KISHORI SINHA MAHILA COLLEGE
AURANGABAD BIHAR

QBASIC NOTES

(AVINASH RANJAN)

1.General Concept

Program and Programming

- Program – A collection of sequential instructions given to computer to carry out certain task.
- Programming – The work of writing a set of sequential instructions is called programming.
- Programmer – The person who writes a set of instructions to a computer.

1.General Concept

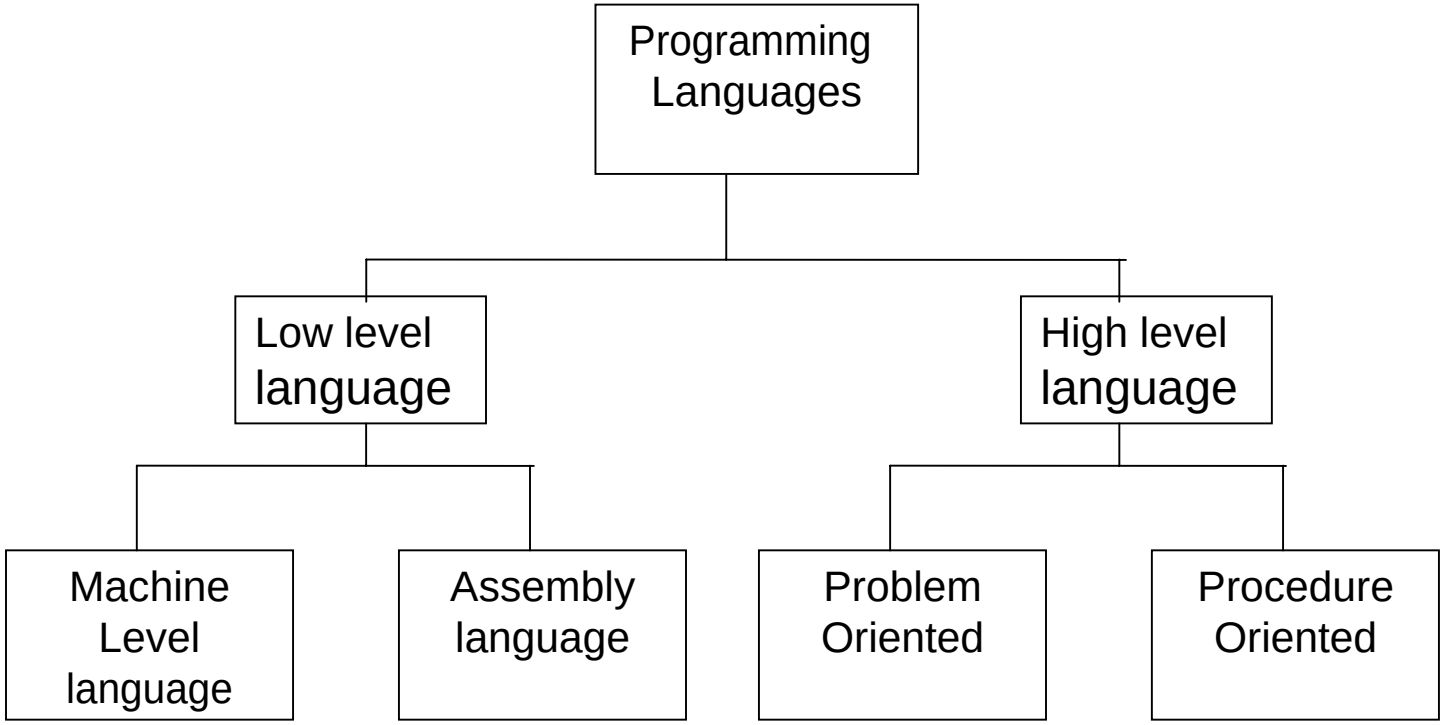
Programming language

- A language used for expressing a set of computer instructions.
- A language medium between the computer and the user in order to understand each other.
- Responsible for the human – computer system communication.
- Consists of necessary symbols, characters and grammar rules.

Programming language

- Many computer language e.g., C, C++, FORTRAN, PASCAL, COBOL, LISP, BASIC, ALGOL etc.
- BASIC – Beginners All Purpose Symbolic Instruction Code.
- GW-BASIC, BASICA, TURBO BASIC, QBASIC are different versions of the BASIC programming language.

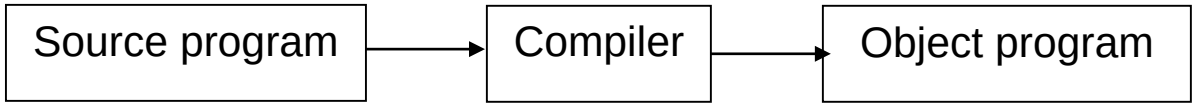
Categories of Programming Language



1.4 Translator (Compilers and Interpreters)

- Computer understands Machine Language only
- Translator - Program that translates a code from one programming language to another (usually machine code).
- Compiler – Program that translates High Level Language into machine code.

1.4 Translator (Compilers and Interpreters)



- Source program – the program written in a high-level-language
- Object program – the machine code equivalent translated by the compiler.

- In a compiler – translation and execution are two separate distinct phases.
- Interpreter – A program execution environment that accepts high-level language statements and performs desired operations immediately without compilation.

Text Editor, Linker and Debugger

- Text Editor – In this programmer types the necessary set of instructions to write a program, which becomes the source code.
- Linker – Converts object program or Object Code to Executable Program, which can directly execute in any computer.
- Debugger – Helps in finding the bugs in a program.

1.5 Program Development Cycle

- Any program to be developed to solve certain problem should follow the development cycle:-
  - Defining the problem
  - Analyzing the problem
  - Designing a solution (Pseudo code, Algorithm, Flowchart)
  - Coding the solution
  - Testing and debugging the program
  - Documenting the program

1.5 Program Development Cycle

- Defining the problem
  - Clearly define the problem for which a program is to be develop.
  - Explain in statements that are clearly understood.
  - E.g., Write a program which
    a) Requests the user to enter a temperature in degree centigrade.
    b) Calculates the corresponding temperature in degrees Fahrenheit.
    c) Prints the given temperature and the converted value.

1.5 Program Development Cycle

- Analyzing the problem
  - Clearly analyze the problem to make sure that we have a clear understanding of the problem.
  - Should be clear about general requirement such as the main inputs to the program, necessary processing and the main outputs from the program.
  - E.g., In our example, the input is the Centigrade temperature C entered using a keyboard, the necessary processing is to convert C by using the formula F = 32 + (9 C / 5).
  - The output will be displayed on the monitor.

1.5 Program Development Cycle

- Designing a solution (Pseudo code, Algorithm, Flowchart)
  - More than one solution - Different ways of solving a problem.
  - Choose the best and efficient one.
  - Pseudo code, Algorithm, Flowchart to produce a solution to a given problem.
  - More than one Algorithm to solve the problem with its own advantages and disadvantages.
  - Programmer should decide the best and efficient algorithm.

Designing a solution (Pseudo code, Algorithm, Flowchart)

- E.g., Searching a word in a dictionary.
  - Search from the beginning ?
  - Search from the end ?
  - Best solution – we use in practice based on the fact that the words in a dictionary are in an alphabetical order.
  - So the first and second solutions are not the best solutions.

Designing a solution (Pseudo code, Algorithm, Flowchart)

- For the temperature conversion one possible algorithm will be:
  1. Ask the user for the Centigrade temperature.
  2. Store the value in a variable C
  3. Calculate the corresponding Fahrenheit temperature
  4. Store it in variable F.
  5. Print the values of variable C and variable F, appropriately labeled.
  6. Stop

1.5 Program Development Cycle

- Coding the solution
  - The Algorithm or Flowchart is then converted into the instruction code understood by the computer to execute them to produce the desired output.
  - Choice of language e.g., BASIC, C, C++, FORTRAN, COBOL etc. for coding the solution depends upon the requirement of the total software solution.

Coding the solution

- The coding of the temperature conversion problem looks like:

```
PRINT “Enter temperature in Centigrade”;
INPUT C
LET F = 32 + (9 * C) / 5
PRINT “ Centigrade          Fahrenheit”
PRINT
PRINT TAB(5);C;TAB(18);F
END
```

1.5 Program Development Cycle

- Testing and debugging the program
  - Test the program for any Logical or Syntax Errors
  - Bug – any error in the program
  - Debugging – removing of errors from the program.
  - Test data – testing data to the program for which we know the answer, to find out whether it is doing the intended job.

Testing and debugging the program

- E.g. some test data for above temperature conversion problem are 0, 100 and 10 for which the answers 32, 212, and 50 should be displayed.
- If the program does not display the desired output, then it contains at least one bug. We must debug it to give the correct output. E.g. suppose in the above program the code line was

```
LET F = 22 + (9 * C) / 5
```

Testing and debugging the program

- Testing the program entering the value 10 degree Centigrade, we know that corresponding Fahrenheit temperature is 50; but the program will display:-

Centigrade	Fahrenheit
10	40

- As it does not agree with what we expect there is a Bug.
- The logical place to look for the bug is the calculation statement, since the conversion is wrong. We can see 22 was mistakenly typed instead of 32. After correction the program works fine.

Program Development Cycle

- Documenting the program
  - An explanation of how the program works ?
  - How to use it ?
  - It should be done at the same time along with all the activities.
  - Each activity produces its portion of documentation.

Documenting the program

- It should include:
  I. Statement of the problem.
  II. Algorithm or flowchart for solving the problem.
  III. Program listing.
  IV. Test data and the output produced by the program.
- Above document makes up a Technical documentation, which is useful to a programmer, perhaps to modify the program later.
- User documentation – document to help non-technical person to use the program without any need to know about the inner working of the program.

1.6 Algorithm and Flowcharts

- Algorithm – Sequential instructions needed to solve any problem.
- Algorithm is translated into Flowchart and program according to the requirement.
- Problem should be divided into smallest modules possible, while developing Algorithm.
- It is called the Modular approach, which makes the program more flexible for future modifications.
- Algorithm must be written in such a way that they can be easily converted into computer instructions.

1.6 Algorithm and Flowcharts

- The desired features of an algorithm are:
  - Each step of the algorithm should be simple.
  - It should be unambiguous in the sense that the logic is clear.
  - It should be effective i.e., it must lead to a unique solution of the problem.
  - It must end in a finite number of steps.
  - It should be as efficient as possible.

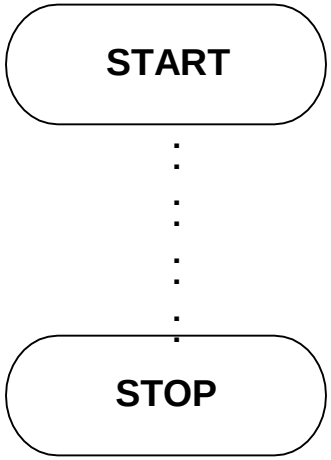
1.6 Algorithm and Flowcharts

- Flowchart – A collection of diagrammatically represented symbols connected with arrowheads, which represents a sequence of instructions for information processing.
- John von Newmann – said to have made the first Flowchart in 1945.

1.6 Algorithm and Flowcharts

- General symbols used in Flowchart
  - **Terminal or 'Start' & 'End' (Oval)**

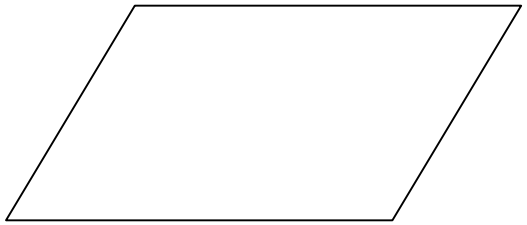
    This symbol is used in every flowchart at the start and for stopping the flow of instruction at the end.

