CHAPTER – I
INTRODUCTION AND CONCEPTUAL FRAMEWORK

1.1 INTRODUCTION

According to Nelson Mandela,

“Education is the most powerful weapon which you can use to change the world”

www.wisdomquotes.com

Education is a conscious and deliberate process in which one personality acts upon another in order to modify the development of the other by communication and manipulation of knowledge. It helps man to make a deliberate and conscious effort to live comfortably and happily in his physical and social environment.

It is considered as the eye opener for the human being, as it gives insight to discriminate between the good and the bad. It teaches how to behave with the others and lead the peaceful live. Education is considered to be the process to bring the overall development of a person. Education gives experience to an individual to make preventive measure to a problem that he/she is going to face in their life. Individuals use the past and present experiences to solve the future problem by the means of education.

1.1.1 Meaning of Education

Etymologically the term “education” is traced from the different sources. According to one view, “education” originated from the Latin word “educare” which means, “to bring up” or “to nourish”. There is another derivation from the Latin word “educere” which means, “to lead”. According to this view, the main purpose of education is to “lead” or “draw out” rather than “to put in”. There is another view according to which the term “education” comes from the Latin word “educatum” which means “the art of teaching or training”. Education is thus means both the acquisition of knowledge and experience as well as the development of skills, habits and attitudes which help a person to lead a full and worth while life in this world.
Webster defines education as the process of educating or teaching (now that’s really useful, isn’t it?) Educate is further defined as “to develop the knowledge, skill, or character of...” Thus, from these definitions, one can assume that the purpose of education is to develop the knowledge, skill, or character of students.

“The only purpose of education is to teach a student how to live his life-by developing his mind and equipping him to deal with reality. The training he needs is theoretical, i.e., conceptual. He has to be taught to think, to understand, to integrate, to prove. He has to be taught the essentials of the knowledge discovered in the past-and he has to be equipped to acquire further knowledge by his own effort.” said Ayn Rand

(Teacher’s Mind Resources: http://www.TeachersMind.com)

1.1.2 Definitions of Education

According to UNESCO (1989), “Education is an organized and sustained instruction designed to communicate a combination of knowledge, skills and understanding valuable for all the activities of life”.

According to Lubbock (1986), “Education is the harmonious development of all our faculties. It begins in the nursery, and goes on at school, but does not end there. It continues through life, whether we well or not”.


Education is a process of adjustment of the individual to adjust himself to the world of nature, the world of men and the world of value. It also exercises influence on one’s vocation, home life, friendship, marriage, travel, recreation and hobbies and tells upon his personality. There has been a constant quest to determine the basic issues of education. Various philosophers and educators have attempted to define the term “education”. The scholars from Socrates and Plato to Dewey and Mahatma Gandhi have expressed divergent views. (Mohanty, 1991).
1.1.3 **Principles of Education**

Swami Vivekananda emphasized the following principles in the Teaching- Learning process

*Self-Teaching:* “No one was ever taught by another. Each of us has to teach himself. A child educates himself.”

*Living by Examples of Teacher:* “Words even thoughts, contribute only one-third of the influence in making an impression the two-thirds”.

*Teaching through Positive Suggestions:* “We should give positive ideas. Negative ideas only weaken men both physically and mentally. If speak kind words to them and encourage them, they are bound to improve in time”.

*Concentration as only Method of Teaching:* “The power of concentration is the only key to the treasure house of knowledge”.

*Qualities of the learner:* “The conditions necessary for the taught are purity, a real thirst after knowledge and perseverance”.

1.1.4 **Structure of Education in India**

In India formal education is given to individuals through.

   i. Primary education.
   ii. Secondary education.
   iii. Higher secondary education.
   iv. Higher education.

1.2 **HIGHER EDUCATION**

The theme of higher knowledge and higher education were fashioned. The early Gurukula system of education flourished in the vedic and upanishadic periods, but a huge
university came to be set up at Takshahila in the $6^{th}$ century B.C. Two other universities, namely Nalanda and Vikramasila were established in the $4^{th}$ and $5^{th}$ centuries A.D. respectively.

The modern higher education system started, when the first three universities were set up in 1857 under the British rule. Policy guidelines given by Macaulay and Wood’s dispatch (1854) shaped the scope and the role of universities in India. To begin with, colleges’ setups in India were affiliated to British universities. In 1857 for the first time, universities were set up in India. Existing colleges got affiliated to these universities. The growth of the Indian higher education system is a result of the nation’s policy, adopted immediately after independence, to promote education amongst the masses.

Higher education provides people an opportunity to reflect on the critical, social, economic, cultural, moral and spiritual issues facing humanity. It contributes to national development through dissemination of specialized knowledge and skills. It is therefore a crucial factor for survival. (National Policy on Education, 1986, Para 5, 24).

1.2.1 Higher Education in India

The term "Higher Education" in India refers to post secondary or tertiary level education. Higher Education can also be defined as all graduate and post graduate degree courses with higher level skills, knowledge and competencies.

According to National Policy on Education Higher Education becomes dynamic. It provides some suggestions for higher education.

i. Autonomous colleges will be developed in large number. The creation of autonomous departments within the universities will be encouraged.

ii. The course and programmes of college education will be redesigned to meet the demands of specialization better. There will be increasing flexibility in the combination of courses.
iii. State level planning and co-ordination will be done through councils of higher education.

iv. An omission will be regulated according to capacity. Too much enrollment leads to congestion and indiscretion.

v. Teaching methods will be changed. Audio Visual aids and electronic equipment will be used.

vi. Research in the universities will be enhanced; steps will be taken to ensure its high quality.

vii. A National body covering higher education in general agricultural, medical, technical, and legal and other professional field will be set up.

In the 21st century there is paradigm shift in the teaching methods and learning styles with advent of information and communication technology.

Our vision is often affected by what we believe about the world; our beliefs often determine the information that we “see.” Extending this concept to technology, a paradigm effect may prevent people from seeing what is happening around them and from realizing the potential in a new application of technology. As Jim Wetherbe, Bobby G. Stevenson Chair in Information Technology at Texas Tech puts it, “The biggest obstacle to innovation is thinking it can be done the olden days. (reference-----)

When we think about how to utilize technology to improve learning, the key is to focus on what we can do with IT that we cannot do without it. Technology can create environments that provide individualized learning approaches that serve each person in ways that he or she can most benefit. Hence Information and Communication Technology cannot be thought without the usage of Computers which plays vital role in this technological era.
1.3 BRIEF HISTORY OF COMPUTER

Mechanical Computer

In 1791, Charles Babbage, invented “Difference Engine” and “Analytical Engine” to solve math table and general purpose calculations. Later on, he was called as “Father of Computers”. Followed by Charles Babbage, Jacquard loom, used punch cards for instructions. These two machines are called as Mechanical Computers.

