GOVERNMENT OF TAMILNADU

HIGHER SECONDARY FIRST YEAR

COMPUTER SCIENCE

Untouchability is Inhuman and a Crime

A publication under Free Textbook Programme of Government of Tamil Nadu

Department of School Education

Government of Tamil Nadu

First Edition - 2018

Revised Edition - 2019, 2020

(Published under new syllabus)

NOT FOR SALE

Content Creation

State Council of Educational
Research and Training
© SCERT 2018

Printing & Publishing

Tamil NaduTextbook and Educational
Services Corporation

www.textbooksonline.tn.nic.in

PREFACE

Human civilization achieved the highest peak with the development of computer known as “Computer era”. Literate are those who have the knowledge in using the computer whereas others are considered illiterate inspite of the other degrees obtained.

The growth of the nation at present lies in the hands of the youth, hence the content of this book is prepared in such a way so as to attain utmost knowledge considering the future needs of the youth.

HOW TO USE THE BOOK

- This book does not require prior knowledge in computer Technology
- Each unit comprises of simple activities and demonstrations which can be done by the teacher and also students.
- Technical terminologies are listed in glossary for easy understanding
- The “ Do you know?” boxes enrich the knowledge of reader with additional information
- Workshops are introduced to solve the exercises using software applications
- QR codes are used to link supporting additional
- materials in digital form

Let’s use the QR code in the text books!

o Download DIKSHA app from the Google Play Store.

o Tap the QR code icon to scan QR codes in the textbook.

o Point the device and focus on the QR code.

o On successful scan, content linked to the QR code gets listed.

2020 QR GUIDE

Note: For ICT corner, Digi Links QR codes use any other QR scanner.

CAREER GUIDANCE AFTER 12TH

COURSES	COLLEGES/ UNIVERSITIES	PROFESSION
B.E / B.Tech	All University and their affiliated Colleges and Self financing Colleges in India and Abroad.	Software Engineer, Hardware Engineer, Software Development, Healthcare Section, IT & ITEs
	Science and Humanities	
B.Sc (Computer Science) BCA B.Sc (Maths, Physics, Chemistry, Bio-Chemistry, Geography, journalism, Library Sciences, Political Science, Travel and Tourism)	All University and their affiliated Colleges and Self financing Colleges in India and Abroad.	Government Job and Private Company BPO, Geologist, Journalist
	LAW	
LLB B.A+LLB B.Com BBM+LLB BBA+LLB	All University and their affiliated Colleges and Self financing Colleges in India and Abroad.	Lawyer, Legal Officer, Govt Job
CA	The Institute of Chartered Accountant of India (ICAI)	CA Private and Govt.
Diploma	Government Polytechnic and Self-financing colleges	Junior Engineer (Government and Private)
	Commerce Courses	
B.com-Regular, B.com-Taxation & Tax Procedure, B.com-Travel &Tourism, B.com-Bank Management, B.com-Professional, BBA/BBM-Regular, BFM- Bachelors in Financial Markets, BMS-Bachelors in Management Studies, BAF- Bachelors in Accounting & Finance, Certified Stock Broker & Investment Analysis, Certified Financial Analyst, Certified Financial Planner, Certified Investment Banker	All University and their affiliated Colleges and Self financing Colleges in India and Abroad.	Private Organization , Government ,Banking sectors and prospects for self – employment.

COURSES	COLLEGES/ UNIVERSITIES	PROFESSION
Management Courses		
Business Management Bank Management Event Management Hospital Management Human Resource Management Logistics Management	All University and their affiliated Colleges and Self financing Colleges in India and Abroad.	Private Organization , Government ,Banking sectors and prospects for self – employment.
LAW		
LLB B.A+LLB B.Com BBM+LLB BBA+LLB	All University and their affiliated Colleges and Self financing Colleges in India and Abroad.	Lawyer, Legal Officer, Private Organization , Government, Banking sectors and prospects for self – employment
CA-Chartered Accountant CMA-Cost Management Accountant. CS-Company Secretary (Foundation)	The Institute of Chartered Accountant of India (ICAI)	CA, Private Organization, Government ,Banking sectors and prospects for self – employment.
Science and Humanities		
B.Sc.Botany B.Sc.Zoology B.Sc.Dietician & Nutritionist B.Sc.Home Science B.Sc.Food Technology B.Sc.Dairy Technology B.Sc. Hotel Management B.Sc. Fashion Design B.Sc. Mass Communication B.Sc. Multimedia B.Sc. -3D Animation	All University and their affiliated Colleges and Self financing Colleges in India and Abroad	Government Job and Private Company BPO, Geologist, Journalist
LAW		
LLB B.A+LLB B.Com BBM+LLB BBA+LLB	All University and their affiliated Colleges and Self financing Colleges in India and Abroad.	Lawyer, Legal Officer, Govt Job
CA	The Institute of Chartered Accountant of India (ICAI)	CA Private and Govt.
Diploma	Government Polytechnic and Self-financing colleges	Junior Engineer (Government and Private)

Table of Contents

CNC1EJ

E - book

CNKHG7

Assessment

B174_11_CS_EM

DIGI links

Unit I | Fundamentals of Computers

CHAPTER 1

Introduction to Computers

Learning Objectives

After learning the concepts in this chapter, the students will be able

- To know about Computers
- To learn about various generations of computer
- To understand the basic operations of computers
- To know the components and their functions.
- To know about booting of a computer

DO YOU KNOW?

Father of Computer

Charles Babbage is considered to be the father of computer, for his invention and the concept of Analytical Engine in 1837. The Analytical Engine contained an Arithmetic Logic Unit (ALU), basic flow control, and integrated memory; which led to the development of first general-purpose computer concept.

1.1 Introduction to Computers

Computers are seen everywhere around us, in all spheres of life, in the field of education, research, travel and tourism, weather forecasting, social networking, e-commerce etc. Computers have now become an indispensable part of our lives. Computers have revolutionised our lives with their accuracy and speed of performing a job, it is truly remarkable. Today, no organisation can function without a computer. In fact, various organisations have become paperless. Computers have evolved over the years from a simple calculating device to high speed portable computers.

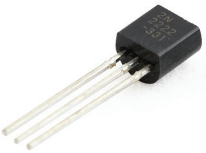
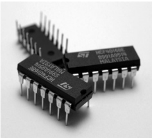
The growth of computer industry started with the need for performing fast calculations. The manual method of computing was slow and prone to errors. So, attempts were made to develop fast calculating devices, the journey started from the first known calculating device (Abacus) which has led us today to an extremely high speed calculating devices.

1.2 Generations of Computers

Growth in the computer industry is determined by the development in technology.

Based on various stages of development, computers can be categroised into different generations.

SN	Generation	Period	Main Component used	Merits/Demerits
1	**First Generation**	1940-1956	 **Vacuum tubes**	• Big in size • Consumed more power • Malfunction due to overheat • Machine Language was used

First Generation Computers - ENIAC , EDVAC , UNIVAC 1 ENIAC weighed about 27 tons, size 8 feet × 100 feet × 3 feet and consumed around 150 watts of power				
2	**Second Generation**	1956-1964	 **Transistors**	• Smaller compared to First Generation • Generated Less Heat • Consumed less power compared to first generation • Punched cards were used • First operating system was developed - Bat ch Processing and Multiprogramming Operating System • Machine language as well as Assembly language was used.
Second Generation Computers **IBM 1401, IBM 1620, UNIVAC 1108**				
3	**Third Generation**	1964 -1971	 **Integrated Circuits (IC)**	• Computers were smaller, faster and more reliable • Consumed less power • High Level Languages were used
Third Generation Computers **IBM 360 series, Honeywell 6000 series**				
4	**Fourth Generation**	1971-1980	 **Microprocessor** Very Large Scale Integrated Circuits (VLSI)	• Smaller and Faster • Microcomputer series such as IBM and APPLE were developed • Portable Computers were introduced.
5	**Fifth Generation**	1980 - till date	 **Ultra Large Scale Integration (ULSI)**	• Parallel Processing • Super conductors • Computers size was drastically reduced. • Can recognise Images and Graphics • Introduction of Artificial Intelligence and Expert Systems • Able to solve high complex problems including decision making and logical reasoning

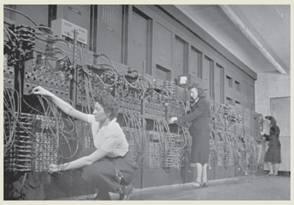
6	**Sixth Generation**	In future		• Parallel and Distributed computing • Computers have become smarter, faster and smaller • Development of robotics • Natural Language Processing • Development of Voice Recognition Software

Table1.1 Generations of computers

DO YOU KNOW?

The first digital computer

The ENIAC (Electronic Numerical Integrator And Calculator) was invented by J. Presper Eckert and John Mauchly at the University of Pennsylvania and began construction in 1943 and was not completed until 1946. It occupied about 1,800 square feet and used about 18,000 vacuum tubes, weighing almost 50 tons. ENIAC was the first digital computer because it was fully functional.



1.3 Sixth Generation Computing

In the Sixth Generation, computers could be defined as the era of intelligent computers, based on Artificial Neural Networks. One of the most dramatic changes in the sixth generation will be the explosive growth of Wide Area Networking. Natural Language Processing (NLP) is a component of Artificial Intelligence (AI). It provides the ability to develop the computer program to understand human language.

DO YOU KNOW?

Optical Character Recognition (Optical Grapheme Recognition) engine for the Indus Scripts has been developed using Deep Learning Neural Networks (a sub-field of Artificial Intelligence).

Given photographs, scans, or any image feed of an Indus Valley Civilization artifact, the system will be able to recognise the inscriptions (the symbol/grapheme sequences) from the image. There are totally 417 Symbols/Graphemes/Characters in the Indus Scripts and just 3700+ text inscriptions of data for the machine to learn and attain expert-level status.

CP4NJH

1.4. Data and Information

We all know what a computer is? It is an electronic device that processes the input according to the set of instructions provided to it and gives the desired output at a very fast rate. Computers are very versatile as they do a lot of different tasks such as storing data, weather forecasting, booking airlines, railway or movie tickets and even playing games.

Data: Data is defined as an un-processed collection of raw facts, suitable for communication, interpretation or processing.

For example, 134, 16 'Kavitha', 'C' are data. This will not give any meaningful message.

Information: Information is a collection of facts from which conclusions may be drawn. In simple words we can say that data is the raw facts that is processed to give meaningful, ordered or structured information. For example Kavitha is 16 years old. This information is about Kavitha and conveys some meaning. This conversion of data into information is called data processing.

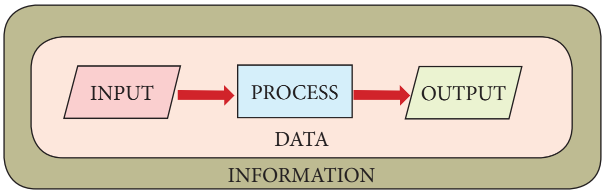


Figure 1.1 Data and Information

"A Computer is an electronic device that takes raw data (unprocessed) as an input from the user and processes it under the control of a set of instructions (called program), produces a result (output), and saves it for future use."

1.5 Components of a Computer

The computer is the combination of hardware and software. Hardware is the physical component of a computer like motherboard, memory devices, monitor, keyboard etc., while software is the set of programs or instructions. Both hardware and software together make the computer system to function.

Figure 1.2: Computer

Let us first have a look at the functional components of a computer. Every task given to a computer follows an Input- Process- Output Cycle (IPO cycle). It needs certain input, processes that input and produces the desired output. The input unit takes the input, the central processing unit does the processing of data and the output unit produces the output. The memory unit holds the data and instructions during the processing.

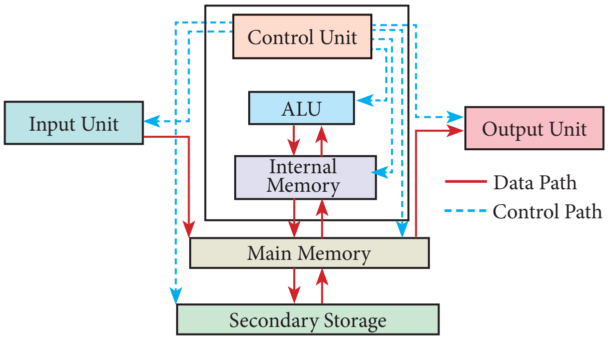


Figure 1.3 components of a computer

1.5.1 Input Unit

Input unit is used to feed any form of data to the computer, which can be stored in the memory unit for further processing. Example: Keyboard, mouse, etc.

1.5.2 Central Processing Unit

CPU is the major component which interprets and executes software instructions. It also control the operation of all other components such as memory, input and output units. It accepts binary data as input, process the data according to the instructions and provide the result as output.

The CPU has three components which are Control unit, Arithmetic and logic unit (ALU) and Memory unit.

1.5.2.1 Arithmetic and Logic Unit

The ALU is a part of the CPU where various computing functions are performed on data. The ALU performs arithmetic operations such as addition, subtraction, multiplication, division and logical operations. The result of an operation is stored in internal memory of CPU. The logical operations of ALU promote the decision-making ability of a computer.

1.5.2.2 Control Unit

The control unit controls the flow of data between the CPU, memory and I/O devices. It also controls the entire operation of a computer.

1.5.3. Output Unit

An Output Unit is any hardware component that conveys information to users in an understandable form. Example: Monitor, Printer etc.

1.5.4. Memory Unit

The Memory Unit is of two types which are primary memory and secondary memory. The primary memory is used to temporarily store the programs and data when the instructions are ready to execute. The secondary memory is used to store the data permanently.

The Primary Memory is volatile, that is, the content is lost when the power supply is switched off. The Random Access Memory (RAM) is an example of a main memory. The Secondary memory is non volatile, that is, the content is available even after the power supply is switched off. Hard disk, CD-ROM and DVD ROM are examples of secondary memory.

1.5.5. Input and Output Devices

Input Devices:

(1) Keyboard: Keyboard (wired / wireless, virtual) is the most common input device used today. The individual keys for letters, numbers and special characters are collectively known as character keys. This keyboard layout is derived from the keyboard of original typewriter. The data and instructions are given as input to the computer by typing on the keyboard. Apart from alphabet and numeric keys, it also has Function keys for performing different functions. There are different set of keys available in the keyboard such as character keys, modifier keys, system and GUI keys, enter and editing keys, function keys, navigation keys, numeric keypad and lock keys.

Figure 1.4 Keyboard

(2) Mouse: Mouse (wired/wireless) is a pointing device used to control the movement of the cursor on the display screen. It can be used to select icons, menus, command buttons or activate something on a computer. Some mouse actions are move, click, double click, right click, drag and drop.

Different types of mouse available are: Mechanical Mouse, Optical, Laser Mouse, Air Mouse, 3D Mouse, Tactile Mouse, Ergonomic Mouse and Gaming Mouse.

Who invented Mouse?

The computer mouse as we know it today was invented and developed by Douglas Engelbart, with the assistance of Bill English, during the 1960's and was patented on November 17, 1970.

CPDJL5

(3) Scanner: Scanners are used to enter the information directly into the computer's memory. This device works like a Xerox machine. The scanner converts any type of printed or written information including photographs into a digital format, which can be manipulated by the computer.

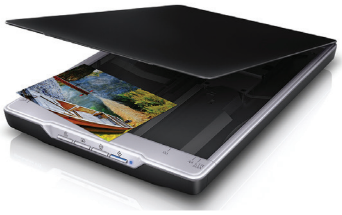


Figure 1.5 Scanner

(4)Fingerprint Scanner: Finger print Scanner is a fingerprint recognition device used for computer security, equipped with the fingerprint recognition feature that uses biometric technology. Fingerprint Reader / Scanner is a very safe and convenient device for security instead of using passwords, which is vulnerable to fraud and is hard to remember.

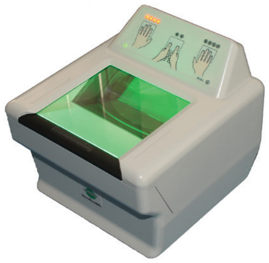


Figure 1.6 Fingerprint Scanner

(5) Track Ball: Track ball is similar to the upside- down design of the mouse. The user moves the ball directly, while the device itself remains stationary. The user spins the ball in various directions to navigate the screen movements.

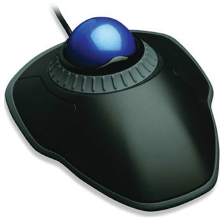


Figure 1.7 Track Ball

(6) Retinal Scanner: This performs a retinal scan which is a biometric technique that uses unique patterns on a person's retinal blood vessels.

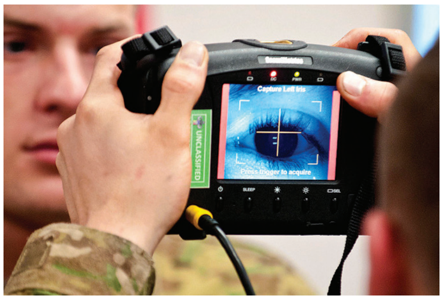


Figure 1.8 Retinal Scanner

(7) Light Pen: A light pen is a pointing device shaped like a pen and is connected to a monitor. The tip of the light pen contains a light-sensitive element which detects the light from the screen enabling the computer to identify the location of the pen on the screen. Light pens

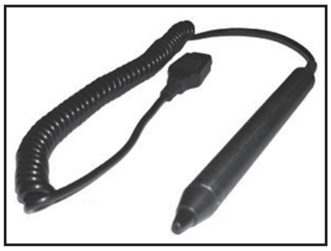


Figure 1.9 Light Pen

have the advantage of 'drawing' directly onto the screen, but this becomes hard to use, and is also not accurate.

(8) Optical Character Reader: It is a device which detects characters printed or written on a paper with OCR, a user can scan a page from a book. The Computer will recognise the characters in the page as letters and punctuation marks and stores. The Scanned document can be edited using a wordprocessor.

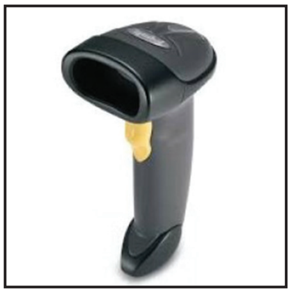


Figure 1.10 Optical Character Reader

(9) Bar Code / QR Code Reader: A Bar code is a pattern printed in lines of different thickness. The Bar code reader scans the information on the bar codes transmits to the Computer for further processing. The system gives fast and error free entry of information into the computer.

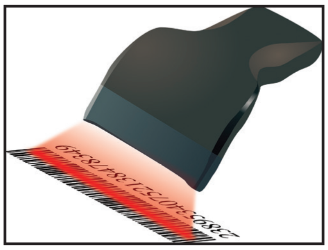


Figure 1.11 Bar code Reader

QR (Quick response) Code: The QR code is the two dimension bar code which can be read by a camera and processed to interpert the image.

(10) Voice Input Systems: Microphone serves as a voice Input device. It captures the voice data and send it to the Computer. Using the microphone along with speech recognition software can offer a completely new approach to input information into the Computer.

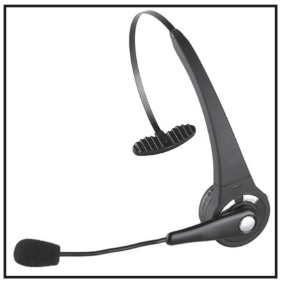


Figure 1.12 Voice input System

(11) Digital Camera: It captures images / videos directly in the digital form. It uses a CCD (Charge Coupled Device) electronic chip. When light falls on the chip through the lens, it converts light rays into digital format.

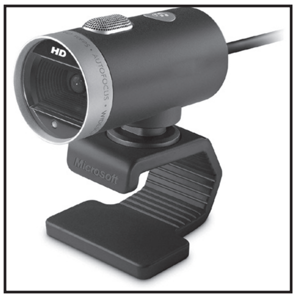


Figure 1.13 Digital Camera

(12) Touch Screen: A touch screen is a display device that allows the user to interact with a computer by using the finger. It can be quite useful as an alternative to a mouse or keyboard for navigating a Graphical User Interface (GUI). Touch screens are used on a wide variety of devices such as computers, laptops, monitors, smart phones, tablets, cash registers and information kiosks. Some touch screens use a grid of infrared beams to sense the presence of a finger instead of utilizing touch-sensitive input.

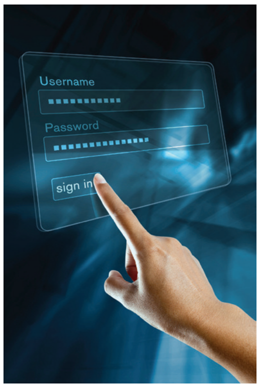


Figure 1.14 Touch Screen

(13) Keyer : A Keyer is a device for signaling by hand, by way of pressing one or more switches. Modern keyers have a large number of switches but not as many as a full size keyboard. Typically,

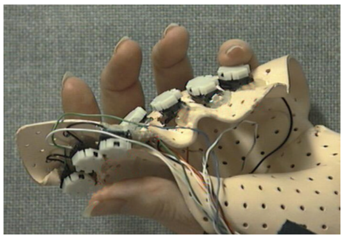


Figure 1.15 Keyer

this number is between 4 and 50. A keyer differs from a keyboard, which has "no board", but the keys are arranged in a cluster.

Output Devices:

(1) Monitor: Monitor is the most commonly used output device to display the information. It looks like a TV. Pictures on a monitor are formed with picture elements called PIXELS. Monitors may either be Monochrome which display text or images in Black and White or can be color, which display results in multiple colors. There are many types of monitors available such as CRT (Cathode Ray Tube), LCD (Liquid Crystal Display) and LED (Light Emitting Diodes). The monitor works with the VGA (Video Graphics Array) card. The video graphics card helps the keyboard to communicate with the screen. It acts as an interface between the computer and display monitor. Usually the recent motherboards incorporate built-in video card.

Figure 1.16 Monitor

The first computer monitor was part of the Xerox Alto computer system, which was released on March 1, 1973.

(2) Plotter: Plotter is an output device that is used to produce graphical output on papers. It uses single color or multi color pens to draw pictures.

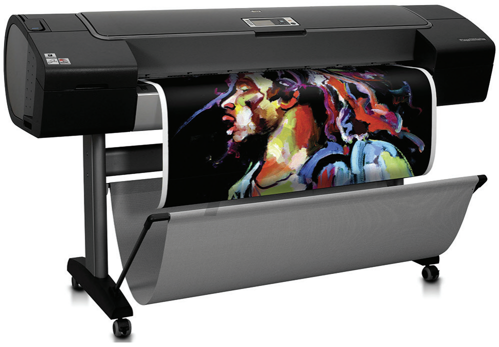


Figure 1.17 Plotter

(3) Printers: Printers are used to print the information on papers. Printers are divided into two main categories:

- Impact Printers
- Non Impact printers

Impact Printers

These printers print with striking of hammers or pins on ribbon. These printers can print on multi-part (using carbon papers) by using mechanical pressure. For example, Dot Matrix printers and Line matrix printers are impact printers.

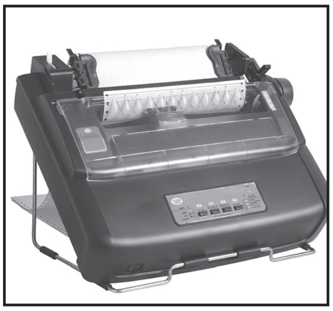


Figure 1.18 Impact Printer

A Dot matrix printer that prints using a fixed number of pins or wires. Each dot is produced by a tiny metal rod, also called a “wire” or “pin”, which works by the power of a tiny electromagnet or solenoid, either directly or through a set of small levers. It generally prints one line of text at a time. The printing speed of these printers varies from 30 to 1550 CPS (Character Per Second).

Line matrix printers use a fixed print head for printing. Basically, it prints a page-wide line of dots. But it builds up a line of text by printing lines of dots. Line printers are capable of printing much more than 1000 Lines Per Minute, resulting in thousands of pages per hour. These printers also uses mechanical pressure to print on multi-part (using carbon papers).

Non-Impact Printers

These printers do not use striking mechanism for printing. They use electrostatic or laser technology. Quality and speed of these printers are better than Impact printers. For example, Laser printers and Inkjet printers are non-impact printers.

Laser Printers

Laser printers mostly work with similar technology used by photocopiers. It makes a laser beam scan back and forth across a drum inside the printer, building up a pattern. It can produce very good quality of graphic images. One of the chief characteristics of laser printer is their resolution – how many Dots per inch(DPI). The available resolution range around 1200 dpi. Approximately it can print 100 pages per minute(PPM)

Figure 1.19 Laser Printer

Inkjet Printers:

Inkjet Printers use colour cartridges which combined Magenta, Yellow and Cyan inks to create color tones. A black cartridge is also used for monochrome output. Inkjet printers work by spraying ionised ink at a sheet of paper. The speed of Inkjet printers generaly range from 1-20 PPM (Page Per Minute).

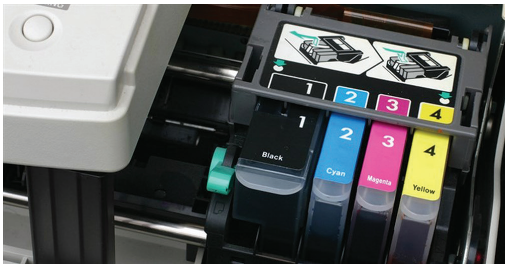


Figure 1.20 Inkjet Printer

They use the technology of firing ink by heating it so that it explodes towards the paper in bubbles or by using piezoelectricity in which tiny electric currents controlled by electronic circuits are used inside the printer to spread ink in jet speed. An Inkjet printer can spread millions of dots of ink at the paper every single second.

Speakers: Speakers produce voice output (audio) . Using speaker along with speech synthesise software, the computer can provide voice output. This has become very common in places like airlines, schools, banks, railway stations, etc.

Figure 1.21 Speakers

Multimedia Projectors:

Multimedia projectors are used to produce computer output on a big screen. These are used to display presentations in meeting halls or in classrooms.

Figure 1.22 Multimedia Projector

1.6 Booting of computer

An Operating system (OS) is a basic software that makes the computer to work. When a computer is switched on, there is no information in its RAM.At the same time, in ROM, the pre-written program called POST (Power on Self Test) will be executed first. This program checks if the devices like RAM, keyboard, etc., are connected properly and ready to operate. If these devices are ready, then the BIOS (Basic Input Output System) gets executed. This process is called Booting. Thereafter, a program called "Bootstrap Loader" transfers OS from hard disk into main memory. Now the OS gets loaded (Windows/Linux, etc.,) and will get executed. Booting process is of two types.

1) Cold Booting

2) Warm Booting

Cold Booting: When the system starts from initial state i.e. it is switched on, we call it cold booting or Hard Booting. When the user presses the Power button, the instructions are read from the ROM to initiate the booting process.

Warm Booting: When the system restarts or when Reset button is pressed, we call it Warm Booting or Soft Booting. The system does not start from initial state and so all diagnostic tests need not be carried out in this case. There are chances of data loss and system damage as the data might not have been stored properly.

Points to Remember:

- Computers are seen everywhere around us, in all spheres of life.
- It is an electronic device that processes the input according to the set of instructions provided to it and gives the desired output at a very fast rate.
- Based on various stages of development, computers can be divided into six different generations.
- The computer is the combination of hardware and software.
- Hardware is the physical component of a computer.
- Input unit is used to feed any form of data to the computer.
- CPU interprets and executes software instructions.
- The ALU is a part of the CPU where various computing functions are performed on data.
- The control unit controls the flow of data between the CPU, memory and I/O devices.
- An Output Unit is any hardware component that conveys information to one or more people in user understandable form.
- The Memory Unit is of two kinds which are primary memory and secondary memory.

Activity

STUDENT ACTIVITY

1. Explain the classification of computers.
2. Give the details of motherboard names, RAM capacity used in the years 1993, 1995, 2005, 2008, 2016.
3. Mention two new input and output devices that are not given in this chapter.

Evaluation

SECTION – A

Choose the correct answer

1. First generation computers used

   (a) Vacuum tubes
   (b) Transistors
   (c) Integrated circuits
   (d) Microprocessors

CPMFMS

2. Name the volatile memory

   (a) ROM (b) PROM
   (c) RAM (d) EPROM

3. Identify the output device

   (a) Keyboard (b) Memory
   (c) Monitor (d) Mouse

4. Identify the input device

   (a) Printer (b) Mouse
   (c) Plotter (d) Projector

5. ……..… Output device is used for printing building plan.

   (a) Thermal printer
   (b) Plotter
   (c) Dot matrix
   (d) inkjet printer

6. Which one of the following is used to in ATM machines

   (a) Touch Screen (b) speaker
   (c) Monitor (d) Printer

7. When a system restarts which type of booting is used.

   (a) Warm booting
   (b) Cold booting
   (c) Touch boot
   (d) Real boot.