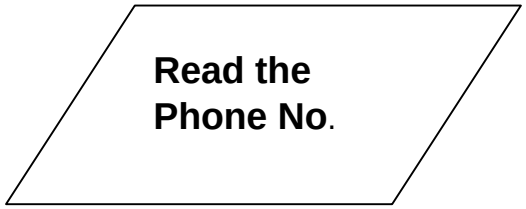


General symbols used in Flowchart

- **Input / Output (Parallelogram)**

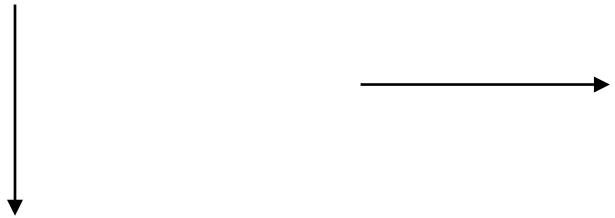
  This indicates input or output of necessary information. In this instructions of QBASIC like INPUT, READ, PRINT etc. are used.





General symbols used in Flowchart

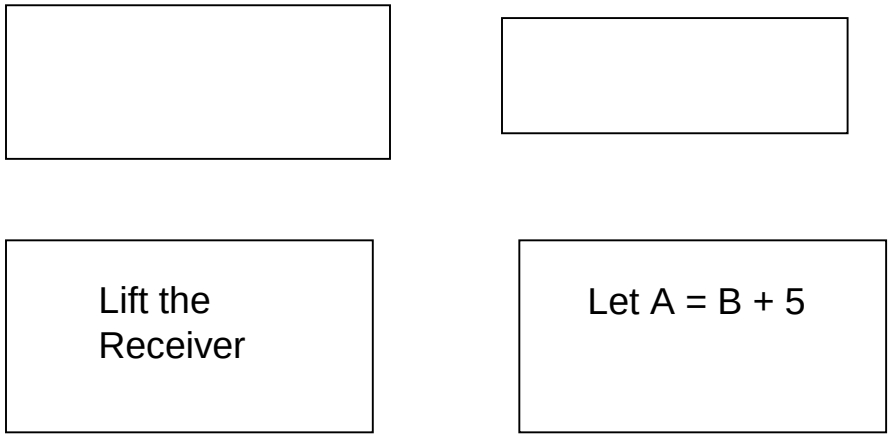
- **Flow Directions or Flow Lines**
  - This is used to indicate the flow of program or direction of the sequence of instruction. Generally the direction are top to bottom or left to right.



It is customary to write arrowhead to the point of line, which enters the flowchart symbol. As far as possible these lines should not cross each other.

General symbols used in Flowchart

- **Process Block (Rectangle)**
  - This symbol is used to indicate information processing. All the information processing work is done inside it. In this instructions of QBASIC like LET, FOR, GOTO etc. are used.

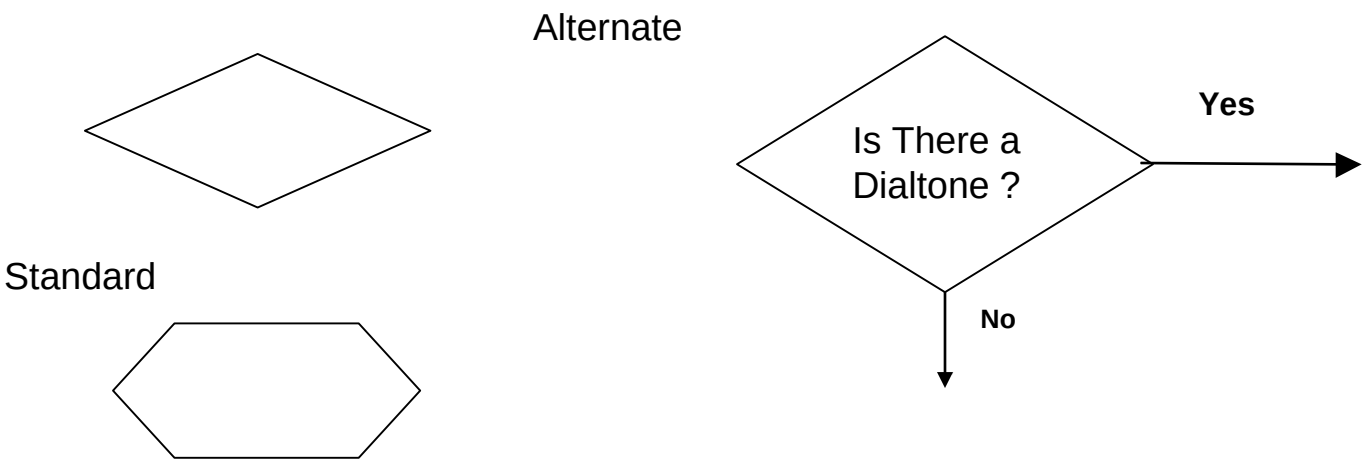


General symbols used in Flowchart

- **Connector (Small Circle)**
  - This symbol is used to interconnect long flowchart or between sub-routines. Usually letters written inside it, indicates that there exits a matching connector with the same letter to interconnect from that point

General symbols used in Flowchart

- **Decision (Diamond)**
  - This is used to indicate the decision stage from which the flow of program has to branch to one between the two choices. All the decisions processed here must produce results in "Yes" or "No". This result depends on the test that is performed inside this decision box. In this instructions of the QBASIC like IF… … THEN etc. are used.



General symbols used in Flowchart

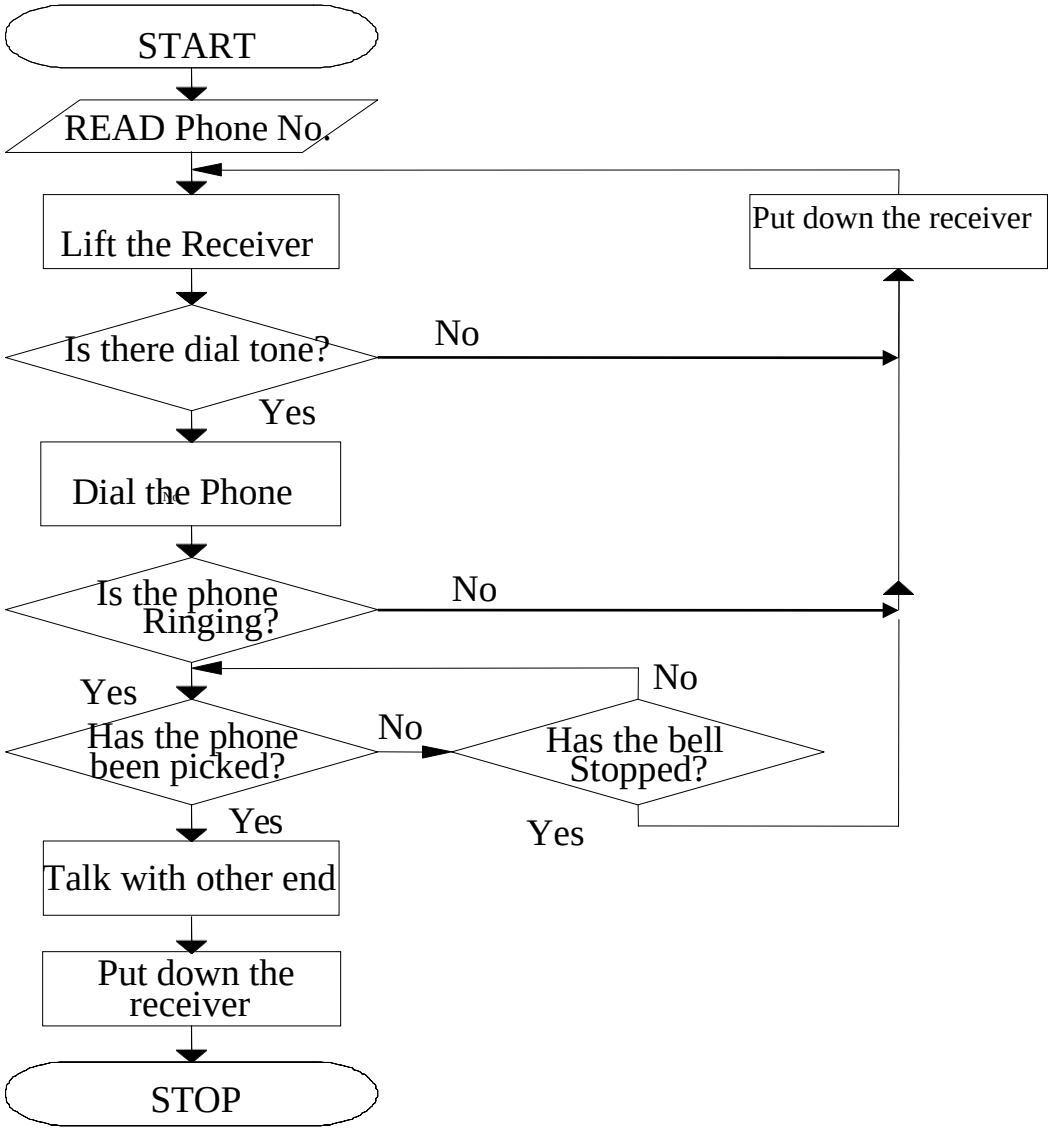
- Example: Make an algorithm and then a flowchart in detail as far as possible, for the tasks performed, while making a phone call.

One of the possible algorithms might look like this:

Step 1. Recall the phone number.
Step 2. Lift the Phone receiver
Step 3. Check for Dial tone?
If YES Go to Step 4
If No Put down the receiver
Go to Step 2
Step 4. Dial the phone number.
Step 5. Is the phone ringing?
If Yes Go to Step 6.
If No Put down the receiver
Go to Step 2
Step 6. Has the phone being picked up?
If Yes Talk with the other end.
Go to Step 8.
If No Go to Step 7.
Step 7. Has the bell stopped ringing?
If Yes Put down the receiver
Go to Step 2.
If No Go to Step 6
Step 8. Put down the receiver.
Stop

General symbols used in Flowchart

- **Flowchart based upon the above algorithm is as follows:**

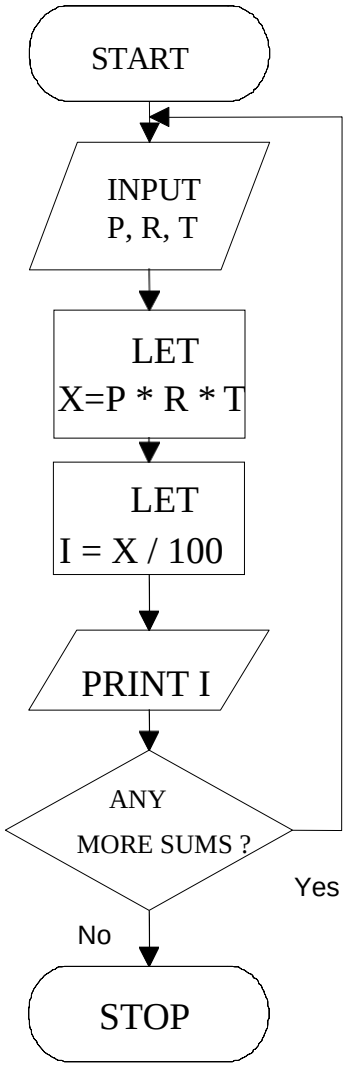


Manual Process into Programming

Flowchart

- Example 1. Develop an algorithm needed to calculate simple interest, and convert it into flowchart.

