

Programming Languages: Low Level, machine language, Assembly, High-level Language (HLL); Algorithm and Flow chart: Concept and Uses	
Prepared by BIKRAM KISHORE BEURA	
Subject Name	Library and Information Science
Paper Name	Programming Languages
Module Name/ Title	Programming Languages
Module Id	LIS/ M-6
Pre-requisites	Basic computer knowledge including hardware, software and internet.
Objectives	To study the need and objectives of programming languages including language translators such as Assembler, Interpreter, Compiler and different levels of programming languages namely Machine language, Assembly language and High Level Languages. It also describes the concept and use of Algorithm and Flowchart in computer programming.
Keywords	Programming Language, Low level Language, Machine Language, Assembly Language, High Level Language, Language Translator, Algorithm, Flowchart, Computer Programming

LIS/M-6: Programming Languages: Low Level, machine language, Assembly, High Level Language (HLL); Algorithm and Flow chart: Concept and Uses

INTRODUCTION

The uses of computers are almost limitless in our everyday lives. Computers can do such a wide variety of things because they can be programmed. Ever since the invention of Charles Babbage's difference engine in 1822, computers have required a means of instructing them to perform a specific task. This means the specification of the computation can be stored in the computer as a program and it is possible to use that program to control the execution of a computer. A program is a set of instructions that when placed in the computer's memory the computer follows these instructions to perform a task.

A set of related programs are commonly referred to as software. Software is essential to a computer because it controls everything the computer does. All of the software that we use to make our computers useful is created by individuals working as programmers or software developers. A

programmer, or software developer is a person with the training and skills necessary to design, create, and test computer programs.

Computer programs tell a computer how to interact with the user, interact with the computer hardware and process data. To write a program for a computer, we must use a programming language. Programming languages act as a system of communication between the computer and computer user that can be used to develop programs. It permits people to communicate with computers by writing instructions in a way that is easier to learn and understand. Programming languages were developed with one simple goal in mind that to facilitate people to use the computers without the need to know in detail, the internal structure of the computer.

WHAT IS PROGRAMMING LANGUAGE?

A program is a sequence of symbols that specifies a computation. A programming language is a set of rules that specify which sequences of symbols constitute a program, and what computation the program describes. Programming languages allow people to give instructions to a computer with commands that both the computer and the programmer can understand. Different programming languages use different commands and different rules for entering those commands.

A programming language is an artificial language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs that control the behaviour of a machine and/or to express algorithms. Any programming language is composed of a set of predefined words that are combined according to predefined rules (syntax) to generate a computer program.

HISTORY OF PROGRAMMING LANGUAGES

The computer programmers are writing codes, since last few decades. New technologies continue to emerge, develop and mature at a rapid pace. In the history of programming languages, thousands of different programming languages have already been created, and still many are being created every year.

The history of programming language can be traced back to the development of Charles Babbage's difference engine, which could only be made to execute tasks by changing the gears which executed the calculations. Thus, the earliest form of a computer language was physical motion. Eventually, physical motion was replaced by electrical signals when the US Government built the ENIAC in 1942.

In 1954 IBM established a project directed by John W. Backus to develop a compiler for the model 704 computer. This project resulted in the creation of the language FORTRAN (FORmulae

TRANslation) that IBM finished in April 1957. The language had a notation orientation to mathematicians and scientists. Though FORTAN was good at handling numbers, it was not so good at handling input and output, which mattered most to business computing.

In 1958, John McCarthy of MIT created the LISt Processing (or LISP) language. It was designed for Artificial Intelligence (AI) research. In 1959 a new language has been developed named COBOL (Common Business Oriented Language) for business data processing.

The Algol language was created by a committee for scientific use in 1960. Although Algol was the first language with a formal grammar, its next version Algol68 became bloated and very difficult to use for general purpose programming. These problems later on lead to the adoption and development of many smaller and more compact languages, such as Pascal, C, C++ and Java.

In the mean time, Dr. John G. Kemeny and Thomas Kurtz developed the BASIC language at Dartmouth in 1964 with an objective to create a simplified computer language for teaching students how to program. BASIC stood for "Beginner's All-Purpose Symbolic Instruction Code".