8. Expand POST

   (a) Post on self Test
   (b) Power on Software Test
   (c) Power on Self Test
   (d) Power on Self Text

9. Which one of the following is the main memory?

   (a) ROM (b) RAM
   (c) Flash drive (d) Hard disk

10. Which generation of computer used IC's?

    (a) First (b) Second
    (c) Third (d) Fourth

SECTION-B

Very Short Answers

1. What is a computer?
2. Distinguish between data and information.
3. What are the components of a CPU?
4. What is the function of an ALU?
5. Write the functions of control unit.
6. What is the function of memory?
7. Differentiate Input and output unit.
8. Distinguish Primary and Secondary memory.

SECTION-C

Short Answers

1. What are the characteristics of a computer?
2. Write the applications of computer.
3. What is an input device? Give two examples.
4. Name any three output devices.
5. Differentiate optical and Laser mouse
6. Write shortnote on impact printer
7. Write the characteristics of sixth generation.
8. Write the significant features of monitor.

GLOSSARY

CPHBPF

SECTION - D

Explain in detail

1. Explain the basic components of a computer with a neat diagram.
2. Discuss the various generations of computers.
3. Explain the following

   a. Inkjet Printer b. Multimedia projector c. Bar code / QR code Reader

References

(1) Fundamentals of Computers – V. Rajaraman – PHI Publications

(2) Computer Science text book – NCERT, New Delhi

Internet Resources

(1) www.wikipedia.org

(2) https://www.computerhope.com/jargon/c/computer.htm

CASE STUDY

Prepare a comparative study of various computers of past and present with respect to speed, memory, size, power consumption and other features

Computer	It is an electronic device that processes the input according to the set of instructions provided to it and gives the desired output at a very fast rate.
Vacuum tube	Vacuum tubes contain electrodes for controlling electron flow and were used in early computers as a switch or an amplifier.
Transistors	The transistor ("transfer resistance") is made up of semi-conductors. It is a component used to control the amount of current or voltage used for amplification/modulation of an electronic signal.
Punched cards	Punch cards also known as Hollerith cards are paper cards containing several punched or perforated holes that were punched by hand or machine to represent data.
Machine Language	Machine language is a collection of binary digits or bits that the computer reads and interprets.
Assembly language	An assembly language is a low-level programming language.

Integrated Circuits	The IC is a package containing many circuits, pathways, transistors, and other electronic components all working together to perform a particular function or a series of functions.
Microcomputer	Micro computer is used to describe a standard personal computer.
High-level languages	A high-level language is a computer programming language that isn't limited by the computer, designed for a specific job, and is easier to understand.
Natural Language Processing (NLP)	Natural Language Processing is a method used in artificial intelligence to process and derive meaning from the human language.
Robotics	Robot is a term coined by Karel Capek in the 1921 to play RUR (Rossum's Universal Robots). It is used to describe a computerized machine designed to respond to input received manually or from its surroundings.
Nanotechnology	Nanotechnology is an engineering, science, and technology that develops machines or works with one atom or one molecule that is 100 nanometers or smaller.
Bioengineering	A discipline that applies engineering principles of design and analysis to biological systems and biomedical technologies

Unit I Fundamentals of Computers

CHAPTER 2

Number Systems

Learning Objectives

- To know how the computer interprets and stores data in the memory.
- To learn various data representations and binary arithmetic.
- To learn conversion between various Number Systems.

2.1 Introduction

The term data comes from the word **datum**, which means a raw fact. The data is a fact about people, places or some objects.

Example:

Let 'Name', 'Age', 'Class', 'Marks' and 'Subject' be some defined variables. Now, let us assign a value to each of these variables.

Name	=	Rajesh
Age	=	16
Class	=	XI
Mark	=	65
Subject	=	Computer Science

Figure 2.1 Example for Data

In the above example, the values assigned to the five different variables are called **data**. When the above data is processed, we get an information "Rajesh is 16 years old, studying in Class XI, has scored 65 marks in Computer Science subject".

2.2 Data Representations

Computer handles data in the form of '0'(Zero) and '1' (One). Any kind of data like number, alphabet, special character should be converted to '0' or '1' which can be understood by the Computer. '0' and '1' that the Computer can understand is called **Machine language**. '0' or '1' are called '**Bi**nary Digi**t**s'(BIT). Therefore, the study of data representation in the computer is important.

- A **bit** is the short form of **Binary digit** which can be '0' or '1'. It is the basic unit of data in computers.
- A **nibble** is a collection of 4 bits (Binary digits).
- A collection of 8 bits is called **Byte**. A byte is considered as the basic unit of measuring the memory size in the computer.
- **Word length** refers to the number of bits processed by a Computer's CPU. For example, a word length can have 8 bits, 16 bits, 32 bits and 64 bits (Present day Computers use 32 bits or 64 bits)

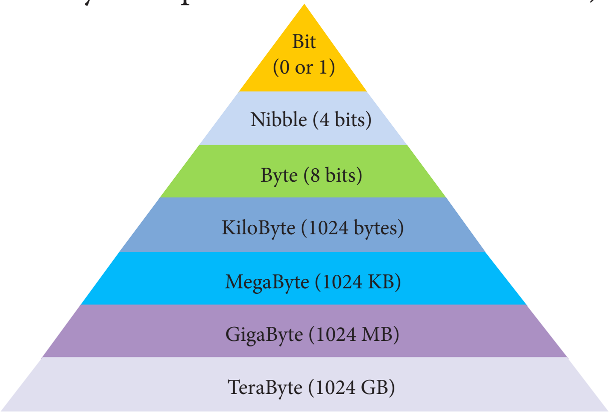


Figure 2.2 Data Representation

Computer memory (Main Memory and Secondary Storage)is normally represented in terms of KiloByte (KB) or MegaByte (MB). In decimal system, 1 Kilo represents 1000, that is , 10^3. In binary system, 1 KiloByte represents 1024 bytes that is 2^{10}. The following table represents the various memory sizes:

Table 2.1 Memory Size (Read 2^10 as 2 power 10)

Name	Abbr.	Size
Kilo	K	2^10 = 1,024
Mega	M	2^20 = 1,048,576
Giga	G	2^30 = 1,073,741,824
Tera	T	2^40 = 1,099,511,627,776
Peta	P	2^50 = 1,125,899,906,842,624
Exa	E	2^60 = 1,152,921,504,606,846,976
Zetta	Z	2^70 = 1,180,591,620,717,411,303,424
Yotta	Y	2^80 = 1,208,925,819,614,629,174,706,173

CQ67R3

Bytes are used to represent characters in a text. Different types of coding schemes are used to represent the character set and numbers. The most commonly used coding scheme is the **American Standard Code for Information Interchange** (ASCII). Each binary value between 0 and 127 is used to represent a specific character. The ASCII value for (blank space) is 32 and the ASCII value of numeric 0 is 48. The range of ASCII values for lower case alphabets is from 97 to 122 and the range of ASCII values for the upper case alphabets is 65 to 90.

DO YOU KNOW?

The speed of a computer depends on the number of bits it can process at once. For example, a 64- bit computer can process 64-bit numbers in one operation, while a 32-bit computer break 64-bit numbers down into smaller pieces, making it slower.

2.3 Different Types of Number Systems

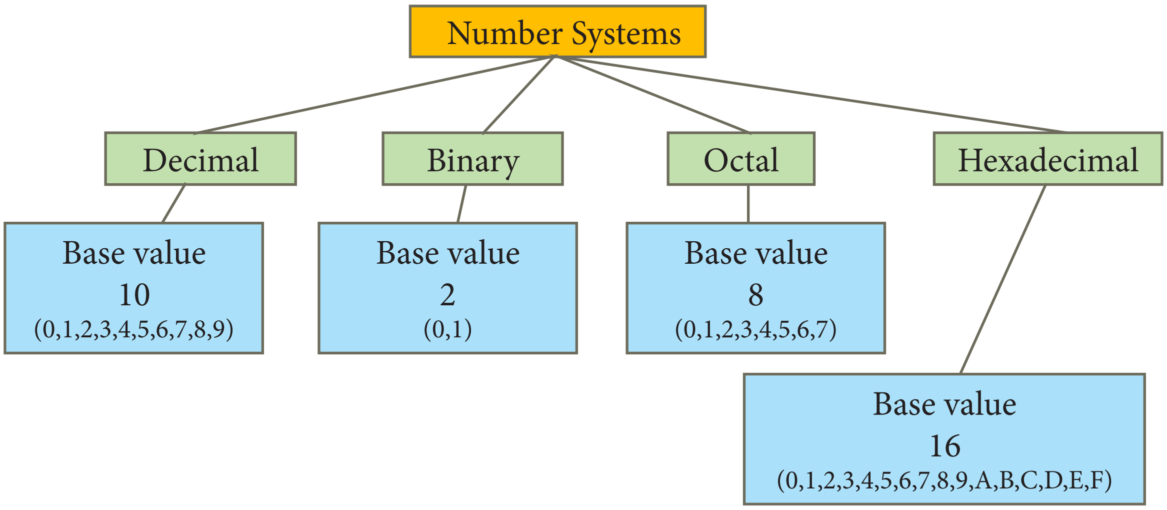

Number Systems
Decimal
Binary
Octal
Hexadecimal
Base value 10 (0,1,2,3,4,5,6,7,8,9)
Base value 2 (0,1)
Base value 8 (0,1,2,3,4,5,6,7)
Base value 16 (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

Figure 2.3. Number Systems

A numbering system is a way of representing numbers. The most commonly used numbering system in real life is Decimal number system. Other number systems are Binary, Octal, Hexadecimal number system. Each number system is uniquely identified by its **base value** or **radix**. Radix or base is the count of number of digits in each number system. Radix or base is the general idea behind positional numbering system.

2.3.1 Decimal Number System

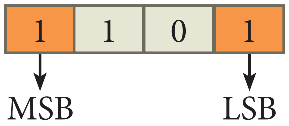
It consists of 0,1,2,3,4,5,6,7,8,9(10 digits). It is the oldest and most popular number system used in our day to day life. In the positional number system, each decimal digit is weighed relative to its position in the number. This means that each digit in the number is multiplied by 10 raised to a power corresponding to that digit's position.

Example

$$\begin{aligned}(123)_{10} &= 1x10^2 + 2x10^1 + 3x10^0\\ &= 100 + 20 + 3\\ &= (123)_{10}\end{aligned}$$

2.3.2 Binary Number System

There are only two digits in the Binary system, namely, 0 and 1. The numbers in the binary system are represented to the base 2 and the positional multipliers are the powers of 2. The left most bit in the binary number is called as the **M**ost **S**ignificant **B**it (MSB) and it has the largest positional weight. The right most bit is the **L**east **S**ignificant **B**it (LSB) and has the smallest positional weight.



Example

The binary sequence $(1101)_2$ has the decimal equivalent:

$$\begin{aligned}(1101)_2 &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0\\ &= 8 + 4 + 0 + 1\\ &= (13)_{10}\end{aligned}$$

2.3.3 Octal Number System

Octal number system uses digits 0,1,2,3,4,5,6 and 7 (8 digits). Each octal digit has its own positional value or weight as a power of 8.

Example

The Octal sequence $(547)_8$ has the decimal equivalent:

$$\begin{aligned}(547)_8 &= 5\times8^2 + 4\times8^1 + 7\times8^0\\ &= 5\times64 + 4\times8 + 7\times1\\ &= 320 + 32 + 7\\ &= (359)_{10}\end{aligned}$$

2.3.4 Hexadecimal Number System

A hexadecimal number is represented using base 16. Hexadecimal or Hex numbers are used as a shorthand form of binary sequence. This system is used to represent data in a more compact manner. Since 16 symbols are used, 0 to F, the notation is called hexadecimal. The first 10 symbols are the same as in the decimal system, 0 to 9 and the remaining 6 symbols are taken from the first 6 letters of the alphabet sequence, A to F, where A represents 10, B is 11, C is 12, D is 13, E is 14 and F is 15.

Table 2.2 Binary, Octal, Hexadecimal equivalent of Decimal Numbers

Decimal	Binary	Octal	Hexadecimal
0	0000	000	0000
1	0001	001	0001
2	0010	002	0002
3	0011	003	0003
4	0100	004	0004
5	0101	005	0005
6	0110	006	0006
7	0111	007	0007
8	1000	010	0008
9	1001	011	0009
10	1010	012	A
11	1011	013	B
12	1100	014	C
13	1101	015	D
14	1110	016	E
15	1111	017	F

Example

The hexadecimal sequence $(25)_{16}$ has the decimal equivalent:

$$\begin{aligned}(25)_{16} &= 2\times16^1 + 5\times16^0\\ &= 32+5\\ &= (37)_{10}\end{aligned}$$

Workshop

1. Identify the number system for the following numbers

S. No.	Number	Number system
1	$(1010)_{10}$	Decimal Number system
2	$(1010)_2$	
3	$(989)_{16}$	
4	$(750)_8$	
5	$(926)_{10}$	

2. State whether the following numbers are valid or not. If invalid, give reason.

S.No.	Statement	Yes / No	Reason (If invalid)
1.	786 is an Octal number		
2.	101 is a Binary number		
3.	Radix of Octal number is 7		

2.4 Number System Conversions

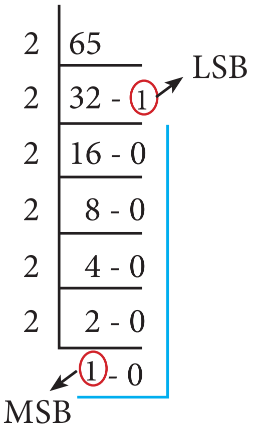
2.4.1 Decimal to Binary Conversion

Generally two methods followed.

Method 1: To convert Decimal to Binary "**Repeated Division by 2**" method can be used. Any Decimal number divided by 2 will leave a remainder of 0 or 1. Repeated division by 2 will leave a sequence of 0s and 1s that become the binary equivalent of the decimal number. Suppose it is required to convert the decimal number N into binary form, dividing N by 2 in the decimal system, we will obtain a quotient N1 and a remainder R1, where R1 can have a value of either 0 or 1. The process is repeated until the quotient becomes 0 or 1. When the quotient is '0' or '1', it is the final remainder value. Write the final answer starting from final remainder value obtained to the first remainder value obtained.

Example

Convert $(65)_{10}$ into its equivalent binary number



Remainder

Note :

65/2 = 32 + 1

32/2 = 16 + 0

16/2 = 8 + 0

8/2 = 4 + 0

4/2 = 2 + 0

2/2 = 1 + 0

$$(65)_{10} = (1\ 0\ 0\ 0\ 0\ 0\ 1)_2$$

Method 2 : Sum of Powers of 2.

A decimal number can be converted into a binary number by adding up the powers of 2 and then adding bits as needed to obtain the total value of the number.

a) Find the largest power of 2 that is smaller than or equal to 65.

$$65_{10} > 64_{10}$$

b) Set the 64's bit to 1 and subtract 64 from the original number

$$65-64=1$$

c) 32 is greater than the remaining total. Therefore, set the 32's bit to 0.

d) 16 is greater than the remaining total. Therefore, set the 16's bit to 0.

e) 8 is greater than the remaining total. Therefore, set the 8's bit to 0.

f) 4 is greater than the remaining total. Therefore, set the 4's bit to 0.

g) 2 is greater than the remaining total. Therefore, set the 2's bit to 0.

h) As the remaining value is equivalent to 1's bit, set it to 1.

$$1-1=0$$

Conversion is complete $65_{10} = (1000001)_2$

Example

The conversion steps can be given as follows:

Given Number : 65

Equivalent or value less than power of 2 is : 64

(1) 65 - 64 = 1

(2) 1 - 1= 0

Power's of 2	64	32	16	8	4	2	1
Binary Number	1	0	0	0	0	0	1

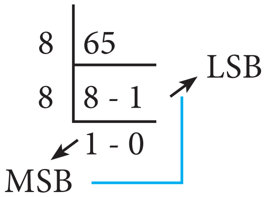
$$65_{10} = (1000001)_2$$

2.4.2 Decimal to Octal Conversion

To convert Decimal to Octal, "**Repeated Division by 8**" method can be used. The method is the same we have learnt in 2.4.1, but in this method, we have to divide the given number by 8.

Example

Convert $(65)_{10}$ into its equivalent Octal number



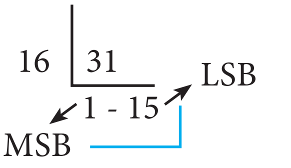
$$(65)_{10} = (1\ 0\ 1)_8$$

2.4.3 Decimal to Hexadecimal Conversion

To convert Decimal to Hexadecimal, "**Repeated division by 16**" method can be used. The method is the same as we have learnt in 2.4.1, but in this method, we have to divide the given number by 16.

Example

Convert $(31)_{10}$ into its equivalent hexadecimal number.



$(16)_{10} = (1F)_{16}$(Refer Table 2.2 F=15)

2.4.4 Conversion of fractional Decimal to Binary

The method of **repeated multiplication by 2** has to be used to convert such kind of decimal fractions.

The steps involved in the method of **repeated multiplication by 2**:

Step 1: Multiply the decimal fraction by 2 and note the integer part. The integer part is either 0 or 1.

Step 2: Discard the integer part of the previous product. Multiply the fractional part of the previous product by 2. Repeat Step 1 until the same fraction repeats or terminates (0).

Step 3: The resulting integer part forms a sequence of 0s and 1s that become the binary equivalent of decimal fraction.

Step 4: The final answer is to be written from first integer part obtained till the last integer part obtained.

	Integer part	
0.2 × 2 = 0.4	0	(first integer part obtained)
0.4 × 2 = 0.8	0	
0.8 × 2 = 1.6	1	
0.6 × 2 = 1.2	1	
0.2 × 2 = 0.4	0	(last integer part obtained)

Note: Fraction repeats, the product is the same as in the first step.

Write the integer parts from top to bottom to obtain the equivalent fractional binary number. Hence $(0.2)_{10}=(0.00110011\ldots)_2 = (0.00110011)_2$

Workshop

3. Convert the following Decimal numbers to its equivalent Binary, Octal, Hexadecimal.

1) 1920 2) 255 3)126

2.4.5 Binary to Decimal Conversion

To convert Binary to Decimal we can use positional notation method.

Step 1: Write down the Binary digits and list the powers of 2 from right to left(Positional Notation)

Step 2: For each positional notation written for the digit, now write the equivalent weight.

Step 3: Multiply each digit with its corresponding weight

Step 4: Add all the values.

Table 2.3 Positional Notation and Weight

Positional Notation	Weight	Positional Notation	Weight
2^0	1	2^6	64
2^1	2	2^7	128
2^2	4	2^8	256
2^3	8	2^9	512
2^4	16	2^{10}	1024
2^5	32		

Example

Convert $(111011)_2$ into its equivalent decimal number.

Weight	32	16	8	4	2	1
Positional Notation	2^5	2^4	2^3	2^2	2^1	2^0
Given number	1	1	1	0	1	1

$$32+16+8+0+2+1 = (59)_{10}$$
$$(111011)_2 = (59)_{10}$$

2.4.6 Binary to Octal Conversion

Step 1: Group the given binary number into 3 bits from right to left.

Step 2: You can add preceding 0 to make a group of 3 bits if the left most group has less than 3 bits.

Step 3: Convert equivalent octal value using "2's power positional weight method"

Table 2.4 Octal numbers and their Binary equivalent

Octal	Binary Equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Example

Convert $(11010110)_2$ into octal equivalent number

Step 1: Group the given number into 3 bits from right to left.

011 010 110

Note: The left most groups have less than 3 bits, so 0 is added to its left to make a group of 3 bits.

Step-2: Find Octal equivalent of each group

$$\underbrace{011}_{3} \qquad \underbrace{010}_{2} \qquad \underbrace{110}_{6}$$
$$(11010110)_2 = (326)_8$$

2.4.7. Binary to Hexadecimal Conversion

Step 1: Group the given number into 4 bits from right to left.

Step 2: You can add preceding 0's to make a group of 4 bits if the left most group has less than 4 bits.

Step 3: Convert equivalent Hexadecimal value using "2's power positional weight method"

Example

Convert $(1111010110)_2$ into Hexadecimal number

Step 1: Group the given number into 4 bits from right to left.

0011 1101 0110

Note: 0's are added to the left most group to make it a group of 4 bits

$$\underbrace{0011}_{3} \qquad \underbrace{1101}_{D} \qquad \underbrace{0110}_{6}$$
$$(1111010110)_2 = (3D6)_{16}$$

2.4.8 Conversion of fractional Binary to Decimal equivalent

Follow the steps to convert fractional Binary number to its Decimal equivalent.

Step 1: Convert integral part of Binary to Decimal equivalent using positional notation method (Procedure is same as discussed in 2.4.5)

Step 2: To convert the fractional part of binary to its decimal equivalent.

Step 2.1: Write down the Binary digits in the fractional part

Step 2.2: For all the digits write powers of 2 from left to right starting from 2^{-1}, 2^{-2}, 2^{-3}...... 2^{-n}, now write the equivalent weight.

Step 2.3: Multiply each digit with its corresponding weight

Step 2.4: Add all the values which you obtained in Step 2.3

Table 2.5 Positional notation and weight

Positional notation	Weight
2^{-1} (1/2)	0.5
2^{-2} (1/4)	0.25
2^{-3} (1/8)	0.125
2^{-4} (1/16)	0.0625
2^{-5} (1/32)	0.03125
2^{-6} (1/64)	0.015625
2^{-7} (1/128)	0.0078125

Step 3: To get final answer write the integral part (after conversion), followed by a decimal point(.) and the answer arrived at Step 2.4

Example

Convert the given Binary number $(11.011)_2$ into its decimal equivalent Integer part $(11)_2 = 3$ (Refer **table 2.2**)

2^1	2^0		2^{-1}	2^{-2}	2^{-3}
↑	↑		↑	↑	↑
1	1	.	0	1	1

$$3 + .\,(0\times0.5 + 1\times0.25 + 1\times0.125)$$
$$= 3.\,375$$
$$(11.011)_2 = (3.375)_{10}$$

Workshop

4. Convert the given Binary number into its equivalent Decimal, Octal and Hexadecimal number.

1) 101110101 2) 1011010 3) 101011111

2.4.9. Octal to Decimal Conversion

To convert Octal to Decimal, we can use positional notation method.

1. Write down the Octal digits and list the powers of 8 from right to left(Positional Notation)
2. For each positional notation of the digit write the equivalent weight.
3. Multiply each digit with its corresponding weight
4. Add all the values

Example

Convert $(1265)_8$ to equivalent Decimal number

Weight	512	64	8	1
Positional Notation	8^3	8^2	8^1	8^0
Given number	1	2	6	5

$$(1265)_8 = 512\times1 + 64\times2 + 8\times6 + 1\times5$$
$$= 512 + 128 + 48 + 5$$
$$(1265)_8 = (693)_{10}$$

2.4.10 Octal to Binary Conversion

For each Octal digit in the given number write its 3 digits binary equivalent using positional notation.

Example

Convert $(6213)_8$ to equivalent Binary number

6	2	1	3
↓	↓	↓	↓
110	010	001	011

$$(6213)_8 = (110010001011)_2$$

Workshop

5. Convert the following Octal numbers into Binary numbers.

(A) 472 (B) 145 (C) 347

(D) 6247 (E) 645

2.4.11 Hexadecimal to Decimal Conversion

To convert Hexadecimal to Decimal we can use positional notation method.

1. Write down the Hexadecimal digits and list the powers of 16 from right to left(Positional Notation)
2. For each positional notation written for the digit, now write the equivalent weight.
3. Multiply each digit with its corresponding weight
4. Add all the values to get one final value.

Example

Convert $(25F)_{16}$ into its equivalent Decimal number.

Weight	256	16	1
Positional Notation	16^2	16^1	16^0
Given number	2	5	F(15)

$(25F)_{16} = 2\times256\ +\ 5\times16\ +15\times1$

$= 512 + 80 + 15$

$(25F)_{16} = (607)_{10}$

2.4.12 Hexadecimal to Binary Conversion

Write 4 bits Binary equivalent for each Hexadecimal digit for the given number using positional notation method.

Example

Convert $(8BC)_{16}$ into equivalent Binary number

8	B	C
↓	↓	↓
1000	1011	1100

$(8BC)_{16} = (100010111100)_2$

Workshop

6. Convert the following Hexadecimal numbers to Binary numbers
(A) A6 (B) BE
(C) 9BC8 (D) BC9

2.5 Binary Representation for Signed Numbers

Computers can handle both positive (unsigned) and negative (signed) numbers. The simplest method to represent negative binary numbers is called **Signed Magnitude**. In signed magnitude method, the left most bit is Most Significant Bit (MSB), is called **sign bit or parity bit**.

The numbers are represented in computers in different ways:

- Signed Magnitude representation
- 1's Complement
- 2's Complement

2.5.1 Signed Magnitude representation

The value of the whole numbers can be determined by the sign used before it. If the number has '+' sign or no sign it will be considered as positive. If the number has '–' sign it will be considered as negative.

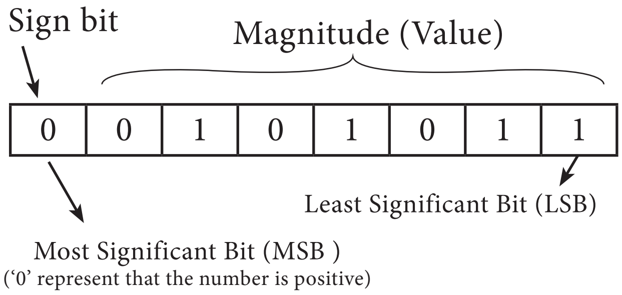
Example:

+43 or 43 is a positive number

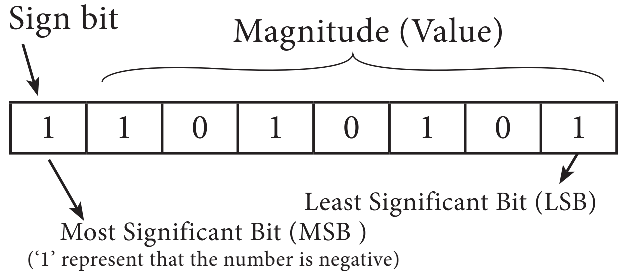
–43 is a negative number

In signed binary representation, the left most bit is considered as sign bit. If this bit is 0, it is a positive number and if it 1, it is a negative number. Therefore a signed binary number has 8 bits, only 7 bits used for storing values (magnitude) and the 1 bit is used for sign.

+43 is represented in memory as follows:


Sign bit
Magnitude (Value)
0 0 1 0 1 0 1 1
Least Significant Bit (LSB)
Most Significant Bit (MSB)
('0' represent that the number is positive)

-43 can be represented in memory as follows.


Sign bit
Magnitude (Value)
1 1 0 1 0 1 0 1
Least Significant Bit (LSB)
Most Significant Bit (MSB)
('1' represent that the number is negative)

2.5.2 1's Complement representation

This is an easier approach to represent signed numbers. This is for negative numbers only i.e. the number whose MSB is 1.

The steps to be followed to find 1's complement of a number:

Step 1: Convert given Decimal number into Binary

Step 2: Check if the binary number contains 8 bits , if less add 0 at the left most bit, to make it as 8 bits.

Step 3: Invert all bits (i.e. Change 1 as 0 and 0 as 1)

Example

Find 1's complement for $(-24)_{10}$

Given Number	Binary Number	1's Compliment
$(-24)_{10}$	00011000	11100111

2.5.3 2's Complement representation

The 2's-complement method for negative number is as follows:

a. Invert all the bits in the binary sequence (i.e., change every 0 to1 and every 1 to 0 ie.,1's complement)

b. Add 1 to the result to the Least Significant Bit (LSB).

Example

2's Complement represent of $(-24)_{10}$

Binary equivalent of +24:	11000
8bit format:	00011000
1's complement:	11100111
Add 1 to LSB:	+1
2's complement of -24:	**11101000**

Workshop

7. Write the 1's complement number and 2's complement number for the following decimal numbers:
(A) 22 (B) -13 (C) -65 (D) -46

2.6 Binary Arithmetic

As decimal numbers, the binary numbers also permit computations like addition, subtraction, multiplication and division. The following session deals only with binary addition and subtraction.

2.6.1 Binary Addition

The following table is useful when adding two binary numbers.

A	B	SUM (A + B)	Carry
0	0	0	-
0	1	1	-
1	0	1	-
1	1	0	1

In 1 + 1 = 10, is considered as sum 0 and the 1 as carry bit. This carry bit is added with the previous position of the bit pattern.