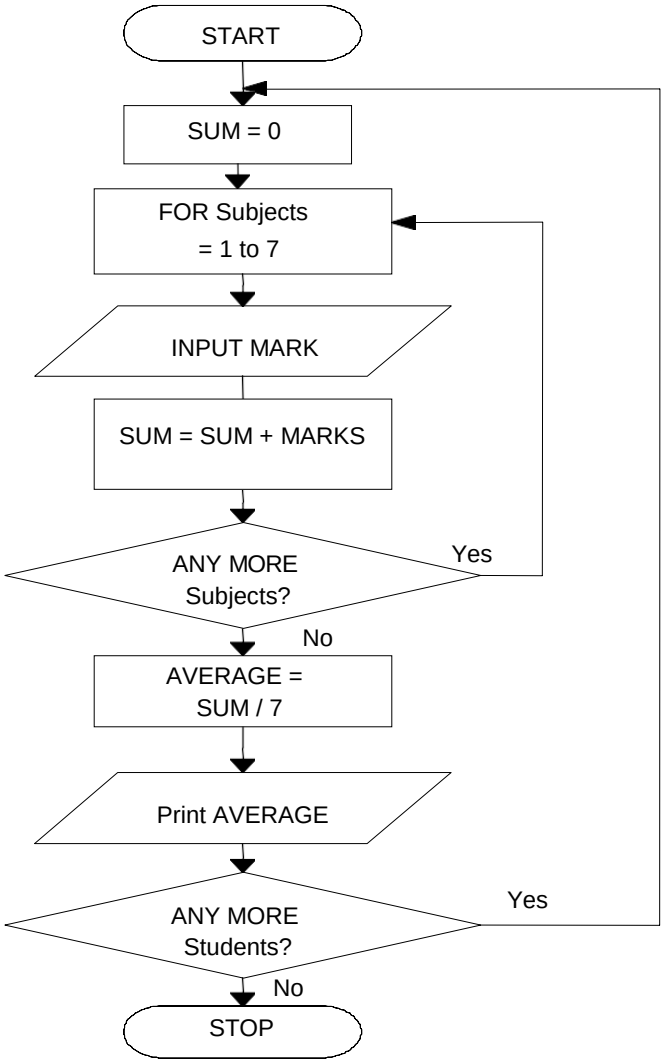
Step 1. Read Principle, Rate, Time
Step 2. Multiply Principle x Rate x Time
Step 3. Divide by 100
Step 4. Write the Answer
Step 5. Any more Calculations?
If Yes Goto Step 1
No Stop



Manual Process into Programming Flowchart

- Example 2. Develop an algorithm needed to calculate average marks from the marks obtained in the seven subjects, and convert it into flowchart.

Step 1. For 7 subjects
Step 2. Read marks
Step 3. Add marks to SUM
Step 4. Any more subjects?
Step 5. If Yes Goto Step 2
No Average = SUM / 7
Step 6. Write the Average
Step 7. Any more students?
If Yes Goto Step 2
No Stop



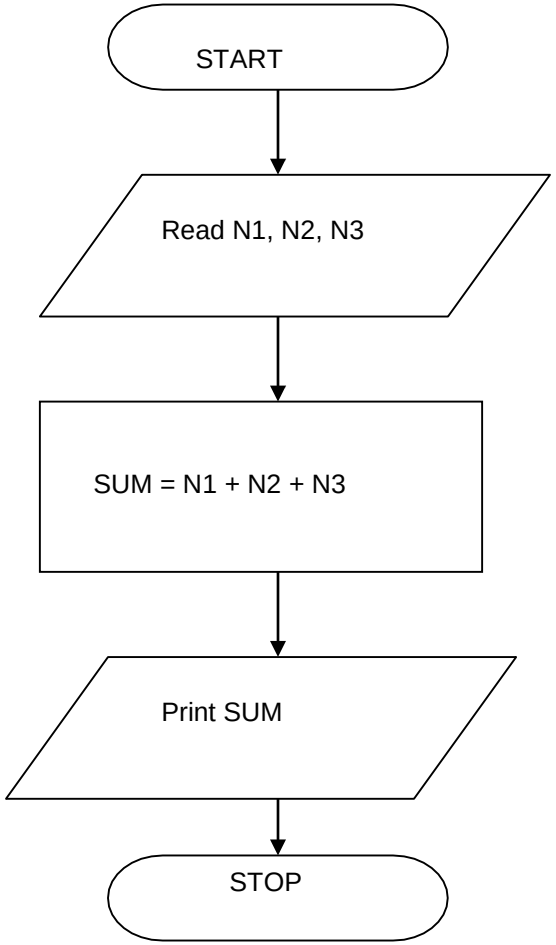
Manual Process into Programming Flowchart

1. Develop an Algorithm to add three numbers and convert it into Flowchart.
2. Develop an Algorithm to read two numbers and print the bigger number. Convert it into Flowchart.
3. Develop an Algorithm to find the biggest of the three given numbers and convert it into Flowchart.
4. Develop an Algorithm to find the sum of 10 given numbers and convert it into Flowchart.
5. Develop an Algorithm to find the average of 5 numbers and convert it into Flowchart.

Examples – Algorithm & Flowchart

1. Develop an Algorithm to add three numbers and convert it into Flowchart.

Read three numbers N1, N2 and N3
SUM = N1 + N2 + N3
Write the SUM
Stop



Examples – Algorithm & Flowchart

2. Develop an Algorithm to read two numbers and print the bigger number. Convert it into Flowchart.

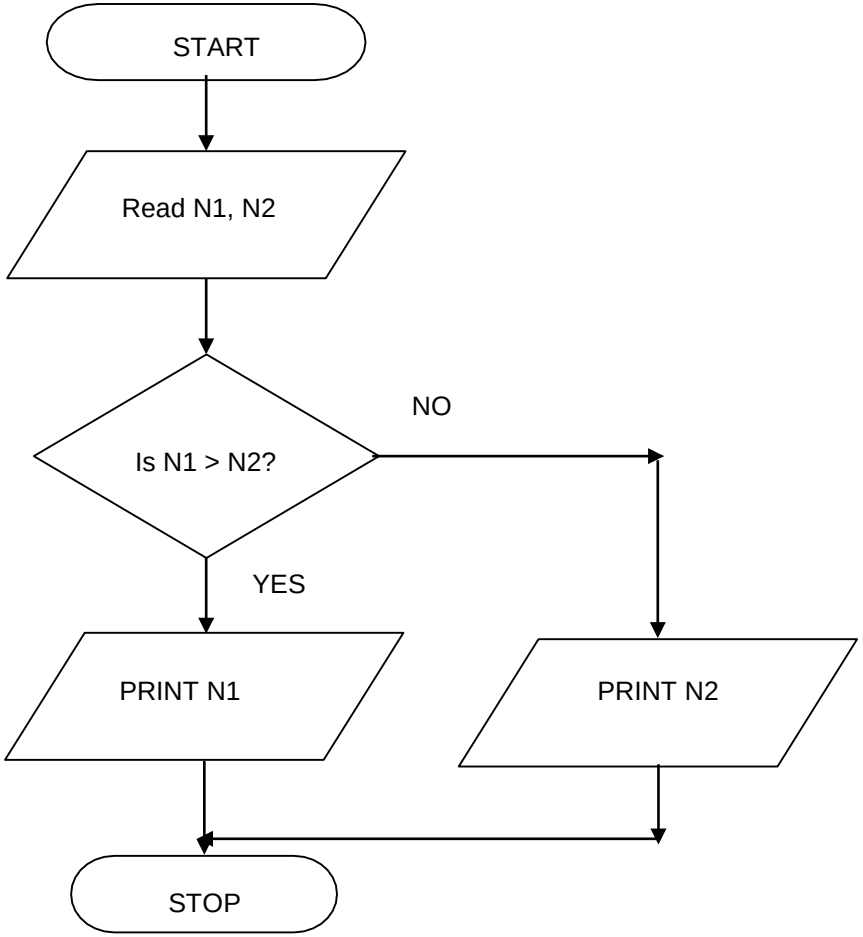
Read two numbers N1 and N2

Is N1 > N2

YES Print N1

NO Print N2

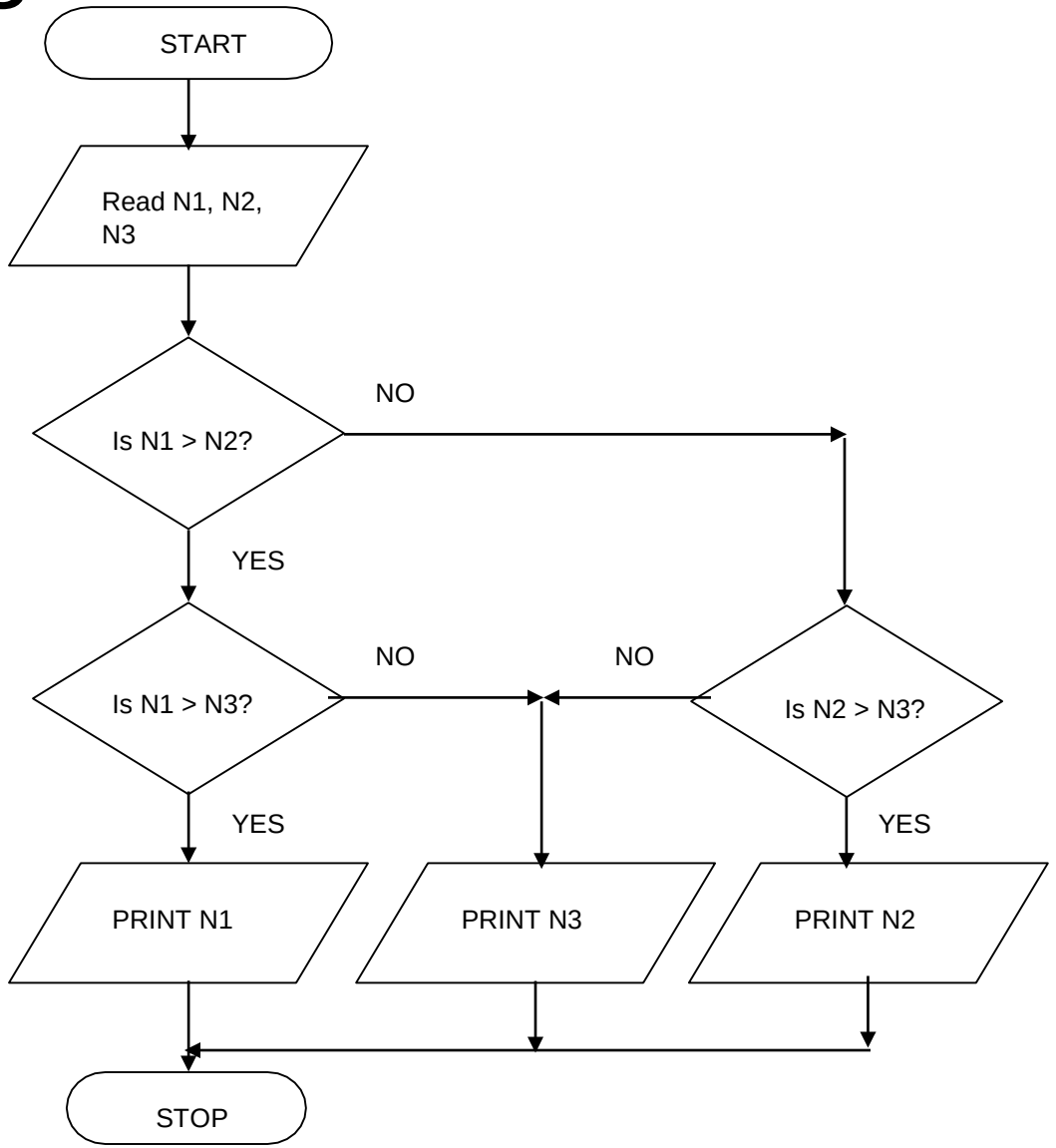
Stop



Examples – Algorithm & Flowchart

3. Develop an Algorithm to find the biggest of the three given numbers and convert it into Flowchart.

Read three numbers N1, N2 and N3
Is N1 > N2
YES Is N1 > N3
YES Print N1
NO Print N3
NO Is N2 > N3
YES Print N2
No Print N3
Stop

