 Logical Expressions and the AND/OR Logical Operators	39
Unit 5 The Counting Process and Basic Statements.....	41
Module 3	46
Unit 1 The Accumulation Process and Statement FOR/NEXT	46
Unit 2 The Accumulation Process	49
Unit 3 Nested Loops	54
Unit 4 Application of Nested Loops in Statistics	55
Unit 5 Project	57
Module 4	58
Unit 1 One – dimensional Arrays	58
Unit 2 Use of Arrays	62
Unit 3 DIM Statement	65
Unit 4 Array Manipulation	67
Unit 5 End – of – File Conditions	72

MODULE 1

PROGRAMMING IN BASIC

Unit 1	Computers: Hardware and Software
Unit 2	Computing: Arithmetic and Logical Operation
Unit 3	Basic Statements
Unit 4	Basic Statements (Continued)
Unit 5	Listing and Debugging Programmes

INTRODUCTION

Basic was created in 1964 by J.G. Kemeny and T.E. Kurtz at Dartmouth College USA. The language was originally designed as a simplified version of FORTRAN for use in Teaching programming. From the simple beginnings, the language has grown to become a very popular multi-purpose language available on a wide variety of machines, in particular microcomputers and minicomputers. This course is therefore devoted to BASIC because of its usefulness

UNIT 1 COMPUTERS HARDWARE AND SOFTWARE

CONTENTS

1.0	Introduction
2.0	Objectives
3.0	Main Contents
3.1	Computers, what are they?
3.1.1	Organization of a computer (Hardware)
3.1.1	Computer Program (Software)
4.0	Conclusion
5.0	Summary
6.0	Tutor Marked Assignment
7.0	Reference/Further Reading

1.0 INTRODUCTION

In this unit you will be introduced to the components of a computer and their function. You will also learn what a program is.

2.0 OBJECTIVES

At the end of this unit, you would be able to:

- define a computer
- define a program

3.0 MAIN CONTENT

3.1 Computers, what are they?

Computers are automatic machines that can accept data, store vast amounts of information and perform arithmetic at high speeds to solve complex machines

3.1.1 Organization of a Computer

The computer may be thought of as a system composed of five components. The components and their functions are as follows:

1. **The Input Unit:** This unit feeds information from the outside world to the computer. Input units are capable of reading information recorded on such different mediums as punched cards or magnetic tape or from terminals. The information read from these devices is placed into appropriate memory locations.
2. **The Memory Unit:** The memory unit stores information. It holds the sequence of instructions necessary to solve particular problem and any additional data required. The memory is divided into locations that each have an address (are addressable). Instructions are stored in these cells.
3. **The Control Unit:** The control unit fetches the instructions and data from memory and executes the instructions one at a time with logical unit. All of the other components operate as directed by the control unit.
4. **The Arithmetic/Logical Unit:** The arithmetic/ logical unit consists of the electronic circuitry that performs arithmetic operations such as comparison of numbers.
5. **The Output Unit:** The output unit transfers or copies the contents of certain memory locations onto some external medium such as punched cards, punched paper tape, magnetic tape, a printed page produced by a teletype or line printer, or a cathode ray tube (CRT) screen for visual display.
The memory, control and arithmetic/ logical units are collectively called the central processing unit (CPU).

3.1.2 Computers Programs

The set of instructions that control the computer is called a program. A program can be executed by the computer only when it is stored in the computer's memory and is in machine language code. Machine language is the only language the computer can understand. It is a language in which arithmetic / logical operations are represented by machine – recognizable numeric codes and in which memory locations containing data and program instructions are represented by numeric addresses.

Machine language vary from one computer manufacturer to another and as such they are machine – dependent other types of languages, called high – level languages, have been developed to allow the user to formulate problems in a much more convenient and efficient manner. High-level languages are machine-independent; programs written in such languages can be processed on any type of computer. However, high-level languages must be ultimately translated into machine language before they can be executed by the computer. Special programs called language translators or compilers have, therefore, been developed to provide this translation service. BASIC (Beginner’s All – purpose Symbolic Instruction Code) is an example of a high – level language.

SELF-ASSESSMENT EXERCISE

- i. What is the name of the computer you will be using?
- ii. What procedure is needed to begin programming in BASIC?

4.0 CONCLUSION

A computer is a problem solving machine.

5.0 SUMMARY

A computer is composed of five components. The CPU may be thought of as the seat of intelligence of the entire system. The input and output functions may be performed by devices located at some distance from CPU.

6.0 TUTOR-MARKED ASSIGNMENT

1. List the five components of a computer system and explain the functions of each.
2. define the following:
 - (i) CPU
 - (ii) CRT
 - (iii) Program
 - (iv) Data.

7.0 REFERENCES/FURTHER READING

M. Boillot and L.W. Horn, BASIC Third Edition, West Publishing Company, New York.

UNIT 2 COMPUTING

CONTENTS

- 1.0 Introduction

2.0	Objectives
3.0	Main Contents
3.1	Arithmetic Operations
3.2	Types of Instructions
3.3	Sample Program
4.0	Conclusion
5.0	Summary
6.0	Tutor-Marked Assignment
7.0	Reference/Further Reading

1.0 INTRODUCTION

In this unit, you will learn how arithmetic operations are written. You will also be exposed to some basic instructions

2.0 OBJECTIVES

At the end of this unit, you would be able to:

- determine the symbols of arithmetic operations in BASIC
- explain how a computer carries out some simple instructions.

3.0 MAIN CONTENTS

3.1 Arithmetic Operations

In BASIC programs, arithmetic operations are indicated by the symbols:

+ addition
- subtraction
x multiplication
** or ↑exponentiation

3.2 Types of Instruction

We consider a simple model computer that operates conceptually as its real-life counterparts. The input is a terminal where numbers have been entered. (typed). The memory unit consists of a group of sequentially numbered “pigeonholes”. Locations can be referred to either by name or by address. The arithmetic/logical unit is represented by a desk calculator capable of performing arithmetic and logical operations. The role of the control unit played by a human operator who can fetch instructions from the memory one

at a time and execute them. The type of instructions the operator is capable of executing includes:

1. **Input:** Read a value from the input medium and store that value in a specified memory location e.g. INPUT F will cause one value on the input medium to be read and stored in a memory location called F.
2. **Conditional branding:** Perform a comparison test between two values using the logical unit and brands to a specified memory location if the test condition is met. If the condition is not met, no transfer occurs, and the following instruction is taken from the next memory location. For example, the instruction, IF $F < 50$
Go To 20
Cause the control to first fetch F from memory and then request the logical unit to compare the value with 50. If the logical unit reports that the value F is less than 50, the control unit fetched the next instruction is taken from the next sequentially numbered location. If F is not less than 50, it processes the next instruction.
3. **Calculations:** Perform calculations using the arithmetic unit and place results in desired memory location. For example, the instruction Let $M = 5 * F$ will cause the control unit to fetch the value contained in the memory location F and activate the arithmetic unit to multiply that value by 5. The final result is then stored in memory location P (product).
4. **Unconditional branding:** Take the next instruction from a specified memory location. For example, the instruction, Go to 700 cause the control units to fetch the next instruction from location 70 rather than from the next sequentially numbered location. Unconditional branding, then simply means transfer directly to a particular instruction in memory.
5. **Output:** Copy a value fro a memory location onto the output medium. For example, the instruction PRINT F, P will cause the contents of F and P to be written out on the output pad.
6. **Termination:** Cease execution of instructions for this program and wait for a new program to be loaded into memory. For example, the instruction, END causes the program to terminate. No more instructions in this program are executed.

3.3 Sample Program

Here we wish to calculate and write out an amount of pay owed an employee who is paid $=N=50$ per hour with time and $=N=100$ for all hours in excess of 40.

1. INPUT F
2. IF F > 40 GO TO 5
3. Let $P = 50 * F$
4. GO TO 6
5. LET $P = 2000 + 100 * (F - 40)$
6. PRINT P, F
7. END

4.0 CONCLUSION

Here you have been introduced to the workings of a computer

5.0 SUMMARY

The term software is generally used to describe the set of programs that causes the computer hardware to function.

6.0 TUTOR-MARKED ASSIGNMENT

How would the execution of the example 2.5, be affected by replacing the statement in location 7 by GO To 1? What would be the advantages and disadvantages of doing this?

7.0 REFERENCES/FURTHER READING

Boillot and Horn, BASIC, 3rd Edition

UNIT 3 BASIC STATEMENTS

LET, PRINT, TERMINATION, SYSTEM COMMANDS

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 Elements of BASIC Language
 - 3.2 Constants
 - 3.2.1 Numeric Constants
 - 3.2.2 Alphanumeric Constants
 - 3.3 Variables
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor Marked Assignment
- 7.0 Further Reading

1.0 INTRODUCTION

Here, you will learn some elements of BASIC, the character set and constants.

2.0 OBJECTIVES

At the end of this unit, you would be able to:

- use some key words LET, PRINT, END
- use characters which are used to define create BASIC instructions.
- define constants.

3.0 MAIN CONTENTS

3.1 Elements of BASIC Language

BASIC is a set of instructions telling the computer what actions (operations) to carry out to solve a particular problem. There are three types of BASIC operations that the computer can carry out. These are:

1. Arithmetic operations (add, subtract, multiply, divide, raise to the power)
2. Input/Output operations (entering data and printing results).
3. Comparison (determining whether one value is less than, equal

to, or greater than another value).

BASIC instruction is made up of different elements (BASIC keywords, constants, variables, expressions, arithmetic operators, relational operators etc) which, when combined together under a proper grammatical structure, yielded a valid BASIC instruction.

We consider the following problem; the investment T is given by the formula

$$T = P (1 + I)^N$$

Where I is the interest rate, N is the number of years, and P is the Principal. Mr. Ojo wants to deposit =N=1,500 at a savings institution. He considers a bank and a credit union. The credit union requires a non-refundable membership fee of =N=15. Deposits earn 12.25 percent at the credit union and 11.75 percent in the bank. Mr. Ojo will need the money in 2^{1/2} years. We wish to determine which institution that Mr. Ojo should use. The program is as follows:

```
10 LET p = 1500
15 LET I1 = .1225
20 LET I2 = .1175
25 LET N = 2.5
30 LET T1 = P * (1 + I1) ^ N - 15
35 LET T2 = p * (1 + I2) ^ N
40 PRINT "CREDIT AMOUNT", T1
50 PRINT "BANK AMOUNT", T2
55 END
```

The output

CREDIT AMOUNT 1987.43

BANK AMOUNT 1980.20

So Mr. Ojo will be advised to use credit union

3.2 Constants

A constant is a quantity whose value is fixed and explicitly stated constants may either be numeric or alphanumeric.

3.2.1 Numeric Constants

Numeric constants are positive or negative numbers; they can be added, subtracted, multiplied, divided and raised to a power by a computer. A constant is made up of any of the digits through a and may be preceded by the + or – symbol.

Examples of valid constants

300 - 2.31 62504. + 0.3 -14

Imbedded blanks (blanks between first and last digit) do not affect the value of the constant.

Example 6 32 = 632 = 6 3 2

Examples of invalid constant

1, 634, 123	No comma is allowed
23.24.	Only one decimal point is allowed
\$40.42	Character \$ is invalid
127 - 423	Character - is invalid

3.2.2 Alphanumeric Constants

Alphanumeric constants are generally words, messages, captions, column headings titles, names, addresses etc. that are endorsed within quotation marks.

Examples of valid alphanumeric constants

“1425 ADEYI ST”
“1425 ADEYI ST”
“IT IS HER’S”

Note: The maximum number of characters allowed in an alphanumeric string varies from one system to another.

Constants may be used in BASIC as follows:

- LET T1 = P + (1 + T1) ↑ N - 15
- PRINT -10.36, + 4.6 + 5.93, .005
- LET N\$ = “CHARLES”
- PRINT “\$”, A

3.3 Variables

Unlike a constant, a variable may assume different values. They are two types of variable.

1. Variables that contain numeric data can be processed arithmetically (added, subtracted etc)
 2. Variables that contain alphanumeric data cannot be processed arithmetically.
- The specifications for standard BASIC variable names are as follows:

Numeric Variables

A single letter of the alphabet such as A, B, Z, P or a single letter followed by one digit (0 – 9) such as B4, Z2, D0.

Alphanumeric Variables

Same as numeric variables except that a \$ is appended to the variable name,

Example: A\$, B\$, Z5\$

The following are invalid variables as indicated I the notes

2B Does not start with letter
KRB Too many letters
B% Invalid character %
C + 2 Invalid character +
BA\$ B2\$ would be valid
“HELLO” Is a constant

SELF-ASSESSMENT EXERCISE

Which of the following are invalid variable names?

1. A34
2. \$B
3. “B4”
4. \$

4.0 CONCLUSION

A BASIC instruction is made up of different elements (constants, variables etc).

5.0 SUMMARY

Here you learn some BASIC statements: LET, PRINT and what a BASIC instruction is made up of.

6.0 TUTOR-MARKED ASSIGNMENT

1. Specify which of the following are valid numeric constants, if invalid, specify the reasons;
 - a. 42 . 93.
 - b. + 45 .
 - c. + 45 . 25 CR
 - d. 124 . 000
 - e. 7 – 3
 - f. 27 %
2. Which of the following are invalid variable names? Why?
 - i. B43
 - ii. \$C
 - iii. B1 %
 - iv. “B4”
 - v. \$
 - vi. “M1\$”

7.0 REFERENCES/FURTHER READING

Boillot and Horn, BASIC 3rd Edition

UNIT 4 BASIC STATEMENTS (Continued)

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 Expressions
 - 3.1.1 Precedence in Expressions
 - 3.2 The Replacement statement LET
 - 3.3 Line Numbers
 - 3.4 The PRINT statement
 - 3.5 The termination statements STOP and END
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor Marked Assignment
- 7.0 Further Reading

1.0 INTRODUCTION

In this unit, we introduce to BASIC expressions. We shall also discover further Basic statements: LET, PRINT, STOP, END.