Example Add: $1011_2 + 1001_2$

$$\begin{array}{lccccc}
\text{(Carry Bit)}\rightarrow & & & 1 & 1 & \\
 & & 1 & 0 & 1 & 1 \\
+ & & 1 & 0 & 0 & 1 \\
\hline
 & 1 & 0 & 1 & 0 & 0
\end{array}
\quad \left.\right\} 1\ 0$$

$1011_2 + 1001_2 = 10100_2$

Example Perform Binary addition for the following: $23_{10} + 12_{10}$

Step 1: Convert 23 and 12 into binary form

23_{10}

2's power	16	8	4	2	1
Binary Number	1	0	1	1	1

$23_{10} = 00010111_2$

12_{10}

2's power	8	4	2	1
Binary Number	1	1	0	0

$12_{10} = 00001100_2$

Step 2: Binary addition of 23 and 12:

Carry Bit →			1	1	1		
23_{10} = 0	0	0	1	0	1	1	1
12_{10} = 0	0	0	0	1	1	0	0
35_{10} = 0	**0**	**1**	**0**	**0**	**0**	**1**	**1**

2.6.2 Binary Substraction

The table for Binary Substraction is as follows:

A	B	Difference (A-B)	Borrow
0	0	0	0
1	0	1	0
1	1	0	0
0	1	1	1

When substracting 1 from 0, borrow 1 from the next Most Significant Bit, when borrowing from the next Most Significant Bit, if it is 1, replace it with 0. If the next Most Significant Bit is 0, you must borrow from a more significant bit that contains 1 and replace it with 0 and 0s upto that point become 1s.

Example Subtract $1001010_2 - 10100_2$

$$\begin{array}{cccccccc}
 & 0 & 1 & 10 & 0 & 10 & & \\
 & \cancel{1} & \cancel{0} & 0 & \cancel{1} & 0 & 1 & 0 \\
(-) & & & 1 & 0 & 1 & 0 & 0 \\
\hline
 & & 1 & 1 & 0 & 1 & 1 & 0 \\
\hline
\end{array}$$

Example Perform binary addition for the following: $(-21)_{10} + (5)_{10}$

Step 1: Change -21 and 5 into binary form

21_{10}

2's power	16	8	4	2	1
Binary Number	1	0	1	0	1

$21_{10} = 00010101_2$

5_{10}

2's power	4	2	1
Binary Number	1	0	1

$5_{10} = 00000101_2$

Step 2:

21_{10}	0	0	0	1	0	1	0	1
1's Compliment	1	1	1	0	1	0	1	0
2's Compliment	1	1	1	0	1	0	1	1

Step 3:

Binary Addition of –21 and 5 :

Carry bit				1	1	1	1	
-21_{10}	1	1	1	0	1	0	1	1
5_{10}	0	0	0	0	0	1	0	1
-16_{10} (Result)	1	1	1	1	0	0	0	0

Workshop

8. Perform the following binary computations:

(A) $10_{10} + 15_{10}$ (B) $-12_{10} + 5_{10}$

(C) $14_{10} - 12_{10}$ (D) $(-2_{10}) - (-6_{10})$

2.7 Representing Characters in Memory

As represented in introduction, all the input data given to the computer should be in understandable format. In general, 26 uppercase letters, 26 lowercase letters, 0 to 9 digits and special characters are used in a computer, which is called character set. All these character set are denoted through numbers only. All Characters in the character set needs a common encoding system. There are several encoding systems used for computer. They are

- BCD – Binary Coded Decimal
- EBCDIC – Extended Binary Coded Decimal Interchange Code
- ASCII – American Standard Code for Information Interchange
- Unicode
- ISCII - Indian Standard Code for Information Interchange

2.7.1 Binary Coded Decimal (BCD)

This encoding system is not in the practice right now. This is 2^6 bit encoding system. This can handle $2^6 = 64$ characters only.

2.7.2 American Standard Code for Information Interchange (ASCII)

This is the most popular encoding system recognized by United States. Most of the computers use this system. Remember this encoding system can handle English characters only. This can handle 2^7 bit which means 128 characters.

In this system, each character has individual number (Refer **Appendix**).

The new edition (version) ASCII -8, has 2^8 bits and can handle 256 characters are represented from 0 to 255 unique numbers.

The ASCII code equivalent to the uppercase letter 'A' is 65. The binary representation of ASCII (7 bit) value is 1000001. Also 01000001 in ASCII-8 bit.

2.7.3 Extended Binary Coded Decimal Interchange Code (EBCDIC)

This is similar to ASCII Code with 8 bit representation. This coding system is formulated by International Business Machine(IBM). The coding system can handle 256 characters. The input code in ASCII can be converted to EBCDIC system and vice - versa.

2.7.4 Indian Standard Code for Information Interchange (ISCII)

ISCII is the system of handling the character of Indian local languages. This as a 8-bit coding system. Therefore it can handle 256 (2^8) characters. This system is formulated by the department of Electronics in India in the year 1986-88 and recognized by Bureau of Indian Standards (BIS). Now this coding system is integrated with Unicode.

2.7.5 Unicode

This coding system is used in most of the modern computers. The popular coding scheme after ASCII is Unicode. ASCII can represent only 256 characters. Therefore English and European Languages alone can be handled by ASCII. Particularly there was a situation, when the languages like Tamil, Malayalam, Kannada and Telugu could not be represented by ASCII. Hence, the Unicode was generated to handle all the coding system of Universal languages. This is 16 bit code and can handle 65536 characters.

Unicode scheme is denoted by hexadecimal numbers. The Unicode table of Tamil, Malayalam, Telugu and Kannada is shown in Table 2.6

Table 2.6

Unicode Table of Tamil

	0B8	0B9	0BA	0BB	0BC	0BD	0BE	0BF
0	▨	ஐ 0B90	▨	ர 0BB0	◌ீ 0BC0	ௐ 0BD0	▨	௰ 0BF0
1	▨	▨	▨	ற 0BB1	◌ு 0BC1	▨	▨	௱ 0BF1
2	◌ஂ 0B82	ஒ 0B92	▨	ல 0BB2	◌ூ 0BC2	▨	▨	௲ 0BF2
3	ஃ 0B83	ஓ 0B93	ண 0BA3	ள 0BB3	▨	▨	▨	௳ 0BF3
4	▨	ஔ 0B94	த 0BA4	ழ 0BB4	▨	▨	▨	௴ 0BF4
5	அ 0B85	க 0B95	▨	வ 0BB5	▨	▨	▨	௵ 0BF5
6	ஆ 0B86	▨	▨	ஶ 0BB6	◌ெ 0BC6	▨	௦ 0BE6	௶ 0BF6
7	இ 0B87	▨	▨	ஷ 0BB7	◌ே 0BC7	◌ௗ 0BD7	௧ 0BE7	௷ 0BF7
8	ஈ 0B88	▨	ந 0BA8	ஸ 0BB8	◌ை 0BC8	▨	௨ 0BE8	௸ 0BF8
9	உ 0B89	ங 0B99	ன 0BA9	ஹ 0BB9	▨	▨	௩ 0BE9	௹ 0BF9
A	ஊ 0B8A	ச 0B9A	ப 0BAA	▨	◌ொ 0BCA	▨	௪ 0BEA	௺ 0BFA
B	▨	▨	▨	▨	◌ோ 0BCB	▨	௫ 0BEB	▨
C	▨	ஜ 0B9C	▨	▨	◌ௌ 0BCC	▨	௬ 0BEC	▨
D	▨	▨	▨	▨	◌் 0BCD	▨	௭ 0BED	▨
E	எ 0B8E	ஞ 0B9E	ம 0BAE	◌ா 0BBE	▨	▨	௮ 0BEE	▨
F	ஏ 0B8F	ட 0B9F	ய 0BAF	◌ி 0BBF	▨	▨	௯ 0BEF	▨

Unicode Table of Malayalam

	0D0	0D1	0D2	0D3	0D4	0D5	0D6	0D7
0	◌ഀ 0D00	ഐ 0D10	ഠ 0D20	ര 0D30	◌ീ 0D40	▨	ൠ 0D60	൰ 0D70
1	◌ഁ 0D01	▨	ഡ 0D21	റ 0D31	◌ു 0D41	▨	ൡ 0D61	൱ 0D71
2	◌ം 0D02	ഒ 0D12	ഢ 0D22	ല 0D32	◌ൂ 0D42	▨	◌ൢ 0D62	൲ 0D72
3	◌ഃ 0D03	ഓ 0D13	ണ 0D23	ള 0D33	◌ൃ 0D43	▨	◌ൣ 0D63	൳ 0D73
4	▨	ഔ 0D14	ത 0D24	ഴ 0D34	◌ൄ 0D44	ൔ 0D54	▨	൴ 0D74
5	അ 0D05	ക 0D15	ഥ 0D25	വ 0D35	▨	ൕ 0D55	▨	൵ 0D75
6	ആ 0D06	ഖ 0D16	ദ 0D26	ശ 0D36	◌െ 0D46	ൖ 0D56	൦ 0D66	൶ 0D76
7	ഇ 0D07	ഗ 0D17	ധ 0D27	ഷ 0D37	◌േ 0D47	◌ൗ 0D57	൧ 0D67	൷ 0D77
8	ഈ 0D08	ഘ 0D18	ന 0D28	സ 0D38	◌ൈ 0D48	൘ 0D58	൨ 0D68	൸ 0D78
9	ഉ 0D09	ങ 0D19	ഩ 0D29	ഹ 0D39	▨	൙ 0D59	൩ 0D69	൹ 0D79
A	ഊ 0D0A	ച 0D1A	പ 0D2A	ഺ 0D3A	◌ൊ 0D4A	൚ 0D5A	൪ 0D6A	ൺ 0D7A
B	ഋ 0D0B	ഛ 0D1B	ഫ 0D2B	◌഻ 0D3B	◌ോ 0D4B	൛ 0D5B	൫ 0D6B	ൻ 0D7B
C	ഌ 0D0C	ജ 0D1C	ബ 0D2C	◌഼ 0D3C	◌ൌ 0D4C	൜ 0D5C	൬ 0D6C	ർ 0D7C
D	▨	ഝ 0D1D	ഭ 0D2D	ഽ 0D3D	◌് 0D4D	൝ 0D5D	൭ 0D6D	ൽ 0D7D
E	എ 0D0E	ഞ 0D1E	മ 0D2E	◌ാ 0D3E	ൎ 0D4E	൞ 0D5E	൮ 0D6E	ൾ 0D7E
F	ഏ 0D0F	ട 0D1F	യ 0D2F	◌ി 0D3F	൏ 0D4F	ൟ 0D5F	൯ 0D6F	ൿ 0D7F

Table 2.6

Unicode Table of Telugu

	0C0	0C1	0C2	0C3	0C4	0C5	0C6	0C7
0	◌ఀ 0C00	ఐ 0C10	ఠ 0C20	ర 0C30	◌ీ 0C40		ౠ 0C60	
1	◌ఁ 0C01		డ 0C21	ఱ 0C31	◌ు 0C41		ౡ 0C61	
2	◌ం 0C02	ఒ 0C12	ఢ 0C22	ల 0C32	◌ూ 0C42		◌ౢ 0C62	
3	◌ః 0C03	ఓ 0C13	ణ 0C23	ళ 0C33	◌ృ 0C43		◌ౣ 0C63	
4		ఔ 0C14	త 0C24	ఴ 0C34	◌ౄ 0C44			
5	అ 0C05	క 0C15	థ 0C25	వ 0C35		◌ౕ 0C55		
6	ఆ 0C06	ఖ 0C16	ద 0C26	శ 0C36	◌ె 0C46	◌ౖ 0C56	౦ 0C66	
7	ఇ 0C07	గ 0C17	ధ 0C27	ష 0C37	◌ే 0C47		౧ 0C67	
8	ఈ 0C08	ఘ 0C18	న 0C28	స 0C38	◌ై 0C48	ౘ 0C58	౨ 0C68	౸ 0C78
9	ఉ 0C09	ఙ 0C19		హ 0C39		ౙ 0C59	౩ 0C69	౹ 0C79
A	ఊ 0C0A	చ 0C1A	ప 0C2A		◌ొ 0C4A	ౚ 0C5A	౪ 0C6A	౺ 0C7A
B	ఋ 0C0B	ఛ 0C1B	ఫ 0C2B		◌ో 0C4B		౫ 0C6B	౻ 0C7B
C	ఌ 0C0C	జ 0C1C	బ 0C2C		◌ౌ 0C4C		౬ 0C6C	౼ 0C7C
D		ఝ 0C1D	భ 0C2D	ఽ 0C3D	◌్ 0C4D		౭ 0C6D	౽ 0C7D
E	ఎ 0C0E	ఞ 0C1E	మ 0C2E	◌ా 0C3E			౮ 0C6E	౾ 0C7E
F	ఏ 0C0F	ట 0C1F	య 0C2F	◌ి 0C3F			౯ 0C6F	౿ 0C7F

Unicode Table of Kannada

	0C8	0C9	0CA	0CB	0CC	0CD	0CE	0CF
0	ಀ 0C80	ಐ 0C90	ಠ 0CA0	ರ 0CB0	◌ೀ 0CC0		ೠ 0CE0	
1	◌ಁ 0C81		ಡ 0CA1	ಱ 0CB1	◌ು 0CC1		ೡ 0CE1	ೱ 0CF1
2	◌ಂ 0C82	ಒ 0C92	ಢ 0CA2	ಲ 0CB2	◌ೂ 0CC2		◌ೢ 0CE2	ೲ 0CF2
3	◌ಃ 0C83	ಓ 0C93	ಣ 0CA3	ಳ 0CB3	◌ೃ 0CC3		◌ೣ 0CE3	
4		ಔ 0C94	ತ 0CA4		◌ೄ 0CC4			
5	ಅ 0C85	ಕ 0C95	ಥ 0CA5	ವ 0CB5		◌ೕ 0CD5		
6	ಆ 0C86	ಖ 0C96	ದ 0CA6	ಶ 0CB6	◌ೆ 0CC6	◌ೖ 0CD6	೦ 0CE6	
7	ಇ 0C87	ಗ 0C97	ಧ 0CA7	ಷ 0CB7	◌ೇ 0CC7		೧ 0CE7	
8	ಈ 0C88	ಘ 0C98	ನ 0CA8	ಸ 0CB8	◌ೈ 0CC8		೨ 0CE8	
9	ಉ 0C89	ಙ 0C99		ಹ 0CB9			೩ 0CE9	
A	ಊ 0C8A	ಚ 0C9A	ಪ 0CAA		◌ೊ 0CCA		೪ 0CEA	
B	ಋ 0C8B	ಛ 0C9B	ಫ 0CAB		◌ೋ 0CCB		೫ 0CEB	
C	ಌ 0C8C	ಜ 0C9C	ಬ 0CAC	◌಼ 0CBC	◌ೌ 0CCC		೬ 0CEC	
D		ಝ 0C9D	ಭ 0CAD	ಽ 0CBD	◌್ 0CCD		೭ 0CED	
E	ಎ 0C8E	ಞ 0C9E	ಮ 0CAE	◌ಾ 0CBE		ೞ 0CDE	೮ 0CEE	
F	ಏ 0C8F	ಟ 0C9F	ಯ 0CAF	◌ಿ 0CBF			೯ 0CEF	

Appendix

American Standard Code for Information Interchange (ASCII)

(Few specific characters only)

Alphabets

Alphabets	Decimal number	Binary number (8 bit)	Octal number	Hexadecimal number
A	65	01000001	101	41
B	66	01000010	102	42
C	67	01000011	103	43
D	68	01000100	104	44
E	69	01000101	105	45
F	70	01000110	106	46
G	71	01000111	107	47
H	72	01001000	110	48
I	73	01001001	111	49
J	74	01001010	112	4A
K	75	01001011	113	4B
L	76	01001100	114	4C
M	77	01001101	115	4D
N	78	01001110	116	4E
O	79	01001111	117	4F
P	80	01010000	120	50
Q	81	01010001	121	51
R	82	01010010	122	52
S	83	01010011	123	53
T	84	01010100	124	54
U	85	01010101	125	55
V	86	01010110	126	56
W	87	01010111	127	57
X	88	01011000	130	58
Y	89	01011001	131	59
Z	90	01011010	132	5A
a	97	01100001	141	61
b	98	01100010	142	62
c	99	01100011	143	63
d	100	01100100	144	64
e	101	01100101	145	65

f	102	01100110	146	66
g	103	01100111	147	67
h	104	01101000	150	68
i	105	01101001	151	69
j	106	01101010	152	6A
k	107	01101011	153	6B
l	108	01101100	154	6C
m	109	01101101	155	6D
n	110	01101110	156	6E
o	111	01101111	157	6F
p	112	01110000	160	70
q	113	01110001	161	71
r	114	01110010	162	72
s	115	01110011	163	73
t	116	01110100	164	74
u	117	01110101	165	75
v	118	01110110	166	76
w	119	01110111	167	77
x	120	01111000	170	78
y	121	01111001	171	79
z	122	01111010	172	7A

Numerals

Alphabets	Decimal number	Binary number (8 bit)	Octal number	Hexadecimal number
0	48	00110000	60	30
1	49	00110001	61	31
2	50	00110010	62	32
3	51	00110011	63	33
4	52	00110100	64	34
5	53	00110101	65	35
6	54	00110110	66	36
7	55	00110111	67	37
8	56	00111000	70	38
9	57	00111001	71	39

Special Characters

Special symbols	Decimal number	Binary number (8 bit)	Octal number	Hexadecimal number
Blank	32	00100000	40	20
!	33	00100001	41	21
"	34	00100010	42	22
#	35	00100011	43	23
$	36	00100100	44	24
%	37	00100101	45	25
&	38	00100110	46	26
'	39	00100111	47	27
(	40	00101000	50	28
)	41	00101001	51	29
*	42	00101010	52	2A
+	43	00101011	53	2B
,	44	00101100	54	2C
-	45	00101101	55	2D
.	46	00101110	56	2E
/	47	00101111	57	2F
:	58	00111010	72	3A
;	59	00111011	73	3B
<	60	00111100	74	3C
=	61	00111101	75	3D
>	62	00111110	76	3E
?	63	00111111	77	3F
@	64	01000000	100	40
[	91	01011011	133	5B
\	92	01011100	134	5C
]	93	01011101	135	5D
^	94	01011110	136	5E
_	95	01011111	137	5F
`	96	01100000	140	60
{	123	01111011	173	7B
|	124	01111100	174	7C
}	125	01111101	175	7D
~	126	01111110	176	7E

Evaluation

CY5YK8

SECTION – A

Choose the correct answer:

1. Which refers to the number of bits processed by a computer's CPU?
   A) Byte B) Nibble C) Word length D) Bit
2. How many bytes does 1 KiloByte contain?
   A) 1000 B) 8 C) 4 D) 1024
3. Expansion for ASCII
   A) American School Code for Information Interchange
   B) American Standard Code for Information Interchange
   C) All Standard Code for Information Interchange
   D) American Society Code for Information Interchange
4. 2^50 is referred as
   A) Kilo B) Tera C) Peta D) Zetta
5. How many characters can be handled in Binary Coded Decimal System?
   A) 64 B) 255 C) 256 D) 128
6. For 1101_2 the equalent Hexadecimal equivalent is?
   A) F B) E C) D D) B
7. What is the 1's complement of 00100110?
   A) 00100110 B) 11011001 C) 11010001 D) 00101001
8. Which amongst this is not an Octal number?
   A) 645 B) 234 C) 876 D) 123

SECTION-B

Very Short Answers

1. What is data?
2. Write the 1's complement procedure.
3. Convert $(46)_{10}$ into Binary number
4. We cannot find 1's complement for $(28)_{10}$. State reason.
5. List the encoding systems that represents characters in memory.

SECTION-C

Short Answers

1. What is radix of a number system? Give example
2. Write note on binary number system.
3. Convert $(150)_{10}$ into Binary, then convert that Binary number to Octal
4. Write short note on ISCII
5. Add a) $-22_{10}+15_{10}$ b) $20_{10}+25_{10}$

SECTION - D

Explain in detail

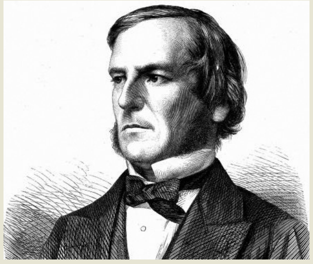
1. a) Write the procedure to convert fractional Decimal to Binary
   b) Convert $(98.46)_{10}$ to Binary
2. Find 1's Complement and 2's Complement for the following Decimal number
   a) -98 b) -135
3. a) Add $1101010_2+101101_2$ b) Subtract 1101011_2 - 111010_2

Part - II - Boolean Algebra

2.8 Introduction

Boolean algebra is a mathematical discipline that is used for designing digital circuits in a digital computer. It describes the relation between inputs and outputs of a digital circuit. The name Boolean algebra has been given in honor of an English mathematician George Boole who proposed the basic principles of this algebra.

DO YOU KNOW?



George Boole (1815-1864) was born to a low class family and only received an elementary school education. Despite that, he taught himself highly advanced mathematics and different languages as a teenager without any assistance. He started teaching at age sixteen, and started his own school at age nineteen. By his mid-twenties, he had mastered most of the important mathematical principles in his day.

2.8.1 Binary valued quantities:

Every day we have to make logical decisions:

1. Should I carry Computer Science book every day? Yes / No
2. 8-10 = 10 is this answer correct? Yes / No
3. Chennai is capital of India? Yes / No
4. What did I say yesterday?

The first three questions thrown above, the answer may be True (Yes) or False (No). But the fourth one, we cannot be answer as True or False. Thus, sentences which can be determined to be True or False are called "Logical Statement" or "Truth Functions". The results True or False are called "Truth Values". The truth values depicted by logical constant 1 and 0; 1 means True and 0 means False. The variable which can store these truth values are called "Logical variable" or "Binary valued variables" or "Boolean Variables" as these can store one of the two values of True or False.

2.8.2 Logical Operations:

Boolean algebra makes use of variables and operations (functions). The basic logical operations are AND, OR and NOT, which are symbolically represented by dot (.), plus (+), and by over bar / single apostrophe respectively. These symbols are also called as "Logical Operators".

2.8.3 Truth Table:

A truth table represents all the possible values of logical variable or statements along with all the possible results of given combination of truth values.

2.8.4 AND operator

The AND operator is defined in Boolean algebra by the use of the dot (.) operator. It is similar to multiplication in ordinary algebra. The AND operator combines two or more input variables so that the output is true only if all the inputs are true. The truth table for a 2-input AND operator is shown as follows:

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

The above 2-input AND operation is expressed as: Y = A . B

2.8.5 OR operator

The plus sign is used to indicate the OR operator. The OR operator combines two or more input variables so that the output is true if at least one input is true. The truth table for a 2-input OR operator is shown as follows:

BUBUIH

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

The above 2-input OR operation is expressed as: Y = A + B

2.8.6 NOT operator

The NOT operator has one input and one output. The input is either true or false, and the output is always the opposite, that is, the NOT operator inverts the input. The truth table for a NOT operator where A is the input variable and Y is the output is shown below:

A	Y
0	1
1	0

The NOT operator is represented algebraically by the Boolean expression: Y = A

Example:

Consider the Boolean equation:

D = A + (B . C)

D is equal to 1 (true) if A is 1 or if $(B . \underline{C})$ is 1, that is, B = 0 and C = 1.

Otherwise D is equal to 0 (false).

The basic logic functions AND, OR, and NOT can also be combined to make other logic operators such as NAND and NOR

2.8.7 NAND operator

The NAND is the combination of NOT and AND. The NAND is generated by inverting the output of an AND operator. The algebraic expression of the NAND function is:

$Y = \overline{A . B}$

The NAND function truth table is shown below:

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

A NAND B = NOT (A AND B)

2.8.8 NOR operator

The NOR is the combination of NOT and OR. The NOR is generated by inverting the output of an OR operator. The algebraic expression of the NOR function is:

$Y = \overline{A . B}$

The NOR function truth table is shown below:

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

A NOR B = NOT (A OR B)

2.9 Basic Logic Gates:

A gate is a basic electronic circuit which operates on one or more signals to produce an output signal. There are three fundamental gates namely AND, OR and NOT. The other logic gates like NAND, NOR, XOR and XNOR are derived gates which are derived from the fundamental gates. NAND and NOR gates are called Universal gates, because the fundamental logic gates can be realized through them.

2.9.1 AND Gate

The AND gate can have two or more input signals and produce an output signal.

The output is "true" only when both inputs are "true", otherwise, the output is "false". In other words the output will be 1 if and only if both inputs are 1; otherwise the output is 0. The output of the AND gate is represented by avariable say C, where A and B are two boolean variables. In boolean algebra, a variable can take either of the values '0' or '1'. The logical symobl of the AND gate is

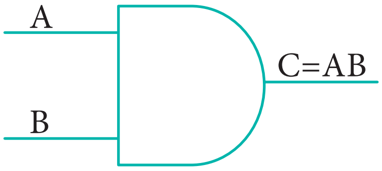


Fig. 2.4 Logic symbol of AND Gate

One way to symbolize the action of an AND gate is by writing the boolean function.

C = A AND B

In boolean algebra the multiplication sign stands for the AND operation. Therefore, the output of the AND gate is

C = A . B or

simply C = AB

Read this as "C equals A AND B". Since there are two input variables here, the truth table has four entries, because there are four possible inputs : 00, 01, 10 and 11.

For instance if both inputs are 0,

$$C = A . B$$
$$= 0 . 0$$
$$= 0$$

The truth table for AND Gate is

Input		Output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Table 2.7 Truth Table for AND Gate

2.9.2 OR Gate

The OR gate gets its name from its behaviour like the logical inclusive "OR". The output is "true" if either or both of the inputs are "true". If both inputs are "false" then the output is "false". In otherwords the output will be 1 if and only if one or both inputs are 1; otherwise, the output is 0. The logical symbol of the OR gate is

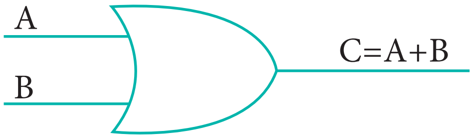


Fig. 2.5 Logic symbol of OR Gate

The OR gate output is

$$C = A \text{ OR } B$$

We use the + sign to denote the OR function. Therefore,

$$C = A + B$$

Read this as "C equals A OR B".

For instance, if both the inputs are 1

$$C = A + B = 1 + 1 = 1$$

The truth table for OR gate is

Input		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Table 2.8 Truth Table for OR Gate

2.9.3 NOT Gate

The NOT gate, called a logical inverter, has only one input. It reverses the logical state. In other words the output C is always the complement of the input. The logical symbol of the NOT gate is

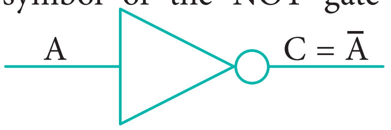


Fig. 2.6 Logic symbol of NOT Gate

The boolean function of NOT gate is

$$C = \text{NOT } A$$

In boolean algerbra, the overbar stands for NOT operation. Therefore,

$$C = \overline{A}$$

Read this as "C equals NOT A" or "C equals the complement of A".

If A is 0,

$$C = \overline{0} = 1$$

On the otherhand, if A is 1,

$$C = \overline{1} = 0$$

The truth table for NOT gate is

Input	Output
A	C
1	0
0	1

Table 2.9 Truth Table for NOT Gate

2.9.4 NOR Gate

The NOR gate circuit is an OR gate followed by an inverter. Its output is "true" if both inputs are "false" Otherwise, the output is "false". In other words, the only way to get '1' as output is to have both inputs '0'. Otherwise the output is 0. The logic circuit of the NOR gateis

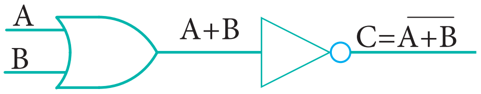


Fig. 2.7 Logic Circuit of NOR Gate

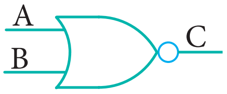


Fig. 2.8 Logic symbol of NOR Gate

The output of NOR gate is

$$C = (\overline{A + B})$$

Read this as "C equals NOT of A OR B" or "C equals the complement of A OR B". For example if both the inputs are 0,

$$C = (\overline{0 + 0}) = \overline{0} = 1$$

The truth table for NOR gate is

Input		Output
A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

Table 2.10 Truth Table for NOR Gate

2.9.5 Bubbled AND Gate

The Logic Circuit of Bubbled AND Gate

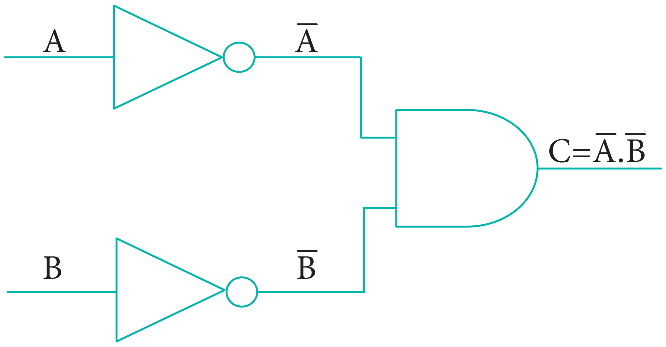


Fig. 2.9 Logic circuit of Bubbled AND Gate

In the above circuit, invertors on the input lines of the AND gate gives the output as

$$C = (\overline{A} \,.\, \overline{B})$$

This circuit can be redrawn as the bubbles on the inputs, where the bubbles represent inversion.

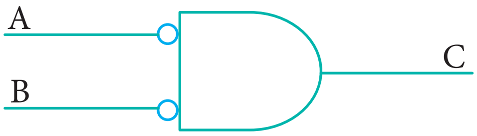


Fig. 2.10 Logic Symbol of Bubbled AND Gate

We refer this as bubbled AND gate. Let us analyse this logic circuit for all input possiblities.