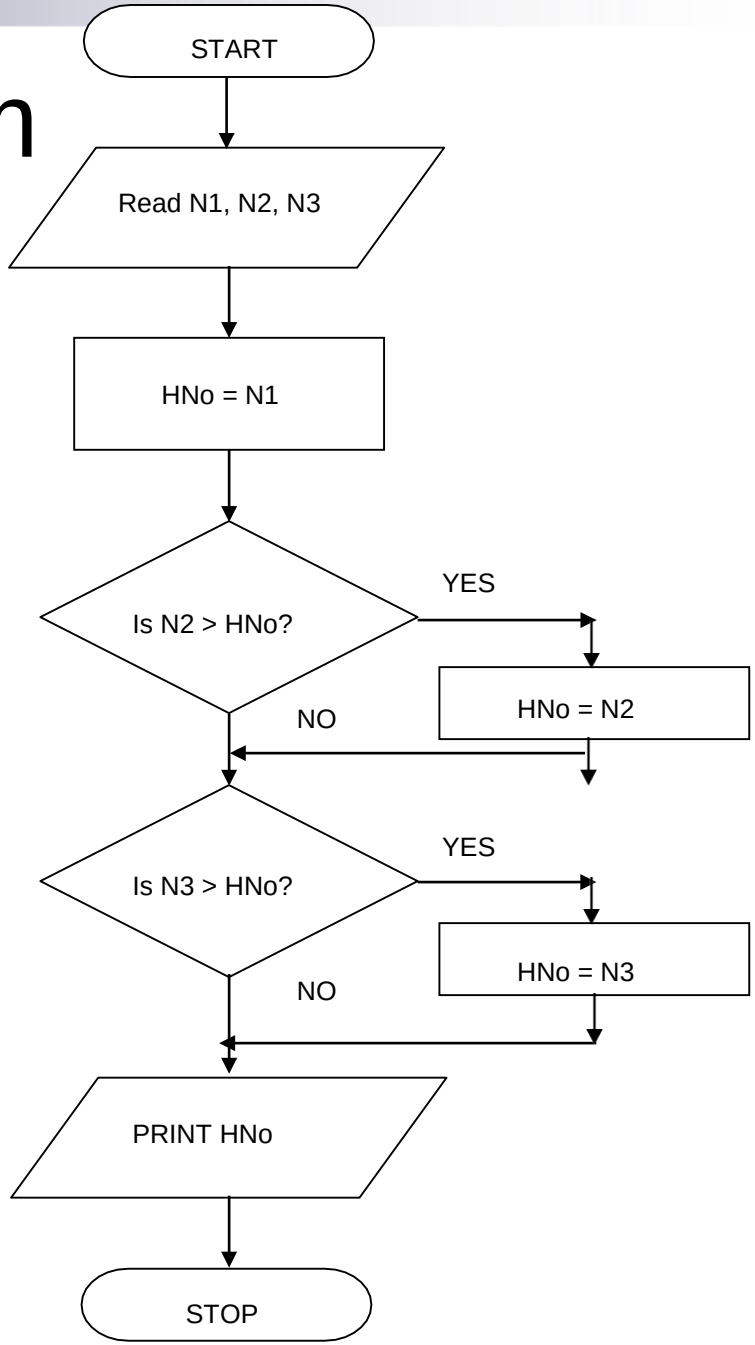


Examples – Algorithm & Flowchart

3. **Develop an Algorithm to find the biggest of the three given numbers and convert it into Flowchart. [Alternate solution]**

Read three numbers N1, N2 and N3
HNo = N1

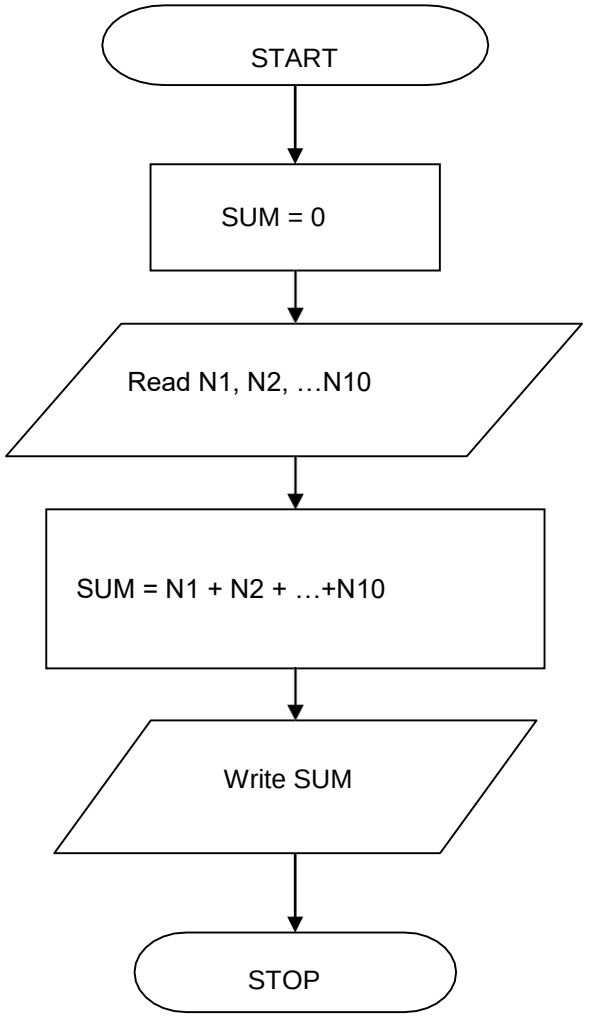
Is N2 > HNo
YES Store N2 to HNo
Is N3 > HNo
YES Store N3 to HNo
Print HNo
Stop



Examples – Algorithm & Flowchart

4. Develop an Algorithm to find the sum of 10 given numbers and convert it into Flowchart.

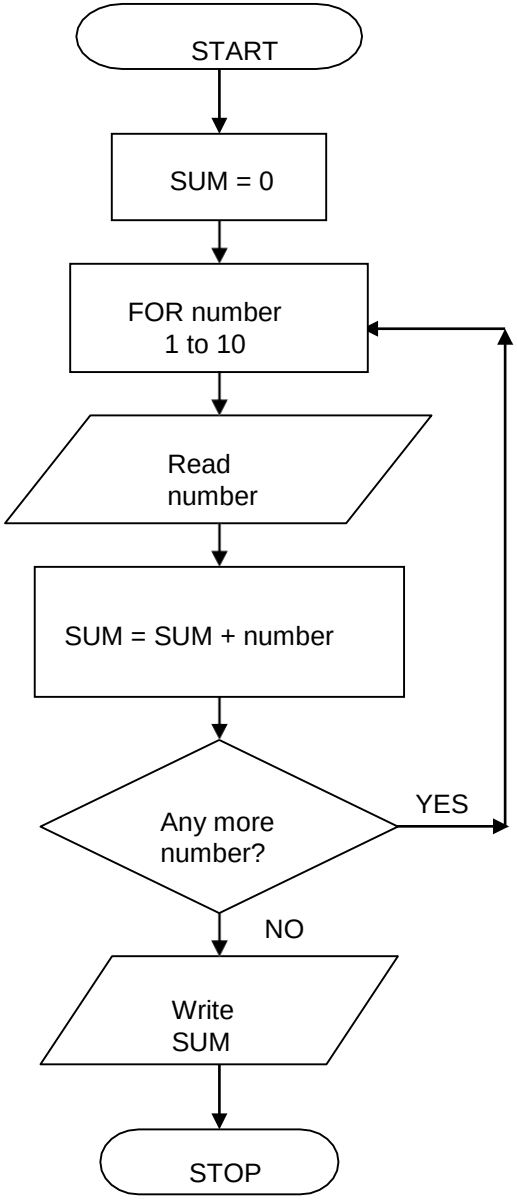
Initialize SUM = 0
Read the numbers N1, N2, …, N10
SUM = N1+N2+ … + N10
Write SUM
Stop



Examples – Algorithm & Flowchart

4. Develop an Algorithm to find the sum of 10 given numbers and convert it into Flowchart. [Alternate solution]

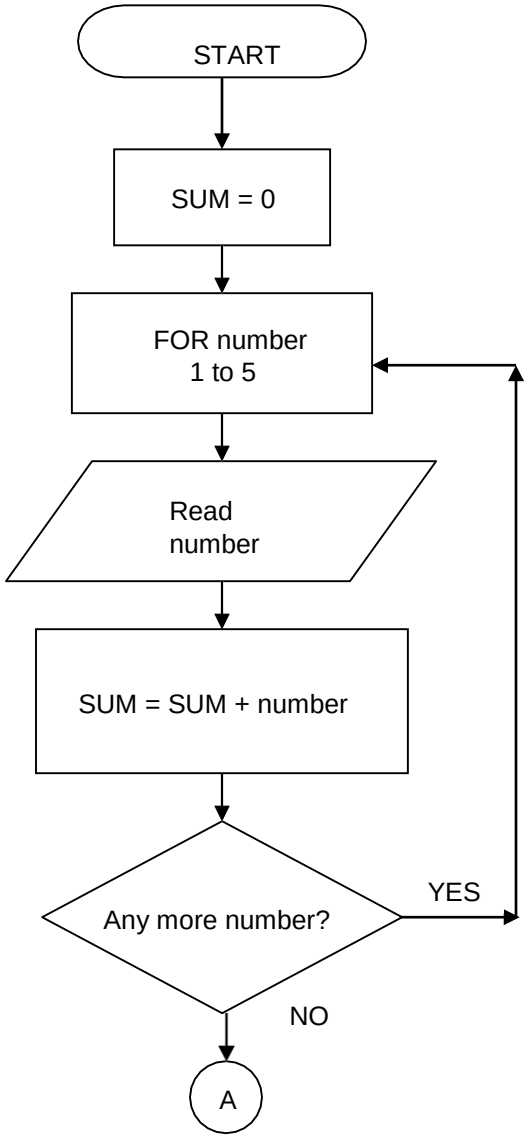
Step 1. Initialize SUM = 0
Step 2. For 10 numbers
Step 3. Read number
Step 4. Add the number to SUM
Step 5. Any more number?
Step 6. If YES Goto Step 2
No Write SUM
Stop

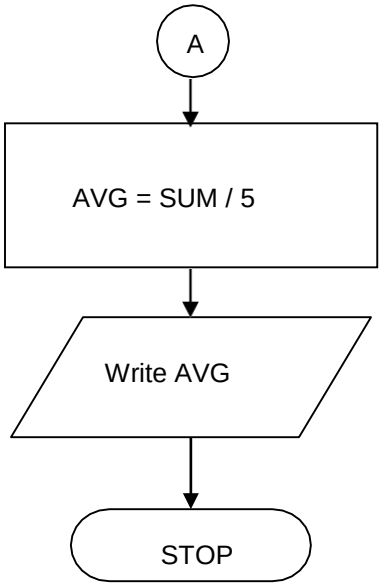


Examples – Algorithm & Flowchart

5. Develop an Algorithm to find the average of 5 numbers and convert it into Flowchart.

Step 1. Initialize SUM = 0
Step 2. For 5 numbers
Step 3. Read number
Step 4. Add the number to SUM
Step 5. Any more number?
Step 6. If YES Goto Step 2
No AVG = SUM / 5
Write AVG
Stop





Flowchart Exercise

- Using standard flowcharting symbols draw a flowchart to convert a temperature in degree Celsius into degree Fahrenheit.
- Draw a flowchart to find the greatest number among ten numbers and display the greatest number.
- Draw a flowchart to find the square and cube of the given number and display the result.
- Draw a flowchart to find the middle number among three numbers and display it.
- Write algorithm and flowchart to find the volume of a box with given Length, Breadth and Height and display the input and the results.

Algorithm and Flowcharts

- **Advantages of Flowchart:**
  - Representing algorithm by flowchart and then converting it to computer program is easier and accurate than writing the program directly.
  - Flowchart is an important aid in the development of programming algorithm.
  - Flowchart is easier to understand than the program
  - Flowcharts are independent of any programming languages. Hence, the algorithm given by a flowchart can be translated in to more than one programming language.

