UNIT 3 INTRODUCTION TO PROGRAMMING AND PROGRAMMING LANGUAGES

Structure

3.0 OBJECTIVES

After reading the Unit, you will be able to:

- understand the basics of programming;
- be acquainted with different categories of programming languages; and
- become familiar with types programming languages.

3.1 INTRODUCTION

A computer can neither think nor make decisions on its own. To solve a problem using computers, a set of instructions should be given to the computer in a form which it can understand. The process of writing these instructions is called programming.

Understanding what is programming can be explained by taking a simple and well

known example. Consider searching the name of an author from the catalogue in the library.

Manual solution to this problem is to flip through the catalogue for the initial letter of the name. Once found, search for the subsequent letters of the name and on finding the name, note the details of the book by the author.

Let us assume the catalogue is available in a computer. We need to write a program to solve this problem of searching a word in the dictionary. The algorithm to solve the problem is given below:

1) Take the name to be searched as an input

2) Match the names available in the catalogue one by one until the exact match is found. (It is simplest logic but we can apply more efficient logic)

3) Display the details of the book as output for the input name.

This solution is in natural language which computers cannot understand. Therefore, we need to write it in a language, as a set of instructions, which computers can understand. Such a set of instructions that are arranged in a sequence that guides the computer to solve a problem is called a program.

The process of writing a program is called Programming. It involves *solving a problem and writing it in a form which a computer system can understand and execute*. While considering how to solve a problem, there are three main specifications which should be taken into consideration:

- What information flows into the system?
- What information flows out of the system?
- What the system does with the information?

Programming is a multi-step process and includes the following five steps:

1) *Identify the programming needs*: Specify the program objectives and program users, and state the input, output and processing requirements.

2) *Design the program*: Design the details using pseudocode or flowcharts.

3) *Code the program:* Select the appropriate high level programming language and code the program in that language following its syntax.

4) *Test the program:* The fourth step is to test the program and "debug" it of errors so that it works properly.

5) *Document and maintain the program:* Written descriptions, procedures about a program and how to use it need to be developed.

Computers cannot understand natural languages such as English or Hindi for solving problems. To communicate instructions and commands we need programming languages. And learning programming languages requires learning the symbols, words and rules of the language. There are many languages available to do programming. These are developed for specific applications.

Based on these applications, programs are of two types- *system programs* and *application programs*

System program: These are the programs which makes computer easier to use.

Operating Systems such as DOS and Windows which consist of many other programs for controlling input/output devices, memory, processor, etc is system program. The *C language* is widely used to develop system software.

Application program: Application programs are designed for specific application such as library automation, payroll information retrieval, processing, inventory control, etc. These applications should be able to get input, produce output, do calculations and store and retrieve data. There are many languages which contains instructions to support all these operations.

Let us see the categories of these languages and their basics, and also, the advanced features of a language. C is the programming language considered through out this Unit.

3.2 CATEGORIES OF LANGUAGES

Programming can be done in any language such as C, C++, COBOL, etc., which largely depends according to 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 they can be executed by the computer.

Just as hardware is classified into generations based on technology, computer languages also have a generations classification based on the level of interaction with the machine. Since 1945, over the years, five categories or generations of programming languages have evolved over the years which are as follows.

- First generation, 1945 – *Machine language*
- Second generation, mid1950s – *Assembly language*
- Third generation, early 1960s – *High level languages*
- Fourth generation, early 1970s – *Very high-level languages*
- Fifth generation, early 1980s – *Natural languages*

Programs written for a computer may be either in one of the categories of languages.

3.2.1 Machine Language

This is a sequence of instructions written in the form of binary numbers consisting of 1s and 0s 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 operations 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. Machine language is considered to be the first generation language.

3.2.2 Assembly Language

When symbols (letters, digits or special characters) are used 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.

Since a computer can execute programs only in the machine language, a translator or converter is needed if the program is written in any other language. An assembler is such a language translator which takes its input in the form of an assembly language program and produces machine language code in some memory location.

