B.Tech
SOE, CUSAT

SOE, CUSAT

1101B / 1201A
Computer Programming

Chapter 1

Module I

1.1 Basics for Computer and Information Technology

1.1.1 Digital Computer System

1.1.1.1 What is a Computer?

A computer is a general-purpose device that can be programmed to carry out a set of arithmetic or logical operations automatically. Since a sequence of operations can be readily changed, the computer can solve more than one kind of problem.
A computer consists of at least one processing element, typically a central processing unit (CPU), and some form of memory. The processing element carries out arithmetic and logic operations, and a sequencing and control unit can change the order of operations in response to stored information. Peripheral devices allow information to be retrieved from an external source, and the result of operations saved and retrieved.
Advantages

1. High Speed
2. Accuracy
3. Storage Capability
4. Diligence - It can work continuously without any error and boredom. It can do repeated work with same speed and accuracy.
5. Versatility - A computer is very flexible in performing the jobs to be done.
6. Reliability - Computers are designed to make maintenance easy.
7. Automation - Automation means ability to perform the given task automatically. Once a program is given to computer i.e., stored in computer memory, the program and instruction can control the program execution without human interaction.
8. Reduction in Paper Work - The use of computers for data processing in an organization leads to reduction in paper work and results in speeding up a process. As data in electronic files can be retrieved as and when required, the problem of maintenance of large number of paper files gets reduced.
9. Reduction in Cost - Though the initial investment for installing a computer is high but it substantially reduces the cost of each of its transaction.

Disadvantages

1. No intelligence - A computer cannot take any decision on its own.
2. Dependency - It functions as per a user's instruction, so it is fully dependent on human being

3. Environment - The operating environment of computer should be dust free and suitable.
4. No Feeling - Computers have no feelings or emotions. It cannot make judgment based on feeling, taste, experience, and knowledge unlike a human being.

1.1.1.2 Five generations of computers

- **First Generation**: 1946-1959. Vacuum tube based. E.g.: EDVAC, MARK II, MARK III, UNIVAC I, UNIVAC II, BURROUGHS
- **Second Generation**: 1959-1965. Transistor based. E.g.: UNIVAC III, 7070, CDC 1604, FORTRAN, COBOL, ALGOL
- **Third Generation**: 1965-1971. Integrated Circuit based. E.g.: UNIVAC 1100, PDP 11
- **Fourth Generation**: 1971-1980. VLSI microprocessor based.
- **Fifth Generation**: 1980-onwards. ULSI microprocessor based.

1.1.1.3 Categories of Computers

1. **Supercomputers**
   It's a term used to describe computers that have the most capable processing power of its time. Modern supercomputers run hundreds of thousands of processors, capable of computing quadrillions of calculations in just a few nanoseconds. Supercomputers are used in computational science to calculate and carry out a plethora of complex tasks. Modeling molecular structures, weather forecasting, and the field of quantum mechanics, among others, rely on supercomputers and their intense processing power to solve their equations. Supercomputers are measured in FLOPS, or floating point operations per seconds. Popular Supercomputers are
   - IBM's Sequoia, in United States
   - Fujitsu's K Computer in Japan
   - IBM's Mira in United States
   - IBM's SuperMUC in Germany
   - NUDT Tianhe-1A in China
2. **Mainframe Computers**
   Like supercomputers, mainframe computers are huge, towering machines with lots of processing power. Mainframe computers are mostly used by corporations, government agencies, and banks - organizations that need a way to store large quantities of information. They are not the same as supercomputers. The processing capabilities of mainframe computers are measured in MIPS, or millions of instructions per second. Popular Mainframe computers are
   - Fujitsu's ICL VME
   - Hitachi's Z800
3. **Minicomputers**
   A minicomputer is a multiprocessing machine that can support up to about 200 users at the same time. It's like a less powerful mainframe computer, and is about the size of a refrigerator. A server can be an example of a minicomputer, but not all servers are minicomputers. Despite their name, a minicomputer is not a personal computer like the desktop machine you might have at home or work. They are much larger than that. Popular Minicomputers are
   - K-202
   - Texas Instrument TI-990
   - SDS-92

- IBM Midrange computers

4. **Microcomputers**
   Microcomputers are the ones people are most familiar with on a daily, non-professional basis, but of course that doesn't mean they're exclusive to the home. Microcomputers are smaller computers that run on microprocessors in their central processing units. They are much, much cheaper than supercomputers, mainframe computers and even minicomputers, because they're meant for everyday uses that are more practical than professional. The range of capabilities for microcomputers are still vast, though. A film editor might use a microcomputer to run many intensive editing programs at once, while a student might use a microcomputer for Facebook and word processing.

   - **Desktop computers**: Desktop are popular for the user's ability to customize them, replace parts and fix them with much more ease than they would a laptop. It's also more convenient to be able to connect peripherals like screens and keyboard and computer mice that fit your needs. In this sense, desktop computers could be used at the office for professional tasks, or at the home. Desktop computers can be specialized for things like gaming as well, equipped with high-end graphics cards and more RAM.
   - **Video game consoles**: They have many of the same hardware components as computers, but are usually less advanced, which is why they're able to cost much less than a top-notch gaming computer.

5. **Mobile Computers**
   Mobile computers usually describe computers that are meant to be carried around and taken from place to place.

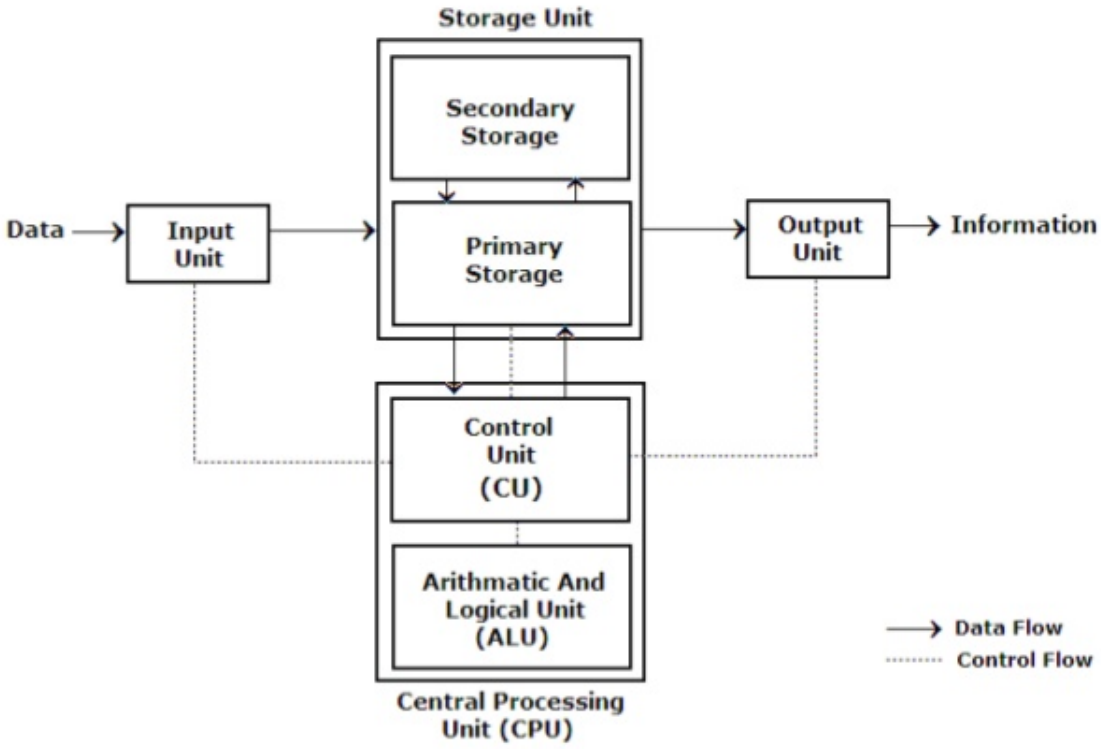
   - **Laptops**: Portable computers designed to be carried from place to place. All of its components are contained inside a panel that functions also as the keyboard, with an attached screen that can be folded over. Because of their size and convenience, these are some of the most popular computers for everyday use.
   - **Netbooks**: Much smaller laptops.
   - **Tablet**: A flat, keyboard-less screen that utilizes touch-screen functionality for navigation and use.
   - Handheld game console: Devices like the Game Boy, Game Boy Color, Game Boy Advance, Sega Nomad, PlayStation Portable (PSP), and PlayStation Vita are handheld game consoles.
   - **Calculators**: Basic calculators, graphic calculators, scientific calculators, programmable calculators, and calculators used for accounting and other financial purposes.
   - **Portable media players**: Also known as MP3 players.
   - **Cellphones** and **smartphones**

1.1.2 Hardware and Software

1.1.2.1 Digital Computer Organization (Working of a digital computer with diagram)

- **Input** is given through the input devices to CPU.
- **CPU** consists of three basic units: Control unit, Arithmetic Logical Unit (ALU) and memory unit.
- **Control unit** controls communication within ALU and memory unit and decides which circuit is to be activated. For reading instruction it uses Fetch-execute mechanism. Control unit gets instruction from memory. Control unit decides what to do of that instruction and transfers it to the ALU.
- **ALU** performs various arithmetic operations like addition, subtraction, multiplication, division and logical operations like AND, OR, NOT, NAND etc. on that instruction. Results of ALU are stored in the memory or resistor for its further operations.
- After completing the instruction, stored results are passed to the **output** devices.

- To synchronize all these operations CPU uses its own system clock. CPU executes stored instructions called as program. It tells rest of the computer system what to do. It executes arithmetic calculation and data manipulation. It holds data and instruction which are in the current use. It is responsible for storing and retrieving information on disks and other media.



1.1.2.2 Hardware

Hardware represents the physical and tangible components of a computer i.e. the components that can be seen and touched. Examples of Hardware are following:

1. Input devices – keyboard, mouse etc.
2. Output devices – printer, monitor etc.
3. Secondary storage devices – Hard disk, CD, DVD etc.
4. Internal components – CPU, motherboard, RAM etc.

