PROBLEM SOLVING WITH PROGRAMMING

EDITORIAL BOARD

Shoolini University Centre for Distance and Online Education

Shoolini University

Solan (HP)

Printed by Shoolini University Centre of Online and Distance Education (SCODE)

First printing, 2022.

Shoolini University

Bajhol, Solan, Himachal Pradesh

17322

Contents

UNIT 1: COMPUTER FUNDAMENTALS: INTRODUCTION

CONTENTS

OBJECTIVES:

- Definition of Computer
- History of Computer
- Generations of Computers

1.1. INTRODUCTION

The term "compute," which means "to calculate," is where the word "computer" originates from. As a consequence of this, the majority of people today utilise a computer as a calculator since it can perform mathematical operations very quickly. In point of fact, the development of a computer was motivated initially by the desire to create a faster calculator. On the other hand, more than 80 percent of the work that is done on computers nowadays does not involve mathematics or numerical calculations. Therefore, if we only think of a computer in terms of a calculator, we are missing out on around 80 percent of what it is capable of.

A computer can be defined as a device that manipulates data, which is a more precise description. The term "data" can refer to anything from "bio-data" to "marks attained by students in when compiling results" to "information (name, age, sex, etc.) of passengers when making railway bookings" to "a number of varied aspects when solving scientific challenges," and so on and so forth.

As a consequence of this, data can take on a variety of shapes and sizes, depending on the type of computer application that is being used. At every point in time, a computer has the capability of retrieving and storing data. Because computers are capable of processing data, the phrase "data processor" has emerged as a result of this capability.

Data processor is a more inclusive term because current computers not only compute in the traditional sense but also execute extra operations based on data that goes to and from the computers themselves. Data processors, for instance, may be responsible for collecting data from a wide variety of incoming sources, merging (combining) it all, sorting it in the right order, and ultimately outputting it in the format that was specified by the user. However, despite the fact that none of these tasks involve traditional mathematical computations, computers are the most effective tools for completing them.

The act of processing data through the use of a computer is referred to as "data processing." The three sub-activities that make up data processing are the capturing of input data, the modification of data, and the management of output outcomes. In the context of data processing, information is defined as data that has been organised in a manner that renders it useful to the individuals who receive it. As a result of this, data is the raw material for data processing, while information is the processed data that is received as a result of data processing.

1.2. COMPUTER ATTRIBUTES AND CHARACTERISTICS

Because of the ever-increasing prevalence of computers in our daily lives, it's become clear that they're an incredibly useful tool. This popular tool's potency and use are mostly attributable to the following characteristics:

1. **Automatic:** An automated machine is one that can function without the assistance of a human operator. Once a computer has started working on a task, it will continue to complete it (often without the assistance of a person) until the task is finished. This is why computers are referred to be automated machines. Computers, on the other hand, are incapable of initiating themselves and are unable to investigate their own problems and find solutions. It is necessary for us to offer a computer with instructions in the form of coded instructions that detail precisely how a task should be completed by the machine. The fact that computers are fully automated and can handle issues without the assistance of a human being is responsible for a number of the benefits associated with their use, including their speed and precision.

2. **Speed**: A computer is an extremely quick piece of machinery. It is able to finish the same amount of work as a human being could finish in a year if that person worked nonstop for 365 days of the year, 24 hours a day, and did nothing else but work. To put it another way, a computer may complete in a matter of minutes what would take a man the entirety of his life to accomplish. Microseconds (106), nanoseconds (10–9), and even picoseconds (10–12) are the units of measurement that are utilised when speaking about the speed of a computer rather than milliseconds or seconds (10–3). A robust computer is able to do tens of billions of operations in the realm of elementary mathematics in a single second.

3. **Accuracy**: Computers are extremely precise in addition to being extremely quick. A computer's accuracy is consistently high, and the degree of accuracy is determined by its design. A computer can execute any computation with the same level of precision. In a computer, though, faults can arise. Human flaws, not technology flaws, are at blame for the majority of these mishaps. Errors might arise as a result of a programmer's sloppy thinking (a person who creates instructions for a machine to tackle a specific problem) or inaccurate input data. Garbage-in-garbage-out is a term used to describe computer failures caused by improper input data or faulty programmes (GIGO).

4. **Diligence**: A computer, unlike humans, is immune to boredom, fatigue, and lack of attention. It can labour for hours at a time without making any mistakes or whining. As a result, computers outperform humans in everyday tasks that demand high precision. If 10 million computations must be completed, a computer will complete the final one with the same precision and speed as the first.

5. **Versatility**: One of the most appealing features of a computer is its versatility. It may be preparing test results one minute, power bills the next, and in between, it may be assisting an office secretary in tracking down a crucial letter in seconds. To modify its skill, all that is necessary is to insert a new programme (a set of instructions for the computer) into it. In summary, if a task can be reduced to a limited set of logical stages, a computer can accomplish practically any work.

6. **The Importance of Remembering**: When a person learns anything new, his or her brain instinctively chooses what it considers to be essential and worth remembering. Unimportant facts are pushed to the back of the memory or forgotten entirely. With computers, this is not the case. Because of its secondary storage (a sort of detachable memory) capabilities, a computer can store and recall unlimited quantity of data. It can save a piece of information for as long as the user wants and then recall it when needed. A user can recall precisely the same information that he or she placed on the computer several years ago, even after several years. Only when a user requests it, does a computer forget or lose its creation information.

7. **There is no IQ**: A computer isn't some sort of mystical machine. It doesn't have any kind of intellect of its own. It has a 0 IQ, at least till now. It must be instructed on what to perform and in what order. As a result, only the user may decide what duties a computer will accomplish. In this case, a computer cannot make its own conclusion.

8. **There are no feelings:** Computers are emotionless machines. Because they are Machines, they have no sentiments or impulses. Despite the fact that men have succeeded in creating a computer memory, no computer can handle the equivalent of a human heart and soul. In our daily lives, we typically make decisions based on our feelings, taste, knowledge, and experience, but computers cannot make decisions on their own. They make decisions depending on the instructions we provide them in the form of computer programmes (human beings).

1.3. EVOLUTION OF COMPUTER

The mother of invention is need. The same may be said for computers. The desire for quick and precise calculating equipment led to the invention of computers.

In 1642, the first mechanical adding machine was invented by Basic Pascal, according to legend. As time passed, the first multiplication calculator was created by German Baron Gottfried Wilhelm von Leibniz in 1671. Around the year 1880, the United States was the birthplace of the keyboard machine, and we continue to make use of them now. Around the same time, Herman Hollerith developed punched cards, which were commonly used in computers until the late 1970s as an input medium. Around the turn of the century, people in Europe and America first started to get their hands on machines and calculators.

Charles Babbage, who served as a professor at Cambridge University in the nineteenth century, is generally acknowledged as the person who pioneered the development of modern digital computers. A team of clerks had been brought on board to assist him in the compilation of mathematical and statistical tables. Because even the most careful attention and measures could not eliminate the possibility of human error, Babbage was required to spend several hours going through these tables. Because his job involved a lot of repeated tasks, he rapidly became dissatisfied and unhappy with it. As a consequence of this, he started to think about the possibility of creating a machine that could calculate tables without making any mistakes. As a component of this process, Babbage developed in 1822 what he called a "Difference Engine" that had the ability to generate reliable tables. In the year 1842, Babbage put out the idea of a fully automated Analytical Engine that would be able to solve any mathematical problem by performing the most elementary arithmetic operations at a rate of sixty additions per minute. A working model of the machine could not be created at the time due to the lack of specific engineering required to build it. The work he accomplished, on the other hand, helped to build a set of concepts that are now necessary to the construction of any digital computer. To gain a better understanding of the evolution of computers, it is helpful to discuss the most famous early computers.

The following are some of them:

1. **The Computer Mark I (1937-44)**: Howard A. Aiken of Harvard University, in collaboration with IBM (International Business Machines Corporation), developed the world's first fully automatic calculating machine. The Automatic Sequence Controlled

calculator was another name for this device at one point. Punched card machines were the inspiration for this electro-mechanical system, which included a mechanical and an electrical component.

The design of this machine was incredibly sophisticated and it was massive in size, yet it proved to be extremely reliable. It was about feet long and 8 feet high, with over 3000 electrically operated switches controlling its activities. It could perform five fundamental arithmetic operations on integers up to 23 decimal digits, including addition, subtraction, multiplication, division, and table referencing. Adding two numbers took around 0.3 seconds, and multiplying two numbers took about 4.5 seconds. In comparison to today's computers, the machine was obviously extremely sluggish.

2. **The Atanasoff-Berry Computer (1939-42):** Dr. John Atanasoff created an electrical gadget that could solve mathematical equations. After its creator, Atanasoff, and his assistant, Clifford Berry, the machine was dubbed the Atanasoff- Berry Computer, or ABC. Internal logic was implemented using 45 vacuum tubes, with storage provided by capacitors.

3. **The ENIAC computer (1943-46) :** The first all-electronic computer was the Electronic Numerical Integrator and Calculator (ENIAC). It was built by a design team directed by Professors J. Presper Eckert and John Mauchly at the University of Pennsylvania's Moore School of Engineering.

   ENIAC was created in response to military requirements. For many years, it was employed to solve ballistic difficulties. It took up 20 x 40 square feet of wall space and employed 18,000 vacuum tubes to add and multiply two integers in 200 microseconds and 2000 microseconds, respectively.

4. **The EDVAC (1946-52):** One of ENIAC's biggest flaws was that its programmes were hardwired on boards, making it impossible to alter them. Later, Dr. John Von Neumann proposed the "stored programme" notion, which helped to solve the problem. The core notion behind this concept is that a computer's memory may be used to store a series of instructions and data for autonomously controlling the flow of processes. Because of the simplicity with which multiple programmes may be loaded and performed on the same computer, this characteristic has a significant impact on the development of

contemporary digital computers. We commonly refer to current digital computers as stored programme digital computers because of this capability. The saved' programme notion was employed in the architecture of the Electronic Discrete Variable Automatic Computer (EDVAC). Von Neumann is also credited for developing the concept of storing both instructions and data in binary form (a system in which all letters are represented by only two digits—0 and I), rather than decimal numbers or human readable phrases.

5. **The EDSAC is number five (1947-49):** The Electronic Delay Storage Automatic Calculator was created almost simultaneously with EDVAC in the United States (EDSAC). In May 1949, the machine ran its first programme. Addition operations took 1500 microseconds in this computer, whereas multiplication operations took 4000 microseconds. This machine was created by a team of scientists at the Cambridge University Mathematical Laboratory led by Professor Maurice Wilkes.

6. **The UNIVAC I is number six (1951):** The Universal Automatic Computer (UNIVAC) was the first non-one-of-a-kind digital computer. The first UNIVAC machine was placed in the Census Bureau in 1951 and was utilised continually for the next ten years. General Electric Corporation used a UNIVAC I computer for the first time in business in 1954. As early as 1952, International Business Machines (IBM) introduced the IBM-701 commercial computer to the world. Improved variants of the UNIVAC I and other 700-series machines were introduced in quick succession. In 1953, IBM released the IBM-650 computer, which went on to sell over 1000 units. The introduction of commercially viable digital computers for business and scientific purposes was highlighted by UNIVAC.

1.4. VARIOUS GENERATIONS OF COMPUTERS

The term "generation" refers to a step in the development of computer technology. It offers a foundation upon which the expansion of the computer industry can occur. The term "generation" was originally employed to differentiate between the various hardware technologies available at the time; but, in recent years, its scope has been broadened to encompass not only the hardware but also the software that collectively compose a computer system.

The practise of referring to the period of computing in terms of generations did not become common practise until after the year 1964. It has been determined that there have been a total of five distinct generations of computers to date. In this section, each generation and the distinguishing characteristics of that generation are discussed. The approximately correct dates that are stated against each generation are usually recognised, despite the fact that there is some overlap across generations.

During the course of the explanation of several computer generations, you will be exposed to a number of new concepts and jargons related to computers, some of which you may not entirely understand. On the other hand, the purpose of this article is to offer you a comprehensive review of the key scientific advances and technological developments that have taken place during the course of the five generations of computers. In the next chapters, we will examine these technological advancements and breakthroughs in further detail.

- **FIRST GENERATION (1942-1955)**

The ENIAC, EDSAC, EDVAC, UNIVAC 1, and IBM 701 are just a few of the early computers that have been investigated in the past. At the time, various pieces of technology, including this one, made use of thousands of vacuum tubes. A vacuum tube was a delicate glass instrument that was used to regulate and amplify electronic signals by employing filaments as an electrical source. The device was known as a vacuum tube. As far as high-speed electrical switching devices were concerned, this was it. The calculations that were performed on these so-called first-generation computers, which used vacuum tubes, could be finished in milliseconds.

Punch cards were used to enter all of the data and instructions into the system, and the memory of these computers was made up of electromagnetic relays. Instructions were written in machine language and assembly language since higher-level programming languages were not developed until much later in computer history. Due to their famously difficult nature, early computers were only able to be programmed by a limited few highly skilled individuals.

The following is a list of characteristics of computers from the first generation:

1. At the time, these calculators had the fastest processing speeds in the world.

2. The rooms in which they were installed had to be very large because of the unreasonably large size of the components.

3. They required hundreds of vacuum tubes, each of which was responsible for producing a significant amount of heat and frequently caught fire. As a consequence of this, the rooms and facilities that contained these computers required enough ventilation in order to function properly.

4. The amount of electricity drawn by each vacuum tube was around a half watt. Due to the fact that these computers often required more than 10,000 vacuum tubes, their overall power consumption was rather high. This was a direct result of this requirement.

5. The lifespan of vacuum tubes was restricted because they used filaments in their construction. Hardware problems were common because of the large number of vacuum tubes in these computers.

6. These computers required frequent upkeep due to their short mean time between failures.

7. These computers each contained hundreds of thousands of individual components that had to be painstakingly combined into electrical circuits. As a direct consequence of this, the process of commercialising these computers was difficult and costly.

8. Due to the complexity of both their development and their use, these computers were only put to use in a limited number of different commercial contexts.

- ## SECOND GENERATION (1955-1964)

William Shockley, John Bardeen, and Walter Brattain of Bell Laboratories came up with the innovative idea for the transistor in 1947. The transistor is an electrical switching device. Due to the following qualities, transistors soon established themselves as a superior electronic switching technology compared to vacuum tubes:

1. In comparison to tubes, they were more long-lasting and more manageable due to the fact that they were formed of germanium semiconductor material rather than glass.

2. They were more dependable than tubes since they did not have any components that may fail, such as a filament, and hence were not susceptible to failure.

3. They were able to swap gears substantially more quickly than tubes (almost 10 times faster). As a consequence of this, switching circuits based on transistors may function far more quickly than switching circuits based on tubes.

4. They utilised approximately a small portion of the energy that was utilised by tubes.

5. They were considerably smaller than a tube in its whole.

6. The production of them required fewer resources than expected.

7. In comparison to vacuum tubes, they were much better at removing heat from the circuit.

In the construction of the second-generation of computers, transistors were employed rather than vacuum tubes. Because of the characteristics of transistors, computers of this generation were superior to those of the first generation in a number of respects, including their capacity, dependability, cost, size, and temperature during operation. As a major type of storage, magnetic memory chips were employed, while magnetic discs and tapes were used for backup. There were still many uses for punched cards, including the building of computer programmes and the transfer of data to them.

During the second generation, high-level programming languages and batch operating systems were developed. Some examples of these languages include FORTRAN, COBOL, ALGOL, and SNOBOL. As a result of the fact that people were better able to comprehend and work with high-level programming languages than they were with machine or assembly languages, second-generation computers were significantly simpler to build and operate than first-generation computers. When the batch operating system was first implemented, many jobs could be bundled together into a single "batch" and sent in all at once. A batch processing operating system will automatically move on to the next task after the current one has been completed. Reduced human participation in the execution of numerous workloads made second-generation computers faster, more efficient, and easier to use thanks to this concept.

Calculating scientific data was the principal function that the first generation of computers were designed to do. Commercial data processing applications including payroll, inventory control, marketing and production planning became increasingly common on second-generation computers. These applications were performed on computers that were of the second generation.

The following is a list of characteristics shared by computers of the second generation:

1. They were more than ten times faster than the computers that had been used in the generation before them.

2. Compared to the computers of the first generation, they were more compact and required less space.

3. In comparison to computers of the first generation, they consumed less energy and created less heat. It was still necessary to ensure that there was appropriate ventilation in the rooms and locations where the computers of the second generation were stored.

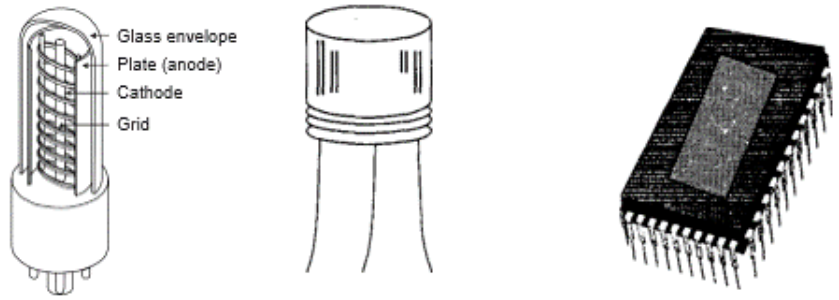
4. In comparison to their predecessors, they were more dependable and had a lower incidence of hardware faults.

5. These newer machines have quicker processors and larger primary and secondary storage capacities than the computers of the first generation.

6. In comparison to the computers of the first generation, they were simpler to use and less difficult to programme on. As a direct consequence of this, their application in business settings increased.

7. The commercial fabrication of early computers was difficult and expensive since thousands of individual transistors had to be incorporated into electrical circuits one at a time.

Figure 1.1: Electronic Devices Used for Manufacturing Computers of Different



(a) A Vacuum Tube (b) A Transistor (c) An IC Chip

- **THIRD GENERATION (1964-1975)**

The integrated circuit was invented by Jack Clair Kilby and Robert Noyce in 1958. Integrated circuits, often known as ICs, are electronic circuits that are created on a single silicon chip and

do not require any physical connectivity between the many electronic components that make up the circuit. These components include transistors, resistors, and capacitors. Because it enabled a greater number of circuit components to be integrated into a relatively small (less than 5 mm square) silicon surface known as a "chip," the IC technology was also known as the 0 technology. In the beginning, integrated circuits consisted of only ten to twenty individual components. This technology is referred to as "small scale integration," and that is its name (SSI). In later years, as the technology used in the production of integrated circuits (IC) improved, it became possible to assemble as many as components onto a single chip. This technique was referred described as "middle scale" throughout the discussion (MSI).

Computers that made use of integrated circuits were the defining characteristic of the third generation. In the beginning, a technology known as SSI was utilised, and later on, a technique known as MSI was utilised. Integrated circuits were more compact, less expensive to make, more long-lasting and reliable, quicker in operation, dissipated less heat, and consumed less power than circuits that were formed by manually connecting electrical components. As a direct consequence of this, computers of the third generation were considerably more powerful, trustworthy, affordable, compact, and easy on the user's thermal comfort than their predecessors.

Parallel developments in storage technology have made it possible to create magnetic discs and tapes with bigger capacities, as well as larger capacities for magnetic core-based random access memory. As a direct consequence of this, most computers of the third generation had only a few megabytes of primary memory (less than 5 megabytes), and their magnetic disc drives could only hold a few tens of megabytes of data at a time. In addition, the memory of these computers was limited.

The third generation of computer science saw the standardisation of high-level programming languages, timesharing operating systems, software decoupling from hardware, and the establishment of an autonomous software industry. At the time, the most widely used high-level programming languages were FORTRAN and COBOL. ANSI FORTRAN and ANSI COBOL were standardised in 1966 and 1968, respectively, by the American National Standards Institute (ANSI). If programmers followed these standards while building programmes, then they would be able to run such programmes on any system that had an ANSI FORTRAN or ANSI COBOL compiler. This was the idea behind the guidelines. During the third generation, many new high-level programming languages were developed and made

available to users. PL/1, PASCAL, and BASIC were some of the most well-known programming languages.

The batch operating system was used on computers from the second generation. Prior to submitting their data and programmes to a central processing centre on these systems, users had to first prepare their data and programmes. Collection and delivery of these user jobs to computers in batches at regular intervals were the responsibility of the operator working out of the computer centre. The end result of their labour could then be retrieved by the users from the computer centre where it had been produced. Some users, particularly programmers, were annoyed by the unavoidable delay caused by the batch processing approach. This was because they had to wait for days in order to find and fix a few errors with the programming. This problem was addressed by Dartmouth College professors John Kemeny and Thomas Kurtz, who devised the timesharing operating system. The timesharing operating system enables multiple users to use the computer at the same time and share its resources, all while each user is under the impression that they have sole control of the device. To do this, simultaneous connections are made between a large number of independent on-line terminals operating at a slow pace and the primary computer. The implementation of the timesharing concept resulted in a significant increase in the productivity of programmers and made it possible for systems to be accessed online. As a consequence, new online applications were developed, such as reservation systems for airlines and interactive inquiry systems, amongst others.

Up to the year 1965, computer manufacturers would sell their wares in bundles that included both the hardware and any accompanying software at no additional charge. Customers, for instance, were given language translators for every language that could be understood by the device that they purchased.

From the point of view of the customer, the software was without cost. However, in 1969 IBM and other computer manufacturers started charging separate prices for their hardware and software, which caused the situation to shift in a different direction. Customers were able to invest solely in the software that they need and found valuable as a result of the decoupling of software from hardware.

Consumers, for instance, no longer have to buy all of the language translators that a computer is capable of supporting because they may now purchase only the language translators that they require. As a direct consequence of this, a number of brand-new software houses appeared, marking the beginning of the independent software industry.

During the time span including the third generation of computers, minicomputers were also created and brought to market. It wasn't until the early 1960s that the first mainframe computers were developed, and only the largest corporations had the financial means to purchase and utilise these machines. In order to fill the gaps that were left by the larger, more expensive, and slower mainframe systems, it was obvious that cheaper, more compact computers were required. In the 1960s, a number of businesspeople recognised the need for more portable computing devices and established new enterprises to meet that demand.

The first minicomputer to be made available to the general public was the PDP-8 (Programmed Data Processor), which was introduced by Digital Equipment Corporation (DEC) in 1965. Because of its compact size, it could be placed in the secluded nook of a room, and it did not require the constant attention of a computer operator.

Due to the fact that it was powered by a timesharing operating system, it was possible for multiple people to access it all at once from different locations within the same building.

Because it cost around one quarter as much as a typical mainframe system, it made it possible for smaller organisations to purchase computers.

The following are some of the characteristics of computers of the third generation:

1. They had a greater processing capacity than computers from the generation before them. They were capable of carrying out close to one million directives each second.

2. In comparison to computers of the second generation, these new machines were more compact, therefore they required less space.

3. When compared to computers from the previous generation, they consumed less energy and produced less heat. The spaces and places that housed computers of the third generation still needed to have the right cooling systems in place.

4. As a result, they required less upkeep and were more reliable than computers of the second generation; also, they were less likely to experience hardware failures.

5. They were speedier and had more primary and secondary storage than computers from the previous generation.

6. Because there was no need to manually assemble individual components into electronic circuits, the assembly process required less time and money due to a reduction in the amount

of labour performed by humans. As a direct consequence of this, the process of commercialising these innovations was simplified and required fewer resources. The production of IC chips, on the other hand, called for very sophisticated technology and an expensive infrastructure to be put up.

7. Because high-level programming languages have been standardised, it is now possible to easily port and run programmes that were generated on one machine on another.

8. The implementation of a timesharing operating system made it possible for multiple users to interact with these systems at the same time.

9. With the assistance of the timesharing operating system, programmers were able to cut the amount of time and money spent on developing software by several orders of magnitude.

- **FOURTH GENERATION (1975-1989)**

Following the year 1965, the annual average of the number of electronic components that could be packed onto a silicon chip rose. Large-scale integration (ISI) and subsequently very large-scale integration (VLSI) were quickly made practicable, allowing for the integration of more than a million electronic components on a single chip in a short period of time after this breakthrough. There has been an enormous leap forward in microprocessor development thanks to this work. A microprocessor is a type of integrated circuit that has all of the necessary circuitry on a single chip to carry out the primary functions of a computer, including arithmetic, logic, and control operations. With the advent of microprocessors, it is now possible to build an entire computer out of just a few primary storage chips, a microprocessor, and several auxiliary components. It marked the beginning of a new period of revolutionary change in society known as the personal computer (PC) revolution. Almost overnight, computers shrunk to sizes previously unimaginable. The cost to produce them decreased, and all of a sudden, anyone could purchase a computer.

When semiconductor memories took over from magnetic core memories in the fourth generation of memory technology, huge random access memories with lightning-fast access times emerged. There is also the fact that hard discs are becoming less expensive and more convenient to use because of their smaller dimensions and higher storage capacity. As a portable medium for transferring programmes and data from one computer system to another, floppy discs, in addition to magnetic tapes, had a great deal of popularity.

One more significant development that occurred during the fourth generation was the widespread implementation of high-speed computer networking. This development enabled multiple computers to connect with one another and share data. Local area networks, sometimes known as LANs, have gained popularity as a means of connecting computers found within an organisation or on a campus. In a similar vein, wide area networks, sometimes known as WANs, rose to prominence as a popular means of connecting computers over greater physical distances. As a direct consequence of this, the invention of networks, computers, and distributed systems took place.

On the front of software development, numerous new developments have developed to support the new technology of the fourth generation. For example, new operating systems have been developed and made available for personal computers (PCs). MS-DOS, MS-Windows, and Apple's in-house designed operating system were among the most well-known. Graphical user interfaces were developed by companies in response to the fact that personal computers were intended for use by people who lacked prior experience with computers (easier to use). In a graphical user interface, users can select different icons (pictures) and menus (lists of options) by using a pointing device called a mouse (U1).

This makes it possible for people who are new to using computers to quickly learn how to navigate a computer. In addition, many new applications designed to run on personal computers (PCs) have been developed in order to make PCs a more effective tool. Word processing and spreadsheet tools, as well as graphics applications, made it easy to create visuals and diagrams, making it easier to use data structured in columns and rows.

The ability to display several windows on a single terminal screen became increasingly popular throughout the fourth generation. As a result of this feature, users were able to simultaneously monitor a huge number of apps running in several windows on the same terminal screen. Throughout the course of the fourth generation, the UNIX operating system and the C programming language gained a significant amount of popularity.

The following is a list of properties that are shared by computers of the fourth generation:

1. Personal computers of the third generation were more portable and cheaper than mainframes or minicomputers.

2. Personal computers, on the other hand, did not necessitate that the rooms or locations in which they were installed be air-conditioned.

3. In comparison to computers of earlier generations, they had a lower energy consumption.

4. In comparison to computers of the third generation, they were easier to rely on, less likely to experience problems with their hardware, and required less upkeep overall.

5. In terms of speed and storage capacity, they were superior to computers from the third generation.

6. These were versatile machines that could be applied to a wide range of applications.

7. It is possible to build these devices at a lower cost since they do not necessitate the physical integration of individual components into electrical circuits. As a direct consequence of this, the process of commercialising these innovations was simplified and required fewer resources. On the other hand, the production of LSI and VLSI chips requires more advanced technology as well as an expensive infrastructure.

8. The utilisation of standard high-level programming languages made it feasible to easily port and execute programmes that were generated for one machine on another machine.