2.0 OBJECTIVES

At the end of this unit, you would be able to:

- work with BASIC expressions
- use BASIC statements: LET, PRINT, STOP and END.

3.0 MAIN CONTENTS

3.1 Expressions

An expression may be a constant, a variable or any combination of constants and/or variables linked by the five arithmetic operators shown below. Note that no two arithmetic operators may be typed side by side. Parenthesis may be included to denote the order of computations. The allowable arithmetic operators are:

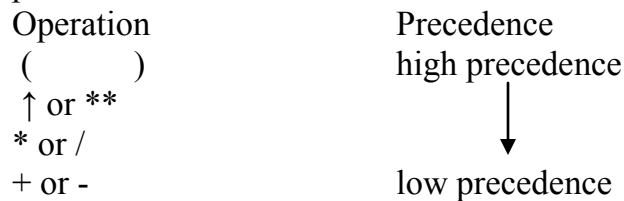
- + Addition
- Subtraction
- * Multiplication
- / Division
- ↑ or ** or ^ Exponentiation (raising to a power)

Valid Expressions

Algebraic Expressions	BASIC Expressions
+ 3.5	3.5
- c	- c
a.b -30	A*B -30
$\frac{a}{b} \cdot c$	(A/B) * c
$Ax^2 + bx + c$	A * X \uparrow 2 + B * x + c
$(a.b)^2$	(A*B) \uparrow 2
$(-c + 1.4)d$	(-c + 1.4) * D
\sqrt{x}	X \uparrow .5
$\sqrt[4]{(a-b)^3}$	((A -B) \uparrow 3) ^ .25
$\frac{\text{Cost} - \text{salvage}}{\text{Years}}$	(c - s) / Y
Bonus + hours x rate	B + H * R
$\frac{1}{r_1} + \frac{1}{r_2} + \frac{1}{r_3}$	$\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$

3.1.1 Precedence in Expressions

Operations within expression are performed according to the following rules of precedence.



Operations with high precedence are performed before operations with lower precedence. The operations addition/subtraction, multiplication/division are performed in order from left to right according to the rule of precedence. Exponentiation is performed in order from right to left. If parenthesis is nested, the operation in the innermost set of parenthesis is performed first.

Consider the expression

$$3 + (2 * 2 \uparrow 3 / 4 + 1) - 2 / 3$$

1. $A - B + C$ B is subtracted from A, and the result is added to C
 $3 - 2 + 5 \quad (3-2) + 5 = 1 + 5 = 6$
2. $A + B * C$ Since multiplication has priority, $B * C$ is computed; the result is then added to A giving $A + (B * C)$.
 $3 + 2 * 3 \quad 3 + (2 * 3) = 3 + 6 = 9$
3. $A/B * C$ Since multiplication and division have the same priority B is first divided into (A/B) , and the result of the division is multiplied by C
 $9./ 4. * 2 \quad (9./ 4.) * 2 = 2.25 * 2 = 4.50$
4. $A/B/C$ First A/B is performed, and the result is then divided by C $8/4/2 \quad 8/4 \div 2 = 2 \div 2 = 1$
5. $(A + B) / C * D = ((A + B) / C) * D$
 So $(3 + 6) / 3 * 6 = ((3 + 6) / 3) * 6 = 3 * 6 = 18$
6. $A + B * C \uparrow 2 = A + (B * (C \uparrow 2))$
 $3 + 3 * 2 \uparrow 2 = 3 + (3 * 2^2) = 3 + (3 * 4) = 3 + 12 = 15$
7. $A \uparrow B \uparrow C = (A \uparrow B) \uparrow C \quad 3 \uparrow 2 \uparrow 3 = (3 \uparrow 2) \uparrow 3 = (3^2)^3 = 9^3 = 729$

SELF-ASSESSMENT EXERCISE

Evaluate the following expressions for $X = 8$, $Y = -4$, $Z = 5$

1. $X / Y + 2$
2. $X + 2 * Y / Z + 1$
3. $1 + (X \uparrow 2 - 1)$
4. $X / Y / Z * 2$

3.2 The Replacement Statement LET

A replacement statement evaluates an expression and stores the value of the expression in a memory location identified by a variable name. The general form of a replacement statement is



In most BASIC systems, the key word LET is optional and need not be specified by the user. When the particular statement is carried out (RUN) the value of the expression is first computed and the result is placed (stored) in the variable (memory location) specified on the left – hand side of the equal sign. When we write $10 X = 2 * X + 1$ We are looking at the expression and saying “find out what is in X, multiply it by 2, add 1, and store the result in memory location X. the equal sign must be understood as a replacement sign. Thus, if the memory location contains 4, then $2 * X + 1 = 9$ and the value 9 is stored in X. i.e. the value X is replaced by 9.

Examples of valid replacement statements are:

- 10 LET X = -0.07 Store value - .07 in X
12 X1= A + B / C -1 Evaluate the expression and store the
 result in X1
14 LET C\$ = “GROSS PROFIT” Store the string “GROSS PROFIT” in C\$
15 I = I + 1 Computer I + 1 and store result in I

3.3 Line Numbers

Each BASIC line must be numbered. Line numbers must be unsigned integers not exceeding five digits. In most systems, statements or lines may be entered in any desired order; the system will sort the statements in order when the program is list or executed. Statements will be processed by the computer in ascending order under a decision statement (IF) or unconditional transfer (Go To) is encountered.

Example

- 10 LET X = 3
20 PRINT X, Y
15 LET Y = X ↑ 3

BASIC would store these statements in its memory in the following orders

```
10 LET X = 3
15 LET Y = X ↑ 3
20 PRINT X, Y
```

3.4 The Print Statement

The general form of the PRINT statement is

Line number PRINT expression List

Where the expression list can be, alternatively,

A constant	PRINT	7.6
A variable	PRINT	X
An expression	PRINT	(H - 50) * R * 2.5 + H * R
A character string	PRINT	"PAY HOURS RATE"
A combination of the above separated by commas or semi-colons	PRINT	"PAY = "; P; "HOURS*=" ; H

3.5 The Termination Statements STOP and END

The last BASIC statement in a BASIC program should always be the END statement. For this reason, the END statement should always have the highest line number in your program. If you type BASIC statements after the END statement, these will not be processed by BASIC.

The STOP statement tells BASIC where in your program you want to stop processing instructions. The general form of termination statements are;

Line number	STOP
Line number	END

Any program can be written without a STOP statement.

4.0 CONCLUSION

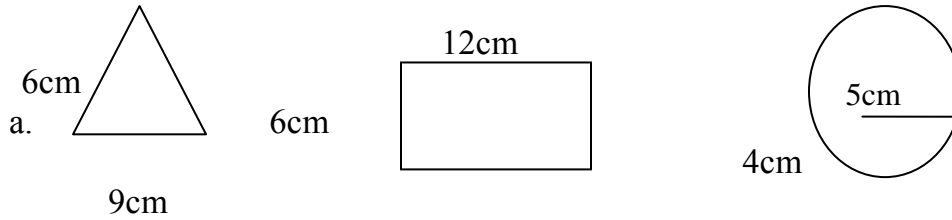
It is important to follow BASIC instruction correctly.

5.0 SUMMARY

Here you learn what expressions are and the order of precedence. You are now in a position to use BASIC statements: LET, PRINT, STOP and END correctly.

6.0 TUTOR-MARKED ASSIGNMENT

Write a program to compute and print the area and perimeter of each of the following:



7.0 REFERENCES/FURTHER READING

Boillot and Horn, BASIC, 3rd Edition

UNIT 5 LISTING AND DEBUGGING PROGRAMS

NUMBER REPRESENTATION AND PROGRAMMING EXAMPLES

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 Bugs
 - 3.2 Listing and Debugging Programs
 - 3.3 Number Representative
 - 3.4 Programming Example
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor-Marked Assignment
- 7.0 Reference/Further Reading

1.0 INTRODUCTION

By now you may have realized how easily one can make errors on a computer. Errors commonly called bugs, come in essentially four varieties: typographical, syntactical, logical, and system related.

2.0 OBJECTIVE

3.0 MAIN CONTENTS

3.1 Bugs

1. Typographical Errors:
These errors are due to mistypes and they are interpreted by the computer as syntax errors. E.g. letter O is typed instead of digit 0.
2. Sy-ntactical Errors:
Syntax error reflects the programmer's inability to observe correctly the grammatical rules of the BASIC language.

Examples

```
10 LET P + 1 Typographical error + should be =
20 LET X = A (B + C)
```

```

20 LET X = A * (B + C)
30 PRINT "HELLO" S1 S2 Punctuation missing between items
40 LET Z = (3 + X * (4 + T) Missing right parenthesis

```

3. Logical Error:
 Logical errors are the most difficult errors to detect because they are very well camouflaged.

Example:

```

LET Y = T / N

```

Although, it is a perfect instruction. Yet it could be a time bomb during the execution if $N = 0$.
 Also a programmer may wish to print the result stored in memory location R, but instead write the instruction PRINTS.

4. System Errors:
 Systems errors relate t the computer system itself, for example, a printer may not be turned on when the CPU needs it, thereby causing a special error message to appear.

3.2 Listing and Debugging Programs

To debug a program means to take the bugs out of a program i.e. to correct it consider the following program

```

10 LET H = 40 The user types letter O instead of zero.
15 LEM R = 5.50 LEM is typed instead of LET
20 LET P = H + R Also there is an error due to the omission
    of PRINT instruction.
25 END
RUN

```

3.3 Number Representation

Constants in BASIC can be expressed in two different notations

1. Decimal notation, with or without decimak points.
Examples: 45.67, -12.3, 11240, -48216.
3. Exponent notation, where the character E is used to represent the decimal base 10, followed by a two digit (with or without sign) to represent the exponent (power).

Examples:

```

10 LET X = 3.456 E + 2 where  $3.456 E + 2 = 3.456 \times 10^2$ 

```

- 12 LET Y = -3. E -3 where $-3.E-3 = -3 \times 10^{-3} = -.003$
 14 LET Z = 1.23 EO where $1.23 EO = 1.23 \times 10^0 = 1.23$

On output, as a result of the PRINT statement the BASIC will use E notation to represent a constant if the value of the constant is outside a certain range. The range will vary from one computer to another.

Example:

```

10 LET X = 123456
20 LET Y = 123456 789012345
30 LET T = .0003245
35 LET Z = 22.1 E 3
45 LET V = -.001
48 LET W = .01
49 PRINT X; Y; T; Z; V; W
    RUN
123456      1.2345679E + 14    3.245 E - 4    22100 -1E-03 .01
  
```

3.4 Programming Example

Income Calculation

Mr. V is a widower with three children aged 12, 10 and 19. His monthly salary is \$1,523.36. His monthly contribution to a retirement plan is 6.6 percent of his first 9 month’s salary; retirement deductions are spread over a 12 month period for each child support for social security. His monthly social security deduction is 6.7 percent of his monthly income, and his federal income tax is 15.6 percent of his yearly gross (deducted on monthly basis). Monthly payments for life insurance equal 9.6 percent of his monthly salary after social security and federal tax deductions. Write a program to compute his monthly spendable income.

```

10 LET C = 119.25      Support per child
15 LET I = 1523.36    Monthly salary
20 LET R = (9*.066)/12 Monthly retirement plan deduction
25 LET S = .067 * I   Monthly social security deduction
30 LET F = .156 * I   Monthly federal income tax
35 LET T = I - (R + S + F) Net after deductions
40 LET L = .096 * (I - F - S) Monthly life insurance payments
45 LET T1 = T - L     Net minus life insurance payments
50 LET T1 = T1 + 2 * C Plus child support (2 less than 18 years)
55 PRINT "SPENDABLE INCOME IS"; T1
60 END
    SPENDABLE INCOME IS 1233.11
  
```


4.0 CONCLUSION

Errors are called bugs. Debugging means correcting errors

5.0 SUMMARY

They are four varieties of errors and you learn here how to correct them. The program example i.e useful and should be studied in detail.

6.0 TUTOR MARKED ASSIGNMENT

Write a program to produce report on cost of operating electrical devices.

COST ANALYSIS

WATTS	HOURS	COST/W	COST
60	6	.087	-
100	6	.08	-

The formula is

$$C = \frac{W \cdot T \cdot K}{100}$$

Where:

W = number of watts

T = line in hours

K = Cost in watts per kilo watt hour

7.0 REFERENCE/FURTHER READING

Boillot and Horn, BASIC, 3rd Edition

MODULE 2

Unit 1	Flowcharting and Basic Statements
Unit 2	The Decision Statement
Unit 3	The “IF” Statement; the ‘REM’
Unit 4	Logical Expressions and the AND/OR Logical Operators
Unit 5	The Counting Process and Basic Statements

UNIT 1 FLOWCHARTING AND BASIC STATEMENTS

CONTENTS

1.0	Introduction
2.0	Objectives
3.0	Main Contents
3.1	Input statement
3.2	The unconditional transfer statement go to
3.3	Flowcharting
4.0	Conclusion
5.0	Summary
6.0	Further reading
7.0	Tutor marked assignment

1.0 INTRODUCTION

Flowchart is a set of symbols linked by directed lines which represent a sequence of operations for flowchart is essentially a pictorial representation (as sort of visual outline) of the sequence of steps that must be taken in order to solve a particular problem.