2. Introduction to QBASIC

- QBASIC Programs are the advanced new form of BASIC (Beginners All Purpose Symbolic Instruction Code)
- BASIC programming language was jointly developed by John G. Kemney and Tomas E. Kurtz in 1963-1964 in Dartmouth college, New Hampshire, USA .
- The instructions used in this language is very much similar to English. Hence it is used to teach Computer Programming to students in schools.

2. Introduction to QBASIC

- 2.1 Features of QBASIC
  - Very simple structured programming language.
  - Easy to follow logic, user friendly, high level programming language.
  - Divided into modules within a program.
  - First language for any beginning programmer.
  - Instruction are very much similar to English e.g., READ, LET, INPUT, GOTO, PRINT etc.
  - Easy to find syntax errors due to its own smart editor.
  - Easy to us since it has pull down menu.
  - Mouse also can be used in its latest versions.

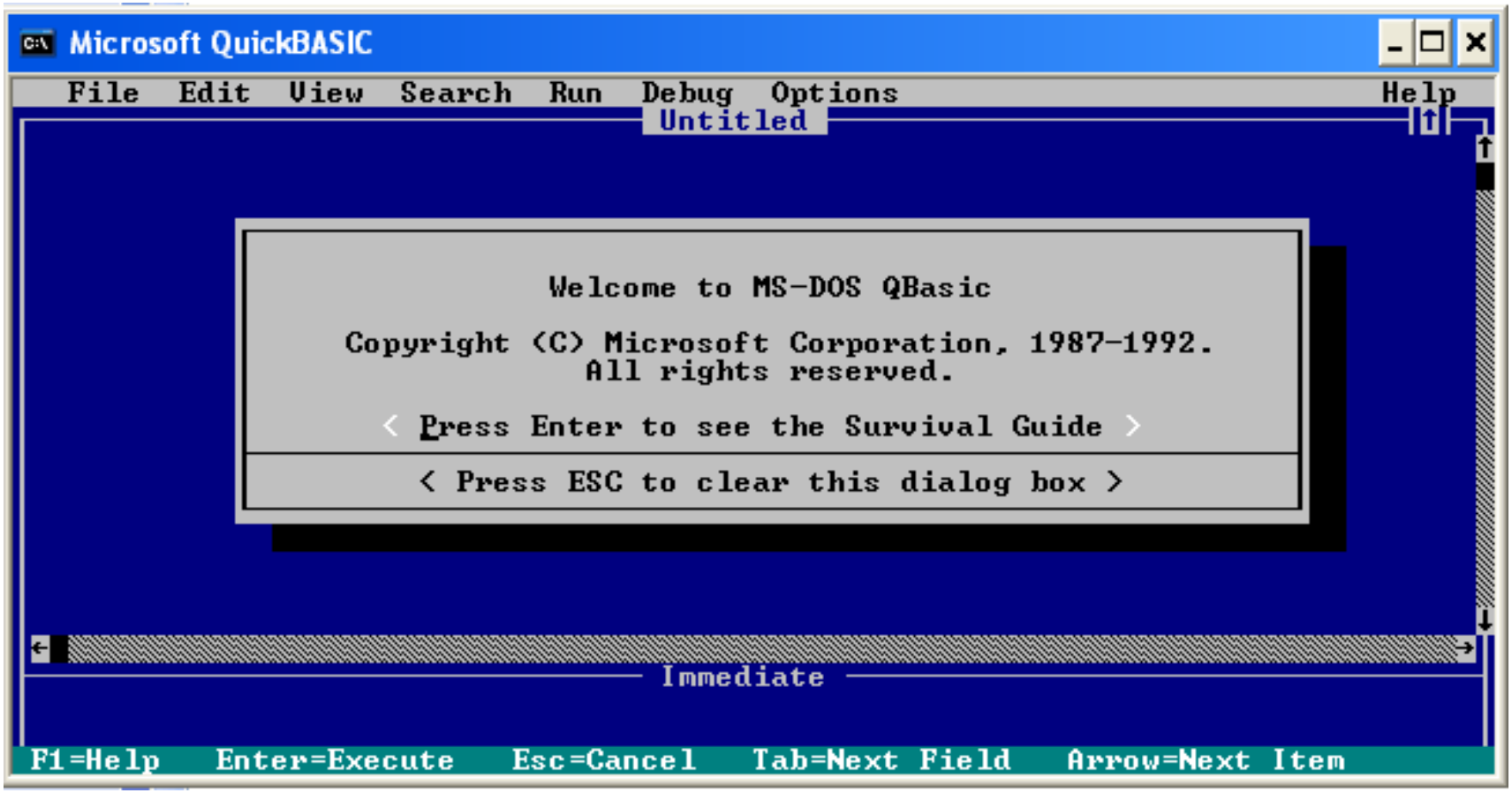
Features of QBASIC

- List of statements, executed one at a time from the beginning to the end.
- Statements are entered through an editor in the QBASIC environment after the QBASIC is loaded.
- Easy to learn and teach than other programming language.
- Easy to write Program, Run and Debug since it has more than one window like interfaces.
- Help is also available if one needs with a click.
- Program can be modified as per the necessity.

2.1 QBASIC Interface

- QBASIC Interpreter Program has to be loaded to develop any QBASIC Program.
- QBASIC.EXE must be present in the Hard disk or floppy disk in order to load it.
- Type QBASIC and then press Enter from the DOS prompt or double click the mouse on the shortcut icon to load it.
- The following screen is displayed when QBASIC is loaded. We can see a Menu Bar displayed at the top of the screen.

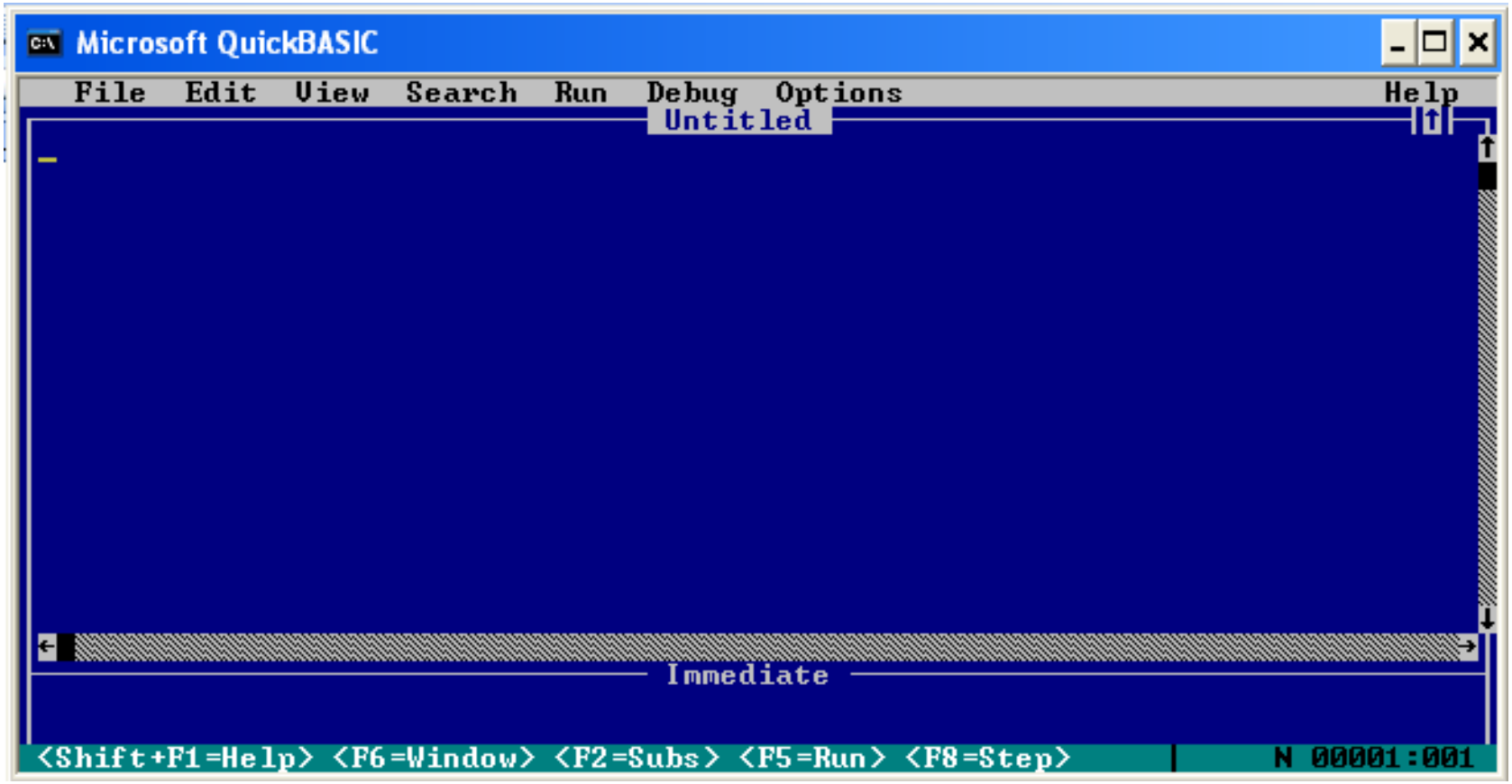
2.1 QBASIC Interface



2.1 QBASIC Interface

- A message box is displayed, which says <Press ESC to clear this dialog box>
- Press Esc from the keyboard then QBASIC Editor is required to input the necessary program, is displayed.

2.1 QBASIC Interface



QBASIC Editor

- Environment with the facilities for entering QBASIC Programs, save and execute them etc.
- Menu Bar is located at the top and below it at the middle is a portion showing file name.
- The editor is divided into two portions.
  - Upper portion is bigger, where program instructions are typed in and is called Program Window.
  - Lower portion is smaller called Immediate Window, where the instruction given is executed immediately. Used to test whether the instruction works or not.
  - Status Bar is located at bottom displays the purpose of the Function Keys and the Cursor Position.

2.3 Menu Commands

- Clicking the **FILE** gives the following pull down Menu.
  - New : Removes currently loaded file from memory.
  - Open : Loads new file into memory.
  - Save : Saves current file.
  - Save As : Saves current file with the specified file name.
  - Print : Prints specified text.
  - Exit : Exits the editor and returns to DOS.

2.3 Menu Commands

- Clicking the **EDIT** gives the following pull down Menu.
  - Cut (Shift+Del) : Deletes selected text and copies it to buffer.
  - Copy (Ctrl+Ins) : Copies selected text into buffer.
  - Paste (Shift+Ins) : Inserts buffer contents at the cursor position.
  - Clear (Del) : Deletes selected text without copying it to buffer.
  - New Sub... : Opens a window for a new sub-program.
  - New Function : Opens a window for a new FUNCTION procedure.

2.3 Menu Commands

- Clicking the **VIEW** gives the following pull down Menu.
  - SUBS... (F2) : Displays a loaded SUB or FUNCTION.
  - Split : Divides screen into two view windows.
  - Output Screen (F4) : Displays output windows.

2.3 Menu Commands

- Clicking the **SEARCH** gives the following pull down Menu.
  - Find : Finds specified text.
  - Repeat Last Find (F3): Finds next occurrence of text specified in previous search.
  - Change : Finds and changes specified text with the given text.

2.3 Menu Commands

- Clicking the **RUN** gives the following pull down Menu.
  - Start (Shift+F5) : Runs current program.
  - Restart : Clears variables in preparation for restarting the current programming in single stepping..
  - Continue (F5) : Continues execution after a break.

2.3 Menu Commands

- Clicking the **DEBUG** gives the following pull down Menu.
  - Step (F8) : Executes next program statement.
  - Procedure step (F10) : Executes next program statement tracing over procedure calls.
  - Trace on : Highlights statements current executed.
  - Toggle Breakpoint (F9) : Sets / clears breakpoint at the cursor position.
  - Clear All Breakpoint : Removes all breakpoints.
  - Set Next Statement : Makes the statement at the cursor position next statement to execute.