3.2.3 High-level Language

High level languages are also called procedural languages. Programming languages such as C, COBOL, FORTRAN and BASIC are high level languages. As the time and cost for creating machine and assembly languages are high, the high level languages were developed.

A program written in a high level language should be translated into a form the machine can understand and this is done by a software (language translator) called compiler which takes the source code input and produces as output the machine language code of the machine on which it is to be executed. There is another kind of software which also does the translation. This is called an interpreter.

3.2.4 Fourth Generation Languages

Fourth Generation Language, referred to as 4GL is a high level language that requires significantly fewer instructions than a third generation language does to accomplish a particular task. Thus, the programmer should be able to write a program faster in 4GL than in the third generation language. Most Fourth Generation Languages are non-procedural languages. The programmer does not have to give the details of procedure in a program, but instead specifies what is wanted.

For example, assume you need to display the details of a particular book (*Introduction to Computers)* from the catalogue. 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 catalogue

Step 2: If this is the record on *Introduction to Computers,* display the details

Step 3: If this is not the record for *Introduction to Computers*, go to Step 1

In a 4GL, however, the programmer would write a single instruction that says:

Get the details of *Introduction to Computers* from the Catalogue

Majority of the fourth generation languages are used to get information from files and databases and 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.

Structured Query Language (SQL) is a standard language and universally used. Like most other computer languages like C, C++, Pascal, or Java, you need a way to use it on your computer. More details about the languages are given in Block 2.

Most database applications use the SQL, as their language. to communicate with the database, like add, delete, modify the data.

3.2.5 Natural Languages

Natural languages are of two types. The first are the ordinary human languages like English. The second are programming languages that use human language to give people more natural interaction with computers. Natural languages allow questions or commands to be framed in a more conversational way. Natural languages are the part of the field of study known as *artificial intelligence*. Artificial intelligence is a group of related technologies that attempt to develop machines capable of emulating human-like qualities, such as learning, reasoning, communicating, seeing, and hearing.

Self Check Exercise

1) Name some third generation languages, and mention what are they used for?

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of this unit.

.....

.....

.....

.....

.....

.....

3.3 BASIC FEATURES OF PROGRAMMING LANGUAGES

Learning a programming language requires learning the basic features and few advanced features for efficient programming. The basic features are data types and expressions. Advanced features include making subroutines, using data structures, etc. These features are briefly explained below

3.3.1 Basic Data Types

The data used in programs are stored in internal memory of the computer. In order to store or retrieve information from memory we need to give name of each location in the memory. These memory location names can be declared so that only a particular type of data will be stored. For example, we can store only whole numbers such as 50 or numbers with fractional part such as 345.570. This specification of types of data stored is done using some key words called data types.

In ‘C’ language, the basic data types are *integer, float* and *char*. Many variations on size leads for further classification on these basic data type like short int.

Int: A type of memory location which can hold only integer values, i.e. no fractional part. The precise range of integers which are supported varies from one version of C to another.

Float: A type of memory location which can hold a real (i.e. floating point) number. e.g. numbers with fractional digits.

Char: A type of memory location which can hold a single character. A character is what you get when you press a key on a keyboard.

Data types and storage capacity are language dependent. One needs to be aware of them while learning a language. There are two broad categories of data. These are called variables and constants.

Variables: This is a single or group of characters assigned by 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. It can be of any data type like integer, character, float.

Constants: Constants have fixed value. For example, p = 3.1444. All data types can have constant value assigned to them. Few examples are given below. Integers can have values like 2, 4, etc. Sequence of characters in a word say, "computer" are string constants (string is sequence of more than one characters). Operations such as addition, deletion, multiplication and division can be performed using constants.

The size and limit of data types in C, C++ and JAVA are given below.

