3.0 INTRODUCTION

A computer contains two basic parts: (i) Hardware and (ii) Software. In the first two units we touched upon hardware issues in quite detail. In this unit and also in the rest of the units of this block we will discuss topics related to software. Without software a computer will remain just a metal. With software, a computer can store, retrieve, solve different types of problems, create friendly environment for software development etc.

The process of software development is called programming. To do programming one should have knowledge of (i) a particular programming language, (ii) set of procedures (algorithm) to solve a problem or develop software.

The development of an algorithm is basic to computer programming and is an important part of computer science studies. Developing a computer program is a detailed process, which requires serious thought, careful planning and accuracy. It is a challenging and exacting task, drawing on the creativity of the programmer.

Once an algorithm is obtained, the next step for a solution using a computer would be to program the algorithm using mathematical and data processing techniques. **Programming languages constitute the vehicle for this stage of problem solving.** The development of programming Languages is one of the finest intellectual achievements in Computer Science. It has been said **"to understand a computer, it is necessary to understand a programming language. Understanding them does not really mean only being able to use them. A lot of people can use them without really fully understanding them".**

An Operating System is system software, which may be viewed as an organised collection of software consisting of procedures for operating a computer and providing an environment for execution of programs. It acts as an interface between users and the hardware of a computer system.

There are many important reasons for studying operating systems. Some of them are:

(1) User interacts with the computer through operating system in order to accomplish his task since it is his primary interface with a computer.

(2) It helps users to understand the inner functions of a computer very closely.

(3) Many concepts and techniques found in operating system have general applicability in other applications.

In this unit, we will discuss about the concepts relating to a programming language and in the next unit we will deal with the operating system concepts.

3.1 OBJECTIVES

After completion of this unit, you should be able to

- define the terms system and application software
- define various categories of language
- define the terms such as computers, interpreters, fourth generation languages
- define the elements of programming languages such as arrays, expressions, variables, input-output statements
- define conditional and looping structures
- define subroutines and functions.

3.2 COMPUTER SOFTWARE

Computer software consists of sets of instructions that mould the raw arithmetic and logical capabilities of the hardware units to perform.

In order to communicate with each other, we use natural languages like Hindi, English, Bengali, Tamil, Marathi, Gujarati etc. In the same way programming languages of one type or another are used in order to communicate instructions and commands to a computer for solving problems. Learning a programming language requires learning the symbols, words and rules of the language.

Program and Programming: A computer can neither think nor make any judgement on its own. Also it is impossible for any computer to independently analyse a given data and follow its own method of solution. It needs a program to tell it what to do. A program is a set of instructions that are arranged in a sequence that guides the computer to solve a problem.

The process of writing a program is called **Programming.** Programming is a critical step in data processing. If the system is not correctly programmed, it delivers information results that cannot be used. There are two ways in which we can acquire a program. One is to purchase an existing program, which is normally referred to as packaged software and the other is to prepare a new program from scratch in which case it is called customised software.

A computer software can be broadly classified into two categories-**System Software** and **Application Software.**

Today, there are many languages available for developing programs software. These languages are designed keeping in mind some specific areas of applications. Thus, some of the languages may be good for writing system **programs/software** while some other for **application software**. Since a computer can be used for writing various types of application/system software, there are different programming languages.

i) **System Programming Languages: System programs are designed to make the computer easier to use:** An example of system software is an operating system, which consists of many other programs for controlling input/output devices, memory, processor etc. To write an operating system, the programmer needs instruction to control the computer's circuitry (hardware part). For example, instructions that move data from one location of storage to a register of the processor. C and C++ languages are widely used to develop system software.

ii) **Application Programming Language**: Application programs are designed for specific applications, such as payroll processing, inventory control etc. To write programs for payroll processing or other applications, the programmer does not need to control the basic circuitry of a computer. Instead the programmer needs instructions that make it easy to input data, produce output, do calculations and store and retrieve data. Programming languages that are suitable for such application programs support these instructions but not necessarily the types of instructions needed for development of system programs.