If A = 0 and B = 0 C=($\overline{0}$.$\overline{0}$) = 1.1 = 1

If A = 0 and B = 1 C=($\overline{0}$.$\overline{1}$) = 1.0 = 0

If A = 1 and B = 0 C=($\overline{1}$.$\overline{0}$) = 0.1 = 0

If A = 1 and B = 1 C=($\overline{1}$.$\overline{1}$) = 0.0 = 0

Here the truth table is

Input		**Output**
A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

You can see that, a bubbled AND gate produces the same output as a NOR gate. So, You can replace each NOR gate by a bubbled AND gate. In other words the circuits are interchangeable.

Therefore

$$\overline{(A + B)} = \overline{A} \,.\, \overline{B}$$

Which establishes the De Morgan's first theorem.

2.9.6 NAND Gate

The NAND gate operates an AND gate followed by a NOT gate. It acts in the manner of the logical operation "AND" followed by inversion. The output is "false" if both inputs are "true", otherwise, the output is "true". In otherwords the output of the NAND gate is 0 if and only if both the inputs are 1, otherwise the output is 1. The logic circuit of NAND gate is

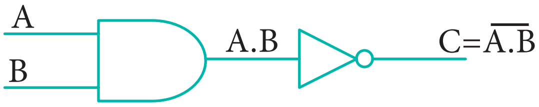


Fig. 2.11 Logic Circuit of NAND Gate

The logical symbol of NAND gate is

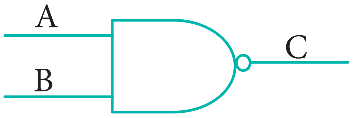


Fig. 2.12 Logic Symbol of NAND Gate

The output of the NAND gate is

$$C = (\overline{A \,.\, B})$$

Read this as "C" equals NOT of A AND B" or "C" equals the complement of A AND B".

For example if both the inputs are 1

$$C = \overline{(1\,.\,1)} = \overline{1} = 0$$

The truth table for NAND gate is

Input		Output
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

Table 2.11 Truth Table for NAND Gate

2.9.7 Bubbled OR Gate

The logic circuit of bubbled OR gate is

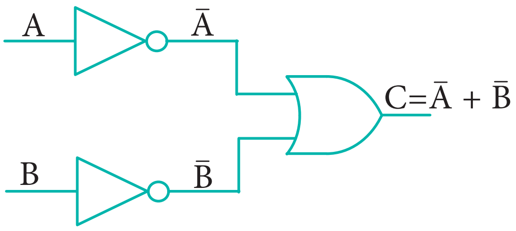


Fig. 2.13 Logic Circuit of Bubbled OR Gate

The output of this circuit can be written as $C = \overline{A} + \overline{B}$

The above circuit can be redrawn as the bubbles on the input, where the bubbles represents the inversion.

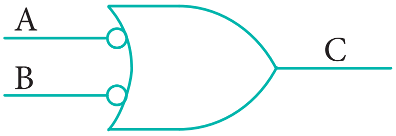


Fig. 2.14 Logic Symbol of Bubbled OR Gate

We refer this as bubbled OR gate. The truth table for the bubbled OR is

Input		Output
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

Table 2.12 Truth Table for Bubbled OR Gate

If we compare the truth tables of the bubbled OR gate with NAND gate, they are identical. So the circuits are interchangeable.

Therefore,

$$\overline{(A\,.\,B)} = \overline{A} + \overline{B}$$

Which establishes the De Morgan's second theorem.

2.9.8 XOR Gate

The XOR (exclusive - OR) gate acts in the same way as the logical "either/or." The output is "true" if either, but not both, of the inputs are "true". The output is "false" if both inputs are "false" or if both inputs are "true." Another way of looking at this circuit is to observe that the output is 1 if the inputs are different, but 0 if the inputs are the same. The logic circuit of XOR gate is

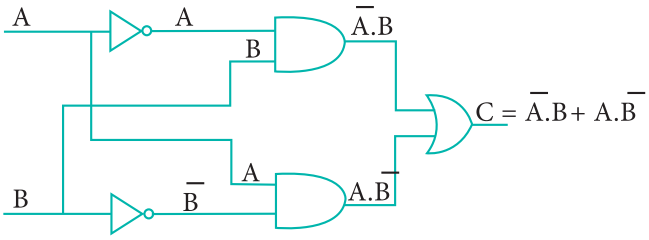


Fig. 2.15 Logic Circuit of XOR Gate

The output of the XOR gate is

The truth table for XOR gate is

Input		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Table 2.13 Truth Table for XOR Gate

In boolean algebra. exclusive - OR operator ⊕ or "encircled plus".

Hence $C = A \oplus B$

The logical symbol of XOR gate is

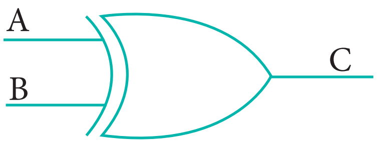


Fig. 2.16 Logic Symbol of XOR Gate

2.9.9 XNOR Gate

The XNOR (exclusive - NOR) gate is a combination XOR gate followed by an inverter. Its output is "true" if the inputs are the same, and "false" if the inputs are different. In simple words, the output is 1 if the input are the same, otherwise the output is 0. The logic circuit of XNOR gate is

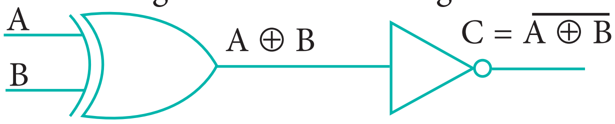


Fig. 2.17 Logic Circuit of XNOR Gate

The output of the XNOR is NOT of XOR

$$C = \overline{A \oplus B}$$

$$= \overline{A} . B + A . \overline{B}$$

$$= AB + \overline{A}\ \overline{B}$$

(Using De Morgan's Theorem)

In boolean algebra, ⊙ or "included dot" stands for the XNOR.

Therefore, $C = A \odot B$

The logical symbol is

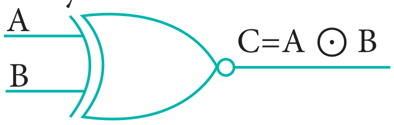


Fig. 2.18 Logic Symbol of XNOR Gate

The truth table for XNOR Gate is

Input		Output
A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

Table 2.14 Truth Table for XNOR Gate

Using combination of logic gates, complex operations can be performed. In theory, there is no limit to the number of gates that can be arranged together in a single device. But in practice, there is a limit to the number of gates that can be packed into a given physical space. Arrays of logic gates are found in digital integrated circuits.

Theorems of Boolean Algebra

Identity

$A + 0 = A$

$A \cdot 1 = A$

Complement

$A + \overline{A} = 1$

$A \cdot \overline{A} = 0$

Commutative

$A + B = B + A$

$A \cdot B = B \cdot A$

Associative

$A + (B + C) = (A + B) + C$

$A \cdot (B \cdot C) = (A \cdot B) \cdot C$

Distributive

$A \cdot (B + C) = A \cdot B + A \cdot C$

$A + (B \cdot C) = (A + B) \cdot (A + C)$

Null Element

$A + 1 = 1$

$A \cdot 0 = 0$

Involution

$\overline{(\overline{A})} = A$

Idempotence

$A + A = A$

$A . A = A$

Absorption

$A + (A \cdot B) = A$

$A \cdot (A + B) = A$

3rd Distributive

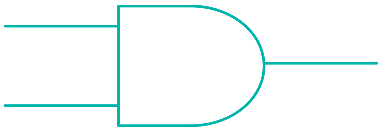
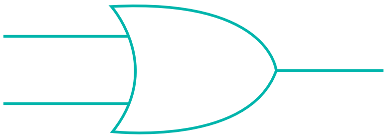
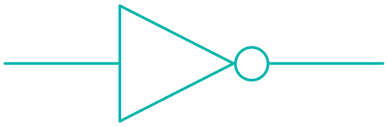
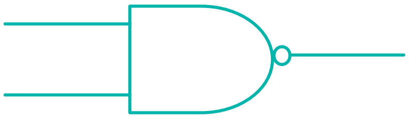
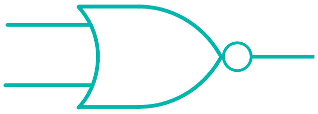
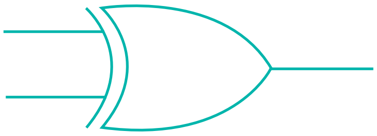
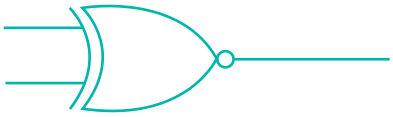
$A + \overline{A} \cdot B = A + B$

De Morgan's

$\overline{A + B} = \overline{A} \cdot \overline{B}$

$\overline{(A \cdot B)} = \overline{A} + \overline{B}$

Table 2. 15
Logic Gates and their corresponding Truth Tables

Logical Gates	Symbol	Truth Table		
AND		A	B	AB
		0	0	0
		0	1	0
		1	0	0
		1	1	1
OR		A	B	A + B
		0	0	0
		0	1	1
		1	0	1
		1	1	1
NOT		A	$\overline{A}$	
		0	1	
		1	0	
NAND		A	B	$\overline{A\ B}$
		0	0	1
		0	1	1
		1	0	1
		1	1	0
NOR		A	B	$\overline{A + B}$
		0	0	1
		0	1	0
		1	0	0
		1	1	0
XOR		A	B	$A \oplus B$
		0	0	0
		0	1	1
		1	0	1
		1	1	0
XNOR		A	B	$\overline{A \oplus B}$
		0	0	1
		0	1	0
		1	0	0
		1	1	1

Evaluation

SECTION – A

Choose the correct answer

1. Which is a basic electronic circuit which operates on one or more signals?

   (A) Boolean algebra (B) Gate

   (C) Fundamental gates (D) Derived gates

2. Which gate is called as the logical inverter?

   (A) AND (B) OR

   (C) NOT (D) XNOR

3. A + A = ?

   (A) A (B) O

   (C) 1 (D) A

4. NOR is a combination of ?

   (A) NOT(OR) (B)NOT(AND)

   (C) NOT(NOT) (D) NOT(NOR)

5. NAND is called as …… Gate

   (A) Fundamental Gate (B) Derived Gate

   (C) Logical Gate (D) Universal gate

BUKQK5

SECTION-B

Very Short Answers

1. What is Boolean Algebra?
2. Write a short note on NAND Gate.
3. Draw the truth table for XOR gate.
4. Write the associative laws?
5. What are derived gates?

SECTION-C

Short Answers

1. Write the truth table of fundamental gates.
2. Write a short note on XNOR gate.
3. Reason out why the NAND an NOR are called universal gates?
4. Give the truth table of XOR gate.
5. Write the De Morgan's law.

SECTION - D

Explain in detail

1. Explain the fundamental gates with expression and truth table.
2. How AND and OR can be realized using NAND and NOR gate.
3. Explain the Derived gates with expression and truth table.

Unit I | Fundamentals of Computers

CHAPTER 3

Computer Organisation

Learning Objectives

- To know the organisation of the computer components and their interconnections.
- To know the processors and their characteristics.
- To know the importance of memory devices and their roles in a computer.
- To explore RAM, ROM and differentiate each of them.
- To know about cache memory and how it improves the performance of a computer
- To know the secondary devices and their usage
- To know about the ports and interfaces so that external devices can be connected

3.1 Introduction

Computer organisation deals with the hardware components of a computer system. It includes Input / Output devices, the Central Processing Unit, storage devices and primary memory. It is concerned with how the various components of computer hardware operate. It also deals with how they are interconnected to implement an architectural specification. The term computer organisation looks similar to the term computer architecture. But, computer architecture deals with the engineering considerations involved in designing a computer. On the other hand, Computer Organisation deals with the hardware components that are transparent to the programmer.

3.2. Basics of Microprocessors

The CPU is the major component of a computer, which performs all tasks. This is realized by the microprocessor which is an Integrated Circuit. Microprocessors were first introduced in early 1970s. The first general purpose microprocessor, 4004 was developed by Intel Inc.

The microprocessor is a programmable multipurpose silicon chip. It is driven by clock pulses. It accepts input as a binary data and after processing, it provides the output data as per the instructions stored in the memory. A block diagram of a microprocessor based system is shown in Figure 3.1.

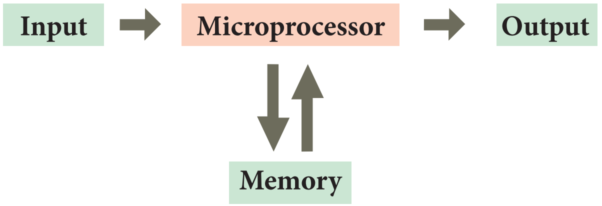


Figure 3.1 A Microprocessor - Based System

The microprocessor is made up of 3 main units. They are:

- **Arithmetic and Logic unit (ALU):** To perform arithmetic and logical instructions based on computer instructions.

- **Control unit:** To control the overall operations of the computer through signals.
- **Registers (Internal Memory):** They are used to hold the instruction and data for the execution of the processor.

Characteristics of Microprocessors

A Microprocessor's performance depends on the following characteristics:

a) Clock speed

b) Instruction set

c) Word size

a) Clock Speed

Every microprocessor has an **internal clock** that regulates the speed at which it executes instructions. The speed at which the microprocessor executes instructions is called the **clock speed**. Clock speed is measured in MHz (Mega Hertz) or in GHz (Giga Hertz).

b) Instruction Set

A command which is given to a computer to perform an operation on data is called an **instruction**. Basic set of machine level instructions that a microprocessor is designed to execute is called as an **instruction set**. This instruction set carries out the following types of operations:

- Data transfer
- Arithmetic operations
- Logical operations
- Control flow
- Input/output

c) Word Size

- The number of bits that can be processed by a processor in a single instruction is called its word size. **Word size** determines the amount of RAM that can be accessed by a microprocessor.

3.3 Data communication between CPU and memory

The Central Processing Unit(CPU) has a Memory Data Register (MDR) and a Memory Address Register (MAR). The Memory Data Register (MDR) keeps the data which is transferred between the Memory and the CPU. The Program Counter (PC) is a special register in the CPU which always keeps the address of the next instruction to be

Speed Measurement

DO YOU KNOW?

Hertz – abbreviated as Hz is the standard unit of measurement used for measuring frequency. Since frequency is measured in cycles per second, one hertz equals one cycle per second.

Hertz is commonly used to measure wave frequencies, such as sound waves, light waves, and radio waves. For example, the average human ear can detect sound waves between 20 and 20,000 Hz. Sound waves close to 20 Hz have a low pitch and are called "bass" frequencies. Sound waves above 5,000 Hz have a high pitch and are called "treble" frequencies.

While hertz can be used to measure wave frequencies, it is also used to measure the speed of computer processors. For example, each CPU is rated at a specific clock speed. This number indicates how many instruction cycles the processor can perform in every second. Since modern processors can perform millions or even billions of instructions per second, clock speeds are typically measured in megahertz or gigahertz.

executed. The Arithmetic and Logic unit of CPU places the address of the memory to be fetched, into the Memory Address Register.

A bus is a collection of wires used for communication between the internal components of a computer.

The word in the RAM has the same size (no. of bits) as the Memory Data Register (MDR). If the processor is an 8-bit processor like Intel 8085, its MDR and the word in the RAM both have 8 bits.

The read operation transfers the data(bits) from word to Memory Data Register. The write operation transfers the data(bits) from Memory Data Register to word.

DO YOU KNOW?

If 5V is applied at one end of a wire, the other end also can receive 5V. In the same way, the buses are wires, and the binary data are voltages (5V as 1 and 0V as 0), and these buses can simply pass the data as voltages from one end to other.

3.4 Types of Microprocessors

Microprocessors can be classified based on the following criteria:

- The width of data that can be processed
- The instruction set

3.4.1 Classification of Microprocessors based on the Data Width

Depending on the data width, microprocessors can process instructions. The microprocessors can be classified as follows:

- 8-bit microprocessor
- 16-bit microprocessor
- 32-bit microprocessor
- 64-bit microprocessor

3.4.2 Classification of Microprocessors based on Instruction Set

The size of the instruction set is important consideration while categorizing microprocessors.There are two types of microprocessors based on their instruction sets.

- Reduced Instruction Set Computers (RISC)
- Complex Instruction Set Computers (CISC)

Examples of RISC processors are Pentium IV, Intel P6, AMD K6 and K7.

Examples of CISC processors are Intel 386 & 486, Pentium, Pentium II and III, and Motorola 68000.

3.5 Memory Devices

A memory is just like a human brain. It is used to store data and instructions. Computer memory is the storage space in the computer, where data and instructions are stored. There are two types of accessing methods to access (read or write) the memory. They are sequential access and random access. In sequential access, the memory is accessed in an orderly manner from starting to end. But, in random access, any byte of memory can be accessed directly without navigating through previous bytes. Different memory devices are arranged according to the capacity, speed and cost as shown in Figure 3.6.

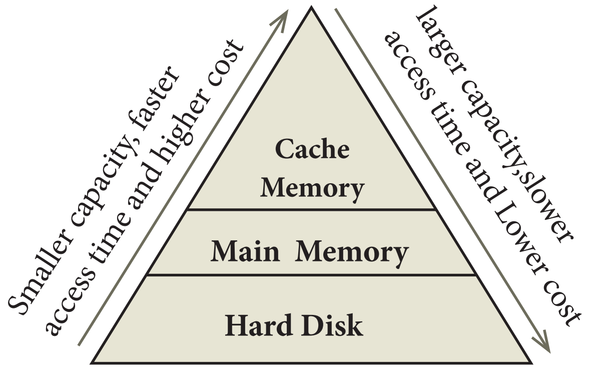

Smaller capacity, faster access time and higher cost
Cache Memory
Main Memory
Hard Disk
larger capacity,slower access time and Lower cost

Figure 3.6 Memory Hierarchy

3.5.1 Random-Access Memory (RAM)

The main memory is otherwise called as **Random Access Memory**. This is available in computers in the form of Integrated Circuits (ICs). It is the place in a computer where the Operating System, Application Programs and the data in current use are kept temporarily so that they can be accessed by the computer's processor. The smallest unit of information that can be stored in the memory is called as a bit. The memory can be accessed by a collection of 8 bits which is called as a byte.

RAM is a volatile memory, which means that the information stored in it is not permanent. As soon as the power is turned off, whatever data that resides in RAM is lost. It allows both read and write operations.

3.5.2 Types of RAM

There are two basic types of RAM

- Dynamic RAM (DRAM)
- Static RAM (SRAM)

These two types differ in the technology they use to hold data. Dynamic RAM being a common type needs to be refreshed frequently. Static RAM needs to be refreshed less often, which makes it faster. Hence, Static RAM is more expensive than Dynamic RAM.

3.5.3 Read Only Memory (ROM)

Read Only Memory refers to special memory in a computer with pre-recorded data at manufacturing time which cannot be modified. The stored programs that start the computer and perform diagnostics are available in ROMs. ROM stores critical programs such as the program that boots the computer. Once the data has been written onto a ROM chip, it cannot be modified or removed and can only be read. ROM retains its contents even when the computer is turned off. So, ROM is called as a non-volatile memory.

3.5.3.1Programmable Read Only Memory (PROM)

Programmable read only memory is also a non-volatile memory on which data can be written only once. Once a program has been written onto a PROM, it remains there forever. Unlike the main memory, PROMs retain their contents even when the computer is turned off.

The PROM differs from ROM. PROM is manufactured as a blank memory, whereas a ROM is programmed during the manufacturing process itself. PROM programmer or a PROM burner is used to write data to a PROM chip. The process of programming a PROM is called burning the PROM.

3.5.3.2 Erasable Programmable Read Only Memory (EPROM)

Erasable Programmable Read Only Memory is a special type of memory which serves as a PROM, but the content can be erased using ultraviolet rays. EPROM retains its contents until it is exposed to ultraviolet light. The ultraviolet light clears its contents, making it possible to reprogram the memory.

An EPROM differs from a PROM, PROM can be written only once and cannot be erased. EPROMs are used widely in personal computers because they enable the manufacturer to change the contents of the PROM to replace with updated versions or erase the contents before the computer is delivered.

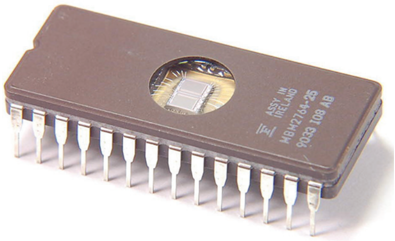


Figure 3.7 Erasable Programmable Read Only Memory

DO YOU KNOW?

Most of the EPROM chips have a transparent area at the top surface which is covered by stickers. If it gets removed, the ultraviolet light in the sunlight may erase the contents.

3.5.3.3 Electrically Erasable Programmable Read Only Memory (EEPROM)

Electrically Erasable Programmable Read Only Memory is a special type of PROM that can be erased by exposing it to an electrical charge. Like other types of PROM, EEPROM retains its contents even when the power is turned off. Comparing with all other types of ROM, EEPROM is slower in performance.

3.5.4 Cache Memory

The cache memory is a very high speed and expensive memory, which is used to speed up the memory retrieval process. Due to its higher cost, the CPU comes with a smaller size of cache memory compared with the size of the main memory. Without cache memory, every time the CPU requests the data, it has to be fetched from the main memory which will consume more time. The idea of introducing a cache is that, this extremely fast memory would store data that is frequently accessed and if possible, the data that is closer to it. This helps to achieve the fast response time, Where response Time, (Access Time) refers to how quickly the memory can respond to a read / write request. Figure 3.8 shows the arrangement of cache memory between the CPU and the main memory.

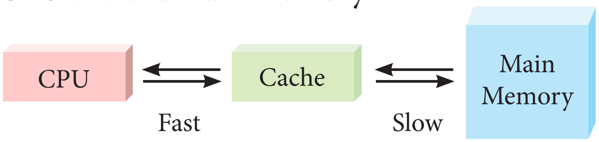


Figure 3.8 Cache Memory Arrangement

3.6 Secondary Storage Devices

A computer generally has limited amount of main memory which is expensive and volatile. To store data and programs permanently, secondary storage devices are used. Secondary storage devices serve as a supportive storage to main memory and they are non-volatile in nature, secondary storage is also called as Backup storage

3.6.1 Hard Disks

Hard disk is a magnetic disk on which you can store data. The hard disk has the stacked arrangement of disks accessed by a pair of heads for each of the disks. The hard disks come with a single or double sided disk.

3.6.2 Compact Disc (CD)

A CD or CD-ROM is made from 1.2 millimeters thick, polycarbonate plastic material. A thin layer of aluminium or gold is applied to the surface. CD data is represented as tiny indentations known as "pits", encoded in a spiral track moulded into the top of the polycarbonate layer. The areas between pits are known as "lands". A motor within the CD player rotates the disk. The capacity of an ordinary CD-ROM is 700MB.

Fig 3.9 Compact Disc

3.6.3 Digital Versatile Disc (DVD)

A **DVD (Digital Versatile Disc or Digital Video Disc)** is an optical disc capable of storing up to 4.7 GB of data, more than six times what a CD can hold. DVDs are often used to store movies at a better quality. Like CDs, DVDs are read with a laser.

The disc can have one or two sides, and one or two layers of data per side; the number of sides and layers determines how much it can hold. Double-layered sides are usually gold-coloured, while single-layered sides are usually silver-coloured, like a CD.

Fig 3.10 Digital Versatile Disc

3.6.4 Flash Memory Devices

Flash memory is an electronic (solid-state) non-volatile computer storage medium that can be electrically erased and reprogrammed. They are either EEPROM or EPROM. Examples for Flash memories are pendrives, memory cards etc. Flash memories can be used in personal computers, Personal Digital Assistants (PDA), digital audio players, digital cameras and mobile phones. Flash memory offers fast access times. The time taken to read or write a character in memory is called access time. The capacity of the flash memories vary from 1 Gigabytes (GB) to 2 Terabytes (TB). A sample of flash memory is shown in Figure 3.11.

Figure 3.11 Flash Memory

3.6.5 Blu-Ray Disc

Blu-Ray Disc is a high-density optical disc similar to DVD. Blu-ray is the type of disc used for PlayStation games and for playing High-Definition (HD) movies. A double-layer Blu-Ray disc can store up to 50GB (gigabytes) of data. DVD uses a red laser to read and write data. But, Blu-ray uses a blue-violet laser to write. Hence, it is called as Blu-Ray.

Blu-ray Disc TM
DEMO DISC
Disque à usage promotionnel
Location, vente et redistribution interdites
Réf. Y16898

Fig 3.12 Blu- Ray Disc

3.7 Ports and Interfaces

The Motherboard of a computer has many I/O sockets that are connected to the ports and interfaces found on the rear side of a computer (Figure 3.13). The external devices can be connected to the ports and interfaces. The various types of ports are given below:

Serial Port: To connect the external devices, found in old computers.

Parallel Port: To connect the printers, found in old computers.

USB Ports: To connect external devices like cameras, scanners, mobile phones, external hard disks and printers to the computer.

USB 3.0 is the third major version of the Universal Serial Bus (USB) standard to connect computers with other electronic gadgets as shown in Figure 3.13. USB 3.0 can transfer data up to 5 Giga byte/second. USB3.1 and USB 3.2 are also released.

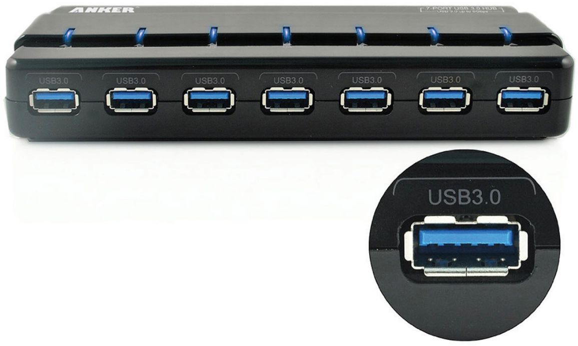


Figure 3.13 USB 3.0 Ports

VGA Connector: To connect a monitor or any display device like LCD projector.

Audio Plugs: To connect sound speakers, microphone and headphones.

PS/2 Port: To connect mouse and keyboard to PC.

SCSI Port: To connect the hard disk drives and network connectors.

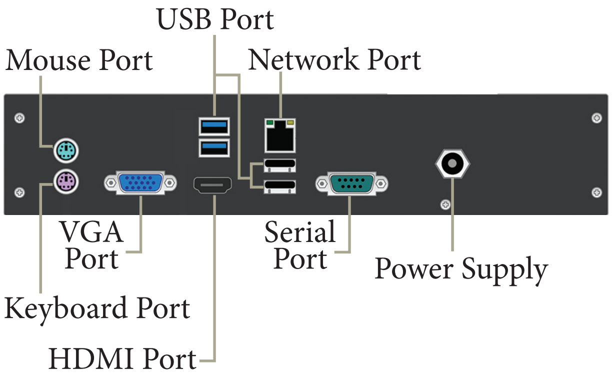


Fig 3.14 Ports and Interfaces

High Definition Multimedia Interface (HDMI)

High-Definition Multimedia Interface is an audio/video interface which transfers the uncompressed video and audio data from a video controller, to a compatible computer monitor, LCD projector, digital television etc.

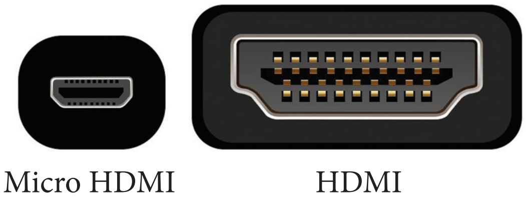


Figure 3.15 HDMI Ports

Activity

Student Activity

- Identify the components of a computer
- Connecting external devices like printer/LCD projector.

Teacher Activity

- Show the components of a computer
- Display different ROM ICs
- Display the flash memory
- Demonstrate various ports and their usage

Evaluation

SECTION – A

Choose the correct answer

1. Which of the following is said to be the brain of a computer?

   (a) Input devices
   (b) Output devices
   (c) Memory device
   (d) Microprocessor

2. Which of the following is not the part of a microprocessor unit?

   (a) ALU (b) Control unit
   (c) Cache memory (d) register

3. How many bits constitute a word?

   (a) 8
   (b) 16
   (c) 32
   (d) determined by the processor used.

4. Which of the following device identifies the location when address is placed in the memory address register?

   (a) Locator (b) encoder
   (c) decoder (d) multiplexer

5. Which of the following is a CISC processor?

   (a) Intel P6 (b) AMD K6
   (c) Pentium III (d) Pentium IV

6. Which is the fastest memory?

   (a) Hard disk
   (b) Main memory
   (c) Cache memory
   (d) Blue-Ray disc

7. How many memory locations are identified by a processor with 8 bits address bus at a time?

   (a) 28 (b) 1024
   (c) 256 (d) 8000

8. What is the capacity of 12cm diameter DVD with single sided and single layer?

   (a) 4.7 GB (b) 5.5 GB
   (c) 7.8GB (d) 2.2 GB

9. What is the smallest size of data represented in a CD?

   (a) blocks (b) sectors
   (c) pits (d) tracks

10. Display devices are connected to the computer through.

    (a) USB port
    (b) Ps/2 port
    (c) SCSI port
    (d) VGA connector

SECTION-B

Very Short Answers

(1) What are the parameters which influence the characteristics of a microprocessor?