In C

Data type	Number of Bytes	Minimum Value	Maximum value
Char	1	-128	127
Int	4	-32768	32767
Float	4	6 digits of precision	6 digits of precision
Unsigned char	1	0	(2^8)-1
Unsigned int	4	0	65535
Long int	4	-2^31	(2^31) -1
Unsigned long int	4	0	(2^32)-1
Short int	2	-32768	32767
Unsigned long int	4	0	(2^32)-1
Short int	2	-2^15	(2^15)-1
Unsigned short int	2	0	(2^16)-1
Double	8	12 digits of precision	12 digits of precision

In C++

Data type	Number of Bytes	Minimum Value	Maximum value
Char	1	-128	127
Int	4	-2^31	(2^31) -1
Float	4	6 digits of precision	6 digits of precision
Unsigned char	1	0	(2^8)-1
Unsigned int	4	0	(2^32)-1
Long int	4	-2^31	(2^31) -1
Unsigned long int	4	0	(2^32)-1
Short int	2	-2^15	(2^15)-1
Unsigned short int	2	0	(2^16)-1
Double	8	12 digits of precision	12 digits of precision
Long double	12	16 digits of precision	16 digits of precision
Bool	1	0	1

In JAVA

Data type	Number of Bytes	Minimum Value	Maximum value
char	2	Unicode 2.0	
byte	1	-128	127
short	2	-32768	32767
int	4	-2,147,483,648	+2,147,483,647
long	8		maximum of over 10^{18}
float	4		Maximum of over 10^{38}
double	8		Maximum of over 10^{308}
Boolean	1	0	1

In the above tables let us consider the C language table, In that the number of bytes column gives details of how many bytes a particular data type can occupy in memory as Char data type, value stored can occupy maximum 1 byte, like values stored in short integer data type cannot exceed 4 bytes. For example, consider three integer variables used in a program, then the maximum storage of the data is 3×4 = 12 bytes.

The minimum and maximum values gives us limit of actual value that can be stored in that particular data type. The range of the values stored is also limited according to the number of bytes. The value that will be stored in Integer data type should be between -32768 to 32767.

3.3.2 User Defined Data Types

In programming we deal with large amount of data. To represent each data element we have to consider them as separate variables. For example, we need process marks of student in an organization. There are 5 semesters, for which we need 5 different variables. Instead if we can use only one name to refer all the five locations then it simplifies the program. This can be done by the user by defining a data type of his own. The basic and simple user defined data type is called array.

Array is a collection of same type of data, all of which are referenced by the same name. Generally used arrays are one dimensional or two dimensional.

One Dimensional Array: It is a list of related data of same type referred by one variable name. For example marks of a student can be referred as follows

Semester(1)	Semester(2)	Semester(3)	Semester(4)	Semester(5)
65	70	82	68	83

Semester(1) specifies the marks obtained in first semester and so on.

Two dimensional array: In a single dimensional array the data of array is dependent upon a single factor such as marks in the semesters. In some cases, we need array which is dependent upon two factors, such marks of five semesters for a particular student who obtained the marks. In this case we use two dimensional arrays which are represented in tabular form (in rows and columns) as shown below. Rows represents the student name and columns the semester.

	Sem 1	Sem2	Sem3	Sem4	Sem5
Bharat	60	73	68	84	53
Jagjit	75	91	82	63	70
Rajan	86	54	75	78	85

3.3.3 Statements

We have seen data types to store values in memory location. Statements are those which manipulate these data using predefined structure. There may be many types of statements, which are described below.

Input/Output Statements

The function of input statement is to get value of input from user through some input device like keyboard. Suppose, we need to get book title for searching the availability of a book, it looks like

scanf("%s", BookTitle);

The format and rule of writing this statement is predefined.

Similarly, if we need to get radius of a circle for calculation of its circumference, it looks like

Scan("%f", & Radius)

Now after obtaining the availability we need to display the output. This is done by the following statement.

printf("%s",BookTitle);

printf("%f", circumference);

Here "Circumference" is the variable in which the output is stored.

These input, output statements are provided as built in library along with the language. In C we need to specify the library file which is called header file in the beginning of the program. The header file for input, output statements in C is "stdio.h". To specify this, you must write include <stdio.h> in the beginning of the program.