In the year 1966, IBM developed a common programming language with multi-tasking feature to meet the requirements of both scientific, commercial users, known as PL/I (Programming Language/I). The pronunciation is Programming Language One.

The development of Pascal language by Niklaus Wirth in 1968 was mainly out of necessity for a good teaching tool for debugging and editing system and support for common early microprocessor machines which were in use in teaching institutions. This language was designed in a very orderly approach, by combining many of the best features of the languages used in that time like COBOL, FORTRAN, and ALGOL.

Dennis M. Ritchie created the 'C' language at AT&T's Bell Laboratories in 1972. This language includes all the features of Pascal with some additional features such as use of CASE statements and pointers. Ritchie developed C for the new UNIX system being created at the same time. Later on C become the most common language to program operating systems such as UNIX, Windows, the MacOS, and Linux.

In early 1980's, a new programming method known as Object Oriented Programming (OOP) with an idea of representing pieces of data on form of 'Objects' that can be packaged and manipulated by the programmer. Based on this idea Bjarne Stroustrup in 1983 extended the features of 'C' language, to develop a new full-featured language known as 'C++'.

Microsoft released Visual Basic for Windows in May 1991. It is a graphical version of BASIC that simplifies the writing of program for windows. In 1995 the Java language hit the scene, rapidly rose to popularity and is widely used truly object oriented programming language in existing today.

In 2000's the expansion of World Wide Web lead to the demand for internet programming. As a result many programming languages such as Perl, PHP, and Python were developed and used by the programmers to develop web applications.

LEVELS OF PROGRAMMING LANGUAGES

People express themselves using a language that has many words. But Computers understands only a simple language that consists of 1s and 0s, with a 1 meaning "on" and a 0 meaning "off." This is because electrical devices seem to fall naturally into one of two possible states s. For example; they are either on or off; they are magnetized in one direction or the other; they are conducting electricity or they are not conducting electricity. Hence, the instructions coded in to computer's memory are configured as 0's and 1's (binary notation).

As the computer operates using a program coded in 0's and 1's, it would be very time consuming for a person to write a program in 0's and 1's. If the program can be written in a language approaching that of English or mathematics or a combination of both, the programmer can concentrate more on programming logic and less on programming details. Thus the program should be less likely to contain errors. The effort to overcome this difficulty has lead to the development of new programming languages that can be well suited to the capabilities of both.

Over the years, computer languages have evolved from machine language (computer's native language) to high-level languages. Also computer programs written in the earliest programming languages were based on the underlying internal structure of the computer. The high level languages facilitate programmers rather than learning the machine language, they can instruct the computer in a way that is easier to learn and understand. Language that favour humans are terms as high level, and those oriented to machines are low level.

Presently, programming involves two following major level of programming languages; Low level Language and High level language. The binary machine language is usually defined as the lowest level, whereas the highest level might be human language such as English.

1. Lower Level Languages (LLL)

Low-level language is a programming language that deals with a computer's hardware components and constraints. It has no or a minute level of abstraction in reference to a computer and works to

manage a computer's operational semantics. Low-level languages are designed to operate and handle the entire hardware and instructions set architecture of a computer directly.

Low-level languages can be converted to machine code without using a compiler or interpreter, and the resulting code runs directly on the processor. A program written in a low-level language can be made to run very quickly, and with a very small memory footprint; an equivalent program in a high-level language will be more heavyweight. Low-level languages are simple, but are considered difficult to use, due to the numerous technical details which must be remembered.

By comparison, a high-level programming language isolates the execution semantics of computer architecture from the specification of the program, which simplifies development.

Low-level programming languages are sometimes divided into two categories: Machine Language and Assembly language

- **Machine Language:** Machine language is the sequence of bits (machine code) that directly controls a processor, causing it to add, compare, move data from one place to another, and so forth at appropriate times. The computer microprocessor can process directly the machine codes without a previous transformation. Specifying programs at this level of detail is an enormously tedious task. Currently, programmers almost never write programs directly in machine code, because it requires attention to numerous details which a high-level language would handle automatically, and also requires memorizing or looking up numerical codes for every instruction that is used.
- **Assembly language:** Assembly language uses structured commands (mnemonics) as substitutions for numbers allowing humans to read the code easier than looking at binary. Although easier to read than binary, assembly language is a difficult language and is usually substituted for a higher language such as C. The problem with assembly language is that it requires a high level of technical knowledge, and it's slow to write.