Input Devices Input devices allow us to interact with computers in a variety of ways. Most input devices come in the form of peripherals, and can be divided into subcategories of their own: visual, audio, etc. Some examples of input devices include computer mice, keyboards, scanners, copy machines, webcams, microphones, MIDI keyboards, game pads and controllers, and so on. Basically, anything you can plug into your computer that lets you input information or carry out tasks is an input device.

Output Devices Output devices are the opposite of input devices. These are peripherals that essentially allow the computer to interact with us âĂŞ they display information that we need to make decisions about how we interface with them. In more technical terms, output devices convey results from processes run by the computer. The most obvious example of an output device is a computer monitor, or screen, because this lets us see the various processes our computers are running in a substantial, visual manner. Other output devices include headphones and speakers, which convey sound, printers, and even CDs, which we can tell the computer to output information onto.

Processing Devices Processing devices are the devices that enable the computer to process information. All computers have some form of a processor. The computer has a CPU, or central processing unit, embedded onto its motherboard. This is where the computer carries out its complex calculations, processes info, and sends it out to its output devices to be conveyed to you in a manner that makes sense. CPU consists of the following features:

- CPU is considered as the brain of the computer.
- CPU performs all types of data processing operations.
- It stores data, intermediate results and instructions (program).
- It controls the operation of all parts of computer.

CPU itself has following three components.

1. Memory or Storage Unit
2. Control Unit
3. ALU (Arithmetic Logic Unit)

Control Unit This unit controls the operations of all parts of computer but does not carry out any actual data processing operations. Functions of this unit are:

- It is responsible for controlling the transfer of data and instructions among other units of a computer.
- It manages and coordinates all the units of the computer.
- It obtains the instructions from the memory, interprets them, and directs the operation of the computer.
- It communicates with Input/Output devices for transfer of data or results from storage.
- It does not process or store data.

ALU (Arithmetic Logic Unit) This unit consists of two subsections namely

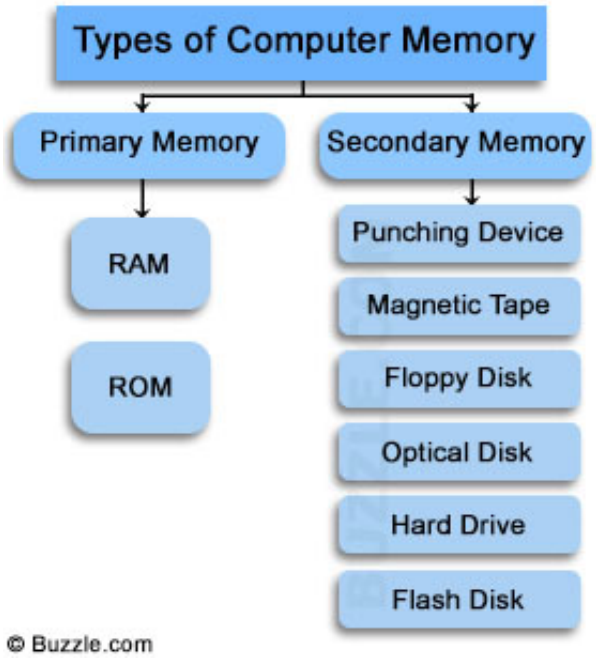
1. **Arithmetic section** - Function of arithmetic section is to perform arithmetic operations like addition, subtraction, multiplication and division. All complex operations are done by making repetitive use of above operations.

2. **Logic Section** - Function of logic section is to perform logic operations such as comparing, selecting, matching and merging of data.

Memory or Storage Unit This unit can store instructions, data and intermediate results. This unit supplies information to the other units of the computer when needed. It is also known as internal storage unit or main memory or primary storage or Random access memory (RAM).
Its size affects speed, power and capability. Primary memory and secondary memory are two types of memories in the computer. Functions of memory unit are:

- It stores all the data and the instructions required for processing.
- It stores intermediate results of processing.
- It stores final results of processing before these results are released to an output device.
- All inputs and outputs are transmitted through main memory.

Computer memory is the storage space in computer where data is to be processed and instructions required for processing are stored. The memory is divided into large number of small parts called cells. Each location or cell has a unique address which varies from zero to memory size minus one. For example if computer has $64k$ words, then this memory unit has $64 * 1024 = 65536$ memory locations. The address of these locations varies from 0 to 65535.



Memory is primarily of three types

1. **Cache Memory** is a very high speed semiconductor memory which can speed up CPU. It acts as a buffer between the CPU and main memory. It is used to hold those parts of data and program which are most frequently used by CPU. The parts of data and programs are transferred from disk to cache memory by operating system, from where CPU can access them.
   Advantages
   - Cache memory is faster than main memory.
   - It consumes less access time as compared to main memory.
   - It stores the program that can be executed within a short period of time.
   - It stores data for temporary use.

   Disadvantages
   - Cache memory has limited capacity.

- It is very expensive.

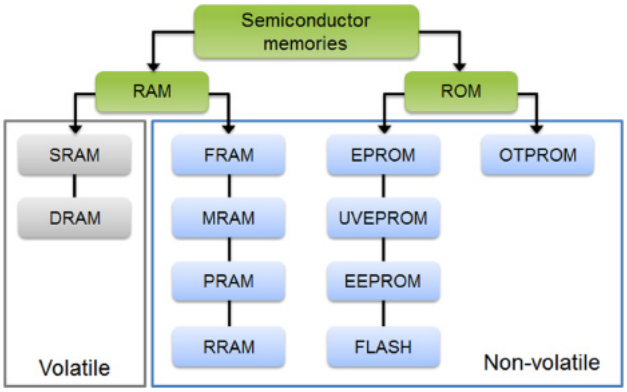
2. **Primary Memory/Main Memory** - Primary memory holds only those data and instructions on which computer is currently working. It has limited capacity and data is lost when power is switched off. It is generally made up of semiconductor device. These memories are not as fast as registers. The data and instruction required to be processed reside in main memory. It is divided into two subcategories RAM and ROM.
   Characteristics of Main Memory
   - These are semiconductor memories
   - It is known as main memory.
   - Usually volatile memory.
   - Data is lost in case power is switched off.
   - It is working memory of the computer.
   - Faster than secondary memories.
   - A computer cannot run without primary memory.

   (a) **RAM (Random Access Memory)** is the internal memory of the CPU for storing data, program and program result. It is read/write memory which stores data until the machine is working. As soon as the machine is switched off, data is erased.
   Access time in RAM is independent of the address that is, each storage location inside the memory is as easy to reach as other locations and takes the same amount of time. Data in the RAM can be accessed randomly but it is very expensive.
   RAM is volatile, i.e. data stored in it is lost when we switch off the computer or if there is a power failure. Hence a backup uninterruptible power system (UPS) is often used with computers. RAM is small, both in terms of its physical size and in the amount of data it can hold.
   RAM is of two types

   i. **Static RAM (SRAM)** - The word static indicates that the memory retains its contents as long as power is being supplied. However, data is lost when the power gets down due to volatile nature. SRAM chips use a matrix of 6-transistors and no capacitors. Transistors do not require power to prevent leakage, so SRAM need not have to be refreshed on a regular basis. Because of the extra space in the matrix, SRAM uses more chips than DRAM for the same amount of storage space, thus making the manufacturing costs higher. So SRAM is used as cache memory and has very fast access.
   Characteristic of the Static RAM
      - It has long life
      - There is no need to refresh
      - Faster
      - Used as cache memory
      - Large size
      - Expensive
      - High power consumption

   ii. **Dynamic RAM (DRAM)** - DRAM, unlike SRAM, must be continually refreshed in order to maintain the data. This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second. DRAM is used for most system memory because it is cheap and small. All DRAMs are made up of memory cells which are composed of one capacitor and one transistor.
   Characteristics of the Dynamic RAM
      - It has short data lifetime
      - Need to be refreshed continuously
      - Slower as compared to SRAM
      - Used as RAM

- Lesser in size
- Less expensive
- Less power consumption



(b) **ROM (Read Only Memory)**. The memory from which we can only read but cannot write on it. This type of memory is non-volatile. The information is stored permanently in such memories during manufacture. A ROM, stores such instructions that are required to start a computer. This operation is referred to as bootstrap. ROM chips are not only used in the computer but also in other electronic items like washing machine and microwave oven. Advantages of ROM

- Non-volatile in nature
- These cannot be accidentally changed.
- Cheaper than RAMs
- Easy to test
- More reliable than RAMs
- These are static and do not require refreshing.
- Its contents are always known and can be verified.

Following are the various types of ROM

MROM (Masked ROM) - The very first ROMs were hard-wired devices that contained a pre-programmed set of data or instructions. These kind of ROMs are known as masked ROMs which are inexpensive.

PROM (Programmable Read only Memory) - PROM is read-only memory that can be modified only once by a user. The user buys a blank PROM and enters the desired contents using a PROM program. Inside the PROM chip there are small fuses which are burnt open during programming. It can be programmed only once and is not erasable.

EPROM (Erasable and Programmable Read Only Memory) - The EPROM can be erased by exposing it to ultra-violet light for a duration of up to 40 minutes. Usually, an EPROM eraser achieves this function. During programming, an electrical charge is trapped in an insulated gate region. The charge is retained for more than ten years because the charge has no leakage path. For erasing this charge, ultra-violet light is passed through a quartz crystal window (lid). This exposure to ultra-violet light dissipates the charge. During normal use the quartz lid is sealed with a sticker.

EEPROM (Electrically Erasable and Programmable Read Only Memory) - The EEPROM is programmed and erased electrically. It can be erased and reprogrammed about ten thousand times. Both erasing and programming take about 4 to 10 ms (milli second). In EEPROM, any location can be selectively erased and programmed. EEPROMs can be erased one byte at a time, rather than erasing the entire chip. Hence, the process of re-programming is flexible but slow.

3. **Secondary Memory** - This type of memory is also known as external memory or non-volatile. It is slower than main memory. These are used for storing data / information permanently. CPU directly does not access these memories instead they are accessed via input-output routines.

Contents of secondary memories are first transferred to main memory, and then CPU can access it. E.g.: disk, CD-ROM, DVD etc.
Characteristic of Secondary Memory

- These are magnetic and optical memories
- It is known as backup memory.
- It is non-volatile memory.
- Data is permanently stored even if power is switched off.
- It is used for storage of data in a computer.
- Computer may run without secondary memory.
- Slower than primary memories.