Electro-Mechanical Computers

The 1880 census took 7 years to tabulate results, but in 1890 census took only 6 weeks to tabulate by Hermann Hollerith. In 1896, he founded the “Tabulating Machine Company”, which merged w/2 others in 1924 to form International Business Machines (IBM). In 1936, Howard Aiken, built the Harvard MARK1, which was 8 ft high and 55 ft long, and noisy. In 1941, Mauchly and Eckert, built the ENIAC, modeled after Atanasoff’s ABC computer, from the 30’s. The ENIAC had 18,000 vacuum tubes, 1,500 relays, in a room 20ft by 40ft. it sounded like a train.

1.4 GENERATIONS OF COMPUTER

First Generation (1937-1953): Three machines have been promoted at various times as the first electronic computers. These machines used electronic switches, in the form of vacuum tubes, instead of electromechanical relays. In 1941 Atanasoff and his student Clifford Berry had built a machine that could solve 29 simultaneous equations with 29 unknowns. However, the machine was not programmable, and was more of an electronic calculator; a second early electronic machine was Colossus, designed by Alan Turing for the British military in 1943. This machine played an important role in breaking codes used by the German army in World War II. The first general purposes programmable electronic computer was the Electronic Numerical Integrator and Computer (ENIAC), built by Presper Eckert and John V. Mauchly at the University of Pennsylvania. Work began in 1943, funded by the Army Ordnance Department, which needed a way to compute ballistics during World War II.
**Second Generation (1954-62):** The second generation saw several important developments at all levels of computer system design, from the technology used to build the basic circuits to the programming languages used to write scientific applications. The first machines to be built with this technology include TRADIC at Bell Laboratories in 1954 and TX-0 at MIT's Lincoln Laboratory. Important innovations in computer architecture included index registers for controlling loops and floating point units for calculations based on real numbers. Floating point operations were performed by libraries of software routines in early computers, but were done in hardware in second generation machines. During this second generation many high level programming languages were introduced, including FORTRAN (1956), ALGOL (1958), and COBOL (1959). Important commercial machines of this era include the IBM 704 and its successors, the 709 and 7094. The Livermore Atomic Research Computer (LARC) and the IBM 7030 (Aka Stretch) were early examples of machines that overlapped memory operations with processor operations and had primitive forms of parallel processing.

**Third Generation (1963-72):** The third generation brought huge gains in computational power. Innovations in this era include the use of integrated circuits, or ICs (semiconductor devices with several transistors built into one physical component), semiconductor memories starting to be used instead of magnetic cores, microprogramming as a technique for efficiently designing complex processors. In 1964, Seymour Cray developed the CDC 6600, which was the first architecture to use functional parallelism. Five years later CDC released the 7600, also developed by Seymour Cray. The SOLOMON computer, developed by Westinghouse Corporation, and the ILLIAC IV, jointly developed by Burroughs, the Department of Defense and the University of Illinois, was representative of the first parallel computers. The Texas Instrument Advanced Scientific Computer (TI-ASC) and the STAR-100 of CDC were pipelined vector processors that demonstrated the viability of that design and set the standards for subsequent vector processors. Early in this third generation Cambridge and the University of London cooperated in the development of CPL (Combined Programming Language, 1963). However, like ALGOL, CPL was large with many features that were hard to learn. In an attempt at further simplification, Martin Richards
of Cambridge developed a subset of CPL called BCPL (Basic Computer Programming Language, 1967). In 1970 Ken Thompson of Bell Labs developed yet another simplification of CPL called simply B, in connection with an early implementation of the UNIX operating system Comment.

**Fourth Generation (1972-84):** The next generation of computer systems saw the use of large scale integration (LSI - 1000 devices per chip) and very large scale integration (VLSI - 100,000 devices per chip) in the construction of computing elements. Semiconductor memories replaced core memories as the main memory in most systems; until this time the use of semiconductor memory in most systems was limited to registers and cache. During this period, high speed vector processors, such as the CRAY 1, CRAY X-MP and CYBER 205 dominated the high performance computing scene. Computers with large main memory, such as the CRAY 2, began to emerge. A variety of parallel architectures began to appear; however, during this period the parallel computing efforts were of a mostly experimental nature and most computational science was carried out on vector processors. Developments in software include very high level languages such as FP (Functional Programming) and Prolog (Programming in Logic). These languages tend to use a *declarative* programming style as opposed to the *imperative* style of Pascal, C, FORTRAN, et al. Compilers for established languages started to use sophisticated optimization techniques to improve code, and compilers for vector processors were able to vectorize simple loops (turn loops into single instructions that would initiate an operation over an entire vector).

**Fifth Generation (1984-1990):** The development of the next generation of computer systems was the widespread use of computer networks and the increasing use of single-user workstations. Toward the end of this period a third type of parallel processor was introduced to the market. In this style of machine, known as a *data-parallel* or SIMD, there are several thousand very simple processors. All processors work under the direction of a single control unit.

In the area of computer networking, both wide area network (WAN) and local area network (LAN) technology developed at a rapid pace, stimulating a transition from
the traditional mainframe computing environment toward a distributed computing environment in which each user has their own workstation for relatively simple tasks (editing and compiling programs, reading mail) but sharing large, expensive resources such as file servers and supercomputers. RISC technology (a style of internal organization of the CPU) and plummeting costs for RAM brought tremendous gains in computational power of relatively low cost workstations and servers. This period also saw a marked increase in both the quality and quantity of scientific visualization.

**Sixth Generation (since 1990):** This generation is beginning with many gains in parallel computing. Parallel systems now compete with vector processors in terms of total computing power and most expect parallel systems to dominate the future. Combinations of parallel/vector architectures are well established, and one corporation (Fujitsu) has announced plans to build a system with over 200 of its high end vector processors. One of the most dramatic changes in the sixth generation will be the explosive growth of wide area networking. Network bandwidth has expanded tremendously in the last few years and will continue to improve for the next several years.

### 1.5 COMPUTER SOFTWARE

A computer or an information system does not rely only on hardware and data resources alone, but on the required software resources. Software consists of a set of instructions called programs that instructs the computer. Software is a very general term encompassing programs employed in all fields written by the beginners or experienced professionals and designed for variety of purpose. For educators, it helps to prepare courseware and learning packages. System depends upon software resources and helps the users to use the computer hardware and to transform the data resources into a variety of information products.

#### 1.5.1 Classification of software

Generally softwares are classified into two: they are system software and application software.
System Software: it makes the computer to understand and do the desired tasks as per the programs or instructions stored in the memory.

1.5.1.1 Application software

Programs directed towards the performance of a particular application or use is called application software. Application programs that direct computer to perform specific information processing activities for users. The programs are called application packages, because they direct the processing required for a particular use or application.

There are several application software programs and some of them are: spreadsheet programs, database programs, word processing programs, graphics programs and communication programs. There are plenty of application software packages, which cover different areas in education, desktop publishing, finance, utility and more.

1.5.1.2 Categories of software

The softwares are classified in the following categories: Instructional software, authoring software, database software, utility software, language software and web authoring software.

**Instructional software**

Instructional software is the CAI material. It consists of programs that teach either subject or develop skills.