Depending on the language the input, output statement varies, for example in BASIC input would be written as:

INPUT Radius

Arithmetic Expressions

We can do arithmetic operations using expressions like A+B, A-B, A*B, A/B and so on. Several simple expressions can even be nested together using parentheses to form complex expressions. Every computer language specifies the order in which various arithmetic operators are evaluated in an expression. An expression may contain operators such as

Parentheses ( )

e.g. Total_Salary = (Basic_Salary + Allowances) – Detections

Total_Salary = (3500 + 1500) -675

Exponentiation ^

Exponentiation is used to find out exponent of any value

e.g. Expo = 3.0098 ^ 3

Negotiation ~

Negotiation is used in Boolean operations to find out inversion of any value

e.g. Invert = ~Value

Invert = ~1

Multiplication, Division *, /

e.g. Salary = NoofDays * Salary_Per_Day

Salary = 27 * 650

Addition, Substractions +, -

e.g. Total_Salary = (Basic_Salary + Allowances) – Detections

Total_Salary = (3500 + 1500) -675

The operators are evaluated in the order given above.

Assignment statement

An assignment statement assigns a value to a program variable. This is accomplished by evaluating some expression and then writing the resulting value in the memory location referenced by the program variable for numerical values and directly assigning the value to string variables. In C it looks like:

Variable = assignment;

e.g AuthorName = “Guha”;

C = A*B+100;

If you assign a value to a variable it looses its previous value and the new value remains in the memory till next assignment is made on the same variable.

Control Statements

In any language sequential flow of the statement is default i.e; a program executes instructions sequentially from first to last. Apart from sequential there are two types of flow of execution available. These are :

Conditional - In this the choice of which instruction to execute next depends on some condition

Looping – In this a group of instructions may be executed many times

In *conditional*, flow of control depends on the result of a Boolean (true/false) condition that decides the next statement to be executed. For example the condition of comparing two variables whether they are equal or greater, or less than each other, the flow is changed. In general the structure will look like

if (Boolean condition)

S1

else

S2

If-then-else allows an empty else case also. For some cases we need few statements to be executed if condition becomes true, else we want to continue the actual flow i.e. nothing is to be done in the else part. In general

if (Boolean condition)

S1

In *looping* it is not just the next statement to be executed that is decided, but a block of statements called loop body which is repeated for the number of times depending upon the result of a Boolean condition. The condition is tested before executing the loop body and when the condition becomes false the loop body will not be executed again. In general;

while (Boolean condition)

S1

3.3.4 Program Structure

After knowing various elements of programming language and their use, we should write them out in a structured manner while writing a complete program. The points given below gives us the order in which the elements are to be used.

- Comment to explain what program does
- Necessary header files (stdio.h is minimal requirement for any program as input, output will be there)
- The start of main function
- Declaration for variables
- Main function body

The C program structure is shown below

Comment (optional)

Include directive (optional)

Main function

{

declarations (optional)

```
        main function body
    }
```

An example of the above is given below:

```
// Program to compare two values and printer the greatest among them.

#include <stdio.h>

main()

{
    int A , B;
    A = 10;
    B = 15;
    if (A>B)
        printf("The greatest value is %d",A)
    else
        printf("The greatest value is %d",B
}
```

3.3.5 Modules

Solution to complex problem needs complex and large programs. If we write the entire thing in a single main function it may be very lengthy and difficult to understand. Changes in the code in future are not easy. So breaking the program into many sub tasks and design proper communication between them will help solve the problem of complex programs more easily.

The sub task or part of code to do a specific task is called function. If we use functions, the main program can just need to invoke these functions. Calculating circumference, area computation of a circle problem can have three functions to get an input, compute circumference, and compute area. This can be illustrated as follows:

```
Get value for user's choice about continuing
While the user wants to continue
        Do the input subtask
        If (Task = 'C') then
                do the circumference subtask
        else
                do the area subtask
Get the user's choice about continuing
```

Once the execution of main is over the execution returns back to main function

The general structure of a function is given below

```
function header
{
```

```
        local declarations              (optional)

        function body

}
```

For example, consider the function swaps two values by receiving them from main function

```
main()

{

        int A = 10, B=15;

        swap(10,15);

}

Swap (int A, int B)

{

        int Temp;

        Temp = b;

        b = a;

        a = Temp;

}
```

The function header contains three parts

- Return type
- Function name
- Parameter list

Return type is the type of variable it is given as a result to the calling function, for example, the main function. If the function is not returning any value like, it simply prints some message then the function's return type is void.