Standard personal computer will come with a primary storage device, which is a storage device that is a part of the hardware itself. This includes RAM and processor registers or caches. Computer's central processing unit is continuously retrieving and scanning data from these storage devices, and computing as required. On the other hand, secondary storage devices are a more permanent means of storage, that do not require the central processing unit to be running in order to save its information. Devices like a hard drive, drive, and other external storage devices like USB flash drives, DVDs, CDs, and floppy disks, are all secondary storage devices.

The memory is measured in the unit called byte. $1byte = 8bits$.

Sl No.	Unit	Abbreviation	Description
1	Kilobyte	KB	1 KB = 1024 bytes
2	Megabyte	MB	1 MB = 1024 KB
3	GigaByte	GB	1 GB = 1024 MB
4	TeraByte	TB	1 TB = 1024 GB
5	PetaByte	PB	1 PB = 1024 TB

1.1.2.3 Software

Computer software or simply software is any set of machine-readable instructions that directs a computer's processor to perform specific operations. Computer software contrasts with computer hardware, which is the physical component of computers. Computer software can be divided into: (Categories of software)

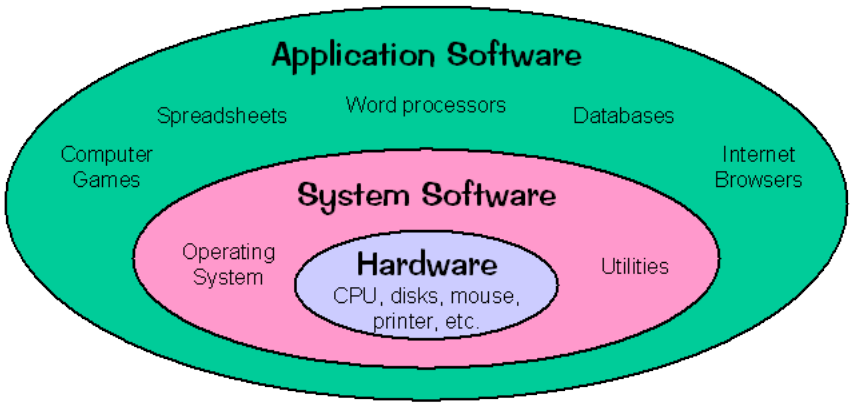
1. **Application software**
   Application software which uses the computer system to perform special functions or provide entertainment functions beyond the basic operation of the computer itself. There are many different types of application software, because the range of tasks that can be performed with a modern computer is so large.
   Application software products are designed to satisfy a particular need of a particular environment. All software applications prepared in the computer lab can come under the category of Application software.
   Application software may consist of a single program, such as a Microsoft's notepad for writing and editing simple text. It may also consist of a collection of programs, often called a software package, which work together to accomplish a task, such as a spreadsheet package.
   Examples of Application software are following:

   - Software
   - Student Record Software
   - Inventory Management Software
   - Income Tax Software
   - Railways Reservation Software
   - Microsoft Word

- Microsoft Excel
- Microsoft PowerPoint
- Payroll

Features of application software are as follows:

(a) Close to user
(b) Easy to design
(c) More interactive
(d) Slow in speed
(e) Generally written in high-level language
(f) Easy to understand
(g) Easy to manipulate and use
(h) Bigger in size and requires large storage space



2. **System software** System software, which is designed to directly operate the computer hardware, to provide basic functionality needed by users and other software, and to provide a platform for running application software.
The system software is collection of programs designed to operate, control, and extend the processing capabilities of the computer itself. System software are generally prepared by computer manufactures. These software products comprise of programs written in low-level languages which interact with the hardware at a very basic level. System software serves as the interface between hardware and the end users.
Features of system software are as follows:

- Close to system
- Fast in speed
- Difficult to design
- Difficult to understand
- Less interactive
- Smaller in size
- Difficult to manipulate
- Generally written in low-level language

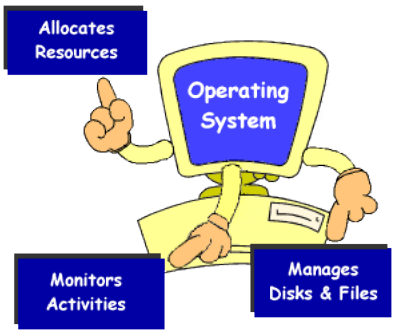
Some examples of system software are Operating System, Compilers, Interpreter, and Assemblers etc.

Operating systems which are essential collections of software that manage resources and provides common services for other software that runs "on top" of them. Supervisory programs, boot loaders, shells and window systems are core parts of operating systems. In practice, an operating system comes bundled with additional software (including application software) so that a user can potentially do some work with a computer that only has an operating system. An operating system has three main responsibilities:

(a) Perform basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers.

(b) Ensure that different programs and users running at the same time do not interfere with each other.

(c) Provide a software platform on top of which other programs (i.e., application software) can run.

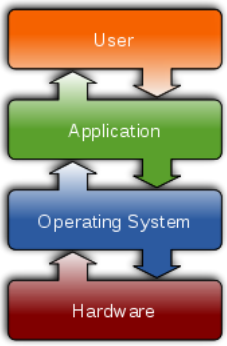


Device drivers which operate or control a particular type of device that is attached to a computer. Each device needs at least one corresponding device driver; because a computer typically has at minimum at least one input device and at least one output device, a computer typically needs more than one device driver.

Utilities which are computer programs designed to assist users in maintenance and care of their computers.

1.1.2.4 Relationship between Hardware and Software

Hardware and software are mutually dependent on each other. Both of them must work together to make a computer produce a useful output.



- Software cannot be utilized without supporting hardware.
- Hardware without set of programs to operate upon cannot be utilized and is useless.
- To get a particular job done on the computer, relevant software should be loaded into the hardware
- Hardware is a one-time expense.
- Software development is very expensive and is a continuing expense.

- Different software applications can be loaded on a hardware to run different jobs.
- A software acts as an interface between the user and the hardware.
- If hardware is the 'heart' of a computer system, then software is its 'soul'. Both are complimentary to each other.

1.1.3 Applications of Computers

Business A computer has high speed of calculation, diligence, accuracy, reliability, or versatility which made it an integrated part in all business organizations. Computer is used in business organizations for:

- Payroll calculations
- Budgeting
- Sales analysis
- Financial forecasting
- Managing employees database
- Maintenance of stocks etc.

Banking
- Banks provide online accounting facility, which includes current balances, deposits, overdrafts, interest charges, shares, and trustee records.
- ATM machines are making it even easier for customers to deal with banks.

Insurance Insurance companies are keeping all records up-to-date with the help of computers. The insurance companies, finance houses and stock broking firms are widely using computers for their concerns. Insurance companies are maintaining a database of all clients with information showing

- starting date of the policies
- next due installment of a policy
- maturity date
- interests due
- survival benefits
- bonus
- procedure to continue with policies

Education The computer has provided a lot of facilities in the education system. The computer provides a tool in the education system known as CBE (Computer Based Education).

- CBE involves control, delivery, and evaluation of learning.
- The computer education is rapidly increasing the graph of number of computer students.
- There are number of methods in which educational institutions can use computer to educate the students.
- It is used to prepare a database about performance of a student and analysis is carried out on this basis.

Marketing .

- Advertising - With computers, advertising professionals create art and graphics, write and revise copy, and print and disseminate ads with the goal of selling more products.
- At Home Shopping - Home shopping has been made possible through use of computerized catalogues that provide access to product information and permit direct entry of orders to be filled by the customers.

Health Care Computers have become important part in hospitals, labs, and dispensaries. The computers are being used in hospitals to keep the record of patients and medicines. It is also used in scanning and diagnosing different diseases. ECG, EEG, Ultrasounds and CT Scans etc., are also done by computerized machines. Some major fields of health care in which computers are used are:

- Diagnostic System - Computers are used to collect data and identify cause of illness.
- Lab-diagnostic System - All tests can be done and reports are prepared by computer.
- Patient Monitoring System - These are used to check patient's signs for abnormality such as in Cardiac Arrest, ECG etc.
- Pharma Information System - Computer checks Drug-Labels, Expiry dates, harmful drug's side effects etc.
- Surgery: Nowadays, computers are also used in performing surgery.

Engineering Design Computers are widely used in engineering purpose. One of major areas is CAD (Computer aided design). That provides creation and modification of images. Some fields are:

- Structural Engineering - Requires stress and strain analysis for design of Ships, Buildings, Budgets, and Airplanes etc.
- Industrial Engineering - Computers deal with design, implementation and improvement of integrated systems of people, materials and equipment.
- Architectural Engineering - Computers help in planning towns, designing buildings, determining a range of buildings on a site using both 2D and 3D drawings.

Military Computers are largely used in defense. Modern tanks, missiles, weapons etc. Military also employs computerized control systems. Some military areas where a computer has been used are:

- Missile Control
- Military Communication
- Military Operation and Planning
- Smart Weapons

Communication Communication means to convey a message, an idea, a picture or speech that is received and understood clearly and correctly by the person for whom it is meant for. Some main areas in this category are:

- E-mail
- Chatting
- Usenet
- FTP
- Telnet
- Video-conferencing

Government Computers play an important role in government. Some major fields in this category are:

- Budgets
- Sales tax department
- Income tax department
- Male/Female ratio
- Computerization of voters lists
- Computerization of driving licensing system
- Computerization of PAN card
- Weather forecasting

1.1.4 Role of Information Technology

In this modern day and age, information technology plays a big role. Information technology is the study and use of systems for storing, retrieving, and sending information. This can include software, hardware, applications, and so much more.
Items as common as cars use information technology. Business, education, and even healthcare have all been redesigned thanks to information technology.

The Importance of Information Technology in Business Using computers and software, businesses use information technology to ensure that their departments run smoothly. They use information technology in a number of different departments including human resources, finance, manufacturing, and security.
Using information technology, businesses have the ability to view changes in the global markets far faster than they usually do. They purchase software packages and hardware that helps them get their job done. Larger businesses have their own information technology department designed to upkeep the software and hardware.
Information technology has allowed businesses to keep up with the supply and demand as consumers grow more anxious to have their items instantly. Using information technology, businesses like Amazon are working to help busy consumers do their grocery shopping. Just a few clicks on a website allows the consumer to submit an order, and information technology sends that order to the company.