2.0 OBJECTIVES

At the end of this unit, you would be able to:

- use input statement
- go to statement
- draw a flowchart of simple programs

3.0 MAIN CONTENTS

3.1 Input Statement

Below is the program to compute each employee of a company's gross pay.

05	REM PAYROLL PROGRAM	The REM instruction is to document the program when the program is Ron, the
10	PRINT "ENTER NAME"	INPUT instruction line 15 will cause the
20	PRINT "ENTER HOURS"	computer to stop and display a question
25	INPUT H	mark (?) on the screen. The user enters a
30	PRINT "ENTER RATE"	name and presses the RETURN key This
35	INPUT R	causes the name entered on the screen to
40	REM	be stored in memory location N \$
45	IF H > 40 GO TO 65	If condition is true, transfer to 65 to
50	LET P = H * R	compute overtime, otherwise ($H < 40$)
55	GO TO 95	the computer automatically goes to the
60	REM	next instruction 50
65	LET P1 = 40 * R	
70		LET V = H - 40 The transfer to line 65 is made
		only if the condition $H >$ at line 45 is true
75	LET P2 = V * R * 1.5	
80	LET P = P1 + P2	P1 is the first 40 hours at regular rate
85	PRINT "OVERTIME HOURS = "; V	V is the overtime pay
90	PRINT "OVERTIME PAY = "; P2	D is the final pay
95		PRINT "PAY TO", N \$; P Line 90 or as
		a result of line 55
98	GO TO 10	

When the input instruction is executed (run), the following happens

1. The computer types a question mark on the terminal
 2. The computer stops the user at this point should enter a value for each of the variables specified by the INPUT statement. These values are then stored in the computer memory in the specified memory locations when the ENTER / RETURN key is pressed.
- 10 PRINT "ENTER AGE PLEASE" Note that when the input instruction is typed and the user presses ENTER key to get to the next instruction (20), the computer does not display a? and it does not stop. This only happens when RUN is typed
- 15 INPUT X
- 20 PRINT "YOU FIBBER, YOU ARE NOT"
- 25 PRINT X; "YEARS OLD" RUN

ENTER AGE PLEASE

? 41 (entered by user) Output produced by instruction above

You FIBBER, YOU ARE NOT 41 YEARS OLD

Remark: Note the GO TO 10 statements at the line program repeatedly. The process of repeating a sequence of instructions is called a Loop. In this case the loop is infinite and the only way the user can escape from it is by pressing a special key on the keyboard (see your user's manual)

The unconditional Transfer Statement GO TO

The General form of the Go to Statement is

line number Go TO transfer line number
--

A BASIC program consists of a sequence of BASIC STATEMENTS. Basic will process these statements one after another in sequential order. When BASIC encounters a GO TO statement it will transfer control to the statement specified, that is, processing will continue at the transfer number. It also allows the program to branch back to repeat (reprocess) certain instructions or certain procedures; this is called **looping**

Examples 1

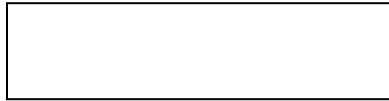
```
GOT TO 50  
20 - }
```

This code is by passed

```
50 LET Y = 3 * Z
```

Example 2

```
20 INPUT H, R, B
```


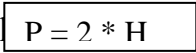


This block of code is processed repeatedly.

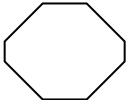
50 GO TO 20 variables) is used. The contents (value) of each location are to be printed on some output device. For example, the Block Indicates that values in locations P and H are to be displayed (written) on some device.

```
PRINT P, H
```

a. The Processing Block

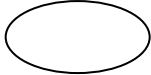
A rectangular symbol  used for processing instructions. The most common form for expressing these instructions is the replacement statement. A replacement statement specifies the arithmetic operations to be performed on constants / or variables and the location (variable) into which the value computed is to be placed. For example, the block  specifies that the contents (value) of H is to be multiplied by 2 and the result be placed into P.

b. The Decision Block

The diamond – shaped symbol  is used to denote decisions. A common means of expressing a decision is in terms of a Yes/No question that can be answered Yes or No. The question must involve only mathematical relations such as equality (=), less than (<)


Flowchart is on the logic required for solving a problem rather than on the mechanics or specifics of a programming language.

c. The Terminal Block

An oval – shaped symbol  is used to mark the point at which execution of instructions is to begin and end. The instruction START may be used to mark the beginning point; the instruction END or STOP may be used to mark the ending point. A

flowchart may have only one starting point (entry point) but may have many ending points.

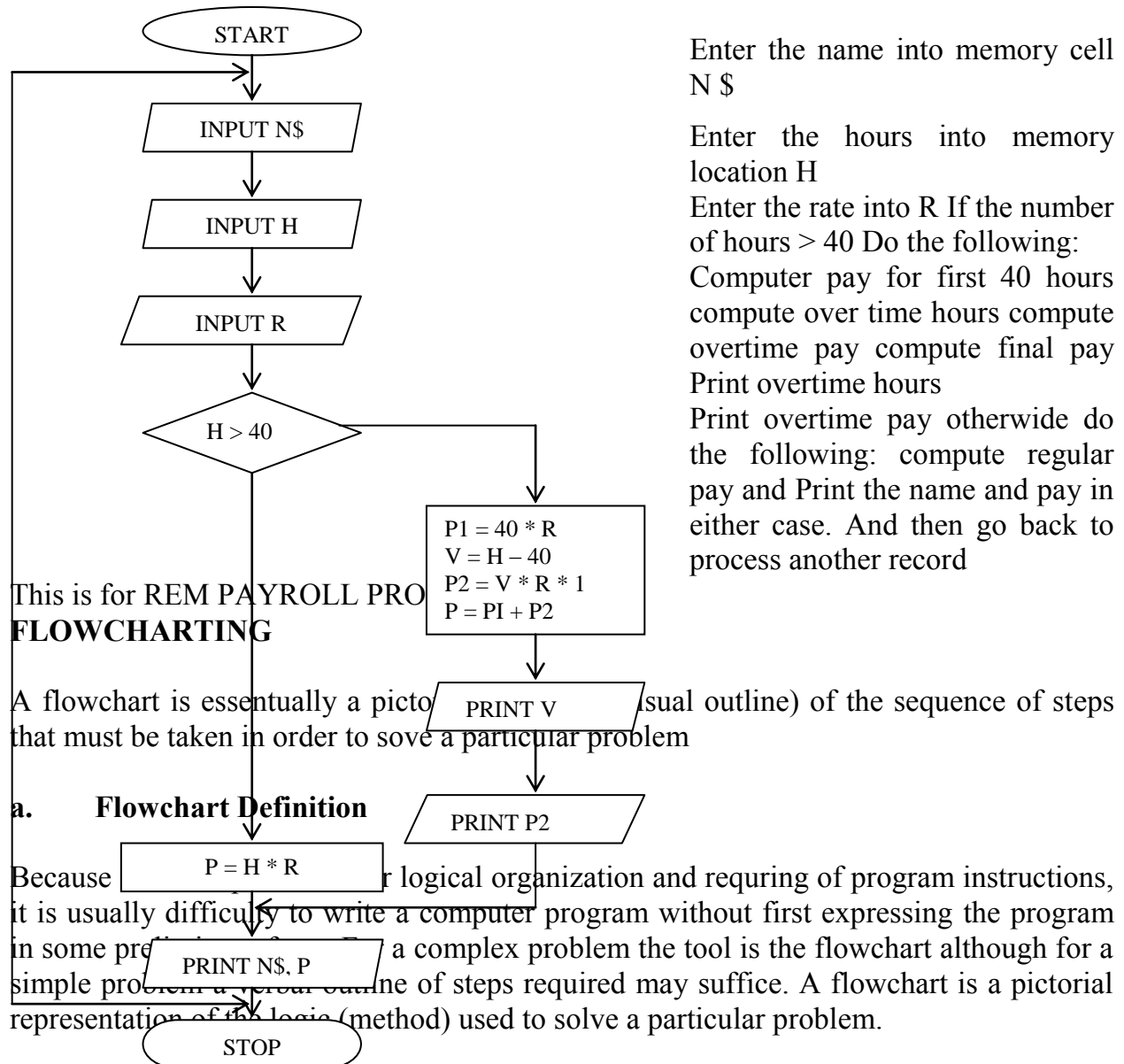
d. The Input / Output Block

A parallelogram – shaped symbol  is used for input and output operation, the instruction (command) READ OR INPUT, followed by a list of names (variables) separated by commas, is used. These names or variables can be thought of as symbolic names given to memory locations in which the data read is to be stored.

Variables names should be chosen to convey the nature of the data to be processed



For an output operation, the instruction PRINT, followed by a list of memory locations



Enter the name into memory cell N \$
 Enter the hours into memory location H
 Enter the rate into R If the number of hours > 40 Do the following:
 Computer pay for first 40 hours compute overtime pay compute final pay Print overtime hours
 Print overtime pay otherwise do the following: compute regular pay and Print the name and pay in either case. And then go back to process another record

This is for REM PAYROLL PROGRAM FLOWCHARTING

A flowchart is essentially a pictorial (visual outline) of the sequence of steps that must be taken in order to solve a particular problem

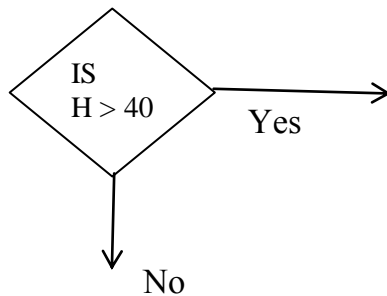
a. Flowchart Definition

Because of the logical organization and requiring of program instructions, it is usually difficult to write a computer program without first expressing the program in some pictorial form. For a complex problem the tool is the flowchart although for a simple problem a verbal outline of steps required may suffice. A flowchart is a pictorial representation of the logic (method) used to solve a particular problem.

A flowchart is particularly useful for visualizing paths through the logic of a program

b. Flowchart symbols

In program flowcharts, the symbols shown in the figure below is normally used. Each symbol or block represents a different type of operation written within the blocks are instructions to indicate (in general terms) what operation is to be prformed. It is not necessary to express the instructions used in a flowchart block in any particular computer language. The emphasis in the less than or equal to (\leq), greater than ($>$), greater than or equal to (\geq) or nhot equal to ($>$). The decision blow is the only block from which two different logical paths may be selected follows indicate the path to be taken depending on the decision block.



If the value of H is greater than 40, the path marked YES is taken, other wise the NO path is taken.

c. Flowlines


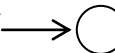
The sequence of instructions to be executed in a flowcharted algorithm is denoted by straight lines with an arrowhead such as \longrightarrow or \downarrow . The direction of flow is always in the direction pointed by the arrowhead.

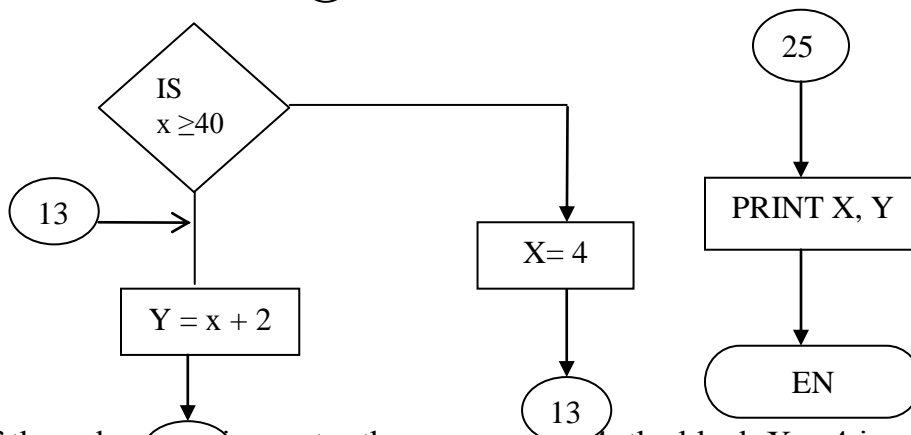
d. Connector Blocks

When it is convenient to draw flowlines to connect one area of the flowchart to another, connectors are often used. Connectors serve two purposes:

1. To identify a block by a label for reference purposes
2. To indicate transfer to another labeled block.

The symbol used for a connector is a circle.

A label is placed in the connector block when the flow line points away from the connector, such as . The connector is being used to denote an entry point, that is, a block to which transfer will be made from some point in the flowchart when the flowline points toward the connector, as in , the connector is being used to indicate a transfer.



If the value 25 is greater than or equal to 4, the block $X = 4$ is executed and transfer is made to the entry point labeled 'B'. If X is not greater than or equal to 4, the block $Y = x + z$ is executed and transfer is made to the entry point '25' which contains the PRINT instruction.