There are two main categories of application programs: business programs and scientific application programs. Most programming languages are designed to be good for one category of applications but not necessarily for the other, although there are some general purpose languages that supports both types. Business applications are characterised by processing of large inputs and large outputs, high volume data storage and retrieval but call for simple calculations. Languages, which are suitable for business program, development, must support high volume input, output and storage but do not need to support complex calculations. On the other hand, programming languages that are designed for writing scientific programs contain very powerful instructions for calculations but rather poor instructions for input, output etc. Amongst traditionally used programming languages, COBOL (Commercial Business Oriented Programming Language) is more suitable for business applications whereas FORTRAN (Formula Translation - Language) is more suitable for scientific applications. Before we discuss more about languages let us briefly look at the categories of software viz. system and application software.

3.2.1 System Software

Language Translator: A language translator is a system software which translates a computer program written by a user into a machine understandable form. We will discuss more about them in the next section.

Operating System

An operating system (OS) is the most important system software and is a must to operate a computer system. An operating system manages a computer's resources very effectively, takes care of scheduling multiple jobs for execution and manages the flow of data and instructions between the input/output units and the main memory. Advances in the field of computer hardware have also helped in the development of more efficient operating systems. More details on operating systems are given in unit 4 of this block.

Utilities

Utility programs are those which are very often requested by many application programs. A few examples are:

SORT/MERGE utilities, which are used for sorting large volumes of data and merging them into a single sorted list, formatting etc.

3.2.2 Application Software

Application software is written to enable the computer to solve a specific data processing task. A number of powerful application software packages, which does not require significant programming knowledge, have been developed. These are easy to learn and use as compared to the programming languages. Although these packages can perform many general and special functions, there are applications where these packages are not found adequate. In such cases, application program is written to meet the exact requirements. A user application program may be written using one of these packages or a programming language. The most important categories of software packages available are:

- Data Base Management Software
- Spreadsheet Software
- Word Processing Desktop Publishing (DTP) and presentation Software Graphics Software
- Data Communication Software
- Statistical and Operational Research Software.

Data Base Management Software

Databases are very useful in creation maintaining query, the databases and generation of reports. Many of today's Database Management System are Relational Database Management System's. Many RDBMS packages provide smart assistants for creation of simple databases for invoices, orders and contact lists. Many database management systems are available in the market these days. You can select any one based on your needs, for example, if you have only few databases then package like dBase, FoxPro etc. may be good. If you require some additional features and moderate work load then Lotus Approach, Microsoft Access are all-right. However, if you are having high end database requirements which requires multi-user environment and data security, access right, very good user interface etc. then you must go for professional RDBMS package like Ingress, Oracle, Integra etc.

Accounting Package

The accounting packages are one of the most important packages for an office. Some of the features, which you may be looking on an accounting, may be:

- tax planner facility
- facility for producing charts and graphs
- finding accounts payable
- simple inventory control facility
- payroll functions
- on-line connection to stock quotes
- creation of invoices easily

One of the good packages in this connection is Quicken for windows.

Communication Package

The communication software includes software for fax. The fax-software market is growing up. Important fax software is Delrina's WinFax PRO 4.0. Some of the features such as Remote Retrieval and Fax Mailbox should be looked into fax software. These features ensure that irrespective of your location you will receive the fax message.

Another important feature is fax Broadcast. This allows you to send out huge numbers of faxes without tying up your fax machine all day.

If you have to transfer files from your notebook computer to a desktop computer constantly then you need a software program that coordinates and updates documents. On such software is Laplink for Windows. This software offers very convenient to use features. For example, by simply dragging and dropping a file enables file transfer. This software can work if a serial cable or a Novell network or a modem connects you.

Desktop Publishing Packages

Desktop Publishing Packages are very popular in Indian context. Newer publishing packages also provide certain in built formats such as brochures, newsletters, flyers etc., which can be used directly. Already created text can be very easily put in these packages, so are the graphics placements. Many DTP packages for English and languages other than English are available. Microsoft Publisher, PageMaker, Corel Ventura are few popular names. Desktop publishing packages, in general, are better equipped in Apple-Macintosh computers.

Information Providers

One of the very interesting information provider which will become popular in India also is Automap road atlas by Microsoft. This package may provide city-to-city driving instructions and maps. You may also get the best route, calculate the time it will take.

Many information providers are the Internet access programs. Today, the Internet access packages comes as a part of operating system however, many other packages can be used for accessing information on the World Wide Web. One very simple to use popular tool of browsing Internet is Netscape Navigator.