2.3 Menu Commands

- Clicking the **OPTIONS** gives the following pull down Menu.
  - Display : To change display attributes.
  - Help Path : Sets search path for help path.
  - Syntax checking : Turns editors syntax checking on or off.

2.3 Menu Commands

- Clicking the **HELP** gives the following pull down Menu.
  - Index : Displays help index.
  - Contents : Displays help table of contents.
  - Topic (F1) : Displays the information about the BASIC keyword where the cursor is on.
  - Using Help (Shift+F1) : Displays information about how to use online help.
  - About : Displays product version and copyright information.

3. Elements of QBASIC Programming Language

- Following elements are included in any programming language.
  1. Character Set
  2. Constants
  3. Variables
  4. Operators
  5. Expression
  6. Statements

3. Elements of QBASIC Programming Language

Character Set

In learning any programming language, we can begin by classifying the keys of the computer keyboard into 3 categories.

- **Alphabetic Characters:** The alphabetic characters are 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, 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.
- **Numeric Characters:** The numeric characters are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- **Special Characters:** The special characters are +, -, *, /, ^, ( ), ., ,, ;, =, <>, <, <=, >, >=, " ", $, :, ?.

3. Elements of QBASIC Programming Language

- 3.2 Constants

  - A symbol, whose value does not change during the execution of the program, is called a constant. Constants can be expressed as Literal e.g., 505, 522.12, "Nepal" etc. or as symbols e.g., PI = 3.142, City = "Kathmandu", FiscalYr = "2061/62". Here PI, City and FiscalYr are symbols representing constants. There are two types of constant.
    - Numeric
    - String

3.2 Constants

- Numeric
  - Constant whose value is numeric e.g., 899.50, -25.46, 97 etc. are Numeric Constant. Among these those with decimal is called Numeric Real Constant, e.g. 899.50, -25.46 etc. And those with integer only are called Numeric Integer Constant, e.g., 97, -568, 0 etc.

3.2 Constants

- String
  - A group of letters or numbers or both, which are enclosed by a pair of " " (quotation marks), is called String. E.g., "ABCD"; "Ramesh"; "10 Downing Street" etc. The variable used to store these strings is called String Variable. The name of string variable starts with character and should ends with $ sign. E.g., LET A$ = "Saturday"; LET CITY$ = "Kathmandu"; LET Adress$ = "10 Downing Street"; LET TEL1$ = "654345" etc. Constant whose value is a string e.g., 899.50, -25.46, 97 etc. are Numeric Constant

3. Elements of QBASIC Programming Language

- 3.3 Variables
  - Simply defined, variable is a name, which can contain a value.
  - Programming involves giving values to these names and presenting them in some form to the user.
  - A variable has a type, which is defined by the kind of value it holds.
  - If the variable holds a number, it may be of integer, floating decimal, long integer, or imaginary.
  - If the variable holds symbols or text, it may be a character variable or a string variable. These are terms you will become accustomed to as you continue programming.

3.3 Variables

- **Data types and memory allocation**
  - There are two data types represented by variables in QBasic: numeric and string.
  - A numeric data type can further be broken down into three categories: integer, long integer, single precision, or double precision and is capable of storing numeric values.
  - A string data type is capable of storing character values.
  - A storage location is allocated in the memory of a computer when a variable representing these data types are defined. This memory allocation serves as a storage box for storing the values.
  - In numeric variable, real number is stored in real variable denoted by a group of letters or letter with an integer. e.g., CASH1 = 899.5, when printed prints the value 899.5 without rounding.

3.3 Variables

- Here are some examples of values a variable might contain:

```
STRING      "hello, this is a string“
INTEGER     5
LONG        92883
SINGLE      39.2932
DOUBLE      983288.18
```

- The first is a string. Strings contain text.
- The last four are number types. But the computer does not know what kind of value you are trying to give a variable unless you tell it!
- There are two methods of telling the computer what kind of variable you are using:
- One-way is to just use the variable. This is called ***implicit declaration***.

3.3 Variables

- Another way is to declare the variable AS a type, which can be done by using the DIM statement.
- E.g. a variable called number, which would contain an integer (whole number, no digits after the decimal point) is defined as:

DIM number AS INTEGER

- Then you would use that variable as an integer.
- The word DIM actually originates from the word Dimension, it will be clear when we discuss arrays. This is called ***explicit declaration***.
- Put a symbol after the variable name, which is defined as representing that type.
- QBasic has a set of symbols, which represent each variable type:

$	String
%	Integer
&	Long
!	Single
#	Double

3.3 Variables

- Appending one of these symbols to the name of a variable when you use it in the program tells the computer that you are using it as that type.
- Declaration of the variable in this way is a popular practice.
- The most common error in QBasic is the infamous Type Mismatch, which means that you are trying to put a value into a variable of the wrong type.
- You might be trying to put the letters "hi there" into an integer variable.
- If you don't define the type of the variable, then QBasic assumes it is of the Single Precision type, which can often yield unexpected results.

3.3 Variables

- **Using type declaration characters (%, &, !, #, and $)**
  - A numeric variable name must begin with a letter, followed by letters, digits, and periods (no blanks or underscores).
  - By default, a variable will automatically be thought to be of single precision type, which is capable of accurately displaying numbers up to seven decimal places. You can specify a variable to be of single precision type by using the following syntax:
  - cost
    cost!

    Notice that an exclamation point (!) is attached to the end of the second variable name. This is not needed but is often used in programs so the programmer will know that data is of single precision type.

Using type declaration characters (%, &, !, #, and $)

- A double precision type is used when the programmer is working with large values that may need to display numbers up to fifteen decimal places. You can specify a variable to be of double precision type by using the following syntax:

  cost#

  Notice the hash symbol (#) attached to the end of the variable. This lets QBasic associate data as being of double precision type.

Using type declaration characters (%, &, !, #, and $)

- When strictly working with integer values and no need to use floating point values (single or double types).
- Integer number is stored in an integer variable represented by a group of letters or a letter with %(percentage) sign.
- specify the variable to be of type integer by using the following syntax:

cost%

Notice the percentage symbol (%) attached to the end of the variable. This lets QBasic associate data as being of integer type.

E.g. CASH% = 899.5, when the value of the variable is printed the value is rounded and printed as 900. If CASH% = 899.4, then only 899 is printed after rounding.

Using type declaration characters (%, &, !, #, and $)

- If working with extremely large integral values, you can specify the variable to be of type long by using the following syntax:

  cost&

  Notice the ampersand symbol (&) attached to the end of the variable. This lets QBasic associate data as being of long integer type.

Using type declaration characters (%, &, !, #, and $)

- If working with string values, you can specify the variable to be of type string by using following syntax:

  cost$

- Notice the dollar symbol ($) attached to the end of the variable. This lets QBasic associate data as data as being of string type.

Variables and its types in brief

Type of variable	Number / Characters	Memory	Example
String	0 to 32767 character	1 Byte of each character	A$
Integer	A whole number – 32767 to 32767	2 Bytes	A%
Long Integer	A whole number- 2 and more than 2 billion	4 Bytes	A&
Single precision	A number with up to 7 digits after decimal point	4 Bytes	A!
Double precision	A number with up to 15 digits after decimal point	8 Bytes	A#

Variables

- **Using DIM AS statement**
  - Declare variable or array in the program

    DIM <variable> As <Type>, <variable> As <Type>,

  - The Data Type can be Integer, Long integer, Single precision, Double precision or String.

  - Array is also a type of variable.
  - To dimension variables in this way, the DIM statement(s) must be placed at the very beginning of your program before the variable(s) are used in the program.
  - After you declare a string variable using the DIM statement, you are not allowed to attach a dollar sign ($) to the end of the variable.

Using DIM AS statement

- The DIM statement is used to declare variables as follows:

```
DIM state AS STRING
DIM number AS INTEGER
DIM profit AS DOUBLE
DIM gpa AS SINGLE
```

DIM example

```
CLS
DIM EXP1 AS STRING
DIM EXP2 AS STRING * 4
EXP1 = “PROGRAMMING IS FUN”
EXP2 = “ I LOVE PROGRAMMING”
PRINT EXP1
PRINT EXP2
END
```

The first string variable EXP1 is a variable length string variable (can store 0 to 32767 characters).

The second variable EXP2 is a fixed length string variable, in which STRING * 4 indicates that it can hold only maximum of 4 characters.

Using DIM AS statement

- Example:

```
CLS
X% = 25
Y% = 9
Sum% = X% + Y%
Print “Total is “; Sum%
END
```

In this X%, Y% and Sum% is used directly without declaring first. This is the Implicit declaration uses specific symbol at the end of the name to indicate the data type.

3.3 Variables

- Some valid an invalid numeric variable names are provided below:

  Valid Numeric Variable Names
  - firstTest
  - profit&
  - gpa!
  - total.Price
  - company8
  - COURSE2D

  Invalid Numeric Variable Names
  - 1stClass --> (must begin with letter)
  - total Price --> (no spaces)
  - name,First --> (no commas)

3.3 Variables

- Some valid and invalid string variable names are provided below:

Valid string variable names:

firstName$
heading$
course23Name$

Invalid string variable names:

firstName --> (must end with $)
company Name$ --> (no spaces)
name,First$ --> (no commas)

3.4 Operators

- Various symbols used to perform arithmetic and logical operations in programming language.
- E.g., 18 + 56, in this + is the operator and 18 and 56 are operand.
- Eg., A – B < C, in this – and < are the operators and A, B and C are operand.
- In any programming language value of arithmetic or logical operations are tested by using various operator symbols. Operators can be classified in 4 types.
  - **Arithmetic Operators**
  - **Relational Operators**
  - **Logical Operators and**
  - **String Operators**

3.4 Operators

- In algebra, we follow the sequence of **BODMAS** to solve problems. But in QBASIC, the order of operations or the operators precedence is different.

Operator	**QBASIC'S order of execution**
( )	1
^	2
* or /	3
+ or -	4
= or, <> or, < or, <= or, > or, >=	5
NOT	6
AND	7
OR	8

In case of operators with same order of execution, the left most operator of the expression is solved first.

3.4 Operators

- Arithmetic operators (+, -, *, /,MOD, ^)
  - used to solve for value of an expression. In QBASIC, the following arithmetic operators can be used.

Operator	Action	Example
+	Addition	A + B OR 10 + 5
-	Subtraction	C – D OR 30 – 23
*	Multiplication	P * Q or 12 * 5
/	Division	R / S OR 35 / 7
^	To the power	X^2 or 6^2

3.4 Operators

- In QBASIC programming language, we should think of the following differences in using Algebra.
- Multiplication is never implied.

  e.g., A = B * C instead of A = BC

  D = E * (F + G) instead of D = E(F + G)
- Operators cannot be written adjacent to each other.

  e.g., H = 5 * (-I) instead of H = 5 * - I

  J = K * (-L) instead of J = K (- L)
- Zero cannot divide.

  e.g., M = 0 N = P / M gives error.
- Instead of exponentiation i.e., 2^6 we write 2^6 or 2*2*2*2*2*2

Example of QBASIC operators

- **Example 1.** Solve given expression by finding stepwise value in the order of QBASIC operators execution.

A = 5^2 + 2 * 5 + (60 – 10) – 125 / 25

Finding step wise value in order of QBASIC operators execution

Step	Expression	Value
1	60 – 10	50
2	5^2	25
3	2 *5	10
4	125 / 25	5
5	25 + 10	35
6	35 + 50	85
7	85 – 5	80

Hence, the value of A is 80.

Example of QBASIC operators

- **Example 2.** Write the algebraic expression given below in QBASIC programming code.

  i. $x + y + z$ ii. LB iii. $\frac{x}{y}$ iv. $\frac{PQ}{S}$

  v. $3x^2+y$

- The above algebraic expression is written in QBASIC as follows:

No.	QBASIC
(i)	x + y + z
(ii)	L * B
(iii)	x / y
(iv)	P * Q / S
(v)	3 * x^2 + y

3.4 Operators

- **Relational operators (=, >, <, >=, <=, <>)**
- **used to compare two or more values. Using this we can compare value/s of variable/s with a constant. In QBASIC the following relational operators can be used.**

Operator	Relation	Example
=	Equal to	IF A = B THEN .....
<>	Not equal to	IF C <> 550 THEN …..
<	Less than	IF A < 250 THEN .....
<=	Less than or equal to	IF A <= 100 THEN .....
>	Greater than	IF A > 450 THEN .....
>=	Greater than or equal to	IF P >= 45 THEN .....