**Authoring Software**

It allows the user to write programs without knowing formal programming languages. Such programs walk through different steps in building a lesson by asking questions about programming needs. This program translates the lesson written in English into a higher level programming language.
**Word Processing software**

It allows creating and editing text electronically on the monitor screen. Thus, what the user types; leaves no permanent impression as a typewriter does on paper. We can make corrections by typing over, what was already there and the newly typed copy leaves no correction.

The programs that computerize the creation, editing and printing of documents such as letters, memos, reports and text data. Thus, the application of word processing package is office automation. The following may be quoted as examples: Word perfect, WordStar, MicroSoft Word, PageMaker, and Open office org 1.3.

**Spreadsheet Software**

Application programs used for analysis, planning and modeling are known as spreadsheet software. It provides replacement for paper, worksheets, pencil and calculators. E.g. Lotus 1-2-3, MS Excel and Quattro pro.

**Telecommunication Software**

These application packages can connect the computer within internet. These packages help the user to explore the World Wide Web. E.g. Internet Explorer, Netscape Navigator, Mosaic, Opera, Avant, Crazy and Mozilla.

**Graphic Software**

It converts the numeric data into graphical displays such as line charts, bar charts, pie-charts and presentations. Presentation of graphics can be produced by graphics packages such as Harvard graphics, Lotus Freelance, SAS Graph and Tell-A-Graph.

**Multimedia Software**

It helps the user to create multimedia-oriented graphics, multimedia sounds and their animations, such as Adobe Premiers, Adobe Premier Pro, Kinetix 3D Max, Macromedia Flash, Maya, Macromedia Director, and sound Forge.
**DTP Software**

DTP package is a package of publishing works; that is, text-oriented applications or word processing. The examples for DTP packages are Ventura, PageMaker, Corel Products, Photoshop, Photo draw, Latex, Sierra, Open Office org 1.3.

**Database Software**

Database Management systems are rapidly making the old manual procedures obsolete. Records stored on disks and retrieval can take place in various ways. Database software is commonly used in education for keeping equipment inventories, class schedules, teaching resource data and attendance records, e.g. SQL server and Oracle.

**Utility software**

Utility software directs the computer to perform certain functions, which facilitate everyday user operations. This description might sound little vague, so here, we have some concrete applications of utility software and they are used for the following purposes:

i. Merging one program with another  
ii. Remembering the statement of a program  
iii. Copying the content form one disk to another  
iv. Producing graphics that can be integrated with existing programs

**Gaming software**

It is the broad category of software designed and purchased with recreation as the main consideration.

**Language Software**

This makes certain programming languages available to the computer. In most of the brands of computer we have the BASIC languages built in. other special software is required to load the languages into the RAM. e.g. C, C++, VISUAL BASIC, JAVA and SQL.
**Web Authoring Software**

Web is a network of networks. For the development of World Wide Web, coding is needed. The design and coding of Worldwide Web is known as web design or web authoring. For developing the web, we need language and for writing the coding, we need editors or packages. Scripting programs are the simple programming language for a special or limited task. In scripting language, there are two scripts and they are client script and server scripts.

**Virtual Reality Modeling Language (VRML)**

VRML is a scene description language. It is an open standard for 3D graphics and interaction on the internet. It is a subject of 3D graphics; that is everything in VRML is 3D graphics, but not every 3D graphics is VRML.

It falls into a broad 3D graphics category called “Real time”. In real time graphics, the computer tracks the user movement or navigates through the virtual scene and updates the image on the screen as the user moves through it. It is like html can have interactions and behaviors such as hyperlinking to other files or animations.

VRML is the acronym of Virtual Reality Modeling Markup Language for creating interactive three dimensional image sequences and possible user interactions with them. VRML file can be embedded in HTML documents but VRML enabled browser is needed for viewing such web pages. It works on the protocol blue tooth for wireless communications. It leads us to future advancement. (Anil Madaan, 2001)

The Virtual Reality Modeling Language (VRML) is a more public domain approach to this technology. VRML is an ASCII based three-dimensional modeling language, often described as the HTML of virtual reality. The basic concept is that any object can be modeled by grouping such basic shapes as: cones, cubes, cylinders and spheres, and custom shapes called “Indexed Face Sets”. By specifying the construction method in the VRML file (as a grouping of shapes in relation to each other with different textures, reflectiveness, and colors), the creator can develop a small, compact model. Once created, the object can be treated as a single unit much like a macro.
It is often suggested that the eventual goal of VRML is to create a “Cyberspace” similar to that depicted in William Gibson’s Mona Lisa Overdrive, where users interact with a global computer network in a three-dimensional virtual world. With the current VRML specifications (version 2.0), it is possible to add animation, sound and links, and to interact with both the environment and other users. Indeed, many sites attempt to demonstrate this capability, but, even with good bandwidth and high-performance computers, the animation and exploration are slow and jerky.

**VRML**

i. Text based

ii. Can be very small

iii. Content must be modeled

iv. Can be played cross-platform

v. Looks like “Computer animation”

vi. VRML 2.0 Built into Netscape 3.0

vii. Slow and jerky. Performance decreases with added complexity

viii. Can be developed on any medium to high performance computer

ix. Must convert everything into points, shapes, surfaces, and locations

x. Absolute freedom of movement in any direction, at any distance, at any angle

xi. Must learn how to use complex CAD or VR software for complex productions.

xii. CPU intensive floating-point mathematical algorithms requiring fast computation

(http://www.collectinscanada.ca)

### 1.6 COMPUTER LANGUAGES

There are many different languages can be used to program a computer. The most basic of these is machine language- a collection of very detailed, cryptic instructions that control the computer’s internal circuitry. This is the natural dialect of the computer. Very few computer programs are actually written in machine language for two significant reasons: first, because machine language is very cumbersome to work with and second, because every different type of computer has its own unique instruction set. Thus, a
machine-language program written for one type of computer cannot be run on another type of computer without significant alterations.

Usually, a computer program will be written in some high-level language, whose instruction set is more compatible with human languages and human thought processes. Most of these are general-purpose languages such as C. Some other general purpose languages are Pascal, FORTRAN, and BASIC. There are also various special-purpose languages that are specifically designed for some particular type of application. Some common examples are CSMP and SIMAN, which are special-purpose simulation languages, and LISP, a list processing language that is widely used for artificial intelligence applications.

As a rule, a single instruction in a high-level language will be equivalent to several instructions in machine language. This greatly simplifies the task of writing complete, correct programs. 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. A high-level language offers three significant advantages over machine language: Simplicity, uniformity and portability (i.e. machine independence).

A program that is written in a high-level language must be translated into machine language before it can be executed. This is known as compilation or interpretation, depending on how it is carried out. Compilers translate the entire program into machine language before executing any of the instructions. Interpreters, on the other hand, proceed through a program by translating and then executing single instructions or small groups of instructions.