9. The graphical user interface, often known as GUI, made it simpler for first-time computer users to pick up the basics of using a computer more quickly.

10. The proliferation of applications designed specifically for personal computers has transformed these devices into powerful resources that can be put to use in a variety of settings, including the workplace and the home.

11. Sharing resources across multiple computers and their users was made possible by the existence of a computer network. This included the ability to share discs, printers, and other resources. They also made it possible for a plethora of new apps to be developed in which users of computers located in different parts of the world communicate with one another. Computer Supported Cooperative Working (CSCW), more commonly referred to as groupware, is a type of software application that enables multiple users to collaborate with one another on a single project while they are located in different locations. This type of programme is known as Computer Supported Cooperative Working (CSCW).

12. These systems offered add-on hardware features in addition to unbundled software, which gave customers the ability to spend their money solely on the hardware configuration and software that met their needs and was important to them.

- **FIFTH GENERATION (1989-PRESENT)**

Even with the fifth generation of personal computers, there was a continuing trend toward increasing the power of microprocessor processors, as well as increasing the storage capacity of both main memory and hard discs The fifth generation of VLSI (very large-scale integration) technology produced ULSI (ultra large scale integration). Microprocessor chips with 10 million electronic components might be produced using this method.

Factors such as main memory capacity have increased by four times every 18 months, as has the speed of microprocessors and hard drive space. Because of this, various operations that were previously only found in the central processing units of massive mainframes of the third and fourth generations were incorporated into the design of the fifth generation of microprocessors.

This resulted in incredibly powerful and small computers becoming more readily available at a lower cost, as well as the collapse of large, outdated mainframe computer systems.

As a result of the continuous advancement in computer technology, we are witnessing the production of smaller, more powerful computers nearly every year at the same price or even lower prices. Desktop PCs and workstations, servers, and supercomputers are all included in this category. Portable laptop computers, which provide users with the power of a PC while they are on the go, are also included in this category.

The technology behind storage expanded at a fast pace, which made it possible for newly introduced computers to feature ever-larger amounts of main memory and disc storage. Optic discs are a common portable mass storage device in fifth-generation technology D-ROM option (Compact Disk - Read Only Memory).

The number of computer networks has grown significantly over the fifth generation. Each day, advancements are made in communication technology, and an increasing number of computers are connected to one another. As a consequence of this pattern, the Internet, as well as other associated technologies and applications, came into being.

Through the use of the function known as electronic mail (often referred to as e-mail), Internet users from all over the world were able to communicate with one another in a matter of minutes.

Users of computers now have convenient access to a vast storehouse of information thanks to

the World Wide Web (known as WWW). E-commerce, virtual libraries, virtual classrooms, and remote schooling were some of the fascinating new uses that emerged during this time period.

Fifth-generation computers are becoming increasingly popular for a wide range of multimedia applications that deal with text, graphics animation, music, and video data due to their vast processing and storage capability. Textual information has a smaller data size than multimedia information because it requires less bits to represent images, animation, music, or video material in digital form. However, multimedia information requires more bits than textual information to do so.

The following is a list of properties that are shared by computers of the fifth generation:

1. Portable personal computers, also known as notebook computers, are far more portable and substantially smaller than fourth-generation personal computers. This allows users to use their computing skills regardless of where they are.

2. Personal computers designed for use as desktops or workstations and belonging to the fifth generation are several times more powerful than personal computers belonging to the fourth generation.

3. In contrast to fifth-generation mainframes, which require a cool atmosphere to operate, laptops, desktop PCs, and workstations don't usually need this.

4. They have a lower overall energy footprint than their predecessors.

5. In comparison to their predecessors, they are less prone to hardware failures, making them more reliable and reducing the amount of maintenance that is required.

6. A few of large-scale fifth-generation systems come equipped with a "hot-plug" function, which enables faulty components to be replaced without the need to power down the entire system. Because of this, the uptime for these systems is exceptionally high.

7. When compared to their predecessors, these devices have primary and secondary storage that is both faster and larger than before. Afterall, they are universally applicable tools.

9. The creation of these devices does not require the physical integration of individual components into electrical circuits; as a result, both the cost of human labour and the cost of assembly are reduced. As a direct consequence of this, the production of these systems in a

commercial setting is both easier and more cost-effective.

The production of IJLSI chips, on the other hand, calls for very sophisticated technology as well as an expensive infrastructure (affordable only to a few enterprises throughout the world).

10. Thanks to the usage of standard high-level programming languages, it is possible to swiftly transfer programmes that have been created on one computer to another machine and then run those programmes there.

11. The more user-friendly interfaces, with multimedia elements, facilitate the learning and use of the systems by all, and particularly by the younger generations.

12. As new and more sophisticated applications, particularly multimedia apps, become available, the systems are becoming more valuable in many fields of work.

13. The expansion of the reach of the Internet, in conjunction with the development of tools and applications that are based on the Internet, has caused these systems to have an effect on the everyday lives of regular people.

14. With these systems, users have the option of purchasing unbundled software and add-on hardware so that they can customise their systems to fit their needs and preferences.

We examined the development of computers over the course of five generations to have a better understanding of how quickly things have progressed over the past few decades.

Even, at this late date, technological progress in this sector is proceeding at the same rapid clip as before. In point of fact, the time in the history of computers with the most rapid expansion could still lie ahead of us.

1.5. SUMMARY

- It is from the verb "to compute" that we derive the English term "computer."
- After that, we will talk about the characteristics of computers, including Automatic Machine, Speed, Accuracy, Diligence, Versatility, and Memory Power.
- The many generations of computers, include the First Generation (1942–1955), the Second Generation (1955–1964), the Third Generation (1964–1975), the Fourth Generation (1975–1989), and the Fifth Generation (1989–present) (1989-Present).

1.6. KEYWORDS

- ***Computer Generations***
- ***Evolution***
- ***Characteristics***

1.7. REVIEWS QUESTION

1. The third generation of computers ranges from..... to.....

   a) 1955-1964
   b) 1964-1975
   c) 1942-1955
   d) 1975-1989

2. Created an electrical gadget that can solve mathematical equations.

   a) Dr John Atanasoff
   b) Herman Hollerith
   c) Professors J. Presper Eckert
   d) All of these

3. Data handling is the action of utilising a computer to process data.

   a) True
   b) False

4. ULSI stands for……

   (a) Ultra Located Scale Integration

   (b) Ultra Large-Scale Integration

5. Portable computers are also known as.....

   (a) Book computer

   (b) Notebook computers

6. Computers of the second generation were made utilising

a) Vacuum tubes

b) Transistors

c) IC Chip

d) None of above

1.8. FURTHER READING

Computer Fundamentals by P.K. Sinha BPB Publications.

https://www.geeksforgeeks.org/generations-of-computers-computer-fundamentals/

UNIT 2: BLOCK DIAGRAM OF COMPUTER

CONTENTS

OBJECTIVES:

Unit are the essential components of a computer:

- Input Unit
- Output Unit
- Memory/Storage Unit
- Arithmetic Logic Unit
- Control Unit
- Central Processing

2.1. INTRODUCTION

Data, photos, sound, and graphics may all be processed by a computer. They have the ability to tackle complex issues fast and accurately.

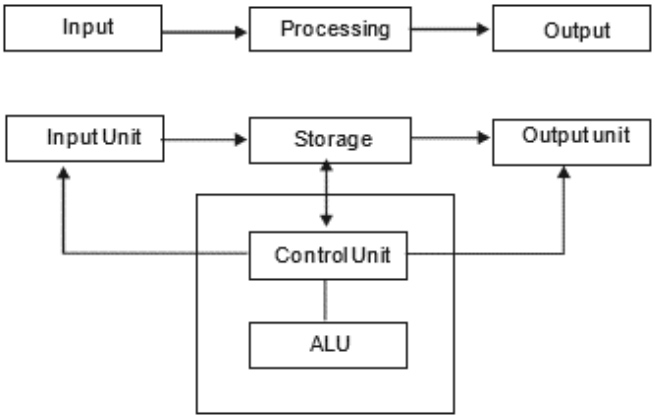


Fig. 2.1 Block Diagram Of Computer

2.2. INPUT UNIT

Data and instructions are two things that computers need in order to solve any problem. The result of this is that we must enter data and commands into the computers. At any given time, the input unit may be made up of one or more input devices. The keyboard is one of the most frequently used input devices. Input technologies such as the mouse, the floppy disc drive, magnetic tape, and others are utilised extensively. All input devices are responsible for carrying out the following functions:

• Comply with directives and information received from the outside world.

• Convert it into a file format that can be read and understood by the computer.

• Move the newly produced data into the computer system so that they may be processed.

2.3. STORAGE UNIT

Until they are processed, the data and instructions entered into the computer through the input unit are held in its storage area until they may be used. Both intermediate and final findings are

saved before being sent to the output devices. Additionally, it stores the information for potential use at a later time. The various storage devices that can be found in a computer system can be divided into two categories:

- **PRIMARY STORAGE**

Data is stored and delivered in a timely manner.

This memory is typically used to store the programme that is now being processed by the computer, the information that was obtained from the input device, as well as the intermediate and final outcomes of the programme. Only temporary information is stored in the primary memory.

The information will be lost once the machine is powered down. The data must be moved to the secondary memory in order to be permanently stored. When compared to secondary storage, main storage is more expensive. As a result, the primary storage capacity of most computers is restricted.

- **SECONDARY STORAGE**

Secondary storage is utilised in the same way as archives are used. It houses a variety of applications, papers, databases, and other data. Before you can run a programme on the computer, it must first be copied to the main memory.

The findings are preserved in the secondary memory every time they are saved. When compared to primary memory, secondary memory operates at a lower speed yet costs significantly less. It is common practise to make use of hard drives, CDs, and various other supplemental memory devices.

2.4. MEMORY SIZE

The binary system, which consists of 0s and 1s, is used by all digital computers. An 8-bit code is used to represent each character or integer. A byte is a collection of eight bits. A character takes up one byte of memory. A number takes up two bytes of memory.

A byte is the amount of memory space occupied. The primary storage capacity is set in KB (Kilobytes) or MB (Megabytes) (Megabyte). One KB equals 1024 bytes, whereas 1000 KB equals one MB. The main storage of a normal PC generally starts at 16 megabytes. Memory

capacities of 32 MB, 48 MB, 128 MB, and 256 MB are prevalent in PCs.

2.5. OUTPUT UNIT

Data and outcomes are conveyed to the outside world via the computer's output unit. Among the various output devices, printers and video displays are by far the most widely used (VDUs). The following are examples of common output devices: floppy disc drives, hard disc drives, and magnetic tape drives.

2.6. ARITHMETIC LOGICAL UNIT

The computer's Arithmetic Logic Unit, or ALU, does all computations. In addition to this, it evaluates alternatives and comes to conclusions.

The ALU is capable of performing elementary mathematical operations such as addition, subtraction, multiplication, and division in addition to logical mathematical operations such as >, =, and amongst others. When computations need to be done, the control unit will send data from the storage unit to the ALU via an intermediate unit. After the computations have been finished, the control unit will transmit the results to the storage unit, and then the storage unit will transmit the results to the output unit so that they can be shown.

2.7. CONTROL UNIT

It is in charge of the various components that make up the computer.

After the control unit has been given the data that the user has input, it will then instruct the input unit where the data should be stored.

It regulates the transfer of information between the system's storage and central processing unit (ALU). In addition to this, it controls how the data travels between the ALU and the storage unit. The control unit is the central nervous system of the computer, and it is responsible for regulating and synchronising the machine's functions.

2.8. CENTRAL PROCESSING UNIT

As a whole, these two components are known as the computer's Central Processing Unit (CPU) (CPU). The central processing unit (CPU), which operates much like a human brain, is responsible for the following activities:

It is responsible for carrying out all computations, making all decisions, and exercising control over all computer units.

A personal computer may contain a variety of different CPU-ICs, including Intel 8088, 80286, 80386, 80486, Celeron, Pentium, Pentium Pro, Pentium II, Pentium III, Pentium IV, Dual Core, AMD, and others.

2.9. SUMMARY

• A computer's input, output, and memory devices are examples of what are referred to as "blocks" in a block diagram of that computer's components.

• The control processing unit is formed when the ALU and CU are combined.

2.10. KEYWORDS

Data processing is the action of utilising a computer to process data. There was a time when the term "generation" was used to differentiate between distinct hardware innovations, but it has since been expanded to include both hardware and software.

There are several more names for **integrated circuits**, including chips and integrated circuits. Complex circuits have been carved into chips composed of semiconductor material that are quite small in size (silicon). Pins are set on a 0.1-inch (2.54-millimeter) grid in a plastic container that houses the integrated circuit. This grid is compatible with the hole spacing of stripboards and breadboards. Inside of the packaging are very fine wires that make the connection from the chip to the pins.

Medium-scale integration is a term used in the semiconductor chip manufacturing industry. Medium-Scale Integration is characterised by the presence of hundreds of transistors on each chip of an integrated circuit (MSI).

Small-scale integration (SSI) refers to the fact that the early integrated circuits only a few transistors. They employed circuits with tens of thousands of transistors, dubbed "Small-Scale Integration" (SSI).

The data and instructions that are input into the computer through the input unit are temporarily saved in the storage unit of the machine before being processed. Both intermediate and final findings are saved before being sent to the output devices.

Integrating thousands of transistors onto one chip is the process of creating integrated circuits. This technology is known as VLSI (very large-scale integration) (VLSI).

2.11. REVIEW QUESTIONS

1.What exactly does it mean to say that something is a computer? What exactly does it mean to say that something is a "data processor"?

2. Can you explain what exactly is meant by the term "data processing"? Make sure you understand the difference between information and data. Which of these is better for the general population, and why is that so?

3. Describe a computer and then make a list of some of the most important features it possesses.

4. What exactly is the meaning behind the phrase "garbage in, rubbish out"?

5. Who is considered to be the "father" of modern digital computers, and why is this title given to them?

6. Who was the one who first thought of the concept of a stored programme? In what ways does this idea contribute to the bigger picture?

7. In the context of modern digital computers, what does it mean to refer to a computer as a "stored programme digital computer"?

8. Which computer was the first to be manufactured for sale to the general public? When and where did it first become available to the public?

9. Please write out the full names of the computer manufacturers listed below in the following order: IBM, ENIAC, EDVAC, EDSAC, and UNIVAC.

10. What does it mean when people talk about "generation" in the context of computers? How many different iterations of computers have been built up to this point?

11. Compile a list of the many generations of computers, together with the fundamental qualities possessed by computers from each of those generations.

12. Name the key piece of hardware technology that was utilised in the creation of each of the five generations of computers.

13. Identify the most important piece of software engineering that went into the creation of each of the five generations of computers.

14. Can you explain exactly what an IC is? In what ways does it contribute to the shrinking of the computer?

15. Why was it so much more challenging and expensive to bring the first and second generations of computers to market than it was to bring the later generations of computers to market?

16. Identify the technology that was utilised in the construction of the primary memory of computers of the first, second, third, and fourth generations.

17. Determine the secondary storage medium that is used the most frequently, such as first-generation, second-generation, third-generation, fourth-generation, and fifth-generation computers.

18. What does it mean to say that something is a microprocessor? What kind of effects did it have on the field of computer manufacturing?

19. Please give an overview of some of the applications that were made possible as a direct result of the development of computer networks.

20. Describe a few of the applications that came about as a direct result of the launch of the Internet.

21. Determine some representative examples of computer systems that were available during each of the five generations of computers.

2.12. FURTHER READING

- *Fundamental Computer Concepts*, William S. Davis.
- *Fundamental Computer Skills*, Feng-Qi Lai, David R. Hofmeister
- http://books.google.co.in/bkshp?hl=en

UNIT 3: INTRODUCTION TO DATA REPRESENTATION

CONTENTS

OBJECTIVES:

- Learn about data representation
- Number System

3.1. INTRODUCTION

Although the bit is the most fundamental memory unit of a digital computer, maintaining a value of 1 or 0 (or, equivalently, ON or OFF), the majority of the information that is relevant to the user requires a more complicated representation. As a consequence of this, it is feasible to organise bits into groups consisting of a predetermined number of bits each and to build computer hardware that can process these groups of bits as if they were whole units. The byte, which is comprised of a total of eight bits, is the most fundamental unit that can be found in modern computers. The most common sizes for data storage in computers are two bytes, four bytes, and eight bytes.

3.2. DATA REPRESENTATION

Before we can investigate how a computer processes data, we need to first understand the structure of the memory in which the data is kept on the computer.

Data in character form and data in numerical form are the two basic types of data that computers store and process respectively. Characters include both regular letters and unique symbols as well. Computers can be taught to read a list of names, then sort the list alphabetically, and finally publish the alphabetized list. For example. VINEET, PRADIP, and GANESH are examples of names that might be entered into an input device, then stored in memory, sorted alphabetically by the software, and finally displayed as the characters VINEET, PRADIP, and GANESH.

Decimal numbers, such as 1234, 456, and so on, are another kind of data. Addition, subtraction, multiplication, and division are only a few of the mathematical operations that can be used to work with numbers. Numbers are given values, and the resulting processing generates new values for those numbers.

Text and numbers entered into computers and the output generated by computers must be readable by people. This includes both the input and output. The use of natural language symbols in addition to decimal digits is appropriate for this task. These are the elements that make up the external data's representation. Computers must store and process data in a manner consistent with the technology used to represent the data in order to ensure that the representation is accurate. As a consequence of this, the first step is to determine the internal representation that most accurately corresponds to the internal representation, and vice versa.

3.3. BINARY NUMBER SYSTEM

The binary number system is quite similar to the decimal number system; the primary difference is that the base in the binary system is 2 instead of 10. We are only allowed to use two symbols or digits in this particular numerical system (0 and 1). The single digit that is the most significant in a binary number is 1, which is one less than the base, and each place in the number reflects a power of the base (2).

As a consequence of this, the position that is the most rightmost in this system is 2's (21), the position from the right is 2's (20), and so on until we reach the position that is the most rightmost in this system for 4's (22), 8's (23), 16's (24), and so on. As a result, the decimal representation of binary number 10101 (written as l010l2) is:

$$(1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 16 + 0 + 4 + 0 + 1 = 21$$

In order to be specific about which system we are referring to, it is a common practice to indicate the base as a subscript. Hence, we write:

$$10101_2 = 21_{10}$$

The short form of "binary digit" is *bit.* Hence, a "bit" in computer terminology means either a 0 or 1.

Binary numbers with n bits are called n-bit numbers. Figure 1.3 contains a listing of all possible 3-bit values, along with their corresponding decimal representations. Due to the fact that the binary number system only consists of two digits, 0 and 1, the binary counterpart of the decimal number 2 is represented as the number 10. (also written as one and zero). A 3-bit integer can have one of eight possible values ranging from 0 to 7, as illustrated in Figure 1.3, because there are only 8(23) different combinations of 0s and 1s that can be used.

This is another important point to keep in mind. In point of fact, an n-bit binary representation of any decimal number between 0 and 2n-1 may be created for any decimal number.

Each and every computer store numbers, letters, and various other special characters in a form known as binary. In a wide range of contexts, computer professionals will find themselves needing access to the unprocessed data that is kept in the memory of a computer. It is common practise to accomplish this by sending the contents of the memory to a printer and having it

printed. This type of printing is referred to as a "memory dump." There are several pages of 0s and 1s in memory dumps. Experienced computer professionals would have a hard time and be prone to mistakes when dealing with enormous numbers.

As a direct consequence of this, the octal and hexadecimal number systems are widely utilised for the purpose of providing binary shorthand notations.

3.4. OCTAL NUMBER SYSTEM

The number 8 serves as the foundation for the octal number system.

As a direct consequence of this, there are only eight signs or numbers utilised: 0, 1, 2, 3, 4, 5, 6, and 7. The numbers 8 and 9 do not exist in this system. The highest single digit number is seven (one less than the base 8). A power of the base is represented by each place in an octal number (8). As a result, the decimal equivalent of the octal number 2057 (written as 20578) is

$(2 \times 8^3) + (0 \times 8^2) + (5 \times 8^1) + (7 \times 8^0) = 1024 + 0 + 40 + 7 = 1071$ As a result, $2057_8 = 1071_{10}$

Because the octal number system has just 8 digits, 3 bits ($2^3 = 8$) are adequate to express any octal number in binary.

3.5. HEXADECIMAL NUMBER SYSTEM

The basis of the hexadecimal number system is 16. A total of 16 symbols or numbers are thus generated. The first ten digits of the decimal number system are the same digits - 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. The symbols A, B, C, D, E, and F indicate the decimal values 10, 11, 12, 13, 14, and 15, respectively, for the remaining six digits.

As a result, the highest single digit is 15 instead of F. There is one fewer than in the base 16 Each hexadecimal position represents a base power in the hexadecimal number system (16). As a result, the decimal equivalent of the hexadecimal number lAF (1AF16) is:

$\underline{23}\ (1 \times 16^{2-}) + (A \times 16^1) + (F \times 16^0) = (1 \times 256) + (10 \times 16) + (15 \times 1) = 256 + 160 + 15 = 431$

Hence, **$IAF_{16} = 431_{10}$**

Because the hexadecimal number system has just 16 digits, 4 bits (24 = 16) are adequate to express any hexadecimal integer in binary.

3.6. CONVERTING FROM ONE NUMBER SYSTEM TO ANOTHER

We find numbers stated in the decimal number system to be far more understandable than those expressed in any other number system. This is due to the fact that we have been utilising decimal numbers in our daily lives since infancy; nevertheless, every number in one number system may be represented in any other number system. Because the input and output values must be in decimal, computer experts must frequently translate numbers from other systems to decimal and vice versa. Converting numbers from one base to another can be done in a variety of ways. The methods for converting from one base to decimal and back again are described in the following paragraphs.

- **CONVERTING FROM ANOTHER BASE TO DECIMAL**

The following steps need to be taken in order to convert a number written in any other base system into a number written in base 10 (decimal):

Step 1: Determine each digit's column (positional) value (this depends on the position of the digit and the base of the number system).

In Step 2, multiply the digits in each column by the column values that were obtained in Step 1.

In the third step, add all of the products that you computed in the second step together. The sum expressed in decimal form is referred to as the total.

Example: 1: $11001_2 = ?_{10}$

Step 1: Determine column Values

Column Number (From Right)	Column
	$2^0 = 1$
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$

Step 2: Multiply the column values by the corresponding column digits.

16	8	4	2	1
x1	x1	x0	x0	x1
16	8	0	0	1

Step 3: Sum up the products

16 + 8 + 0 + 0 + 1 = 25

Hence, $11001_2 = 25_{10}$

Example: 2: $4706_8 = ?_{10}$

Step 1: Determine column values

Column Number (From Right)	Column
1	$8^0 = 1$
2	$8^1 = 8$
3	$8^2 = 64$
4	$8^3 = 512$

Step 2: Multiply the column values by the corresponding column digits

512	64	8	1
x 4	x 7	x 0	x 6
2048	448	0	6

Step 3: Sum up the products

2048 + 448 + 0 + 6 = 2502

Hence, $4706_8 = 2502_{10}$

Example: 3: $1AC_{16} = ?_{10}$

$1AC_{16}=1 \times 16^2 + A \times 161 + C \times 16^0$

$=1 \times 256 + 10 \times 16 + 12 \times 1$

$=256 +160 +12$

$=428_{10}$

Example: 4: $4052_7=?_{10}$

$4052_7 = 4 \times 7^3 + 0 \times 7^2 + 5 \times 7^1 + 2 \times 7^0$

$= 4 \times 343 + 0 \times 49 + 5 \times 7 +2 \times 1$

$= 1372 + 0 + 35 + 2 = 1409_{10}$

Example: 5: $4052_6=?_{10}$

$4052_6 = 4 \times 6^3 + 0 \times 6^2 + 5 \times 6^1 + 2 \times 6^0$

$= 4 \times 216 + 0 \times 36 + 5 \times 6 +2 \times 1$

$= 864 + 0 + 30 +2$

$= 896_{10}$

- **CONVERTING FROM DECIMAL TO ANOTHER BASE**

The following steps must be taken in order to change a number written in base 10 (decimal) into a number written in a different base.

In the first step of the process, divide the decimal number by the value of the new base.

Step 2: Record the residual that was obtained from Step 1 as the rightmost digit (the digit with the least amount of significance) of the new base number.

The quotient from the previous division will be divided by the new base in the third step.

The next digit (to the left) of the new base number should include the residual from Step 3. This completes Step 4. Steps 3 and 4 should be repeated until the quotient in Step 3 equals zero, and the remainders should be recorded from right to left as you go. Take note that the digit with the most significance in the newly derived base number will be the last remainder that was acquired.

Example:

Steps 1 and 2: 25/2 = 12 and remainder 1
Steps 3 and 4: 12/2 = 6 and remainder 0
Steps 3 and 4: 6/2 = 3 and remainder 0
Steps 3 and 4: 3/2 = 1 and remainder I
Steps 3 and 4: 1/2 = 0 and remainder 1.

As mentioned in Steps 2 and 4, the remainders are now arranged in the reverse order, making the first remainder the least significant digit (LSD)and the last remainder the most significant digit (MSD).

Hence, $25_{10} = 11001_2$

Compare the result with the result obtained in above Example 1.

- **CONVERTING FROM A BASE OTHER THAN 10 TO ANOTHER BASE OTHER THAN 10**

To change a number written in a base other than 10 into a number written in a base other than 10, use one of the following methods:

The first thing you need to do is convert the initial number to a base number (decimal).

Step 2: Convert the decimal number obtained in step 1 by making use of the newly acquired base number.

Example: $545_6 = ?_4$

Step 1: Convert from base 6 to base 10

$545 = 5 \times 6^2 + 4 \times 6^1 + 5 \times 6^0$

$= 5 \times 36 + 4 \times 6 + 5 \times 1$

$= 180 + 24 + 5$

$= 209_{10}$

Example: **Step 2:** Convert 209_{10} to base 4

	4 209	Remainder
	52	1
	13	0
	3	1
	0	3

$209_{10} = 3101_4$

Therefore, $545_6 = 209_{10} = 3101_4$

Hence, $546_6 = 3101_4$

The preceding example provides a visual representation of the steps required to convert a binary number to an octal number.

A fast and straightforward method for converting binary to octal. The following is a list of the steps involved in this process:

Divide the binary digits into three different groups as the first step (starting from the right).

In the second step, you will reduce each set of three binary digits to a single octal digit. Due to the fact that the octal number system only contains eight digits (0 to 7), the technique for converting binary to decimal in this stage only requires three bits (23 = 8), which is sufficient to represent any octal number.

Example: $101110_2 = ?_8$

Step 1: Divide the binary digits into groups of 3, starting the right (LSD).

101 110

Step 2: Convert each group into one digit of octal (use binary- to- decimal conversion method).

$101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$ $\quad 110_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 4 + 0 + 1$ $\quad 4 + 2 + 0$

$= 5_8$ $\quad 6_8$

Hence, $101110_2 = 56_8$

- **SHORTCUT METHOD FOR OCTAL TO BINARY CONVERSION**

Listed below are the steps that make up this process:

The first step is to convert each octal digit into a three-digit binary number (the octal digits may be treated as decimal numbers for this conversion).

The second step is to create a single binary number by combining all of the binary groups, which each have three digits.

Example: $562_8 = ?_2$

Step 1: Convert each octal digit to 3 Binary digits.