The Importance of Information Technology in Education Information technology plays a key role in students being able to keep their jobs and go to school. Now, most schools offer online classes that can be accessed on computers or laptops, tablets, and even mobile phones. A busy student at work can easily check in or submit assignments while on their lunch break.
Using technology, teachers can prepare their students for a future flooded with gadgets including tablets, mobile phones, computers, and so much more.
Information technology is helping to prevent more high school and college dropouts as well. Life events can happen to anyone at any time, and even high schools are taking advantage of online classes so that students can continue their education instead of considering dropping out.

The Importance of Information Technology in Finance Information technology might just working its hardest with internet transactions. As more transactions are done, the internet requires more networks, more computers, and more security programs to keep its consumers safe. Without information technology, these purchases would be impossible, and it would be impossible for banks to keep these purchases secure.
Information technology has also made it faster and easier than ever to send or receive money. It's now also easier to open an online small business to sell whatever you might want. If you don't want to have to file for a domain name, set up a website, and all the other steps required for registering an online business, you can use other websites like Amazon, eBay, and Etsy to sell things.
Information technology also makes it easy for finance to function on a global level. In this modern age, your credit score and credit rating is available online securely. This allows lenders, insurance companies, and businesses to run a quick credit check on you making it far easier to open credit.

The Importance of Information Technology in Healthcare Improvements in information technology have allowed for great reform in healthcare. Most medical offices can now send and receive digital medical information from doctors you've had in the past. Changes like this allow costs to be lowered and increase the amount of time doctors can spend on patients compared to paperwork. Security improvements with information technology have made it so that your medical information is secure no matter where it's sent.
Information technology has also updated the technology a doctor can use to diagnose or treat you. Using computerized axial tomography (CAT) or magnetic resonance imaging (MRI) scans, the doctor can use a computer to create detailed images of your organs along with creating ; images that show changes in your body chemistry and blood flow. This can be helpful to find illnesses that aren't found with blood tests or other medical tests.

The Importance of Information Technology in Security With so many transactions done online and so much information available online, it's important to keep all of that safe. Information technology makes it possible for your online data to stay secure until accessed by the proper channels. Using passwords and encryption, information technology hides your personal digital data away, and the only way it can be accessed is by companies who have permission from you.

1.1.4.1 Analog, Digital, Hybrid computers

Analog Computers Analog computers are used to process analog data. Analog data is of continuous nature and which is not discrete or separate. Such type of data includes temperature, pressure, speed weight, voltage, depth etc. These quantities are continuous and having an infinite variety of values. It measures continuous changes in some physical quantity e.g. The Speedometer of a car measures speed, the change of temperature is measured by a Thermometer, the weight is measured by Weights machine. These computers are ideal in situations where data can be accepted directly from measuring instrument without having to convert it into numbers or codes.
Analog computers are the first computers being developed and provided the basis for the development of the modern digital computers. Analog computers are widely used for certain specialized engineering and scientific applications, for calculation and measurement of analog quantities. They are frequently used to control process such as those found in oil refinery where flow and temperature measurements are important. They are used for example in paper making and in chemical industry. Analog computers do not require any storage capability because they measure and compare quantities in a single operation. Output from an analog computer is generally in the form of readings on a series of dial (Speedometer of a car) or a graph on strip chart.

Digital Computers A Digital Computer works with digits to represent numerals, letters or other special symbols. Digital Computers operate on inputs which are ON-OFF type and its output is also in the form of ON-OFF signal. Digital computers process information which is based on the presence or the absence of an electrical charge or we prefer to say a binary 1 or 0.
A digital computer can be used to process numeric as well as non-numeric data. It can perform arithmetic operations like addition, subtraction, multiplication and division and also logical operations. Most of the computers available today are digital computers. The most common examples of digital computers are accounting machines and calculators.
The results of digital computers are more accurate than the results of analog computers. Analog computers are faster than digital. Analog computers lack memory whereas digital computers store information.

Hybrid Computers A hybrid is a combination of digital and analog computers. It combines the best features of both types of computers. It has the speed of analog computer and the memory and accuracy of digital computer. Hybrid computers are used mainly in specialized applications where both kinds of data need to be processed. Therefore, they help the user, to process both continuous and discrete data. In hospital Intensive Care Unit (ICU), an analog device is used which measures patient's blood pressure and temperature etc., which are then converted and displayed in the form of digits. Hybrid computers for example are used for scientific calculations, in defense and radar systems.

1.1.5 Internet Services

The Internet is a global system of interconnected computer networks that use the **Internet protocol suite (TCP/IP)** to link several billion devices worldwide. It is a network of networks.
The origin of the Internet is ARPANET commissioned by the United States government in the 1960s .
The Internet is a globally distributed network comprising many voluntarily interconnected autonomous networks. It operates without a central governing body.
The technical aspects and standardization of the core protocols (**IPv4 and IPv6**) is undertaken by *Internet Engineering Task Force (IETF)*.
To maintain inter-operability, the principal name spaces of the Internet are administered by the *Internet*

Corporation for Assigned Names and Numbers (ICANN). ICANN is the authority that coordinates the assignment of unique identifiers for use on the Internet, including domain names, Internet Protocol (IP) addresses, application port numbers in the transport protocols.

1.1.5.1 World Wide Web

WWW was developed as a hypertext system at the Center for Nuclear Energy Research (CERN) in Geneva. The Web is a global set of documents, images and other resources, logically interrelated by hyperlinks and referenced with **Uniform Resource Identifiers (URIs)**. URIs symbolically identify services, servers, and other databases, and the documents and resources that they can provide. **Hypertext Transfer Protocol (HTTP)** is the main access protocol of the World Wide Web.
World Wide Web browser software, such as Microsoft's Internet Explorer, Mozilla Firefox, Opera, Apple's Safari, and Google Chrome, lets users navigate from one web page to another via **hyperlinks** embedded in the documents. These documents may also contain any combination of computer data, including graphics, sounds, text, video, multimedia and interactive content that runs while the user is interacting with the page.

1.1.5.2 Telnet

Telnet is an Internet service that allows the user logs on and run on a remote computer and use programs installed on the remote computer. The remote computer must be enabled in the user access rights to the computer.
Today's programs use **SSH** (Secure Shell) connections over TCP / IP as the carrier of connection, but all traffic is encrypted.

1.1.5.3 Electronic Mail

Email is an important communications service available on the Internet. The concept of sending electronic text messages between parties in a way analogous to mailing letters or memos. Pictures, documents and other files are sent as email attachments. Emails can be carbon copied to multiple email addresses.
Internet telephony is another common communications service made possible by the creation of the Internet. **VoIP** stands for Voice-over-Internet Protocol, referring to the protocol that underlies all Internet communication.

1.1.5.4 FTP - File Transfer Protocol

FTP is a method to transfer large files between computers of Internet users. Allows access to a computer on the Internet (if it's public or available to the password), preview content on its hard disk, finding the necessary files and copy on its own disk.

1.1.5.5 Chat

This service allows the Internet to one or more of Internet users who are hooked to the same channel at the same time see the text that you type on your computer, and the texts of other active users typing on his computer. This system is much faster than e-mail because it is conducted almost simultaneously send and read messages. It is much more expensive than e-mail, because it requires a long term connection with the Internet

1.1.5.6 Newsgroups

As part of this service, it is possible to follow the discussion on any topic or even start a new topic for discussion. Similar to the e-mail message is sent to all members of the group, and everyone signed on the group read the message and optionally answer them.

1.1.5.7 Gopher/WAIS/Archie/Veronica

Gopher is a server that helps you find information on the Internet using menus. Gopher began as the University of Minnesota's version of PennInfo, a menu-driven campus-wide information system (CWIS). By providing a menu of all Gopher servers that any other Gopher could access, an inter-operating set of information systems linking several hundred organizations around the world, all with a common user interface, was made available.
The next step in Gopher's evolution was addition of gateways to FTP, Telnet , Archie, WAIS, and WWW. Gopher was thus transformed from an integrated set of CWIS programs into the most successful Internet navigation tool.

WAIS - Wide Area Information Servers searches for documents on the Internet. Wide Area Information System provides a uniform interface to many full-text databases, together with a sophisticated "relevance search" capability. WAIS databases are commonly collections of related data , primary source documents , or reference works . There are currently almost 400 WAIS databases, and new ones appear frequently. Since it can be difficult to determine the focus of a WAIS database from its name, a Directory of Servers, itself a WAIS database, was developed.

Archie servers that contain a list of files that can be downloaded using FTP . Archie (ARCHIvE server) was developed at McGill University to index the contents of all FTP servers and provide keyword searching of the index. Every night it re-indexes roughly one thirtieth of the servers; the result is a database that is completely refreshed each month.

Veronica is a program to search Gopherspace, the world of Gopher menus. A search tool, Veronica, was contributed to Gopher. It provides a search through all menus using a single keyword. The result is a dynamically created menu of all Gopher resources that contain the keyword in their menus.

1.1.5.8 Voice / Video Communication

Internet networking service that allows users to voice / video communication. This implies that the devices that use this service have built-in camera and microphone. This service should not be confused with VoIP (Voice over Internet Protocol) technology that allows the transmission of voice communications over Internet networks for the purpose of protocol design. Examples of this kind of Peer-to-Peer (P2P) communication includes Skype, Viber, Whatsapp.

1.2 Problem Solving Methodology

1.2.1 Problem Solving Methodology

Problem Definition This is the first step of programming. In the steps the various input and data are identified. There desired output and its format is determined. The problem is defined in a very detailed and precise manner. Sample input data and its corresponding output is gathered.
Knowing the objective is the first consideration. Knowing who the end user will be is also important. Next determine the inputs and outputs. How will the program operate and what data is needed to make it happen. Feasibility is considered next. How many programmers will it take, is the project within budget, does the project have a realistic outline. Finally, if the project is a go, then one must take measures to ensure the project is properly documented and analyzed.
Three mini steps:

- Clarify desired processing
- Double - check feasibility of implementing the program
- Document the analysis

Program Planning Type and format of data at each step is identified. Algorithm / Pseudo code / Flowchart are used to derive at the solution. The aim of the programmer is to create algorithms that

are clear and simple. Algorithms are expressed first in logical hierarchical form known as modularization. Using modules, the programmer creates a logical thought process for the computer to follow. After that the program is broken down in greater detail using pseudo code.
Two mini steps:

- Determine program logic through top down approach and modularization, using a hierarchy chart.
- Design details using pseudo code and/or flowcharts, preferably involving control structure.