Function name can be any user defined name (Don't use keywords of language as function name)

The parameter list is a list of variables given by main function as an input to the sub function. The list and order of parameter in the main function and parameter list in function should match. The data type is given as part of the list of parameters.

Consider a program to calculate Area and Circumference of a circle The main function should pass the radius to sub functions to calculate Area and Circumference. The sub functions and main functions for Area and Circumference calculation of circle is given below:

```
#include <stdio.h>

void main( )
```

```
{
        float  Radius;
        char  Task_to_do;
        Get_Input(Radius, "C");
}
/* Function to get an input of radius and task to be done */
void Get_Input(float Radius, char Task_to_do)
{
        printf("Enter Radius value  ");
        scanf("%f",&Radius);
        if (Task_to_do  = = 'C')
                Do_Circumference(Radius);
        Else
                Do_Area(Radius);
}
/* Function to compute circumference*/
void Do_Circumference(float Radius)
{
        float Circumference;
;       Circumference = 2*3.144*Radius;
        printf("Circumference  is %f",Circumference);
}
/* Function to compute Area*/
void Do_Area(float Radius)
{
        float Area;
;       Area = 3.144*Radius*Radius;
        printf("Area is  %f",Area);
}
```

Here the return type of the functions Do_Area and Do_Circum- ference are void. The parameters in the functions are passed as their values from main function. This type of parameter passing is called *passing by values*. That is the copy of the

radius value is passed to them. But they are aware of the memory location where the original value is stored.

If the main function wants to change the original value then instead of sending the copy of the value present in the memory location it will send reference of that memory location itself. The changed value will be stored in the memory location by the sub function. This is called *pass by reference*.

The parameters should be passed by value if the function doesn't need to change the values instead just use them for some operations. If the change should reflect in the memory of that variable then pass by reference is preferred. The clarity in the choice of pass by value and pass by reference is efficiency in modularity.

Self Check Exercise

2) Declare an integer array with three columns and three rows and what will be the size of total array in bytes if the integer takes four bytes of memory?

3) Compare two values (15, 7) and give the higher value as an output? Write a program.

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of this unit.

.....

.....

.....

.....

.....

.....

3.4 TYPES OF PROGRAMMING LANGUAGES

We have learnt in the previous sections that programming is writing a set of instructions. If it is as simple as that, then why can't we use one programming language? Programming languages are designed to meet specific needs. Consequently, one language may be better suited than others for writing certain kinds of functions. There are programming languages that are used for solving engineering problems, writing out complicated sales reports, interact with databases, manipulate graphics or even design web pages. So, with proliferation of programming tasks, proliferation of programming languages have also taken place. Let us see the various families of programming languages and the types of programming languages that belong to them.

3.4.1 Procedural Languages

Languages in which programs are written, as sequence of statements that manipulate data item and change the contents of memory cells, are called procedural languages (also called imperative languages).

The fundamental operation of these languages are storing and retrieving data values. For example

a = 1

This statement stores value 1 in location a

a, c = a + b

This statement retrieves *a* and *b*, add and stores the result in *c*. Some of the procedural languages are discussed below.

FORTRAN

The name FORTRAN derives from *FORmula TRANslation*. The name itself indicates that the applications of language with "formula" of engineering-type application. The first commercial version of FORTRAN released in 1957. This makes FORTRAN the first high-level programming language. FORTRAN has some features ideally suitable for problems that are heavily mathematical or computational oriented.