Typically, one machine instruction is represented as one line of assembly code. Assemblers produce object files which may be linked with other object files or loaded on their own. Most assemblers provide macros.

2. High Level Languages (HLL)

At present, High Level languages have replaced machine and assembly language in all areas of programming. Programming languages were designed to be high level if it is independent of the underlying machine. High-level languages (also known as problem-oriented languages) enable a programmer to write programs that are more or less independent of a particular type of computer.

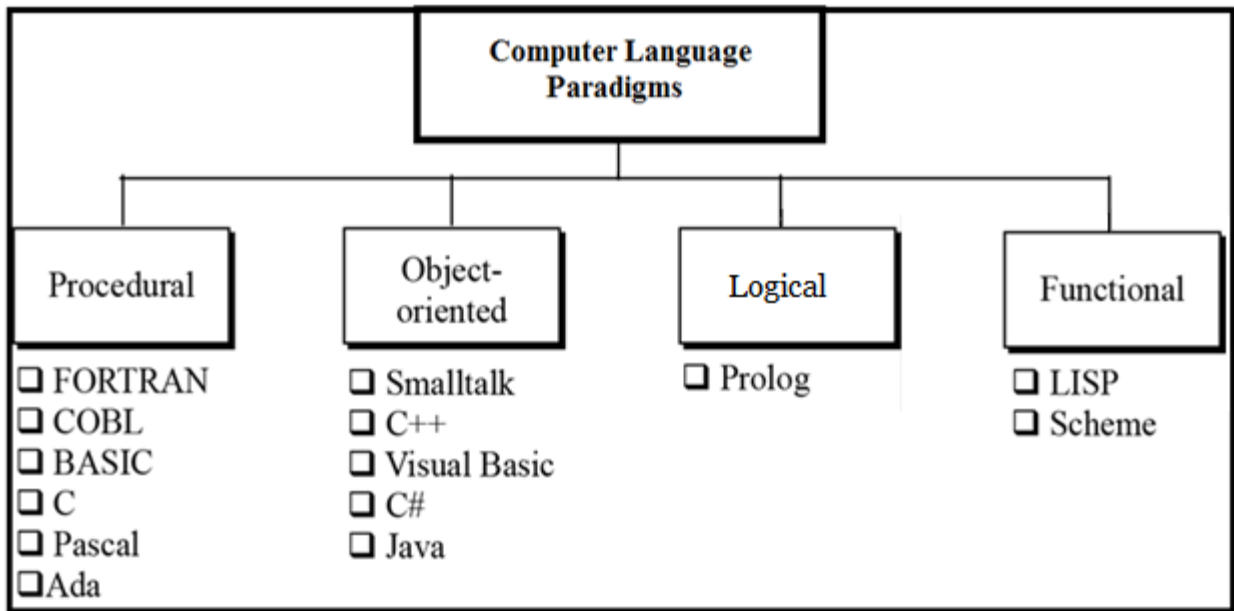
Such languages are considered high-level because they are closer to human languages and farther from machine languages.

High level languages are portable (machine independent) as it can be run on different machines with little or no change. Furthermore, the rules for programming in a particular high-level language are much the same for all computers, so that a program written for one computer can generally be run on many different computers with little or no alteration. Thus, we see that a high-level language offers three significant advantages over machine language: simplicity, uniformity and portability. Higher-level languages provide a richer set of instructions and support, making the programmer's life even easier. The languages such as BASIC, COBOL, FORTRAN, C, C++, JAVA and Visual Basic are popular examples of high level languages. High level languages use translator programs such as compiler and interpreter to convert it into a machine language program.