4.0 CONCLUSION

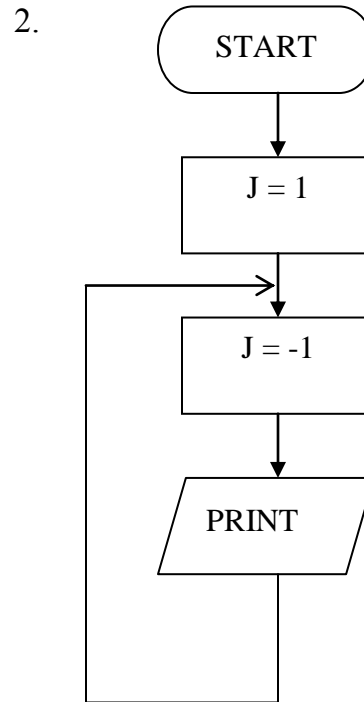
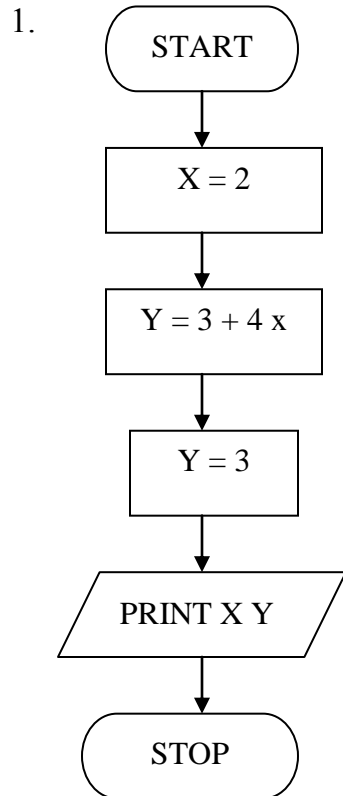
In a flowchart it is possible to have many connectors

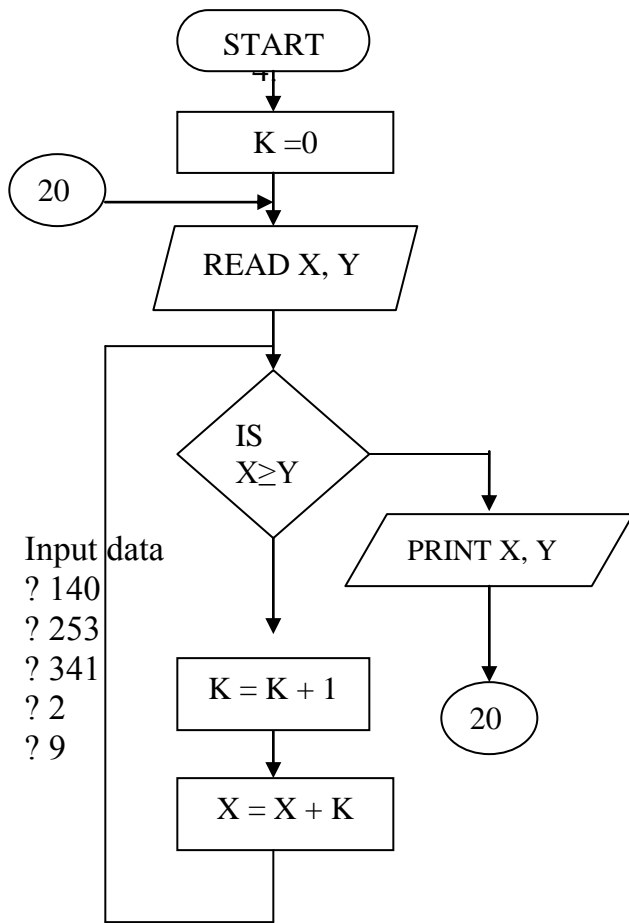
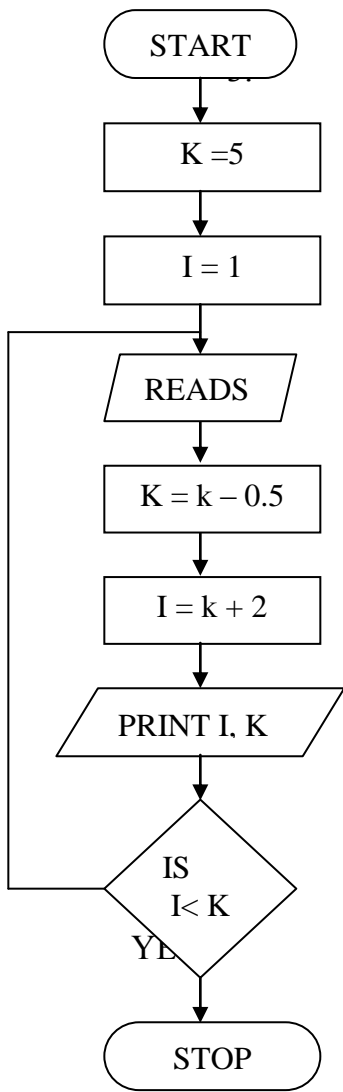
5.0 SUMMARY

Here you are introduced to BASIC statements INPUT and GO TO. You also learn how to write flowcharts.

6.0 TUTOR-MARKED ASSIGNMENT

Determine the output produced by each of the following flowcharts





Input data
 ? 140
 ? 253
 ? 341
 ? 2
 ? 9

NO

7.0 REFERENCE/FURTHER READING

Boilot and Horn, BASIC, 3rd Edition

UNIT 2 BASIC STATEMENTS: THE DECISION STATEMENT

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 If / then
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Further reading
- 7.0 Tutor marked assignment

1.0 INTRODUCTION

The central processing unit (CPO) has a logical unit which allows the computer to compare one number with another as well as one alphanumeric string with another. This decision-making capability is available to BASIC through the If statement which allows the computer to transfer to a nonsequential instruction.

2.0 OBJECTIVES

At the end of this unit, you would be able to:

- use IF / THEN statement

3.0 MAIN CONTENTS

3.1 If / Then

Then general form of the If statement is

line number if condition {GO TO THEN} transfer line number
--

Where the condition consists of two arithmetic expressions or character string variables linked together by one of the relational operations shown in figure 1 below:

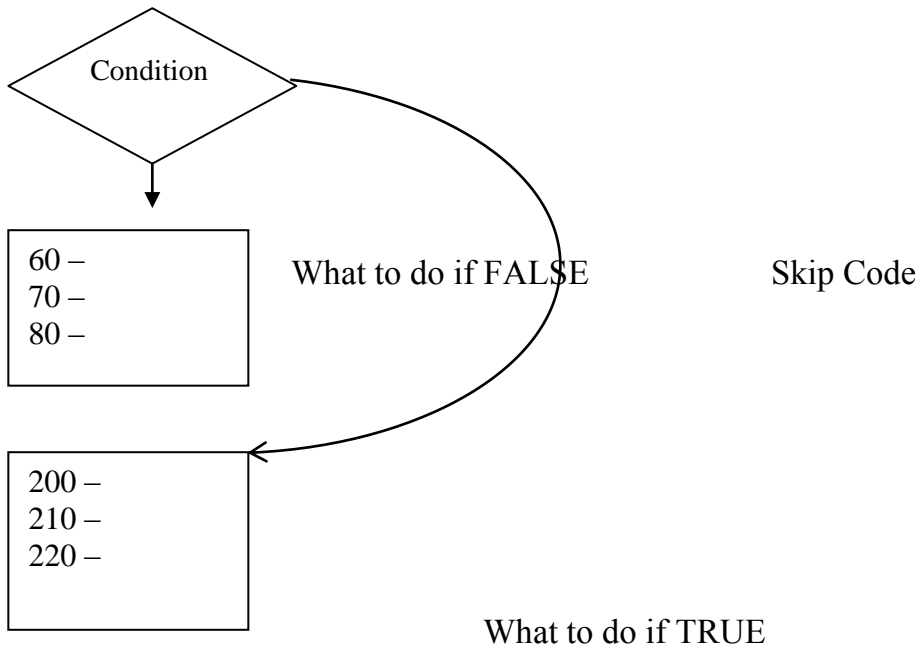
BASIC Operators	Relational	Mathematical symbol	Meaning
=		=	Equal to
<		<	Less than
< = or = <		≤	Less than or
>		>	Equal to
> = or =>		≥	Greater than
< > or > <		≠	Nor equal to

Figure 1

BASIC relational Operators

The If statement can be flowcharted as follows

If condition GO TO 200



Example 1

Write the code to input an age and print the message “OLD” if the age > 50, and print “YOUNG” otherwise, then stop

```
10 INPUT A
```

```
NO 20 IF A > 50 GO TO 80
    >30 PRINT “YOUNG”
    40 GO TO GO
    80 PRINT “OLD”
    90 END
```

Transfer to statement 80 if the age is > 50 otherwise print YOUNG than go to go is stop if age > 50 skip over lines 30 – 40 and print OLD

Note the importance of statement 40. IF line 40 had been omitted, then if the age were less than or equal to 50 not only would the message YOUNG be printed, but also the message OLD.

Example 2

Write the code to input a persons name and a corresponding pay. If the name read happens to be LUCKY, add \$ 100 to the pay. In any event, print the persons name and the pay.

```
10 INPUT N$, P
```

```
NO 20 IF N$ <> “LUNCKY” GO TO 40
    >30 P = P + 100
    40 PRINT N$, P
```

If the name is not LUCKY, go and print name and pay, otherwise, if its LUCKY, add 100 to pay and print results

Example 3

Write the code to input a code with value either 1 or 2. If the code equals 1, compute the circumference of a circle of radius 1 – 34, if the code has value 2, compute the area of a circle of radius 3.2. Print results.

```

10 INPUT K
20 IF K = 1 GO TO 50
25 A = 3.14, 3.212
30 PRINT "AREA ="; A
40 GO TO GO YES If K = 1, computer circumference
50 C = 2 * 3.13 * 1.344
55 PRINT "CIRC ="; ; C
90 END

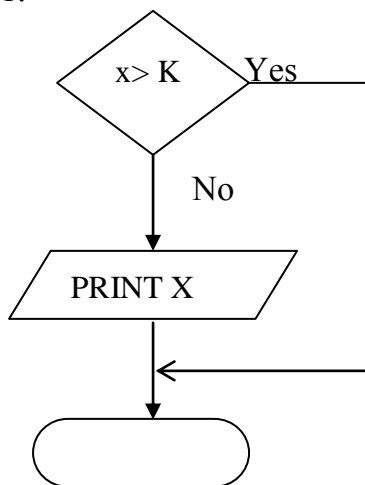
```

Line 25 is executed if K is not equal to 1

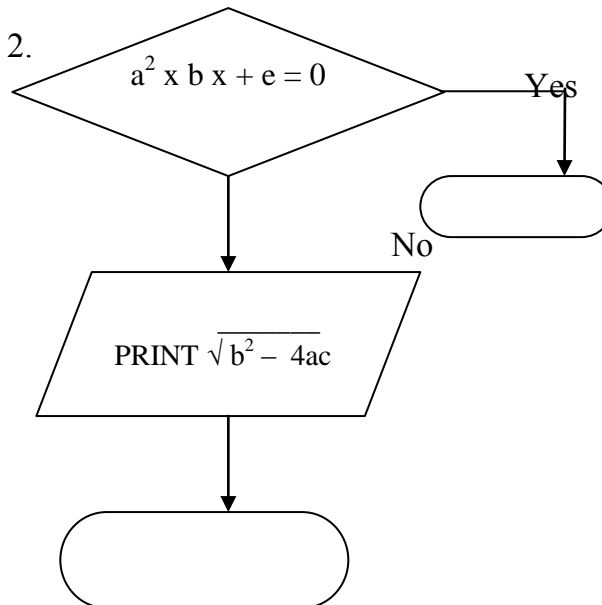
If K = 1, computer circumference of circle.

Write the BASIC code for the following flowcharts

1.



2.



Answers

1. 10 If $x > 0$ GO TO 99
 20 PRINT X
 99 END
2. 10 IF $a * X^2 + b * X + C = 0$ THEN 99
 20 PRINT $(B^2 - 4 * A * C) . 5$
 99 END

4.0 CONCLUSION

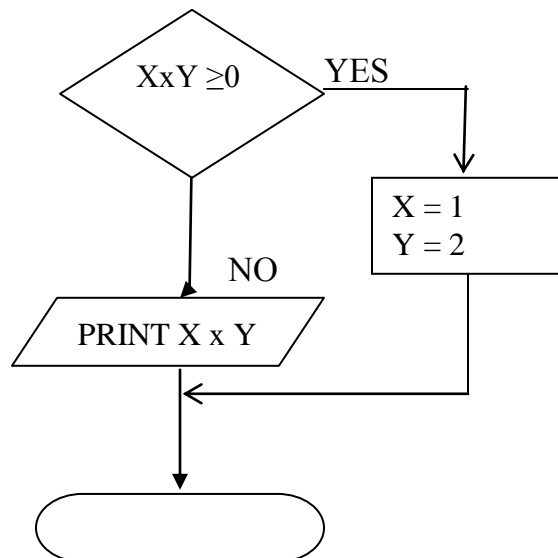
If the condition specified is true, control will be transferred to a block code that does not immediately follow the if statement.

5.0 SUMMARY

The line number specified by the GO TO statement in the If instruction need not physically follow the If statement.

6.0 TUTOR-MARKED ASSIGNMENT

1. Explain what would happen in example 3 above if at line 20 we had If $K = 1$ THEN 25
2. Write the BASIC code for the following flowcharts



7.0 REFERENCE/FURTHER READING

UNIT 3 THE IF STATEMENT REVISITED; THE REM STATEMENT

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 If Statement Revisited
 - 3.2 The Rem Statement
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Further reading
- 7.0 Tutor marked assignment

1.0 INTRODUCTION

In many BASIC systems the If statement is some what more powerful than has been previously shown as gen will learn here.