Organisers, Contact Managers, PIMs

Some of the tasks of an office manager can be:
- to be able to track contacts
- to balance schedules
- to manage projects
- to prioritise tasks

These things can be easily done using organisers programs, which have a phone book model for maintaining lists of contacts. They also have a calendar for entering appointments and to-dos. Some of these packages are Okna's DeskTop Set for Windows, Lotus organiser, Microsoft Outlook, etc.

If you are interested in knowing more, than only names and addresses about your contacts such as details like the industry they are working with, the products they are manufacturing, their business with you last year, when did you last spoke to them etc., then you must look to a contact management software. One such software is "Symantec Act! for Windows".

If you want to go even further then you can look for a personal information manager (PIM). PIM is a tool that stores virtually any information such as reference materials, project details etc. The PIM document contains outlines, folders and links. Most of the data in the PIMs is presented as an outline, for example, the clients may represent the top level, the date of an appointment with him at the next level, and the details of the meeting indented further below. This item can be linked to any other, allowing data to be entered only once and linked up to all other appropriate places.

Suites

Suites are a set of packages sold as a group package mainly for the business user. The suite package includes programs for Word-processing, Electronic Spreadsheet, Databases, and Presentation Graphics software and may be a mail software. For example, Microsoft Office Professional for Windows includes programs as Microsoft Word, Microsoft Excel and Microsoft Access, and a license for Microsoft Mail etc. The word-processing, spreadsheet, and presentation-graphics software interfaces in a suite are well-integrated allowing easy data transfer among these applications. Today there is a growing family of Office-compatible products, which will be included in suites.

In the fast developing software era the list discussed above cannot be complete. Please refer to latest PC journals for most recent software trends.

Check Your Progress 1

1. What is computer software?

   ....

   ....

2. Explain the following terms in one or two sentences each.

   (a) Operating Systems

   ....

   ....

   (b) Database Management Software

   ....

   ....

3.3 CATEGORIES OF LANGUAGES

We can choose any language for writing a program according to the need. But a computer executes programs only after they are represented internally in binary form (sequences of 1s and 0s). Programs written in any other language must be translated to the binary representation of the instructions before the computer can execute those. Programs written for a computer may be in one of the following categories of languages.

3.3.1 Machine Language

This is a sequence of instructions written in the form of binary numbers consisting of l s, 0s to which the computer responds directly. The machine language was initially referred to as code, although now the term code is used more broadly to refer to any program text.

An instruction prepared in any machine language will have at least two parts. The first part is the command or Operation, which tells the computer what functions, is to be performed. All computers have an operation code for each of its functions. The second part of the instruction is the operand or it tells the computer where to find or store the data that has to be manipulated.

Just as hardware is classified into generations based on technology, computer languages also have a generation classification based on the level of interaction with the machine. Machine language is considered to be the first generation language.

Advantage of Machine Language

It is faster in execution since the computer directly starts executing it.

Disadvantage of Machine Language

It is difficult to understand and develop a program using machine language. Anybody going through this program for checking will have a difficult task understanding what will be achieved when this program is executed. Nevertheless, the computer hardware recognises only this type of instruction code.

The following program is an example of a machine language program for adding two numbers.

0011	1110	Load A register with
0000	0111	value 7
0000	0110	Load B register with 10
0000	1010	A = A+ B
1000	0000	Store the result
0011	1010	into the memory location
0110	0110	
0000	0000	whose address is 100 (decimal)
0111	0110	Halt processing

3.3.2 Assembly Language

When we employ symbols (letter, digits or special characters) for the operation part, the address part and other parts of the instruction code, this representation is called an assembly language program. This is considered to be the second-generation language.

Machine and Assembly languages are referred to as low level languages since the coding for a problem is at the individual instruction level.

Each machine has got its own assembly language, which is dependent upon the internal architecture of the processor.

An assembler is a translator, which takes its input in the form of an assembly language program and produces machine language code as its output.

The following program is an example of an assembly language program for adding two numbers X and Y and storing the result in some memory location.

LDA, 7	Load register A with 7
LDB, 10	Load register B with 10
ADD A, B	$A \leftarrow A + B$
LD (100), A	Save the result in the location 100
HALT	Halt process

From this program, it is clear that usage of mnemonics in our example LD, ADD, HALT are the mnemonics) has improved the readability of our program significantly.