$5_8 = 101_2$

$6_8 = 110_2$

$2_8 - 010_2$

Step 2: Combine the binary groups.

$562_8 = \underset{5}{\underline{101}} \quad \underset{6}{\underline{110}} \quad \underset{2}{\underline{010}}$

Hence, $562_8 = 101110010_2$

- **SHORTCUT METHOD FOR BINARY TO HEXADECIMAL CONVERSION**

Listed below are the steps that make up this process:

In the first stage, the binary digits are divided into four separate groups (starting from the right).

Each binary group of four digits must then be converted into a single Hexadecimal digit. A to F correspond to the decimal numbers 10-15 in hexadecimal digits 0 to 9, whereas C to G correspond to the decimal values 11-15. As a consequence of this, we make use of the method for converting binary to decimal for this stage, and we represent the decimal values 10 to 15 as the hexadecimal letters A to F.

Example: $11010011_2 = ?_{16}$

Step 1: Divide the binary digit into groups of 4, starting from the right (LSD)

1101 0011

Step 2: Convert each group of 4 binary digit to 1 hexadecimal digit

$1101_2 = 1x2^3 + 1x2^2 + 0x2^1 + 1 \times 2\ 0011_2 = 0x2^3 + 0x2^2 + 1x2^1 + 1x2^0$

$= 8+4+0+1 = 0+0+2+1$

$= 13_{10} = 3_{16}$

$= D_{16}$

Hence, $11010011_2 = D3_{16}$

- **SHORTCUT METHOD FOR HEXADECIMAL TO BINARY CONVERSION**

The following is a list of the steps involved in this process:

First, convert each hexadecimal digit's decimal value to its corresponding four-digit binary representation.

The second step is to obtain a single binary number by adding all of the binary groups, which consist of four digits each.

Example: $2AB_{16} = ?_2$

Step 1: Convert decimal equivalent each hexadecimal digit to 4 binary digits.

$2_{16} = 2_{10} = 0010_2$

$A_{16} = 10_{10} = 1010_2$

$B_{16} = 11_{10} = 1011_2$

Step 2: Combine the binary groups.

$$2AB_{16} = \underset{2}{0010}\ \underset{A}{0010}\ \underset{B}{1011}$$

Hence, $2AB_{16} = 001010101011_2$

Hexadecimal, binary, and octal are the three different number systems.

It is important to take note that the maximum number that can be represented by a single octal digit (7) is identical to the maximum value that can be represented by three binary digits.

The range of possible values that may be represented by a single octal digit is the same as the range that can be represented by three binary digits.

The replacement that takes place when binary digits are changed to octal digits is the range one to three.

When taking a memory dump, computers that print octal numbers rather than binary numbers use one-third less space and time than those that produce binary numbers.

This is because octal numbers are converted to binary numbers before printing.

To further explain, the maximum value in binary that can be expressed by four digits equals the maximum number in the hexadecimal system.

As a result, the available values for a single hexadecimal digit are equal to the possible values for four binary digits. Because of this, utilising the hexadecimal shortcut notation results in a reduction of the required amount of space as well as the amount of time by a factor of four.

3.7. SUMMARY

- In octal number system there are only eight symbols or digits i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8.
- In hexadecimal number system each position represent a power of the base 16.
- Graphical representation of data i.e. bar graph, excel chart wizard etc.

3.8. KEYWORDS

When it comes to the **binary number system**, the only difference is that its base is 2, rather than 10, and it only employs two symbols instead of the decimal one (0 and 1).

n-bit value: An n-bit number is the name given to a binary number that contains n bits.

Due to the fact that there are a total of ten symbols or digits, the base in the **decimal number system** is equal to the number ten (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9).

3.9. REVIEW QUESTIONS

1. The value range of one digit of octal duplicates the value range of three digits of binary.
   (a) True (b) False
2. A powerful computer is capable of performing several billion (10^9) arithmetic operation per second.
   (a) True (b) False
3. 3-digits bit number, i.e. 101 has .....decimal equivalent.
   (a) 6 (b) 8
   (c) 2 (d) 5
4. Memory dump is a printout of memory contents on a paper.
   (a) True (b) False
5. Find decimal equivalent of following binary numbers:
   (a) 1101011
   (b) 1000
   (c) 10110011
   (d) 11011101
   (e) 1110101
   (f) 1000
   (g) 10110001100
   (h) 110001
   (i) 1010101100
   (j) 111
6. Find octal equivalent of the binary numbers of Question 22.
7. Find hexadecimal equivalent of the binary numbers of Question 22.
8. Convert the following numbers to decimal numbers:
   (a) 110110_2 (b) 2573_8
   (c) $2A3B_{16}$ (d) 1234_9
9. Carry out the following conversions:
   (a) $125_6 = ?_4$ (b) $24_5 = ?_3$
   (c) $ABC_{16} = ?_8$

3.10. FURTHER READINGS

- *Fundamental Computer Concepts*, William S. Davis.
- *Fundamental Computer Skills*, Feng-Qi Lai, David R. Hofmeister.
- http://books.google.co.in/bkshp?hl=en

UNIT 4: MEMORY- INTRODUCTION

CONTENTS

OBJECTIVES:

- Learn about Memory
- Learn about ROM, RAM

4.1. INTRODUCTION

The central processing unit (CPU) of a computer houses the circuitry that controls the other components of the computer and processes data. On the other hand, there is no storage space for the programmes and data that are required for the data processing. The central processing unit (CPU) includes a large number of registers that can store data and instructions; however, each register is limited to holding just a few bytes at a time. They can only carry one or two instructions and the data associated with those instructions because of their limited size. In order to keep the CPU busy, the instructions and data for a programme running on a CPU would have to be retrieved and loaded into CPU registers sequentially as the programme ran. Data and instructions would have to be loaded one at a time into the registers. Since the CPU can process data at a much higher pace than the disc, this results in a slower transfer rate from disc to CPU registers. CD to CPU transfer speeds are substantially lower than the CPU's. CPUs process data at a rate of 5 nanoseconds per byte, while disc readers read data at about 5 microseconds per byte. As a consequence of this, a central processing unit (CPU) may process one thousand bytes in the same amount of time that it takes a disc to give one byte of data. Even if a system had a fast central processing unit (CPU), the overall performance would be excruciatingly sluggish due to this. In order to find a solution to this issue, there needs to be a storage space that is sufficiently large so that it can accommodate the data and instructions for the program(s) that the CPU is now working on. In comparison to the amount of time it takes to retrieve and load data from disc storage into CPU registers, this storage space's loading and retrieval time should be minimal.

4.2. UNIT OF MEMORY

If you start with the "Unit of Memory," you'll have a better grasp of terminology like "Kilobyte" and "Megabyte" and others connected to computer memory.

The tables that are presented below will assist you in understanding:

4.3. TYPES OF MEMORY

1) RANDOM ACCESS MEMORY

A computer's RAM, or volatile memory, is generally referred to when people talk about its memory. On the motherboard or on a smaller circuit board that is attached to it, the integrated circuit chips that make up this memory can be discovered (see Figure 1.1 for an illustration of

this). The mainboard of a computer is designed in such a way that it is possible to increase the amount of memory it can hold by by adding additional memory chips. As a consequence of this, if you believe that your computer need more memory, you are able to purchase additional memory chips and place them in the slots that are now vacant on the motherboard. In most cases, this is taken care of by service engineers. Additional random access memory chips can be added to a computer through the use of single-in-line memory modules, which plug into specific sockets on the motherboard (SIMMs).

Name	Symbol	Binary Measurement	Decimal Measurement	Number of Bytes	Equal to
kilobyte	KB	2^10	10^3	1,024	1,024 bytes
megabyte	MB	2^20	10^6	1,048,576	1,024 KB
gigabyte	GB	2^30	10^9	1,073,741,824	1,024 MB
terabyte	TB	2^40	10^12	1,099,511,627,776	1,024 GB
petabyte	PB	2^50	10^15	1,125,899,906,842,624	1,024 TB
exabyte	EB	2^60	10^18	1,152,921,504,606,846,976	1,024 PB
zettabyte	ZB	2^70	10^21	1,180,591,620,717,411,303,424	1,024 EB
yottabyte	YB	2^80	10^24	1,208,925,819,614,629,174,706,176	1,024 ZB

Table 4.1

2) READ ONLY MEMORY

It is a non-volatile memory chip, known as read-only memory (ROM), that stores data in a manner inaccessible to ordinary programmes and hence cannot be changed. ROM is a form of Random Access Memory (RAM). The process of recording data permanently into this kind of memory is referred to as "burning in the data." Using fuse-links to save the data in such memory is to blame. The process of burning a fuse link is irreversible. No changes can be made to the data stored on the ROM chip; it can only be read and used. Because of this characteristic, it is known as read-only memory (ROM). The data that is stored in ROM chips is not lost when the power is switched off or when there is a disruption in the power supply. This is in contrast to RAM chips, which are volatile. Field stores, permanent stores, and dead stores can all be used to describe ROM.

When storing data and programmes that aren't subject to regular or large updates, ROMs are a common choice. Wired electrical circuits are used for the vast majority of computer processes. However, in order to carry out particular higher-level processes that are also frequently utilised, it is necessary to make use of exceedingly complicated electronic circuits. As a consequence of this, rather than designing electronic circuits to carry out particular functions, specific computer programmes are developed to do so.

These programmes are referred to as micro programmes due to the fact that they are responsible for low-level machine functions and serve as hardware replacements. Microprograms are stored in a way that makes it impossible for end users to modify them in ROMs.

It is a microprogram when a computer system is powered up and the instructions that must be followed are included in the microprogram.

This microprogram is referred to as the "system boot programme," and it contains a set of start-up instructions for assessing whether or not components of the system, such as memory, I/O devices, and other components, are functional.

After searching for an operating system, it loads the operating system's core components into the device's volatile RAM and displays the first prompt. This mini-program needs to be preserved even after the computer has been powered off since it is used each and every time the machine is switched on. As a consequence of this, ROM is a suitable media for it to be stored on.

3) PROGRAMMABLE READ ONLY MEMORY

Two forms of read-only memory are user-coded and original equipment manufacturer (OEM)-programmed ROM (ROM). The producer of the electronic device that employs it writes data into a read-only memory (ROM) chip that has been pre-programmed by the manufacturer. Manufacturers of PCs may choose to save the system boot programme on a ROM chip that is embedded in the motherboard of every single one of their devices. For each printer that it manufactures, for instance, a printer manufacturer might save the printer controller software in a ROM chip that is located on the circuit board of the printer. ROMs that have been pre-programmed by the manufacturer are typically only used in circumstances in which there is a significant demand for such ROMs. It is important to keep in mind that the makers of electronic devices supply ROM chips that have been pre-programmed by the manufacturer. Users are unable to modify the programmes or data that are saved in ROM chips. However, a user-programmed ROM, on the other hand, allows the user to load and store "read-only" programmes and data into the ROM. This type of ROM is called a user-programmable ROM. A user can "customise" a system by converting their own programmes into micro-programs and putting them in a user-programmable ROM chip. This allows the user to make the system more specific to their needs. This kind of ROM is referred to as programmable read-only memory (ROM) since it can be altered through programming (EPROM). User programmes can typically be executed in a much shorter amount of time once they have been saved in a PROM chip than was previously required. When the chip is programmed, the PROM transforms into a ROM, although this does not happen immediately. In other words, the data stored there can only be retrieved by reading it (it cannot be changed). As a non-volatile storage media, PROM ensures data integrity even if the power is interrupted or shut off.

4) ERASABLE PROGRAMMABLE ONLY MEMORY (EPROM)

After it has been written, the information that is stored in a ROM or PROM chip cannot be altered in any way. This issue can be remedied by the use of a technology known as EPROM, which stands for erasable programmable read-only memory. An EPROM chip can be reprogrammed to store fresh data after the data stored in it has been removed. This capability is hinted at by the chip's name. To test the performance of a computer system with new applications, researchers and developers (experimenters) frequently use EPROM devices. Additionally, EPROMs can be useful in situations where a programme is stored in an unmodified ROM but must be updated due to an unanticipated incident. When an EPROM is

utilised, the data that has been recorded in it can only be "read," and the data will remain on the chip until it is deleted. This is because an EPROM does not have a write function.

To erase the data stored on EPROM chips, one of two methods is used: either the chip is exposed to ultraviolet light for a predetermined period of time, or high-voltage electric pulses are applied to the device. Ultra Violet EPROM (also known as UVEPROAM) is one variety, while Electrically EPROM (also known as EPROAM) is another (EEPROM).

To modify the data contained on EEPROM chips, compared to UVEPROM chips, is easier. EPROM is also referred to as flash memory because of the simplicity with which programmes stored on EPROM can be modified. One of the most common types of current input/output and storage devices uses flash memory, such as USB (Universal Serial Bus) flash drives and MP3 music players.

5) CACHE MEMORY

A computer's CPU can access data from its main memory at a pace that is around one hundred times quicker than the rate at which it can retrieve data from a high-speed secondary storage medium like a disc, so using main memory significantly reduces the disk-processor performance mismatch. Even when using main memory, the CPU encounters a bottleneck due to a performance disparity of 1–10 between the processor and memory, resulting in slow instruction processing times.

This speed discrepancy exists even when main memory is used. In other words, the rate at which data can be retrieved from memory may be approximately ten times slower than the rate at which data can be processed by the CPU. It should come as no surprise that reducing the speed gap between the processor and the memory can significantly boost the overall performance of a processor. A common solution for this problem is to use cache memory, which can also be called "cash." There is a little amount of memory that is located between the central processing unit (CPU) and the main memory.

This memory is extremely quick, and its access times are much closer to the processing speed of the CPU. It acts as a high-speed buffer between the central processing unit (CPU) and the main memory, storing highly active data and instructions momentarily while the operation is being carried out.

The performance of processing can be improved by ensuring that the data and instructions

required for present processing are stored in the cache rather than the main memory. Main memory is slower than cache memory. Cache memory is located between the central processing unit (CPU) and the main memory.

It is characterised by its extreme speed and its small size. Its access time is quicker than the processing speed of the central processing unit. It acts as a high-speed buffer between the central processing unit (CPU) and the main memory, storing highly active data and instructions momentarily while the operation is being carried out.

4.4. SUMMARY

• The hardware necessary for data processing is contained within the central processor unit (CPU).

• The amount of memory that can be stored on a computer motherboard can have its capacity readily expanded by installing additional memory chips.

• Micro programmes are one-of-a-kind computer programmes that are utilised in the building of electronic circuits that carry out functions.

• An electronic unit or piece of electronic equipment that uses a ROM is one in which data is burned into the menu structure of the electronic unit or piece of electronic equipment. Electronic equipment's menu structure stores the data that is stored on the disc.

4.5. KEYWORDS

RAM clips that are designed to slot into specific sockets on the mother board are known as ***single-line memory modules.***

***PROM*:** When an electrical gadget uses a ROM programmed by its maker, the data is burnt into it. This type of Rom is also known as a read-only memory (ROM).

***Ultra violet EP-ROM (UVEPOAN)*:** In an ultra violet EP-ROM, the information that has been stored is deleted through the use of high-voltage electric pulses (UVEPOAN).

When processing particularly active data and instructions, the data and instructions might be temporarily stored in ***CACHE MEMORY.***

Cache memory is also used to speed up the processing of data and instructions.

4.6. REVIEW QUESTIONS

1. The extra RAM chips that connect into dedicated sockets on the mother board are referred to as:

(a) SIMMs **(b) SIMs**

(c) PROM **(d) All of these**

2. Read-only memory (ROM) is divided into two types: factory-programmed and user-programmed:

(a) True **(b) False**

3. Which programs are commonly used to store non-changing programs and data?

(a) RAMs **(b) ROMs**

4. Data is burned onto PROMs by the maker of the electronic device in which they are utilized.

(a) True **(b) False**

5. A megabyte (MB) is the same as 1,048,576 bytes (220).

(a) True **(b) False**

4.7. SHORT QUESTIONS

1. Describe the memory unit.

2. Define memory and the various forms of memory.

3. What exactly is ROM? Use a relevant figure to explain.

4. What exactly is an EPROM? Use appropriate examples to explain.

5. What is the difference between RAM and ROM?

4.8. FURTHER READINGS

Fundamental Computer Skills, Feng-Qi Lai, David R. Hofmeister.

Study of Computer Fundamental, Shu-Jung Yi.

http://books.google.co.in/bkshp?hl=en

UNIT 5: SECONDARY STORAGE DEVICES

CONTENTS

OBJECTIVES:

- Learn about Secondary storages
- Learn about HDD, Access Time

- Learn about Input & Output Devices and their latest devices in Market

5.1. INTRODUCTION

Random Access Memory (RAM) is the most often used primary storage device.

5.2. SECONDARY STORAGE DEVICES

Secondary storage devices, as its name suggests, save data after it has already been saved by the primary storage device (Random Access Memory). Your work is saved to RAM as soon as you begin typing a letter in Microsoft Word until you press the "Save" button. This continues until you exit Microsoft Word. All of your work will be lost if you shut down the computer, and the only copy will be saved on a secondary storage device. An optical drive for CDs or DVDs, an internal hard drive, or a USB flash drive are all examples of this.

The primary storage of a computer system is subject to the restrictions listed below:

- **LIMITED CAPACITY**

A computer's ability to store many millions, billions, or even trillions of bytes of data is frequently required. A large majority of data centres are unable to store the enormous amounts of data they handle because the primary storage capacity of today's computers simply cannot keep up with the demand.

- **VOLATILE**

Primary storage is volatile, meaning that when the power is cut off or interrupted, the data stored there is lost. On the other hand, computer systems are required to keep data for an undetermined amount of time, which could be several days, months, or even years.

As a consequence of this, the majority of computer systems call for additional memory, which is also sometimes referred to as auxiliary memory or secondary storage.

Secondary storage is not susceptible to data loss and has a cheaper cost per bit kept when compared to primary storage, despite the fact that it functions at a significantly slower speed. In most cases, it is employed for the purpose of permanently storing vast quantities of data, which can afterwards be partially moved to primary storage so that it can be processed whenever this step becomes necessary.

- ## SEQUENTIAL AND DIRECT-ACCESS DEVICES

The information that is kept in a secondary storage device can be accessed in a variety of ways, which is the primary factor that determines which device is used for a given application. A secondary storage device could be any one of a number of different devices. Sequential access (also known as serial access) and random access (also known as random access) are two ways to get at data. The arrival to the requested site in a sequential-access storage device is preceded by sequencing via prior locations; as a result, the amount of time required to access the device varies depending on where the data is being retrieved from.

In other words, information placed on a sequential-access device can only be accessed in the same order in which it was initially stored. Chores like the preparation of monthly pay stubs and monthly utility bills, as well as other similar tasks, are good candidates for sequential processing. places where the vast majority of the data records, if not all of them, have to be handled individually. However, while working with a sequential-access device, if an address is needed out of sequence, the only way to find it is to search through all of the previously stored addresses. This is the only way it is possible to find the address. This is very much like a cassette tape that contains music.

There are secondary storage devices that enable an access that is either more direct or more swift to the specific information being stored.

These direct-access devices are also referred to as random-access devices due to the fact that any piece of information that has been randomly selected can be retrieved fast from the entirety of the information that has been stored. With these factors in mind, the term "random-access" refers to the ability to access data at any location within a storage device at any time, as well as how long it takes to go to any given location. This can be compared to a music compact disc. As long as the CD includes ten tracks, and the sixth song is your favourite, you don't need to listen (or access) or fast-forward through the five songs that precede it. You can simply access the sixth song directly. Simply selecting track number six on the player will cause it to play automatically after moving the pickup arm to the groove where the sixth song begins. If you have more than one song loaded, the player will play the first one.

Memory storage devices, magnetic discs, and optical discs are all types of media that fall under the category of direct-access storage devices.

5.3. HARD DISKS

Most computer systems today use hard discs as their primary on-line secondary storage device. They're built of inflexible metal platters (usually aluminium) and come in a variety of sizes ranging from 1 to 14 inches in diameter.

- **TYPES OF HARD DISKS**

Hard discs are usually classified into three types based on how they are packaged:

A. ZIP/BERNOULLI DISK.

For added security, it's enclosed in a plastic cartridge, which protects the hard disc platter from damage. The disc has a storage capacity of approximately 100 megabytes and typically measures 3H inches in diameter.

The storage capacity of a computer system might shift subtly in response to the particular method of file organisation that the computer system makes use of. It is possible to either move or install the disc drive, which is referred to as a zip drive. A component of a computer system that is inextricably bound to the system at all times is referred to as the fixed type. The kind that may be carried in and connected to a computer for a brief length of time before being detached and withdrawn might be referred to as the portable variety. Similar to how a video cassette or floppy disc may be inserted and withdrawn from a VCR, a zip disc can be inserted into and removed from a zip drive in the same manner.

B. DISK PACK.

It is composed of many hard disc platters, with at least two of them arrayed in a circular pattern around a single central shaft. Each of the discs rotates at the same time with the others. Read/Write heads for each suitable disc surface can be found in a separate place on its disc drive (recall that when multiple discs are used together in a disc device, the upper surface of the topmost disc and the lower surface of the bottommost disc are sometimes not used). Because the disc drive may be replaced, it is possible to load and unload multiple disc packs according to the requirements of the situation. When it is not being used, a disc pack is stored in a plastic casing that is kept offline. Today's disc packs come with an almost endless amount of storage space.

C. WINCHESTER DISK

The Winchester disc is constructed using multiple hard disc platters, with at least two of them arranged in a circular pattern around a single central shaft. There are two types of drives: a disc pack drive and a Winchester disc drive. [Needs citation] The hard disc platters and disc drive are hermetically sealed together in a container that is free of contamination, so they cannot be separated from one another. As a direct consequence of this, Winchester CDs have a restricted storage capacity. On the other hand, considering the same number of disc platters of the same size. It is possible for Winchester discs to have a greater storage capacity than disc packs due to the following factors: number of disc platters with dimensions that are similar to one another. It is possible for Winchester discs to have a greater storage capacity than disc packs due to the various factors.

D. FLASH DRIVE (*PEN DRIVE*)

Smaller than a pen, flash drives come in a wide range of colours and patterns to suit any style (such as pen shape, wallet shape, and so on). Additional features (such as a camera or built-in N1P3lWMA/FM Radio playback for listening to music on the go) are possible. Flash drives are used to store and transfer digital data. It makes it possible for data to be transmitted quickly and easily from one computer to another. The device connects to the Universal Serial Bus (USB) port of a computer and can be used immediately after being plugged in. The computer is able to determine that it is a removable drive without any additional input from the user. Flash drives can now be used to read, write, copy and delete files on a computer's hard disc. They can also be used to transfer files between a computer and a flash memory device. It is also capable of running applications, watching movies, and playing MP3 files. Once it is finished, you can easily remove it from the USB connection on the computer and put it in your pocket so that you can carry it with you wherever you go. In order to use a flash drive, you do not require a battery cord or software, and it is compatible with the vast majority of PCs, desktops, and laptops (laptops). People who are constantly on the go and need to move data from one computer to another will find it a great choice for use as an external data storage device because of all of these qualities. The name "flush memory storage" references to the fact that it makes use of that technology. EEPROM (Electrically Erasable Programmable Read Only Memory) is a non-volatile EEPROM chip that is stored on the device. It is a high-capacity store in the Aid state that has the potential to retain data for more than ten years.

There are a variety of storage capacities available, including 8MB, 16MB, 64MB, 128MB, 256MB, 512MB, 1GB, 2GB, 4GB, and 8GB. A floppy disc that has a capacity of 16 megabytes has 5600 times the storage capacity of a floppy disc that has an IAMB capacity.

E. OPTICAL DISC DRIVE

An optical disc is inserted into an optical disc drive for reading and writing data. This device holds an optical disc and performs read/write operations on it using a combination of mechanical, electrical, and electronic components. The disc tray, read/write laser system, and disc motor are all housed within.

5.4. ACCESS TIME

1. The following factors contribute to the slower access time and overall performance of optical disc drives compared to magnetic disc drives:

2. The optical disc's separate sectors are arranged on a spiral track. Because sectors are always found on a certain track at a predetermined distance from the centre, this data structure results in slower random access to a sector than the concentric tracks organisation (used by magnetic discs).

3. On an optical disc, the length of each individual sector stays constant regardless of how far it is from the disc's centre. This data structure requires a more complex drive mechanism in order to accommodate the requirement that the rotation speed of the disc must change in the opposite direction of the radius. When reading and writing data towards its edges, it is necessary to slow down the disk's rotation speed, whereas when reading and writing data near the disk's centre, a faster rotation speed is required. Unlike magnetic discs, where the discs rotate at a constant speed regardless of the location of the data that has to be retrieved, this access approach results in slower data access when compared to this method (longer access time).

4. Due to the fact that an optical disc is a removable media, it is prone to damage during use in the form of scratches, dust, sticky prints (including fingerprints), and other sorts of damage. As a consequence of this, the read mechanism employs several error correcting processes such as re-reading, modifying the angular velocity, data reconstruction utilising parity, and so on.

5. The read/write and disc components taken together are not hermetically sealed. Because of this, the disc drive is unable to spin the disc at a high pace because it could potentially cause damage to the disc as well as to the other components. Access times for optical discs typically fall in between 100 and 300 milliseconds. When compared to the access timings of a floppy disc, which range from 100 to 200 milliseconds, and the access times of a hard disc, which range from 10 to 30 milliseconds, this one is quite slow.

5.5. TYPES OF OPTICAL DISKS

Optical discs are platters that are completely circular. They come in a wide range of shapes, sizes, and carrying capacity to choose from. The most prevalent kinds of optical discs are CD-ROM, WORM (CD-R), CD-RW, and DVDs. Optical discs can also be written to. The paragraphs that follow include information pertaining to them.

A. CD-ROM

CD-ROM is an abbreviation that stands for "compact disc read-only memory," which is what the term actually refers to. The technology behind it is derived from audio CDs, and it performs in music players in a manner analogous to that of audio CDs. It is possible to use your personal computer to play music CDs if it is equipped with both a sound card and speakers.

A CD-ROM disc is a shiny disc made of metal that is silver in colour and has a diameter of 5/4 inches (12 cm). Polycarbonate plastic is used in its construction, and a very thin film of pure aluminium is layered on the top so that it is reflective. Occasionally, a gold covering is applied to extremely high-quality discs. An extremely thin layer of lacquer protects and preserves it. In previous models, its storage capacity is approximately 650 Megabytes, whereas more recent models have a storage capacity of 700 Megabytes. The fact that it is a read-only storage medium and has a massive capacity packed into a very small disc is where the term "read-only" comes from. This indicates that the data contained on these CDs has been pre-recorded and cannot be altered in any way.

B. WORM DISK/CD-RECORDABLE (CD-R) DISKS

The phrase "write-once, read-many" is often abbreviated as the abbreviation "write-once, read-many." Users are able to produce their own CD-ROM discs using WORM discs by attaching a CD-R drive to a computer in the same way that they would connect any other standard

peripheral device. The data on WORM discs is written using a CD-R drive after the blank discs have been purchased. They are similar in appearance to regular CD-ROM discs. It is possible for any standard CD-ROM drive to read the information that was written to a WORM disc by a CD-R drive. There is no way to change the data on a WORM disc once it has already been written to, yet it can be read multiple times. The data on WORM discs cannot be reversed, unlike CD-ROM data, and can only be read after it has been written on the disc. WORM discs, on the other hand, allow for several recording sessions to be used to capture all of the data that needs to be saved on the disc. Following the initial session, following sessions are always additive, and the information that was etched or burned during earlier sessions cannot be altered in any manner. It is possible to conceal the information entered in one session in another by establishing a new File Allocation Table (FAT), but the etchings that are already on the surface cannot be removed. This type of disc is referred to as a Multi-Session Disk (MSD for short). Laser beam technology is utilised for both the recording and reading of data (described earlier).