A **module**, a processing step of a program, made up of logically related program statements.
A **hierarchy chart**, which represents top-down program design, explains the main purpose of the program.
Pseudo code, a way of designing a program which uses normal language statements in order to describe the logic and the processing flow.
Algorithms are like equations that tell the computer what task to perform.
Program flowcharts, graphically shows the detailed series of steps.
Three control structure are defined.

1. Sequence control structure - No decision making
2. Selection control structure
3. Loop control structure / Repetition or Iteration Structure

Coding In Coding, programmer writes the instructions in a computer language to solve the problem. All coding process depends upon the information we obtained in previous steps. Choice of language depends upon the requirements and facilities available with a language.
After the program has been designed it must be coded or written. An appropriate programming language must be selected. Once the appropriate code language has been chosen, it is imperative that the programmer follow the syntax rules with as little deviation as possible in order for the program to have high accuracy.
Two mini steps:

- Select the appropriate high-level programming language.
- Code the program in that language following the syntax carefully

Debugging / Testing After the program is written it then enters the programming debugging and testing phase of the Program Development Life Cycle (PDLC).
Debug means locate and correct errors or bugs in the source program. There are two types of errors - **logical errors** and **syntax errors**. Debugging will uncover errors in both logic and syntax.
Syntax errors will prevent the program from executing. They can be such simple things as misspelled words or can involve breaking the syntax rules of the programming language used. **Logic errors** will allow the program to run but will provide incorrect results. Errors of this kind may consist of merely using the wrong relational operator or other, larger, mistakes in writing formulas.
Once the programmer locates the errors they are then fixed and the program is run again. This will happen multiple times, often called "execute, check, and correct", until the program runs flawlessly. The program will then enter the testing phase. The program is tested to ensure accuracy and effectiveness of the program. The test is executed on test data.
Testing the program comes in two phases:

- Alpha testing is the process of reading through the program in search of errors in logic. The second step is to run a diagnostic program to search for syntax or input errors.
- Beta testing involves using the program in the real world to see if it contains any bugs or other deficiencies.

Documentation and maintenance The above step are documented clearly. The documents include problem definition, description of input data, flow chart or algorithms, operating instructions, sample test

data and final results. Documentation should be ongoing from the very.
Four mini steps:

- Write user documentation
- Write operator documentation
- Write programmer documentation
- Maintain the program

1.2.2 Design tools

Algorithm An algorithm is a self-contained step-by-step set of operations to be performed. An algorithm is an effective method that can be expressed within a finite amount of space and time and in a well-defined formal language for calculating a function.
Starting from an initial state and initial input, the instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing output and terminating at a final ending state.

Flowchart A flowchart is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem.
Flowcharts are used in analyzing, designing, documenting or managing a process or program. The two most common types of boxes in a flowchart are:

- a processing step, usually called activity, and denoted as a **rectangle**.
- a decision, usually denoted as a **diamond**.

There are four types of flowcharts.

1. Document flowcharts, showing controls over a document-flow through a system.
2. Data flowcharts, showing controls over a data-flow in a system.
3. System flowcharts showing controls at a physical or resource level.
4. Program flowchart, showing the controls in a program within a system

Pseudo code Pseudo code is an informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a programming language, but is intended for human reading rather than machine reading.
Pseudo code typically omits details that are essential for machine understanding of the algorithm, such as variable declarations, system-specific code and some subroutines.
The purpose of using pseudo code is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm. It is commonly used in textbooks and scientific publications that are documenting various algorithms, and also in planning of computer program development, for sketching out the structure of the program before the actual coding takes place.
No standard for pseudo code syntax exists, as a program in pseudo code is not an executable program.

1.2.3 Data and Information

1.2.3.1 What is data?

Data can be defined as a representation of facts, concepts or instructions in a formalized manner which should be suitable for communication, interpretation, or processing by human or electronic machine.
Data is represented with the help of characters like alphabets (A-Z, a-z), digits (0-9) or special characters (+,-,/,*,= etc.).

1.2.3.2 Data Representation

Data are of two types.

1. Numeric data e.g.: numbers.
2. Non-numeric data e.g.: alphabets and symbols

Data can be represented in two ways.

External Data .

- Data used outside computers. Uses alphabets, numbers, and symbols.
- Input and output are in this form.
- External data can be integers or real.
- Integers are used for counting. It is also called Fixed point. It can unsigned / signed. It does not use decimal point or punctuation. E.g. 15, -18, 0, 395.
- Real numbers are used for measuring. It is also called Floating point. It can unsigned / signed. It uses decimal point, but punctuation are not allowed. E.g. 24.45, -48.606. Real numbers can be represented in decimal form or exponential form.
- ASCII (American Standard Code for Information Interchange) is used for representing alphabets and symbols. ASCII encodes 128 specified characters into seven-bit binary integers. The characters encoded are numbers 0 to 9, lowercase letters a to z, uppercase letters A to Z, basic punctuation symbols, control codes and a space.

Internal Data Only binary data i.e. bits (0 and 1). A sequence of bits is used to represent a unit of numeric / non-numeric data.

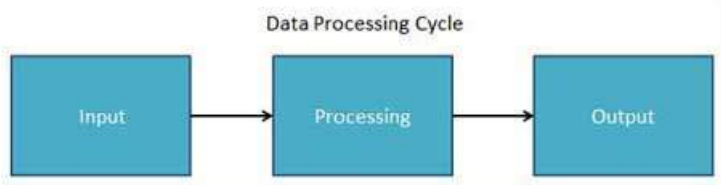
1.2.3.3 What is Information?

Information is organized or classified data which has some meaningful values for the receiver. Information is the processed data on which decisions and actions are based. For the decision to be meaningful, the processed data must qualify for the following characteristics:

- Timely - Information should be available when required.
- Accuracy - Information should be accurate.
- Completeness - Information should be complete.

1.2.3.4 Data Processing Cycle

Data processing is the re-structuring or re-ordering of data by people or machine to increase their usefulness and add values for particular purpose. Data processing consists of basic steps input, processing and output. These three steps constitute the data processing cycle.



Input In this step the input data is prepared in some convenient form for processing. The form will depend on the processing machine. For example, when electronic computers are used, the input data could be recorded on any one of several types of input medium, such as magnetic disks, tapes and so on.

Processing In this step input data is changed to produce data in a more useful form. For example, pay-checks may be calculated from the time cards, or a summary of sales for the month may be calculated from the sales orders.

Output Here the result of the proceeding processing step are collected. The particular form of the output data depends on the use of the data. For example, output data may be pay-checks for employees.

1.2.3.5 Number Systems

"A set of values used to represent different quantities is known as Number System".
Some important number systems are as follows.

1. Decimal number system
   - Base-10.
   - Decimal numbers uses digits from 0..9.
   - These are the regular numbers that we use.
2. Binary number system
   - Base-2.
   - Binary numbers uses only 0 and 1 digits. These are called bits.
   - Binary numbers or bits are used in digital computers.
3. Octal number system
   - Base-8.
   - Octal numbers uses digits from 0..7.
   - Prefix is 0.
4. Hexadecimal number system
   - Base-16.
   - Hex numbers uses digits from 0..9 and A..F.
   - H or 0X or 0x denotes hex prefix.

The decimal number system is used in general. However, the computers use binary number system. The octal and hexadecimal number systems are used in the computer.

1.3 Programming Languages

A **programming language** is a formal constructed language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs to control the behavior of a machine or to express algorithms.
The description of a programming language is usually split into the two components of **syntax** (form) and **semantics** (meaning). Some languages are defined by a specification document (for example, the C programming language is specified by an ISO Standard), while other languages (such as Perl) have a dominant implementation that is treated as a reference.
Programming language defines a vocabulary and set of grammatical rules for instructing a computer to perform specific tasks. The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, FORTRAN, Ada, and Pascal. Each language has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions.

1.3.1 Types of programming languages

High-level programming languages, while simple compared to human languages, are more complex than the languages the computer actually understands, called machine languages. Each different type of CPU has its own unique machine language.

Lying between machine languages and high-level languages are languages called assembly languages. Assembly languages are similar to machine languages, but they are much easier to program in because they allow a programmer to substitute names for numbers. Machine languages consist of numbers only.

Machine languages .

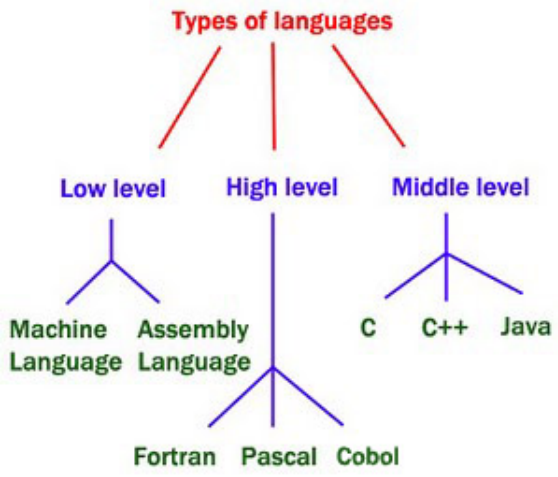
- Absolute language
- Language of 0 and 1
- Programmer has to remember list of code numbers to represent operation codes and storage locations of data and instructions.
- Time consuming
- Expensive
- Error prone.
- Format is op-code operand E.g. 0001 10011001

Assembly Languages .

- Also called symbolic languages
- Source code is converted to binary by using an assembler.
- Time saving, reduce error
- Detailed knowledge of the processor is required. Machine dependent.
- E.g: INX H . This increments the contents of registers H and E.

High level language .