Operators

- When we compare strings (i.e., the characters inside the double quotes " ") using above relational operators, single character from same position of the strings are compared at one time.
- Actually, the ASCII code value of those characters is compared. If the ASCII value of both the compared strings is same then those two strings are equal.

Example:

- (i) If NAME1$ = "HARI“ and NAME2$ = "HARI" then NAME1$ = NAME2$ or "HARI" = "HARI“
- (ii) If FNAME$ = "RAM" and MNAME$ = "KRISHNA" then we can join them as FNAME$ + " " + MNAME$, whose value becomes "RAM KRISHNA".
- (iii) The value of "AB" > "AA" is true.

3.4 Operators

- Logical operators (AND, OR, NOT)
  - Used to examine two or more relations.
  - The outcome of this examination is given in True or False (Yes or No).
  - Logical operators most often used are AND, OR and NOT.

Operators - AND

- The outcome of AND becomes true, only when all the conditions joined by it are true.
- e.g., IF A = 10 AND B = 7 ... ..., here the outcome of this condition becomes true, only if the value of A is 10 and the value of B is 7.

3.4 Operators – AND

Outcome Table of AND		
Conditions		**Outcome**
I	**II**	**I AND II**
True	True	True
True	False	False
False	True	False
False	False	False

- If both the conditions I and II are true, the outcome becomes true.
- If any one condition is false then outcome becomes false.
- Also, if both the conditions are false, the outcome also becomes false.
- This type of table is also known as **Truth Table**.

3.4 Operators – OR

- The outcome of OR becomes true, when any one of the conditions joined by it is true.
- e.g., IF A = 12 OR B = 9 ... ..., here the outcome becomes true when either the value of A is 12 or the value of B is 9.

3.4 Operators – OR

Outcome Table of OR		
Conditions		**Outcome**
I	**II**	**I OR II**
True	**True**	**True**
True	**False**	**True**
False	**True**	**True**
False	**False**	**False**

- If both the conditions I and II are true or either is true, the outcome becomes true.
- Only when both the conditions are false the outcome also becomes false.

3.4 Operators – NOT

- NOT gives contrary outcome of the value of any condition.
- E.g., IF A NOT = 560 ... ... here, the outcome of the condition becomes true, if the value of A is not 560.
- If the value of A is 560 then the outcome becomes false.

3.4 Operators – NOT

Outcome Table of NOT	
Condition	**Outcome**
I	**NOT I**
True	**False**
False	**True**

- It is clearly seen that if the value of condition I is True the outcome becomes False.
- And if the value of condition I is False, the outcome becomes True. Hence, it is obvious that the outcome is quite reverse of the value of the condition.

3.5 Expression

- Arithmetic expression
  - Consists of numbers or variables or both made by using the operators ^, *, /, +, -.
  - The arithmetic expression always has some sort of numeric value.
  - Example:

```
CLS
Y = 2
X = Y^4
Z = X*Y
Z1 = X/Y
Z2 = X+Y
Z3 = X-Y
```

  - In above example Y^4, X*Y, X/Y, X+Y, and X-Y are arithmetic expressions. All of these arithmetic expressions have certain numeric value.

3.5 Expression

- Logical (Boolean) expression
  - A logical or Boolean expression consists of numbers or variables or both made by using the operators >, <, <>, =, <=, >=.
  - Unlike an arithmetic expression, which has some sort of numeric value, a Boolean expression has a value of either true or false.
  - It is also referred as conditional expression.
  - These expressions can be used to test a "relationship" between two values.
  - For example, the conditional expression

    3 > 2 is true, while

    3 > 4 is false.

3.5 Expression - Logical (Boolean) expression

- A key idea is that we can also use variables and/or arithmetic expressions on either side of the relational operator.
- In such cases, these are evaluated (to some pair of numbers) before the relation is evaluated.

- For example, if we had previously performed the following assignments:

  X = 5

  Y = 8

  If we then evaluated the following conditional expression:

  X+1 < Y

  We would first evaluate the expression on either side of the <, giving:

  6 < 8

  which evaluates to true.

  Also note that we can involve strings in conditional expressions, usually to test equality:

  "yes" = answer$

  would be true if the variable answer$ currently stores the string yes.

3.5 Expression - Logical (Boolean) expression

- Example:

```
IF 7 < 10 THEN
    PRINT "seven is less than 10"
ENDIF
```

- In above example the Boolean expression is: 7<10. If we evaluate the Boolean expression for True or False, the result is true.
- When the Boolean expression is true then the program code following the THEN will be executed next.

3.5 Expression - Logical (Boolean) expression

- Example using variables:

```
CLS
number% = 7
value% = 18
IF number% >= value% THEN
    PRINT "seven is greater than or equal to eighteen "
ENDIF
```

- In above example the Boolean expression is: number% > value%. If we evaluate the expression by replacing the variable names with the values assigned to them, the result is false.
- So do you think the PRINT statement will be executed?
- No.

Expression

- String expression
  - A string expression consists of string constants, string variables and other string expressions combined by string operators.
  - There are two classes of string operations: concatenation and string function.
  - The act of concatenating two strings is called concatenation. The plus (+) symbol is the concatenation operator for strings.
  - For example, the following program fragment combines the string variables A$ and B$ to produce the value FILENAME:

```
A$ = "FILE": B$ = "NAME"
PRINT A$ + B$
PRINT "NEW " + A$ + B$
Output:
FILENAME
NEW FILENAME
```

Expression - String expression

- Strings can be compared using the following relational operators: <>, =, <, >, <=, and >=
- Note that these are the same relational operators used with numbers.
- String comparisons are made by taking corresponding characters from each string and comparing their ASCII codes.
- The following are examples of true string expressions:

  "AA" < "AB"

  "FILENAME" = "FILE"+"NAME"

  "X&" > "X#"

  "CL " > "CL"

  "kg" > "KG"

  "SMYTH" < "SMYTHE"

  B$ < "9/12/78" 'where B$ = "8/12/85"
- String comparisons can be used to test string values or to alphabetize strings.
- All string constants used in comparison expressions must be enclosed in quotation marks.

3.6 Statements

- Keyword in any programming language that instructs the computer to carryout the actions that we want is called commands.
- In QBASIC or any programming language it is necessary to learn its keywords, before embarking on writing programs.
- A collection of commands used in the lines of a program is called statements.
- When the program is executed, the statements inside are executed one after another in a controlled sequence.
- Like all programming language, QBASIC also has its own grammar and vocabulary.
- They are used to check the syntax of commands and statements in a program when it is executed.
- It gives message about the validity of commands and statements based on this.

3.6 Statements

- According to the programming tasks, statement can be generally grouped into four categories.
  1. **Assignment Statement**
  2. **Declaration Statement**
  3. **Input / Output Statement**
  4. **Control Statement**
- Besides these, others commands not inside above classifications are:
  - file system commands,
  - string manipulation commands,
  - mathematical calculation,
  - procedure definition commands and
  - commands used in calling other procedures.

Statements – 1. **Assignment Statement**

- A statement used to assign value of a variable is called Assignment Statement. e.g.,

```
LET A = 25
B = 17
LET C = A* B
LET Y = C
```

- In the examples above, the value of expression left of the equal to sign (=), is stored in the variable.
- This is quit different from what we learnt in the algebra, where A = B also implies that B = A.
- But here, A = B means that the value B is stored in the variable A.
- The word LET is optional in the assignment statement. e.g., LET A = B + C can be written as A = B + C.

Statements – 2. **Declaration Statement**

- A statement used to define or declare a constant, variable or array etc. is called Declaration Statement. e.g.,

  a. CONST PI = 3.141593
  b. DIM A(4,4)
  c. REM This program gives the sum of integers
  d. SWAP a%, b%

  In the examples above:

  - PI as a constant,
  - A as an array of 4x4 dimension,
  - REM as a remark is declared.
  - This program gives the sum of integer and a statement for exchanging the values of the integer variables a and b, are also declared.

3.6 Statements – 3. **Input / Output Statement**

- A statement to get data or display processed data in the screen or write it to a printer or a file is called Input / Output Statement. e.g.,

```
INPUT A
INPUT "What is your Name ?"; Name$
PRINT "How are you ? "; Name$
PRINT TAB(25); "WEL – COME"
WRITE #1, Name$, Age$
```

- In the examples above, statements are declared to receive date in the variable A. "What is your Name?" is displayed on the screen and it waits to enter the name in the variable Name$.
- "How are you? " is displayed along with the data that you entered for Name$.
- The WEL-COME is displayed at the 25th column on the screen and the variable data Name$ and Age$ are written in the #1 file.

Statements – 4. **Control Statement**

- A statement, which controls the program flow, while executing the program instructions one after another, is called Control Statement. e.g.,

  a. GOTO lab10:

  b. IF Mark >= 80 THEN GOTO lab5:

  c. IF I% = 15 THEN STOP

  d. END

- In the examples above, statements to control program flow are given.
- Branch to a line with label lab10.
- If Mark is greater or equal to 80 then branch to line with label lab5 and carryout the instruction of that line.
- To stop the program flow if the value of the integer variable I is equal to 15.
- End of the program.

Some Important concepts used in programming.

- **Some Important concepts used in programming.**
  - **Looping and Termination: In programming, repeated execution of a sequence of statements is called Looping. e.g.,**

```
Ans$ = "N"
DO UNTIL Ans$ <> "Y"
.
.
   READ Ans$
..
LOOP
```

  - **In above pseudo code example, the loop is repeatedly executed until the value of the variable Ans$ becomes "Y". In this way, when the value of Ans$ becomes "Y", termination of the loop occurs.**

Some Important concepts used in programming.

- If the termination of a loop does not occur in any conditions, then such loop is called Endless Loop. e.g.,

```
LET msg$ = "Pleased to know you"
Lab10:
PRINT msg$
GOTO Lab10:
END
```

- Here, "Pleased to know you" is printed after the Lab10 and when the execution reaches next line.
- It again returns to label Lab10 and prints the same message again and again.
- The only way out to get out of this endless loop is to terminate the execution by pressing Ctrl + c or Ctrl + Break.
- To keep this type of endless loop in any program is not a good programming, but this is taken as a Logical Error.
- Sometimes, it becomes difficult to find out such errors.

Some Important concepts used in programming.

- **Counter:** This is a variable in which the count of number of occurrences of certain event is stored while executing the program. e.g.,

```
Boy_cnt = 0
Girl_cnt = 0
READ record
DO UNTIL end
    IF sex = "BOY" THEN
            Boy_cnt = Boy_cnt + 1
    ENDIF
    IF sex = "GIRL" THEN
            Girl_cnt = Girl_cnt + 1
    ENDIF
    READ record
LOOP
```

- In above pseudo code example, count of number of boys is stored in Boy_cnt and the count of number of girls is stored in Girl_cnt. After reading each record, it checks whether the sex is boy or girl and then increments the respective counter.

If, in the first record the Sex is "BOY"	then	Boy_cnt = 0 + 1 = 1
If, in the second record the Sex is "GIRL"	then	Girl_cnt = 0 + 1 = 1
If, in the third record the Sex is "GIRL"	then	Girl_cnt = 1 + 1 = 2
If, in the fourth record the Sex is "BOY"	then	Boy_cnt = 1 + 1 = 2
If, in the fifth record the Sex is "BOY"	then	Boy_cnt = 2 + 1 = 3
If, in the sixth record the Sex is "GIRL"	then	Girl_cnt = 2 + 1 = 3

- In this way up to the end, the count of boy and girl is incremented and stored in the counter Boy_cnt and Girl_cnt respectively. These counters can be used in mathematical calculations in the program.