2.0 OBJECTIVES

At the end of this unit, you would be able to:

- use IF statement further
- use REM statement

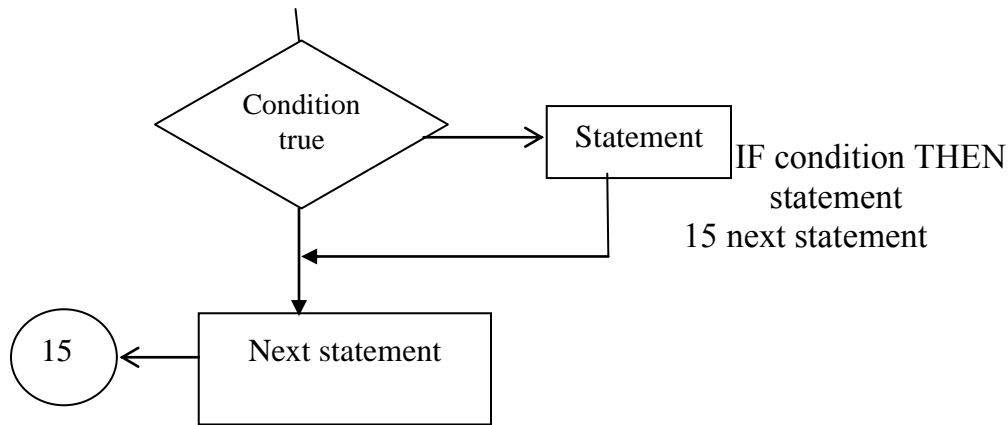
3.0 MAIN CONTENTS

3.1 The If Statement Revisited

Another form of the If statement is

line number if condition THEN statement

Where the statement can be an input / output instruction, a replacement statement or a STOP. If the condition is met, the statement (and only one) is executed and the control is transferred to the condition is not met control is transferred to the statement following the If. This If statement can be visualized as follows



Example 1

10 If $A < B$ THEN $K = K + 1$

15 If $A = 4$ GO TO 67

If $A > B$ the statement $K = K + 1$ will be executed after which statement 15 will be executed. If $A \geq B$ 15 will be processed next.

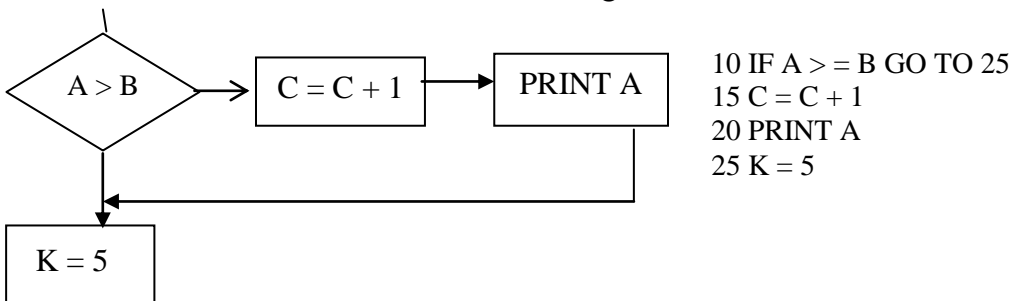
Example 2

10 If $C \neq 0$ THEN PRINT A

15 STOP

IF $C \neq 0$ print the value A and stop otherwise ($C \neq 0$) stop

Since only one statement is permissible after the THEN keyword the following code would be needed to account for the following flowchart



3.2 The REM Statement

The general form of REM statement is

line number REM literal characters

Where REM (remark) is a key word, and the literal characters are supplied by the programmer. Although the REM statement requires a line number, it is not an executable statement.

4.0 CONCLUSION

REM statement help understand the overall program structure

5.0 SUMMARY

The If statement should be used with care.

6.0 TUTOR MARKED ASSIGNMENT

1. Write a program to determine the roots of the quadratic equation $ax^2 + bx + c = 0$ if there are no real roots, print the message "NO REAL ROOTS"
2. If the roots are unequal, print message "ROOT 1 = xx, ROOT 2 = xx"

7.0 REFERENCE/FURTHER READING

Boillot and Hon, BASIC, 3rd Editor

UNIT 4 LOGICAL EXPRESSIONS AND THE AND / OR LOGICAL OPERATORS

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 And
 - 3.2 On / Go To Statement
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor Marked Assignment
- 7.0 Further Reading

1.0 INTRODUCTION

A logical expression can be thought of as a proposition. A proposition is a statement that is either true or false.

2.0 OBJECTIVES

At the end of this unit, you would be able to:

- use AND / OR

3.0 MAIN CONTENTS

3.1 And

Examples

1. To determine whether A lies between 7 and 14, the following statement could be used:
If A > 7 AND A < 14 THEN PRINT "YES" Note that the statement If A > 7 AND A < 14 is invalid. No logical operator may be side by side with a relational operator.
2. Parentheses may be used to devote the order in which the expressions are to be evaluated
If (S = 1 AND M1 <> 0) OR (S = 3 AND M1 > 4) THEN PRINT S, M1

3.2 On / Go To Statement

The ON/GO TO statement is a useful and convenient statement that allows transfer to many different points in a program

Line number ON expression GO TO line number line

Another form of the If statement is

line number on expression GO TO line number 1, - - line number n
--

The rule is the following

If the value of the expression is 1 control is transferred to line number 1

If the value of the expression is 2, control is transferred to the line number 2

Examples

20 ON N GO TO 3, 57, 100, 4. IF N = 3 control is transferred to 100

15 ON J - 2 * K GO TO 30, 10 IF J - 2 * K = 1 go to statement 30

30 ON Y ↑1 GO TO 10, 20, 60 IF Y ↑ = 1, 2, 3, go to 10, 20 and 60 respectively.

4.0 CONCLUSION

A complete program is useful in understanding how to use ON / GO TO as we shall show later

5.0 SUMMARY

Here you learn how to use AND / OR Logical operators and ON / GO TO.

6.0 TUTOR-MARKED ASSIGNMENT

Final grades in a course are determined by adding scores obtained on there tests T₁, T₂ and T₃ students get a PASS grade if the sum of the three scores is above 186 and a FAIL otherwise write a program to enter three scores, and determine the grade. Print the input scores, the average, and the final letter grade.

7.0 REFERENCE/ FURTHER READING

Boillot and Horn, BASIC, 3rd Editor

UNIT 5 THE COUNTING PROCESS AND BASIC STATEMENTS READ / DATA, REST 0 RE, PRINT USING

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 Read Statement
 - 3.2 Data Statement
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor marked assignment
- 7.0 Reference/Further Reading

1.0 INTRODUCTION

This unit introduces a new programming concept, counting, which is of great importance in writing programs.

2.0 OBJECTIVES

At the end of this unit, you would be able to:

- use the programming concept counting

THE COUNTING PROCESS AND BASIC STATEMENTS

```
10 REM THE RESPONSES ARE
```

```
11 REM RECORDED ON THE
```

```
12 REM DATA STATEMENTS
```

```
15 LET I = 0
```

```
20 READ C
```

```
25 IF C = 9 THEN 50
```

```
30 PRINT C
```

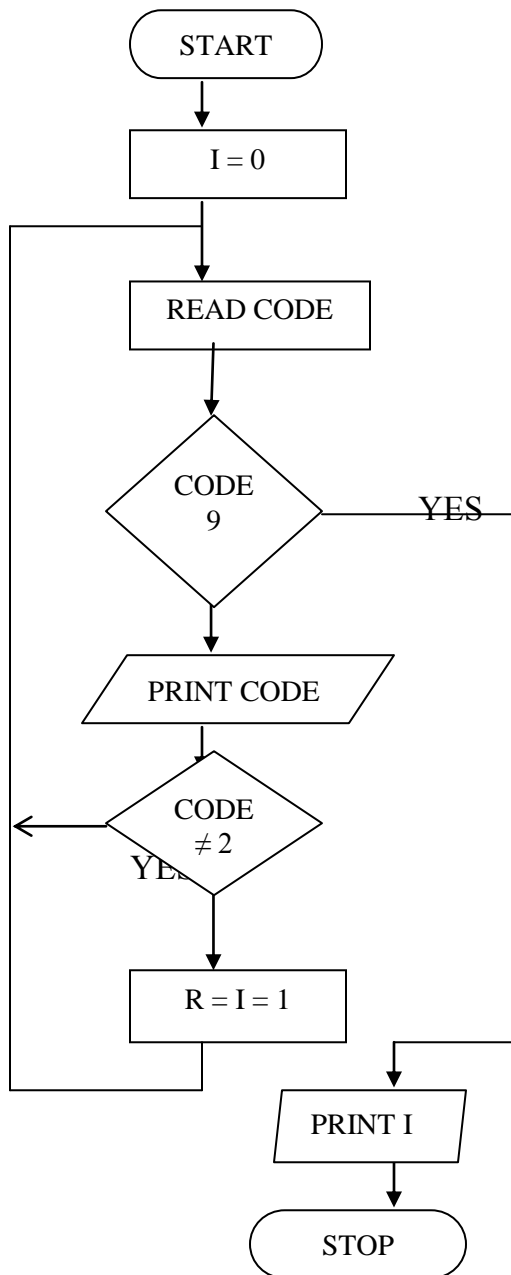
```
35 IF C <> 2 THEN 20
```



```

40 LET I = I + 1
45 GO TO 20
47 DATA 0, 1, 1, 0, 2, 0, 0, 1
48 DATA 2, 2, 0, 0, 0, 1, 2, 9
50 PRINT "ISOLATION ISTS =" ; I
99 END

```



is used to count number of isolationist students

Read Code

Is it the last ON?
Yes; go and write out total of isolationists

No; print code

Is it an isolationist response

No; go read another code Yes; count it and read more codes

Print total of isolationists students

SELF-ASSESSMENT EXERCISE

1. Modify the program to determine the number of students
 - . for isolationism
 - . neutral
 - . against isolationism
2. Modify the program to determine the percentage of students in each category.

3.0 MAIN CONTENTS

3.1 Data Statement

The general form of the DATA Statement is

line number DATA constant - list

Where DAT is the keyword. The constant – list consists of numeric or character constants. Items in the list must be separated by commas.

DATA is a non executable statement, unlike all other BASIC statements, except REM – encountered so far.

The DATA statement inform the BASIC system that the numbers specified in its constant – list are to be stored in memory until the user decides to process these numbers during program execution.

Example

```
1 DATA 1, 3.1
5 READ X, Y, Z, W$
10 PRINT X, Y, Z, W$
15 DATA 3, HI
20 END
```

RUN

```
1 3.1 3 HI
```

1	3.1	3	HI
↓			

Data Block

Memory

Example 2

```
1 DATA 1, 3.1, 3, HI
5 READ X
10 READ Y
15 READ Z, W$
20 PRINT X, Y, Z W$
25 END
```

RUN

```
1 3.1 3 HI
```

Example 3

```
1 DATA 1
5 DATA 3.1
10 READ X, Y, Z, W$
15 DATA 3, HI
20 PRINT X, Y, Z, W$
25 END
```

RUN

```
1 3.1 3 HI
```

2. Parentheses may be used to devote the order in which the expressions are to be evaluated

If (S = 1 AND M1 <> 0) OR (S = 3 AND M1 > 4) THEN PRINT S, M1

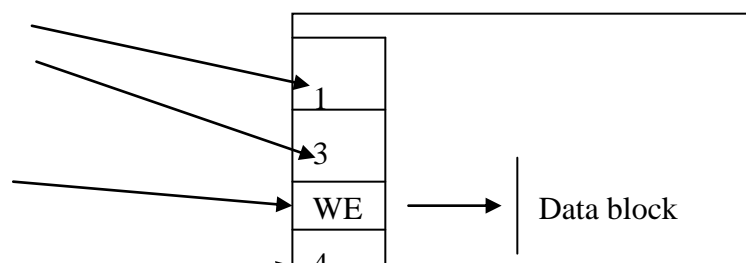
3.2 Read Statement

The general form of the READ statement is

Line number READ variable list

Where READ is a key word. The variable list consists of variables separated from one another by commas

The READ statement causes as many values to be fetched from the DATA block as there are variables in the READ list.



```
5      READ X, Y
10     DATA 1, 3
15     READ 2$
20     DATA WE. 4. 6
25     READ W
```

4.0 CONCLUSION

The assignment of values to variables is made in order in which variables appear in the READ list.

50 SUMMARY

Counting is an essential technique in programming

6.0 TUTOR-MARKED ASSIGNMENT

1. What will be the values assigned to each variable in the following program?

```
10    READQ, R
20    READS$, T$
30    DATA      25, -6, T, S
40    END
```

2. What output will be produced by the following program?

```
10    DATA      6, 9
20    DATA      -1, 0, 13
30    READ X
40    PRINT X
50    IF X <> 0 THEN 10
60    END
```

7.0 REFERENCE/FURTHER READING

Boillot and Horn BASIC 3rd Edition

MODULE 3 ACCUMULATION PROCESS STATEMENTS FOR/NEXT

Unit 1	The Accumulation Process and Statement FOR/NEXT
Unit 2	The Accumulation Process
Unit 3	Nested Loops
Unit 4	Application of Nested Loops in Statistics
Unit 5	Project

UNIT 1 THE ACCUMULATION PROCESS AND STATEMENT FOR/NEXT

CONTENTS

1.0	Introduction
2.0	Objectives
3.0	Main Contents
3.1	For/Next Statements
3.1.1	The General form of Next Statement
4.0	Conclusion
5.0	Summary
6.0	Tutor Marked Assignment
7.0	Further Reading

1.0 INTRODUCTION

The statement FOR/NEXT represent no new programming concept that have not been already been discussed. The purpose of these two statements is strictly one of convenience to the parameter.