- It uses a compiler or interpreter to convert source code to binary.
- It has rules and specification.
- Less time, easier to learn



1.3.2 Generation of programming languages

First generation programming language is pure machine code, that is just ones and zeros. Programmers have to design their code by hand then transfer it to a computer by using a punch card, punch tape or flicking switches. There is no need to translate the code and it will run straight away. Benefits are

- Code can be fast and efficient.
- Code can make use of specific processor features such as special registers.

Drawbacks are

- Code cannot be ported to other systems and has to be rewritten.
- Code is difficult to edit and update.

Second-generation programming languages By using codes resembling English, programming becomes much easier. The usage of these mnemonic codes such as LDA for load and STA for store means the code is easier to read and write.
To convert an assembly code program into object code to run on a computer requires an **Assembler** and each line of assembly can be replaced by the equivalent line of object (machine) code. Assembly code is often used when writing low level fast code for specific hardware. Benefits are

- Code can be fast and efficient.
- Code can make use of specific processor features such as special registers.
- As it is closer to plain English, it is easier to read and write when compared to machine code.

Drawbacks are

- Code cannot be ported to other systems and has to be rewritten.

Third-generation programming languages (High Level Languages) brought programmer-friendly features to code such as loops, conditionals, classes etc. This means that one line of third generation code can produce many lines of object (machine) code, saving a lot of time when writing programs. Third generation codes are **imperative**. Imperative means that code is executed line by line, in sequence.
Third generation languages can be platform independent, meaning that code written for one system will work on another. To convert a thirrd generation program into object code requires a **Compiler** or an **Interpreter**. Benefits are

- Hardware independence, can be easily ported to other systems and processors.
- Time saving programmer friendly, one line of third gen is the equivalent of many lines of first and second gen.

Drawbacks are

- Code produced might not make the best use of processor specific features unlike first and second gen.

Fourth-generation languages (very high level languages) are designed to reduce programming effort and the time it takes to develop software, resulting in a reduction in the cost of software development. They are non-procedural languages, because they allow programmers and users to specify what the computer is supposed to do without having to specify how the computer is supposed to do it. They need approximately one tenth the number of statements that a high level languages needs to achieve the same results.
4GL include languages to query databases (SQL), languages to make reports and languages to construct user interface (XUL). An example of 4th generation programming type is the declarative language. Declarative languages describe what computation should be performed and not how to perform it.

Fifth generation languages Natural Languages represent the next step in the development of programming languages, i-e fifth generation languages. The text of a natural language statement very closely resembles human speech. These languages are also designed to make the computer "smarter".
Natural languages already available for microcomputers include Clout, Savvy Retriever (for use with databases) and HAL (Human Access Language). The use of natural language touches on expert systems, computerized collection of the knowledge of many human experts in a given field, and artificial intelligence, independently smart computer systems.

1.3.3 Compiler, Linker, Interpreter, Loader

Assembler An assembler is a program that takes basic computer instructions and converts them into a pattern of bits that the computer's processor can use to perform its basic operations.

Compiler A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses. Typically, a programmer writes language statements in a language such as Pascal or C. The file that is created contains what are called the source statements. The programmer then runs the appropriate language compiler, specifying the name of the file that contains the source statements.
Translates source code (C, C++ etc.) and generates Object code (machine readable but not directly executable).

Interpreter Same as compiler but do that interactively (simply saying line by line). Interpreted languages don't usually require a linker, and employ a loading process different from that of a compiled program. The interpreter itself is linked and loaded like a compiled program, but the program interpreted by the interpreter is typically loaded through interpreter-specific means.

Linker Linker performs the linking of libraries with the object code to make the object code into an executable machine code.
Connects the compiler generated object code with library code to generated independent executable (like in C, you dont write how printf() works, linker add the code for printf() function in your program).

Loader In a computer operating system, a loader is a component that locates a given program (which can be an application or part of the operating system) in offline storage (such as a hard disk), loads it into main storage (in a personal computer, it's RAM), and gives that program control of the compute.
Load the machine readable codes in memory to be executed. This step typically includes relocating the code output by the linker to a specific memory location into which the program was being loaded.

Chapter 2

Module II

2.1 Basics of C

2.1.1 Character set

1. *Letters* - A to Z and a to z
2. *Digits* - 0 to 9
3. *Special Symbols* - Space ! @ # $ % ( ) _ { } +− = []| : ; "'< > , . ? /
4. *White spaces* - Blank space, Horizontal tab, Carriage return, New line, Form feed
5. 256 ASCII characters

2.1.2 Identifiers

- Refer to a name of variable, function, array.
- Case sensitive.
- Rules for identifiers are
    1. Arbitrary sequence of letters and digits, underscore.
    2. First Character must be a letter.
    3. Underscore is valid and is counted as a letter.
    4. Only first 31 characters are significant.
    5. Cannot use a keyword.
    6. Must not contain white space.
    7. Both upper case and lower case are allowed.
- Eg: radius1, radius2, _year, simple_interest

2.1.3 Keywords

- Convey special meaning to compiler.
- Basic building blocks of a language.
- All keywords are written in lower case.
- Eg: auto, break, case, char, const, do, double, else, enum, extern, float, for, if, int, long, register, return, short, static, struct, switch, union, void, while

```c
// hello world program
# include <stdio.h>
main()
{
   printf("Hello World");
}
```

2.1.4 Basic data types

- Identify type of data and associated operators for handling the data.
- Character data are enclosed in single quotes.
- Numbers without fractions represent integer data.
- Fractions represent real data.
- String data are enclosed in double quotes.
- Three types of data types
    1. Fundamental data types
    2. Derived data types
    3. User defined data types

Fundamental data types:

1. char
    - The char type is used to represent individual characters.
    - char type will generally require only one byte of memory.
    - Each char type has an equivalent integer interpretation.
    - char is **one** byte.
    - Can be signed (with values ranging from -128 to +127) or unsigned (with typical values ranging from 0 to 255).
2. int
    - No fractions.
    - Fixed point.
    - Decimal point and Punctuation are not allowed.
    - short int is at least **two** bytes.
    - int is two bytes in 16 bits compiler and four bytes in 32 bits compiler.
    - long int is **four** bytes.
3. float
    - Contain fractions or floating pointing numbers.
    - Punctuation is not allowed.
    - Includes decimal point or exponent form of the number.
    - float is **four** bytes and has 7 digits of precision.
4. double

- Double precision floating point numbers.
- Occupies twice the memory of float.
- double is **eight** bytes and has 15 digits of precision.
- long double is **ten** bytes and has 19 digits of precision.

Four **qualifiers** for integer data types.

1. short
   - A short int may require less memory than an ordinary int or it may require the same amount of memory as an ordinary int , but it will never exceed an ordinary int in word length.
   - If short int and int both have the same memory requirements (e.g., 2 bytes), then long int will generally have double the memory requirements (e.g., 4 bytes).
2. long
   - A long int may require the same amount of memory as an ordinary int or it may require the more memory than an ordinary int , but it will never be less than an ordinary int in word length.
   - If int and long int both have the same memory requirements (e.g., 4 bytes), then short int will generally have half the memory requirements (e.g., 2 bytes).
3. signed
   - The leftmost bit is reserved for the sign.
   - It can vary from -32,768 to +32,767 (which is typical for a 2-byte int)
   - Positive and Negative values are permitted.
4. unsigned
   - An unsigned int has the same memory requirements as an ordinary int.
   - With anunsigned int, all of the bits are used to represent the numerical value.
   - An unsigned int can be approximately twice as large as an ordinary int.
   - Negative values are not permitted.
   - unsigned int will be allowed to vary from 0 to 65,535.
   - The unsigned qualifier can also be applied to other qualified unsigned ints , e.g., unsigned short int or unsigned long int.

2.1.5 Variables

- It is an identifier that change values in the program.
- When variable is defined, memory is reserved.
- Each variable is associated with a symbolic name.
- Variable can be assigned any value. Information represented by variable can be changed during execution of the program.
- Data type of variable cannot be changed.
- A **declaration** associated a group of variables with a specific data type. A declaration consists of a data type, followed by one or more variable names, ending with a semicolon.
- All variables must be declared before they can be used in executable statements.
- **Declaration**: <dataType> <variableName>

- Eg: int num1, num2;
  char flag;
  float root1, root2;
  short int num3;
  long int num4;
  unsigned num5;
- Initialisation during definition <dataType> <variableName> = <value>
- Eg: char star = '*';
  int num = 12;
  float radius = 0.0;
- Initialisation during execution <dataType> <variableName> = <expression>
- Eg: int sum = num1 + num2;
  float perimeter = 2 * 3.14 * radius;

2.1.6 Constants

Items that **never change** their value during a program execution. Keyword const is used. THe following rules apply to all integer and floating point constants.

1. Commas and blank spaces are not allowed.
2. Constant can be preceded by the minus sign (-) if required.
3. The value of the constant cannot exceed specified the minimum and maximum bounds.

1. Integer Constant
   (a) Should contain atleast one digit.
   (b) No decimal point and commas.
   (c) Can contain + or - symbol.
   (d) Three formats
      i. *Decimal* - Any integer, not beginning with 0.
      Eg: 5890
      ii. *Octal* - Starts with **0**. Digits are 0 to 7.
      Eg: 0527
      iii. *Hexadecimal* - Starts with **0x or 0X**. Digits are 0 to 9, A to F.
      Eg: 0x7fff
   (e) An unsigned integer constant can be identified by appending the letter U or u at the end of the constant.
   Eg: 500U
   (f) A long integer constant can be created by appending the letter L or l at the end of the constant.
   Eg: 56789L
   (g) An unsigned long integer constant can be identified by appending the letter UL or ul at the end of the constant. U must precede L always.
   Eg: 3456 UL
2. Floating or Real Constant
   (a) Base 10 number that contains either decimal point or an exponent or both.
   (b) Can contain + or - symbol.
   (c) Single precision floating point - append F or f at the end

(d) Double precision real number - append L or l at the end

(e) Two Forms

i. *Fractional Form* - One digit before decimal point and atleast one digit after decimal point. Eg: 365.8905

ii. *Exponent Form* - Two parts Mantissa and Exponent. Mantissa is an integer or a real constant. Mantissa is followed by E or e and the exponent. Exponent must be an integer. Eg: 3e7, 3.67e5

(f) Single precision floating point constants have F appended to it.