Some Important concepts used in programming.

- **Accumulator:** Any variable, in which results of mathematical calculations are stored while the execution of the programs goes on, is called Accumulator. e.g.,

```
Amount = 0
READ cost
DO UNTIL end
.
    Amount = Amount + Cost
.
.
    READ cost
LOOP
```

- In above pseudo code example, every time the loop is executed, different values of the variable Cost is added to the Amount and stored in it. At the end, the total value of cost is accumulated in the variable Amount.
- In above example,

  In the first record, if Cost = 120 then Amount = 0 + 120 = 120

  In the second record, if Cost = 50 then Amount = 120 + 50 = 170

  In the third record, if Cost = 500 then Amount = 170 + 500 = 670, and so on.

In this way up to the end, the values of Cost are accumulated in the variable Amount. This accumulator can be used in mathematical calculations in program.

Some Important concepts used in programming.

- **Branching or Jumping:** If the execution of any program departures conditionally or unconditionally from its sequential flow, depending on the result of a test, it is called Branching. e.g.,

```
Lab10:
INPUT "Any more"; Ans$
IF Ans$ = "Y" THEN
   GOTO Lab10:
ELSE
   GOTO Lab20:
ENDIF
Lab20:
END
```

- In the above example, response to the prompt Any more? is stored to the string variable Ans$.
- This is tested for “Y”. If the result of this test is True or Yes then the execution of the program branches to lab10: and continues the execution.
- If the result is False or No then the execution of the program branches to Lab20: and ends execution.

Some Important concepts used in programming.

- **Debugging:** Any error or fault present in a program is called bug. Finding out of these bugs and correcting them is called debugging.
- **Flag:** An indicator, which is set or unset depending upon the condition of the program is called a Flag. e.g.,

```
OPEN "TEST.DAT" FOR INPUT AS #1
DO
.
.
.
   READ record
   LOOP UNTIL (EOF(1))
```

- Here, in every loop, it is examined whether the EOF (end of file) is reached for the file opened in file number 1.
- EOF is a kind of flag, which sets to True, when the records reached the end of the file.

Some Important concepts used in programming.

- **Indentation:** Indentation should be followed while writing computer programs. This helps in understanding the programming easily, minimizes bugs and makes the debugging easy. e.g.,

```
REM Program to say Hello
Lab10: INPUT "Your Name :"; Name$
PRINT "Hello! "; Name$
Ans$ = "N"
INPUT "Any more "; Ans$
IF Ans$ = "Y" or Ans$ = "y" THEN
   GOTO Lab10:
ENDIF
END
```

- Like in the IF…. THEN…… ENDIF statement of this program, indentation should be used. Writing in this type of nested structure helps in the easy understanding of the program.

Exercises

1. Explain the terms:
   a. Program and programming
   b. Source program and object program
   c. Compiler and interpreter
   d. Linker and Debugger
   e. Algorithm and flowchart
   f. Coding and debugging.
2. Explain the program development cycle with an example.
3. Write an algorithm and then a flowchart to give instructions for
   a. Making tea
   b. Coming to school from your home.
   c. Arranging your books according to daily routine.
   d. Your friend to visit your home from his residence.
   e. Finding the word “Computer” in your dictionary.

Exercises

4. Write an algorithm and convert it to a flowchart. Then write QBASIC program codes according to the flowchart (after covering related topics).
   a. Read two numbers P and Q, multiply them and print the result.
   b. Calculate the area [A] of a rectangle with given length [L] and breadth [B]. Print the result and make option for more requests.
   c. Read two numbers M and N, subtract second from the first and print the result.
   d. Calculate the circumference (C) of a circle with a given radius (r).
   e. Sum integer numbers from 1 to 50 and print the sum.
   f. Sum odd numbers from 1 to 50 and print the sum.
   g. Sum even numbers from 1 to 50 and print the sum.
   h. Read two numbers A and B divide A by B and print the result. Stop the process if the value of B is equal to 0 otherwise repeat the whole process from the beginning.
   i. Read two numbers P and Q, compare them and print the greatest one.
   j. Calculate the total wages earned at the rate of Rs.125 per day for “n” number days, where n is supplied by the user. Stop the process if the value of n is equal to 0.

Exercises

5. What do you understand by modular approach of developing algorithm?
6. Explain the general symbols used in flowcharts.
7. Define and distinguish between Algorithm and Flowchart.
8. What are the advantages of Flowchart?
9. Draw a flowchart to find the greatest number among ten numbers.
10. Draw a flowchart to find the middle number among three numbers.
11. Draw a flowchart to print the sum of first ten positive integers.
12. Draw a flowchart after reading the following algorithm:

```
Start
Step 1: Store 1 to a
Step 2: Store 1 to n
Step 3: Store 0 to S
Step 4: Get sum of n and S and store in S
Step 5: Double the value of n
Step 6: Increase a by 1
Step 7: Is S less than 131071
            (Yes: Goto step 4
            No:)
Step 8: Print a
Stop
```

Exercises

13. Draw a flowchart according to the following algorithm:

    Start
    Step 1: Read A
    Step 2: Read B
    Step 3: Read C
    Step 4: Store A in H
    Step 5: Is B greater than H
    (Yes: Store B in H)
    Step 6: Is C greater than H
    (Yes: Store C in H)
    Step 7: Display H
    Stop

Exercises

14. Draw a flowchart to add N terms of the series.
    S = 12 + 22 + 32 + 42 + 52+ ..... + N2
15. What is programming? What tasks does a programming language perform?
16. What is the full form of QBASIC?
17. How can you categorize the symbols in a computer keyboard? Explain.
18. How can you choose Menu items, if the computer does not have a mouse?
19. What are the Menu bar items displayed, when the QBASIC is loaded.
20. What are the actions that can be performed under the pull down menu of following menu items?
    a. FILE
    b. EDIT
    c. SEARCH
21. Describe the necessary steps to write a program (WAP) in QBASIC and run it?
22. What is the major difference between QBASIC and other versions of BASIC?
23. How do you save a QBASIC program?
24. What is the use of the Output Screen when executing a QBASIC program?

Exercises

25. What do you understand by Operators? How many types of operators are there?
26. What differences do you have to think about, while using algebraic expression in QBASIC.
27. Make a table of Operators in the sequence of execution.
28. What do you understand by Constant and Variable?
29. What do you understand by Arithmetic Operators in QBASIC, in what tasks they are used?
30. What is the importance of Relational Operators in QBASIC?
31. What do you understand by Logical Operators in QBASIC? What are the mostly used logical operators?
32. Describe the following logical operator with outcome table and explain with example?

    (i) AND (ii) OR (iii) NOT

Exercises

33. Write down the algebraic expressions given below into QBASIC operation:

(i) $XY/2$ (ii) $(A + B)(C - D)$ (iii) $M N + 100$ (iv) $\frac{A}{B} + \frac{C}{D}$ (v) $\frac{AB}{N}$

(vi) $3(M + N)$ (vii) $P(Q + R)$ (viii) A^2 (ix) $10M + N$

(x) $P + 8 PQ$ (xi) $(K + L)M2$ (xii) $1 PR^2$ (xiii) X^5Y^5

(xiv) $\frac{X}{Y^3}$ (xv) $\frac{A^2B^2}{2} - 5B$

(xvi) $UT + \frac{1}{2} gT^2$ (xvii) $VA(A-B)(A-C)(A-D)$

(xviii) $A = \sqrt{s(s-a)(s-b)(s-c)}$

(xix) $V = \frac{P(1 + R)N}{100}$ (xx) $\frac{PQ}{5} - 3(R)$

Exercises

34. Correct the QBASIC operations given below:

(a) A = 2D (b) X(Y*Z) (c) P = Q*QR (d) I = PRT /100

(e) X = Y – 2 (f) K = 5.1 L + M (g) R = E/A/D (h) X = Y* - Z

(i) X = 25Y+Z (j) A = P + PNR / 100

35. Which expressions given below do not give an outcome of 4 in QBASIC?

36. (a) (4 + 4) / 4 (b) 4 * (4 / 4) (c) 4 / (4 / 4) (d) 4+ 4 / 4
(e) 8 / 2

37. Find step-wise outcome to solve the expression given below according to the QBASIC execution sequence.

(a) X = 7^2 + 2 * (7 * 9) + 9^2 (b) Y = (6 + 10)^2 – 5 * (2 + (3+1) / 2)

(c) Z = 16 / 8 – (60 – 10) / (20 + 5) (d) P = 2^2^2^2^2

(e) Q = 5 * 3^2^3 * 2^3^2 (f) R = -5*(10+2)+(5+(4+(2*3)– 2) * 10)

(g) X = 5^2 + 2*5*4 + 4^2 (h) Y= 5 * 3 + 5 * (7 – 5) – 2^2

(i) X = 9^2 – (2^5 – 2 * 5) (j) Y = 15 / 5 + (5^2 – 5 * 2)

Exercises

38. Write down the QBASIC operations given below in to algebraic expressions.

    (a) X^2 + 2 * X * Y + Y^2 (b) (A * (A + B))^(1/2)

    (c) X / Y + (X^2 – X * Y) (d) P * Q – 5

    (e) E * (F + G) (f) 5 * (-P)

    (g) (1/2) * b * h (h) (a + b + c) / (a + b) (i) U * T + (G * T) / 2

    (j) X / Y ^ 3

39. How do you compare string using relational operators? Explain with example.

40. What is the convention for writing variables name that stores single precision number and double precision number in QBASIC? Give examples.

41. Distinguish between real number and integer number with examples of where they are used.

Exercises

42. What is the convention for writing variables name that stores real number and integer number in QBASIC? Give examples.
43. What do the following declarations represent?
    i. salesvalue# ii. amount% iii. address$
    iv. costprice! v. netamount&
44. What do you understand by Command and Statement?
45. In how many types Statement can be categorized? Explain one of them.
46. Explain the following concepts in programming.
    (a) Looping and Termination (b) Counter (c) Accumulator
    (d) Branching or Jumping (e) Debugging (f) Flag
47. Explain the difference in the meaning of P = Q between algebra and programming statement
48. Explain the difference between a counter and an accumulator concept in programming.

Exercises

49. What can we achieve by devising a loop in a program?
50. What is the importance of writing nested structure in programming?
51. What are the difference between the Assignment Statement and the Declaration Statement? Explain with examples.
52. How does QBASIC check the validity of the syntax of commands and statements?
53. What is endless loop in programming? How do you avoid it? Explain with an example.
54. What can we achieve by devising a loop in a program?

Exercises

55. Evaluate the following Boolean Expressions

Expression	Evaluation (True/False)
7 > 3	
11 >= 11	
10 > 9	
5 <= 6	
4=14	

56. Create two Boolean expressions and evaluate them.

Boolean Expression	Evaluation
1.	
2.	

Exercises

57. What would the output of the following program be?

```
IF 2 < 4 THEN
    PRINT "Today is Wednesday"
ENDIF
```

58. What would the output of the following program be?

```
IF 25 <= 26 THEN
        PRINT "Qbasic is fun!"
ENDIF
```

59. What would the output of the following program be?

```
IF 6 > 9 THEN
        PRINT "Black and Gold Dance this Friday!"
ENDIF
```

Exercises

60. What would the output of the following program be?

```
number1% = 100
number2% = 97
IF number1% < number2% THEN
        PRINT "Halloween is on Friday"
ENDIF
```

61. What would the output of the following program be?

```
IF "a" < "b" THEN
        PRINT "Now we are testing strings."
ENDIF
```

62. What would the output of the following program be?

```
IF "hear" = "here" THEN
        PRINT "These words sound the same"
ENDIF
```

63. What would the output of the following program be?

```
word1$ = "lasalle"
words2$ = "knights"
IF word1$ > word2$ THEN
        PRINT "We are the Lasalle Black Knights!"
ENDIF
```