(2) What is an instruction?

(3) What is a program counter?

(4) What is HDMI?

(5) Which source is used to erase the content of a EPROM?

SECTION-C

Short Answers

(1) Differentiate Computer Organisation from Computer Architecture.

(2) Classify the microprocessor based on the size of the data.

(3) Write down the classifications of microprocessors based on the instruction set.

(4) Differentiate PROM and EPROM.

(5) Write down the interfaces and ports available in a computer.

(6) Differentiate CD and DVD

(7) How will you differentiate a flash memory and an EEPROM?

SECTION - D

Explain in detail

(1) Explain the characteristics of a microprocessor.

(2) How the read and write operations are performed by a processor? Explain.

(3) Arrange the memory devices in ascending order based on the access time.

(4) Explain the types of ROM.

CYXLQ6

A-Z
GLOSSARY

Computer hardware	The physical parts or components of a computer, such as the CPU, mother board, monitor, keyboard, etc.
Intel	Intel Corporation is an American multinational corporation and technology company involving in hardware manufacturing, especially mother board and processors
Silicon chip	Silicon chip is an integrated , set of electronic circuits on one small flat piece of semiconductor material, silicon.
Multipurpose	Multipurpose is several purpose
Address bus	Address bus is a collection of wires that carry the address as bits
Data bus	Data bus is a collection of wires to carry data in bits
Control bus	Control bus is a control line/collection of wires to control the operations/functions
Arithmetic operations	Arithmetic operations are the mathematical operations on data like add, subtract etc
Data Transfer	Data Transfer means moving data from one component to another
Logical operations	Logical operations are the operations on binary/Boolean data like AND, OR , NOT
Bidirectional	Bidirectional means both the directions/ways
Unidirectional	Unidirectional means only one direction
Access time	Access time is the time delay or latency between a request to an electronic system, and the access being completed or the requested data returned

Theoretical concepts of Operating System

Learning objectives

- ✓ To know the concept of Operating Systems and their types.
- ✓ To acquire the basic Knowledge of Operating Systems and its functions.

4.1 Introduction to Software

A software is set of instructions that perform specific task. It interacts basically with the hardware to generate the desired output.

4.1.1 Types of Software

Software is classified into two types:

1) Application Software

2) System Software

CZ7HRT

Application Software:

Application software is a set of programs to perform specific task. For example MS-word is an application software to create text document and VLC player is familiar application software to play audio, video files and many more.

System Software:

System software is a type of computer program that is designed to run the computer's hardware and application programs. Example Operating System and Language Processor

4.2 Introduction to Operating System (OS):

An Operating System (OS) is a system software which serves as an interface between a user and a computer. This controls input, output and other peripheral devices such as disk drives, printers and electronic gadgets. The functions of an Operating System include file management, memory management, process management and device management and many more.

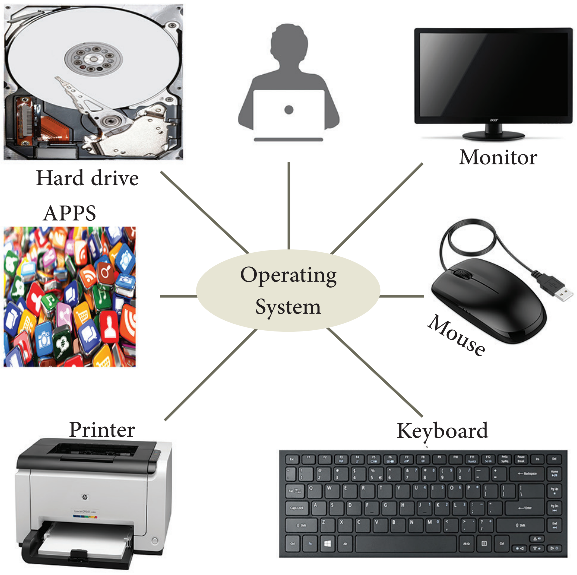


Figure: 4.1 Operating System

Without an Operating System, a computer cannot effectively manage all the resources. When a computer is switched on, the operating system is loaded in to the memory automatically.

Some of the popular Operating Systems used in personal computers and laptops are **Windows**, **UNIX** and **Linux**. The mobile devices mostly use Android and ioS as mobile **OS.**

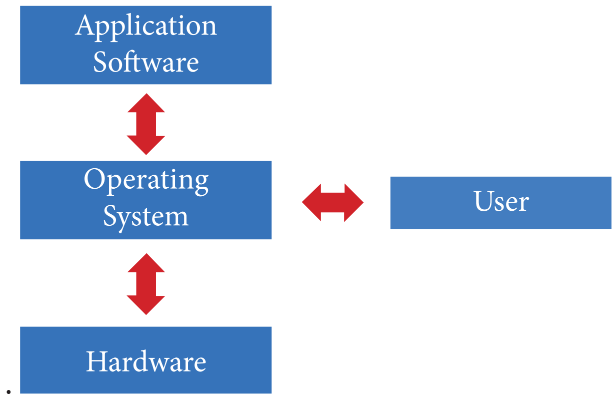


Figure: 4.2 Interaction of Operating system and user

Uses of Operating Systems

The following are few uses of Operating System

The main use of Operating System is

- to ensure that a computer can be used to extract what the user wants it do.
- Easy interaction between the users and computers.
- Starting computer operation automatically when power is turned on (Booting).
- Controlling Input and Output Devices
- Manage the utilisation of main memory.
- Providing security to user programs.

4.3 Types of Operating System

Operating System are classified into the following types depending on their processing capabilities.

4.3.1 Single User Operating Systems

An operating system allows only a single user to perform a task at a time. It is called as a Single user and single Task operating system.MS-DOS is an example for a single user and single task Operating System.

4.3.2 Multi-user Operating Systems

It is used in computers and laptops that allow same data and applications to be accessed by multiple users at the same time. The users can also communicate with each other. Windows, Linux and UNIX are examples for multi-user Operating System.

4.4 Key features of the Operating System

The various key features are given below

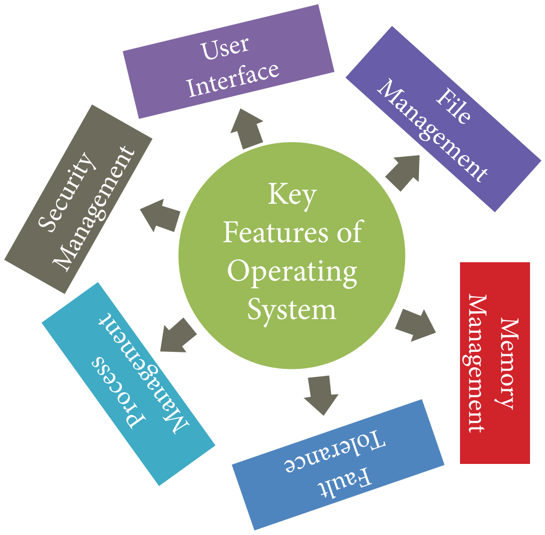


Figure: 4.3 Key Features of the Operating System

4.4.1 User Interface (UI)

User interface is one of the significant feature in Operating System. The only way that user can make interaction with a computer. This is a main reason for key success of GUI (Graphical User Interface) based Operating System. The GUI is a window based system with a pointing device to direct I/O, choose from menus, make selections and a keyboard to enter text.Its vibrant colours attract the user very easily.

Now Linux distribution is also available as GUI based Operating System. The following points are considered when User Interface is designed for an application.

1. The user interface should enable the user to retain this expertise for a longer time.
2. The user interface should also satisfy the customer based on their needs.
3.The user interface should save user's precious time.
4. The ultimate aim of any product is to satisfy the customer. The User Interface is also to satisfy the customer.
5. The user interface should reduce number of errors committed by the user

4.4.2 Memory Management

Memory Management is the process of controlling and coordinating computer's main memory and assigning memory block (space) to various running programs to optimize overall computer performance. The Memory management involves the allocation of specific memory blocks to individual programs based on user demands.

The objective of Memory Management process is to improve both the utilization of the CPU and the speed of the computer's response to its users via main memory. For these reasons the computers must keep several programs in main memory that associates with many different Memory Management schemes.

The Operating System is responsible for the following activities in connection with memory management:

- Keeping track of which portion of memory are currently being used and who is using them.
- Determining which processes (or parts of processes) and data to move in and out of memory.
- Allocation and de-allocation of memory blocks as needed by the program in main memory. (Garbage Collection)

4.4.3 Process management

Process management is function that includes creating and deleting processes(program) and providing mechanisms for processes to communicate and synchronize with each other.

. A system task, such as sending output to a printer or screen, can also be called as a Process.

A computer consists of a collection of processes, they are classified as two categories:

- Operating System processes which is executed by system code
- User Processes which is execute by user code

All these processes can potentially execute concurrently on a single CPU.

The following algorithms are mainly used to allocate the job (process) to the processor.

1. FIFO 2. SJF 3. Round Robin

4. Based on Priority

FIFO (First In First Out)Scheduling:

This algorithm is based on queuing technique. Assume that a student is standing in a queue (Row) to get grade sheet from his/her teacher. The other student who stands first in the queue gets his/her grade sheet first and leaves from the queue (Row). Followed by the next student in the queue gets it corrected and so on. This is the basic logic of the FIFO algorithm.

Technically, the process that enters the queue first is executed first by the CPU, followed by the next and so on. The processes are executed in the order of the queue (row).

SJF (Shortest Job First)Scheduling:

This algorithm works based on the size of the job being executed by the CPU.

Consider two jobs A and B.

1) A = 6 kilo bytes 2) B = 9 kilo bytes

First the job "A" will be assigned and then job "B" gets its turn.

Round RobinScheduling

The Round Robin (RR) scheduling algorithm is designed especially for time sharing systems. Jobs (processes) are assigned and processor time in a circular method. For example take three jobs A, B, C. First the job A is assigned to CPU then job B and job C and then again A, B and C and so on.

Based On Priority

The given job (process) is assigned based on a Priority. The job which has higher priority is more important than other jobs. Take two jobs A and B. Let the priority of A be 5 and priority B be 7.

Job B is assigned to the processor before job A.

4.4.4 Security Management

The major challenge in computer and software industry is to protect user's legitimate data from hackers. The Operating System provides three levels of securities to the user end. They are

(1) File access level

(2) System level

(3) Network level

In order to access the files created by other people, you should have the access permission. Permissions can either be granted by the creator of the file or bythe administrator of the system.

System level security is offered by the password in a multi-user environment.

Both windows and Linux offer the password facility.

Network security is an indefinable one. So people from all over the world try to provide such a security.

All the above levels of security features are provided only by the Operating System.

4.4.5 Fault Tolerance

The Operating Systems should be robust. When there is a fault, the Operating System should not crash, instead the Operating System have fault tolerance capabilities and retain the existing state of system.

4.4.6 File Management

File management is an important function of OS which handles the data storage techniques. The operating System manages the files, folders and directory systems on a computer.The FAT(File Allocation Table) stores general information about files like filename, type (text or binary), size, starting address and access mode.The file manager of the operating system helps to create, edit, copy, allocate memory to the files and also updates the FAT. There are few other file management techniques available like Next Generation File System (NTFS) and ext2(Linux).

4.4.7 Multi-Processing

This is a one of the features of Operating System. It has two or more processors for a single running process (job). Processing takes place in parallel is known as parallel processing.Since the execution takes place in parallel, this feature is used for high speed execution which increases the power of computing.

4.4.8 Time-sharing

This is a one of the features of Operating Systems. It allows execution of multiple tasks or processes concurrently. For each task a fixed time is allocated. This division of time is called Time- sharing. The processor switches rapidly between various processes after a time is elapsed or the process is completed.

For example assume that there are three processes called P1, P2, P3 and time allocated for each process 30, 40, 50 minutes respectively. If the process P1 completes within 20 minutes then processor takes the next process P2 for the execution. If the process P2 could not complete within 40 minutes, then the current process P2 will be paused and switch over to the next process P3.

4.4.9 Distributed Operating Systems

The Distributed Operating System is used to access shared data and files that reside in any machine around the world using internet/intranet.The users can access as if it is available on their own computer.

The advantages of distributed Operating System are as follows:

- A user at one location can make use of all the resources available at another location over the network.
- Many computer resources can be added easily in the network
- Improves the interaction with the customers and clients.
- Reduces the load on the host computer.

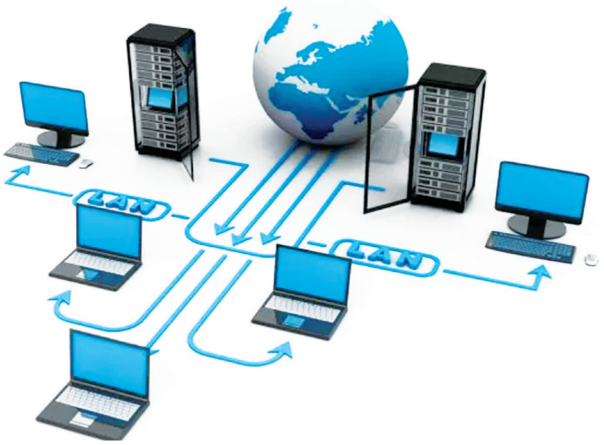


Figure: 4.4 Distributed Operating Systems

4.5 Prominent Operating Systems

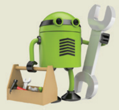
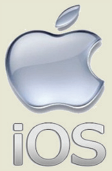
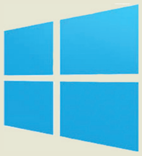
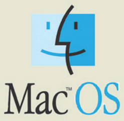
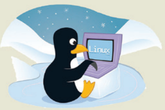
Prominent OS are as follows:

- UNIX
- Microsoft Windows
- Linux
- iOS
- Android

Modern operating systems use a Graphical User Interface(GUI). A GUI lets use to your mouse to click icons, buttons, menus and everything, is clearly displayed on the screen using a combination of graphics and text elements.

Student Activity

Activity 1: Draw a line between the operating system logo and the correct description.

A command-line operating system is an example of Open Source software development and Free Operating System	
A popular Operating System for mobile phone technology which is not linked with Apple products.	
Used with Apple computers and works well with cloud computing.	
Designed to be used for the Apple iPhone	
Is an Operating System that is very popular in universities, companies, big enterprises etc	
The most popular GUI Operating System for personal computers.	

Evaluation

SECTION – A

Choose the correct answer

1) Operating system is a
   A)Application Software
   B) Hardware
   C)System Software
   D)Component

CZGDTG

2) Identify the usage of Operating Systems
   A) Easy interaction between the human and computer
   B) Controlling input & output Devices
   C) Managing use of main memory
   D) All the above

3) Which of the following is not a function of an Operating System?
   A)Process Management
   B)Memory Management
   C)Security management
   D)Complier Environment

4) Which of the following OS is a Commercially licensed Operating system?
   A)Windows B)UBUNTU
   C)FEDORA D)REDHAT

5) Which of the following Operating systems support Mobile Devices?
   A)Windows 7 B)Linux
   C)BOSS D)iOS

6) File Management manages
   A) Files
   B) Folders
   C) Directory systems
   D) All the Above

7) Interactive Operating System provides

A)Graphics User Interface (GUI)

B)Data Distribution

C)Security Management

D)Real Time Processing

8) An example for single task operating system is

A)Linux

B) Windows

C)MS-DOS

D) Unix

9) The File management system used by Linux is

A) ext2

B) NTFS

C) FAT

D) NFTS

SECTION-B

Very Short Answers

1) List out any two uses of Operating System?

2) What is multi-user Operating system?

3) What is a GUI?

4) What are the security management features available in Operating System ?

5) What is multi-processing?

6) What are the different Operating Systems used in computer?

SECTION-C

Short Answers

1) What are the advantages and disadvantages of Time-sharing features?

2) List out the key features of Operating system

3) Write a note on Multiprocessing

SECTION - D

Explain in detail

1) Explain the concept of a Distributed Operating System along with its advantages.

2) List out the points to be noted while creating a user interface for an Operating system.

3) Explain the process manangement algorithms in Operating System.

References

1) Silberschatz, galvin gagne, Operating System concepts – john wiley&sons,inc

2) Andrew s. Tanenbaum, modern Operating Systems – pearson publication

3) Andrew s. Tanenbaum , Operating Systems design and implementation, prentice hall publication

4) Tom anderson, Operating Systems: principles and practice, recursive books

5) Thomas w. Doeppner, Operating Systems in depth: design and programming, john wiley & sons, inc

CHAPTER 5

Working with Windows Operating System

Learning Objectives

After learning the concepts in this chapter, the students will be able

- To know the concepts of Operating System.
- To know the versions of the windows operating system.
- To know the concepts like desktop and the elements of window.
- To explore the document window.
- To compare the different types of icons.
- To explore the windows directory structure.
- To practice creating files and folders in specific drives.
- To manage the files and folders.
- To know the procedure to start and shutdown the computer.

CZQ9V4

5.1. Introduction to Operating System

An Operating System (OS) is a system software (Figure 5.1) that enables the hardware to communicate and operate with other software. It also acts as an interface between the user and the hardware and controls the overall execution of the computer.

Following are some of the important functions of an Operating System as discussed in the previous chapter:

- Memory Management
- Process Management
- Device Management
- File Management
- Security Management
- Control overall system performance

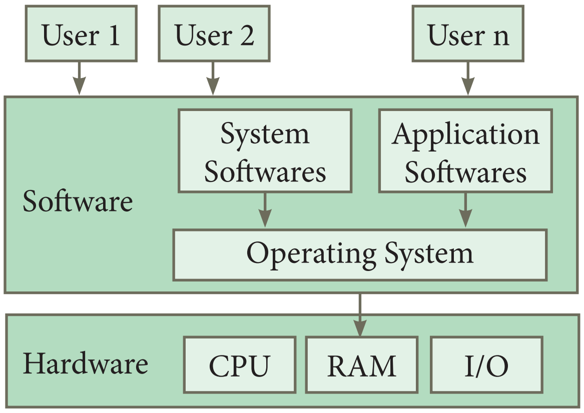


Figure 5.1. Overview of an Operating System

5.2. Introduction to Windows Operating System

Every computer needs an Operating System to function. Microsoft Windows is one of the most popular Graphical User Interface (GUI). Multiple applications can execute simultaneously in Windows, and this is known as **"Multitasking".**

Windows Operating System uses both Keyboard and mouse as input devices. Mouse is used to interact with Windows by clicking its icons. Keyboard is used to enter alphabets, numerals and special characters.

Some of the functions of Windows Operating System are:

- Access applications (programs) on the computer (word processing, games, spread sheets, calculators and so on).
- Load any new program on the computer.
- Manage hardware such as printers, scanners, mouse, digital cameras etc.,
- File management activities (For example creating, modifying, saving, deleting files and folders).
- Change computer settings such as colour scheme, screen savers of your monitor, etc.

With reference to the Table 5.1, let us see the versions of Windows Operating System.

5.4. Handling the mouse

Before learning Window Operating System, you should know more about mouse and its actions.

5.3. Various versions of Windows

Versions	Logo	Year	Specific features
Windows 1.x	MICROSOFT WINDOWS	1985	• Introduction of GUI in 16 - bit. processor • Mouse was introduced as an input device.
Windows 2.x	MICROSOFT WINDOWS	1987	• Supports to minimize or maximize windows. • Control panel feature was introduced with various system settings and customising options.
Windows 3.x	MICROSOFT WINDOWS	1992	• Introduced the concept of multitasking. • Supported 256 colours which brought a more modern, colourful look to the interface.
Windows 95	MICROSOFT WINDOWS	1995	• Introduced Start button, the taskbar, Windows Explorer and Start menu. • Introduced 32 - bit processor and focused more on multitasking.
Windows 98	MICROSOFT WINDOWS	1998	• Integration of the Web browser (Internet Explorer) with the Operating System. • DOS gaming began to disappear as Windows based games improved. • Plug and play feature was introduced.
Windows NT	MICROSOFT WINDOWS		• Designed to act as servers in network.
Windows Me	Microsoft Windows Me Millennium Edition	2000	• It introduced automated system diagnostics and recovery tools.

Windows 2000	Microsoft Windows 2000 Professional	2000	• Served as an Operating System for business desktop and laptop systems. • Four versions of Windows 2000 were released: Professional (for business desktop and laptop systems), Server (both a Web server and an office server), Advanced Server (for line-of-business applications) and Data Centre Server (for high-traffic computer networks).
Windows XP	Microsoft Windows xp	2001	• Introduced 64-bit Processor. • Improved Windows appearance with themes and offered a stable version.
Windows Vista	Microsoft Windows Vista	2006	• Updated the look and feel of Windows.
Windows 7	Windows 7	2009	• Booting time was improved, introduced new user interfaces like Aero Peek, pinning programs to taskbar, handwriting recognition etc. and Internet Explorer 8.
Windows 8	Windows 8	2012	• Windows 8 is faster than previous versions of Windows. • Start button was removed. • Windows 8 takes better advantage of multi-core processing, solid state drives (SSD), touch screens and other alternate input methods. • Served as common platform for mobile and computer.
Windows 10	Windows 10	2015	• Start Button was added again. • Multiple desktop. • Central Notification Center for App notification and quick actions. • Cortana voice activated personal assistant.

Table 5.1 Versions of Windows Operating System.

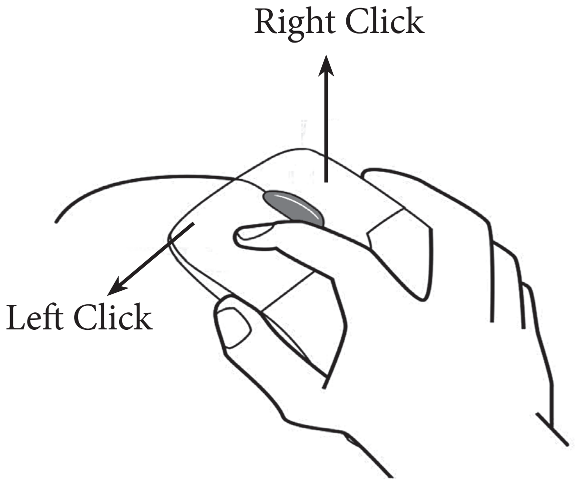


Figure 5.2.Mouse actions

The following are the mouse actions:

Action	Reaction
Point to an item	Move the mouse pointer over the item.
Click	Point to the item on the screen, press and release the left mouse button.
Right click	Point to the item on the screen, press and release the right mouse button. Clicking the right mouse button displays a pop up menu with various options.
Double-click	Point to the item on the screen, quickly press twice the left mouse button.

Drag and drop	Point to an item then hold the left mouse button as you move the pointer press and you have reached the desired position, release the mouse button.

5.5. Windows Desktop

The opening screen of Windows is called "Desktop".

The desktop of your computer may look different from what is seen in Figure 5.3.

This is because Windows allows you to change the appearance of the desktop.

In Figure 5.3, the desktop shows the Start button, Taskbar, Notification Area and date and time.

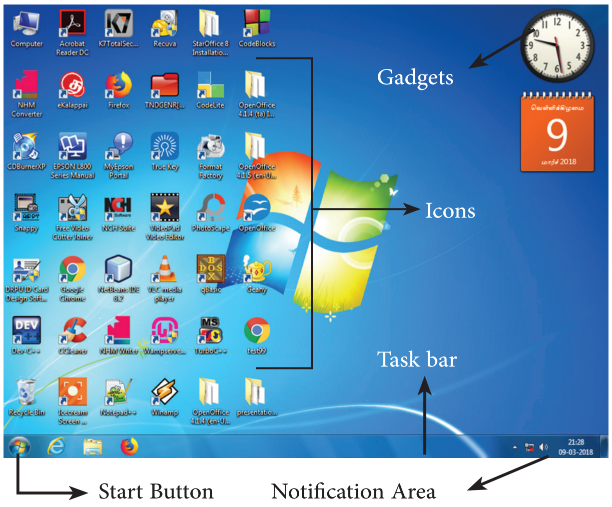


Figure 5.3. Microsoft Windows 7 Desktop

5.5.1. The Icons

Icon is a graphic symbol representing the window elements like files, folders, shortcuts etc., Icons play a vital role in GUI based applications.

5.5.1.1.Standard Icons

The icons which are available on desktop by default while installing Windows OS are called standard icons. The standard icons available in all Windows OS are My Computer, Documents and Recycle Bin.

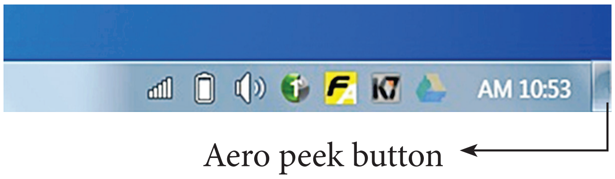


Figure 5.4. Aero peek button

5.5.1.2. Shortcut Icons:

Shortcut icons can be created for any application or file or folder. By double clicking the icon, the related application or file or folder will open.

(Figure5.5)

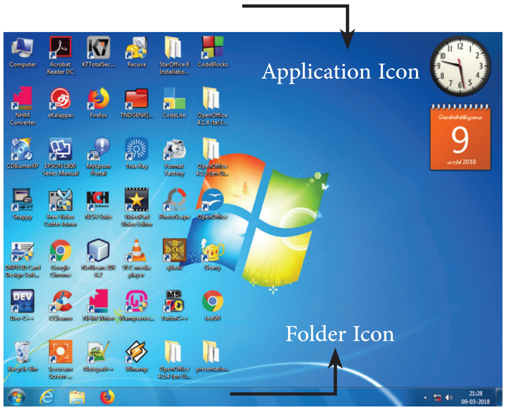
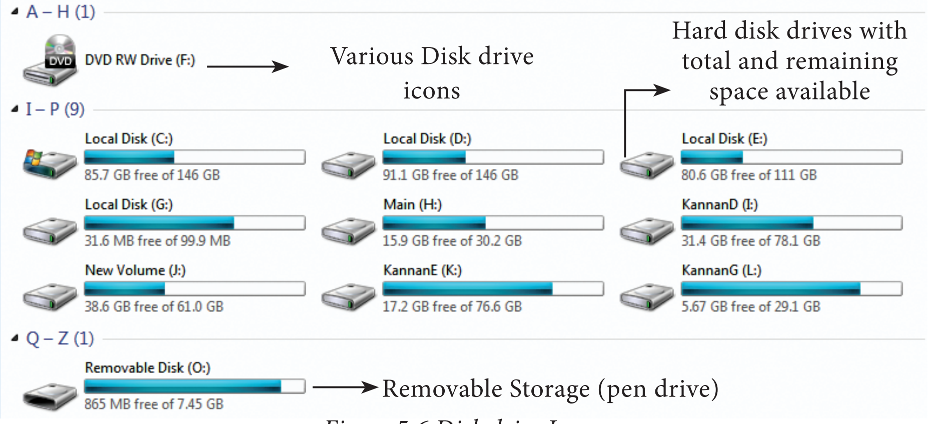


Figure 5.5.The types of Icons



A – H (1)

DVD RW Drive (F:)

I – P (9)

Local Disk (C:)
85.7 GB free of 146 GB

Local Disk (D:)
91.1 GB free of 146 GB

Local Disk (E:)
80.6 GB free of 111 GB

Local Disk (G:)
31.6 MB free of 99.9 MB

Main (H:)
15.9 GB free of 30.2 GB

KannanD (I:)
31.4 GB free of 78.1 GB

New Volume (J:)
38.6 GB free of 61.0 GB

KannanE (K:)
17.2 GB free of 76.6 GB

KannanG (L:)
5.67 GB free of 29.1 GB

Q – Z (1)

Removable Disk (O:)
865 MB free of 7.45 GB

Various Disk drive icons
Hard disk drives with total and remaining space available
Removable Storage (pen drive)

Figure 5.6.Disk drive Icons

5.5.1.3. Disk drive icons:

The disk drive icons graphically represent five disk drive options. (i) Hard disk (ii) CD-ROM/DVD Drive (iii) Pen drive (iv) Other removable storage such as mobile, smart phone, tablet etc., (v) Network drives if your system is connected with other system.

DO YOU KNOW?

You can move to the Desktop any time by pressing the Winkey + D or using Aero Peek while working in any application. You can see Figure 5.4 to know where Aero peek lies in the Taskbar.

5.6. The Window

Window is a typical rectangular area in an application or a document. It is an area on the screen that displays information for a specific program.

5.7. Application Window

It is an area on a computer screen with defined boundaries, and within which information is displayed. Such windows can be resized, maximised, minimised, placed side by side, overlap, and so on.

An Application Window contains an open application i.e. current application such as Word or Paint. When two or more windows are opened, only one of them is active and the rest are inactive. Figures 5.6 and 5.7 display the Application Window of OpenOffice Writer and the appearance of the Multiple Windows opened (overlapped) in the Desktop.