COBOL

The name COBOL derives from *Common Business Oriented Language*. COBOL was designed to serve business needs such as managing inventories and payrolls. In such applications summary reports are important output. Much of the business world concerns updating master files with change in transaction files. For example master file contain names, manufacturers, and quantities available for various items inventory; a transaction file would contain names, quantities and items sold out of inventory over a period of time. The master file would be updated from the transaction file on weekly or daily basis to reflect new quantities available and print summary report.

PASCAL

The programming language Pascal was named after Blaise Pascal, the inventor of Pascaline calculator.

The design of Pascal was to easily learn and enforce good programming techniques. Pascal looks very similar to pseudocode. So it is easy to read and the syntax is easy to learn. Even though Pascal is not much useful as commercial language but, programming environment like Delphi uses Pascal language with a facility to develop windows based applications with modern graphical user interface.

C

C was developed in early 1970s by Dennis Ritchie at AT&T Bell Laboratories. It was originally developed for system programming, particularly for writing operating system UNIX. It is a popular general purpose language for two reasons. One is the relationship with UNIX. Second reason is its efficiency that is, speed with which its operations can be executed. The efficiency derives from the fact that C programs can make use of low-level information such as knowledge, where data is stored in memory, yet has powerful statements and portability to machines that high level languages offer.

The strong feature of C is that it provides a data type called pointer. Variables of pointer type contains memory addresses instead of integers, real numbers and characters. For example the statement,

```
int*  intPointer;
```

declares *intPointer* as a pointer variable that will contain the address of a memory location containing integer data.

```
int A = 3;

intPointer = & A.
```

The first statement declares a variable called A and assigns the value 3 to that. The second statement makes the intPointer pointing to A. We can also assign the integer value 10 to variable A by the statement

```
*intPointer = 10;
```

The powerful use of pointers are in the areas of writing system programming, operating systems, assemblers, and programs that allows the computer to interact with input/output devices and so on.

For example consider a problem of writing a device driver (software to interact with a hardware device) for mouse on a PC. The port in which the mouse is connected reads the changes in mouse position by voltage levels. It stores voltage levels in a fixed location of memory. C provides facilities to access this memory and manipulate it.

Even though C is powerful for system programming it is widely used for general purpose programming also.

ADA

Ada development was for a common high-level language needed for various branches of United States armed services for defense contracts. Ada is a large language like C++ and has been accepted not only in the defense industry, but for other technical applications and is a general purpose language as well.

Ada is known for its multi-processing capability – the ability to allow multiple tasks to execute independently and then synchronize the communication when directed.

3.4.2 Object Oriented Programming Languages

Object Oriented Programming started by Alan Kay's work at Xerox Palo Alto Research Centre in early 1970s. The resulted language is smalltalk.

The concept of Object-oriented programming is dividing the major task into sub tasks. The program can be thought of a giant statement executor designed to carry out major tasks, even though the main program may simply call the various modules that do the sub task work.

To understand objects, let us take an example of library system. Circulation, Acquisition, Catalogue, etc are major objects. Each object is an example of a task performed in library system.

3.4.3 Special Purpose Languages

The procedural languages that we saw so far are more or less general purpose languages. There are several special purpose languages designed for specialized tasks. Here we take three representative special purpose languages, which are very popularly used.

Structured Query Language (SQL)

SQL is designed to be used with databases, which are collections of related facts and information. A database stores data; the user of the database must be able to add new data and to retrieve data already stored. For example, a database of library catalogue may contain titles of book, its author's name, publisher name, year of publication, etc. The database of catalogue user should be able to add information on a book and to retrieve information on a book already in the database. In addition, databases can also be queried with the user posing questions to the database. For

example, the library catalogue could be queried to reveal books by a particular author or by a publisher. Such queries may be framed in SQL. SQL is the language used to frame database queries. Sample SQL statement is as follows:

```
SELECT BOOK_TITLE, PUBLISHER, YEAR

FROM CATALOGUE

WHERE AUTHOR NAME = "GUHA";
```

This SQL statement will retrieve all the book titles with publisher and year details of the author Guha.