Computer languages were first composed of a series of steps to wire a particular program; these morphed into a series of steps keyed into the computer and then executed; later these languages acquired advanced features such as logical branching and object orientation. Many programming languages require computation to be specified in an imperative form (i.e., as a sequence of operations to perform), while other languages utilize other forms of program specification such as the declarative form (i.e., the desired result is specified, not how to achieve it). Based on this programming paradigms computer languages are classified into four main categories as follows.

- **Procedural languages:** These languages uses a programming approach, where a developer writes code that describes in exacting detail the steps that the computer must take to accomplish the goal.
- **Object oriented languages:** These languages use a programming paradigm that represents concepts as “objects” that have data fields and associated procedures known as methods. Objects, which are instances of classes, are used to interact with one another to design applications and computer programs.
- **Logical Languages:** The programming paradigm in these languages is based on formal logic. A program written in a logic programming language is a set of sentences in logical form, expressing facts and rules about some problem domain.
- **Functional languages:** This paradigm was explicitly created to support a pure functional approach to problem solving. The programming approach in these languages involves composing the problem as a set of functions to be executed. These functions are predefined blocks of code intended to behave like mathematical functions.

The detailed examples of these languages are displayed in the following diagram.



LANGUAGE TRANSLATORS

Translator is meant to translate one language to another. So a translator is mainly related to computer language. In case of a computer, its hardware part only can operate when instructions are made of 0s and 1s, i.e., in machine language. But it is not that easy for human being to remember them correctly. Thus a move towards substituting such instructions of machine code by letter symbol-mnemonics was taken. Mnemonic codes ease programmer to write efficient program. But an intermediary agent is required to translate these mnemonic codes into machine codes. There are different types of translators for different categories of languages Software viz. assembler for assembly language, interpreter and compiler for high-level language.

1. Assembler

At one time, the computer programmer had at his disposal a basic machine that interpreted, through hardware, certain fundamental instructions. He would program this computer by writing a series of 1s and 0s (machine language), place them into the memory of the machine, and press a button, where upon the computer would start to interpret them as instructions

Programmers found it difficult to write or read programs in a machine language. In their quest for a more convenient language they began to use a mnemonic (symbol) for each machine instruction, which they would subsequently translate into machine language. Such a mnemonic machine language is now called an assembly language. Translator Programs known as assemblers were written to automate the translation of assembly language into machine language. As the assembly language is machine dependent, assembler is also a machine dependent system program which must be supplied by the computer manufacturer.

Hence, an assembler program translates source program in assembly language into object program in machine language.

2. Interpreter

An interpreter is a program which executes a source program usually on a step by step, line by line, or unit by unit basis. In other words an interpreter will usually execute the smallest possible meaningful unit in the programming language. The output of an interpreter is an actual answer, i.e. the result of performing the actions designated in the program.

The greatest disadvantage of an interpreter is that certain phases of work and analysis must be done repeatedly. In particular the scan of a statement which is to be executed for varying values of a particular parameter must take place each time that a new value is to be used. This contrasts with the compiler, which performs this translation function only once. On the other hand, the disadvantage to the compiler is that it does not produce answers; as soon as a change in the program is made a recompilation must be made.

3. Compiler

A compiler is simply a program which translates a source program written in a particular programming language to an object program which is capable of being run on a particular computer. The compiler is therefore both language and machine dependent. The most important characteristic of a compiler is that its output is a program in some form or another and not an answer of any kind. A compiler must perform at least the following functions: analysis of source code, retrieval of appropriate subroutines from a library, storage allocation, and creation of actual machine code.

The compilation of programs is a complex process and consists of broadly two phases – analysis phase and synthesis phase. In the analysis phase, the source program is analyzed for its lexical, syntactical and semantic structure. The second phase is the synthesis of the object program in the machine language. The compiler checks all kinds of limits, ranges, errors, etc.

Once the result of checking is satisfactory and error free, it produces a complete machine language program of minimized length ready for execution. This executable program, stored as a separate file, can be transferred along with a storage media like floppy, CDROM from one machine to another where it can be executed without the presence of the compiler at all. A compiler takes more time to compile than to interpret, but a compiled program runs much faster than an interpreted program.