5.8. Document Window

A document window is a section of the screen used to display the contents of a document. Figure 5.8 is an example of a document window.

Note

When you open any application, such as OpenOffice Writer, OpenOffice Impress or OpenOffice Calc etc., you will find two Windows on the screen. The larger Window is called the Application Window. This Window helps the user to communicate with the Application program. The smaller window, which is inside the Application Window, is called the Document window. This Window is used for typing, editing, drawing, and formatting the text and graphics.

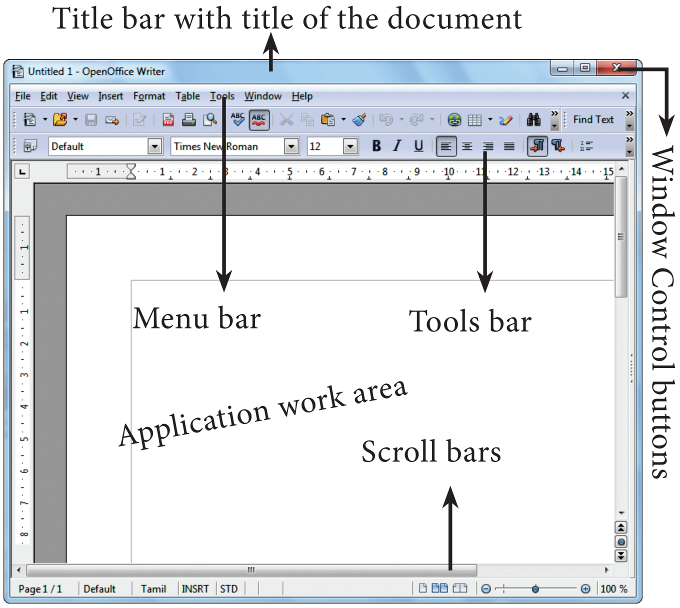
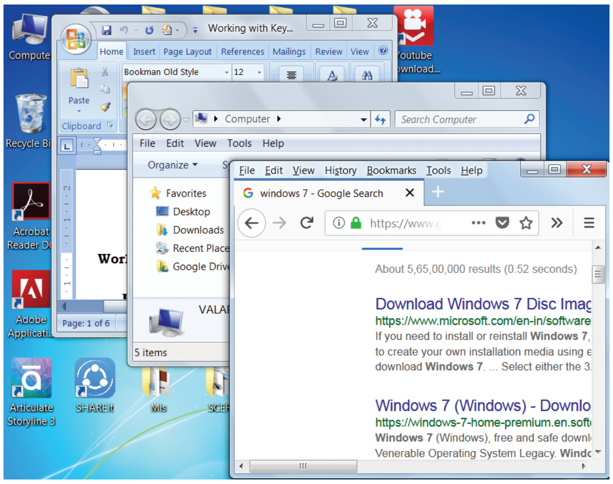

Title bar with title of the document
Untitled 1 - OpenOffice Writer
File Edit View Insert Format Table Tools Window Help
Default
Times New Roman
12
Find Text
Menu bar
Tools bar
Application work area
Scroll bars
Window Control buttons
Page 1 / 1 Default Tamil INSRT STD 100 %

Figure 5.6. Application Window



About 5,65,00,000 results (0.52 seconds)

Download Windows 7 Disc Imag
https://www.microsoft.com/en-in/software
If you need to install or reinstall Windows 7,
to create your own installation media using
download Windows 7. ... Select either the 3

Windows 7 (Windows) - Downlo
https://windows-7-home-premium.en.soft
Windows 7 (Windows), free and safe downl
Venerable Operating System Legacy. Windo

Figure 5.7. Multiple Windows opened in Desktop

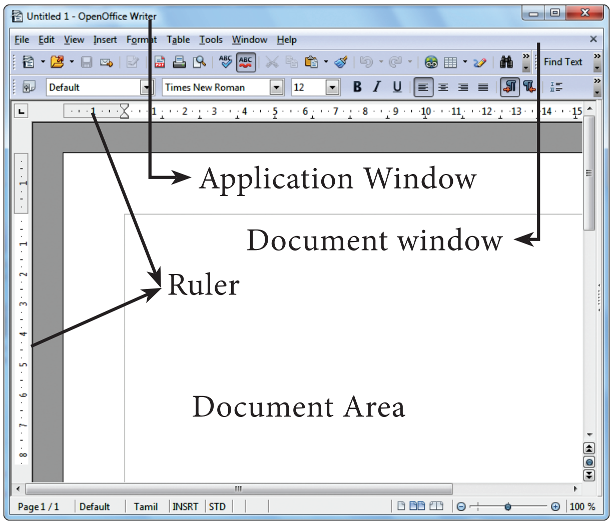

Untitled 1 - OpenOffice Writer
File Edit View Insert Format Table Tools Window Help
Default
Times New Roman
12
Find Text
Application Window
Document window
Ruler
Document Area
Page 1 / 1 Default Tamil INSRT STD 100 %

Figure 5.8.Document Window

5.9. Elements of a window

Figure 5.9 helps to understand the elements of a window.

5.9.1. Title Bar – The title bar will display the name of the application and the name of the document opened. It will also contain minimize, maximize and close button.

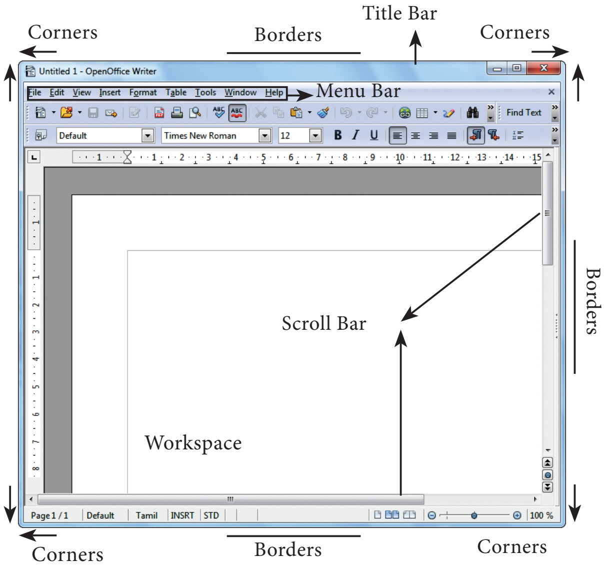


Figure 5.9 The elements of a window.

5.9.2 Menu Bar

The menu bar is seen under the title bar. Menus in the menu bar can be accessed by pressing Alt key and the letter that appears underlined in the menu title. Additionally, pressing Alt or F10 brings the focus on the first menu of the menu bar.

In Windows 7, in the absence of the menu bar, click **Organise** and from the drop down menu, click the **Layout** option and select the desired item from that list.

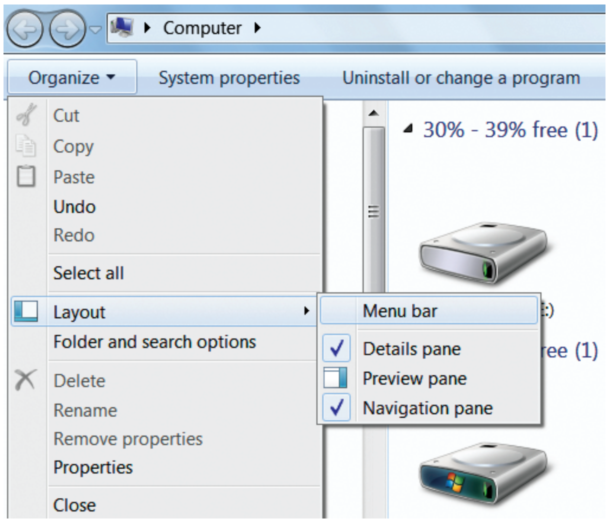


Figure 5.10. To display Menu Bar

Figure 5.10 helps to understand how to make menu bar visible in its absence.

5.9.3. The Workspace

The workspace is the area in the document window to enter or type the text of your document. Figure 5.10 Shows the workspace area in the document window.

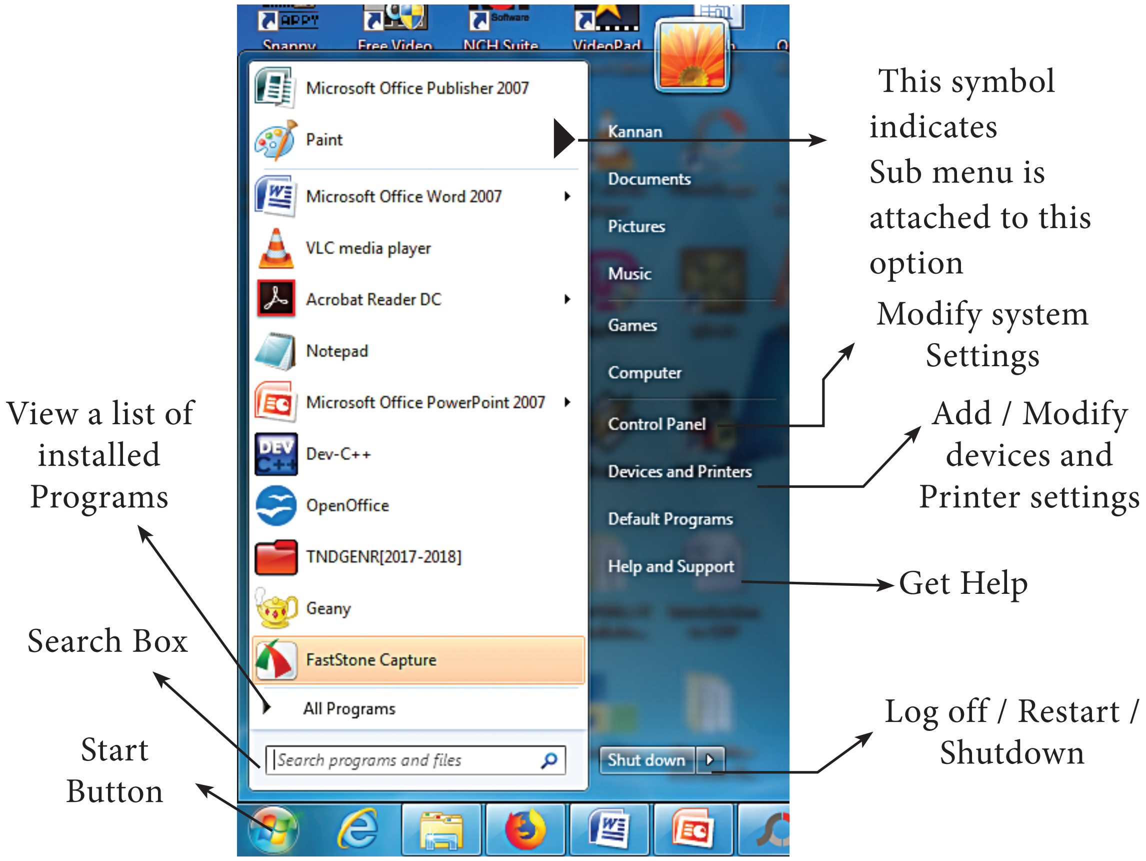
5.9.4. Scroll bars - The scroll bars are used to scroll the workspace horizontally or vertically. Figure 5.9 shows the Scroll bars.

5.9.5. Corners and borders

The corners and borders of the window helps to drag and resize the windows. The mouse pointer changes to a double headed arrow when positioned over a border or a corner. Drag the border or corner in the direction indicated by the double headed arrow to the desired size as shown in Figure 5.9. The window canbe resized by dragging the corners diagonally across the screen.

5.10.1. Start Menu

In the lower left-hand corner of the windows screen is the Start button. When you click on the button, the Start menu will appear. Using the start menu, you can start any application.



This symbol indicates Sub menu is attached to this option

Modify system Settings

Add / Modify devices and Printer settings

Get Help

Log off / Restart / Shutdown

View a list of installed Programs

Search Box

Start Button

Figure 5.11 - Start Menu

Taskbar

At the bottom of the screen is a horizontal bar called the taskbar. This bar contains (from left to right) the Start button, shortcuts to various programs, minimised programs and in the extreme right corner you can see the system tray which consist of volume control, network, date and time etc. Next to the Start button is the quick Launch Toolbar which contains task for frequently used applications.

5.10.2. Computer Icon

By clicking this icon, the user can see the disk drivers mounted in the system. In windows XP, Vista, this icon is called "My computer" in Windows 8 and 10, it is called "This PC". The functionality of computer icon remains the same in all versions of windows as shown in Figure 5.13.

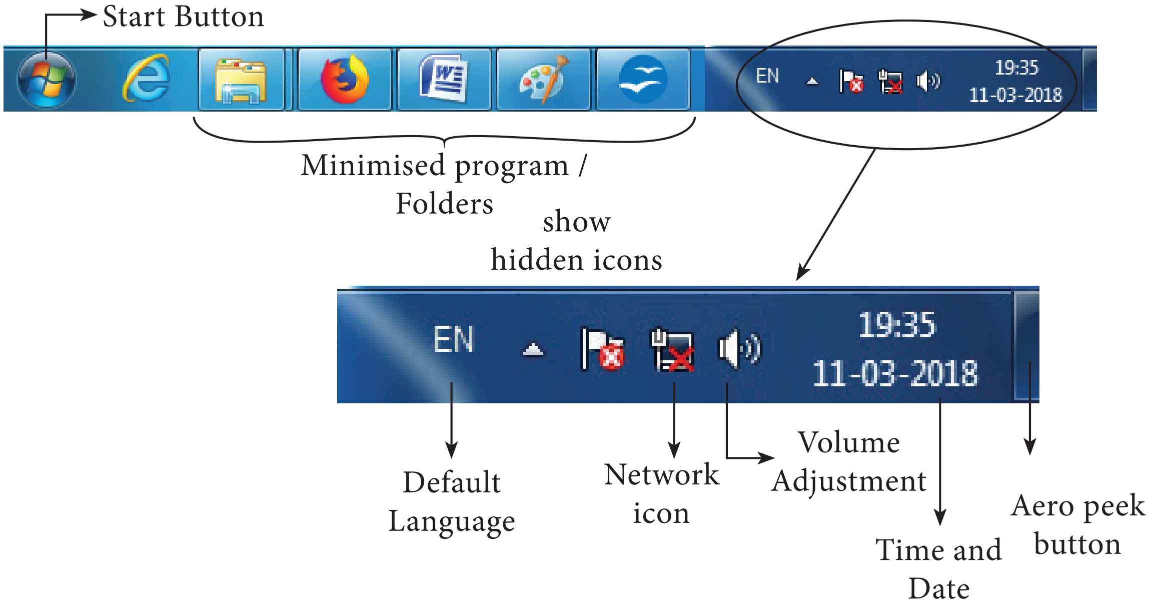


Figure 5.12.Taskbar

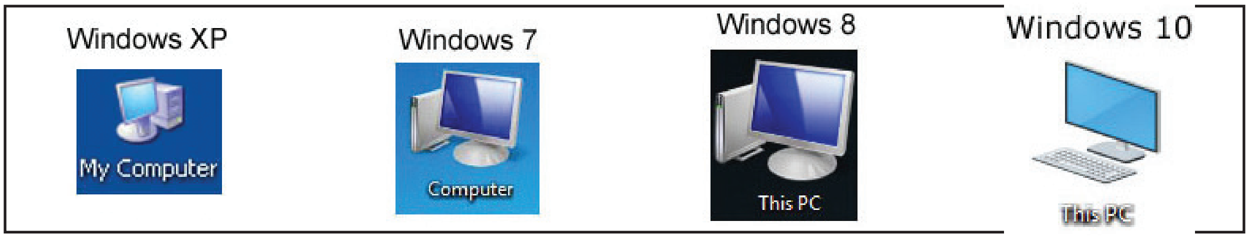


Figure 5.13. Computer icon in versions of Windows OS

5.10.3. Starting and Closing Applications

Most of the applications installed on your computer are available through the start menu. Depending on the system setup, the applications in the Start menu varies. To start an application:

1. Click the Start button and then point to All Programs. The Program menu appears.(Figure 5.14)
2. Point to the group that contains the application you want to start, and then click the application name.

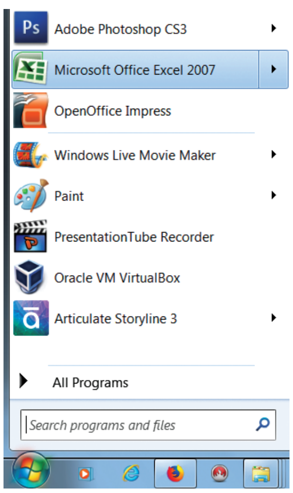
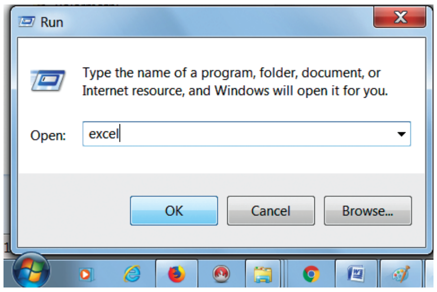


Figure 5. 14.Starting a applicatioin using Start menu

3. You can also open an application by clicking Run on the Start menu, and the name of the application. (Figure 5.15)



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: excel

Figure 5.15.Starting a program using Run option

4. To quit an application, click the Close button in the upper right corner of the application window. (Figure 5.16)

Workshop

1. ◆ Start the application Wordpad using Start menu and Run option.
   ◆ Close the Wordpad application using File menu.

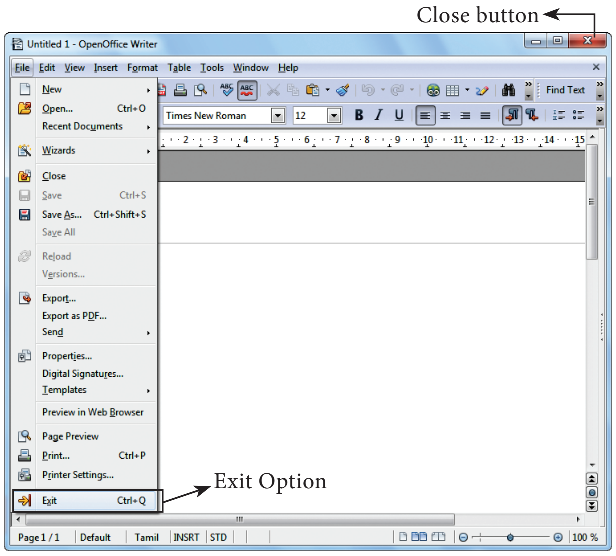


Figure 5.16. Closing the application using Close button and Exit option

5. You can also quit an application by clicking on File → Exit and File → Close option in Windows 7. (Figure 5.16)

5.11. Managing Files and Folders

In Windows 7, you can Organise your documents and programs in the form of files and folders. You can move, copy, rename, delete and search the files and folders.

5.11.1. Creating files and Folders

5.11.1.1 Creating Folders

You can store your files in many locations – on the hard disk or in other devices. To better organise your files, you can store them in folders.

There are two ways in which you can create a new folder:

Method I:

Step 1: Open **Computer Icon**.

Step 2: Open any drive where you want to create a new folder. (For example select D:)

Step 3: Click on File → New → Folder.

Step 4: A new folder is created with the default name "New folder". (Figure 5.19)

Step 5: Type in the folder name and press Enter key. (Figure 5.20 shows the newly created Folder named "Test Folder ").

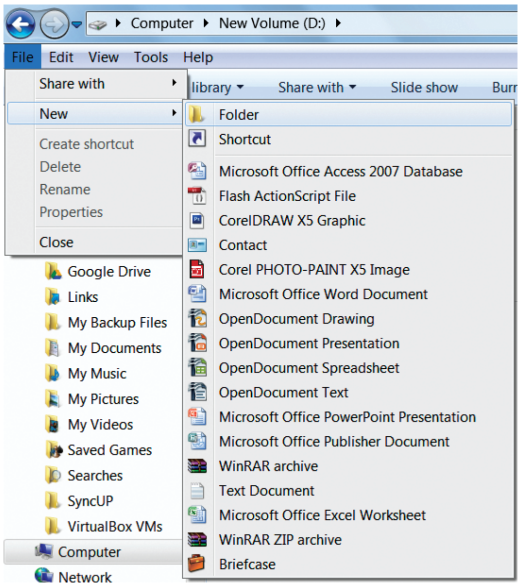


Figure 5.17. Creating a Folder using File menu

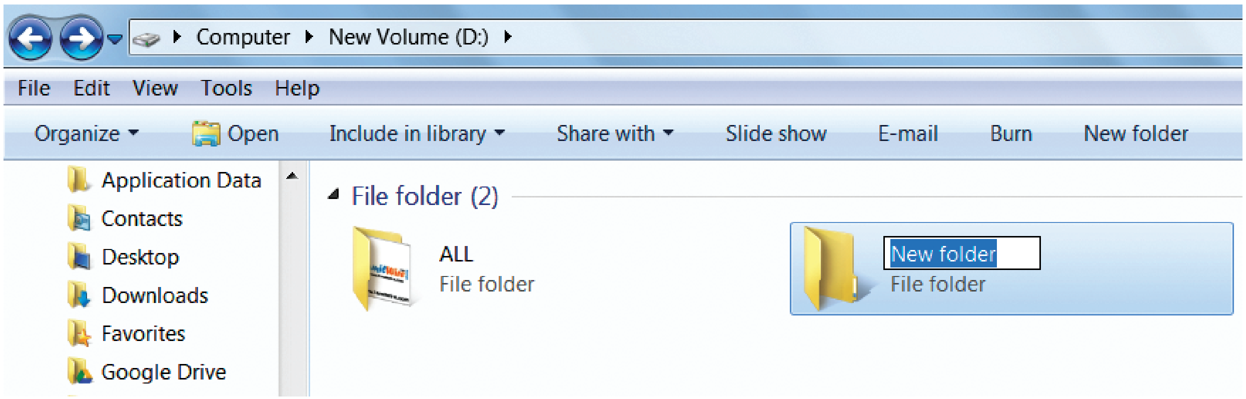


Figure 5.18. New Folder created with the default name

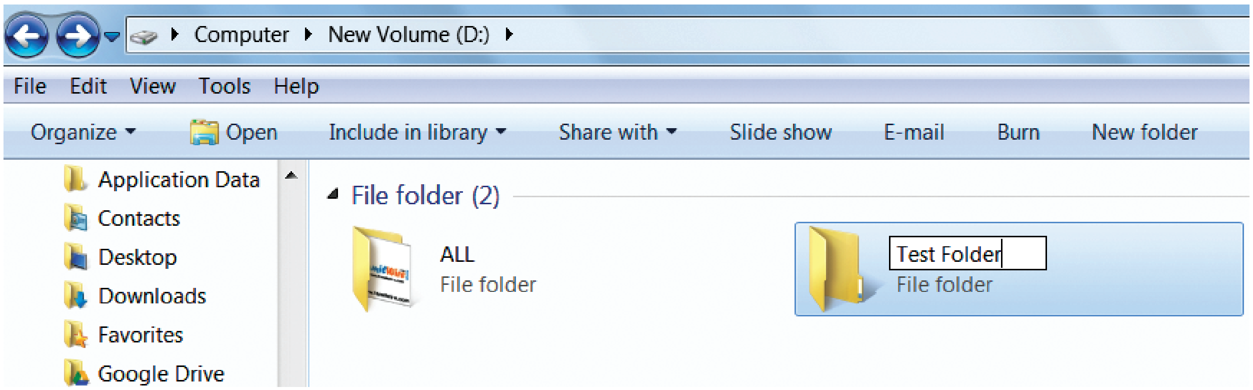


Figure 5.19. Renaming the new Folder

Method II:

In order to create a folder in the desktop:

Step 1: In the Desktop, right click → New → Folder. (Figure 5.20 Shown the procedure)

Step 2: A Folder appears with the default name "New folder" and it will be highlighted as shown in the Figure 5.22.

Step 3: Type the name you want and press Enter Key.

Step 4: The name of the folder will change.

Workshop

2. Create a Folder in My Documents with your name using any one of the methods discussed.

5.11.1.2 Creating Files (Wordpad)

Wordpad is an in-built word processor application in Windows OS to create and manipulate text documents.

In order to create files in wordpad you need to follow the steps given below.

1. Click Start → All Programs → Accessories → Wordpad or Run → type Wordpad, click OK. Wordpad window will be opened as shown in Figure 5.22.
2. Type the contents in the workspace and save the file using File → Save or Ctrl + S.
3. Save As dialog box will be opened.
4. In the dialog box, select the location where you want to save the file by using **look in** drop down list box.
5. Type the name of the file in the **file name** text box.
6. Click save button.

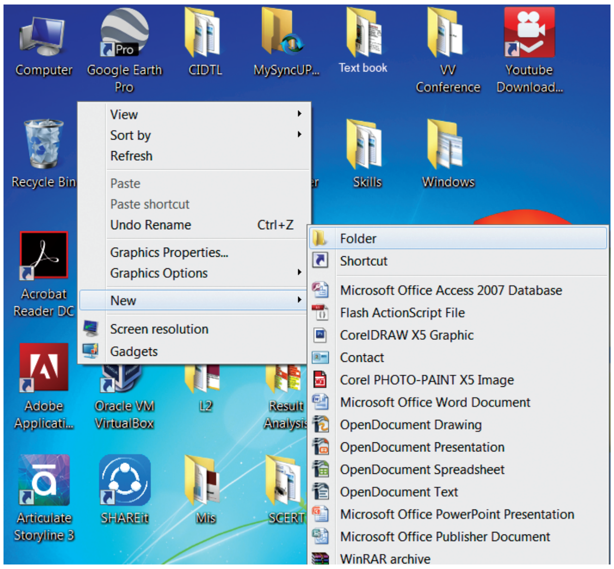


Figure 5.20. Creating a folder in the desktop

Workshop

3. Open the Wordpad application and save it under a folder created with your name in My Documents.

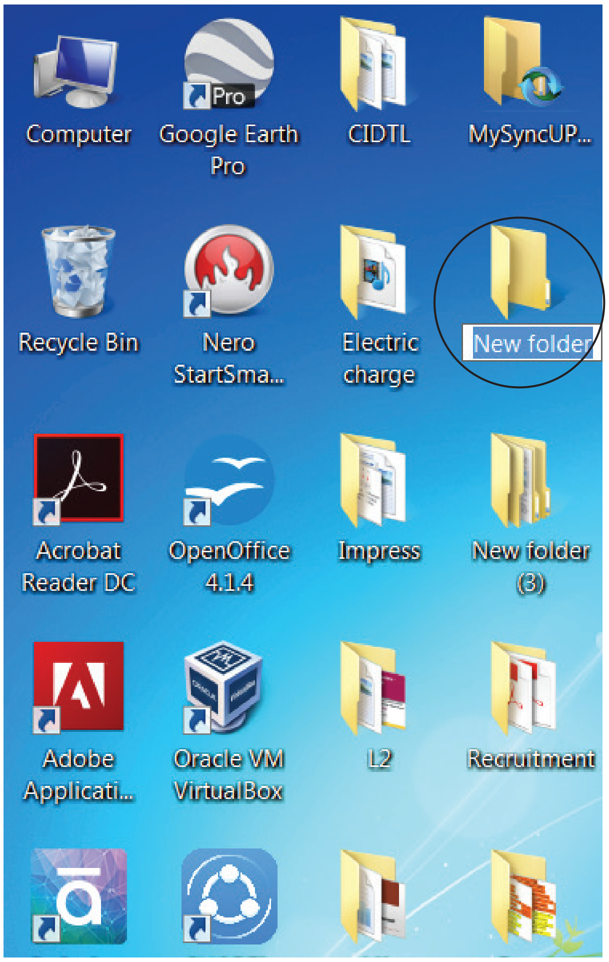


Figure 5.21 New folder icon on the dektop

5.11.2. Finding Files and Folders

You can use the **search** box on the **Start** menu to quickly search a particular folder or file in the computer or in a specific drive.

To find a file or folder:

1. Click the **Start** button, the **search** box appears at the bottom of the start menu.
2. Type the name of the file or the folder you want to search. Even if you give the part of the file or folder name, it will display the list of files or folders starting with the specified name. (Figure 5.23)
3. The files or the folders with the specified names will appear, if you click that file, it will directly open that file or the folder.

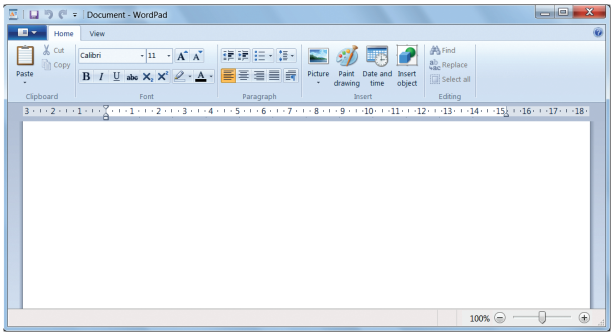


Figure 5.22. Wordpad - Word Processor application

4. There is another option called "**See more results**" which appears above the **search** box.
5. If you click it, it will lead you to a **Search Results** dialog box where you can click and open that file or the folder.