1.7 FUNDAMENTALS OF ‘C’ LANGUAGE

C is a general purpose, structured programming language. Its instructions consists of terms that resemble algebraic expressions, augmented by certain English keywords such as if, else, for, do and while. In this respect C resembles other high-level programming languages such as Pascal and FORTRAN. C also contains certain additional features, that allow it to be used at a lower level, thus bridging the gap between machine language and
the more conventional high-level languages. This flexibility allows C to be used for systems programming (e.g. for writing operating systems) as well as for applications programming. (e.g. for writing a program to solve a complicated system of mathematical equations, or for writing a program to bill customers).

C is characterized by the ability to write very concise programs, due in part to the large number of operators included within the language. It has a relatively small instruction set, through accrual implementations include extensive library functions which enhance the basic instructions. Furthermore, the language encourages users to write additional library functions of their own. Thus the features and capabilities of the language can easily be extended by the user.

C compilers are commonly available for computers of all sizes, and C interpreters are becoming increasingly common. The compilers are usually compact, and they generate object programs that are small and highly efficient when compared with programs compiled from other high-level languages.

Another important characteristic of C is that its programs are highly portable, even more so than with other high-level languages. The reason for this is that C relegates most computer-dependent features to its library functions. Thus, every version of C is accompanied by its own set of library functions, which are standardized, however, and each individual library function is generally accessed in the same manner form one version of C to another. Therefore, most C programs can be processed on many different computers with little or no alteration.

1.7.1 History of C

C was originally developed in the 1970s by Dennis Ritchie at Bell Telephone Laboratories, Inc. (Now a part of AT&T). It is an outgrowth of two earlier languages, called BCPL and B, which were also developed at Bell Laboratories. C was largely confined to use within Bell Laboratories until 1978, when Brian Kernighan and Ritchie published a definitive description of the languages.* The Kernigham and Ritchie
description is commonly referred to as “K&RC”.

Following the publication of the K&R description, compute professional, impressed with C’s many desirable features, began to promote the use of the language. By the mid 1980s, the popularity of C had become widespread. Numerous C compilers and interpreters had been written for computers of all sizes, and many commercial application programs had been developed. Moreover, many commercial software products that were originally written in other languages were rewritten in C in order to take advantage of its efficiency and its portability.

Every commercial implementation of C differed somewhat from Kernighan and Ritchie’s original definition, resulting in minor incompatibilities between different implementations of the language. These differences diminished the portability that the language attempted to provide. Consequently, the American National Standards Institute (ANSI committee and X3J11) has developed a standardized definition of the C language. Virtually all commercial C compilers and interpreters now adhere to the ANSI standard. Many also provide additional features of their own.

1.7.2 Structure of ‘C’ program

Every C program consists of one or more modules called functions. One of the functions must be called main. The program will always begin by executing the main function, which may access other functions. Any other function definitions must be defined separately, either ahead of or after main. Each function must contain:

i. A function heading, which consists of the function name, followed by an optional list of arguments enclosed in parentheses.

ii. A list of argument declarations, if arguments are included in the heading.

iii. A compound statement, which comprises the remainder of the function.

The arguments are symbols that represent information being passed between the function and other parts of the program, which are also referred to as parameters. Each compound statement is enclosed within a pair or braces, i.e. {}. The braces may contain one or more elementary statements (called expression statements) and other
compound statements. Thus compound statements may be nested, one within another. Each expression statement must end with a semicolon (;). Comments (remarks) may appear anywhere within a program, as long as they are placed within the delimiters /* and */. Such comments are helpful in identifying the program’s principal features or in explaining the underlying logic of various program features.

1.7.3 DESIRABLE PROGRAM CHARACTERISTICS

Following are the desirable characteristics of any computer program

i. **Integrity**: This refers to the accuracy of the calculations. It should be clear that all other program enhancements will be meaningless if the calculations are not carried out correctly. Thus, the integrity of the calculations is an absolute necessity in any computer program.

ii. **Clarity**: It refers to the overall readability of the program, with particular emphasis on its underlying logic. If a program is clearly written, it should be possible for another programmer to follow the program logic without undue effort. It should also be possible for the original author to follow his or her own program after being away from the program for an extended period of time. One of the objectives in the design of C is the development of clear, readable programs through an orderly and disciplined approach to programming.

iii. **Simplicity**: The clarity and accuracy of a program are usually enhanced by keeping things as simple as possible, consistent with the overall program objectives. In fact, it may be desirable to sacrifice a certain amount of computational efficiency in order to maintain a relatively simple, straightforward program structure.

iv. **Efficiency**: It is concerned with execution speed and efficient memory utilization. These are generally important goals, though they should not be obtained at the expense of clarity or simplicity. Many complex programs require a tradeoff between these characteristics. In such situations, experience and common sense are key factors.

v. **Modularity**: Many programs can be broken down into a series of identifiable subtasks. It is good programming practice to implement each of these subtasks as a separate program module. In C, such modules are written as functions. The
use of a modular programming structure enhances the accuracy and clarity of a program, and it facilitates future program alterations.

vi. **Generality:** Usually a program to be as general as possible, within reasonable limits. For example, we may design a program to read in the values of certain key parameters rather than placing fixed values into the program. As a rule, a considerable amount of generality can be obtained with very little additional programming effort.

### 1.7.4. THE ‘C’ CHARACTER SET

The basic elements used to construct simple c statements include the c character set, identifiers and keywords, data types, constants, variables and arrays, declarations, expressions and statements. C uses the uppercase letters A to z, the digits 0 to 9, and certain special characters as building blocks to form basic program elements. The special characters are

+ , - , * , / , = , % , & , # , ! , ? , ^ , " , ' , ~ , \ , | , < , > , ( , ) , [ , ] , { , ) , ; , ; , , , , , _ (blank space).

Most versions of the language also allow certain other characters, such as @ and $, to be included within strings and comments. C uses certain combinations of these characters, such as \b, \n, and \t, to represent special conditions such as backspace, new line and horizontal tab, respectively. Theses character combinations are known as escape sequences.

### 1.7.5 IDENTIFIERS AND KEYWORDS

Identifiers are names that are given to various program elements, such as variables, functions and arrays. Identifiers consist of letters and digits, in any order, except that the first character must be a letter. Both upper and lowercase letters are permitted, though common usage favors the use of lowercase letters for most types of identifiers. The underscore character ( _ ) can also be included, and is considered to be a letter. An underscore is often used in the middle of an identifier. An identifier may also begin with an underscore. The following names are valid identifier
There are certain reserved words, called keywords, that have standard, predefined meanings in C. These keywords can be used only for their intended purpose; they cannot be used as programmer–defined identifiers. The standard keywords are:

- auto
- break
- case
- char
- const
- Continue
- default
- do
- double
- else
- enum
- extern
- float
- for
- struct
- goto
- if
- int
- long
- register
- return
- short
- signed
- static
- switch
- typedef
- union
- volatile
- while

Some compilers may also include some or all of the following keywords:

- ada
- far
- near
- asm
- fortran
- pascal
- entry
- huge

### 1.7.6 DATA TYPES

C supports several different types of data, each of which may be represented differently within the computer’s memory. The basic data types are listed below. Typical memory requirements are also given.