C. CD READ/WRITE (CD-RW) DISK

The only difference between a WORM disc and a CD Read/Write (CD-RW) disc is that the latter can be overwritten and deleted multiple times while the former cannot. These discs have a layer of a metallic alloy applied to them. During the writing process (also known as the burning process), the laser beam alters the chemical property, which in turn changes the reflectivity in the right spots. CD drives need to be compatible with one another in order to read CD-RW discs. This is due to the fact that CD-land-pit RW's differential is insignificant. At the very least, CD-RW discs can withstand 100 read-erase cycles before needing to be replaced. A disc that has previously been written on can have the previous writing removed by altering a chemical characteristic, and the disc can then be written on again.

D. DIGITAL VIDEO (OR VERSATILE) DISK (DVD)

The DVD format was developed primarily for the purpose of storing and delivering motion pictures. On the other hand, as prices continue to drop and the demand for storage with enormous capacities develops, it is rapidly becoming an optical disc standard. It functions exactly like a CD-ROM, but it stores data in a much more compact format. An eight-to-fourteen modulation (plus one percent) encoding is utilised instead of CD-eight-to-twelve ROM's modulation (EFM). Single-layer discs and double-layer discs are the two varieties of DVDs. A disc with a single layer has a storage capacity of 4.7 gigabytes, while a disc with two layers has

a storage capacity of 8.5 gigabytes.

DVDs have enough space to store an entire film, and they can also accommodate several audio tracks and subtitle languages, in addition to a variety of camera angles and other features. Additionally, it is capable of supporting region tagging, which helps prevent piracy as well as the transfer of DVDs between different regions.

Additionally, it supports the CPPM (Content Protection for Pre-recorded Media) security mechanism, which safeguards against n; subsequently, it safeguards against etc.

5.6. ADVANTAGES AND LIMITATIONS OF OPTICAL DISKS

- **ADVANTAGES**

1) Because of their very cheap upfront cost and relatively high storage density, optical discs have an extremely low cost-per-bit of storage.
   The fact that some optical discs may be overwritten and used several times without losing their data contributes to the cost reduction.
2) Due to the fact that optical discs only contain a single helical track, they are excellent for reading large blocks of sequential data, such as music or video.
3) In optical disc drives, the read/write heads are not mechanical, thus they do not rub against or otherwise damage the disc surface.
   Consequently, optical discs are a more trustworthy medium for data storage than magnetic tapes or discs.
4) More than thirty years' worth of data can be stored on optical discs. As a result of this, they are an improved storage medium for archiving data as compared to magnetic tapes or magnetic discs.
5) Because the data that is recorded on CD-ROM and WROM discs is permanent, there is no chance that the data that has been stored will be erased or overwritten by accident.
6) Because of their diminutive size and low mass, optical discs are very easy to handle, store, and transport.
7) If your computer has a CD-ROM drive, speakers, and a sound card, you can listen to audio CDs on it.
   As a consequence of this, computer systems may be utilised at any given time in the capacity of music systems.

- **LIMITATIONS**

1) Read-only storage medium, often known as permanent storage media, include WORM discs and CD-ROMs. After it has been captured, the data cannot be removed in any way. Because of this, it is impossible to recycle or reuse them.
2) The data access speed of optical discs is significantly lower than that of magnetic drives.
3) Optical discs require a more complex drive mechanism than magnetic discs do due to the requirement for a laser producing source and detecting lens. Because of the complexity of these two components and the time and effort they demand, optical discs necessitate a more complex drive mechanism.
4) Because an optical disc is a removable medium, it is subject to damage during usage from a variety of sources, including scratches, dust, sticky prints (including fingerprints), and other sorts of harm. Because of this, they need to be handled with extreme care.
5) If they are going to be used for off-line storage, they need to have adequate labels so that they may be easily identified.

5.7. INPUT\OUTPUT DEVICES

In order to be of any use, a computer system needs to be able to connect with its environment (its users). This functionality is provided by the input-output devices, also abbreviated as UO devices, of a computer system. The components of a computer that are located outside of its central processing unit and memory are known as its peripheral devices.

There are input devices that are now being used. The data that comes in from the outside world is put into primary storage, and the output devices are the ones that give the results of the processing done in primary storage to the users. There is currently a very wide variety of input and output devices on the market. It's possible that one kind is better suited for a certain use than another. There are devices that can function in both the input and output roles. It is essential to keep in mind that primary storage and the CPU are far faster than I/O devices, which are notoriously slow. That's because mechanical components often determine a vehicle's top speed, and there's not much room to increase that speed. As a result, they are restricted in their forward motion.

To stay up with CPU and memory speeds, it is difficult to develop I/O devices that can keep

up, and there is a constant demand for better and faster I/O devices.

- **INPUT DEVICES**

A device that accepts data from the outside world and changes it into information that a computer can understand is called an input device. This device is an electromechanical device. There are many different types of input devices on the market today. They are capable of being classified into the following groups:

•*Devices with keyboards*

•*Devices with a point-and-draw interface*

•*Scanning devices for data*

• *Electronic cards-based devices*

• *Digitizer*

•*Speech-to-text software*

•*Vision-based gadgets*

Below are descriptions of many types of input devices and their applications.

- **KEYBOARD DEVICES**

Keyboards are the input devices that are utilized the most frequently in the modern world. They make it possible for users to enter data into a computer system by pressing a set of keys (buttons with labels) that are neatly installed on a computer keyboard. The 101-key QWERTY keyboard is currently the most popular and widely used type of keyboard.

- **POINT AND DRAW DEVICES**

In the beginning, the primary mode of interface with computers was done through the use of text. But it quickly became apparent that working with computers in text mode is both time-consuming and cumbersome.

As a direct consequence of this, a novel kind of user interface known as graphical user interface (GUI) was developed for the purpose of connecting with computers. A graphical user interface,

also known as a GUI, is a type of computer interface that displays a screen with icons (small images on the screen) or menus and enables a user to make speedy selections from those icons or menus in order to communicate their instructions to the computer. In order to get the most of the OUI, you will need an input device that allows you to swiftly point to and choose a graphic icon or menu item from among the many alternatives that are displayed on the screen. It was concluded that the keyboard, despite having the capability to perform its intended job, was cumbersome and insufficient for this endeavor. The hunt for an input device that matched this criterion led to the development of numerous input devices, including the mouse, track ball, joystick, light pen, and touch screen. The track ball is another option for users. To create visual elements on a screen, such as lines and curves and freehand shapes, many of these gadgets, such as a mouse and a light pen were very effective. An example is a freehand shape made up of straight lines, curvy lines, and other shapes. Point-and-and-draw devices got their name from the increased functionality they offered. Computers are now more accessible to a wider range of people, including youngsters, the illiterate, and graphic designers, thanks to the widespread adoption of these input devices. As a result of this, computers have become a much more accessible tool. The following are some of the point-and-draw gadgets that are utilized most frequently.

- **MOUSE**

The mouse is the most frequent device used for pointing and drawing. Most modern personal computers and workstations use GUIs, making this device a necessity for data entry and data input. In the palm of the user's hand, a mouse is a little hand-held gadget. It's a computer command and control system. There may or may not be buttons on the top of it, depending on how much space is available on the bearing. User terminals display a graphical pointer when a mouse linked to a user terminal rolls across a flat surface. To access a menu item or an icon, simply move the mouse to where the graphics cursor appears on the screen. This allows you to more easily navigate the on-screen interface.

The graphical cursor can take the appearance of a variety of different symbols, including an arrow, a wrist, a pointing finger, and others. The text cursor and the graphics cursor may appear on the screen at the same time, depending on the programme. In order to determine where the graphics cursor should be placed on the screen, the graphics cursor has a point on it the size of a pixel. It is claimed that the graphics cursor is pointing to a particular menu item or icon when the hot-spot of the graphics cursor is positioned at the item or icon in question. to the item on

the menu or icon that you are searching for Simply pressing the button on your mouse will send an alert to the operating system about this choice. It is important to note that making use of a mouse, as opposed to employing a number of different key combinations, is a simpler method for informing the computer of a specific selection from the variety of options provided by the software. When combined with the appropriate program, a mouse can be used to both draw on the screen and edit text.

- **DATA SCANNING DEVICES**

Users are able to immediately enter data from source documents into a computer system through the use of data scanning devices, which are input devices. In addition to that, some of them are able to identify particular marks or characters. The following is a list of the characteristics of various different types of devices:

They lessen the amount of work that needs to be done by hand by humans. The utilisation of robotic data input improves the correctness of the data as well as the timeliness of the information that is processed.

Due to the fact that the data is entered directly from the source documents, they require input papers of a high quality. Documents that contain typographical errors, strikethroughs, or erasures are routinely turned down.

Because of this, form design and ink specification become more important when these devices are used to input data from forms. There is a wide variety of variety when it comes to the types of data scanning equipment. The most prevalent ones are described in the following list.

- **IMAGE SCANNER**

An image scanner is a piece of hardware that can be plugged into a computer to read physical documents and turn them into digital copies that may be saved.

Input documents can be made up of a variety of different types of data, including handwritten content, typed text, pictures, and graphics.

This input device has shown to be quite effective in the preservation of paper records in an electronic version. As time passes, the quality of the document will not degrade or get yellowed, and it can be viewed or printed at any point in the future. If the computer has image-processing

software installed, the saved pictures can be altered and manipulated in a variety of intriguing ways.

There are two primary kinds of scanners that are typically utilized:

- **FLATBED SCANNER**

There are many similarities between a copier and a flatbed scanner. A paper that is going to be scanned is inverted before being placed on the glass plate. When the switch is turned on, a light source that is located beneath the glass panel will move laterally from one end to the other. After finishing the scan of one line, the light beam moves slightly upward before beginning the scan of the next line. Every single line is worked through the method in its entirety. A paper measuring 21 centimetres by 28 centimetres can be scanned in around twenty seconds.

- **HAND-HELD SCANNER**

: There are several light emitting diodes in a compact shell that make up a hand-held scanner, which can be easily held in one hand. A piece of paper is scanned by slowly dragging a scanner over it from one end to the other while keeping the light on throughout the process. If the paper is not moved slowly and carefully across the scanner's glass, the bit map equivalent of the document will not be transformed appropriately. As a direct consequence of this, hand-held scanners are only used in circumstances in which an exceptionally high level of accuracy is not necessary. When there aren't a lot of documents to scan, you might still benefit from using them. In comparison to flatbed scanners, their prices are substantially more reasonable.

5.8. OPTICAL CHARACTER RECOGNITION DEVICE

Image scanners have two constraints when being used to input typed or handwritten text documents. These limits are as follows:

Due to the fact that a scanned document is kept on the computer as an image rather than as text, word processing cannot be performed on it (the data in the paper is unintelligible to the computer since it is not represented by alphabetic or numeric codes).

When compared to the amount of storage space required to store the same content in text format, the amount of storage space required to store a document as an image is substantially larger. A page of printed text that has two thousand characters can be saved as two thousand

bytes if the ASCII encoding is used. Data requirements for a bitmap representation of the same document can range from 10 to 15 times greater than those required for an electronic document.

OCR technology is employed in order to get around these drawbacks. Character recognition software, also known as optical character recognition software, is built into the scanner in this instance. ASCII values are generated from bitmap images of characters. So, the scanner takes a picture of the page and then uses OCR software to turn that picture into ASCII text that can be recognised by a computer. In other words, the scanner creates an image of the document in two steps.

OCR software is notoriously difficult to use since it is mathematically impossible to instruct a computer to distinguish between an endless number of types and fonts. They are therefore built to understand text written in traditional typefaces, which is why (called OCR fonts). These two fonts are known as OCR-A and OCR-B, respectively, for their use in American and European documents (European standard). You can't use an OCR programme if your document has italics or boldface letters, or if it's written in a typeface that's not one of those supported by your software.

- **MAGNETIC LINK CHARACTER RECOGNITION (MICR)**

One sort of optical character recognition is known as MICR (OCR). In order to speed up the processing of the large number of checks that are handled on a daily basis, the banking industry makes use of it. When a bank cheque is pre-printed (encoded), characters from a certain character set are used to include the bank's identification code (name, branch, etc.), account number, and check number. This is done using the cheque's cheque number.

The characters on the check are pre-printed using an unique ink that contains magnetizable iron oxide particles. This allows the characters to be read magnetically.

When a consumer brings a filled-in cheque to the bank, an employee of the bank must manually enter (key in) the information the customer has provided, including the amount that has been written on the check. After that, a magnetic ink character reader-sorter (also known as a MICR reader-sorter) is utilised in order to process the check.

The magnetic ink character recognition (MICR) reader-sorter reads the data that has been pre-printed on the checks and sorts them so that they can be sent to other banks or processed further.

The E13B typeface, which includes the digits 0 to 9 as well as four other characters, is the character set that is utilised by MICR machines the majority of the time.

Cheques are fed into a MICR reader-sorter, which then sends the encoded information in the form of these typefaces to a computer. When the cheques are fed into the reader, they pass through a magnetic field, which magnetizes the ink particles on the cheques as they travel through the field. After you've read the heads, you can decipher the characters by looking at the contours of their bodies. The cheques are separated into several pockets by the sorter on the basis of the identification code numbers printed on each one.

- **DIGITIZER**

A computer input device known as a digitizer translates analogue data such as photographs, maps, and drawings into digital format so that the information can be saved. Things like the coordinates of points in a drawing can be recorded digitally, for example. This makes it possible to easily recreate the drawing using the previously recorded data and to easily incorporate changes into the design whenever they are required.

The digitising tablet, sometimes referred to as a graphics tablet, and the stylus are the two components that make up a digitizer. The digitising tablet is a flat surface that has a grid etched into it. The grid is made up of hundreds of thin copper wires. Over the course of each copper wire, electric pulses are transmitted. The digitising tablet is extendable across a working surface and is linked to a computer for use. As an optical cursor with a button and cross hair, you can write with the stylus.

A button on the device allows users to enter the tablet's coordinates (x, y) for a specified location. The operator receives visual feedback in the form of the movement of a cursor on the computer screen in response to movements of the stylus on the tablet itself.

Because of this, the operator can swiftly draw sketches or enter sketches into the computer. The digitizer automatically converts illegible lines, arcs, and other graphical elements into mathematically exact objects, such as straight lines and smooth curves, when creating or developing a sketch. This occurs whether the digitizer is being used to create drawings or input drawings.

Digitizers are tools that are used by engineers and architects in the process of creating products using computer-aided design (CAD). Automobiles, structures, medical gadgets, robots, and

mechanical parts are all included in this category (CAD). Maps typically produced on paper can also be digitised with the use of these devices in Geographical Information Systems (GIS).

5.9. OUTPUT DEVICES

An output device is a device that receives data from a computer and converts it into a format that can be used by the rest of the world. This device is an electromechanical one (users). There are many different types of output devices that can be used today. They can be divided into the following categories:

- *Monitors*
- *Printers*
- *Plotters*
- *Image projector on a screen*
- *Voice response system*

The next page describes several types of output devices as well as their uses.

Computer output is generated through output devices, which can be divided into two categories.

- **SOFT COPY OUTPUT**

Soft copies are not printed on paper or other material that can be held in the hand and moved around to show other people, but rather are stored electronically and may be accessed at any time.

They will only be there for a brief period of time. S011-copy output includes both the text displayed on a terminal's screen and the audio prompts that users hear.

- **HARD COPY OUTPUT**

A "hard copy" is a version of an output that can be exhibited to others since it is made on a physical medium that can be handled and moved.

They are able to be kept in paper files and read at a later time when the person in question is not using a computer.

They are permanent. To give one illustration: The output that is printed or plotted on paper is referred to as hard-copy output.

5.10. MONITORS

Monitors are the output device that is used the most frequently to provide soft copies of output in this day and age. They display the output that was made on a screen that resembles a television. A video display terminal is often formed by the combination of a keyboard and a monitor, both of which are linked to one another (VDT). A video display terminal (VDT) is the input/output (I/O) device that is the most typical in usage with modern computers (commonly referred to simply as terminal).

It can act as both an input and an output device. The monitor is used to display the output of a computer, while the keyboard is used to enter data into the computer. Because it is situated at the point where a communication connection comes to an end, it is referred to as a "terminal," hence the name. Monitors can be broken down into two primary categories: those that use cathode ray tubes (also known as CRTs) and those that use liquid crystal displays (also known as LCDs). flat-panel. CRT displays, which are utilised with desktop or other non-portable computer systems, operate in a manner analogous to that of a television screen. LCD flat-panel displays, on the other hand, are both lighter and thinner than CRT monitors. and are most frequently found in mobile computer devices like laptops.

As the price of LCD flat panel monitors continues to drop, users are increasingly turning to this type of display for their stationary desktop computers. They are also preferred because they take up less room on the table, which is another reason for their popularity.

- **LCD MONITOR**

LCD, which is an abbreviation for "liquid crystal display," is the name of the technology that enables flat-panel monitors to display their images. While previous CRT monitors may weigh up to 23 kilogrammes, LCD screens normally weigh less than 4 kilogrammes. This contrasts with the huge footprints and depths of CRT monitors, which could weigh between 13 and 23 kilogrammes.

A lack of modern technology meant that LCD screens were first used in portable computing devices like laptop computers, rather than desktop monitors. In an LCD display, a backlight,

polarised glass, a "mask" of coloured pixels, a layer of liquid crystal solution responsive to a wired grid of (x,y) coordinates, and an additional polarizer sheet of glass are all necessary components to create the image. Additionally, an LCD display has a layer of liquid crystal solution that is responsive to a wired grid of x, y coordinates. The crystals function as miniature shutters, opening and closing in response to the stimulus. Using accurate electrical charges of changing degrees and voltages, precise crystal orientations can be manipulated. This opens the door for varying degrees of light to illuminate the screen, which ultimately results in the formation of a picture.

5.11. PRINTERS

Printers are the output devices that are used the most frequently today for the production of hard copies of output. The following sections will provide an overview of the many categories of printers.

- **DOT-MATRIX PRINTERS**

Dot-matrix printers are a type of character printer that produce one character at a time and are also known as character printers. They are used as dot patterns to produce a wide variety of characters and images. Dot patterns allow for the creation of a variety of different forms of characters, as demonstrated in Figure 4.16. The print head of a dot matrix printer moves horizontally across the paper during printing (from left to right and right to left). Print is composed of a number of pins that may be operated independently of one another to stretch and strike against an inked surface, which results in dot patterns being printed on the paper. These patterns are called dots. When the print head moves horizontally, the printer selects the appropriate pins to print a character and starts the printing process. In order to print documents more quickly, many dot-matrix printers print in both directions simultaneously. On the first pass, they print from left to right, and on the second pass, they print from right to left. One example of such a technology is bidirectional printing.

Because the output of dot matrix printers is comprised of patterns of dots, these printers are capable of printing any shape of character that can be specified by a gramme.

As a consequence of this, they are able to print a diverse selection of special characters, a variety of print sizes, and graphics like graphs and charts.

Impact printers are those that generate ink impressions on paper by smashing pins on an already inked ribbon. Dot-matrix printers are examples of impact printers. As a consequence of this, it is possible to manufacture multiple copies of them by using carbon paper or another substance with the same properties. Because they use impact printing, dot-matrix printers produce more noise than printers that don't use impact.

The number of characters that may be printed by a dot-matrix printer in one second can range anywhere from 30 to 600. They are, however, inexpensive, both initially and in terms of the recurring costs that they entail. As a consequence of this, individuals and businesses favour them for the production of printed outputs, provided that the speed and quality of the printing are not significant problems. Shipping forms and invoices are just two examples of multi-copy output applications that rely on the print impact of the printer for making several copies of the document.

- **INKJET PRINTERS**

Character printers are known as inkjet printers, and they work by spraying small drops of ink onto paper in order to create a variety of different pictures and characters. The print head of an inkjet printer can have as many as sixty-four discrete nozzles, each of which can be heated by a single integrated circuit resistor in a matter of microseconds. One drop of ink is left on the paper in front of the print head after it vaporises and is ejected from the printer's nozzle. This happens when the ink is heated up near the resistor. When producing a character, the printer moves the print head in a horizontal direction and heats up only the nozzles that belong to the appropriate set.

The output quality of inkjet printers is superior to that of dot-matrix printers because the characters produced by inkjet printers are made up of very small ink dots. The print resolution of a high-resolution inkjet printer can be as high as 360 dots per inch if it has as many as 64 nozzles packed into a height of only 7 millimetres.

Due to the fact that they print as patterns of small dots, inkjet printers are able to reproduce any character shape that can be described by a computer programme. Because of this, the printer is able to produce a diverse assortment of special characters, a varied range of print sizes, and graphics including charts and graphs. Inkjet printers are not considered to be impact printers because the printing process involves spraying ink onto the paper. As a direct consequence of this, they make less noise when in operation. Non-impact printers cannot be used to print

several copies of a document at the same time since they are non-impact printers. Ink cartridges for colour inkjet printers are typically provided with black and three-color inks. Colors red, blue, and yellow are all contained on a single cartridge. These colours, along with the black from another cartridge, can be combined to create any colour with the specified saturation level. It is because of this that inkjet printers are able to create output that is both multicoloured and of photo-quality, despite the fact that their printing speeds range anywhere from 40 to 300 characters per second, which is far slower than those of dot-matrix printers. The price of an inkjet printer is often higher than the price of a dot matrix printer. When the speed of printing is not a concern, it is recommended to use them.

- **DRUM PRINTERS**

Drum printers are capable of producing a whole line of text all at once. Printers that print one character at a time include dot matrix and inkjet printers. Let's take an example of a drum printer. It is constructed out of a solid cylindrical drum, and on its surface are characters that are embossed, or elevated, in the shape of circular bands. The character set of the printer is divided into bands, and each band comprises all of the printable characters that the printer can produce. The maximum amount of text that may be printed on a single line is proportional to the total number of bands, which is also known as print positions. To put it another way, the total number of embossed characters on the drum printer's surface with 132 characters per line and 96 characters in the character set is 12,672 (132 times 96).

In addition, the printer contains a set of hammers that are situated in front of the drum. An inked ribbon and paper can be put between the hammers and the drum using this particular arrangement of hammers. When you add up all the print locations on a single line, you get to the same number of hammers and bands that are on the drum.

The drum rotates at a breakneck pace. As soon as the band's embossed logo passes below the intended print place, the corresponding hammer is engaged and a character is printed there. As a result, the drum would have to complete one full round for each output line. So while not all characters on a line are printed simultaneously, it appears as if only one line is printed at a time because of how quickly one whole line is printed.

Changing the drum in a drum printer on a regular basis is time consuming and expensive. The embossed characters on the drum limit the printing capabilities of drum printers to a limited set

of characters and formatting. The drum printer is unable to produce any specific type of character, variable print sizes or graphics like charts and graphs as a result of this.

Drum printers are a type of impact printer since the printing process involves striking a sheet of paper that has been printed on with ink against the raised characters that are embossed on the drum. As a consequence of this, it is possible to make numerous copies of them by use carbon paper or an equivalent. Drum printers generate a lot of noise during operation due to the impact printing process, which is why they typically have a cover. Drum printers have speeds that range anywhere from 300 to 2000 lines per minute for printing.

- **CHAIN/BAND PRINTERS**

Chain or band printers are a type of line printer that produce one line at a time and are also known as band printers. Figure 4.20 is an illustration of the print mechanism used in a chain or band printer. It's a metal band or chain with etched characters from the printer's supported character set. Its construction is described above. The number of characters contained in a standard character set might range anywhere from 48 to 96. On the chain or band, the characters that make up the character set are embossed many times so that the printing quality can be improved. There may be four sets of 64 characters on the chain or band of a printer with a 64-character set. In this case, the band or chain will have 256 characters engraved on it (64 times 4). The printer has a series of hammers that are situated in front of the chain or band, and between those hammers and the paper is an inked ribbon that is sandwiched. This machine's print places are represented by the number of hammers. Because of this, a printer that has 132 print positions will require 132 hammers to function properly. A high rate of rotation is maintained by the chain or band. When the character that is embossed on the chain or band passes below it, the appropriate hammer is engaged and a character is created at the designated print location. Before placing the appropriate character in the correct print location, the chain or band does not have to complete a full revolution. On chains and bracelets, the same type of character set is used multiple times.

In contrast to drum printers, chain and band printers include easily removable and replacement chain and band components. As a result, a single printer may support a wide range of typefaces (character styles) as well as scripts (languages). The characters embossed on the chain/band that is utilised with the chain/band printer are confined to printing only pre-defined sets of characters, like drum printers. Band printers are unable to print any shape of character, multiple

sizes of print, or visuals such as charts and graphs. Chain/as a result of this, band printers cannot print.

The characters are embossed on the chain or band of an impact printer, and the printing is done by striking a paper and inked ribbon against those characters. As a consequence of this, it is possible to make numerous copies of them by use carbon paper or an equivalent. Because impact printing is a noisy process, the operation of chain and band printers typically requires the use of a cover to muffle the sound. Printing speeds for chain and band printers range anywhere from 400 to 3000 lines per minute.

- **LASER PRINTERS**

Printers that are designed specifically for pages, such as laser printers, only print one page at a time. The laser beam source, the multi-sided mirror, the photoconductive drum, and the toner are the primary elements that make up a laser printer (tiny particles of oppositely charged ink). When printing a page of output, the spinning multi-sided mirror is responsible for focusing the laser beam on the electrostatically charged drum.

After being guided onto drum surface by mirror, laser beam creates patterns of characters and graphics that will be printed onto page by laser printer.

When a laser beam is focused on a piece of the drum's surface, the drum becomes photoconductive, resulting in a difference in the drum's electric charge. Laser beams can charge a drum's surface, causing toner, which is formed of individual ink particles with opposing charges, to be drawn to the drum. Finally, the toner is firmly affixed to the paper. The paper is heated and then pressed to create printed output. To remove the toner that has adhered to the drum's surface, it is spun and then cleaned with a rubber blade. The next page will now be available for printing.

Because they use very little ink particles to generate characters, laser printers are able to produce output of an exceptionally high quality. The resolution of the majority of standard laser printers is measured in dots per inch (dpi), while the resolution of certain high-end laser printers is measured in dots per inch (1200). As a result of their high resolution, these printers produce graphical art of an exceptionally high quality.

Since laser printers provide printed output in the form of patterns created by laser beams, these printers are capable of printing any character shape that can be described by a computer

programme. Non-impact printers, often known as laser printers, have the ability to print a wide variety of special characters, as well as a range of different font sizes and images, such as charts and graphs. As a direct consequence of this, they operate with an exceptionally low level of noise. Due to the fact that they are non-impact, a single printing of a document cannot make numerous copies of itself using these printers.

Even though the vast majority of laser printers can only print in black since their toner and drum units are combined into a single unit, colour laser printers are now readily available for a price that is more affordable. A colour laser printer, similar to a colour inkjet printer, possesses four distinct colour toner cartridges, each of which has its own drum.

Inkjet printing still makes use of drums and toners, but the drums are now charged in such a way that ink and toners work together to produce the desired colour.