Searching Files or folders using Computer icon

1. Click **Computer Icon** from desktop or from **Start menu**.
2. The Computer disk drive screen will appear and at the top right corner of that screen, there is a **search** box option. (Figure 5.24)
3. Type the name of the file or the folder you want to search. Even if you give the part of the file or folder name, it will display the list of files or folders starting with the specified name.

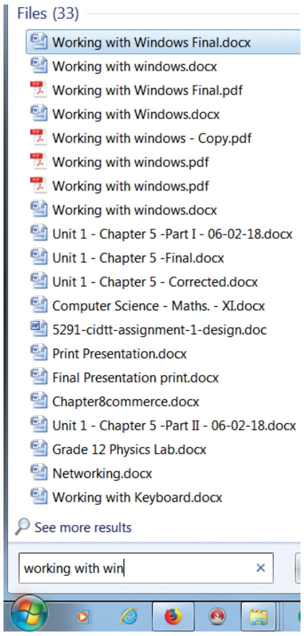

Files (33)
Working with Windows Final.docx
Working with windows.docx
Working with Windows Final.pdf
Working with Windows.docx
Working with windows - Copy.pdf
Working with windows.pdf
Working with windows.pdf
Working with windows.docx
Unit 1 - Chapter 5 -Part I - 06-02-18.docx
Unit 1 - Chapter 5 -Final.docx
Unit 1 - Chapter 5 - Corrected.docx
Computer Science - Maths. - XI.docx
5291-cidtt-assignment-1-design.doc
Print Presentation.docx
Final Presentation print.docx
Chapter8commerce.docx
Unit 1 - Chapter 5 -Part II - 06-02-18.docx
Grade 12 Physics Lab.docx
Networking.docx
Working with Keyboard.docx
See more results
working with win

Figure 5.23. Finding a File/Folder using Start button

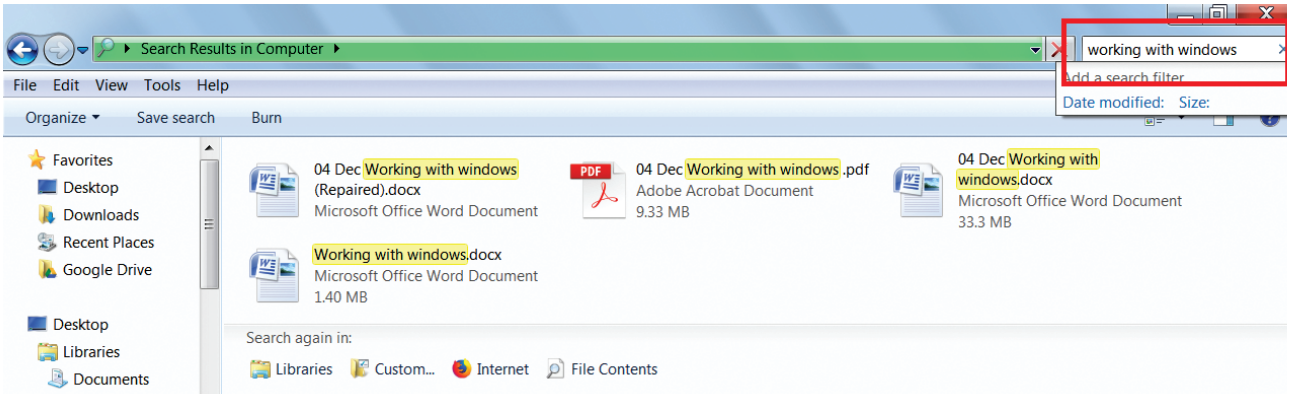

Search Results in Computer
working with windows
Add a search filter
Date modified: Size:
File Edit View Tools Help
Organize Save search Burn
Favorites
Desktop
Downloads
Recent Places
Google Drive
Desktop
Libraries
Documents
04 Dec Working with windows (Repaired).docx
Microsoft Office Word Document
04 Dec Working with windows.pdf
Adobe Acrobat Document
9.33 MB
04 Dec Working with windows.docx
Microsoft Office Word Document
33.3 MB
Working with windows.docx
Microsoft Office Word Document
1.40 MB
Search again in:
Libraries
Custom...
Internet
File Contents

Figure 5.24. Finding a File/Folder in the Computer icon screen

4. Just click and open that file or the folder.

Workshop

> 4. Find the file created in Workshop-3 using the above procedure

5.11.3. Opening existing Files or Folders

The most common way of opening a file or a Folder is to double click on it.

5.11.4. Renaming Files or Folders

There are number of ways to rename files or folders. You can rename using the File menu, left mouse button or right mouse button.

Method 1

Using the FILE Menu

1. Select the File or Folder you wish to Rename.
2. Click File→ Rename.
3. Type in the new name.
4. To finalise the renaming operation, press Enter as in Figure 5.25.

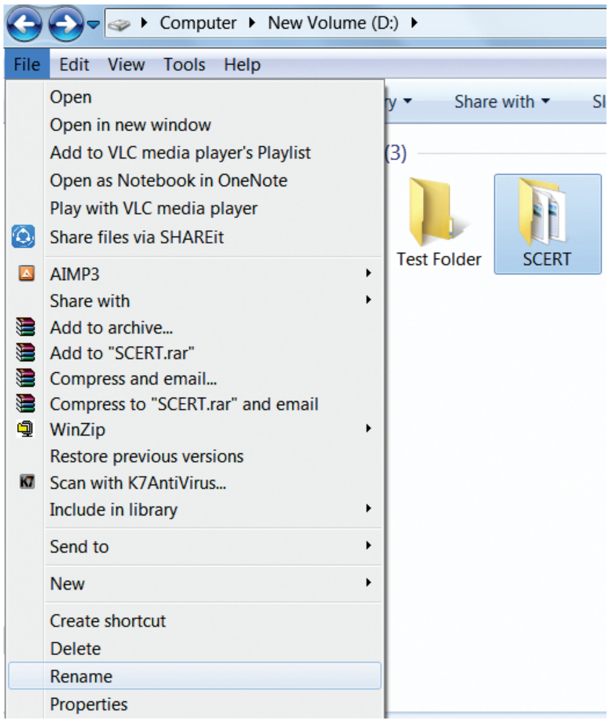


Figure 5.25. Renaming File/Folders using the File menu

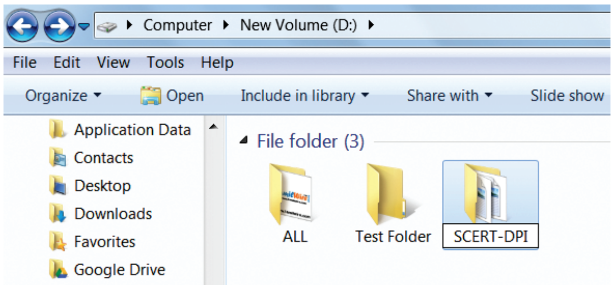


Figure 5.26.Folder renamed

Figure 5.26, you can see that the folder is renamed as SCERT-DPI from SCERT.

Method 2

Using the Right Mouse Button

1. Select the file or folder you wish to rename.
2. Click the right mouse button over the file or folder. (Figure 5.27)
3. Select Rename from the pop-up menu.

4. Type in the new name.
5. To finalise the renaming operation, press Enter.
6. Figure 5.29. Shows that the folder "New Folder" is renamed as C++.

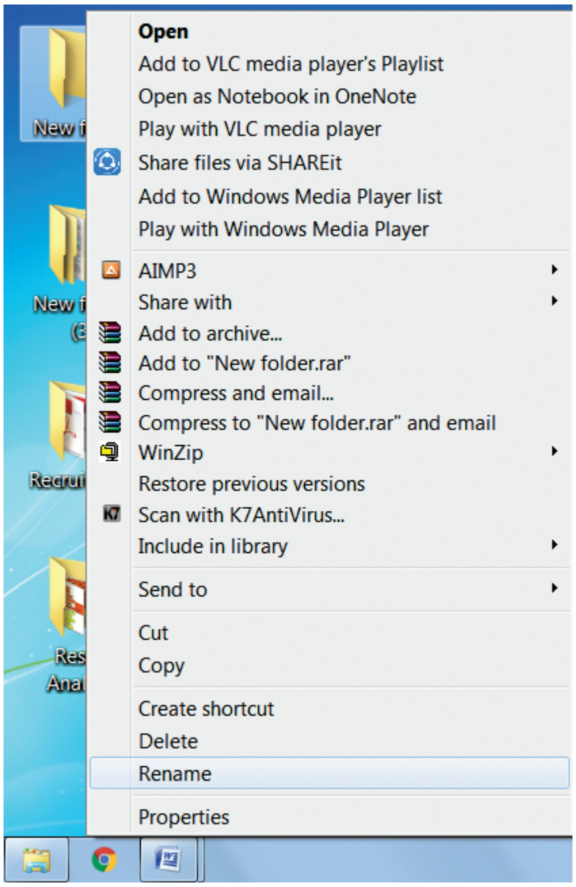


Figure 5.27. Renaming File/Folders using the Right Mouse Button

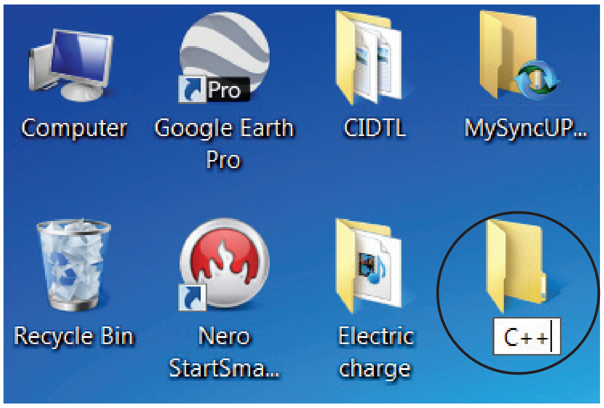


Figure 5.28. New Folder is renamed as C++

Method 3

Using the Left Mouse Button

1. Select the file or folder you wish to rename.
2. Press F2 or click over the file or folder. A surrounding rectangle will appear around the name.
3. Type in the new name.
4. To finalise the renaming operation, press Enter.

Workshop

5. Rename the file created by you using the File menu, left mouse button or right mouse button.

5.11.5. Moving/Copying Files and Folders

You can move your files or folders to other areas using variety of methods.

Moving Files and Folders

Method I-CUT and PASTE

To move a file or folder, first select the file or folder and then choose one of the following:

- Click on the **Edit** → **Cut** or **Ctrl** + **X** Or right **click** → **cut** from the pop-up menu.
- To move the file(s) or folder(s) in the new location, navigate to the new location and paste it using Click **Edit** → **Paste** from edit menu or **Ctrl** + **V** using keyboard.
- Or Right **click** → **Paste** from the pop-up menu. The file will be pasted in the new location.

Method II – Drag and Drop

In the disk drive window, we have two panes called left and right panes. In the left pane, the files or folders are displayed like a tree structure. In the right pane, the files inside the specific folders in the left pane are displayed with various options.

- In the right pane of the Disk drive window, select the file or folder you want to move.
- Click and drag the selected file or folder from the right pane, to the folder list on the left pane.
- Release the mouse button when the target folder is highlighted (active).
- Your file or folder will now appear in the new area. Figrue 5.29 shows how to move files or folders using drag and drop method.

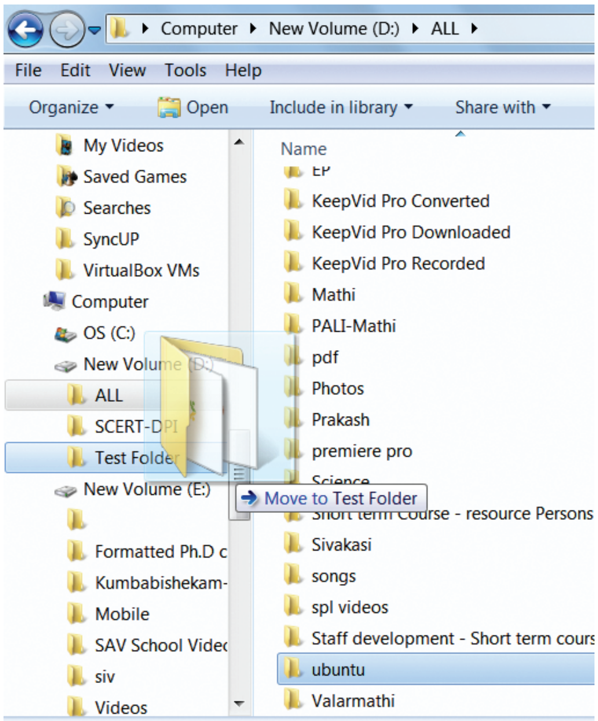


Figure 5.29.Moving the File/ Folder using drag and drop

Copying Files and Folders

There are variety of ways to copy files and folders:

Method I - COPY and PASTE

To copy a file or folder, first select the file or folder and then choose one of the following:

- Click **Edit** → **Copy** or **Ctrl + C** or **right click**→ **Copy** from the pop-up menu.
- To paste the file(s) or folder(s) in the new location, navigate to the target location then do one of the following:
- Click **Edit** → **Paste** or **Ctrl + V.**
- Or **Right click** → **Paste** from the pop-up menu.

Method II – Drag and Drop

- In the RIGHT pane, select the file or folder you want to copy.
- Click and drag the selected file and/or folder to the folder list on the left, and drop it where you want to copy the file and/or folder.
- Your file(s) and folder(s) will now appear in the new area.

Note

If you want to select multiple files or folders, use **Ctrl + Click.**

5.11.6. Copying Files and Folders to removable disk

There are several methods of transferring files to or from a removable disk.

- Copy and Paste
- Send To

METHOD I - Copy and Paste

- Plug the USB flash drive directly into an available USB port.
- If the USB flash drive or external drive folder does NOT open automatically, follow these steps:
- Click Start→Computer. (Figure 5.31)

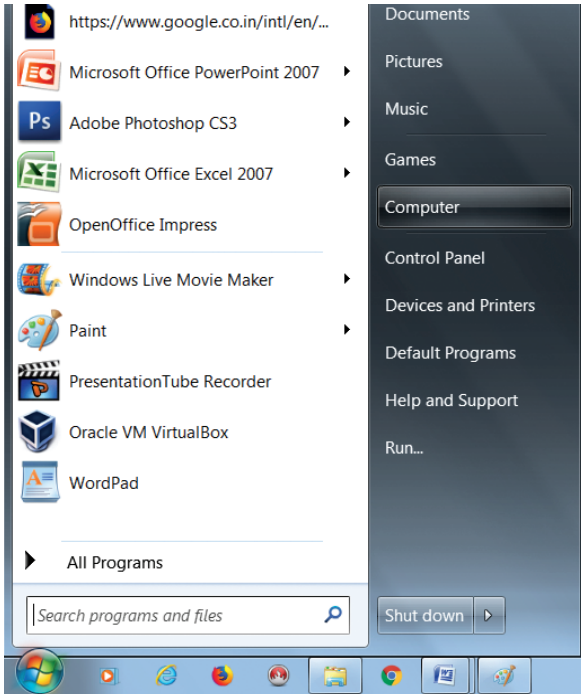


Figure 5.30. Selecting Computer option from Start menu

- Double-click on the Removable Disk associated with the USB flash drive. (Figure 5.31)

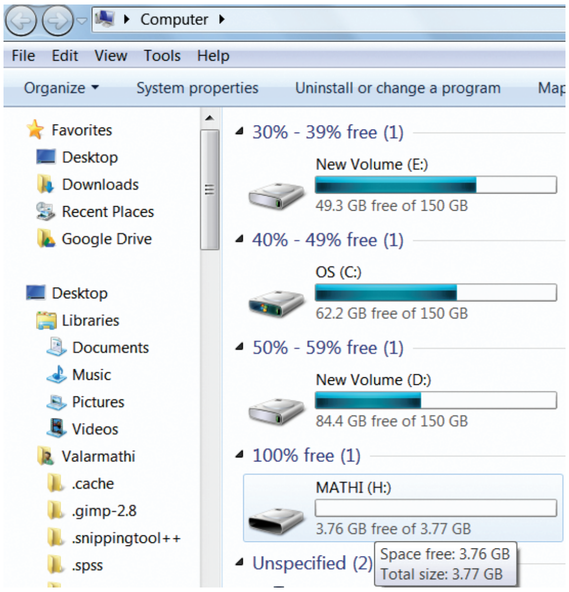


Figure 5.31. Double Clicking Removable Disk

- Navigate to the folders in your computer containing files you want to transfer.

Right-click on the file you want to copy, then select **Copy.** (Figure 5.32)

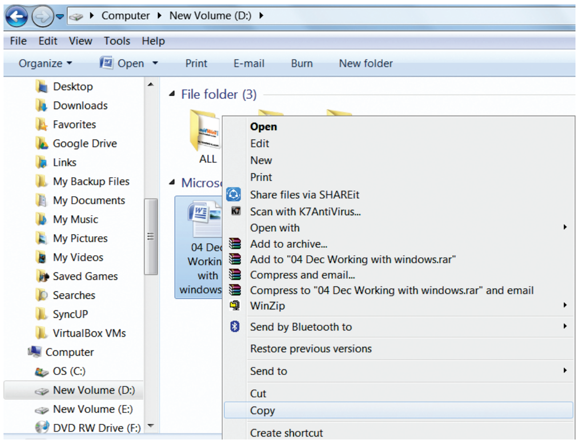


Figure 5.32. Copying File using right click

- Return to the Removable Disk window, right-click within the window, then select **Paste**. (Figure 5.33)

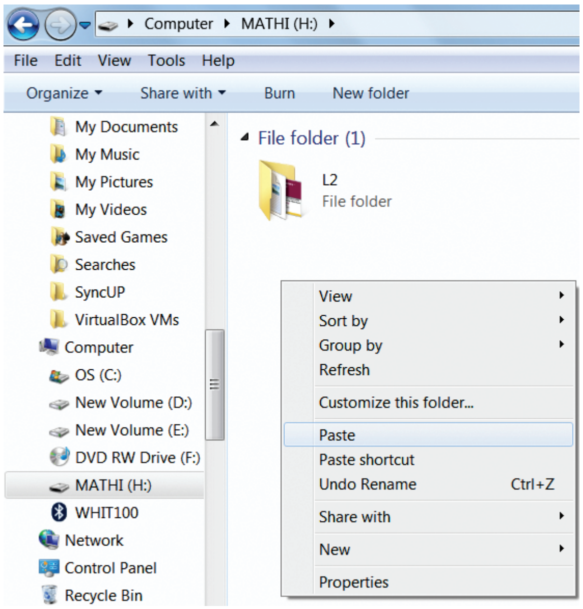


Figure 5.33. Pasting File using right click

METHOD II - Send To

- Plug the USB flash drive directly into an available USB port.
- Navigate to the folders in your computer containing files you want to transfer.
- Right-click on the file you want to transfer to your removable disk.
- Click **Send To** and select the Removable Disk associated with the USB flash drive. (Figure 5.34)

Workshop

6. ◆ Move the file created by you in My Documents to Drive D:.

   ◆ Copy the file created by you from drive D: to a removable disk.

5.11.7. Deleting Files and Folders

- When you delete a file or folder, it will move into the Recycle Bin.

To delete a file or folder:

Select the file or folder you wish to delete.

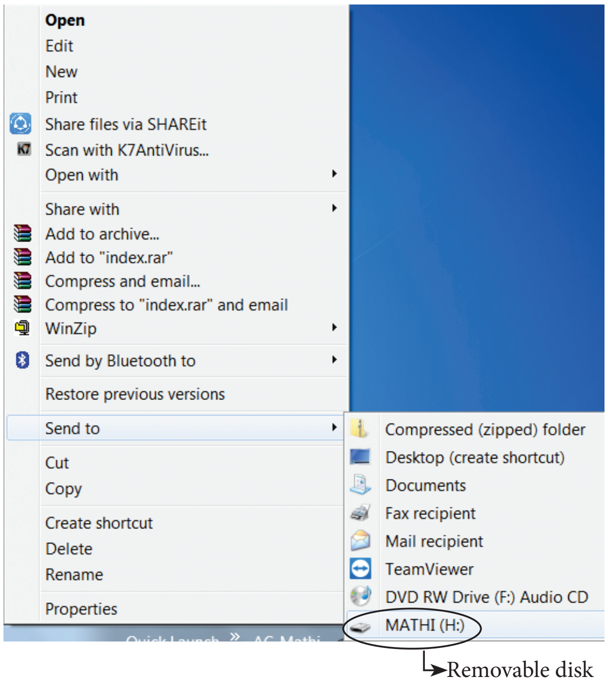


Figure 5.34. Copying File using Send to option

1. Right- click the file or folder, select **Delete** option from the po-pup menu or Click **File** → **Delete** or press **Delete** key from the keyboard.
2. The file will be deleted and moved to the Recycle bin.

Workshop

7. Delete the file created by you after duplicating the same under My Documents.

Note

To permanently delete a file or folder (i.e. to avoid sending a file or folder to the Recycle Bin), hold down the SHIFT key, and press **delete** on the keyboard.

Recycle Bin

Recycle bin is a special folder to keep the files or folders deleted by the user, which means you still have an opportunity to recover them. The user cannot access the files or folders available in the Recycle bin without restoring it. To restore file or folder from the Recycle Bin

- *Open Recycle bin.*
- Right click on a file or folder to be restored and select **Restore** option from the pop-up menu.
- To restore multiple files or folders, select Restore all items.
- To delete all files in the Recycle bin, select **Empty the Recycle Bin.**

5.12. Creating Shortcuts on the Desktop

Shortcuts to your most often used folders and files may be created and placed on the Desktop to help automate your work.

- Select the file or folder that you wish to have as a shortcut on the Desktop.
- Right click on the file or folder.
- Select **Send to** from the shortcut menu, then select Desktop (create shortcut) from the sub-menu.
- A shortcut for the file or folder will now appear on your desktop and you can open it from the desktop in the same way as any other icon. Figure 5.36.

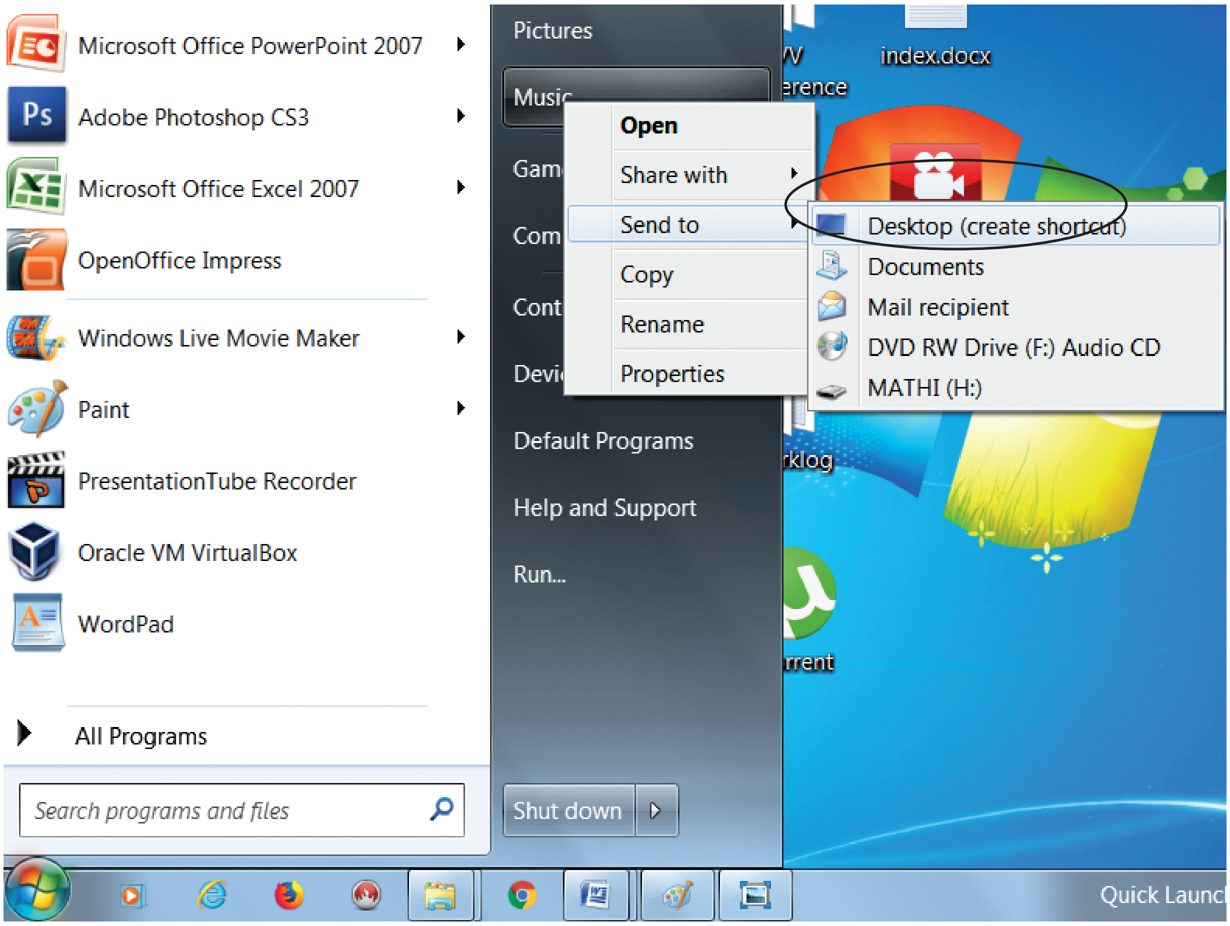


Figure 5.35 Creating Shortcut

5.13. Shutting down or Logging off a Computer

Once you have closed all open applications, you can either log off your computer or shut down the computer.

Log Off

To Log off/Shut down the computer:

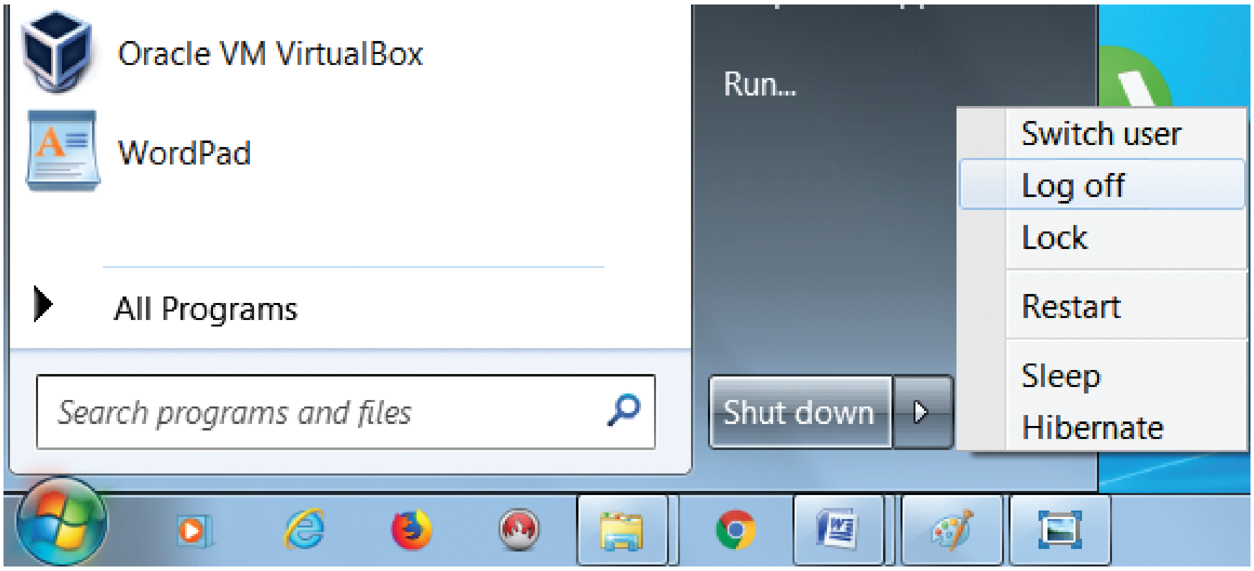


Figure 5.36. Log off option

- Click **start** → **log** off (click the arrow next to Shut down) or **Start** → **Shutdown** . (Figure 5.37.)
- If you have any open programs, then you will be asked to close them or windows will Force shut down, you will lose any un-saved information if you do this.
- **Switch User:** Switch to another user account on the computer without closing your open programs and Windows processes.
- **Log Off:** Switch to another user account on the computer after closing all your open programs and Windows processes.
- **Lock:** Lock the computer while you're away from it.
- **Restart:** Reboot the computer. (This option is often required as part of installing new software or Windows update.)
- **Sleep:** Puts the computer into a low-power mode that retains all running programs and open Windows in computer memory for a super-quick restart.
- **Hibernate** (found only on laptop computers): Puts the computer into a low-power mode after saving all running programs and open Windows on the machine's hard drive for a quick restart.