<table>
<thead>
<tr>
<th>S.NO</th>
<th>DATA TYPES</th>
<th>DESCRIPTION</th>
<th>MEMORY REQUIREMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>int</td>
<td>Integer quantity</td>
<td>2 bytes or one word (varies from one compiler to another)</td>
</tr>
<tr>
<td>2</td>
<td>char</td>
<td>Single character</td>
<td>1 byte</td>
</tr>
<tr>
<td>3.</td>
<td>float</td>
<td>Floating point number</td>
<td>1 word (4 bytes)</td>
</tr>
<tr>
<td>4.</td>
<td>double</td>
<td>Double-precision floating point number</td>
<td>2 words (8 bytes)</td>
</tr>
</tbody>
</table>
1.7.6.1 CONSTANTS

There are four basic types of constants in C. They are integer constants, floating-point constants, character constants and string constants.

Integer and floating-point constants represent numbers. They are often referred to collectively as numeric-type constants. The following rules apply to all numeric – type constants.

i. Commas and blank spaces cannot be included within the constant
ii. The constant can be preceded by a minus sign if desired.
iii. The value of a constant cannot exceed specified minimum and maximum bounds.

For each type of constant, these bounds will vary from one c compiler to another.

**Integer Constants**

An integer constant is an integer-valued number. Thus it consists of a sequence of digits. Integer constants can be written in three different number systems: decimal (base 10), octal (base 8) and hexadecimal (base 16). A decimal integer constant two or more digits, the first digit must be something other than 0. Some of the valid decimal integer constants are: 0, 1, 743, 5280, 32767 and 9999. A hexadecimal integer constant must begin with either 0x or 0X. It can then be followed by combination of digits taken from the sets 0 through 9 and a through f. Some of the valid hexadecimal integer constants are 0x 0X1 0X7fff 0xabbcd.

**Unsigned and Long integer constants**

It may exceed the magnitude of ordinary integer constants by approximately a factor of 2, though they may not be negative. An unsigned integer constant can be identified by appending the letter U (either upper or lower case) to the end of the constant. Long integer constants may exceed the magnitude of ordinary integer constants, but require more memory within the computer. An unsigned long integer may be specified by appending the letters UL to the end of the constant. The letters may be written in either upper or lower case.
**Floating-point constants**

A floating point constant is a base-10 number that contains either a decimal point or an exponent.

**Character Constants**

A character constant is a single character, enclosed in apostrophes.

e.g. ‘a’, ‘x’, ‘3’, ’?’’, ‘’. Character constants have integer values that are determined by the computer’s particular character set. Thus, the value of a character constant may vary from one computer to another. The constants themselves, however, are independent of the character set. This feature eliminates the dependence of a C program on any particular character set. Most computers, and virtually all personal computers, make used of the ASCII(American Standard Code for Information Interchange) character set, in which each individual character is numerically encoded with its own unique 7-bit combination (hence a total of $2^7 = 128$ different characters).

**Escape Sequences**

Certain nonprinting characters, as well as the backslash (\) and the apostrophe (‘), can be expressed in terms of escape sequences. An escape sequence always begins with a backward slash and is followed by one or more special characters. For example, a line feed, which is referred to as a newline in C, can be represented as \n. Such escape sequences always represent single characters, even though they are written in terms of two or more characters. Of particular interest is the escape sequence \0. this represents the null character, which is used to indicate the end of a string, which is not equivalent to the character constant ‘0’.

**String Constant**

A string constant consists of any number of consecutive characters (including none), enclosed in (double) quotation marks.
e.g. “Green”, “270-875”.

1.7.7 VARIABLES AND ARRAYS

A variable is an identifier that is used to represent some specified type of information within a designated portion of the program. A variable is an identifier that is used to represent a single data item i.e., a numerical quantity or a character constant. The data item must be assigned to the variable at some point in the program. The data item can then be accessed later in the program simply by referring to the variable name.

A given variable can be assigned different data items at various places within the program. Thus, the information represented by the variable can change during the execution of the program. However, the data type associated with the variable cannot change.

```c
int a,b,c;
char d;
A = 3;
B = 5;
```

The array is another kind of variable that is used extensively in C. An array is an identifier that refers to a collection of data items that all have the same name. The data items must all be of the same type. The individual data items are represented by their corresponding array elements. The individual array element is distinguished from one another by the value that is assigned to a subscript.

Example: suppose that X is a 10 element array. The first element is referred to as X [0], the second as X [1], and so on. The last element will be X [9]. The subscript always ranges from 0 to n-1.

Since the array is one dimensional, there will be a single subscript (sometimes called as index) whose value refers to individual array elements. If the array contains n elements, the subscript will be an integer quantity whose values range from 0 to n-1.
character string will require an n+1 element array, because of the null character(\0) that is automatically placed at the end of the string.

**Declarations**

A declaration associates a group of variables with a specific data type. All variables must be declared before they can appear in executable statements. A declaration consists of a data type, followed by one or more variable names, ending with a semicolon. Each variable must be followed by a pair of square brackets, containing a positive integer which specifies the size of the array.

eg. int a,b,c;
float root1, root2;
char flag, text [80];

**1.7.8 EXPRESSION**

An expression represents a single data item, such as a number or a character. The expression may consist of a single entity, such as constant, a variable, an array element or a reference to a function. It may also consist of some combination of such entities, interconnected by one or more operators. The use of expressions involving operators is particularly common in C, as in most other programming languages. Expressions can also represent logical conditions that are either true or false. However, in C the conditions true and false are represented by the integer values 1 and 0 respectively. Hence logical type expressions really represent numerical quantities.

e.g. A +b;
X = y;
C = a+b;
X<=y;
X == y;
++ l;
1.7.9 STATEMENTS

A statement causes the computer to carry out some action. There are three different classes of statements in C: expression statements, compound statements, and control statements. An expression statement consists of an expression followed by a semicolon. The execution of an expression statement causes the expression to be evaluated.

Eg.
A = 3
C = a+b
++i
printf("area = %f", area);

1.7.10 SYMBOLIC CONSTANTS

A symbolic constant is a name that substitutes for a sequence of characters. The characters may represent a numeric constant, a character constant, or a string constant. Thus, a symbolic constant allows a name to appear in place of a numeric constant, a character constant, or a string. When a program is compiled, each occurrence of a symbolic constant is replaced by its corresponding character sequence. A symbolic constant is defined by writing

#define name text

Where name represents a symbolic name, typically written in uppercase letters, and text represents the sequence of characters that is associated with the symbolic name.

1.7.11 OPERATORS AND EXPRESSIONS

Constants, variables, array elements, and function references can be joined together by various operations to form expressions. C includes a large number of operators which fall into several different categories.