Laser printers are far quicker than the other types of printers that were just described. Printing at a pace of four pages per minute is possible with low-speed laser printers.

In addition to that, high-speed laser printers are at your disposal. Because of their quicker print speeds and higher print quality, laser printers are typically more expensive than other types of printers.

- **PLOTTERS**

Dot matrix, inkjet, and laser printers have all been shown to be capable of creating graphics in the past. When it comes to architectural drawings for buildings and aircraft or automobiles and other engineering designs, high-quality graphics are often required in huge sheets on which the proportions are exactly matched.

Examples of such applications include AutoCAD and Adobe Illustrator.

The various kinds of printers that we covered earlier are not suitable for fulfilling the needs of these kinds of applications in terms of output.

This kind of output is typically handled by plotters, which are a specific kind of equipment. A plotter is ideal for those who need to produce high-quality hard-copy graphic output in a wide range of sizes on a regular basis, such as architects, engineers, and city planners.

Plotters come in a wide variety of sizes and can print in a wide range of resolutions. Plotters can be divided into two primary categories: drum plotters and flatbed plotters. These are as follows:

- **DRUM PLOTTERS**

An anti-clockwise or clockwise rotation of the drum propels the design paper in the vertical direction in a drum plotter. Penholders, either one or several, are attached to the mechanism in a direction that is perpendicular to the surface of the drum. The holder(s) of the pen(s) can be moved either left to right or right to left in order to achieve horizontal motion using the pen(s). The movements of the drum and pen(s) are controlled by a graph-plotting programme; more specifically, the drum and pen(s) move simultaneously to draw designs and graphs under the control of the computer; the sheet is placed on the drum. Drawing characters of varying sizes using the pen can be used by the plotter as a method for annotating the drawings and graphs that have been made.

5.12. LATEST INPUT DEVICES IN THE MARKET

1. Keyboard

2. Pointing Instruments

(a) Mouse

I Mechanical mouse

(ii) Optical mouse

(b) Track Ball

(c) Joystick

(d) Light Pen

3. Touch Screen

4. digital camera

5. Scanner

6. Microphone

7. Image Digitizer

8. Scanners for optical data

9. Optical Character Recognition (OCR) (OCR)

10. Detection of Optical Marks (OMR)

11. Bar Code Reader (BCR)

12. Magnetic-Ink Character Recognition (MICR)

5.13. LATEST OUTPUT DEVICES IN THE MARKET

1. Monitor

2. floppy disc

3. Printers

4. Speaker

5. Hard Disk Drives

7. floppy diskette

8. Headphone

9. Projector

5.14. SUMMARY

- Additional data can be stored on a computer's secondary storage, which is often a hard disc.
- On the user's side of the computer, a device known as an input device is one that takes information from the user.
- Displaying results from a computer requires the use of several output devices.

- The digitising tablet, which is often referred to as a graphics tablet, and the stylus are the two components that make up a digitizer.
- A printer is a specific kind of output device that can be found in computers.
- Lager printers are printers that do not use impact and have a silent operation. Due to the fact that they are non-impact, a single printing of a document cannot make numerous copies of itself using these printers.

5.15. KEYWORDS

The term "**terminal**" refers to the combination of a monitor and a keyboard, which together often make up a Video Display Terminal (VDT). Input/output (I/O) devices such as video display terminals (also known as terminals) are the most often utilised devices with modern computers. **Graphics hotspot**: The graphics cursor, despite its size and shape, has a point that is the size of a pixel, and this point is utilized to establish where the cursor is located on the screen. The hotspot of the graphical cursor can be found at this particular location. The storage technology that is commonly referred to as flush memory You should keep in mind that flash is an electrically **erasable programmable read-only memory (EEPROM)** chip.

Plotter: In order to accomplish this, a certain type of output device known as a plotter must be used. A plotter is ideal for those who need to produce high-quality hard-copy graphic output in a wide range of sizes on a regular basis, such as architects, engineers, and city planners. Plotters come in a wide variety of sizes and can print in a wide range of resolutions.

LCD is an abbreviation for "liquid crystal display," which is the name given to the technology that is used in today's most common flat panel monitors. The term "**digitizer**" refers to an input device that "digitizes" analogue data such as photographs, maps, and drawings so that they can be saved in digital format on a computer. The x and y coordinates of points on a drawing, for example, can be saved digitally.

5.16. SELF-ASSESSMENT QUESTIONS

1. RAM chips that are plugged into special sockets on the motherboard are known as.....

(a) SIMMs (b) SINNs

(c) SIMNs (d) None of these

2. The act of retrieving data from a storage location is known as.....

(a) Memory write Operation (b) Memory Read Operation

3. ..... storage system consists of a rotating disc coated with a thin metal layer.

(a) Memory disc (b) Optical disk

(c) Hard disk (d) None of these

4. What does the LED indicator stand for?

(a)Light-emitting diode (LED) (b)level-emitting diode (LED)

(c)low-emitting diode (LED) (d) None of these are true.

5.A..... is a pointing device that looks like a roller-ball mouse.

(a) Trackball (b) All of the above

(c)Electronic pen (d) None of the above

6. A colour inkjet printer has two ink cartridges: black and.....…

(a) Tricolor (b)Blue, green, white (c)None of these

7. A 4 size corresponds to an 8" by 12" page.

(a)True (b) False

8. A CTR stands for?

(a) Crystal ray tube (b) Cathode ray tube (c) Cathode ray tube

9.The patterns of UPC bar codes are decoded as digits.

(a) 5 (b) 10

(b) 12 (d) 2

5.17. REVIEW QUESTIONS

1. What is secondary storage, and how does it differ from primary storage? What distinguishes it from primary storage?

2. What are the key constraints of a computer system's primary storage?
3. What are the advantages and disadvantages of using I/O devices in a computer system?
4. What do you mean when you say "peripheral devices"? What is the meaning of their name?
5. Why are I/O devices so slow in comparison to primary storage and the CPU?
6. What is the definition of an input device? Name a few of the most prevalent input devices.
7. What are the different types of keyboard devices?
8. What are the different types of point-and-draw devices? Name a few popular point-and-draw gadgets.
9. Explain the process of recording data on a magnetic tape.
10. What is the primary purpose of secondary storage in most computer systems?
11. List some of the most common secondary storage devices found in today's computers.
12. Can you explain what a sequential-access storage device is? Make a list of a few applications that would benefit from such a storage device.
13. Can you explain what a random-access storage device is? Make a list of a few applications that would benefit from such a storage device.
14. Identify the differences between a sequential, direct, and random access storage device. Make a single example of each.
15. What variables influence disc storage capacity?

5.18. FURTHER READING

Fundamental Computer Skills, Feng-Qi Lai, David R. Hofmeister.

A Study of Computer Fundamental, Shu-Jung Yi.

http://books.google.co.in/bkshp?hl=en

UNIT 6: INTERNET

CONTENTS

OBJECTIVES:

- Learn about the history of Internet
- Learn about Web Pages, Website
- Learn about URL, HTML
- Learn about ISP

6.1. INTRODUCTION

Internet has become a prominent way of communication in today's information technology environment. It has infiltrated every aspect of existence. Anyone can access information on the internet. With a minimum of financial resources, wherever in the globe. The Internet is a network of millions of computers. Phone lines or other means of communication were used to connect them. Ethernet ISDN or cable modems, for example, are used to share data. The internet is an enormous network.

Because of its size and strength, one of the experts likened to the Internet as a "ocean." Some people are referred to "surf," "ride," or "navigate" the "internet." Most individuals start by ‘swimming' across the Internet until they figure out how to navigate on their own. The options are unlimited when it comes to the Internet. A few of them are listed below.

Schoolchildren in Delhi, for example, can 'speak' to their classmates in Mumbai.

The ability to hold a conference with colleagues from all over the world is available to educators and administrators. With a few mouse clicks, a business traveller can send dozens or even hundreds of mails from a personal computer at home or at work. Students may acquire knowledge on any subject via the Internet at any time. Jobseekers may obtain information on various types of employment from all around the world. Students interested in pursuing higher education can obtain information from a variety of universities located across the world. And this is only the beginning when it comes to the Internet. There is only one limit to how much you can get out of it: your own desire.

6.2. INTERNET-BRIEF HISTORY

In 1969, the United States Department of Defence initiated an experiment that would later become known as the Internet. The US military need an internet-based method for its researchers to communicate and share software. ARPANET, which stood for the Advanced Research Projects Agency Network, was the first computer network to be used over vast distances. It was developed by defence computer scientists. Remote military installations were then 'linked' through telephone lines. Long-distance networking was recognized as a benefit by universities and scientists. They started connecting to ARPANET and to each other. By using the computer network, businesses and individuals began to connect. The Internet, a huge network of networks, eventually emerged.

The Internet is now owned, administered, and maintained by everyone who uses it, not by any single person, business, or government. The United States Department of Defence launched the Internet as an experiment in 1969. For their researchers to communicate and exchange programmes over the internet, the US military is in desperate need of a solution. ARPANET (Advanced Research Projects Agency—Network) was the first long-distance computer network created by defence computer scientists. Remote military installations were then 'linked' through telephone lines.

Long-distance networking was recognized as a benefit by universities and scientists. They started connecting to ARPANET and to each other. By using the computer network, businesses and individuals began to connect. The Internet, a huge network of networks, eventually emerged. The Internet is now owned, administered, and maintained by everyone who uses it, not by any single person, business, or government.

(i) Proceed to a computer lab that is equipped with a connection to the internet.

(ii) Establish a connection to the internet by being familiar with the process of entering a phone number, user name, and password into a computer in order to gain access to the internet.

On the computer's desktop, click the Dial-up icon, which has been generated as a shortcut.

6.3. WEB PAGE

A document or information resource that is appropriate for use on the World Wide Web and that can be seen on a monitor or mobile device using a web browser is referred to as a Web page, which is also referred to simply as a Web page. These details are typically presented in HTML or XHTML format, and hypertext links may be followed in order to visit a variety of other websites. The final presentation of web pages typically includes additional resources such as style sheets, scripts, and graphics. This is standard practise.

Pages from the World Wide Web can be obtained from either a local or remote web server. The web server has the ability to either restrict access to a private network, such as an intranet for the corporation, or publish content to the public Internet. The Hypertext Transfer Protocol, or HTTP, is the protocol that is utilised to inquire for and receive web pages from web servers (HTTP).

Static web pages are comprised of files that are stored on the web server's file system and contain static text and other content. On the other hand, dynamic web pages are constructed by server-side software whenever the user makes a request for one of these pages (dynamic web pages). It is possible to make web pages more responsive to user interaction after they have been loaded on the client browser using client-side scripting.

6.4. WEBSITE

In most browsers, the URL of the page you were on before the one you're on can be found in the address bar (in Netscape Navigator) (in Internet Explorer). It's possible to get directly to a specific web page by typing its address into a field supplied in the location bar's address field.

When you want to input an address in the location bar, follow these steps:

1. Using your mouse, point and click within the location bar. You should take that address off the list.

2. Delete the previous address and enter the new one. One example of this is http://www.gmail.com.

3. Press the key labelled "Return."

Right next to the Location bar, you'll find a triangle-shaped button. When you click on the triangle, you'll be presented with a drop-down menu that lets you select addresses that you've recently been to.

6.5. BROWSERS

It is possible to use a graphical browser like Internet Explorer or Mozilla Firefox to access the internet; there are other text browsers like Lynx and Links. Accessing web pages typically requires users of the internet who have disabilities to make use of assisting technologies and adaptive strategies. Users may be colour-blind, may or may not wish to use a mouse due to repetitive stress injury or motor-neuron difficulties, may be deaf and require captioning, may be blind and require a screen reader or braille display, may require screen magnification, and so on. Users may also require other accessibility features such as screen magnification, braille displays, and screen readers. Both able-bodied and impaired users have the option to pause the downloading and viewing of images and other types of media. This can be done to save time

or network bandwidth, or just to make the browsing experience more straightforward. Display and bandwidth options are frequently restricted for users using mobile devices.

Anyone who does not wish to use the fonts, font sizes, styles, or colour schemes chosen by the web page designer is free to apply their own CSS styling to the page in question. Both the World Wide Online Consortium (W3C) and the Online Accessibility Initiative (WAI) believe that all web sites ought to be built with all of these various options in mind during the development process.

6.6. URL

In general, the level of interactivity that can be found on websites is increasing these days. After an end-user makes a request for a dynamic web page, the page is constructed on the server and then sent to the user. A permalink, or a static URL, is usually not linked with these sorts of web pages. This may now be found in a variety of prominent forums, online commerce, and even Wikipedia. Instead of keeping the appropriate web page information in a database, this method is meant to decrease the number of static pages. Because certain search engines have trouble indexing dynamic web pages, static web pages can be offered in those cases.

6.7. HTML

For creating web pages, the most used markup language is **HTML** (Hyper Text Markup Language). All websites are built on HTML Within the web page content, HTML is written in the form of HTML elements, which are tags contained in angle brackets (like html>). HTML tags are usually used in pairs, such as h1> and /h1>. The start tag is the first of two tags in a pair, while the end tag is the second (they are also called opening tags and closing tags).

Reading HTML text and creating visually or audibly appealing online pages are the primary responsibilities of a web browser. The tags are used to comprehend the page's content rather than being displayed by the browser.

All webpages are constructed using HTML components. HTML enables for the embedding of pictures and objects, as well as the creation of interactive forms. You can design structured documents by expressing structural semantics for text elements like headings, paragraphs, lists and other aspects of the text. This includes JavaScript-based scripts that alter the way HTML pages function.

6.8. INTERNET SERVICE PROVIDER (ISP)

It is possible for an Internet service provider (ISP) to be both a data host and an Internet access provider (ISP). Customers of access ISPs can be linked to the Internet via copper, wireless, or fibre connections, depending on the type of connection they choose. Small businesses can rent server space from hosting Internet service providers, which also host servers for other organizations (colocation). Large tubes are provided by transit ISPs for linking hosts.

6.9. SUMMARY

- Within the scope of this study, we shall investigate the function of the internet.
- You can access a web page by going to a local computer, or by going to a remote web sense.
- A bookmark is a record of the URL of a website.
- A range of distinct kinds of information, such as sun, may be included as one of the components of a website page.
- An intuitive and user-friendly web browser is one that displays web pages in a graphical format.

6.10. KEYWORDS

***Browsers*:** Your personal computer's web browser lets you browse the World Wide Web, which is a network of interconnected computers and devices.

The most common form of markup language for websites is known as ***HTML***, which stands for hypertext markup language. It serves as the basis for each and every webpage.

***ISP (internet service provider*):** The term "internet service provider" (abbreviated as "ISP") refers to a business that either provides internet access or hosts data.

6.11. FURTHER READING

Fundamental Computer Skills, Feng-Qi Lai, David R. Hofmeister.

A Study of Computer Fundamental, Shu-Jung Yi.

http://books.google.co.in/bkshp?hl=en

UNIT 7: INTRODUCTION TO PROGRAMMING

CONTENTS

OBJECTIVES:

- Understanding the basic concepts of Programming
- Learn about pre-processors
- Learn about Using the Instructions
- Learn about ISP

7.1 INTRODUCTION

In order to write even the most rudimentary programmes in any language, you will first need to acquire some essential knowledge. This chapter covers the introduction of three such fundamentals: the basic building blocks of a programme, variables, and input/output (I/O). In addition to it, it covers other language features such as comments, arithmetic operators, the increment operator, data conversion, and library functions.

7.2 THE FIRST STEPS

This book can be utilised with either a Microsoft or a Borland compiler, as was mentioned in the beginning of the text.

Compilers are tools that take source code and transform it into executable files, which can then be run on a computer just like any other programme. The extension denotes that the files in question are text files. CPP. Files that can be executed will end with the.exe suffix. EXE extension and can be executed straight from a DOS window or from within your compiler if you are familiar with MS-DOS.

7.3 CONSTRUCTION OF A BASIC PROGRAM

First, let's have a look at a simple programme written in C++. FIRST. This program's source code is located in the CPP file. It just displays a sentence on the computer screen. It's as follows:

```
#include <iostream>
using namespace std;
int main()
{
cout << “Every age has a language of its
own\n”;
return 0;
}
```

Despite its short size, this programme demonstrates a lot about C++ programme building. Let's take a closer look at it.

- **FUNCTIONS**

One of the most fundamental constructing elements in C++ are called functions. The primary function can almost entirely be credited as being completely responsible for the success of the FIRST programme (). The only lines in this programme that aren't part of the function are the first two lines, which are the ones that begin with #include and using. The rest of the lines are all part of the function.

The enclosing parenthesis that come after the word "main" are the telltale sign of a function. If the parentheses are not there, the compiler will make the assumption that main is referring to a variable or another component of the programme. When it comes to discussing functions in this article, we will follow the same convention as C++: After the name of the function, we will place the closing parentheses. As we are going to see in a little, the parentheses don't always have to be left empty. They are employed for the storage of function arguments, which are the values that are sent on to the function from the programme that is making the call.

If you see the prefix "int" before the name of a function, it indicates that the function will return a result of type "int." You don't need to worry about it right now; we'll go into data types in a bit more detail later on in this chapter.

- **THE FUNCTION OF THE BODY IN RELATION TO BRACES**

Braces surround the body of a function and hold it together (sometimes called curly brackets). These brackets fulfil the same function in a number of different languages as the BEGIN and END keywords: They are employed for the purpose of enclosing or separating a collection of programme statements. In every function, you are required to wrap this particular set of brackets around the function body. In this particular illustration, the body of the function is composed of just two statements: the line that starts with cout and the line that starts with return. The body of a function, on the other hand, may include a considerable number of statements.

- **ALWAYS BEGIN WITH THE MOST IMPORTANT ()**

When you run a programme written in C++, the statement that is executed first can be found at the beginning of a function called main (). At programme launch, control is always transferred to the main function, despite the fact that the application may have a huge number of functions, classes, and other types of programme objects (). If your programme does not

contain a function with the name main(), you will receive an error message when you attempt to run it.

- **STATEMENTS OF THE PROGRAM**

In C++ programming, the programme statement is the most basic unit. The FIRST programme contains two statements:

cout<< "Every age has its own language";

and the return statement

return 0;

The computer is given the first instruction, which tells it to display the sentence that was quoted. The vast majority of statements provide the computer instructions to carry out a certain task. In this respect, statements in C++ are comparable to statements in other programming languages. In point of fact, as we've seen, the vast majority of statements in C++ are identical to statements in C.

A semicolon is used as the punctuation after the sentence. Despite the fact that this is an essential part of the syntax, most people seem to overlook it. In some programming languages, such as BASIC, the end of a line denotes the conclusion of a statement; however, this is not the case in C++. If you forget to include the semicolon in your code, the compiler will almost always—but not always—give you an error message.

7.4 DIRECTIVES FOR THE PRE-PROCESSORS

Although the first line of the FIRST programme, which reads "#include iostream>," may give the impression that it is a programme statement, this is not the case. It is not supposed to be in the main body of a function, and it does not conclude with a semicolon as required by the format of programmed statements. In its place, a number sign, denoted by the symbol #, serves as the initial character.

It is referred to as a pre-processor directive in the industry. It is important to keep in mind that programme statements are the instructions that tell the computer how to perform actions such as adding two numbers or printing a message. On the other hand, an instruction given to the compiler is known as a pre-processor directive. These directives are processed by the pre-

processor, which is a component of the compiler, prior to the beginning of the process of actually compiling the code.

The #include pre-processor directive gives the compiler the instruction to include another file in the source file that you are working on. The #include directive is, in practise, superseded by the contents of the file that was supplied. The technique of including another file into your source file by using a #include directive is comparable to incorporating a section of text into a document created by a word processor.

- **FILES WITH HEADERS**

Before beginning the compilation process, the pre-processor directive #include in the FIRST example instructs the compiler to include the source file IOSTREAM in the source file for FIRST.CPP. What is the purpose of your actions? IOSTREAM is a good illustration of a header file (sometimes called an include file). It covers the fundamental operations of input and output and includes declarations that are necessary for the cout identifier and the operator.

Without these declarations, the compiler will be unable to recognise the cout variable and will assume that it is being used incorrectly. There are a significant number of inclusion files available today. Although some of the earlier Standard C++ header files, some of which date all the way back to the days of the C programming language, had a file extension, the more recent Standard C++ header files do not have one. H extension.

7.5 THE USING INSTRUCTIONS

In a C++ application, you have the ability to construct many namespaces. A namespace is a region of the programme in which specific names are recognised; these names are unknown anywhere else in the programme. When you use the directive using namespace std;, you are telling the computer that all of the statements that follow will be placed in the std namespace. This namespace has declarations for a variety of programme components,

including cout, amongst others. If we did not utilise the using directive, we would have to insert the std name into a great number of different areas of the programme. In the FIRST programme, for example, we'd need to say

std:: cout <<"Every generation has its own language.";

We utilise the using directive instead of std:: hundreds of times in our scripts to avoid adding std:: dozens of times.

7.6 COMMENTS

Without comments, a piece of software can never be considered complete. They make it simpler for programmers and anyone else who must read the source code to understand what is going on in the programme. Because the compiler disregards comments, their presence has no bearing on the total size of the program's executable file or the amount of time it takes to run.

Syntax Comment

Let's rewrite our FIRST programme with comments included in the source file. The new programme will be called COMMENTS:

```
// comments.cpp
// demonstrates comments
#include                    //preprocessor directive
using namespace std;        //"using" directive
int main()                  //function name "main"
{                           //start function body
cout << "Every age has a language of its own\n";
//statement
return 0;                                   //statement
}                       //end function body
```

- **WHEN SHOULD YOU USE COMMENTS?**

Your thoughts and opinions are almost always welcome. The majority of programmers don't make nearly enough use of them. Keep in mind that not everyone is as educated as you are, and they may require more explanation than you do about what your software is doing if they are using it for the first time. This is especially true if you are inclined to skip the comments.

A month from now, when you will have likely forgotten essential components of the way your application works, you will not be as perceptive as you are right now.

Make use of the comments to provide the person who is looking at the list with an explanation of what it is that you are trying to do. Because the specifics are contained inside the programme statements themselves, the comments should concentrate on the bigger picture and explain the reasoning behind why a certain statement or group of statements was utilised.

- **SYNTAX FOR ALTERNATIVE COMMENTS**

In C++, there's also a second comment style:

/* This is a traditional comment */

This style of comment (which was the only one available in C at the time) starts with the /* character pair and finishes with */. (not with the end of the line). Because these symbols are more difficult to type (because / is lowercase and * is uppercase) and take up more space on the line, they are not commonly used in C++. It does, however, have advantages in certain circumstances. Only two comment symbols are required to write a multiline comment:

/* This might

be a multiline

comment with a

lot of lines */

7.5 VARIABLES

The most fundamental components of any programming language are called variables. A symbolic name is given to a variable, and the variable can have any one of a wide variety of values assigned to it. The memory of the computer is divided up into distinct sections, and each of those sections is where variables are kept.

When a value is assigned to a variable, the value is saved in the portion of memory that has been designated as the variable's storage location. mostly due to the fact that the majority of popular programming languages make use of the same fundamental types of variables, such as integers, floating-point numbers, and characters.

- **NAMING OF VARIABLES**

What are the guidelines for writing an identifier? You are welcome to make use of uppercase and lowercase letters, as well as numerals ranging from one to nine. There is also the option of using an underscore ( ). The very first character must either be a letter or an underscore. You have complete control over how long an identifier is, but the majority of compilers will only read the first few hundred characters of it. Because the compiler makes a distinction between uppercase and lowercase characters, the term Var should not be confused with var or VAR.

7.6 SUMMARY:

- Computer programming is introduced in this Unit, along with its definition and syntax.

7.7 KEYWORDS:

Variables

Comments

ISP

Pre-processors

7.8 REVIEW QUESTIONS:

1. Which of the following best describes the object-oriented programming concept of polymorphism?

   a. The properties of an object can change with the context.

   b. Hiding internal mechanisms and data structures behind a defined interface.

   c. New classes are created from existing classes and take on the general characteristics of those classes.

   d. Hiding many of the details about an object and showing only the most relevant information.

   2. An object is

   a. the hiding of internal mechanisms and data structures behind a defined interface.

b. a procedure associated with a class.

c. a component of a program that knows how to perform certain actions and how to interact with other elements of the program.

d. a combination of instructions that are combined to achieve some result.

3. Which of the following best describes the enumerated data type?

a. Can only represent two values: true or false.

b. Values can only represent one of a limited number of predefined categories.

c. Represents alphanumeric data.

d. Elements are identified by one or more indices.

4. Which of the following best describes the Boolean data type?

a. Values can only represent one of a limited number of predefined categories.

b. Can only represent two values: true or false.

c. Elements are identified by one or more indices.

d. Represents alphanumeric data.

5. Which is one of the earliest programming languages still in use?

a. COBOL

b. C#

c. JavaScript

d. Visual Studio .NET

7.9 FURTHER READINGS:

Fundamental Computer Skills, Feng-Qi Lai, David R. Hofmeister.

A Study of Computer Fundamental, Shu-Jung Yi.

http://books.google.co.in/bkshp?hl=en

UNIT 8: DATA TYPES

CONTENTS

OBJECTIVES:

- Understanding the basic concepts of Variable declaration and defining Programming
- Learn about Escape Sequences
- Learn about Input and Floating Points
- Learn about Type bool and Casts

8.1. INTRODUCTION

There is a wide range of sizes available for integer variables, with the type int being the most common. Depending on the architecture, various types of integers require varying amounts of memory.

8.2. IDENTIFYING AND DEFINING INTEGER VARIABLES

On a 32-bit operating system such as Windows, an integer consumes four bytes of memory (32 bits). This makes it possible for an int to store numbers in the range of –2,147,483,648 to 2,147,483,647.

```
#include <iostream>
using namespace std;
int main()
{    int var1;                  //define var1
        int var2;               //define var2
        var1 = 20;              //assign value to var1
        var2 = var1 + 10;   //assign value to var2
         cout << "var1+10 is ";     //output text
        cout << var2 << endl;      //output  value  of
var2
        return 0;
 }
```

- **VARIATIONS IN OUTPUT**

As we've seen before, the statement

```
cout <<"var1+10 is ";
```

displays a string constant. The following statement,

```
cout <<var2<< endl;,
```

shows the value of the variable var2. As you can see in the window labelled "console output," the output of the programme is var1+10, which equals 30. On the output screen, you can see that the output of

the two cout statements appears on the same line. There is no linefeed that is automatically inserted. If you would like to begin writing on a new line, you will need to manually enter a linefeed. With the endl, we were able to see how to proceed with this.

- **TYPES OF INTEGERS**

There are a variety of different numerical integer types available in addition to the type int. The two most common lengths are long and short. Neither one is more common than the other. (Character type is theoretically also an integer type, but we'll deal with it on an individual basis.) As was noted before, the size of an int type variable is determined by the system. On the other hand, types long and short always have the same dimensions, independent of the particular system that is being used.

Type long always uses up four bytes, the same as type int on 32-bit Windows machines. This is the same as the type int value. As a direct consequence of this, the range remains unchanged at -2,147,483,648 to 2,147,483,647. You could alternatively write it as long int, which is essentially the same thing as long. Both of these forms mean the same thing. On 32-bit computers, the use of the long data type is unnecessary because it is equivalent to the int data type. On 32-bit computers, the use of the long data type is unnecessary because it is equivalent to the int data type. If your software is required to function on a 16-bit system, such as MS-DOS or an earlier version of Windows, giving the type long will guarantee a four-bit integer type. On 16-bit platforms, the range of type int is identical to that of type short.