A machine cannot execute an assembly language program directly, as it is not in a binary form. An assembler is needed in order to translate an assembly language program into the object code executable by the machine. This is illustrated in the figure 1.

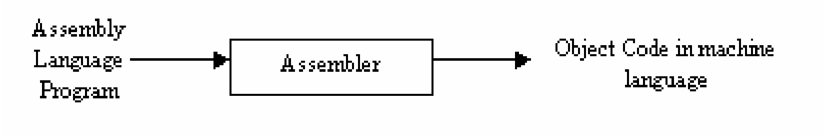


Figure 1: Assembler

Advantage of Assembly Language

Writing a program in assembly language is more convenient than in machine language. Instead of binary sequence, as in machine language, it is written in the form of symbolic instructions. Therefore, it gives a little more readability.

Disadvantages of Assembly Language

Assembly language (program) is specific to particular machine architecture. Assembly languages are designed for specific make and model of a microprocessor. It means that assembly language programs written for one processor will not work on a different processor if it is architecturally different. That is why the assembly language program is not portable.

Assembly language program is not as fast as machine language. It has to be first translated into machine (binary) language code.

3.3.3 High-level Language

We have talked about programming languages as COBOL, FORTRAN and BASIC. They are called high level programming languages. The program shown below is written in BASIC to obtain the sum of two numbers.

```
10      LET      X       =      7

20      LET      y       =      10

30      LET      sum     =      X+Y

40      PRINT    SUM

50      END
```

The time and cost of creating machine and assembly languages was quite high. And this was the prime motivation for the development of high level languages.

Since a high level source program must be translated first into the form the machine can understand, this is done by a software called Compiler which takes the source code as input and produces as output the machine language code of the machine on which it is to be executed. This is illustrated in figure 5.

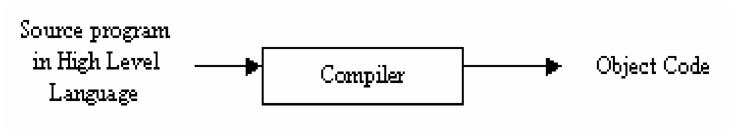


Figure 2: Compiler

During the process of translation, the Compiler reads the source programs statement-wise and checks the syntax (grammatical) errors. If there is any error, the computer generates a printout of the errors it has detected. This action is known as diagnostics.

There is another type of software, which also does the translation. This is called an Interpreter. The Compiler and Interpreter have different approaches to translation. The following table lists the differences between a Compiler and an Interpreter.

Compiler	**Interpreter**
1) Scans the entire program first and then translates it into machine code	Translates the program line by line.
2) Converts the entire program to machine code; when all the syntax errors are removed execution takes place.	Each time the program is executed, every line is checked for syntax error and then converted to equivalent machine code.
3) Slow for debugging (removal of mistakes from a program)	Good for fast debugging
4) Execution time, is less	Execution time is more.

Advantages of High-level Programming Language

There are four main advantages of high-level programming languages. These are:

i) **Readability:** Programs written in these languages are more readable than assembly and machine language.

ii) **Portability**: Programs could be run on different machines with little or no change. We can, therefore, exchange software leading to creation of program libraries.

iii) **Easy debugging:** Errors could easily be removed (debugged).

iv) **Easy Software development:** Software could easily be developed. Commands of programming language are similar to natural languages (ENGLISH).

High level languages are also called Third generation languages.

3.3.4 Fourth Generation Language

The fourth generation of programming languages is not as clearly defined as are the other earlier generations. Most people feel that a fourth generation language, commonly referred to as 4GL is a high level language that requires significantly fewer instructions to accomplish a particular task than a third generation language does. Thus, a programmer should be able to write a program faster in 4GL than in third generation language.

Most third generation languages are procedural languages. This means that the programmer must specify the steps, that is the procedure, the computer has to follow in a program. By contrast, most fourth generation languages are non-procedural languages. The programmer does not have to give the details of procedure in the program, but instead, specifies what is wanted. For example, assume that a programmer needs to display some data on a screen, such as the address of a particular employee (SANTOSH) from the personal file. In a procedural language, the programmer would have to write a series of instructions in the following steps:

Step 1 Get a record from the personal file

Step 2 If this is the record for SANTOSH, display the address

Step 3 If this is not the record for SANTOSH, go to Step 1.