Examples of prominent database management systems that use SQL are Visual FoxPro, Oracle, MS-Access and SQL Server. Visual FoxPro is a commonly used database management system in libraries. For example, the library catalogue can be created in Visual FoxPro and the SQL can be used to query the database by name of the author, title of the book, keyword, etc.

Practical Extraction and Report Language (PERL)

Perl is designed to scan arbitrary text files, extract various kinds of information that is contained within the text, and print reports based on the extracted information. The language syntax is somewhat based on C. Perl uses sophisticated pattern matching techniques to speed up the process of scanning large amounts of data for a particular text string.

Hyper Text Markup Language (HTML)

This is the language used to create HTML documents that, when viewed with Web browser software, become Web pages. An HTML document consists of the text to be displayed on the Web page, together with a number of special characters called tags that achieve formatting, special effects, and references to other HTML documents. Tags are enclosed in angle brackets (<>) and often come in pairs. The end tag, the second tag in the pair, looks like the begin tag, the first tag in the pair, but with an additional / in front.

The format for an HTML document is

```
<html>

<head>

<title> text that is appear as title </title>

</head>

<body>

        text that is to appear as body on the page

</body

</html>
```

Self Check Exercise

4) If Mark is an integer variable and intPointer is a pointer to Mark and if value stored in Mark is 90 then what is the value of *intPointer.

5) Write an SQL statement to display the books issued from the library for a member named Amit.

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of this unit.

.....

.....

.....

.....

.....

.....

3.5 SUMMARY

In this Unit, we have dealt with programming and programming languages. The focus of attention in this Unit are the following:

- Concept of programming and programming languages;
- Machine, assembly and high level languages;
- Features of programming languages like data types, statements, program structure;
- Common programming languages and uses.

3.6 ANSWERS TO SELF CHECK EXCERCISES

1) The high level such as C, COBOL, FORTRAN and BASIC are third generation languages. C is used for scientific as well as system programming, COBOL is a business oriented language used for systems like payroll and inventory and FORTRAN is used for formula translation in mathematical and scientific applications.

2) int arr [3][3];
The total array size will be 36 bytes

3) int a = 15, b = 7;

```
if a > b
      printf("%d",a);
else
      printf("%d",b);
```

4) The value of *intPointer is 90

5) SELECT BOOK_ISSUED
FROM CIRCULATION_DATA
WHERE BORROWER_NAME = "Amit"

3.7 KEYWORDS

Application program : Program that has been developed to solve a particular problem, perform useful work on general-purpose tasks, or provide entertainment

Assembly language : It is a low level programming language that allows a computer user to write a program using abbreviations or more easily remembered words instead of numbers.

Compiler : A language translator that converts the entire program of a high-level language into machine language before the computer executes the program.

Floating points : A method for storing and calculating numbers in which the decimal points don't line up as in fixed point numbers.

Flow Charts : A graphical representation of the sequence of operations in an information system or program.

High level language : Also known as the procedural language it resembles human languages such as English.

Interpreter : A language translator that converts each procedural language statement into machine language and executes it immediately.

Machine language : A binary type low level language consisting of 1s and 0s that the computer can run directly.

Procedural language : A programming language that requires programming discipline, such as COBOL, FORTRAN, BASIC, etc.

Programming language : A language used to write instructions for the computer.

Pseudocode : A notation for writing algorithms using normal human-language statements to describe the logic and processing flow.

System program : Program that helps the computer to perm essential operating tasks and enables the application program to run.

3.8 REFERENCES AND FURTHER READING

Williams, B.K. and Sawyer, S.C. (2003). *Using Information Technology : A Practical Introduction to Computers and Communications* (5th Edition). New Delhi: Tata McGraw-Hill Publishing Company Limited.

Pratt, T.W. and Zelkowitz, M.V.(2003). *Programming Languages: Design and Implementation*(4th ed). New Delhi: Pearson Education (Singapore) Pte Ltd

Schneider, G.M. and Gersting, G.L. (1998). *An Invitation to Computer Science* (2nd ed). California: Brooks/Cole Publishing Company.