On any system, the type short uses up two bytes, which gives it a range that can go from – 32,768 to 32,767. If you're using a modern Windows machine, it's generally not worth your time to use type short unless you really need to save some memory. Even though type int has twice as much space as type short, it may be accessed much more quickly.

8.3. VARIABLES OF CHARACTER

The char data type may store integers with values that range from -128 all the way up to 127. Memory is only consumed by one byte (eight bits) when a variable of this kind is being used. Even though they are most commonly used to store ASCII characters, character variables can on occasion also be used to store numbers that are constrained to this range. However, this is much less common. As you may already be aware, the ASCII character set is a notation that

uses integers to represent characters such as "a," "B," "$," and "3," amongst others. These numbers can take on any value between 0 and 127.

```
#include <iostream> //for cout, etc.
using namespace std;
int main()
{
char charvar1 = 'A';            //define char variable as character
char charvar2 = '\t';           //define char variable as tab
cout << charvar1;               //display character
cout << charvar2;               //display character
charvar1 = 'B';                 //set char variable to char
constant
cout << charvar1;               //display character
cout << '\n';                   //display newline character
return 0;
}
```

8.4. SEQUENCES OF ESCAPE

'\t', the second character constant, is an unusual one. It's an example of an escape sequence, similar to '\n', which we saw earlier. The name comes from the fact that the backslash causes characters to "escape" from their typical interpretation. The t is regarded as the tab character rather than the character 't'. When you press the tab key, the printing will continue to the next tab stop.

Tab stops are placed every eight spaces in console-mode programmes. In the last line of the programme, another character constant, '\n,' is passed directly to cout.

Escape sequences can be used as individual characters or as part of a string constant. A list of frequent escape sequences can be found in Table.

Escape Sequence	Character
\ a	Bell (beep)
\ b	Backspace
\ f	Formfeed
\ n	Newline
\ r	Return
\ t	Tab
\ \	Backslash
\ ‘	Single quotation mark
\ “	Double quotation marks
\ xdd	Hexadecimal notation

8.5. INPUT WITH CIN

Now that we've seen several different types of variables in action, let's take a look at how a programme deals with incoming data.

The following programme prompts the user for the current temperature in Fahrenheit degrees, then converts it to Celsius degrees and displays the result. Integer variables are used.

```
#include using namespace std;
 int main()
{ int ftemp;                    //for temperature in fahrenheit
cout << “Enter temperature in fahrenheit: “;
cin >> ftemp;
int ctemp = (ftemp-32) * 5 / 9;
cout << “Equivalent in Celsius is: “ << ctemp << ‘\n’;
 return 0; }
```

The statement

```
cin >> ftemp;
```

instructs the software to wait for a number to be entered by the user.

The value obtained is stored in the variable ftemp.

The term cin (pronounced "C in") refers to a C++ object that represents the standard input stream.

The data from the keyboard is represented by this stream (unless it has been redirected). The extraction or get from operator is the >>. It transfers the value from the stream object on the left to the variable on the right.

Here's an example of how you might interact with the programme:

Here's some sample interaction with the program:

Enter temperature in fahrenheit: 212

Equivalent in Celsius is: 100

8.6. FLOATING POINT DATA TYPES

Integers have been represented as type int and characters as type char, both of which have been covered (i.e., numbers without a fractional part). Let's take a look at floating-point variables, which is yet another approach for storing numerical values.

Floating-point variables are used to represent numbers that include one or more places of decimal representation, such as 3.1415927, 0.0000625, and –10.2.

There is an integer component that can be found to the left of the decimal point, and there is a fractional component that can be found to the right of the decimal point.

Real numbers are what floating-point variables are, and mathematicians use them to express quantifiable qualities like distance, area, and temperature.

Floating-point variables are what mathematicians refer to as real numbers. A fractional component is commonly included in them.

The user is prompted to fill in a floating-point integer that represents the radius of a circle in the following sample programme. The area of the circle is then calculated and displayed.

```
#include<iostream>                          //for cout, etc.
using namespace std;
int main() {
float rad;                                 //variable of type
float
const float PI = 3.14159F;                 //type const float
cout << "Enter radius of circle: ";        //prompt
cin >> rad;                                //get radius
float area = PI * rad * rad;               //find area
cout << "Area is " << area << endl;        //display answer
return 0; }
```

Here's a sample interaction with the program:

```
Enter radius of circle: 0.5
Area is 0.785398
```

- **DOUBLE AND LONG DOUBLE**

Compared to float, the larger floating point types, double and long double, provide a greater range of values and precision, but they also consume more memory than float does. Float is the smallest of the floating point types. The double data type requires 8 bytes of storage and has the ability to work with numbers that have a precision of 15 digits within the range of 1.7x10-308 to 1.7x10308. Although the type long double can be different depending on the compiler, it is typically the same as double.

8.7. THE #DEFINE DIRECTIVE

Constants can also be declared using the pre-processor directive #define, though this is not encouraged in C++. This directive establishes a text phrase's equivalency with an identifier. For example, the line

```
#define PI 3.14159
```

At the beginning of your programme, you should specify that the identification PI will be replaced by the phrase 3.14159 at all other points throughout the programme. Since the beginning of time, programmes written in C have frequently utilised this form.

#define, on the other hand, does not permit you to specify the data type of the constant, which can lead to problems in the programming; as a result, const, which is used with regular variables, has mainly replaced #define in C. const is used with regular variables. This structure, on the other hand, is sometimes present in older programmes.

8.8. TYPE BOOL

Even though we won't need to worry about this type of thing until we get to the section on relational operators in the following chapter, we should still go ahead and specify it now just to be thorough. Variables of type char can only contain a maximum of 256 potential values, whereas variables of type int can have a maximum of billions. Boolean variables can only take the values true and false as their respective values.

In principle, a bool type needs only one bit of storage, not a byte, but in practise, compilers usually store them as bytes. This is because a byte can be retrieved fast, whereas a single bit needs to be extracted from a byte, which takes more time.

Summary of Data Types:

Type	Definition	Control Character	Limits
int	Integer		-2147483648 to 2147483647
short	Short Integer		-32768 to 32767
long	Long Integer	l or L	-2147483648 to 2147483647
float	Floating Decimal Number	f or F	1.17549e-038 to 3.40282e+038
double	Double Decimal Number		2.22507e-308 to 1.79769e+308
long double	Long Decimal Number		2.22507e-308 to 1.79769e+308
char	Character		-128 to 127
unsigned int	Unsigned Integer		0 to 4294967295
unsigned short	Unsigned Short Integer		0 to 65535
unsigned long	Unsigned Long Integer		0 to 4294967295
unsigned char	Unsigned Character		0 to 255
bool	True or False		True = 1 and False = 0

8.9. TYPE CASTS

Casts may sound like something from India's social classes, but in C++, the term refers to data conversions that the programmer defines rather than the automatic data conversions that we just covered. This is in contrast to the automatic data conversions that we just discussed. Casts can also be referred to by their type name.

What exactly are the reasons castings are done?

A programmer is required to manually convert a value from one type to another when the compiler is unable to do it automatically or when it does so with an error message.

```
#include <iostream>
using namespace std;
int main()
{
int intVar = 1500000000;          //1,500,000,000
intVar = (intVar * 10) / 10;              //result too large
cout << "intVar = " << intVar << endl;       //wrong answer
intVar = 1500000000;                      //cast to double
intVar = (static_cast<double>(intVar) * 10) / 10;
cout << "intVar = " << intVar << endl;                 //right
answer
return 0;
}
```

8.10. SUMMARY

- This section provides in-depth understanding on a variety of topics, including escape sequences, characters, and variables. We investigated the input using cin, as well as type bool and casts.

8.11. KEYWORDS

Characters

Cin

Type

Directives

8.12. REVIEW QUESTIONS

1. What is a variable in C language?
2. Give the rules for variable declaration?
3. How to declare a variable in C language?
4. What is the difference between declaring a variable and defining a variable?
5. Define the term Scope, Visibility and Lifetime of a Variable?
6. What are the different types of variables depending on variable scope?

8.13. FURTHER READINGS

Fundamental Computer Skills, Feng-Qi Lai, David R. Hofmeister.

A Study of Computer Fundamental, Shu-Jung Yi.

http://books.google.co.in/bkshp?hl=en

UNIT 9: OPERATORS IN C++

CONTENTS

OBJECTIVES:

- Understanding the basic concepts of Operators in Programming
- Learn about Library Function
- Learn about Conditional and Logical Operators

9.1. INTRODUCTION

An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators and provides the following types of operators

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators

9.2. ARITHMETIC OPERATORS

C++ uses the four conventional arithmetic operators, which are the plus sign (+), the minus sign (-), the multiplication symbol (*), and the division symbol (/), to perform operations such as adding, subtracting, multiplying, and dividing. These operations can be carried out on data that is represented in both integer and floating-point forms. They are utilised in a manner that is analogous to the manner in which they are utilised in other languages, as well as the manner in which they are utilised in mathematics. On the other hand, the applications of the many other arithmetic operators are not as straightforward.

9.3. THE REMAINDER OPERATOR/MODULO DIVISION

A fifth arithmetic operator can only work with variables that are integers, including char, short, int, and long variables. The symbol for the remaining operator, which is also referred to as the

```
#include <iostream>
using namespace std;
int main()
{
cout << 6 % 8 << endl // 6
<< 7 % 8 << endl // 7
<< 8 % 8 << endl // 0
<< 9 % 8 << endl // 1
<< 10 % 8 << endl; // 2
return 0;
}
```

remainder operator, is the percent sign (percent).This operator, which is also referred to as the modulus operator, determines the residual that is left over when one integer is divided by another integer.

The remainder operator is used to divide the digits 6–10 by 8. The remainders of these divisions are 6, 7, 0, 1, and 2, respectively. The remainder operator can be used in a variety of ways. As we go ahead, we'll provide examples. A word regarding the order of things: Because the remaining operator takes precedence over the operator in the expression cout 6 percent 8, it is evaluated first. If it didn't, we'd have to insert parenthesis around 6% 8 to make sure it was assessed before being acted on by.

9.4. ARITHMETIC ASSIGNMENT OPERATORS

C++ provides a number of options for shortening and clarifying your code. The arithmetic assignment operator is one of them. This operation contributes to the distinctive appearance of C++ listings.

In most languages, the following type of statement is widespread.

total = total + item; / "item" is added to the "total"

In this situation, you are doing a mathematical operation on a value that already exists, which may involve adding something to the value or performing another operation. However, individuals who value brevity will be offended by the fact that the word total appears twice in this statement. As a result, C++ provides a streamlined solution: the arithmetic assignment operator is a hybrid that combines the functions of an arithmetic operator and an assignment operator. The following proclamation accomplishes precisely the same goal as the one that came before it.

total += item; // adds “item” to “total”

```
#include <iostream>
using namespace std;
int main()
{
int ans = 27;
ans += 10;                  //same as: ans = ans + 10;
cout << ans << “, “;
```

```
    ans *= 2;                   //same as: ans = ans * 2;
    cout << ans << ", ";
    ans /= 3;                   //same as: ans = ans / 3;
    cout << ans << ", ";
    ans %= 3;                   //same as: ans = ans % 3;
    cout << ans << endl;
    return 0;
}
```

Here's the output from this program: 37, 30, 60, 20, 2

9.5. INCREMENT OPERATORS

Here's an even more specialized operator. You often need to add 1 to the value of an existing variable. You can do this the "normal" way:

```
count = count + 1;          // adds 1 to "count"
```

Or you can use an arithmetic assignment operator:

```
count += 1;                 // adds 1 to "count"
```

But there's an even more condensed approach:

```
++count;                    // adds 1 to "count"
```

The ++ operator increments (adds 1 to) its argument.

- **PREFIXES AND SUFFIXES**

As if that wasn't strange enough, the increment operator can also be used in two different ways: first, it can be used as a prefix, which means it comes before the variable, and second, it can be used as a postfix, which means it comes after the variable.

Both of these uses are just as odd as the first. What exactly differentiates the two from one another? It is common practise to conduct an increment on a variable as part of a statement that also performs another operation on the variable.

TotalWeight = avgWeight * ++count,

for example.

The issue at hand is this: Is the multiplication done before or after incrementing the count?

In this particular instance, the count is increased right off the off. What kinds of evidence do we have to support this claim? because the expression ++count is represented by the prefix notation. If we had used postfix notation, which is represented as count++, the multiplication would have been performed first, and then the count would have been incremented.

```
#include <iostream>
using namespace std;
int main()
{
int count = 10;
cout << "count=" << count << endl; //displays 10
cout << "count=" << ++count << endl; //displays 11 (prefix)
cout << "count=" << count << endl; //displays 11
cout << "count=" << count++ << endl; //displays 11 (postfix)
cout << "count=" << count << endl; //displays 12
return 0;
}
```

Here's the program's output:

```
count=10
count=11
count=11
count=11
count=12
```

When the count is incremented for the first time, the operator with the prefix "++" is utilised. As a direct consequence of this, the increment takes place right at the beginning of the evaluation of the statement, far before the output operation is finished.

When the value of the expression ++count is displayed, it has already been incremented, thus it observes the number 11 instead of the previous value. The increment of the second time count

is accomplished by the usage of the postfix ++ operator. The value of 11 is not increased when the expression count++ is supplied; instead, it remains the same.

As soon as this line is finished being typed, the increment begins to take effect, and we can see that count has now reached the number 12 in the statement that comes before this one in the programme.

9.6. LIBRARY FUNCTIONS

C++ library functions are responsible for a wide array of responsibilities. File access, mathematical computations, and the conversion of data are only some of the things that are handled by these functions. You don't need to understand how simple library functions operate in order to use them, but we don't want to get into too much depth regarding library functions before we discuss how they work.

The square root of a value given by the user is calculated using the library function sqrt() in the next example.

```
#include <iostream>                    //for cout, etc.
#include <cmath>                       //for sqrt()
using namespace std;
int main()
{
double number, answer;                 //sqrt()  requires  type
double
cout << "Enter a number: ";
cin >> number;                         //get the number
answer = sqrt(number);                 //find square root
cout << "Square root is "
<< answer << endl;                     //display it
return 0;
}
```

The programme starts by asking the user for a number. In the statement

answer = sqrt(number);

this number is used as an argument to the sqrt() function.

The input to the function is called an argument, and it is enclosed in parentheses after the function name. After that, the function processes the input and produces a value, which is the function's output. The return value in this situation is the square root of the original integer.

Returning a value indicates that the function expression accepts the value, which may then be assigned to another variable, in this example answer.

Here's some output from the program:

```
Enter a number: 1000
Square root is 31.622777
```

9.7. RELATIONAL OPERATORS

A comparison operator that examines two values simultaneously is referred to as a relational operator. As we will see in the following section, values can be any built-in C++ data type, such as char, int, or float, or they can be user-defined classes. The comparison makes use of a variety of relationship comparisons, including equal to, less than, and greater than. A true or false conclusion can be drawn from the comparison; for instance, two values are either equivalent (which would be true) or not (which would be false).

```
#include <iostream>
using namespace std;
int main()
{
int numb;
cout << "Enter a number: ";
cin >> numb;
cout << "numb<10 is " << (numb < 10) << endl;
cout << "numb>10 is " << (numb > 10) << endl;
cout << "numb==10 is " << (numb == 10) << endl;
return 0;
}
```

This program performs three kinds of comparisons between 10 and a number entered by the
user. Here's the output when the user enters 20:

```
Enter a number: 20
numb<10 is 0
numb>10 is 1
numb==10 is 0
```

Operator	Description	Left operand (a)	Right operand (b)	Outcome
==	Checks if both operands are equal	4	5	4 == 5 false
!= or <>	Checks if both operands differ from each other	4	5	4 != 5 true
>	Checks if the left operand is greater than the right operand	4	5	4 > 5 false
<	Checks if the left operand is smaller than the right operand	4	5	4 < 5 true
>=	Checks if the left operand is greater than or equal to the right operand	4	4	4 >= 4 true
<=	Checks if the left operand is smaller than or equal to the right operand	3	4	3 <= 4 true

9.8. CONDITIONAL OPERATOR

This operator is composed of two symbols, and it can function with three different operands. It is unique among all other operators in C++, which only accept either one or two operands to do their tasks. The following is an alternative representation of the same portion of the programme that uses a conditional operator:

min = (alpha<beta) ? alpha : beta;

The part of this statement to the right of the equal sign is called the conditional expression:

(alpha<beta) ? alpha : beta // conditional expression

The question mark and the colon are the components that make up the conditional operator. The expression that comes before the question mark (alpha beta) is the one that will be evaluated. It, alpha, and beta are the two operands in this equation.

If the test expression produces a successful result, the entire conditional expression will be evaluated using the value of the operand that is followed by the question mark; in this example,

that value will be alpha. If the value of the test expression is not true, then the value of the beta operand that comes after the colon will be accepted by the conditional expression.

9.9. LOGICAL OPERATORS

These operators allow you to logically join Boolean variables together, and you can utilise them to do so (that is, variables of type bool, with true or false values). For example, the statement "today is a weekday" has a Boolean value since the statement can either be true or false.

Another example of a Boolean expression is "Maria drove the automobile." These idioms are connected in a way that is consistent with logic:

If it is a weekday and Maria has already used the car, then I will have no choice but to ride the bus. The conjunction "and" is the logical link in this situation; it determines whether the combined value of the two propositions is true or false. If both of those things are true, then it's certain that I'll have to ride the bus.

Table 9.1

Logical Operators		
For all examples below consider a = 10 and b = 5		
Operator	**Description**	**Example**
&&	Logical AND	(a>b) && (b==5) gives true
||	Logical OR	(a>b) || (b==2) gives true
!	Logical NOT	!(b==5) gives false

9.10. SUMMARY

Let's take a look at the order of precedence for the operators that we've been learning about up until this point. The operators that are located at the top of the list are given priority over those that are located further down.

The evaluation begins with the operators that have the highest priority and continues with those that have the lowest priority. The priorities of operators located in the same row are same. By enclosing an expression in parentheses, you can force the expression to be evaluated before anything else.

9.11. KEYWORDS:

Operators

Library Function

Remainder

9.12. REVIEW QUESTIONS

1. Arrange in order of precedence (highest first) the following kinds of operators: logical, unary, arithmetic, assignment, relational, conditional.
2. The && and || operators
    a. compare two numeric values.
    b. combine two numeric values.
    c. compare two Boolean values.
    d. combine two Boolean values

3. Write a temperature-conversion program that gives the user the option of converting

Fahrenheit to Celsius or Celsius to Fahrenheit. Then carry out the conversion. Use

Floating-point numbers. Interaction with the program might look like this:

Type 1 to convert Fahrenheit to Celsius,
2 to convert Celsius to Fahrenheit: 1
Enter temperature in Fahrenheit: 70
In Celsius that's 21.111111

4. Write a program to calculate maximum number among three numbers using coditional operator.
5. Write a program to swap two variables without using 3rd variable.

9.13. FURTHER READING

Fundamental Computer Skills, Feng-Qi Lai, David R. Hofmeister.

A Study of Computer Fundamental, Shu-Jung Yi.

http://books.google.co.in/bkshp?hl=en

UNIT 10: CONTROL STATEMENTS

CONTENTS

OBJECTIVES:

- Understanding the basic concepts of control statements in Programming
- Learn about if statements
- Learn about switch and break

10.1 INTRODUCTION

C++ allows you to make decisions in a variety of ways. The most crucial is the if...else phrase, which allows you to pick between two options. A straightforward if statement can be constructed out of this expression by omitting the otherwise clause. The switch statement is another type of decision statement that allows the programmer to select one of several different paths through the code based on the value of a single variable. Lastly, the conditional operator is utilised when appropriate in a number of contexts. Each of these structures will receive its own undivided attention.

10.2 IF STATEMENT

The if statement is the simplest of the decision statements. Our next program, IFDEMO, provides an example.

```
#include <iostream>
using namespace std;
int main()
{
int x;
cout << "Enter a number: ";
cin >> x;
if( x > 100 )
cout << "That number is greater than 100\n";
return 0;
}
```

The if keyword is followed by a test expression in parentheses. The difference is that the statements following the if are executed only once if the test expression is true;

The output of above program is:

Enter a number: 2000

That number is greater than 10

- **MULTIPLE STATEMENTS IN THE IF BODY**

The code in an if body can be a single statement or a block of statements separated by braces.

```
#include <iostream>
using namespace std;
int main()
{
int x;
cout << "Enter a number: ";
cin >> x;
if( x > 100 )
{
cout << "The number " << x;
cout << " is greater than 100\n";
}
return 0;
}
```

Here's some output from :

```
Enter a number: 12345
The number 12345 is greater than 100
```

- **THE IF...ELSE STATEMENT**

The if statement enables you to carry out some action in the event that a predetermined condition is satisfied. If that is not true, then nothing will occur.

On the other hand, suppose we wish to carry out one set of actions if a condition is satisfied, and another set of actions if it is not.

In this particular scenario, the if...else statement is utilised. It begins with an if statement, then continues with a statement or block of statements, then moves on to the keyword else, and finally continues with yet another statement or series of statements.

```
#include <iostream>
using namespace std;
int main()
{
int x;
cout << "\nEnter a number: ";
cin >> x;
if( x > 100 )
cout << "That number is greater than 100\n";
else
cout << "That number is not greater than 100\n";
return 0;
}
```

If the if statement's test expression is true, the programme produces one message; if it isn't, the programme prints the other. The following are the results of two different commands of the programme:

```
Enter a number: 300
That number is greater than 100
Enter a number: 3
That number is not greater than 100
```

- **NESTED IF...ELSE STATEMENTS**

It's possible that sometimes we'll require nesting an if statement inside of another if statement. These kinds of statements are referred to as "nested if statements."

Think of it as a string of if statements that have been piled one on top of the other. After the initial, parenthetical if statement comes the subsequent, parenthetical if statement. The following is how its syntax looks:

```
#include <iostream>
using namespace std;
int main() {
 int num;
 cout << "Enter an integer: ";
  cin >> num;
 // outer if condition
 if (num != 0) {

  // inner if condition
  if (num > 0) {
   cout << "The number is positive." << endl;
  }
  // inner else condition
  else {
   cout << "The number is negative." << endl;
  }
 }
 // outer else condition
 else {
  cout << "The number is 0 and it is neither positive nor
negative." << endl;
 }
 cout << "This line is always printed." << endl;
 return 0;
}
```

The integer that was entered by the user is read into the num variable, where it is stored. After that, we check to see if num is greater than zero using an if...else expression. If the condition is met, the expression contained within the if...else block is executed. In the event that the initial condition is not met, the code contained within the outer else condition will be executed, which will output "The number is 0 and neither positive nor negative." If num is greater than 0, the if...else statement in the middle will determine whether or not the input number is positive. If the condition is met, a comment indicating that the number is positive will be produced. If the condition is not met, a negative value is assigned to this integer.

10.3 SWITCH STATEMENT

If you have a big decision tree where all of the decisions are based on the value of the same variable, you should generally use a switch statement rather than a ladder of if... statements.

```
#include <iostream>
using namespace std;
int main()
{
int speed; //turntable speed
cout << "\nEnter 33, 45, or 78: ";
cin >> speed; //user enters speed
switch(speed) //selection based on speed
{
case 33: //user entered 33
cout << "LP album\n";
break;
case 45: //user entered 45
cout << "Single selection\n";
break;
case 78: //user entered 78
cout << "Obsolete format\n";
break;
}
return 0;
}
```

This application will print one of three different messages depending on whether the user enters the number 33, 45, or 78 into its input field. Long-playing records, often known as LPs, could hold many songs and rotated at a speed of 33 revolutions per minute (rpm). In comparison, 45s could only store a single track, and 78s were the format that came before LPs and 45s.

The switch keyword is then followed by the variable that it refers to, which is surrounded by parentheses. Following that, a number of case statements will have switch Braces positioned

between them. Each case keyword is then followed by a constant that does not have parentheses surrounding it, and then a colon comes after that. The data type of the case constants should be the same as the data type of the switch variable.

10.4 BREAK STATEMENT

The switch statement is completely terminated when the break keyword is used. After the completion of the switch construction, control will then proceed to the first statement, which in PLATTERS will result in the termination of the programme. Don't forget the break; if you don't include it, control will go on (or "fall through") to the statements of the following case, which is not what you want most of the time.

```
#include <iostream>
using namespace std;
#include <conio.h>                    //for getche()
int main()
{
char dir='a';
int x=10, y=10;
while( dir != '\r' )
{
cout << "\nYour location is " << x << ", " << y;
cout << "\nEnter direction (n, s, e, w): ";
dir = getche(); //get character
switch(dir) //switch on it
{
case 'n': y--; break;              //go north
case 's': y++; break;              //go south
case 'e': x++; break;              //go east
case 'w': x--; break;              //go west
case '\r': cout << "Exiting\n"; break;  //Enter key
default: cout << "Try again\n";                //unknown char
        }                                  //end switch
}                                  //end while
return 0;
}                                  //end main
```

10.5 SUMMARY

Relational operators evaluate two integers to determine whether or not they are comparable to one another and whether or not one is greater than the other.

The outcome is a logical or Boolean value of type bool that is either true or false. The value 0 is used to represent false, while the value 1 (or any other non-zero value) is used to represent true.

Decision statements can be broken down into one of these four categories.

The if statement will take action based on whether or not a test phrase is true.

The if...else sentence will carry out one set of actions depending on the value of the test expression; if the expression is false, the phrase will carry out a different set of actions.

A readable form of rewriting a hierarchy of nested if...else expressions can be created by using the else if structure. The value of a single variable determines which of several different sections of code are executed when the switch statement is executed.

The conditional operator will return one value depending on the result of a test expression; if the expression's result is false, it will return a different value.

While the logical AND and OR operators combine two Boolean expressions to make a new one, the logical NOT operator changes a Boolean value from true to false or false to true.

This is in contrast to the other logical operators, which combine two Boolean expressions to produce a new one.

The control will be moved to the finish line of the loop or switch that contains the break statement when it is executed. The control is taken back to the beginning of the loop that the continue statement is now in.

10.6 KEYWORDS:

If

Switch

Break

10.7 REVIEW QUESTIONS

1. A relational operator

    a. assigns one operand to another.

    b. yields a Boolean result.

    c. compares two operands.

    d. logically combines two operands.

Create an expression that, if the variable george is not equal to sally, will yield true thanks to the utilisation of a relational operator.

3. Is –1 true or false?
4. Create an if-then statement that will output "Yes" if the age of a variable is more than 21 and will output "No" otherwise. In any other case
5. Construct a statement that makes use of a logical operator and evaluates to the value true if the speed restriction is 55 and the speed is more than 55.
6. Construct something that works like a calculator with four functions. The user should be prompted to enter a number, followed by an operator, and then another number while using the software. (Floating point should be used.) After that, it should carry out the arithmetic operation that has been provided, which could be adding, subtracting, multiplying, or dividing the two numbers. To pick the procedure, you might make use of a switch statement. Finally, show the result to the user. When the calculation is complete, the application should inquire as to

Enter the first number, followed by the operator, and then the second number: (10 / 3)

Answer = 3.333333

Do another (y/n)? y

Input the first number, followed by the operator, and then the second number: 12 + 100

Answer = 112

Do another (y/n)? n

10.8 FURTHER READING

Fundamental Computer Skills, Feng-Qi Lai, David R. Hofmeister.