2.0 OBJECTIVES

At the end of this unit, you would be able to use:

- FOR/NEXT Statements for loop control

3.0 MAIN CONTENTS

3.1 For/Next Statements

Any BASIC program can be written without FOR/NEXT statements because the two represent no new programming concepts. They are used primarily for loop control. The usual procedure for loop control is to initialize a counter to a certain value, then increment that counter by a constant and finally compare the counter to the terminal value for loop exiting. Using the FOR/NEXT statements the user specifies in the FOR statement the initial incremental, and terminal value of the counter (index) and identifies the range of the loop by making the NEXT statement the last statement of the procedure to be repeated. The general form of the FOR statement is

Line number FOR index = e, Toe, [STEP e,]

The FOR statement must always be used in conjunction with

1. The NEXT statement

WITHOUT FOR/NEXT	WITH FOR/NEXT
05 DATA , 60, 70,80	05 DATA, 60, 70, 80
06 DATA 20, 100	06 DAPO 20, 100
10 LET S = 0	10 LET S = 0
15 LET I = 1	20 FOR I = 1 TO 5
20 If I>5 then 45	25 READ G
25 Reading	30 LET S = S + G
30 LET S = STG	40 NEXT
35 LET I = I+1	45 LET A = S/5
40 GO TO 20	
45 LET A = S/5	
50 PRINT A	50 PRINT A
55 STOP	55 STOP
99 END	99 END

S is used as an accumulation to add to all grades. It is set to zero initially.

Process the next two statements 5 times (as I ranges from 1 to 5) is initially 1. As long as it does not exceed 5 read a grade and add to it to 5 (thereby forming a running sum of grades)s added to 1 automatically and the loop is repeated until I >5 at which time the average is computed and printed.

SELF-ASSESSMENT EXERCISE

- i. How would you modify the program to compute the average of 10 grades

- ii. Answer Change line 20 to 20 If 1>10 then 45 or 20 FOR 1=1 TO 10 Change line 45 to 45 LET A = 5/10

3.1.1 The General Form of NEXT Statement

Line number NEXT index

Where NEXT is the key word and index is a variable name that must be the same name as the variable specified for the index in the corresponding FOR statement. It is used to indicate the physical end of a loop initiated by a FOR statement.

```
10 FOR I = E1 TO E2 STEP E3
    ↓
90 NEXT I
```

To make your program more readable, it is a good idea to indent all statements between the FOR and NEXT statements.

4.0 CONCLUSION

Any BASIC program can be written without the FOR/NEXT statement. The purpose of these two statements is one convenience to the programmer.

5.0 SUMMARY

For/next are used primarily for loop control.

6.0 TUTOR MARKED ASSIGNMENT

What is the expected output of each of the following program? Tabulate the Program

```
10 DATA 20, 40, 70, 90
15 LET A = 0
20 FOR I = 1 TO 4
30 READ X
40 LET A = A + X
50 NEXT I
60 PRINT A
70 END
```

7.0 REFERENCE/FURTHER READING

M. Boillot and L.W. Horn, BASIC, Third Edition, West publishing company New York.

UNIT 2 THE ACCUMULATION PROCESS

CONTENT

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 The accumulation Process
 - 3.2 Accumulation and FOR/NEXT
 - 3.3 The FOR/NEXT Revisited
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor Marked Assignment
- 7.0 Reference/Further Reading

1.0 INTRODUCTION

The main different between counting and accumulating is that, instead of repeat adding a constant (example 1) to a counter, a variable is added repeatedly to an accumulator.

2.0 OBJECTIVE

At the end of the unit you would be able to:

- perform an accumulation process

3.0 MAIN CONTENTS

3.1 The Accumulation Process

You now know how counting in BASIC is made possible by repeated execution of such statements as $1 = 1 + 1$, Where 1 is initial set a beginning value. Each line $1 = 1 + 1$ is executed. The value 1 is added is added to the counter 1 which then takes a successive value 1, 2, 3, 4 and so on, if it is initially set to zero.

START



$C = 0$



$C = C + 1$



START



$S = 0$



READ G



$S = S + G$



Counting

Accumulating

DATA 60, 70, 80,
20, 100

Each time, 1 is added to the previous count (C).

Each time a new Grade (G)

is added to the current sum (S)

We consider an example on accumulation process and statements for/next.

10 LET S = 0 Initialize sum to 0.8 um is as an accumulator to accumulate all grades

20 FOR 1 =1 TO 5 Process statements 25 to 30 five times

25 READ G Read a grade (G)

30 LET S = S+G The new sum is equal to the old sum plus the grade just read. The first time around the loop sum plus grade = 0 + 60 hence new sum = 60. The second time around, sum plus grade = 60 + 70 = 130. Each time through the loop, we are adding the grade just read to form a running or partial sum of grades. When five grades have been read, the sum of five grades will have been computed.

40 NEXT 1 Go back to statement 25, until the loop has been processed five times

45 LET A = S/5 Now that all the grades have been read, that is, the loop has gone through the full cycle.

50 PRINT A We can compute the average and print it out

55 DATA 20, 100

90 END

The above shows accumulating process

SELF-ASSESSMENT EXERCISE

What is the expected output of each of the following program?

```
10 DATA 1, 5, 17, -2
15 LET P=1
20 FOR I =1 TO 4
30 READ X
40 LET P = P * X
50 NEXT I
60 PRINT P
70 END.
```

Answer

Expected output -170

3.2 The For/Next Revisited

The FOR/NEXT statement can be quite convenient when it is desired to execute one or more statements at a specific number of times.

```
5 FOR Index J = Initial value 1 to terminal 303 STEP 1 increment for index.
10 READ X
15 PRINT X
20 NEXT J
25 LET K = 5
```

2.4 Here you learn the difference between counting and accumulating

4.0 CONCLUSION

5.0 SUMMARY

FOR/NEXT is very useful when you wish to execute one or more statements specific number of times.

6.0 TUTOR-MARKED ASSIGNMENT

Examine the following program, how many times will the loop be repeated?

```
5 FOR I = -2 TO 2.5 STEP .5
```

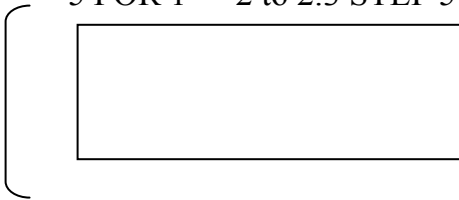
37 EXT 1

7.0 REFERENCE/FURTHER READING

M. Boillot and L. W. Horn, BASIC, Third Edition West publishing Company New York.

TMA
Solution

5 FOR 1 = -2 to 2.3 STEP 5



37 NEXT 1

The values assumed by the index 1 are -2, -1.5, -1, -0.5, 0, 0.5, 1.0, 1.5, 2.3

It will be executed 9 times.

UNIT 3 NESTED LOOPS

CONTENT

- 10 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 Nested loops
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor Marked Assignment
- 7.0 Reference/Further Reading

1.0 INTRODUCTION

2.0 OBJECTIVE

2.0 MAIN CONTENTS

4.0 CONCLUSION

Repeating a loop for a certain number of times is an example of a loop within a loop.

5.0 SUMMARY

In nested loops, a complete loop is part of the body of another loop. In such cases, each pass in the outer loop causes the inner loop to run through its complete cycle.

Exercise (see below another sheet)

6.0 TUTOR MARKED ASSIGNMENT

A DATA statement contains an unknown number of grades (0 – 100) with a maximum of 100 entries in the DATA statement including the trip record. Write a program to determine the percentage of passing grades. Passing grades are grades above 73

7.0 REFERENCE/FURTHER READING

M. Boillot and L. W. Horn, BASIC, Third Edition West publishing Company New York.

UNIT 4 APPLICATION OF NESTED LOOPS IN STATISTICS

CONTENT

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 Standard Deviation
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor marked Assignment
- 7.0 Reference/Further Reading

1.0 INTRODUCTION

Here you will solve the standard deviation problem using nested loops

2.0 OBJECTIVE

At the end of the loop you would be able to:

- write a program to compute the standard deviation of M grades

3.0 MAIN CONTENTS

3.1 Standard Deviation

The general formula to compute the standard for n grades x_1, \dots, x_n is

$$SD = \sqrt{\frac{n(x_1^2 + x_2^2 + \dots + x_n^2) - (x_1 + \dots + x_n)^2}{n(n-1)}}$$

SELF-ASSESSMENT EXERCISE 1

Write a program to input N grades and compute the grade average and the standard deviation – a negative grade indicates the end of grades

To compute the standard deviation it is necessary to

1. Accumulate the sum of grades $A = x_1 + x_2 + \dots + x_n$

2. Accumulate the sum of the square of each grade
 $X_1^2 + X_2^2 + \dots + X_n^2$

Based on the above we can use the program below

```

1. LET N = 0 N counts grades
5 LET Q = 0 set variable to accumulate the sum of the square of each grade to 0
10 LET S = 0 set variable to accumulate sum of grades to 0
20 INPUTE G Read one grade each lime through the loop
24 IF G < 0 GO TO 40
25 LET S = S+G Accumulate sum of grades
30 LET Q = Q+G *G Accumulate the sum of the square of each grade
34 LET N = N+1 Count each grade
35 GO TO 20 Go and accept next grade
37 REM COMPUTE THE STANDARD DEVIATION D
40 LET D = (N*Q-S*S)/(N*(N-1)) 1.5
45 PRINT "STANDARD DEVIATION" = "; D
50 PRINT "AVERAGE GRADE =" S/N
60 END

```

SELF-ASSESSMENT EXERCISE

1. To what depth or level can loops be rested?

4.0 CONCLUSION

Nested loops are useful in solving standard deviation

5.0 SUMMARY

A good report is one which is self- explanatory, self contained and organized in a way that allows the reader to capture the essence of the report as well as the detail.

6.0 TUTOR MARKED ASSIGNMENT

Using the previous problem, calculate

- i mean
- ii standard deviation of the following data

X Marks	21	42	53	60	73	81
Frequency	2	4	5	5	3	1

7.0

REFERENCE/FURTHER READING

Boillot and L.W. Horn, BASIC 3rd Edition

UNIT 5 PROJECT

CONTENT

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
- 3.1 Projects
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor Marked Assignments
- 7.0 Reference/Further Reading

1.0 INTRODUCTION

Here you will use your knowledge of programmes to solve the following problems

2.0 OBJECTIVES

At the end of this Unit, you would be able to:

- give a report of the assigned projects

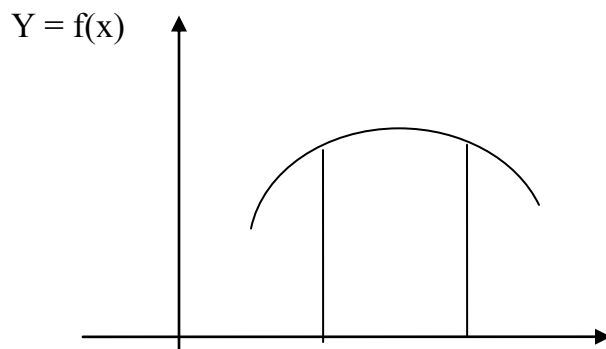
3.0 MAIN CONTENT

4.0 CONCLUSION

5.0 SUMMARY

6.0 TUTOR MARKED ASSIGNMENTS

Write a program to compute the area under a curve



$$X_1 \quad X_2$$

Your program should approximate the area under the curve $y = e^{-X^2/2}$

2 Write a program to compute the following sequence of sums;

$$\begin{aligned} S_1 &= 1 \\ S_2 &= 1 + 1/2 \\ S_3 &= 1 + 1/2 + 1/3 \\ S_4 &= 1 + 1/2 + 1/3 + 1/4 \end{aligned}$$

How many different sums would you have to compute before the sum exceed 3.9

For X between 1 and 2 for three values of
 $h = 0.1, h = 0.01, h = 0.001$

7.0 REFERENCE/FURTHER READING

M. Boillot and L.W. Horn, BASIC 3rd Edition

MODULE 4

Unit 1	One – dimensional Arrays
Unit 2	Use of Arrays
Unit 3	DIM Statement
Unit 4	Array Manipulation
Unit 5	End – of – File Conditions

UNIT 1 ONE-DIMENSIONAL ARRAYS

CONTENTS

1.0	Introduction
2.0	Objectives
3.0	Main Contents
3.1	One-Dimensional Arrays
4.0	Conclusion
5.0	Summary
6.0	Tutor Marked Assignment
7.0	Reference/Further Readings

1.0 INTRODUCTION

You will learn here how to calculate the average of five grades read on a DATA statement and the difference of each grade and the average. Up till now, it has been possible to compute an average of grades imply by reading one grade at a time into a variable using the statement READ G and accumulating the grades as they are read. This procedure cannot be used in this case, however, since each new grade destroys the previous grade stored in G, thereby making it impossible to compare each grade with average once the average has been computed.

2.0 OBJECTIVES

At the end of this unit, you would be able to print the difference between each grade and the average.

3.0 MAIN CONTENTS

3.1 One-Dimensional Array

Each grade must be preserved, and for that reason five distinct memory locations (variables) are needed as shown in the figure below.