Arithmetic operators

There are five arithmetic operators in C. They are
<table>
<thead>
<tr>
<th>Operator</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operator</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
</tr>
<tr>
<td>%</td>
<td>Reminder after integer division</td>
</tr>
</tbody>
</table>

The % operator is sometimes referred to as the modulus operator. There is no exponentiation operator in C. However, there is a library function (pow) to carry out exponentiation.

e.g. If a and b carries 10 and 3 respectively.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a+b</td>
<td>13</td>
</tr>
<tr>
<td>a-b</td>
<td>7</td>
</tr>
<tr>
<td>a*b</td>
<td>30</td>
</tr>
<tr>
<td>a/b</td>
<td>3</td>
</tr>
<tr>
<td>a%b</td>
<td>1</td>
</tr>
</tbody>
</table>

Unary operators

C includes a class of operators that act upon a single operand to produce a new value. Such operators are known as unary operators. The most common unary operation is unary minus. In C, however, all numeric constants are positive. Thus, a negative number is actually an expression, consisting of the unary minus operator, followed by a positive numeric constant.

Relational and logical operators

There are four relational operators in C. They are

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
</table>

26
These operators all fall within the same precedence group, which is lower than the arithmetic and unary operators. The associativity of these operators is left to right and closely associated with the relational operators are the following two equality operators.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>

The equality operators fall into a separate precedence group, beneath the relational operators. These operators also have a left to right associativity. These six operators are used to form logical expressions, which represent conditions that are either true or false. The resulting expressions will be of type integer, since true is represented by the integer value 1 and false is represented by the value 0.

**Assignment operators**

There is several different assignment operators in C. All of them are used to form assignment expressions which assign the value of an expression to an identifier. The most commonly used assignment operator is =. Assignment expressions that make use of this operator are written in the form

```
Identifier = expression
```

Where identifier generally represents a variable and expression represents a constant, a variable or a more complex expression.

e.g.  
```
A = 3;
X = y;
Del= 0.001;
```
Sum = a+b;
Area = len * bred;

The conditional operator

Simple conditional operations can be carried out with the conditional operator (\%). An expression that makes use of the conditional operator is called conditional expression. Such an expression can be written in place of the more traditional if-else statement.
Expression 1 ? expression 2 : expression 3

When evaluating a conditional expression, expression1 is evaluated first. If expression 1 is true, then expression 2 is evaluated and this becomes the value of the conditional expression. However, if expression 1 is false, then expression 3 is evaluated and this becomes the value of the conditional expression.

1.7.12 LIBRARY FUNCTIONS

The C language is accompanied by a number of library functions that carry out various commonly used operations or calculations. These library functions are not a part of the language. Library functions are functionally similar are usually grouped together as object programs in separate library files. These library files are supplied as a part of each C compiler. All C compilers contain similar group of library functions, though they lack precise standardization. Thus there may be some variation in the library functions that are available in different versions of the language.

1.7.13 DATA INPUT AND OUTPUT

An input/output function can be accessed from anywhere within a program simply by writing the function name, followed by a list of arguments enclosed in parentheses. The arguments represent data items that are sent to the function. Some input/output functions do not require arguments, though the empty parentheses must still appear. The names of those functions that return data items may appear within expressions, as though each function reference were an ordinary variable, or they may be referenced as separate
statements. Some functions do not return any data items. Such functions are referenced as though they were separate statements. Most versions of C include a collection of header files that provide necessary information in support of the various library functions. Each file generally contains information in support of group of related library functions. These files are entered into the program via an # include statement at the beginning of the program. As a rule, the header file required by the standard input/output library functions is called stdio.h.

1.7.13.1 Single character input – the getchar() function

Single characters can be entered into the computer using the C library function getchar(). The getchar function is a part of the standard C I/O library. It returns a single character from a standard input device. The function does not require any arguments, though a pair of empty parentheses must follow the word ‘getchar’.

In general terms, a function reference would be written as

\[
\text{Character variable} = \text{getchar();}
\]

Where \textit{character variable} refers to some previously declared character variable.

e.g. char c;

\[
\text{......}
\]

\[
c = \text{getchar();}
\]

If an end-of-file condition is encountered when reading a character with the getchar function, the value of the symbolic constant EOF will automatically(-1) be returned. The detection of EOF in this manner offers a convenient way to detect a end of file, whenever and wherever it may occur.

1.7.13.2 Single character output – the putchar() function

Single characters can be displayed using the C library function putchar. This function is complimentary to the character input function getchar. The putchar function, like getchar, is a part of the standard C I/O library. It transmits a single character to a standard output device. The character being transmitted will normally be represented as a
character –type variable. It must be expressed as an argument to the function, enclosed within parentheses, following the word putchar.

`putchar( character variable)`

Where character variable refers to some previously declared character variable.

### 1.7.13.3 Entering input data - the scanf() function

Input data can be entered into the computer from a standard input device by means of the C library function `scanf`. This function can be used to enter any combination of numerical values, single characters and strings. The function returns the number of data items that have been entered successfully.

The general format is

`scanf(control string, arg1, arg2, …., argn);`

Where control string refers to a string containing certain required formatting information, and arg1, arg2, …., argn are arguments that represent the individual input data items.

### 1.7.13.4 Writing output data – the printf() function

Output data can be written from the computer onto a standard output device using the library function `printf`. This function can be used to output any combination of numerical values, single characters and strings. It is similar to the input function `scanf`, except that its purpose is to display data rather than to enter it into the computer. That is, the `printf` function moves data from the computer’s memory to the standard output device, where the `scanf` function enters data from the standard input device and stores it in the computer’s memory.

The general format is

`printf(control string, arg1, arg2, …., argn);`

When control string refers to a string that contains formatting information, and arg1, arg2, …., argn are arguments that represent the individual output data items.
1.7.13.5 The gets and puts function

C contains a number of other library functions that permit some form of data transfer into or out of the computer. Each of these functions accepts a single argument. The argument must be a data item that represents a string. The string may include white space characters. In the case of gets, the string will be entered from the keyboard, and will terminate with a new line character. The gets and puts function offer simple alternatives to the use of scanf and printf for reading and displaying strings.

For eg.

```c
#include <stdio.h>
main()
{
    char line[80];
    gets(line);
    puts(line);
}
```

1.8 FACTORS INFLUENCING EFFICIENT PROGRAM WRITING ABILITY

Every computer programmer should know the technique of writing efficient programs. Meta-cognitive knowledge and meta-cognitive activity are the most required skills, while developing such programs using “Pointers in C” language. Meta-cognition is cognition about cognition, or “Knowing about knowing” (Flavell, 1999; Flavell, Miller, & Miller, 2002). Meta-cognitive Knowledge involves monitoring and reflecting on one’s current or recent thoughts. This includes both factual knowledge, such as knowledge about the task, ones goals, or oneself, and strategic knowledge such as how and when to use specific procedures to solve problems. Meta-cognitive activity occurs when students consciously adapt and manage their thinking strategies during problem solving and purposeful thinking (Ferrari & Sternberg, 1998; Kuhn & others, 1995). Meta-cognitive skills have been taught to students to help them solve math problems (Cardelle-Elwar, 1992). One expert on children’s thinking, Deanna Kuhn (1999a, 1999b), believes that meta-cognition should be a stronger focus of efforts to help children become better critical thinkers,
especially at the middle school and high school levels. She distinguishes between first-order cognitive skills, which enable children to know about the world (and have been the main focus of critical thinking programs), and second-order cognitive skills-meta-knowing skills—which involve knowing about one’s own (and other’s) knowing. Pointers are the variables store the address of another variable in the memory. To develop programs using pointers in C, there is a need for higher order cognition. Therefore, metacognition is considered as one of the influencing factor in developing programs.