A Study of Computer Fundamental, Shu-Jung Yi.

http://books.google.co.in/bkshp?hl=en

UNIT 11: LOOPS

CONTENTS

OBJECTIVES:

- Understanding the basic concepts of looping in Programming
- Learn about for and while loop
- Learn about how to use while loop

11.1 INTRODUCTION

A portion of your software can be made to repeat itself an infinite number of times by using loops. As long as a certain condition is met, the pattern will be repeated. Once the condition is no longer met, the iteration of the loop is finished, and control moves on to the statements that come after it.

There are three distinct kinds of loops that can be used in C++: the for loop, the while loop, and the do loop.

11.2 FOR LOOP

The for loop is the C++ loop that is the easiest to understand (at least for many people). In contrast to other loop structures, in which the portions that control the loop are scattered throughout the programme, this one's loop control components are centralised in a single location. This makes it much simpler to understand how this structure's loops operate.

A section of code can be made to be repeated an unlimited number of times by using the for loop. It is utilised somewhat frequently (though not always) in situations in which the iteration count of the loop is known before the iteration itself is started.

```
#include <iostream>
using namespace std;
int main()
{
int j; //define a loop variable
for(j=0; j<15; j++) //loop from 0 to 14,
cout << j * j << " "; //displaying the square of j
cout << endl;
return 0;
}
```

Here's the output:

```
0 1 4 9 16 25 36 49 64 81 100 121 144 169 196
```

For statement controls the loop. It consists of the keyword for, followed by parentheses that contain three expressions separated by semicolons:

for(j=0; j<15; j++)

These three expressions all use the same variable, which we will refer to as the loop variable. This variable is shared by all three expressions. The value of j is used as the loop variable in the example. It is given a definition before the statements that make up the loop body begin to execute.

The body of the loop is the portion of the programme code that will be executed each time the loop is repeated. The purpose of the loop is to execute this code multiple times. The body of the loop in this example consists of just a single statement: cout j * j " "; The square of j is displayed after this statement, followed by two spaces. The square can be obtained by multiplying j by itself. As the loop is executed, the variable j iteratively moves through the numbers 0 through 14, displaying the squares of the numbers 0 through 1, 4, 9, and so on up to 196 as it goes.

- **MULTIPLE STATEMENTS IN THE LOOP BODY**

Naturally, you could wish to make advantage of the loop body to carry out a number of different statements. The use of braces, much like the use of functions, is to separate several statements. There are semicolons placed after each of the statements that make up the loop body; however, there is not a semicolon placed after the closing brace of the loop body.

Let's See an Example:

```
#include <iostream>
#include <iomanip> //for setw
using namespace std;
int main()
{
int numb;                              //define loop variable
for(numb=1; numb<=10; numb++)   //loop from 1 to 10
```

```
{
cout << setw(4) << numb;              //display 1st column
int cube = numb*numb*numb;            //calculate cube
cout << setw(6) << cube << endl;      //display 2nd column
}
return 0;
}
```

Here's the output from the program:

```
1 1
2 8
3 27
4 64
5 125
6 216
7 343
8 512
9 729
10 1000
```

- **FOR LOOP VARIATIONS**

It is not necessary to use the increment expression in order to increment the loop variable; the variable can be used for anything else the programme chooses.

In the following illustration, the value of the loop variable is decreased. An example of such a programme is FACTOR, which prompts the user to enter a number before calculating that number's factorial. (To find the factorial of a number, multiply the original number by all positive integers that are smaller than the number itself.

The factorial of 5 is 120, which can be written as (5*4*3*2*1)

```
#include <iostream>
using namespace std;
int main()
{
unsigned int numb;
unsigned long fact=1;          //long for larger numbers
cout << "Enter a number: ";
cin >> numb;                   //get number
for(int j=numb; j>0; j--)              //multiply 1 by
fact *= j;                     //numb, numb-1, ..., 2, 1
cout << "Factorial is " << fact << endl;
return 0;
}
```

In this scenario, the initialization statement assigns the value that was entered by the user to the variable j. The test expression will continue to trigger the loop to execute as long as the value of j is greater than 0.

The increment statement has the effect of reducing j at the end of each iteration.

We picked the unsigned long data type for the factorial since factorials can exist for even very small numbers, which are really rather large. int and long are the same thing in 32-bit operating systems like Windows; however, long gives you more customization possibilities.

The capacity of 16-bit systems has recently increased. Even if the numbers entered are integers, the output seen below highlights how enormous factorials are small:

```
Enter a number: 10
Factorial is 3628800
```

11.3 WHILE LOOP

A predetermined number of times can be iterated through using the for loop. What should you do in this situation if you don't know how many times you want to perform a task before you start the loop?

In this particular instance, a different kind of loop, known as the while loop, could be utilised.

In the following illustration, the user will be prompted to enter a number sequence. When the user enters the number 0, the loop will finish. It's important to keep in mind that the computer software has no way of predicting how many digits will be entered before it gets to the zero.

```
#include <iostream>
using namespace std;
int main()
{
int n = 99;                    // make sure n isn't initialized to 0
while( n != 0 )                // loop until n is 0
cin >> n;                      // read a number into n
cout << endl;
return 0;
}
```

Here's some sample output. The user enters numbers, and the loop continues until 0 is entered,
at which point the loop and the program terminate.

```
1
27
33
144
9
0
```

- **MULTIPLE STATEMENTS IN A WHILE LOOP**

The following illustration shows how to use a while loop with many statements inside of it. It is a for looping version of the CUBE programme, but rather than calculating the cube, it calculates the fourth power of a number of numbers in a sequence.

Let's imagine that displaying the outcomes of this programme in a column that has four digits is absolutely necessary. In order to ensure that the results are compatible with the width of this column, we need to exit the loop before the count reaches 9999.

Because we do not have access to earlier calculations, we are unable to determine what number will lead to a result of this magnitude; therefore, we let the software to determine it for us. If the test phrase determines that the powers are at an unsafe level, the while statement will prevent the programme from continuing to run.

```
#include <iostream>
#include <iomanip>              //for setw
using namespace std;
int main()
{
int pow=1;                      //power initially 1
int numb=1;                     //numb goes from 1 to ???
while( pow<10000 )              //loop while power <= 4 digits
{
cout << setw(2) << numb;                //display number
cout << setw(5) << pow << endl;                 //display     fourth
power
++numb;                                         //get    ready    for
next power
pow = numb*numb*numb*numb;                      //calculate    fourth
power
}
```

```
cout << endl;
return 0;
}
```

We simply multiply numb by itself four times to get the fourth power. We increment numb each time we cycle through the loop. In while, however, we don't utilise numb in the test expression; instead, the result of pow determines when the loop should be terminated.

Here's the output:

```
1 1
2 16
3 81
4 256
5 625
6 1296
7 2401
8 4096
```

11.4 DO LOOP

When using a while loop, the test expression is evaluated right at the beginning of the loop. In the event that the test expression is found to be untrue when the loop is entered, the loop body will not be carried out. In certain circumstances, this is precisely what you had in mind for the outcome.

However, there are some circumstances in which you want to make sure that the body of the loop is executed at least once, independent of the starting condition being tested for by the test expression. As the test expression will be executed at the conclusion of the do loop, this particular circumstance calls for its utilisation.

```
#include <iostream>
using namespace std;
int main()
{
long dividend, divisor;
char ch;
do                                          //start of do loop
{                                           //do          some
processing
cout << "Enter dividend: "; cin >> dividend;
cout << "Enter divisor: "; cin >> divisor;
cout << "Quotient is " << dividend / divisor;
cout << ", remainder is " << dividend % divisor;
cout << "\nDo another? (y/n): ";
//do it again?
cin >> ch;
}
while( ch != 'n' );                                 //loop
condition
return 0;
}
```

Here is the output:

```
Enter dividend: 11
Enter divisor: 3
Quotient is 3, remainder is 2
Do another? (y/n): y
Enter dividend: 222
Enter divisor: 17
Quotient is 13, remainder is 1
Do another? (y/n): n
```

11.5 WHEN TO USE WHICH LOOP

Several generalisations about the operation of loops have already been discussed here. The for loop is appropriate to use when it is already known in advance how many times the loop will be carried out.

Utilize the while and do loops when you are unsure of when the loop will complete (use the while loop when you are unsure of whether or not you want to execute the body of the loop even once, and use the do loop when you are positive that you want to run the body of the loop at least once).

These criteria are somewhat subject to personal interpretation. The choice of which loop type to use is less a matter of adhering to a predetermined set of rules and more of a matter of personal preference. Practically any scenario may be adapted to work with any of the possible loop structures.

Pick the style that will make your software the most comprehensible to the broadest possible audience

11.6 SUMMARY

There are three distinct kinds of loops that can be used in C++. The for loop is most commonly utilised in situations in which the desired number of iterations of the loop may be determined in advance. The while loop and the do loop are utilised whenever the condition that triggers the end of the loop occurs while the loop is still active.

The do loop will always run at least once, but it may not run at all depending on the circumstances. The "body" of a loop can either be a single statement or a collection of statements that are delineated from one another using braces. Only the variables that are defined inside of a block are visible to code that is written outside of that block.

11.7 KEYWORDS:

For

while

do

11.8 REVIEW QUESTIONS

1. Name and describe the usual purpose of three expressions in a for statement.

2. In a for loop with a multi statement loop body, semicolons should appear following

   a. the for statement itself.

   b. the closing brace in a multi statement loop body.

   c. each statement within the loop body.

   d. the test expression.

3. True or false: The increment expression in a for loop can decrement the loop variable.

4. Write a for loop that displays the numbers from 100 to 110.

5. A block of code is delimited by _____.

6. A variable defined within a block is visible

   a. from the point of definition onward in the program.

   b. from the point of definition onward in the function.

   c. from the point of definition onward in the block.

   d. throughout the function.

10. Write a while loop that displays the numbers from 100 to 110.

11. Use for loops to construct a program that displays a pyramid of Xs on the screen. The
pyramid should look like this

X

XXX

XXXXX

XXXXXXX

XXXXXXXXX

except that it should be 20 lines high, instead of the 5 lines shown here. One way to do
this is to nest two inner loops, one to print spaces and one to print Xs, inside an outer
loop that steps down the screen from line to line.

11.9 FURTHER READING

Fundamental Computer Skills, Feng-Qi Lai, David R. Hofmeister.

A Study of Computer Fundamental, Shu-Jung Yi.

http://books.google.co.in/bkshp?hl=en

UNIT 12: ARRAYS AND STRINGS

CONTENTS

OBJECTIVES:

- Understanding the basic concepts of array in Programming
- Learn about Array elements and settings up arrays
- Learn about Multidimensional array
- Learn about Strings

12.1. INTRODUCTION

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

12.2. WHY WE REQUIRE ARRAY?

In our day-to-day lives, we routinely organise things that are connected into units. We purchase canned peas and eggs in cartons most of the time. In computer languages, we are also required to group data objects that are of the same type. When it comes to reaching this goal in C++, the most fundamental method is the use of arrays. The amount of data that can be stored in an array can range anywhere from a few to tens of thousands. An array's constituent data components can be of a simple type, such as an integer or a floating-point value, or of a user-defined type, such as a structure or an object. A straightforward example programme will be utilised so that arrays may be illustrated. This piece of code will produce a four-integer array that will represent the ages of four different people. After that, the user will be prompted to enter four values, all of which will eventually be saved in the array. At long last, each of the four values is revealed.

```
#include <iostream>
using namespace std;
int main()
{
int age[4];                 //array 'age' of 4 ints
for(int j=0; j<4; j++)   //get 4 ages
{
cout << "Enter an age: ";
cin >> age[j];                    //access array element
}
for(j=0; j<4; j++)                        //display 4 ages
cout << "You entered " << age[j] << endl;
return 0;
}
```

Here's a sample interaction with the program:

```
Enter an age: 44
Enter an age: 16
Enter an age: 23
Enter an age: 68
You entered 44
You entered 16
You entered 23
You entered 68
```

- **DEFINING ARRAYS**

Before an array can be used to store data in C++, it must be defined, much like other variables. An array definition, like all other definitions, specifies a variable type and a name. It does, however, have another feature: a size. The size of the array determines how many data items it will hold. It comes after the name and is encircled by square brackets.

The array in the preceding example is of type int. Following that is the array's name, followed by an opening bracket, the array size, and a closing bracket. The number in brackets must be an integer and must be a constant or an expression that evaluates to a constant. The value 4 is used in this example.

- **ELEMENTS IN AN ARRAY**

Elements are the items in an array (in contrast to the items in a structure, which are called members). As previously stated, all of the items in an array are of the same type; the only difference is the values.

The initial array entry is numbered 0 as you can see. Since there are four elements, the third is the last. This is a potentially perplexing circumstance; you may expect the fourth member in a four-element array to be the fourth element, but it isn't.

- **ACCESSING THE ELEMENTS OF AN ARRAY**

Each array entry is accessed twice in the example above. The line cin >> age[j]; is used to insert a value into the array for the first time.

We read it out the second time with the line

cout<< "\nYou entered " age[j];

The expression for the array element is always age[j], regardless of the context.

This consists of the name of the array followed by braces that separate a variable that is referred to as j. The value of j is what defines which of the four array elements this expression specifies; age[0] corresponds to the first element, age[1] to the second element, age[2] to the third element, and age[3] to the fourth element. age[4] relates to the last element. The variable (or constant) that is included in brackets serves as the index of the array.

Because j is the loop variable in each of the for loops, it begins at 0 and continues to be incremented until it reaches 3, which enables you to access each element of the array on an individual basis.

12.3. ARRAY ELEMENTS AVERAGED

One more illustration of how an array can be used. The user of this one is prompted to provide a string of six values representing widget sales for each day of the week (with the exception of Sunday), after which the average is computed. In order to accept entries for monetary quantities, we make use of a double-type array in our system.

```
#include <iostream>
using namespace std;
int main()
{
const int SIZE = 6;                    //size of array
double sales[SIZE];                    //array of 6 variables
cout << "Enter widget sales for 6 days\n";
for(int j=0; j<SIZE; j++)                        //put figures in array
cin >> sales[j];
double total = 0;
for(j=0; j<SIZE; j++) //read figures from array
total += sales[j]; //to find total
double average = total / SIZE; // find average
cout << "Average = " << average << endl;
return 0;
}
Enter widget sales for 6 days
352.64
867.70
781.32
867.35
746.21
189.45
Average = 634.11
```

12.4. SETTING UP ARRAYS

When the array is first defined, you will have the opportunity to assign a value to each individual element. Here is an example, DAYS, which adjusts the number of array elements in the days per month array to reflect the new value of the number of days in each month.

```
#include <iostream>
using namespace std;
int main()
{
int month, day, total_days;
int days_per_month[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
30, 31 };
cout << "\nEnter month (1 to 12): ";
                    //get date
cin >> month;
cout << "Enter day (1 to 31): ";
cin >> day;
total_days = day; //separate days
for(int j=0; j<month-1; j++) //add days each month
total_days += days_per_month[j];
cout << "Total days from start of year is: " << total_days
<< endl;
return 0;
}
```

The user can input a date, and the application will calculate the number of days that have passed since the beginning of the year. (It is important to note that this method will not work for leap years.) The following is an illustration of a conversation:

Please select the month from 1 to 12: 3

Please enter the day from 1 to 31: 11

70 days have passed since the first of the year.

After retrieving the information for the month and the day, the software initially assigns the value for the day to the total days variable.

After that, it enters a loop that adds values to total days by pulling them from an array that contains the number of days in each month. When compared to the total number of months, there is one fewer such number to add. If the user chooses month 5, the values of the first four array items (31, 28, 31, and 30) are added to the sum of the selections made.

12.5. ARRAYS WITH MULTIPLE DIMENSIONS

To this point, we have only examined one-dimensional arrays, which are as follows: A single variable is used to specify each individual element of an array. Arrays, on the other hand, have the ability to contain more than one dimension. A two-dimensional array is utilised by the SALEMON programme in order to keep sales data for a number of different regions and different months.

```
#include <iostream>
#include <iomanip>              //for setprecision, etc.
using namespace std;
const int DISTRICTS = 4;                //array dimensions
const int MONTHS = 3;
int main()
{
int d, m;
double sales[DISTRICTS][MONTHS];          //two-dimensional
array
cout << endl;
for(d=0; d<DISTRICTS; d++)                    //get array values
for(m=0; m<MONTHS; m++)
{
cout << "Enter sales for district " << d+1;
cout << ", month " << m+1 << ": ";
cin >> sales[d][m];                          //put number in array
}
```

```
cout << "\n\n";
cout << " Month\n";
cout << " 1 2 3";
for(d=0; d<DISTRICTS; d++)
{
cout <<"\nDistrict " << d+1;
for(m=0; m<MONTHS; m++)                    //display       array
values
cout << setiosflags(ios::fixed)            //not exponential
<< setiosflags(ios::showpoint)                    //always
use point
<< setprecision(2)                         //digits to right
<< setw(10)                                //field width
<< sales[d][m];                            //get number from
array
}                                    //end for(d)
```

12.6. DEFINING MULTIDIMENSIONAL ARRAYS

The array is defined by utilising two size specifiers, each of which is surrounded by brackets:

double sales[DISTRICTS][MONTHS];

Think of the revenue generated by sales as a checkerboard-patterned array in two dimensions. One more way of looking at it is to consider sales to be a collection of other collections. It is a collection of elements known as DISTRICTS, each of which is itself a collection of elements known as MONTHS.

Arrays with more than two dimensions are certainly not impossible to construct. A three-dimensional array is one that contains arrays of arrays of arrays.

12.7. STRINGS

Similar to other types of data, strings can operate as either variables or constants. First, we'll take a look at these two entities, and only then will we move on to more advanced string operations. The following is an illustration of the definition of a single string variable. It will

ask the user for a string, and then it will save that string in the variable called string. After that, the string will be seen.

```
#include <iostream>
using namespace std;
int main()
{
const int MAX = 80;              //max characters in string
char str[MAX];                   //string variable str
cout << "Enter a string: ";
cin >> str;                 //put string in str
        //display string from str
cout << "You entered: " << str << endl;
return 0;
}
```

The following is an example of how the string variable str can be defined, which is also an example of how an array of type char can be defined:

char str[MAX]; To read a string from the keyboard and save it in the string variable str, we use the extraction operator >>. MAX denotes the maximum length of the string. This operator is familiar with working with strings, and it recognises that strings are really arrays of character values.

Each character requires one byte of RAM to be stored in. C-strings are required to finish with a byte that contains the value 0. This is a feature that should not be overlooked. This is typically represented by the character constant '0,' which has a value of 0 according to the ASCII character set. The zero that comes at the very end of a string is known as the null character. When showing the string, the operator displays characters until it hits the null character, at which point it stops displaying characters.

12.8. STRING CONSTANTS

You can initialize a string to a constant value when you define it. Here's an example, that does just that

```
#include <iostream>
using namespace std;
int main()
{
char str[] = “Farewell! thou art too dear for my possessing.”;
cout << str << endl;
return 0;
}
```

The string constant is formatted like a sentence in regular English, with quotes between each element of the string.

This could come as a shock to you given that a string is actually an array of type char. Arrays used to be initialised with a list of values that were delimited by brackets and separated by commas when they were first created.

Why isn't the str variable initialised in the same way as the other variables are? In point of fact, the following list of character constants might be utilised:

char str[] = { ‘F’, ‘a’, ‘r’, ‘e’, ‘w’, ‘e’, ‘l’, ‘l’, ‘!’,’ ‘, ‘t’, ‘h’,}

and so forth. Fortunately, the designers of C++ (and C) saw our plight and created the above-mentioned shortcut approach. The result is identical: The characters are arranged in the array sequentially.

12.9. READING EMBEDDED BLANK

You might have gotten an unpleasant surprise if you used the above software with strings that comprised more than one word. Here's an illustration:

```
Enter a string: Law is a bottomless pit.
You entered: Law
```

What happened to the rest of the phrase?

It turns out that a space is treated as a terminating character by the extraction operator >>. As a result, it will read single-word strings, but anything typed following a space will be ignored.

Another function, cin.get, is used to read text with blanks ().

This syntax refers to the stream class's get() member method, of which cin is an instance. Consider the following scenario:

```
#include <iostream>
using namespace std;
int main()
{
const int MAX = 80;              //max characters in string
char str[MAX];                   //string variable str
cout << "\nEnter a string: ";
cin.get(str, MAX);               //put string in str
cout << "You entered: " << str << endl;
return 0;
}
```

Using this function, the input string is now stored in its entirety.

```
Enter a string: Law is a bottomless pit.
You entered: Law is a bottomless pit.
```

- **READING MULTIPLE LINES**

It's possible that we've figured out how to read strings that contain blanks embedded inside them, but what about strings that span multiple lines?

It has been discovered that the cin::get() function is able to accept a third argument, which will be of use in the current scenario.

This argument allows the reader to choose the text character that will cause the function to terminate reading.

If you call the function with a different character for this parameter, the supplied character will take precedence over the default, which is the newline ('n') character. However, if you leave this argument blank, the default will be the newline character.

```
#include <iostream>
using namespace std;
const int MAX = 2000; //max characters in string
char str[MAX]; //string variable str
int main()
{
cout << "\nEnter a string:\n";
cin.get(str, MAX, '$'); //terminate with $
cout << "You entered:\n" << str << endl;
return 0;
}
Enter a string:
Ask me no more where Jove bestows
When June is past, the fading rose;
For in your beauty's orient deep
These flowers, as in their causes, sleep.


$
You entered:
Ask me no more where Jove bestows
When June is past, the fading rose;
For in your beauty's orient deep
These flowers, as in their causes, sleep.
We terminate each line with Enter, but the program continues to
accept input until we enter '$'.
```

- **THE DIFFICULTY OF COPYING A STRING**

Dealing with strings character by character is the best method to grasp their fundamental nature. This is accomplished using the application below.

```
#include <iostream>
#include <cstring>              //for strlen()
using namespace std;
int main()
{                               //initialized string
char str1[] = "Oh, Captain, my Captain! "
              "our fearful trip is done";
const int MAX = 80;             //size of str2 buffer
char str2[MAX];                 //empty string
for(int j=0; j<strlen(str1); j++)   //copy strlen
characters
str2[j] = str1[j];              // from str1 to str2
str2[j] = '\0';                 //insert NULL at end
cout << str2 << endl;           //display str2
return 0;
}
```

This programme generates a string variable with the name str2, in addition to a string constant with the name str1. After that, a for loop is utilised in order to copy the string constant into the string variable. The statement is copied one character at a time using the method described above.

str2[j] = str1[j];

This application will also instruct you on how to use the functions found in the C-string library. Since C++ does not have any built-in string operators, C-strings often need to be handled through the use of library functions.

To our good fortune, there are a great many of them. This piece of software makes use of the strlen() function, which determines the length of a cstring (that is, how many characters are in it). The for loop makes advantage of this length as its limit, which guarantees that the appropriate number of characters are copied. Whenever string functions are required, the header file CSTRING (or STRING.H), whichever is appropriate, must be incorporated into the programme using the #include directive.

```
#include <iostream>
#include <cstring> //for strcpy()
using namespace std;
int main()
{
char str1[] = "Tiger, tiger, burning bright\n"
"In the forests of the night";
const int MAX = 80; //size of str2 buffer
char str2[MAX]; //empty string
strcpy(str2, str1); //copy str1 to str2
cout << str2 << endl; //display str2
return 0;
}
```

Note that you call this function with the destination first:

strcpy(destination, source)

The right-to-left order is reminiscent of the format of normal assignment statements: The variable on the right is copied to the variable on the left.

12.10. SUMMARY

Collections of data components that are all of the same kind can be referred to as arrays. This type can be represented by a basic data type, a structure, or a class at the same time. The components that go into an array are referred to as its elements. Accessing elements requires using a number, which is why this number is referred to as an index.

After the array has been formed, the individual items can have the values that you want assigned to them. Multiple dimensions are conceivable in arrays. A two-dimensional array is an array that contains other arrays. Arrays of char-type are what C-strings are. A C-string must always end with the character '0', often known as the null character. C-string constants have a special format, which consists of the text being surrounded by double quotes. This is done to make writing them easier. A number of different library functions are called upon to perform manipulation on C-strings.

A collection of character arrays is known as an array of C-strings.

The creator of a C-string variable is responsible for ensuring that the array is sufficiently large to store any text that may be inserted into the variable. C-strings are a type of string that can be used as an argument in C-style library methods. Older applications will contain C-strings. Generally speaking, their implementation is not recommended for use in newly developed programmes.

12.11. KEYWORDS

Array

String,

Elements

Multiple Dimension

12.12. REVIEW QUESTIONS

1. An array element is accessed using
    a. a first-in-first-out approach.
    b. the dot operator.
    c. a member name.
    d. an index number.
2. All the elements in an array must be the _____ data type.
3. Write a statement that defines a one-dimensional array called double Array of type double that holds 100 elements.
4. The elements of a 10-element array are numbered from _____ to _____.
5. Write a statement that takes element j of array doubleArray and writes it to cout with the insertion operator.

6. Element doubleArray[7] is which element of the array?

a. The sixth

b. The seventh

c. The eighth

d. Impossible to tell

7. Start with a program that allows the user to input a number of integers, and then stores them in an int array. Write a function called maxint() that goes through the array, element by element, looking for the largest one. The function should take as arguments the address of the array and the number of elements in it and return the index number of the largest element. The program should call this function and then display the largest element and its index number.

12.13. FURTHER READING

Fundamental Computer Skills, Feng-Qi Lai, David R. Hofmeister.

A Study of Computer Fundamental, Shu-Jung Yi.

http://books.google.co.in/bkshp?hl=en

UNIT 13: FUNCTIONS

CONTENTS

OBJECTIVES:

- Understanding the basic concepts of function in Programming
- Learn about recursion

13.1. INTRODUCTION

Utilizing functions can help lower the overall size of your application, which is still another advantage of doing so. Any group of instructions that occurs more than once inside the context of a programme is a potential candidate for becoming a function. The code for the function is only ever stored in a single location in memory, despite the fact that it is used numerous times during the course of the programme.

13.2. WHAT IS FUNCTION?

A unit of programming that is given a name and contains many statements of the programme is called a function. After that, the component can be accessed from any point within the application. The most important reason to use functions in a programme is so that they can assist with the conceptual organisation of the programme.

- **SIMPLE FUNCTIONS**

Our first example demonstrates how to produce a line with 45 asterisks by making use of a very straightforward function. The example software produces a table, and the table is then formatted with asterisk lines in order to make the table more legible.

```
#include <iostream>
using namespace std;
void starline();          //function declaration
                          // (prototype)
int main()
{
starline();                              //call to function
cout << "Data type Range" << endl;
starline();                              //call to function
cout << "char -128 to 127" << endl
<< "short -32,768 to 32,767" << endl
<< "int System dependent" << endl
<< "long -2,147,483,648 to 2,147,483,647" << endl;
```

```
starline();                                    //call to function
return 0;
}
// function definition
void starline() //function declarator
{
for(int j=0; j<45; j++) //function body
cout << "*";
cout << endl;
}
```

The output from the program looks like this:

```
*********************************************
Data type Range
*********************************************
char -128 to 127
short -32,768 to 32,767
int System dependent
long -2,147,483,648 to 2,147,483,647
```

There are two functions in the programme: main() and starline() (). You've probably seen a lot of apps that only utilise main(). What are the additional components required to add a function to the programme? The function declaration, the function calls, and the function definition are the three.