ALGORITHM & FLOW CHART

To cope with a problem we need to make a better plan which will be convenient and easy to solve that particular problem. It is not that we cannot tackle with that problem but the thing is that we may not be successful. So, in computer programming too we need to plan before designing any problem. Algorithm and flowchart are the two basic terms which aids for the development of a software package conveniently.

1. Algorithms

An algorithm is a step wise set of finite instructions written to solve a problem. It will be easier to code a program after we can have well prepared algorithm and flowchart. An algorithm is written on simple language and must be efficient and limited to finite number of steps.

An algorithm is a finite sequence of steps expressed for solving a problem. An algorithm can be defined as “a process that performs some sequence of operations in order to solve a given problem”. Algorithms are used for calculation, data processing, and many other fields.

In computing, algorithms are essential because they serve as the systematic procedures those computers require. A good algorithm is like using the right tool in the workshop. It does the job with the right amount of effort. There are often many different algorithms to accomplish any given task. The reasons for using algorithms in programming are efficiency (time and cost factor), abstraction (simpler) and reusability (reusable).

The use of algorithms provides a number of benefits. One of these benefits is in the development of the procedure itself, which involves identification of the processes, major decision points, and variables necessary to solve the problem.

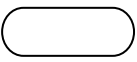
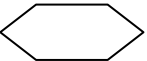
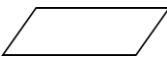
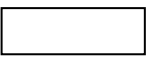




2. Flowchart

Flowchart is the diagrammatic representation of an algorithm with the help of symbols carrying certain meaning. Using flowchart, we can easily understand a program. Flowchart is not language specific. We can use the same flowchart to code a program using different programming languages. Though designing a flowchart helps the coding easier, the designing of flowchart is not a simple task and is time consuming.

A Flowchart is a type of diagram (graphical or symbolic) that represents an algorithm or process. Each step in the process is represented by a different symbol and contains a short description of the process step. The flow chart symbols are linked together with arrows showing the process flow direction. A flowchart typically shows the flow of data in a process, detailing the operations/steps in a pictorial format which is easier to understand than reading it in a textual format.

A flowchart describes what operations (and in what sequence) are required to solve a given problem. A programmer prefers to draw a flowchart prior to writing a computer program.

The purpose of all flow charts is to communicate how a process works or should work without any technical or group specific jargon. The flowchart plays a vital role in the programming of a problem and is quite helpful in understanding the logic of complicated and lengthy problems. Once the flowchart is drawn, it becomes easy to write the program in any high level language. The following table shows the details of the various flowcharting symbols used for developing flowcharts.

Flowcharting Symbols		
Symbol	Name	Description
	Terminal	Defines the starting and ending point of a flowchart
	Initialization	Initialization of memory space for data processing.
	Input/output	Input of data for processing and printing of processed data.
	Process	Manipulation of data (assignments and mathematical computations)
	Flow lines	Defines logical sequence of the program.
	On-page connector	Create a cross-reference from one process to another on the same page of the flow chart
	Off-page connector	Create a cross-reference from a process on one page to a process on another page.
	Decision	Process conditions using relational operators. Used for trapping and filtering data.

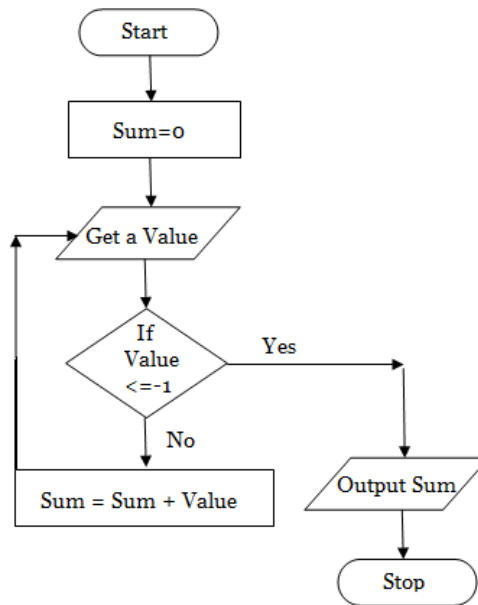
3. Examples of Algorithm and Flowchart

The concept of algorithm and flowchart can be understood in a better way through the following example. A very simple example of an algorithm would be to add a list of positive integers to find out the total sum. The algorithm for this problem can be described as follows:

1. Start
2. Sum = 0
3. Get a value
4. If the value is equal to -1, go to step 7
5. Add to sum (sum= sum + value)

6. Go to step 3 to get next Value
7. Output the sum
8. Stop

The corresponding flowchart of the above algorithm is displayed in the following figure.