Besides this; the knowledge of Logical-Mathematical Intelligence is also required. Because it is the capacity to analyse problems logically, carry out mathematical operations and investigate issues scientifically. Since this intelligence is closely associated with scientific and mathematical thinking, the process of addressing the memory location using ‘c’ pointers involves more mathematical and scientific calculations. Therefore, this intelligence is included as one of the influencing factors in writing C programs.

But, to address the memory location using ‘C’ pointers and to develop the program with the available information, there is a need for two more skills such as i. The Information Processing approach. ii. Imagination/Visualization. These two skills are also playing vital role in developing such programs. According to Mc Carthy’s 4 Mat type 2 learning styles; analytic learners are primarily interested in acquiring facts in order to deeper their understanding of concepts and processes. They are capable of learning effectively from lectures, and enjoy independent research, analysis of data, and hearing what “the experts” have to say. Right brain dominates the learning of every individual through visualization and information processing approach. But, in day-to-day learning, much importance is not given to stimulate the right brain, which is essential to develop efficient programs. Therefore, the type of Brain dominance is considered as another factor for this study, which influences the program writing ability of the students.

However, no program can be developed efficiently without proper knowledge of error diagnosis and debugging techniques. Programming errors often remain undetected
until an attempt is made to compile or execute the program. Most C compilers will generate diagnostic messages when syntactic errors have been detected during the compilation process and an execution error, which occurs during program execution, after a successful compilation. As a beginning programmer, one should have the knowledge of different methods available to find the location of execution errors and logical errors within a program. Such methods are generally referred to as debugging techniques. Some of the commonly used debugging techniques are Error isolation, Tracing, Watch values, Break Points and Stepping.

Apart from the above mentioned factors, which influence the program writing ability, self-efficacy plays the most important role in all aspects of life. According to Bandura 1977, Self-Efficacy is the belief in one’s capacity to organize and execute the course of action required to manage prospective situations. Any programmer with low level of Self-efficacy but with good level of knowledge and in all the above mentioned factors will end up with failure result. Therefore, self-efficacy is considered as another influencing factor which comprises of Mastery experience, Vicarious Experience, persuasion and Encouragement, and interpretation of emotional arousal.

1.9. TEACHING METHODS

The word ‘method’ in Latin means ‘mode’ or ‘way’. It means the mode by which the material is communicated from the teacher to the student. Method of teaching may be redefined as the methods by which the teacher impart knowledge and skills while teaching the students comprehend knowledge and acquire the skills in the process of learning. This definition clarifies that method includes both teaching (teacher activity) and learning (learners’ activity). Rage defined, ‘teaching method are patterns of the teacher behaviors that are recurrent, applicable to various subject matters, characteristics of more than one teacher, and relevant to learning’. It means methods are part of the behaviour of teacher which he/she uses as a strategy or tactics of teaching. The method is also related to content and is helpful in generating learning.
1.9.1 History of Teaching Methods

Pestolozzi laid emphasis on ‘psychologizing education’:

i. To develop methods in line with the developmental pattern of children’s growth, and

ii. To make the process of perception as the central element in his teaching method.

Froebel methods laid emphasis on the study of child taking account of emotional as well as intellectual development.

Herbert devised a series of instructional steps which is known as Herbertian Teaching Method.

The most popular form of 5 steps is as follows:

i. Preparation

ii. Presentation

iii. Association

iv. Assimilation

v. Application

Similarly the supporters of Progressive Education and John Dewey have observed that the essential element in all methods is activity. It will be relevant to say that teaching methods have been influenced over the years, by many factors, such as:

i. Educational goals

ii. Cultural and political factors

iii. Study of learner’s intellectual growth

iv. Educational psychology

v. Analysis of learning and teaching

vi. Technology
1.9.2 Grouping of Teaching Methods

These are numerous methods of teaching with common characteristics. These common characteristics are related to classroom interactions and also indicate the behaviors of teacher. These are also related to different modes of learning. Zerve and Vaidya have tried to arrange different methods into different groups

I. Oral Method

i. Narration
ii. Recitation
iii. Lecture
iv. Discussion

Common Characteristics are
i. Teacher-centered Method
ii. The teacher communicates information or gives knowledge through verbal means. Learner is a passive listener.

II. Activity Method

i. Demonstration
ii. Activity
iii. Project
iv. Laboratory
v. Heuristic
vi. Discovery learning/Inquiry approval
vii. Problem-solving
viii. Supervised method

Common Characteristics are
i. Learner centered
ii. Learner takes place due to active involvement of learner
iii. teacher functions as a facilitator of learning or as a stage setter for learning
III. Special Method

i. Programmed learning
ii. Team teaching
iii. Computer assisted learning
iv. Personalized system of instruction

These fulfill a specific requirement which is based on psychological theories or technological facilities. These methods seek participation of learner. Selection of suitable teaching method is based on the objective of the lesson, needs of the learner and nature of the content. Some of the commonly used teaching methods are:

i. Lecture Method
ii. Discussion Method
iii. Project Method
iv. Inquiry Approach
v. Demonstration Method

1.10. COMPUTER ASSISTED INSTRUCTION

Computer has contributed a lot in each and every sectors of life. Computer Assisted Instruction has emerged as effective and efficient media of instruction in the advanced countries of the world. In fact, CAI is being used in formal and non-formal education at all the levels. In India too, computer has been introduced in most of the areas such as data processing decision making.

The Computer Assisted Instruction may be defined as the use of a computer as an integral part of an instructional system the learner generally engaging in two-way interaction with the computer. In teaching-learning process, computer plays a major role. In the Computer Assisted Instructions, it interacts directly with the learners while presenting lessons. The Computer delivers instructions directly to students and permits them to interact with the computer through the lessons programmed in the system. There are various instructional modes which can be facilitating by Computer Assisted
Instructions (CAI). The teacher can use any one or combination of the following modes in the classroom for the better understanding of their subjects:

i. Tutorial Mode  
ii. Drill and Practice Mode  
iii. Discovery Mode  
iv. Gaming Mode  
v. Simulation Mode  
vi. Problem – Solving Mode

1.11. INTELLIGENT TUTORING SYSTEMS (ITS)

Computer technologies are changing the practice of research and business, and very slowly the content and practice of education are beginning to follow suit. Imparting basic cognitive skills of teachers to computers, we might delegate some teaching to machines and thus improve educational outcomes.