In a non-procedural language (4GL), however, the programmer would write a single instruction that says:

Get the address of Santosh, from personal file.

Major fourth generation languages are used to get information from files and data bases, as in the previous example and to display or print the information.. These fourth generation languages contain a query language which, is used to answer queries or questions with data from a database. For example the following figure shows a query in a common query language SQL,

```
SELECT ADDRESS FROM PERSONNEL
WHERE NAME = "SANTOSH"
```

Some fourth generation languages are used to produce complex printed reports. These languages contain certain types of program called generators. With a report generator available, the programmer specifies the headings, detailed data and totals needed in a report. Thus, the report generator produces the required report using data from a file. Other fourth generation languages are used to design screens to be used for data input and output and for menus. These languages contain certain types of programs called screen painters. The programmer designs how the screen is to look and, therefore, we can say that the programmer paints the screen, using the screen painter program. Fourth generation languages are mostly machine independent. Usually they can be used on more than one type of computer. They are mostly used for office automation or business applications, but not for scientific programs. Some fourth generation languages are designed to be easily learnt and used by end users.

Check Your Progress 2

1. What are the differences between a compiler and an interpreter?

   ....

   ....

2. **State True of False.**

   (a) Assembly language programs are machine independent.
   True ☐ False ☐

   (b) Machine language programs are machine independent.
   True ☐ False ☐

   (c) High level languages are machine independent.
   True ☐ False ☐

   (d) All programs are machine independent.
   True ☐ False ☐

   (e) 4th generation languages are dependent or databases they are using.
   True ☐ False ☐

3.4 ELEMENTS OF A PROGRAMMING LANGUAGE

Learning a programming language requires understanding of concepts such as representation of different types of data in the computer, various methods of expressing mathematical and logical relationship among data elements and the mechanics for controlling the sequence in which operations can be executed for inputting, processing and outputting of data.

3.4.1 Variables, Constants, Data type, Array and Expressions

These are the smallest components of a programming language.

Variable: The first thing we must learn is how to use the internal memory of a computer in writing a program. Memory may be pictured as a series of separate memory cells as shown in figure 3. Computer memory is divided into several locations. Each location has got its own address.

Cell 1
Cell 2
Cell N

Figure 3: Memory Organisation

Each storage location holds a piece of information. In order to store or retrieve information from a memory location, we must give that particular location a name. Now study the following definition.

Variable: It is a character or group of characters assigned by the programmer to a single memory location and used in the program as the name of that memory location in order to access the value stored in it.

For example in expression A = 5, A is a name of memory location i.e. a variable where 5 is stored.

Constant: It has fixed value in the sense that two cannot be equal to four. String constant is simply a sequence of characters such as "computer" which is a string of 8 characters. The numeric constant can be integer representing whole quantities or a number with a decimal point to represent numbers with fractional part. Constant would be probably the most familiar concept to us since we have used it in doing everything that has to do with numbers. Numeric constants can be added, subtracted, multiplied, divided, and also compared to say whether two of them are equal, less than or greater than each other.

As string constants are a sequence of characters, a related string constant may be obtained from a given one, by chopping off some characters from beginning or end or both or by appending another string constant at the beginning or end. For example, from 'Gone with the wind', we can get 'one with ', 'Gone with wind', and so on. String constants can also be compared in a lexicographic (dictionary) sense to say whether two of them are equal, not equal, less than or greater than each other.

Data type: In computer, programming, the term data refers to anything and everything processed by the computer. There are different types of data processed by the computer, numbers are one type of data and words are of another type. In addition, the operations that are performed on data differ from one type of data to another type. For example multiplication applies to numbers and not words or sentences.

Data type defines a set of related values/integers, number with fraction, characters and a set of specific operations that can be performed on those values.

In BASIC a statement LET A = 15 denotes that A is a numeric data type because it contains numbers but in a statement LET A$ = "BOMBAY", A$ is a variable of character data type. Data type also defines in terms of contiguous cells should be allocated for a particular variable.