CONCLUSION

Programming is a part of software engineering used for producing a program- a list of instructions for the computer. Programmers use special programming languages to create program's code. The instructions in a program consist of statements written in a programming language. A programming language is a set of rules that tells the computer what operation to do during the execution of programs. Based on consideration such as what purpose the program is designed to serve and what a languages are already being used in the organisations, the programmer selects programming languages for development of software. The evolution of programming language continues, in both industry and research based on the requirements of current web technologies including mobile applications. Current focus in programming language is Increasing support for functional programming, constructs to support concurrent and distributed programming, Mechanisms for adding security and reliability verification to the language: extended static checking, dependent typing, information flow control, static thread safety, Open source as a developmental philosophy for languages, including the GNU compiler collection and recent languages such as PHP, Python, Ruby, and Squeak etc.

CHUNK TEXT

1. Computers require a set of instructions and data to be stored in their memory to perform a specific task. This stored program concept was originated ever since the invention of Charles Babbage's difference engine in 1822.
2. Computer understands only the binary system consists of only 1s and 0s, with a 1 meaning "on" and a 0 meaning "off" known as machine language. Machine language is the native language of the computer.
3. Rather than learning machine language, one can use a programming language to instruct the computer in a way that is easier to learn and understand.
4. A programming language facilitates easier communication between the computer and computer user.
5. A programming language is an artificial language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs that control the behavior of a machine and/or to express algorithms.
6. A programming language is a system of notation for describing computations. A useful programming language must therefore be suited both for describing (i.e. for human writers and readers of programs), and for computation (i.e. for efficient implementation on computers). But human beings and computers are so different that it is difficult to find notational devices that are well suited to the capabilities of both. Language that favour humans are terms as high level, and those oriented to machines are low level.
7. Low-level language is a programming language that deals with a computer's hardware components and constraints. It has the capability to operate and handle the entire hardware and instructions set architecture of a computer directly.
8. Machine Language and Assembly language are two well known low level languages.
9. Machine language is the sequence of bits (machine code) that directly controls a processor, causing it to add, compare, move data from one place to another, and so forth at appropriate times. The computer microprocessor can process directly the machine codes without a previous transformation.
10. Assembly language uses structured commands (mnemonics) as substitutions for numbers allowing humans to read the code easier than looking at binary.
11. High level languages are portable (machine independent) as it can be run on different machines with little or no change. Furthermore, the rules for programming in a particular high-level language are much the same for all computers, so that a program written for one computer can generally be run on many different computers with little or no alteration.
12. The languages such as BASIC, COBOL, FORTRAN, C, C++, JAVA and Visual Basic are popular examples of high level languages.