By looking at the development of techniques and algorithms for CAI, developed over the last 30 years, we see that they have improved on the quality of feedback and the degree of individualized instruction. CAI systems seem to have improved beyond expectation in computational sophistication from their humble beginnings of replacing the programmed learning machines. However, they fall far short of being any match for human teachers. Human tutors unlike CAI tutor seem to know almost intuitively what the best way to teach individual learners is. The main problem is the impoverishment of knowledge, which they contain. In generative systems there is a mismatch between the program’s internal processes (Boolean Arithmetic) and those of the student’s cognitive processes.

The ability to learn is central to human intelligence. This ability permits us to adapt to changing environment, to develop a great variety of skills, acquire expertise in almost unlimited number of specific domains. People are capable of learning from information carried by multiple physical media and expressed in an unbounded variety of forms. This information can be stated at different level of abstraction, with different
degrees of precision, with or without errors and with different degrees of relevancy to the
knowledge ultimately acquired. – Michalski

The hallmark of artificial intelligence applications in education is that they attempt to explicitly represent some of the reasoning skills and knowledge of expert practitioners, and to exploit that expertise for teaching and learning. The learning systems making use of artificial intelligence techniques are referred as intelligent tutoring systems. These systems continually assess a student’s knowledge, monitor each response, analyse answers or solutions to problems, detect misconceptions and learning difficulties and depending on all this information, provide remedial help and highly individualized instruction. The addition of intelligence makes the CAI more responsive and interactive than traditional CAI. The notion is that for many subjects such as medicine, a student will learn better with a personalized tutor i.e. intelligence CAI that has some sort of intelligence also available with it. This will help in modeling the student learning pattern i.e. the tutor can understand how the student is progressing and why the student is making errors.

Therefore ITS is a program designed to teach as a good teacher would. They concentrate on developing systems which provide supportive environment for more limited topics i.e. a specific domain of knowledge such as antibiotics, solution of quadratic equations, electronic fault diagnostic procedures etc. Since the ITS are domain specific, the subject domain should be chosen carefully as its development involves major investment of resources. Since the efforts have been in the direction of developing experimental ITS, they are designed with future technology in mind and hence required substantial computing resources to run. This makes the ITS at present a cost - effective proposition.

1.11.1 Components

Although, ITS differs in a variety of ways, most have characteristic structure. There are four components to any ITS. They are
i. The specific domain
ii. The learner or the student
iii. The instructional strategies
iv. How the strategies should be adapted to the needs of an individual learner/student

The heart of an ITS is its expert system. The expert system embeds sufficient knowledge of a particular topic area to provide “ideal” answers to questions, correct not only in the final result but in each of the smaller intermediate reasoning steps. The expert system thus allows the ITS to demonstrate or model a correct way of solving the problem. For e.g. if a student needs an explanation of why or how an algebra ITS did a step in solving an equation, the system might first say that it used the distributive rule. If the student requested more justification, it could elaborate by describing the terms that were distributed and the arithmetic “cleanup” steps that followed. Explanations thus turn exert systems from opaque “black box” experts into in spectacle “glass box” (Foss, 1987). A strong ITS is thus an expert which is able to solve a problem along with the student, recognize partially correct as well as irrelevant responses, judge when and how much assistance should be offered when a student gets stuck.

The difference in ITS and CAI reflect engineering and psychological enhancements that permits ITS to tutor in a knowledge-based fashion. Unlike previous CAI systems, ITS represent some of the knowledge and reasoning of good one-on-one human teachers, and consequently can coach in a much more detailed way than CAI systems.

1.11.2 Role in Teaching

In principle, ITS offers a highly interactive environment that is responsive to individual learner’s needs. Like a good teacher it acts as a facilitator and a discerning advisor who determines what the student knows and guides in a manner that the student remains in control and is always at the centre of learning. However, all this requires accurate and extremely refined understanding of characteristics of learners as well as of the domain and the methodology of instruction. As ITS within a teaching environment
therefore fulfills two broad roles: advisory and instructional. In its advisory role, the system is used to provide advice about particular courses of action relevant to any given situation that might arise during CAI-oriented activity. For e.g., ITS might be used to analyze student behavioral data and advise an instructor or student appropriate counseling. In its instructional role, an ITS is able to make explicit both its knowledge and its mechanism of inference in order to aid student’s learning activity. This can be achieved in two ways: through the use of specially designed human-computer didactic dialogues or by means of knowledge driven course ware. This course ware is teaching software that calls upon one or more knowledge bases in order to derive its instructional material and strategies.

ITS is a coach as well as model expert problem solving. In particular, they can monitor the student as he or she solves a problem and can determine if every step is right. Thus, while questions were the atomic unit of discourse in CAI systems, in IT’S the basic unit is the individual reasoning step. To support this detailed coaching, ITS often create and update a student model. The student model reflects the correct rules the ITS thinks the student knows-ones that are also found in the expert system or in the “ideal” student model- then attempts to remediate it with every detailed advice about how the expert system would do the step. This process repeats at every step in the evolution of a complete solution to a problem.

One of the parts of most ITS receive relatively little mention is the pedagogical component. While the expert system contains rules and knowledge that driven an ITS subject – specific performance, the pedagogical component is supposed to contain similar rules that encode expertise about tutoring itself – for example, when to interrupt students and what kinds of information to provide them.

1.12 COMPUTERS IN EDUCATION

The computer is now regarded as super teaching machine. Its use in education has been tried as an innovation and it has proved its teaching efficiency in many developed countries. The computer has been helping the teacher in the following areas
i. Evaluation of students’ performance and classification of children according to abilities.

ii. Preparation of timetable and schedules

iii. Allocation of learning materials according to individual needs and interest

iv. Maintenance of progress cards efficiently and confidentially

v. Providing information or data for guidance and preference

vi. Provision of direct interaction between pupils and subject matter

The computer's ability to perform logical operation is a major characteristic and must surely be central to any computer application. In the context of learning, the rapid response to a learner’s action is of particular benefit as there can be quick reinforcement of good ideas which the learner has and any misconceptions may be corrected. Many motor skills can only be learnt by direct use of the equipment concerned.

To sum up, the electronic revolution in the field of science and technology has paved the way of outcome of Digital Technology, and the Digital Technology has given rise to so many new modern Information and Communication Technology for the benefit of the human society. Information Technology is useful for information handling and information processing. It is the fastest growing technology. The increased availability of CD-ROM produces, electronic publication, internet access and training activities in the country have focused on the application of new technologies in educational institutions.

Using these technologies, retrieval of information is much faster and easier compared to the manual system. Information Technology can play a crucial role in the effective learning process of students engaged in higher studies. The growth of knowledge in science and technology was estimated to be doubling every ten years in 1950’s and every five years in 1970’s. Now it is estimated to have reached the size of doubling every two years or less. Bibliographic Instruction through Computer Assisted Instruction (CAI) package can be used, as effective leaning aids for imparting discipline wise instruction to students in various subjects like Physics, Chemistry, Mathematics and
Computer Science. The use of computers and Internet, and the changing trends in technology particularly the World Wide Web is a blessing for the teachers.