(g) Long floating point constants have L appended to it.

3. Character Constant

(a) **Single character** enclosed in **single quotes**.

(b) ASCII character set (7 bits) is used. 128 different characters are possible.

(c) *Escape sequences* for non-graphic characters. Begin by \ followed by a special character.

(d) $\backslash n \ldots \backslash b \ldots \backslash t \ldots \backslash v \ldots \backslash" \ldots \backslash' \ldots \backslash? \ldots \backslash\backslash \ldots \backslash 0.$

(e) $\backslash 0$ represents the null character - ASCII 000. It is used to indicate the end of a string. Eg: 'A', 'x', '\ n'

4. String Constant

(a) **Multiple characters** enclosed in **double quotes**.

(b) Terminating character or End of Line - **'\0'** is automatically appended.

(c) The length of the literal is always incremented by 1 because of end of line (eol). Eg: "A", "Enter values", "To continue press \" Enter \" key."

2.1.7 Statements

1. Statements

- Instruction given to computer to be executed.
- Smallest executable unit.
- Statements manipulate, control or modify data.
- Always terminated by a **;**.
- Null statement has only **;**.
- Three different classes of statements are

(a) Expression statement - An expression statement consists of an expression, followed by a semicolon.

(b) Compound statement - A compound statement consists of a sequence of two or more consecutive statements enclosed in braces ({ and }).

(c) Control statement - These involve tests to determine if certain conditions are true or false, the repeated execution of groups of statements, and execution of individual groups of statements on a selective basis.

2. Block Statements

- Sequence of statements enclosed in {} .
- Block is treated as a single unit.
- Normal flow of program is statements are executed sequentially.

2.1.8 Operators

Specific tasks are called **operations**. Operations are represented by **operators**. Objects of operations are called **operands.** Operators can be classified based on the number of operands or based on the purpose of the operator.
Classification based on the purpose of the operator:

1. Arithmetic operators
    - Addition +
    - Subtraction -
    - Multiplication *
    - Division / - requires that the second operand be non-zero. For integer division, the quotient is truncated toward zero.
    - Remainder % - works only with int. - requires that the second operand be non-zero.
    - Unary + and Unary - is also supported.
2. Increment / Decrement operator
    - Increment $++$ adds 1 to its operand.
    - Decrement $--$ subtracts 1 from its operand.
    - Prefix $++x$ or $--x$ C++ performs operation before using the value of operands. Change then Use rule
    - Postfix version $x++$ or $x--$ C++ uses the value of operands first and then perform the operation. Use then change rule.
3. Relational Operators
    - Compare the operands (numbers or characters)
    - True results in 1.
    - Four **relational** operators $<, >, <=, >=$
    - Two **equality** operators $==, !=$
4. Logical Operators
    - Logical AND $\&\&$
    - Logical OR $||$
    - Logical NOT !. It is an Unary operator.
5. Bitwise Operators
    - Bitwise AND $\&$
    - Bitwis OR $|$
    - Bitwise NOT !. It is an Unary operator.
    - Bitwise XOR ~
    - Left shift $<<$
    - Right shift $>>$
6. Conditional Operator
    - Ternary operator
    - result = expression 1 ? expression 2 : expression 3

- If expression 1 is true, then result = value of expression 2, otherwise result = value of expression 3.

7. Assignment operator

- Assignment operators are used to form assignment expressions. It is used to assign a value to an identifier.
- The associativity is from right to left.
- $=, +=, -=, *=, /=, \%=$
- Assignment expression is written in the form *identifier* = *expression*.
  Eg. $area = length * breadth$
- Assignment expression is written in the form
  expression1 += *expression2*.
  expression1 = *expression1* + *expression2*
  Eg. $count+ = 2$

8. sizeof operator

- Unary Compile time operator
- sizeof variable - returns length in bytes of the variable.
- sizeof (dataType) - returns length in bytes of the data type.

9. Comma operator

- String together several expressions.
- Evaluated from left to right.
- b = (a=3, a+1); First 3 is assigned to a. Then $a + 1$ is evaluated and result stored in b.
- Lower precedence than Assignment operator.

Classification based on number of operands:

1. Unary operators

- Only one operand
- Unary + and Unary - represent positive and negative numbers. Eg: +10, -15
- Increment ++
- Decrement −−
- Logical NOT !
- Bitwise NOT !
- sizeof
- (type) used for type casting

2. Binary operators

- Two operands are essential.
- **+ - * / %** $==, <, >, <=, >=, != \&\&|| \&| \tilde{}<<>>$

3. Ternary operators

- Has three operands.
- result = expression 1 ? expression 2 : expression 3

Operator Precedence and Associativity

- Precedence tells you the order in which operators are executed when they are found in the same expression.
- Associativity tells you how the operators are evaluated; either from right to left or from left to right.

Category	Operators	Associativity
Unary	$-, ++, --, !$, sizeof, (type)	R $\rightarrow$ L
Arthmetic	$*, /, \%$	L $\rightarrow$ R
Arthmetic	$+, -$	L $\rightarrow$ R
Relational	$<, >, <=, >=$	L $\rightarrow$ R
Equality	$==, !=$	L $\rightarrow$ R
Logical AND	$\&\&$	L $\rightarrow$ R
Logical OR	$\|\|$	L $\rightarrow$ R
Conditional	$?:$	R $\rightarrow$ L
Assignment	$=, +=, -=, *=, /=, \%=$	R $\rightarrow$ L

```c
// adding two numbers, using i/o
# include <stdio.h>
main()
{
   int num1, num2, sum;
   printf("Enter two numbers ");
   scanf("%d %d", &num1, &num2);
   sum = num1 + num2 ;
   printf("Sum of %d and %d = %d ", num1, num2, sum);
}
```

2.1.9 Expressions

An expression represents

- a single data item such as number or character.
- a single entity such as constant, variable, array or reference to function.
- combinations of entities interconnected by one or more operators.
- logical conditions that are either true or false.
  Eg: num1 + num2
  root1 <= root2

1. Arithmetic Expressions
   - Integer or real or mixed expressions.
   - Pure integer expressions gives an integer result.
   - Can contain unsigned and signed variables, constants joined by arithmetic operators.
   - When variable of different data types are mixed they are converted to same data type. This is called **Type conversion**.
   - **Implicit type conversion or type promotion** is done by compiler, converting all operands to the type of largest operand.
     - char or int and float result in float
     - char or int converted to int
     - int and long int result is long int
     - float and double result is double

  - float and long double result in long double
  - double and long double result in long double
- **Explicit type conversion of type casting** is programmer defined and forces an expression to be of specific type. (type) expression Eg: (int) num / 2.5;

2. Relational or Logical Expressions
   - Can contain unsigned and signed variables, constants joined by relation or logical operators.
   - Expression results in 1 or 0.
3. Compound Expressions
   - Assignment statement.
   - variable operator= expression

```c
// solution of quadratic equation
# include <stdio.h>
main()
{
   float a,b,c, roo1, root2, deter;
   printf("Enter three numbers as a, b, c ");
   scanf("%f %f %f", &a, &b, &c);
   deter = sqrt(b*b - 4*a*c);
   root1 = (-b + deter) / 2 * a;
   root2 = (-b - deter) / 2 * a;
   printf("Determinent = %f \n Roots are %f and %f ", deter, root1, root2);
}
```

2.1.10 Input and Output statements

1. getchar()
   - read single character
   - It returns a single character from a standard input device.
   - If **end of file** condition is encountered when reading a character with the getchar(), the value of the symbolic constant **EOF (-1)** will be returned.
   - Syntax char variable = getchar();
   - Eg: char choice = getchar();
2. putchar()
   - write a single character
   - It transmits a single character to a standard output device.
   - Syntax putchar(char variable);
   - Eg: putchar(ch);
3. gets()
   - read string
   - Eg: char sentence[50]; gets (sentence);
4. puts()
   - write string

- Eg: puts(sentence);

5. scanf()

- formatted input
- enters data from standard input device and stores it in the computer's memory.
- Syntax scanf(control_string, arg1, arg2, ..., argn);
  Control string refers to a string containing certain required formatting information, and arg1, arg2, ..., argn are arguments that represent the individual input data items.
- The control string consists of individual groups of characters, with one character group for each input data item. Each character group must begin with a percent sign (**%**). In its simplest form, a single character group will consist of the percent sign, followed by a **conversion character** which indicates the type of the corresponding data item.
- The arguments are written as variables or arrays, whose types match the corresponding character groups in the control string. Each variable name must be preceded by an *ampersand* (&).
- Eg: scanf("%d %f %15s", &integer, &real, &string);
  scanf("%[ABCDEFGHIJKLMNOPQRSTUVWXYZ]", line); - enter a string consisting of uppercase letters and whitespace.

Conversion Character	Meaning
c	data item is a single character
d	data item is a decimal integer
f	data item is a floating-point value
h	data item is a short integer
s	data item is a string followed by a whitespace character
u	data item is an unsigned decimal integer

6. printf()

- formatted output
- used to output any combination of string, single character, numerical values to standard output device.
- moves data from computer's memory to standard output device.
- Syntax printf(control_string, arg1, arg2, ..., argn);
  Control string refers to a string containing certain required formatting information, and arg1, arg2, ..., argn are arguments that represent the individual output data items.
- The arguments in a printf function do **not** represent memory addresses and therefore are not preceded by **ampersands**.
- The control string consists of individual groups of characters, with one character group for each input data item. Each character group must begin with a percent sign (**%**). In its simplest form, a single character group will consist of the percent sign, followed by a **conversion character** which indicates the type of the corresponding data item.
- Eg: printf("integer = %d real = %f string = %15s", integer, real, string);

Conversion Character	Meaning
c	data item is a single character
d	data item is a decimal integer
f	data item is a floating-point value without exponent
e	data item is a floating-point value with exponent
h	data item is a short integer
s	data item is a string followed by a whitespace character
u	data item is an unsigned decimal integer

7. Use the library stdio.h

```
// conversion of case of a single character
# include <stdio.h>
# include <ctype.h>

main()
{
  int ch;
  printf("Enter any character value");
  ch = getchar();
  printf("The equivalent uppercase letter is");
  putchar(toupper(ch));
}
```

2.2 Control Statements

- C program may require that a logical test be carried out and one of several possible actions will then be carried out, depending on the outcome of the logical test. This is known as **branching**.
- There is also a special kind of branching, called **selection**, in which one group of statements is selected from several available groups.
- A program may require that a group of instructions be executed repeatedly, until some logical condition has been satisfied. This is known as **looping**.
- Use the four relational operators ¡, <=, >, >= and the two equality operators, ==, ! =, two logical operators && (AND) and || (OR), and the unary negation operator !