13. In 1954 IBM established a project directed by John W. Backus to develop a compiler for the model 704 computer. This project resulted in the creation of the language FORTRAN (FORmulae TRANslation) that IBM finished in April 1957. The language had a notation orientation to mathematicians and scientists.
14. Dr. John G. Kemeny and Thomas Kurtz developed the BASIC language at Dartmouth in 1964. BASIC stood for "Beginner's All-Purpose Symbolic Instruction Code". Their objective: to create a simplified computer language for teaching students how to program
15. Dennis M. Ritchie created the C language at AT&T's Bell Laboratories in 1972. The transition in usage from the first major languages to the major languages of today occurred with the transition between Pascal and C. The language was designed to be portable, fast and compact. The UNIX operating system was later reprogrammed using the C language.
16. Microsoft released Visual Basic for Windows in May 1991. It is a graphical version of BASIC that simplifies the writing of program for windows.
17. In 1995 the Java language hit the scene, rapidly rose to popularity and is widely considered the most popular programming language in existence today.
18. For rapidly building web sites Perl (1987) and PHP (1995) have been two popular choices for a number of years. Among other things they make it easy to merge business data and page layout for delivery to web browsers.
19. The big technology companies have each largely aligned themselves with different languages stacks. Oracle and IBM are aligned with Java (Oracle actually owns Java).
20. Google are known for their use of Python (1997), a very versatile, dynamic and extensible language, although in reality they are also heavy users of C++ and Java. They have also created their own language called Go (2009).
21. An algorithm is a finite sequence of steps expressed for solving a problem. An algorithm can be defined as "a process that performs some sequence of operations in order to solve a given problem". Algorithms are used for calculation, data processing, and many other fields.
22. In computing, algorithms are essential because they serve as the systematic procedures that computers require. The reasons for using algorithms in programming are efficiency (time and cost factor), abstraction (simpler) and reusability (reusable).
23. A Flowchart is a type of diagram (graphical or symbolic) that represents an algorithm or process. Each step in the process is represented by a different symbol. A flowchart typically shows the flow of data in a process, detailing the operations/steps in a pictorial format which is easier to understand than reading it in a textual format. Once the flowchart is drawn, it becomes easy to write the program in any high level language.
24. A computer program is a sequence of instructions which a person (called a programmer) writes. This sequence of instructions is to be placed in the computer's memory. Needless to say, the

sequence of instructions (program) is designed to perform a certain function. The actual program written by the programmer in a higher level language is called the source program.

25. A source program can usually be translated to an object program that can be executed by the computer.
26. Translator is meant to translate one language to another. So a translator is mainly related to computer language.
27. In case of a computer, its hardware part only can operate when instructions are made of 0s and 1s, i.e., in machine language. But it is not that easy for human being to remember them correctly. Thus a move towards substituting such instructions of machine code by letter symbol-mnemonics was taken. Mnemonic codes ease programmer to write efficient program. But an intermediary agent is required to translate these mnemonic codes into machine codes. There are different types of translators for different categories of languages Software viz. assembler for assembly language, interpreter and compiler for high-level language.
28. Assembler is a translator programs used to automate the translation of source program in assembly language into object program in machine language.
29. An interpreter is a program which executes a source program in high level language, usually on a step by step, line by line, or unit by unit basis.
30. A compiler is simply a program which translates a source program written in a particular programming language to an object program which is capable of being run on a particular computer.
31. The compilation of programs is a complex process and consists of broadly two phases – analysis phase and synthesis phase. In the analysis phase, the source program is analyzed for its lexical, syntactical and semantic structure. The second phase is the synthesis of the object program in the machine language.
32. The greatest disadvantage of an interpreter is that certain phases of work and analysis must be done repeatedly. This contrasts with the compiler, which performs this translation function only once. On the other hand, the disadvantage to the compiler is that it does not produce answers; as soon as a change in the program is made a recompilation must be made.
33. A compiler takes more time to compile than to interpret, but a compiled program runs much faster than an interpreted program.
34. Programming is a part of software engineering used for producing a program- a list of instructions for the computer. Based on consideration such as what purpose the program is designed to serve and what a languages are already being used in the organisations, the programmer selects programming languages for development of software.
35. Current focus in programming language is Increasing support for functional programming, constructs to support concurrent and distributed programming, Mechanisms for adding security and reliability verification to the language.

Component-1 (A) – Module Structure: Detailed content structure of this module in the format given below.

Structure of Module/Syllabus of a module (Define Topic of module and its subtopic)	
Programming Languages	Programming Languages: Low Level, machine language, Assembly, High Level Language (HLL); Algorithm and Flow chart: Concept and Use.

1.2 Component-II - Description of Module: Describe module in the format given below:

	Description of Module
Subject Name	Library and Information Science
Paper Name	Programming Languages
Module Name/Title	Programming Languages
Module Id	LIS/ M-6
Pre-requisites	Basic computer knowledge including hardware, software and internet.
Objectives	To study basic programming concepts including Algorithm, Flowchart and language translators. It also discusses the different levels of programming languages namely low level language (machine and assembly) and high level language.
Keywords	Programming Language, Low level Language, Machine Language, Assembly Language, High Level Language, Language Translator, Algorithm, Flowchart