Array: In programming we deal with large amount of related data. To represent each data element we have to consider them as separate variables. For example if we have to analyse for the sales performance of a particular company for the last 10 years, we can take ten different variables (names) each one representing sales of a particular year. If we analyse sales information for more than 10 years, then accordingly number of variables will further increase. It is very difficult to manage with large number of variables in a program. To deal with such situation an array is used.

An array is a collection of **same type of data** (either string or numeric), all of that **are referenced by the same name.**

For example, list of 5 years sales information of a company can be referred to by same array name A.

A(1)	A(2)	A(3)	A(4)	A(5)
50,000	1,00,000	5,00,000	8,00,000	9,00,000

A(1) specifies Sales information of a first year

A(2) specifies Sales information of a second year

A(3) specifies Sales information of a fifth year

Expression: We know that we can express intended arithmetic operations using expressions such as X +Y+ Z and so on. Several simple expressions can even be nested together using parentheses to form complex expressions. Every computer language specifies an order by in which various arithmetic operators are evaluated in a given expression. An expression may contain operators such as

Parentheses	( )
Exponentiation	^
Negation	-
Multiplication, division	*, /
Addition, subtraction	+, -

The operators are evaluated in the order given above. For example, the expression

2+8*(4 - 613)

can be considered to be evaluated as follows:

2+8*(4 - 6/3)	Sub expression (4 – 6/3) taken up first
2+8*(4 - 2)	division 6/3 within (4 - 6/3) has higher priority than 4 - 6
2+8*2	Subtraction (4 - 2) is performed next (4 – 6/3) is now complete.
2+8*2	8*2 will be executed first then its result will be added with 2 that is 16 + 2 = 18

It is useful to remember the order of priority of various operators. But it is safer to simplify expressions and enclose them in parentheses to avoid unpleasant surprises. So far we have focused on arithmetic expressions. But expression is a very general concept. We mentioned earlier that apart from arithmetic operations we could compare numbers or strings. We do it by using relational operators in expressions.

The following is a list of relational operators:

=	equal to
< >	not equal to
<	less than.
>	greater than
<=	less than or equal to
>=	greater than or equal to

These operations have the same level of Priority among themselves but a lower priority than arithmetic operators mentioned earlier. The relational expressions result in one of the truth-values, either TRUE or FALSE. When a relational expression such as (3 > 5) is evaluated to be FALSE by such languages, a value 0, that is false, is assigned, whereas (5, < 7) will be evaluated to be TRUE, and value 1 will be assigned.

Note that relational expressions are capable of comparing only two values separated by appropriate relational operator. If we want to an express idea such as whether number 7 happens to be within two other numbers 4 and 10, we may be tempted to write relational expression 4 <= 7 <= 10. Such reasonable expectation from us may be a bit too complex for a computer language. In such cases, we need to explain our idea in terms of simple relational expressions such as (4 <= 7) AND (7 <= 10) which means that 7 is between 4 and 10. To combine several relation expressions for expressing complex conditions, we use logical operators such as AND or OR operators. Among other logical operators NOT simply negates a truth value in the sense that NOT TRUE is FALSE, and NOT FALSE is TRUE. The logical operators have lower priority than relational operators. Among themselves they have the following order of priority during evaluation of logical expressions.

Operator	**Meaning**
NOT	Simply negates a truth-value. NOT (2 >5) is TRUE
AND	TRUE is both adjoining expressions are TRUE otherwise it is FALSE. For example (4 < 7) and (7 < 10) is TRUE whereas (4 > 7) and (7 < 10) is FALSE
OR	FALSE is both adjoining expressions are FALSE otherwise it is TRUE For example (4 < 2) OR (7 > 2) is FALSE whereas (4 > 2) OR (7 > 2) is TRUE
XOR	TRUE only if one of the adjoining expressions is TRUE and other is FALSE. The XOR has same priority as OR. (4 < 7) XOR (7 < 10) is FALSE.

Of the four logical operators NOT, AND and OR are sufficient to express any logical condition.

Expressions would be our basic programming unit. Our Programs would be full of them, and while executing the program, computer would be preoccupied most of the -time evaluating them. But the computer would also be doing other things such as taking some values as input from an input device, assigning a value or result of an expression to a variable, send a value for display to a display unit, pondering to be or not to be on some conditional expression, Repeating some tasks until we are satisfied, and calling some slave-programs to perform specific tasks. In the following discussion we shall elaborate on these actions.