- **FUNCTION DECLARATION**

It is impossible to utilise a function or a variable without first explaining their purpose to the compiler, just as it is impossible to use a function without first explaining its purpose to the compiler. There are two different ways that this can be accomplished. Declaring the function before actually using it is the strategy that we employ in this situation. (Another option is to define it before we call it; we'll discuss that in the following section.) Within the TABLE programming, the starline() method is denoted as being declared on the line

void starline()

The compiler is informed by the declaration that a function with the name starline will be presented at a later stage in the process. The function does not provide a return value, as indicated by the void keyword, and it does not take any arguments, as shown by the absence of parentheses around the function name. In C, you can also use the keyword void in parentheses to indicate that the function does not take any arguments. However, in C++, it is more common to simply leave the parentheses empty. In the future, we are going to have more conversation regarding return values and arguments.

It is important to take note that the declaration of the function is completed with a semicolon. Function declarations are also referred to as prototypes due to the fact that they provide a model or blueprint for the function to be declared. They tell the compiler that "a function that looks like this is coming up later in the programme," and that "it's fine if you see references to it before you see the function itself" because "a function that looks like this is coming up later in the programme." The information contained in the declaration is referred to as the function signature. This information includes the return type of the number as well as the types of any parameters.

- **CALLING THE FUNCTION**

The function is called (also known as invoked, invoked, and executed) three times from the main (). The following description applies to each of the three calls:

starline();

The only thing that is required to call a function is the function's name, followed by parentheses. The syntax for the call is quite similar to the syntax for the declaration; the only difference is that the return type is not used in the call. The end of the call is indicated by a semicolon.

- **FUNCTION DEFINITION**

At long last, we have arrived at the function itself, also referred to as the definition of the function. The definition of the function includes the actual code for the function. The following is an example of starline format ()

```
void starline()                 //declarator
{
for(int j=0; j<45; j++)         //function body
cout << '*';
cout << endl;
}
```

Both the line that begins the definition and the line that immediately follows it are referred to as the declarator and the function body, respectively.

Statements are what make up the function body, and braces are what divide them from one another to keep them from overlapping and becoming confusing.

The declarator and the declaration have to be identical in the following respects: the declarator has to accept the same types of arguments in the same sequence (if there are any), and the declarator has to return the same type.

Additionally, the declarator needs to have the same function name as the declaration.

- **THE STEP-BY-STEP PROCEDURE FOR PASSING ARGUMENTS TO FUNCTIONS**

Let's pretend for a moment that we've reached the decision that the starline() function, which was utilised in the prior example, doesn't offer enough room for customization. We would love to have a function that will print any letter any number of times rather than one that will always write 45 asterisks, so we may choose how many times each letter is printed.

```
#include <iostream>
using namespace std;
void repchar(char, int);                    //function declaration
int main()
{
repchar('-', 43);                    //call to function
cout << "Data type Range" << endl;
repchar('=', 23);                    //call to function
cout << "char -128 to 127" << endl
<< "short -32,768 to 32,767" << endl
<< "int System dependent" << endl
<< "double -2,147,483,648 to 2,147,483,647" << endl;
repchar('-', 43);                    //call to function
return 0;
}
```

```
// function definition
void repchar(char ch, int n)            //function declarator
{
for(int j=0; j<n; j++)                  //function body
cout << ch;
cout << endl; }
```

Here is the output:

```
----------------------------------------
Data type Range
======================
char -128 to 127
short -32,768 to 32,767
int System dependent
long -2,147,483,648 to 2,147,483,647
```

In this particular instance, the arguments were constants such as '–', 43, and so on. Let's look at an example of using variables as arguments rather than constants so that we can better understand this concept.

This programme makes use of the repchar() method, just like the one that came before it, but it gives the user the ability to select the character that should be repeated as well as the number of times it should be repeated.

The Process of Returning Values from Functions

When a function is finished, it can hand back one value to the programme that called it.

This value is known as the return value. In most cases, the solution to the issue that was addressed by the function can be found inside this value that is returned. The following code demonstrates a function that accepts a weight in pounds as an input and returns the value in kilogrammes for that weight.

```
#include <iostream>
using namespace std;
float lbstokg(float);                    //declaration
int main()
{
float lbs, kgs;
cout << "\nEnter your weight in pounds: ";
cin >> lbs;
kgs = lbstokg(lbs);
cout << "Your weight in kilograms is " << kgs << endl;
return 0;
}
// converts pounds to kilograms
float lbstokg(float pounds)
{
float kilograms = 0.453592 * pounds;
return kilograms;
}
The output
Enter your weight in pounds: 182
Your weight in kilograms is 82.553741
```

When a function returns a value, the data type of that value must be specified before the function can be called again. In both the declaration and the definition, the data type, which is a float in this illustration, comes before the name of the function. The preceding software samples had functions that did not return any value; as a result, the return type was void.

- **THE RETURN STATEMENT**

The parameter for the function lbstokg() is set to the value that is passed to it in the form of a weight in pounds. This amount in pounds is multiplied by a constant to obtain the corresponding weight in kilogrammes, which is then stored in the variable kilos. The value of this variable is then sent back to the programme that initially called the function using a return statement:

return kilo;

13.3. RECURSION

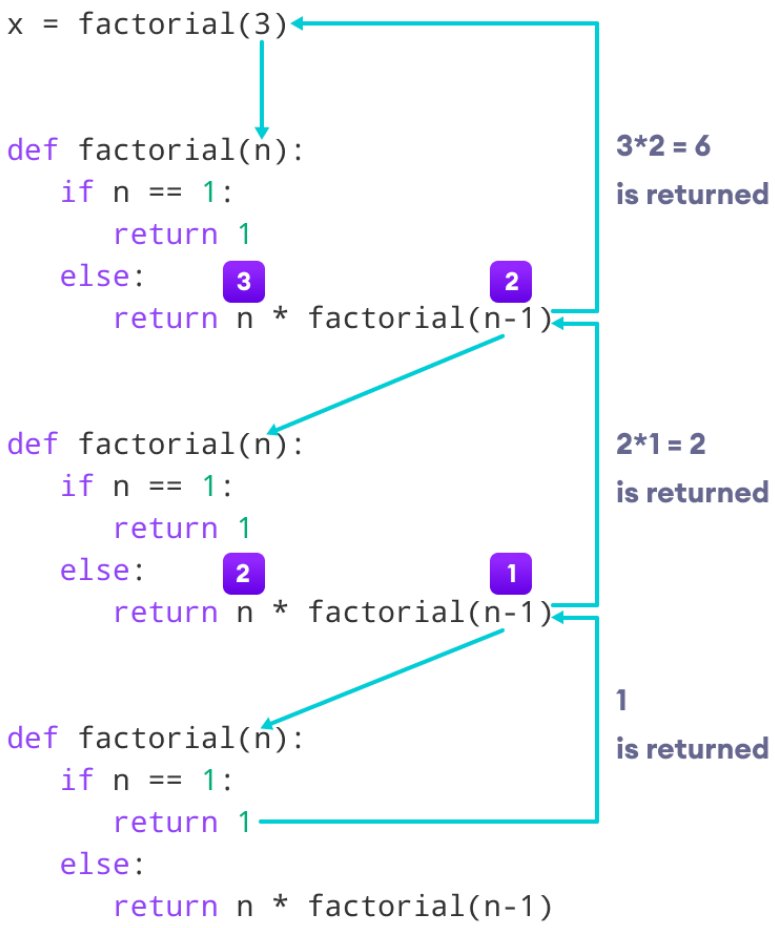
Recursion is a programming method made feasible by the existence of functions. A recursive function is one that calls itself. This appears improbable because a function calling itself is frequently a problem. When utilised appropriately, though, this strategy can be very effective.

Let's add recursion to a programme we've seen before: the Factorial of a Number, in which the factorial of a number was calculated using a for loop.

```
#include <iostream>
using namespace std;
unsigned long factfunc(unsigned long);
//declaration
int main()
{
int n;                                  //number entered by user
unsigned long fact;                     //factorial
cout << "Enter an integer: ";
cin >> n;
fact = factfunc(n);
cout << "Factorial of " << n << " is " << fact << endl;
return 0;
}
unsigned long factfunc(unsigned long n)
{
if(n > 1)
return n * factfunc(n-1);          //self call
else
return 1;
}
```

The output of the program:

```
Enter an integer:3
Factorial of 5 is 6
```



13.4. SUMMARY

- This part offers a comprehensive insight on a number of subjects, such as functions and recursions, among others.

13.5. KEYWORDS

Function

Recursion

13.6. REVIEW QUESTIONS

1. Write a recursive function that takes a number and returns the sum of all the numbers from zero to that number.

2. Write a recursive function that takes a number as an input and returns the factorial of that number.

3. Write a recursive function that takes a number ‘n’ and returns the nth number of the Fibonacci number.

4. Write a recursive function that takes a list of numbers as an input and returns the product of all the numbers in the list.

5. Write a function that takes a string and returns if the string is a palindrome.

13.7. FURTHER READING

Fundamental Computer Skills, Feng-Qi Lai, David R. Hofmeister.

A Study of Computer Fundamental, Shu-Jung Yi.

http://books.google.co.in/bkshp?hl=en

UNIT 14: POINTER

CONTENTS

OBJECTIVES:

- Understanding the basic concepts of pointers in Programming
- Learn about pointer variables and accessing

14.1. INTRODUCTION

As was mentioned before, pointers are more commonly employed to store addresses than they are data. The declaration of pointers takes place like this.

It's remarkable for such a straightforward idea to generate so much confusion, but pointers in C++ (and C) programming are the computer equivalent of the big bad wolf. However, you shouldn't worry about it. In this chapter, we are going to make an attempt to dissect pointers and show how they might be utilised in the C++ programming language.

What exactly is the point of using pointers?

The following are some examples:

• *Getting memory from the computer's operating system*

• *Constructing data structures, such as linked lists*

• *Sending arguments to a function when that function has to make changes to the parameter they were passed in originally*

• *Providing functions with string and array arguments*

Many programming languages, like Visual Basic and Java, do not have any pointers at all, in contrast to C++ and C, which both have a large number of pointers. Is it essential to place such a significant focus on pointing out?

It is clear from their absence in the earlier chapters that it is possible to do a great deal even without them.

There is more than one way to accomplish certain C++ operations that require the use of pointers. For instance, members of an array can be accessed by using array notation rather than pointer notation, and a function can modify referenced as well as pointer-passed inputs. Arrays are also used in programming to store data.

14.2. ADDRESSES AND POINTERS

Each byte in the memory of the computer has a specific address associated with it. Addresses, like home numbers on a street, are numbers. The numbers begin at 0 and progress upwards—1, 2, 3, and so on.

- **ADDRESS-OF OPERATOR &**

The address-of operator (&) can be used to find the address occupied by a variable.

```
#include <iostream>
using namespace std;
int main()
{
int var1 = 11; //define and initialize
int var2 = 22; //three variables
int var3 = 33;
cout << &var1 << endl //print the addresses
<< &var2 << endl //of these variables
<< &var3 << endl;
return 0;
}
```

The addresses occupied by variables in a programme are determined by a variety of factors, including the computer on which the programme is executing, the size of the operating system, and whether any other programmes are currently in memory. As a result, you're unlikely to acquire the same addresses as we did when you run this software. The following is the output from our machine:

```
0x8f4ffff4 ← address of var1
0x8f4ffff2 ← address of var2
0x8f4ffff0 ← address of var3
```

14.3. POINTER VARIABLES

Addresses are relatively limited on their own. While it's convenient to know where things are in memory, printing out address values isn't very useful.

The ability to increase our programming strength necessitates the adoption of a new concept: variables that store address values.

There have been instances of variable types that may store a variety of data, including characters, integers, floating-point numbers, and others.Addresses are saved in the same way. A pointer variable, or simply a pointer, is a variable that stores an address value.

```
#include <iostream>
using namespace std;
int main()
{
int var1 = 11; //two integer variables
int var2 = 22;
cout << &var1 << endl //print addresses of variables
<< &var2 << endl << endl;
int* ptr; //pointer to integers
ptr = &var1; //pointer points to var1
cout << ptr << endl; //print pointer value
ptr = &var2; //pointer points to var2
cout << ptr << endl; //print pointer value
return 0;
}
```

This piece of code initiates the creation of two integer variables, var1 and var2, and then assigns the values 11 and 22 to them correspondingly. The addresses of those people are then printed down on a piece of paper. The next thing that the software does is establish a pointer variable in the line after that.

int* ptr;

The asterisk denotes a pointer to something. As a result, the variable ptr is defined as a pointer to int in the statement. This is another way of indicating that this variable can store integer variables' addresses.

```
The syntax used in C++ allows pointers to any type to be declared:
char* cptr; // pointer to char
int* iptr; // pointer to int
float* fptr; // pointer to float
```

14.4. ACCESSING THE VARIABLE POINTER

Imagine that we are unaware of the name of a variable but are familiar with its location. Is it feasible to access the information that is stored in the variable?

To get the value of a variable by using its address rather than its name, you'll need to use a particular syntax.

```
#include <iostream>
using namespace std;
int main()
{
int var1 = 11; //two integer variables
int var2 = 22;
int* ptr; //pointer to integers
ptr = &var1; //pointer points to var1
cout << *ptr << endl; //print contents of pointer (11)
ptr = &var2; //pointer points to var2
cout << *ptr << endl; //print contents of pointer (22)
return 0;
}
```

Here's the output:

```
11
22
```

- **POINTERS AND FUNCTIONS**

It is important to keep in mind that there are three distinct ways in which you are able to pass parameters to a function: by value, by reference, and by pointer.

It is not possible to supply the variable by value if the function is designed to modify variables in the application that called it because all that the function does is make a copy of the variable.

In this particular scenario, though, one may make use of either a pointer or a reference argument.

Passing Simple Variables:

```
#include <iostream>
using namespace std;
int main()
{
void centimize(double&); //prototype
double var = 10.0; //var has value of 10 inches
cout << "var = " << var << " inches" << endl;
centimize(var); //change var to centimeters
cout << "var = " << var << " centimeters" << endl;
return 0;
}
void centimize(double& v)
{
v *= 2.54; //v is the same as var
}
```

In this particular instance, we want to convert a variable called var in main from inches to centimetres ().

By reference, the value of the variable centimize is passed along to the function centimize().

(It is important to keep in mind that the & character that comes after the data type double in the prototype for this method indicates that the parameter is passed in by reference.)

The centimize() method results in a multiplication of the initial variable by a factor of 2.54.

Take a mental note of the references to the variable that are made throughout the code. It just refers to the argument by its shorthand name, v; the term var can be used interchangeably with v.

main() will show the result of the conversion from var to cm after it has been completed. The results are as follows:

```
var = 10 inches
var = 25.4 centimeters
```

The next example, shows an equivalent situation when pointers are used:

```
#include <iostream>
using namespace std;
int main()
{
void centimize(double*); //prototype
double var = 10.0; //var has value of 10 inches
cout << "var = " << var << " inches" << endl;
centimize(&var); //change var to centimeters
cout << "var = " << var << " centimeters" << endl;
return 0;
}
void centimize(double* ptrd)
{
*ptrd *= 2.54; //*ptrd is the same as var
}
```

Output of both the program is same.

- **PASSING ARRAYS**

When passing arrays to functions, it's more typical to use pointer notation rather than array notation.

```
#include <iostream>
using namespace std;
const int MAX = 5; //number of array elements
int main()
{
void centimize(double*); //prototype
double varray[MAX] = { 10.0, 43.1, 95.9, 59.7, 87.3 };
```

```
centimize(varray); //change elements of varray to cm
for(int j=0; j<MAX; j++) //display new array values
cout << "varray[" << j << "]="
<< varray[j] << " centimeters" << endl;
return 0;
}
void centimize(double* ptrd)
{
for(int j=0; j<MAX; j++)
*ptrd++ *= 2.54; //ptrd points to elements of varray
}
```

The prototype of the function is the same as PASSPTR, and the function accepts a pointer to double as its one and only parameter. In array notation, this is denoted by the expression void centimize(double[]);. In other words, the double* syntax is identical to the double[] syntax in this particular setting, despite the fact that the double[] syntax is used more frequently. It is not essential to apply the address operator & when the function is invoked because the address of an array is the same as the array's name.

14.5. STRINGS AS FUNCTION ARGUMENTS

The following is an illustration of a string being utilised in the role of a function parameter. The string is easily printed by using the technique, which works by reading each character in turn.

```
#include <iostream>
using namespace std;
int main()
{
void dispstr(char*); //prototype
char str[] = "Idle people have the least leisure.";
dispstr(str); //display the string
return 0;
}
```

```
void dispstr(char* ps)
{
while( *ps ) //until null character,
cout << *ps++; //print characters
cout << endl;
}
```

In the call to function dispstr, the array address str is utilised as the parameter (). Although this address is a constant, it is duplicated in dispstr since it is provided by value (). ps. This copy is a pointer The function increments ps to display the string because a pointer can be altered.

The expression *ps++ returns the string's subsequent characters. The loop repeats until the null character ('0') at the end of the string is found. The while loop ends at this point since this has the value 0, which represents false.

14.6. SUMMARY

This has been a whirlwind tour of the territory inhabited by pointers. The topics that we have discussed up to this point will serve as a basis for the examples that are presented in the remaining chapters of the book as well as for any additional research that may be conducted in the future. As, you should know by now, pointer constants are addresses, and everything in the memory of the computer has its own unique address. Finding the addresses of variables can be done with the help of the address-of operator, which is denoted by the symbol &.Pointers, which are considered variables, are used to hold the values of addresses. When referring to a pointer to something, an asterisk (*) is typically used. Because the compiler has to know what the pointer is pointing to in order to perform accurate arithmetic on it, a data type must always be specified in pointer declarations. The only exception to this rule is the void* data type. We make use of the asterisk as the dereference operator, which refers to the contents of the variable that is referred to by. This allows us to gain access to the thing that is pointed to in a new manner.

14.7. KEYWORDS

Pointers

Accessing

Function

14.8. REVIEW QUESTIONS

1. Write a statement that displays the address of the variable testvar.

2. The contents of two pointers that point to adjacent variables of type float differ by _____.

3. A pointer is

    a. the address of a variable.

    b. an indication of the variable to be accessed next.

    c. a variable for storing addresses.

    d. the data type of an address variable.

4. Write expressions for the following:

    a. The address of var

    b. The contents of the variable pointed to by var

    c. The variable var used as a reference argument

    d. The data type pointer-to-char

5. An address is a _____, while a pointer is a _____.

6. Write a definition for a variable of type pointer-to-float.

7. Pointers are useful for referring to a memory address that has no _____.

8. If a pointer testptr points to a variable testvar, write a statement that represents the contents of testvar but does not use its name.

9. An asterisk placed after a data type means _____. An asterisk placed in front of a variable name means _____.

10. The expression *test can be said to

    a. be a pointer to test.

    b. refer to the contents of test.

    c. dereference test.

    d. refer to the value of the variable pointed to by test.

11. Is the following code correct?

```
int intvar = 333;
int* intptr;
cout << *intptr;
```

12. Create a programme that takes a set of integers from the user and stores them in a float array. The software should then average

14.9. FURTHER READING

Fundamental Computer Skills, Feng-Qi Lai, David R. Hofmeister.

A Study of Computer Fundamental, Shu-Jung Yi.

http://books.google.co.in/bkshp?hl=en

UNIT 15: STRUCTURES

CONTENTS

OBJECTIVES:

- Understanding the basic concepts of structure in Programming
- Learn about initializing the member of structure

15.1. INTRODUCTION

A structure is built from a collection of its component variables. The variables that make up a structure can be of any number of different kinds, including ints, floats, and so on. (This is in contrast to the array, which we will talk about previously and in which all of the variables are required to have the same type.) The individual data components that comprise a structure are referred to as the structure's members.

To get started, create a structure that contains three variables: two integers and a value with a floating-point representation. This structure represents a part that is kept in the parts inventory of a company that manufactures widgets. In the sense that it outlines the information that must be present for a single component, the structure functions similarly to a blueprint. The model number of the widget is the first component of the structure. This is due to the fact that the company manufactures a wide variety of widgets. The member that comes after it is the part's number, and the member that comes after that is the price of the part.

```
#include <iostream>
using namespace std;
struct part //declare a structure
{
int modelnumber; //ID number of widget
int partnumber; //ID number of widget part
float cost; //cost of part
};
int main()
{
part part1; //define a structure variable
part1.modelnumber = 6244; //give values to structure members
part1.partnumber = 373;
part1.cost = 217.55F;
//display structure members
cout << "Model " << part1.modelnumber;
cout << ", part " << part1.partnumber;
cout << ", costs $" << part1.cost << endl;
return 0;
}
```

The program's output looks like this:

```
Model 6244, part 373, costs $217.55
```

15.2. DEFINING THE STRUCTURE

The structure definition describes the organisation of the structure: It determines who will be a part of the structure. It's as follows:

```
struct part
{
int modelnumber;
int partnumber;
float cost;
};
```

- **THE STRUCTURE DEFINITION SYNTAX**

The structure definition is introduced by the keyword struct. The structural name or tag, which is part, comes next. Braces surround the declarations of the structural members—modelnumber, partnumber, and cost. The closing brace is followed by a semicolon, which marks the end of the structure.

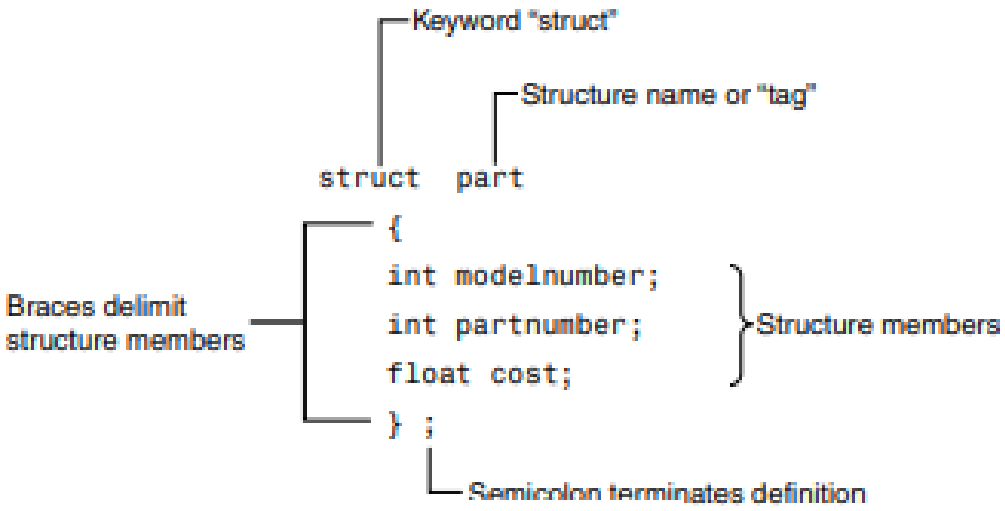


Fig. 15.1

- **IMPLEMENTATION OF THE STRUCTURE DEFINITION**

The only purpose served by the structure definition is to serve as a template for the creation of variables of type part. It does not construct any structural variables of its own accord, which means that it does not reserve any memory and does not even name any variables. On the other hand, when you define a basic variable, you do not need to take memory into account. A structure specification is comprised of nothing more than a description of the form that structure variables will take when they are put to use.

- **DEFINING A VARIABLE IN A STRUCTURE**

The first statement in main()

part part1;

declares a variable of type structure part named part1. This definition makes memory space available for component 1. How much room do you have? Enough to hold all of the constituents of part 1—modelnumber, partnumber, and cost—in one place. In this example, each of the two ints will be 4 bytes long (assuming a 32-bit architecture) and the float will be 4 bytes long.

In certain aspects, the part structure can be viewed as the specification for a new data type. This will become clearer as we progress, but keep in mind that the format for declaring a structural variable is the same as for defining a fundamental built-in data type like int:

part part1;

int var1;

- **MEMBERS OF THE STRUCTURE CAN BE ACCESSED**

After a structure variable has been defined, the. operator can be utilised to gain access to the members of the structure variable. The following is an example of how a value can be assigned to the first member:

part1.modelnumber = 6244;

The member of the structure is made up of three parts: the first part is the name of the structure variable, the second part is the dot operator (.), and the third part is the member name (modelnumber). This refers to "part1's modelnumber member." The member access operator

is the actual name of the dot operator, although almost no one uses such a lengthy term. "part1's modelnumber member" is a more common abbreviation.

15.3. INITIALIZING THE MEMBERS OF THE STRUCTURE

Next the definition of the structure variable, the following example demonstrates how the structure's members can be initialised. In addition to this, it demonstrates that a certain type of structure can have multiple variables.

```
#include <iostream>
using namespace std;
struct part //specify a structure
{
int modelnumber; //ID number of widget
int partnumber; //ID number of widget part
float cost; //cost of part
};
int main()
{ //initialize variable
part part1 = { 6244, 373, 217.55F };
part part2; //define variable
//display first variable
cout << "Model " << part1.modelnumber;
cout << ", part " << part1.partnumber;
cout << ", costs $" << part1.cost << endl;
part2 = part1; //assign first variable to second
//display second variable
cout << "Model " << part2.modelnumber;
cout << ", part " << part2.partnumber;
cout << ", costs $" << part2.cost << endl;
return 0;
}
```

This programme defines two part1 and part2 variables of type part.
It prints out the values of part1's members, assigns part1 to part2,
and prints out the members of part2.
The output:

```
Model 6244, part 373, costs $217.55
Model 6244, part 373, costs $217.55
```

15.4. SUMMARY

Structures are an essential component of C++, and one of the reasons for this is that their syntax is identical to that of classes. Classes, in point of fact, are nothing more than structures that contain functions (at least syntactically).

Structures are frequently employed as a means of consolidating a large number of data points into a single entity.

A structure definition will often include a list of the factors that go into making up the structure. In subsequent definitions, the memory that was previously reserved for structure variables is now freed up. When setting the value of one structure variable to that of another, for example, the structure variables are treated as indivisible units. However, in other circumstances, the members of the structure are available on an individual basis (often using the dot operator).

15.5. KEYWORDS

Structure

Initialization

15.6. REVIEW QUESTIONS

1. A structure brings together a group of
   a. items of the same data type.
   b. related data items.
   c. integers with user-defined names.
   d. variables.
2. True or false: A structure and a class use similar syntax.
3. The closing brace of a structure is followed by a _____.
4. Write a structure specification that includes three variables—all of type int—called hrs, mins, and secs. Call this structure time.
5. True or false: A structure definition creates space in memory for a variable.

6. When accessing a structure member, the identifier to the left of the dot operator is the name of

a. a structure member.

b. a structure tag.

c. a structure variable.

d. the keyword struct.

7. Write a statement that sets the hrs member of the time2 structure variable equal to 11.

8. If you have three variables defined to be of type struct time, and this structure contains three int members, how many bytes of memory do the variables use together?

15.7. FURTHER READING

Fundamental Computer Skills, Feng-Qi Lai, David R. Hofmeister.

A Study of Computer Fundamental, Shu-Jung Yi.

http://books.google.co.in/bkshp?hl=en

Shoolini University

Shoolini University Centre for Distance and Online Education (SCDOE)
Oachghat-Kumarhatti Road, Kasauli Hills, Solan – 173229, H.P.