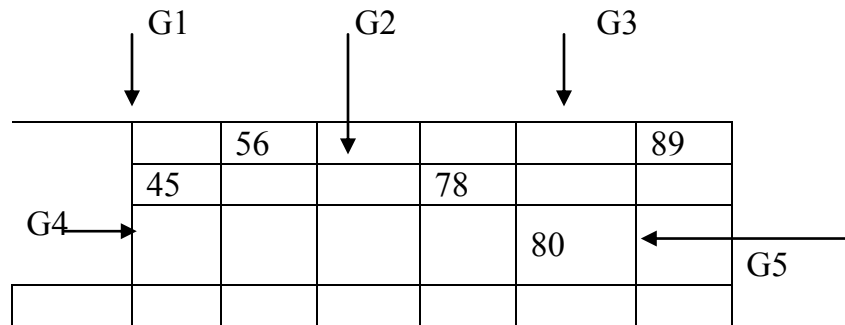


Figure I: Five Memory Locations

1.0 REM DEVIATION OF FIVE GRADES IWTHOUT AN
ARRAY

15 READG, G2, G3, G4, G5

20 LET A = (G1 + G2 + G3 + G4 + G5)/5

30 PRINT G1, G1 - 1

35 PRINT G3, G3 - A

40 PRINT G5, G5 - A

50 PRINT G5, G5 - A

60 DATA 56, 78, 89, 45, 80

99 END

Figure 2

One method for solving the problem is shown in Figure 2. We shall consider another method in the next unit.

4.0 CONCLUSION

To compare grades with the average of the grades, each grade must be preserved.

5.0 SUMMARY

Distinct memory locations (variables) are needed to preserve grades.

6.0 TUTOR-MARKED ASSIGNMENT

Compute the average of 50, 72, 74, 81, 87

Input / Output of Arrays

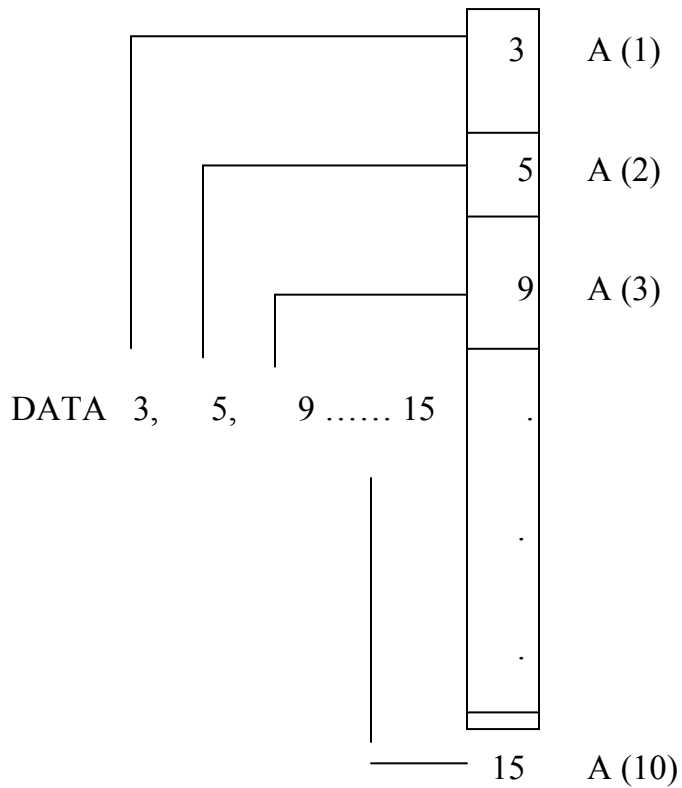
Arrays can be read and printed out by indexing the arrays with the index of a FOR / NEXT Loop.

Example I

To read 10 data from a DATA statement, the following code might be used.

```
10  DIM (10)
15  FOR I = 1 to 10
20  READ A(I)
25  READ A (I)
30  DATA 3, 5, 9, ..... 15
```

Array A



The first time through the loop I is 1, and A (I) is read from the DATA statement.

The second time through the loop, I is 2 and the next data item is read into A (10).

7.0 REFERENCE/FURTHER READING

Boillot, M. and Horn, L. W. BASIC, 3RD Edition, West Publishing Company, New York.

UNIT 2 USE OF ARRAYS

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 Use of Arrays to Store the Grades
 - 3.1.1 Definition
 - 3.1.2 An Array Storage
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor Marked Assignment
- 7.0 Reference/Further Readings

1.0 INTRODUCTION

Here you will learn another method that is not as cumbersome as the first.

2.0 OBJECTIVES

At the end of this unit, you would be able to:

- compute average of grades and compare each grade with the average.

3.0 MAIN CONTENTS

3.1.1 Use of Arrays to Store the Grades

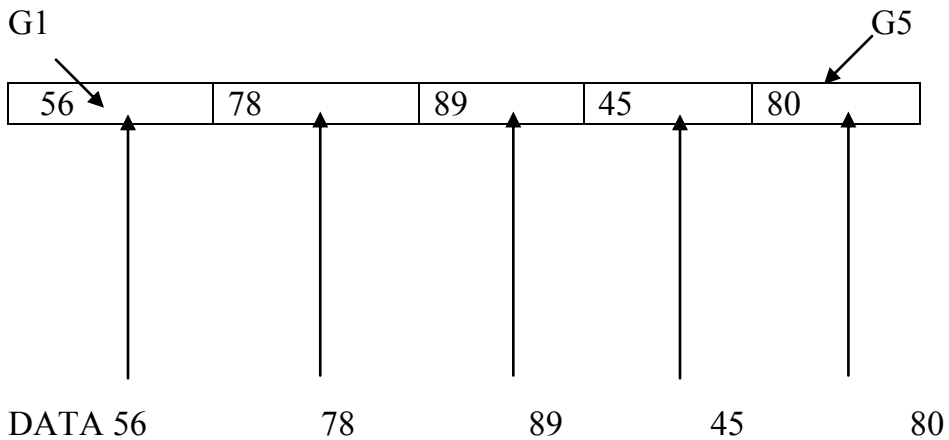
3.1.1 Definition

An array is a sequence of consecutive memory locations in which data elements are stored.

56	79	89	45	80
G(1)	G(2)	G(3)	G(4)	(G5)

Figure 1

3.1.2 An Array Storage



All grades (G(1) G(5)) are stored into the array one at a time starting with G (1) through the statement READ (G(1)) as I varies from 1 to 5

Figure 2

```
10  REM DEVIATION FOR FIVE GRADES USING ARRAYS
15  DIM G(5)           Reserve 5 memory locations
                        For array G.
20  LET S = 0          S will accumulate grades
25  FOR I = 1 TO 5    When I is 1 read 1st grade in
G(1)
30  READ G (I)        When I is 2 read 2nd grade in
G (2)
35  LET S = s + G (I) Add grades, one at a time, to
                        the accumulator
40  NEXT I
45  LET A = S/5       Compute the average
50  FOR I = 1 TO 5
```

55	PRINT G (I), G(I) – A	Print each grade and the difference between each grade and average
60	NEXT I	When I is 1, (I) and G(I) – A is printed
65	STOP	
70	DATA 56, 78, 89, 45, 80	When I is 5, G(5) and G (5) – A is printed.
99	END	

Figure 3

Through the use of indexing, accumulation logic can be used to calculate the sum of the grades. The statement $LET S = S + G(I)$ (Figure 3) accomplishes the accumulation process. The suit line through the loop $I = 1$ and $S = S + G(I) = 0 + g(I) = 56$

The first grade in the array is added to the sum, which is initially zero. The second time through the loop $I = 2$ and $S = S + G(I) = 56 + G (2) = 56 + 78 = 134$. Finally, when $I = 5$, the fifth grade in the array is added to the sum of the four previous grades. Output can be handled in a loop by using a variable subscript on the array G, in much the same way as the input.

4.0 CONCLUSION

5.0 SUMMARY

6.0 TUTOR MARKED ASSIGNMENT

Modify Figure 3 to handle 10, 100 grades

7.0 REFERENCE/FURTHER READING

Boillot, M. and Horn, L. W. BASIC, 3RD Edition, West Publishing Company, New York.

UNIT 3 DIM STATEMENT

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 DIM Statement
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor Marked Assignment
- 7.0 Reference/Further Readings

1.0 INTRODUCTION

Here you will be introduced the DIM statement

2.0 OBJECTIVES

At the end of this unit you would be able to:

- use the DIM Statement.

3.0 MAIN CONTENTS

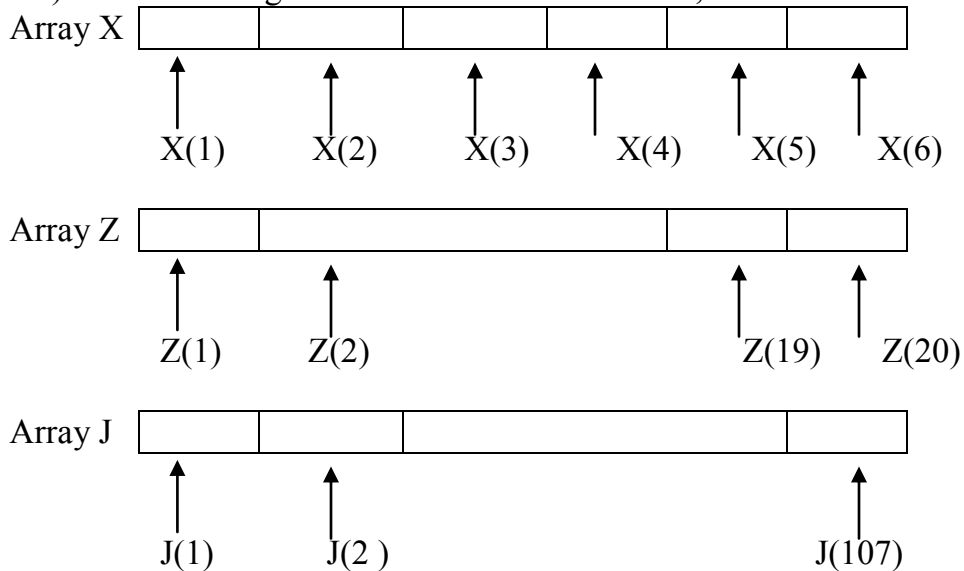
3.1 DIM Statement

The general form of the DIM statement is

Line number DIM variable ₁ (limit ₁ ,), variable ₂ / limit ₂ /

Where DIM is a BASIC keyword, variable₁, variable₂, Are names of the various arrays (any valid variable name), and limit₁, limit₂, Are unsigned integer constants representing the desired number of memory locations reserved for each array. This does not mean that the reserved locations must be used when processing the array. Array subscripts may vary from 1 to the limit declared in the DIM statement and may not exceed that limit. Any array used in a program must first be declared in a DIM statement. Any number of arrays may be declared in a DIM list.

For example DIM X (10), Z 1 (20), J (107) declares X and 21 and J as arrays. In this case the array X may contain up to 10 elements, the array Z1 up to 20 elements, and the array J up to 107) values one might visualize the elements of X, Z and J as follows



4.0 CONCLUSION

Here you learn how to use the DIM Statement.

5.0 SUMMARY

Subscripts are used with array names to locate specific elements of an array.

6.0 TUTOR MARKED ASSIGNMENT

What will be the content of each of element of the array Y after each of the following:

1. 10 DIM Y (10)
 20 FOR I = I TO 10
 30 LET Y (I) = I
 40 NEXT I
2. 10 DIM Y (10)
 20 FOR I = I TO 10
 30 LET Y (I) = 11 - I
 40 NEXT I

7.0 REFERENCE/FURTHER READING

Boillot, M. and Horn, L. W. BASIC, 3RD Edition, West Publishing

Company, New York.

UNIT 4 ARRAY MANIPULATION

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 DIM Statement
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor Marked Assignment
- 7.0 Reference/Further Readings

1.0 INTRODUCTION

Here you will be introduced to manipulation of arrays

2.0 OBJECTIVES

At the end of this unit, you would be able to:

- manipulate arrays.

3.0 MAIN CONTENTS

3.1 Arrays Manipulation

When working with arrays, it is often necessary to initialize arrays to certain values to create duplicate arrays, to inter-change elements within arrays, to merge two or more arrays into one to search or to accumulate array entries, to sort arrays etc. In this section, you will be introduced to certain commonly used array manipulation techniques.

Array Initialization and Duplication

The following code sets all elements of array A to zeros, sets each element of array B equal to the variable X, and the duplicates array D into array S:

```
10    DIM A(100), B(100), S(100), D(100)
```

```
12    LET X = 5
```

```

15  INPUT N      (Read a value for N. This value must
                  not exceed the size of the arrays declared in the DIM statement).

20  FOR I = 1 TO N

25  LET A (I) = 0      A (I), A(2), ....., A (N) are set to 0 one
                       at a time as I ranges from 1 to N.

30  LET B (I) = X      Similarly, B (1), B(2), ....., B (N) are
                       set to the value in X.
                       Finally S (I) = D (I)

40  NEXT I      S (2) = D (2), ..... S(N) = D (N)

```

Sometimes it might be necessary to set an array C equal to the sum of two other arrays A and B in such a way that $C(I) = A(I) + B(I)$, $C(1) = A(1) + B(1)$, $C(2) = A(2) + B(2)$, $C(100) = A(100) + B(100)$. The following code might be used:

```

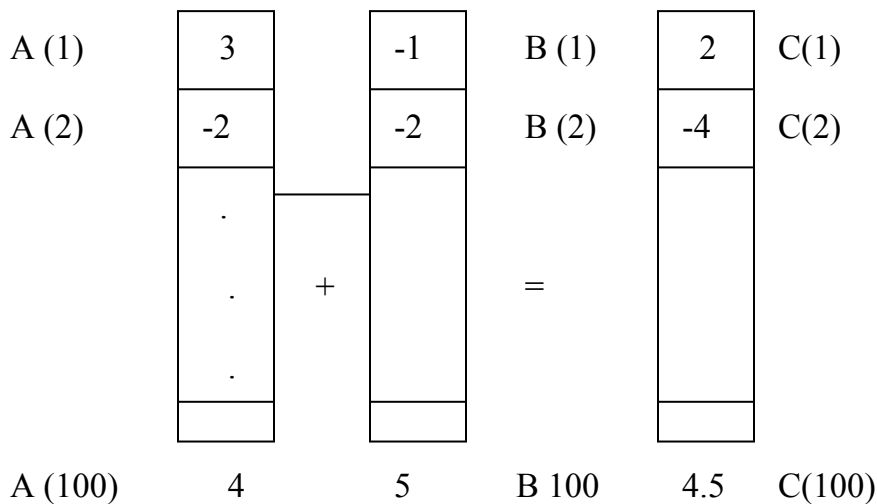
10  DIM A (100), B (100), C(100)

15  FOR J = 1 TO 100

20  LET C (J) = A (J) + B(J)

25  NEXT J

```



Suppose it is desired to initialize two arrays A and B as follows:

A (1) = B (10) = 1	10	DIM A (10), B (10)
A (2) = B (9) = 2	15	FOR I = 1 TO 10
A (3) = B (8) = 3	20	LET A (I) = I
.	25	LET K = 10 - I + 1
.	30	LET B (K) = I
.	35	NEXT I

A (10) = B (1) = 10

The variable K generates the numbers 10, 9, 81 as I ranges from 1 to 10. If I ranged from 1 to N, the formula $K = N - I + 1$ would generate the numbers N, N - 1, N - 2,, 3, 2, 1.