3.4.2 Input and Output Statement

Syntax: input list of variables

Input statements are normally used for supplying values to a list of variables through an input device such as a keyboard. On execution, a question mark? (in BASIC) would be displayed on the screen, in response to which we are expected to supply a sequence of desired values matching appropriately the type (i.e. integer, floating-point or string type) and the number of variables in the list. For example in BASIC, INPUT A, B, C is an input statement with list of 3 variables A,B,C.

Assignment Statement

Syntax: variable=expression

This is the most obvious statement used, for assigning value to an identifier (i.e. variable). In BASIC languages this is also known as LET ' statement though the key word LET is optional in most of them. Note the use of = as assignment operator. You would recall that it was also used as relational operator. Some purists justifiably object to ambiguous use of =. In Pascal, : = is used for assignment instead of just = which is used as relational operator. In language C '=' is used for assignment but for relational operator == is used. Few examples of assignment statement in BASIC are given below.

```
LET SUM = X+Y

LET X     = 5
```

SUM and X are called variables whereas A + B is called an expression.

Print Statement

Syntax: print list of variables

This is again an obvious way to display values of specified list of variables on display screen. We shall make use of print to represent generic output statement independent of any particular output device such as display screen, printer and so on. Computer languages normally provide very rich facilities for making the display as attractive as possible. An Example of print statement in BASIC is PRINT A,B,C, which will display values of A,B,C on display screen.

3.4.3 Conditional and Looping Statement

Conditional Statement

If then_else structure

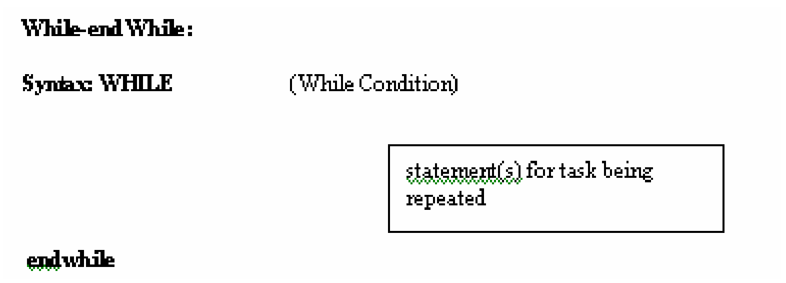
Syntax:

```
if (condition)

Then

                                  statement(s) for Task A

else

                                  statement(s) for Task A

endif
```

This is one of the most important program-structures that enable a modern programmer to develop well-structured programs. Note that we are calling it as structure rather than statement. As structure it accommodates statements corresponding to alternate tasks to be performed depending upon truth-value of the condition. The structure has three components: a conditional expression the truth-value of which determines the action, a then-clause denoting task to be performed if the condition is TRUE (Task A), and an optional else clause with task to be performed if the condition is FALSE.

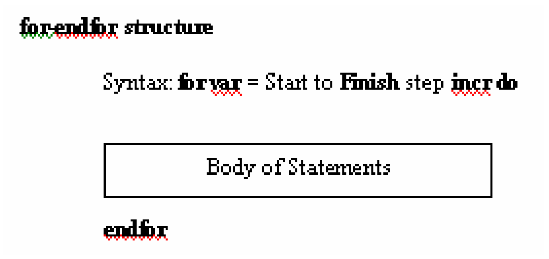
Looping Statement

Obviously, the purpose of a loop structure is to repeat certain tasks until no more repetition is desired. We shall be presenting several variations of loop structure appropriate to handle different situations.



```
While-end While:

Syntax: WHILE          (While Condition)

                          statement(s) for task being
                          repeated

endwhile
```

A loop structure is characterised by the body of statements to be repeated, and the condition for termination of loop. In while-end while version the condition is tested before every execution of the body. The body is executed if the condition is found to be TRUE. The loop is active as long a condition evaluates to be TRUE and terminates when the condition is evaluated to be FALSE. In case the condition is found to be FALSE for the first time, the body will be skipped.



```
for-endfor structure

    Syntax: for var = Start to Finish step incr do

        Body of Statements

    endfor
```

This loop structure is distinctly different from the while condition - end while we have discussed so far. Unlike it, the for endfor structure counts the number of times the body is to be executed on the basis of the number of times a control variable is incremented from start to finish in specified steps. Managing the control variable is the responsibility of the for - endfor structure. So we need not and should not worry about assigning any value to it in the body of statements. Any attempt to do so may land us in trouble.