2.2.1 Selection

1. if statement
   - Used to carry out a logical test and then take one of two possible actions, depending on the outcome of the test.
   - Syntax if (expression) statement
   - The else portion of the if - else statement is optional.
   - The expression must be placed in parentheses.
   - The statement will be executed only if the expression has a nonzero value (i.e., if expression is true). If the expression has a value of zero (i.e., if expression is false), then the statement will be ignored.
   - The statement can be either simple or compound.

```
// Biggest of two numbers
# include <stdio.h>

main()
{
  int first, second;
  scanf("%d %d", &first, &second);
  if (first > second)
    printf("First number is bigger. \n");
  if (first < second)
```

```
    printf("Second number is bigger. \n");
  if (first == second)
    printf("Both numbers are equal. \n");
}
```

```
if (pastdue >0)
    credit = 0;

 // another example
if (x<=3.0) {
  y= 3 *pow(x,2);
  printf(("%f ",y);
}

// another example
if (( balance < 1000) || (status =='R' ))
  printf("%f", balance);
```

2. if - else statement

- If a condition is true an action is taken and another action if the condition is false.
- Syntax if (expression) statement 1 else statement 2
- If the expression has a nonzero value (i.e., if expression is true), then statement 1 will be executed. Otherwise (i.e., if expression is false), statement 2 will be executed.

```
// Check if a given number is even or odd
# include <stdio.h>

main()
{
  int number;
  scanf("%d ", &number);
  if (number % 2 == 0)
    printf("The given number is even. \n");
  else
    printf("The given number is odd. \n");
}
```

```
if (pastdue >0)
  credit = 0;
else
  credit = 1000;

  // another example
if (x<=3.0) {
  y= 3 *pow(x,2);
  printf(("3 times x power 2 = %f ",y);
}
else {
  y= 2 *pow(x-3,2);
  printf(("2 times (x-2) power 2 = %f ",y);
}
```

```
// another example
if (status =='R')
  tax = 0.20 * pay;
else
  tax = 0.14 * pay;

tax = (status =='R') ? (0.20 * pay) : (0.14 * pay);
```

2.2.2 Iteration

Program construct to repeat a block of statements.

2.2.2.1 while

- Condition is checked first.
- Then the statements are executed.
- Syntax while (expression) statement
- The statements will be executed repeatedly, as long as the expression is true.
- Best suited for situations where the iteration continues indefinitely until the specified logical conditional has been satisfied.
- Entry controlled loop
- Used even if the number of iterations are not known.

Example: Lowercase to uppercase conversion

```
// Lowercase to uppercase conversion
# include <stdio.h>
# include <ctype.h>

main()
{
        char letter[80];
        int length, count = 0;
         // read lowercase text
         while (( letter[count] = getchar()) != '\n' )
                ++count;
        length = count;

        //display uppercase text
        while (count < length)
        {
                putchar(toupper(letter[count]));
                count++;
        }
}
```

Example: Average of list of numbers

```
// Average of list of numbers
# include <stdio.h>

main()
```

```
{
            int len, count = 1;
            float num, avg, sum =0;

            // read number of numbers
            printf(''How many numbers?: '');
            scanf(''%d'', &len);

             // read numbers
             while (count <= len  )
             {
                    printf (''Number = '');
                    scanf(''%f'', &num);
                    sum += num;
                    count++;
             }

            // calculate average and display it
            avg = sum / len;
            printf(''\n Average = %f'', avg);
}
```

2.2.2.2 do...while

- Condition is checked at the each iteration.
- Atleast once the statements are executed before the condition is checked.
- Syntax do statement while (expression);
- The statements will be executed repeatedly, as long as the expression is true.
- Best suited for situations where the statements are to be executed atleast once before the expression is evaluated.
- Menu driven programs normally use do...while.

Example: Lowercase to uppercase conversion

```
// Lowercase to uppercase conversion
# include <stdio.h>
# include <ctype.h>

main()
{
            char letter[80];
            int length, count = 0;
             // read lowercase text
             do
                    ++count;
      while (( letter[count] = getchar()) != '\n' );
            length = count;

            //display uppercase text
            do
            {
```

```
                putchar(toupper(letter[count]));
                count++;
            } while (count < length);
}
```

Example: Average of list of numbers

```
// Average of list of numbers
# include <stdio.h>

main()
{
            int len, count = 1;
            float num, avg, sum =0;

            // read number of numbers
            printf(''How many numbers?: '');
            scanf(''%d'', &len);

             // read numbers
             do
             {
                    printf (''Number = '');
                    scanf(''%f'', &num);
                    sum += num;
                    count++;
             } while (count <= len  );

            // calculate average and display it
            avg = sum / len;
            printf(''\n Average = %f'', avg);
}
```

2.2.2.3 for

- Most commonly used loop.
- This includes an expression 1 that specifies an inital value for an index.
- Second expression that determines whether or not the loop is continued.
- Third expression allows the index to be modified at the end of each pass.
- Syntax for (expression1; expression2; expression3) statement
- Expression 1 is an assignment statement to initalize and loop variable, expression 2 is a logical expression and expression 3 is a unay expression or an assignment expression that changes the value of loop variable.
- The expression 2 is evaluated and tested at the beginning of each iteration.
- Expression 3 is evaluated at the end of each iteration.
- Best suited for situations where the number of iterations is known in advance.

Example: Consecutive odd integers.

```c
// Display odd numbers through 0 and 10.
# include <stdio.h>

main()
{
      int digit;
            for ( digit = 1; digit <= 10; digit +=2)
                  printf(''%d \t '', digit);
}

// Average of list of numbers
# include <stdio.h>

main()
{
            int len, count = 1;
            float num, avg, sum =0;

            // read number of numbers
            printf(''How many numbers?: '');
            scanf(''%d'', &len);

             // read numbers
             for (count=1; count<= len; count++)
             {
                    printf (''Number = '');
                    scanf(''%f'', &num);
                    sum += num;
                    count++;
             }

            // calculate average and display it
            avg = sum / len;
            printf(''\n Average = %f'', avg);
}
```

2.2.3 Branching

2.2.3.1 switch

- The switch statement causes a particular group of statements to be chosen from several available groups.
- The selection is based upon the current value of an expression which is included within the switch statement.
- Syntax switch (expression) statement
- Expression results in an integer value.
- Statement is a compound statement that specifies alternate courses of action.
- For each alternative, the first statement within the group must be preceded by one or more case labels (also called case prefixes).

- Each group of statements is written as
  case expression : statement 1
  statement 2
  .....
  statement n

Example

```
switch (choice = toupper(getchar())) {
        case 'R':
                printf ("RED") ;
                break;
        case 'W':
                printf("WHITE");
                break;
        case 'B':
                printf ("BLUE") ;
                break;
        default:
                printf("error");
        }
```

2.2.3.2 break

- The break statement is used to terminate loops or to exit from a switch.
- It can be used within a for, while, do -while, or switch statement.
- Syntax break;
- In nested loop, using break statement in the inner loop exits to outer loop.

2.2.3.3 continue

- The continue statement is used to bypass the remainder of the current pass through a loop.
- The loop does not terminate when a continue statement is encountered. The remaining loop statements are skipped and the computation proceeds directly to the next pass through the loop.
- The continue statement can be included within a while, a do -while or a for statement.
- Syntax continue;

2.2.3.4 goto

- The goto statement is used to alter the normal sequence of program execution by transferring control to some other part of the program.
- Syntax goto label;
- Label is an identifier that is used to label the target statement to which control will be transferred.
- The target statement will appear as Syntax label: statement
- Label must have a unique ; i.e., no two statements can have the same label.

2.2.4 Nesting of control statements

2.2.4.1 Nesting of if-else

It is also called if-else ladder.

```
if (e1)
  if ( e2)
    s1
  else
    s2
else
  if (e3)
    s3
  else
    s4
```

Example

```
if  (( time >= 0) && (time < 12)) printf("Good Morning");
  else if  (( time >= 12) && (time < 18)) printf("Good Afternoon");
    else if  (( time >= 18) && (time < 24)) printf("Good Evening");
      else printf("Time is out of range");
```

Find biggest of three numbers

```
if (num1 > num2)
  if (num1 > num3)
          printf(''Num1 = %d is biggest'', num1);
  else
          printf(''Num3 = %d is biggest'', num3);
  else
  if (num2 > num3)
          printf(''Num2 = %d is biggest'', num2);
  else
          printf(''Num3 = %d is biggest'', num3);
```

2.2.4.2 Nesting of loops

Any loop can be nested inside another loop.

Example: Simple interest problem

```
# include <stdio.h>
# include <math.h>

main()
{
  float princi, rate, years, interest;
  printf("Enter value for principal \n");
  printf("To exit from the program type 0 for principal \n");
  scanf(''%f'', &princi);
  if (princi < 0) {
     printf("Error: please try again \n");
     scanf(''%f'', &princi);
  }
  while (p > 0) {
          printf("Enter value for interest rate \n");
          scanf(''%f'', &rate);
```

```
            if (rate < 0) {
                printf("Error: please try again \n");
        scanf(''%f'', &rate);
            }
            printf("Enter value for number of years \n");
            scanf(''%f'', &years);
            if (years < 0) {
                printf("Error: please try again \n");
        scanf(''%f'', &years);
            }

            rate /= 100;
            interest = princi * pow((1 + rate), years);

            // display output
            printf("Interest = %0.2F \n", interest);

            // repeat input
            printf("Enter value for principal \n");
            printf("To exit from the program type 0 for principal \n");
     scanf(''%f'', &princi);
     if (princi < 0) {
            printf("Error: please try again \n");
      scanf(''%f'', &princi);
   }
      }
}
```

2.2.5 Simple programs