Reversing Arrays

Suppose A is an array of size N where N has been previously defined and it is desired to interchange A(1) with A(N), A (2) with A(N - 1), A(3) with A(N - 2), etc. The following code could be used:

10	DIM A (100)	Since each interchange step
15	INPUT N	involves a pair of array elements
20	FOR I = 1 TO N/2	(A ₁ , A _N), (A ₂ , A _{N-1}), etc., the
25	LET T = A(I)	interchange process needs be
30	LET K = N - I + 1	repeated only N/2 times. If N is
35	LET A (I) = A (K)	odd, the median element
40	LET A (K) = T	remains unchanged. K.
45	NEXT I	generates the number N, N-1,
		..., N/2 + 1. T is a temporary
		location needed to save A (1)
		before A (1) = A (N) is
		executed, otherwise, A (1)
		would be destroyed

If we used N instead of $N/@$ in statement 20, the array would “reverse” itself and end up as if nothing had changed.

Accumulation of Array Elements

To compute the product of the elements of the array A

=

10	20	30	40	50
----	----	----	----	----

The following code could be used:

```
10  LET S = 1           S initially set to 1 before the
                          loop is entered.
15  FOR K = 1 TO 5      The first time through the loop,
                          S = S*A (1) = 1*10 = 10
20  LET S = S* A (k)    The second time through the
                          loop, S = S* A (1)
25  NEXT K
```

To compute the sum of two arrays $S = A (1) + B (1) + A (2) + B (2) + \dots A (50) + B (50)$, we could use

```
10  LET S = 0
15  FOR K = 1 TO 50
20  LET S = S + A (K) + B (K)
25  NEXT K
```

Array Merge

Suppose A and B are two arrays of size 10 and we want the array C to contain the data $A_1, B_1, A_2, B_2, \dots, A_{10}, B_{10}$ arranged in that order. Any of the following codes could be used.

```
10  LET K = 1           10  LET K = 1
15  FOR I = 1 TO 10     15  FOR I = 1 TO 20 STEP 2
20  LET C (K) = A (I)   20  LET C (I) = A (K)
25  LET K = K + 1      25  LET C (I + 1) = B (K)
30  LET C (K) = B (I)  30  LET K = K + I
```

```

35   LET K = K + 1           35   NEXT I
40   NEXT I
10   FOR I = 1 TO 10      2*I - 1 generates the odd entries of C
15   LET C (2*I - 1) = A (I)  2*I generates the even entries of C
20   LET C (2*i) = b(i)
25   NEXT I

```

Array Search

Assume array A contains 10 grades, and we want to know the number of grades over 60. The following code could be used:

```

10   LET K = 0           K is used to count grades over 60.
15   FOR I = 1 TO 100
20   IF A (I) ≤ 60 THEN 30   If A (I) ≤ 60, skip the counting
                             of grades over 60, but stay in the
25   LET K = K + 1         loop by connecting to the NEXT
                             I statement.
30   NEXT I

```

4.0 CONCLUSION

Here you are introduced to array manipulation.

5.0 SUMMARY

Arrays could be manipulated in several ways.

6.0 TUTOR-MARKED ASSIGNMENT

What will be the content of each element of the array Y after the following:

```

10   DIM Y (10)
20   LET Y (1) = 1
30   LET Y (2) = 2
40   FOR I = 3 TO 10
50   LET Y (I) = Y (I - 1) + Y (I - 2)
60   NEXT I

```

7.0 REFERENCE/FURTHER READING

Boillot, M. and Horn, L. W. BASIC, 3RD Edition, West Publishing Company, New York.

UNIT 5 END-OF-FILE CONDITIONS

CONTENTS

- 1.0 Introduction
- 2.0 Objectives
- 3.0 Main Contents
 - 3.1 End-of-File Conditions
- 4.0 Conclusion
- 5.0 Summary
- 6.0 Tutor Marked Assignment
- 7.0 Reference/Further Readings

1.0 INTRODUCTION

Sometimes data may have to be read into arrays from an unknown number of records. Here you will learn how to handle such cases.

2.0 OBJECTIVES

At the end of this unit, you would be able to:

- read data into array from unknown courses.

3.0 MAIN CONTENTS

3.1 End-of-File Conditions

Sometimes data may have to be read into arrays from an unknown number of records. For example, someone may give you a large data file to read into an array A. since the DIM statement must specify an integer constant for the size of the array, the programmer must decide ahead of time what he thinks is the maximum number of locations he will need for the array.

Consider the following problems. Each record of data file contains a student's number and two test scores. Read into an array N, the student's number and store in array the

average of each student's score. Print the number of records processed and the average of all grades (not more than 100 students' records are expected). A negative value for the student's number terminates the data file. A program to solve the problem is as follows:

10	DIM S (100), N (100)	Array S is reserved for the scores,
15	REM	N for student numbers
20	LET SI = 0	SI is used to accumulate all grades.
25	LET I = 0	I counts all students' records, exclusive of the trip.
30	REM	
35	READ N1, T1, T2	Read student number and two scores.
40	IF N1 < 0 THEN 70	If last record, compute average etc.
42	LET I = I + 1	Increase student counter by 1.
45	LET N (I) = n1	State student number in array N.
50	LET S (I) = (T1+T2)/2	Compute each student average
55	LET S1 = S1 + S(I)	Accumulate sum of grades.
65	GO TO 35	Go back and read a new student's record.
70	LET A = S1/I	Compute average
75	PRINT "AVERAGE=" ; A	
80	PRINT "NO. SUTDENTS": I	
85	DATA III, 60, 50	} Data file, student number and two grades
90	DATA 222, 60, 40	
92	DATA -3, 0,0	
95	END	

4.0 CONCLUSION

Here you are introduced to End-of-File conditions and how to handle such cases.

5.0 SUMMARY

End-of-File Conditions could be handled as stated in the example above.

6.0 TUTOR MARKED ASSIGNMENT

Devise a code to search an array G to print the largest grade.

7.0 REFERENCE/FURTHER READING

Boillot, M. and Horn, L. W. BASIC, 3RD Edition, West Publishing Company, New York.

MODULE 1

UNIT 1

Exercise

How would the execution of the program be affected by replacing statement in location 7 by GO To 1?

Answer

After procession one record, the system will input another.

MODULE 2

UNIT 1

Exercise

What is a flow chart?

Answer: A flowchart is a pictorial representation of the logic used to solve a particular problem

UNIT 3

Exercise

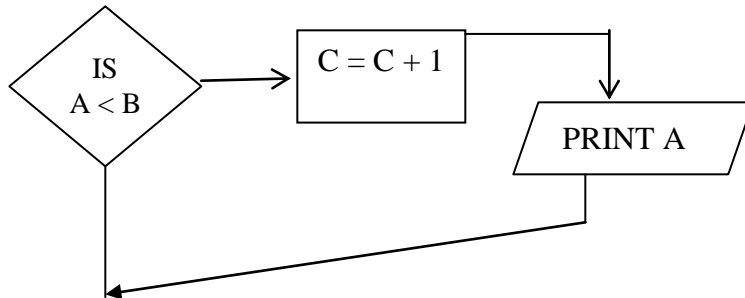
Predict the output of the suit six line of the following

```
10 I = 1
15 PRINT TAB (I); "*"
20 I = I + 1
25 GO TO 15
```

Answer

Unit 4 Exercise

Write the BASIC code for the following



Answer

```
10 if a >= B THEN 40
```

```
20 LET C -= C +
```

```
30 PRINT A
```

MODULE 3 UNITS 3 Exercise

What output is expected from the following?

```
10 FOR I = 2 TO 10 STEP 2  
20 FOR J = 10 TO 1 STEP -1  
30 FOR PRINT I, J  
40 NEXT J  
50 NEXT I
```

SOLUTION

I	J
2	10
2	9
.	.
.	.
.	.
2	1
4	10
4	9
.	.
.	.
4	1
.	.
.	.
.	.
10	10
10	9
.	.
.	.
.	.
10	1

OUTER LOOP: Since I varies from 1 to 3 the loop will be processed three times.

INNER LOOP: The inner loop will cause the PRINT statement to be processed 4 times. Since the outer loop is processed altogether 12 times

```
10  FOR I = 1 TO 3
    12  FOR J = 1 TO 4
    14  PRINT I, J
    16  NEXT J
20  NEXT I
```

The result produced by the above code

I	J	
1	1	} First time through the inner loop (outer loop index I = 4)
1	2	
1	3	
1	4	
2	1	} Second time through the inner loop (outer loop index I = 2)
2	2	
2	3	
2	4	
3	1	} Third time through the inner loop (outer loop index I = 3)
3	2	
3	3	
3	4	

Exercise

State whether the following are valid or invalid FOR/NEXT loops and give reasons if invalids.

```
1   FOR I = 1 TO 10
    FOR I = 1 TO 6
    PRINT I
    NEXT I
```

Answer – Invalid, nested loops may not use the same variable

```
2   FOR L = 8 TO 1 STEP -1
    FOR K = 1 TO 3
    FOR L = L + 1
    NEXT K
    NEXT L
```

Answer – Invalid, synthax statement FOR L = L + - 1 is incorrect (L)

UNIT 4

Answer: It depends on the manufacturer's BASIC. It is safe to assume that all BASIC systems will allow a depth of three nested loops.

2. What happens to the FOR/NEXT loop in the following cases?

- a. FOR I = 5 TO 5
- b. FOR J = 100 TO 1
- c. FOR K = 2, 10, - 1

In most cases the following actions will be taken:

- a. The loop is executed only once
 - b. The loop is not executed and the control is passed to the statement following the NEXT statement
 - c. Same as b since value K is less than test value and the increment is negative.
3. How can I determine whether an integer N is even or odd?

Answer: We use the integer function INT by computing the expression

$$2 * \text{INT}(N/2) - N$$

If N is even the result of the expression is always zero.

If N is odd, the result of the expression is always negative

MODULE 4

UNIT 1

Exercise

Which of the following code segments will compute the average (A) of 10 grades?

a) Let S = 0

 For I = 1 to 10

 READ G

 Let S = S + G

 NEXT I

 Let A = S/I

 PRINT A

b) LET S = 0

 For I = 1 to 10

 READ G

 LET S = S + G

 NEXT I

 PRINT A

Answer (A) and (B)

Unit 2

Exercise

Modify statements 50 -60 of the programme Figure 3 to print the grades in reverse order, i.e. G(5), G(4) G(1)

Solution

Charge the following statements:

15	DIM G (10)	15	DIM G(100)
25	FOR I = 1 TO 10	25	FOR I = 1 TO 100
45	LET A = S/10	45	LET A = S/100
50	FOR I = 1 TO 10	50	FOR I = 1 TO 100

UNIT 3

Linear Arrays

Exercise

Discuss the validity of the following DIM statements:

DIM A (3, 2)

DIM A (N)

Answer

Not valid - Invalid limit should be an integer constant

Not valid - N is not a constant. N must be a positive Integer

Solution

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Y(1) Y(2) Y(3) Y(4) Y(5) Y(6) Y(7) Y(8) Y(9) Y(10)

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Y(1) Y(2) Y(3) Y(4) Y(5) Y(6) Y(7) Y(8) Y(9) Y(10)

Unit 4 Exercise

1. Write DIM statement (s) to create an array Q containing 10 elements and an array R containing 25 elements.
2. Suppose an array X has the following content

Array X	-2	2.3	0	3	-2	6	10
	X(1)	X(2)	X(3)	X(4)	X(5)	X(6)	X(7)

And suppose $l = 3$ and $j = 2$. Evaluate each of the following expressions:

- a. $X(3)$
- b. $X(1+4)$
- c. $X(1) + X(4)$
- d. $X(1)$
- e. $X(1-j)$
- f. $X(1) - X(j)$
- g. $X(X(4))$

Answers

1. 10 DIM Q (10), R (25) OR 10 DIM Q (10)
20 DIM R (25)
2. a. 0
b. -2
c. -3 + 3
d. 0

- e. -3
- f. $0 - 2.3 = -2.3$
- g. $X(3) = 0$

Solution

						3	1	4	5
Y (1)	Y(2)								Y (10)

**UNIT 5
Trip Record**

Exercise

Modify the code of the code above example to include a listing of each student's number and his / her initial test scores.

Answer

```

71  FOR J = 1 TO I
72  PRINT N(J), S(J)
73  NEXT J

```