This loop structure is so famous that it is available in all languages, structured or not. As usual, key words provided may be different. The Microsoft BASIC calls it for-next, and optionally provides for specification of control variable after next. Many other languages insist on mention of the control variable after next.

We have discussed program structures quite elaborately. Let us turn our attention to data structures that hold our values. So far we have been dealing with simple variables that hold only one value of a desired type. So we have integer variable, floating-point variable, and string variable holding only one value on the corresponding type at any moment. Can we have data structures that are capable of holding more than one value? Programming languages do offer complex data structures required for advanced programming. Array and record are two of them. A detailed discussion of these topics is beyond this unit, however you can refer to further readings for more details.

3.4.4 Subroutine and Functions

When writing a program sometimes, it is necessary to repeat a statement or group of statements at several -points. For example, it may be necessary to perform the same computation several times. Repeating the same, statement in a program each time makes a program lengthy and also gives poor readability. These problems could be sorted out if the necessary statements could be written (coded) once and then referred to each time they are needed.

This is the purpose of subroutine and function. Subroutine and function consist of one or more basic statements that can be referred to at different points in a program. Each time a subroutine and function is referenced, we say it is *called.*

A subroutine is a group statement that can be executed from different points in a program. Syntax for defining subroutine varies from one language to another language.

Functions: The word function in programming means essentially what it does in mathematics. Function is a rule or series of rules that assign one and only one value Y to a given value X. Thus, X and Y is called variables. Value of Y is dependent upon value of X. Variable X is also called the argument of the function. The notation used for function should already be quite familiar to you.

$$Y= F(X)$$

In this notation "F" is the name of the function. Consider a specific example

$$Y = F(X) = 3X + 3$$

If we let X equal 3, then this rule or function directs us to compute 12 for Y If we assign the value of X = - 3, then Y is - 6. Using function notation, F(3) = 12 and F(-3) = -6. Each language has different rules to define a function.

Library functions: These are functions supplied with your language compiler interpreter. The code for a library function does not appear in the program. The computer inserts it when the program is translated into machine language. Some built-in functions in C programming language are Printf, Scanf etc.

Check Your Progress 3

Question 1 What is the purpose of a looping statement in a programming language?

....

....

Question 2 What is the advantage of using an array in a programming language?

....

....

Question 3 What is the difference between subroutine and function ?

....

....

3.5 SUMMARY

In this unit, we have introduced you many aspects relating to computer software and programming languages. We have defined the terms system and application software giving details on specific application software. We have also discussed about various categories of programming languages starting from machine language to fourth generation language. The concepts of compilers and interpreters have also been introduced along with this discussion. Finally we have discussed about basic elements of any programming language. The concepts introduced are variables, constants, arrays, expressions, input output statements, assignment statements, conditional and looping structures and subroutine functions. These concepts are essential to acquire a proper know-how about any programming language. You can refer to further readings for more details.

3.6 MODEL ANSWERS

Check Your Progress 1

1. It consists of sets of instructions, which enable hardware unit to perform.

2. (a) Operating System is a system software for effective management of computer resources. It is a must to operate a computer system.

   (b) A software that maintains and integrates large volume of data and provide simpler user interaction.

Check Your Progress 2

1.

Compiler	Inerpreter
Scans the entire program in a single go	Scans line by line
Displays all syntax errors	Stops on the first syntax error
Slow for debugging	Fast in debugging
Execution time is less	Execution time is high

2. (a) False (b) False (c) True (d) False (e) False

Check Your Progress 3

1. Looping statements are used to execute a set of instructions repeatedly.

2. Arrays are simple and efficient way of referring to and performing computation on data of similar type.

3. Mainly the number of values which are returned. Functions return only a single value.

3.7 FURTHER READINGS

1) How to solve it by computers, R.J. Dromey, Prentice Hall of India

2) Programming Languages, Pratt, Prentice Hall of India