Activity

Student Activity

1. Create files and folders using Windows and Ubuntu and compare them.
2. Create a File/Folder in Windows 7, Windows 8 and Windows 10. Prepare a report on the differences you face while creating the same.

Evaluation

SECTION – A

Choose the correct answer

D191YE

1. From the options given below, choose the operations managed by the operating system.
   a. Memory
   b. Processes
   c. Disks and I/O devices
   d. all of the above
2. Which is the default folder for many Windows Applications to save your file?
   a. My Document
   b. My Pictures
   c. Documents and Settings
   d. My Computer
3. Under which of the following OS, the option Shift + Delete – permanently deletes a file or folder?
   a. Windows 7 b. MS-DOS
   c. Linux d. Android OS
4. What is the meaning of "Hibernate" in Windows XP/Windows 7?
   a. Restart the Computer in safe mode
   b. Restart the Computer in hibernate mode
   c. Shutdown the Computer terminating all the running applications
   d. Shutdown the Computer without closing the running applications
5. The shortcut key used to rename a file in windows
   a. F2 b.F4
   c.F5 d. F6

SECTION-B

Very Short Answers

1. what is known as Multitasking?
2. What are called standard icons
3. Differentiate Files and Folders.
4. Differentiate Save and save As option.
5. How will you Rename a File?

SECTION-C

Short Answers

1. What are the functions of Windows Operating system.
2. Write a note on Recycle bin.
3. Write a note on the elements of a window.
4. Write the two ways to create a new folder.
5. Differentiate copy and move

SECTION - D

Explain in detail

1. Explain the versions of Windows Operating System.
2. Explain the different ways of finding a file or Folder
3. Write the procedure to create shortcut in Windows OS.

D1HX12

GLOSSARY

Operating System (OS)	System software that enables the harware to communicate and operate with other software.
Mouse	Handheld hardware input device that control a cursor in a GUI and can move and slect text, icons, files, and folders.
Windows	Familer operating system developed by Microsoft corpn.
Desktop	Opening screen of windows operating system.
Icon	Tiny image represent a command.
Folder	Container of files
Linux	An operating system.

Unit II | Algorithmic Problem Solving

CHAPTER 6

Specification and Abstraction

Learning Objectives

After learning the concepts in this chapter, the students will be able

- To understand the concept of algorithmic problem solving.
- To apply the knowledge of algorithmic technique in problem solving.

A number of processes performed in our daily life follow the step-by-step execution of a sequence of instructions. Getting ready to school in the morning, drawing "kolams", cooking a dish, adding two numbers are examples of processes. Pro-cesses are generated by executing algorithms. In this chapter, we will see how algorithms are specified and how elements of a process are abstracted in algorithms.

6.1 Algorithms

An algorithm is a sequence of instructions to accomplish a task or solve a problem. An instruction describes an action. When the instructions are executed, a process evolves, which accomplishes the intended task or solves the given problem. We can compare an algorithm to a recipe, and the resulting process to cooking.

We are interested in executing our algorithms in a computer. A computer can only execute instructions in a programming language. Instructions of a computer are also known as statements.Therefore, ultimately, algorithms must be expressed using statements of a programming language.

A problem is specified by given input data, desired output data and a relation between the input and the output data. An algorithm starts execution with the input data, executes the statements, and finishes execution with the output data. When it finishes execution, the specified relation between the input data and the output data should be satisfied. Only then has the algorithm solved the given problem.

DO YOU KNOW?

G Polya was a Hungarian mathematician. He made fundamental contributions to combinatorics, number theory, numerical analysis and probability theory. He is also noted for his work in heuristics and mathematics education, identifying systematic methods of problem solving to further discovery and invention in mathematics for students and teachers. In "How to Solve It", he suggests the following steps when solving a mathematical problem: 1. Understand the problem. 2. Devise a plan. 3. Carry out the plan. 4. Review your work.

An algorithm is a step by step sequence of statements intended to solve a problem. When executed with input data, it generates a computational process, and ends with output data, satisfying the specified relation between the input data and the output data.

Example 6.1. Add two numbers: To add two numbers, we proceed by adding the right most digits of the two numbers, then the next right most digits together with carry that resulted from the previous (right) position, and so on. The computational process for adding 2586 and 9237 is illustrated in Table 6.1.

Step	5	4	3	2	1
Carry	1	0	1	1	-
Number	1	2	5	8	6
Number	2	9	2	3	7
Sum	1	1	8	2	3

Table 6.1: The process for adding two numbers

6.2 Algorithmic Problems

There are some principles and techniques for constructing algorithms. We usually say that a problem is algorithmic in nature when its solution involves the construction of an algorithm. Some types of problems can be immediately recognized as algorithmic.

Example 6.2. Consider the well-known Goat, grass and wolf problem: A farmer wishes to take a goat, a grass bundle and a wolf across a river. However, his boat can take only one of them at a time. So several trips are necessary to across the river. Moreover, the goat should not be left alone with the grass (otherwise, the goat would eat the grass), and the wolf should not be left alone with the goat (otherwise, the wolf would eat the goat). How can the farmer achieve the task? Initially, we assume that all the four are at the same side of the river, and finally, all the four must be in the opposite side. The farmer must be in the boat when crossing the river. A solution consists of a sequence of instructions indicating who or what should cross. Therefore, this is an algorithmic problem. Instructions may be like

Let the farmer cross with the wolf.

or

Let the farmer cross alone.

However, some algorithmic problems do not require us to construct algorithms. Instead, an algorithm is provided and we are required to prove some of its properties.

Example 6.3. Consider The Chameleons of Chromeland problem: On the island of Chromeland there are three different types of chameleons: red chameleons, green chameleons, and blue chameleons. Whenever two chameleons of different colors meet, they both change color to the third color. For which number of red, green and blue chameleons it is possible to arrange a series of meetings that results in all the chameleons displaying the same color? This is an algorithmic problem, because there is an algorithm to arrange meetings between chameleons. Using some properties of the algorithm, we can find out for which initial number of chameleons, the goal is possible.

6.3 Building Blocks of Algorithms

We construct algorithms using basic building blocks such as

- Data
- Variables
- Control flow
- Functions

6.3.1 Data

Algorithms take input data, process the data, and produce output data. Computers provide instructions to perform operations on data. For example, there are instructions for doing arithmetic operations on numbers, such as add, subtract, multiply and divide. There are different kinds of data such as numbers and text.

6.3.2 Variables

Variables are named boxes for storing data. When we do operations on data, we need to store the results in variables. The data stored in a variable is also known as the value of the variable. We can store a value in a variable or change the value of variable, using an assignment statement.

Computational processes in the real-world have state. As a process evolves, the state changes. How do we represent the state of a process and the change of state, in an algorithm? The state of a process can be represented by a set of variables in an algorithm. The state at any point of execution is simply the values of the variables at that point. As the values of the variables are changed, the state changes.

Example 6.4. State: A traffic signal may be in one of the three states: green, amber, or red. The state is changed to allow a smooth flow of traffic. The state may be represented by a single variable signal which can have one of the three values: green, amber, or red.

6.3.3 Control flow

An algorithm is a sequence of statements. However, after executing a statement, the next statement to be executed need not be the next statement in the algorithm. The statement to be executed next may depend on the state of the process. Thus, the order in which the statements are executed may differ from the order in which they are written in the algorithm. This order of execution of statements is known as the control flow.

There are three important control flow statements to alter the control flow depending on the state.

- In sequential control flow, a sequence of statements are executed one after another in the same order as they are written.
- In alternative control flow, a condition of the state is tested, and if the condition is true, one statement is executed; if the condition is false, an alternative statement is executed.
- In iterative control flow, a condition of the state is tested, and if the condition is true, a statement is executed. The two steps of testing the condition and executing the statement are repeated until the condition becomes false.

6.3.4 Functions

Algorithms can become very complex. The variables of an algorithm and dependencies among the variables may be too many. Then, it is difficult to build algorithms correctly. In such situations, we break an algorithm into parts, construct each part separately, and then integrate the parts to the complete algorithm.

The parts of an algorithm are known as functions. A function is like a sub algorithm. It takes an input, and produces an output, satisfying a desired input output relation.

Example 6.5. Suppose we want to calculate the surface area of a cylinder of radius r and height h.

$$A = 2\pi r^2 + 2\pi rh$$

We can identify two functions, one for calculating the area of a circle and the other for the circumference of the circle. If we abstract the two functions as circle_area(r) and circle_circumference(r), then cylinder_area(r, h) can be solved as

cylinder_area (r,h) = 2 X circle_area (r) +
circle_circumference (r) X h

6.4 Algorithm Design Techniques

There are a few basic principles and techniques for designing algorithms.

1. **Specification:** The first step in problem solving is to state the problem precisely. A problem is specified in terms of the input given and the output desired. The specification must also state the properties of the given input, and the relation between the input and the output.

2. **Abstraction:** A problem can involve a lot of details. Several of these details are unnecessary for solving the problem. Only a few details are essential. Ignoring or hiding unnecessary details and modeling an entity only by its essential properties is known as abstraction. For example, when we represent the state of a process, we select only the variables essential to the problem and ignore inessential details.

3. **Composition:** An algorithm is composed of assignment and control flow statements. A control flow statement tests a condition of the state and, depending on the value of the condition, decides the next statement to be executed.

4. **Decomposition:** We divide the main algorithm into functions. We construct each function independently of the main algorithm and other functions. Finally, we construct the main algorithm using the functions. When we use the functions, it is enough to know the specification of the function. It is not necessary to know how the function is implemented.

6.5 Specification

To solve a problem, first we must state the problem clearly and precisely. A problem is specified by the given input and the desired output. To design an algorithm for solving a problem, we should know the properties of the given input and the properties of the desired output. The goal of the algorithm is to establish the relation between the input and the desired output.

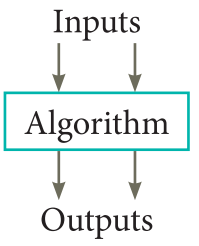


Figure 6.1: Input-output relation

An algorithm is specified by the properties of the given input and the relation between the input and the desired output. In simple words, specification of an algorithm is the desired input-output relation.

The inputs and outputs are passed between an algorithm and the user through variables. The values of the variables when the algorithm starts is known as the initial state, and the values of the variables when the algorithm finishes is known as the final state.

Let P be the required property of the inputs and Q the property of the desired outputs. Then the algorithm S is specified as

1. algorithm_name (inputs)
2. -- inputs : P
3. -- outputs: Q

This specification means that if the algorithm starts with inputs satisfying P, then it will finish with the outputs satisfying Q.

A double dash -- indicates that the rest of the line is a comment. Comments are statements which are used to annotate a program for the human readers and not executed by the computer. Comments at crucial points of flow are useful, and even necessary, to understand the algorithm. In our algorithmic notation, we use double dashes (—) to start a comment line. (In C++, a double slash // indicates that the rest of the line is a comment).

Example 6.6. Write the specification of an algorithm to compute the quotient and remainder after dividing an integer A by another integer B. For example,

divide (22, 5) = 4, 2

divide (15, 3) = 5 , 0

Let A and B be the input variables. We will store the quotient in a variable q and the remainder in a variable r. So q and r are the output variables.

What are the properties of the inputs A and B?

1. A should be an integer. Remainder is meaningful only for integer division, and
2. B should not be 0, since division by 0 is not allowed.

We will specify the properties of the inputs as

— inputs: A is an integer and B ≠ 0

What is the desired relation between the inputs A and B, and the outputs q and r?

1. The two outputs q (quotient) and r (remainder) should satisfy the property

   $A = q \text{ X } B + r$, and

2. The remainder r should be less than the divisor B,

   $0 \le r < B$

Combining these requirements, we will specify the desired input-output relation as

— outputs: $A = q \text{ X } B + r$ and $0 < r < B$.

The comment that starts with — inputs: actually is the property of the given inputs. The comment that starts with — outputs: is the desired relation between the inputs and the outputs. The specification of the algorithm is

1. **divide (A , B)**
2. **-- inputs: A is an integer and B ≠ 0**
3. **-- outputs : $A = q \text{ X } B + r$ and $0 \le r < B$**

Specification format: We can write the specification in a standard three part format:

- The name of the algorithm and the inputs.
- Input: the property of the inputs.
- Output: the desired input-output relation.

The first part is the name of the algorithm and the inputs. The second part is the property of the inputs. It is written as a comment which starts with — inputs: The third part is the desired input-output relation. It is written as a comment which starts with — outputs:. The input and output can be written using English and mathematical notation.

Example 6.7. Write the specification of an algorithm for computing the square root of a number.

1. Let us name the algorithm square_root.
2. It takes the number as the input. Let us name the input n. n should not be negative.
3. It produces the square root of n as the output. Let us name the output y. Then n should be the square of y.

Now the specification of the algorithm is

square_root(n)

-- inputs: n is a real number, n ≥ 0.

-- outputs: y is a real number such that y^2= n.

6.5.1 Specification as contract

Specification of an algorithm serves as a contract between the designer of the algorithm and the users of the algorithm, because it defines the rights and responsibilities of the designer and the user.

Ensuring that the inputs satisfy the required properties is the responsibility of the user, but the right of the designer. The desired input-output relation is the responsibility of the designer and the right of the user. Importantly, if the user fails to satisfy the properties of the inputs, the designer is free from his obligation to satisfy the desired input-output relation.

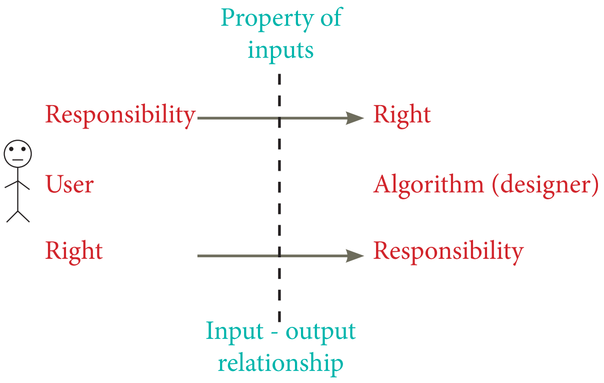


Figure 6.2: Input property and the input-output relation as rights and responsibilities

Example 6.8. Consider the specification of the algorithm square_root.

square_root(n)

-- inputs: n is a real number, n ≥ 0.

-- outputs : y is a real number such that y^2 = n.

The algorithm designer can assume that the given number is non-negative, and construct the algorithm. The user can expect the output to be the square root of the given number.

The output could be the negative square root of the given number. The specification did not commit that the output is the positive square root. If the user passes a negative number as the input, then the output need not be the square root of the number.

6.6 Abstraction

To ride a bicycle, it is sufficient to understand the functioning of the pedal, handlebar, brakes and bell. As a rider, we model a bicycle by these four features. A bicycle has a lot more details, which the rider can ignore. Those details are irrelevant for the purpose of riding a bicycle.

A problem can involve a lot of details. Several of these details are irrelevant for solving the problem. Only a few details are essential. Abstraction is the process of ignoring or hiding irrelevant details and modeling a problem only by its essential features. In our everyday life, we use abstractions unconsciously to handle complexity. Abstraction is the most effective mental tool used for managing complexity. If we do not abstract a problem adequately, we may deal with unnecessary details and complicate the solution.

Example 6.9. A map is an abstraction of the things we find on the ground. We do not represent every detail on the ground. The map-maker picks out the details that we need to know. Different maps are drawn for different purposes and so use different abstractions, i.e., they hide or represent different features. A road map is designed for drivers. They do not usually worry about hills so most hills are ignored on a road map. A walker's map is not interested in whether a road is a one-way street, so such details are ignored.

Example 6.10. In medicine, different specialists work with different abstractions of human body. An orthopaedician works with the abstraction of skeletal system, while a gastroenterologist works with digestive system. A physiotherapist abstracts the human body by its muscular system.

We use abstraction in a variety of ways while constructing algorithms — in the specification of problems, representing state by variables, and decomposing an algorithm to functions. An algorithm designer has to be trained to recognize which features are essential to solve the problem, and which details are unnecessary. If we include unnecessary details, it makes the problem and its solution over-complicated.

Specification abstracts a problem by the properties of the inputs and the desired input-output relation. We recognize the properties essential for solving the problem, and ignore the unnecessary details.

State: In algorithms, the state of a computation is abstracted by a set of variables.

Functions: When an algorithm is very complex, we can decompose it into functions and abstract each function by its specification.

6.6.1 State

State is a basic and important abstraction. Computational processes have state. A computational process starts with an initial state. As actions are performed, its state changes. It ends with a final state. State of a process is abstracted by a set of variables in the algorithm. The state at any point of execution is simply the values of the variables at that point.

Example 6.11. Chocolate Bars: A rectangular chocolate bar is divided into squares by horizontal and vertical grooves. We wish to break the bar into individual squares.

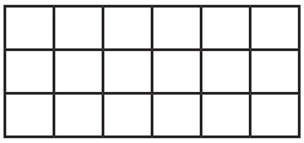
To start with, we have the whole of the bar as a single piece. A cut is made by choosing a piece and breaking it along one of its grooves. Thus a cut divides a piece into two pieces. How many cuts are needed to break the bar into its individual squares?

In this example, we will abstract the essential variables of the problem. We solve the problem in Example 8.6.

Essential variables: The number of pieces and the number of cuts are the essential variables of the problem. We will represent them by two variables, p and c, respectively. Thus, the state of the process is abstracted by two variables p and c.

Irrelevant details:

1. The problem could be cutting a chocolate bar into individual pieces or cutting a sheet of postage stamps into individual stamps. It is irrelevant. The problem is simply cutting a grid of squares into individual squares.



2. The sequence of cuts that have been made and the shapes and sizes of the resulting pieces are irrelevant too. From p and c, we cannot reconstruct the sizes of the individual pieces. But, that is irrelevant to solving the problem.

Example 6.12. Consider Example 6.2, Goat, grass and wolf problem. In this example,

we will write a specification of the problem. We will solve it in Example 7.1. The problem involves four individuals, and each is at one of the two sides of the river. This means that we can represent the state by four variables, and each of them has one of the two values. Let us name the variables as farmer, goat, grass and wolf, and their possible values L and R. A value of L means "at the left side". A value of R means "at the right side". Since the boat is always with the farmer, it is not important to introduce a variable to represent its position.

In the initial state, all four variables farmer, goat, grass, wolf have the value L.

farmer, goat, grass, wolf = L, L, L, L

In the final state, all four variables should have the value R.

farmer, goat, grass, wolf = R, R, R, R

The specification of the problem is

cross_river

-- **inputs: farmer, goat, grass, wolf = L, L, L, L**

-- **outputs: farmer, goat, grass, wolf = R, R, R, R**

subject to the two constraints that

1. the goat cannot be left alone with the grass:

   if goat = grass then farmer = goat

2. the goat cannot be left alone with the wolf:

   if goat = wolf then farmer = goat

6.6.2 Assignment statement

Variables are named boxes to store values. Assignment statement is used to store a value in a variable. It is written with the variable on the left side of the assignment operator and a value on the right side.

variable := value

When this assignment is executed, the value on the right side is stored in the variable on the left side. The assignment

m := 2

stores value 2 in variable m.

m

$\boxed{\mathbf{2}}$

If the variable already has a value stored in it, assignment changes its value to the value on the right side. The old value of the variable is lost.

The right side of an assignment can be an expression.

variable := expression

In this case, the expression is evaluated and the value of the expression is stored in the variable. If the variable exists in the expression, the current value of the variable is used in evaluating the expression, and then the variable is updated. For example, the assignment

m := m + 3

evaluates the expression m + 3 using the current value of m.

m + 3
= 2 + 3
= 5

and stores the value 5 in the variable m.

m
$\boxed{5}$

The two sides of an assignment statement are separated by the symbol :=, known as assignment operator, and read as "becomes" or "is assigned". The assignment statement

v := e

is read as v "becomes" e. Note that assignment operator is not equality operator[1]. The meanings of v := e and v = e are different. Assignment does not state a mathematical equality of a variable, but changes the value of a variable. The assignment m := m + 3 does not state that m is equal to m + 3. Rather, it changes the value of the variable m to the value of the expression m + 3.

An assignment statement can change the values of multiple variables simultaneously. In that case, the number of variables on the left side should match the number of expressions on the right side. For example, if we wish to assign to three variables v1, v2 and v3, we need 3 expressions, say, el, e2, e3.

vl, v2, v3 := el, e2 , e3

The left side is a comma-separated list of variables. The right side is a comma-separated list of expressions. To execute an assignment statement, first evaluate all the expressions on the right side using the current values of the variables, and then store them in the corresponding variables on the left side.

Example 6.13. What are the values of variables m and n after the assignments in line (1) and line (3)?

1. **m, n := 2 , 5**
2. **-- m, n = ? , ?**
3. **m,n:=m+3,n-1**
4. **-- m, n = ? , ?**

The assignment in line (1) stores 2 in variable m, and 5 in variable n.

m	**n**
2	5

The assignment in line (3) evaluates the expressions m + 3 and n - 1 using the current values of m and n as

m + 3 , n - 1

=2+3,5-1

= 5,4

and stores the values 5 and 4 in the variables m and n, respectively.

m	n
5	4

1. **m, n := 2,5**
2. **-- m, n = 2 , 5**
3. **m, n := m + 3, n - 1**
4. **-- m, n = 2 + 3, 5-1 = 5, 4**

Values of the variables after the two assignments are shown in in line (2) and line(4).

Example 6.14. In Example 6.11, we abstracted the state of the process by two

^Unfortunately, several programming languages, including C++, use the symbol = as assignment operator, and therefore, another symbol, == as equality operator.

variables p and c. The next step is to model the process of cutting the chocolate bar. When we make a single cut of a piece, the number of pieces (p) and the number of cuts (c) both increase by 1. We can model it by an assignment statement.

p, c := p + 1, c+1

Points to Remember:

which is read as p and c "become" p + 1 and c + 1, respectively.

- A programming language provides basic statements and a notation for composing compound statements.
- An algorithm is a step-by-step sequence of statements to solve a problem.
- As an algorithm is executed, a process evolves which solves the problem.
- Algorithmic problem solving involves construction of algorithms as well as proving properties of algorithms.
- The specification of an algorithm consists of the name of the algorithm (together with its inputs), the input property, and the desired input-output relation.
- Specification of an algorithm is a contract between the designer and users of the algorithm.
- Abstraction is the process of hiding or ignoring the details irrelevant to the task so as to model a problem only by its essential features.
- Specification abstracts a problem by the essential variables of the problem.
- The values of the variables in an algorithm define the state of the process.
- Assignment statement changes the values of variables, and hence the state.

Evaluation

BV4HNF

SECTION – A

Choose the correct answer

1. Which of the following activities is algorithmic in nature?

   (a) Assemble a bicycle. (b) Describe a bicycle.
   (c) Label the parts of a bicycle. (d) Explain how a bicycle works.

2. Which of the following activities is not algorithmic in nature?

   (a) Multiply two numbers. (b) Draw a kolam.
   (c) Walk in the park. (d) Swaping of two numbers.

3. Omitting details inessential to the task and representing only the essential features of the task is known as

   (a) specification (b) abstraction (c) composition (d) decomposition

4. Stating the input property and the input-output relation a problem is known

   (a) specification (b) statement (c) algorithm (d) definition

5. Ensuring the input-output relation is

   (a) the responsibility of the algorithm and the right of the user.

   (b) the responsibility of the user and the right of the algorithm.

   (c) the responsibility of the algorithm but not the right of the user.

   (d) the responsibility of both the user and the algorithm.

6. If i = 5 before the assignment i := i-1 after the assignment, the value of i is

   (a) 5 (b) 4 (c) 3 (d) 2

7. If 0 < i before the assignment i := i-1 after the assignment, we can conclude that

   (a) 0 < i (b) 0 ≤ i (c) i = 0 (d) 0 ≥i

SECTION-B

Very Short Answers

1. Define an algorithm.
2. Distinguish between an algorithm and a process.
3. Initially,

   farmer, goat, grass, wolf = L, L, L, L

   and the farmer crosses the river with goat. Model the action with an assignment statement.
4. Specify a function to find the minimum of two numbers.
5. If √2 = 1.414, and the square_root() function returns -1.414, does it violate the following specification?

   -- square_root (x)

   -- inputs: x is a real number , x ≥ 0

   -- outputs: y is a real number such that y^2=x

SECTION-C

Short Answers

1. When do you say that a problem is algorithmic in nature?
2. What is the format of the specification of an algorithm?
3. What is abstraction?
4. How is state represented in algorithms?
5. What is the form and meaning of assignment statement?
6. What is the difference between assignment operator and equality operator?

SECTION - D

Explain in detail

1. Write the specification of an algorithm hypotenuse whose inputs are the lengths of the two shorter sides of a right angled triangle, and the output is the length of the third side.

2. Suppose you want to solve the quadratic equation ax2 + bx + c = 0 by an algorithm.

   quadratic_solve (a, b, c)

   -- inputs : ?

   -- outputs: ?

   You intend to use the formula and you are prepared to handle only real number roots. Write a suitable specification.

$$x = \frac{— b \pm \surd b^{2} — 4ac}{2a}$$

3. Exchange the contents: Given two glasses marked A and B. Glass A is full of apple drink and glass B is full of grape drink. For exchanging the contents of glasses A and B, represent the state by suitable variables, and write the specification of the algorithm.

Books

1. Roland Backhouse, Algorithmic Problem Solving, John Wiley & Sons Ltd, 2011.

2. Krysia Broda, Susan Eisenbach, Hessam Khoshnevisan, Steve Vickers, Reasoned Programming, Prentice Hall, 1994

Composition and Decomposition

Learning Objectives

After learning the concepts in this chapter, the students will be able

- To know the notations used in algorithmic techniques.
- To understand Composition and Decomposition in algorithmic techniques.

In Chapter 1, we saw that algorithms are composed of statements. Statements can be grouped into compound statements, giving rise to a hierarchical structure of algorithms. Decomposition is one of the elementary problem-solving techniques.An algorithm may be broken into parts, expressing only high level details. Then, each part may be refined into smaller parts, expressing finer details, or each part may be abstracted as a function.

7.1 Notations for Algorithms

We need a notation to represent algorithms. There are mainly three different notations for representing algorithms.

- A programming language is a notation for expressing algorithms to be executed by computers.
- Pseudo code is a notation similar to programming languages. Algorithms expressed in pseudo code are not intended to be executed by computers, but for communication among people.
- Flowchart is a diagrammatic notation for representing algorithms. They give a visual intuition of the flow of control, when the algorithm is executed.

7.1.1 Programming language

A programming language is a notation for expressing algorithms so that a computer can execute the algorithm. An algorithm expressed in a programming language is called a program. C, C++ and Python are examples of programming languages. Programming language is formal. Programs must obey the grammar of the programming language exactly. Even punctuation symbols must be exact. They do not allow the informal style of natural languages such as English or Tamil. There is a translator which translates the program into instructions executable by the computer. If our program has grammatical errors, this translator will not be able to do the translation.

BVDDQ3

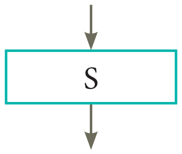
7.1.2 Pseudo-code

Pseudo code is a mix of programming-language-like constructs and plain English. This notation is not formal nor exact. It uses the same building blocks as programs, such as variables and control flow. But, it allows the use of natural English for statements and conditions. An algorithm expressed as pseudo code is not for computers to execute directly, but for human readers to understand. Therefore, there is no need to follow the rules of the grammar of a programming language. However, even pseudo code must be rigorous and correct. Pseudo code is the most widely used notation to represent algorithms.

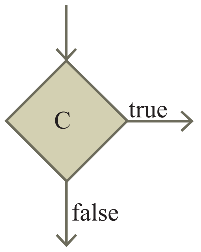
7.1.3 Flowcharts

Flowchart is a diagrammatic notation for representing algorithms. They show the control flow of algorithms using diagrams in a visual manner. In flowcharts, rectangular boxes represent simple statements, diamond-shaped boxes represent conditions, and arrows describe how the control flows during the execution of the algorithm. A flowchart is a collection of boxes containing statements and conditions which are connected by arrows showing the order in which the boxes are to be executed.

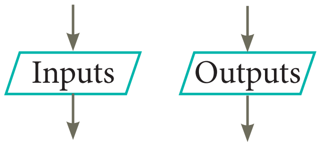
1. A statement is contained in a rectangular box with a single outgoing arrow,which points to the box to be executed next.



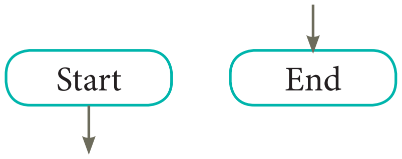
2. A condition is contained in a diamond-shaped box with two outgoing arrows, labeled true and false. The true arrow points to the box to be executed next if the condition is true, and the false arrow points to the box to be executed next if the condition is false.



3. Parallelogram boxes represent inputs given and outputs produced.



4. Special boxes marked Start and the End are used to indicate the start and the end of an execution:



The flowchart of an algorithm to compute the quotient and remainder after dividing an integer A by another integer B is shown in Figure 7.1, illustrating the different boxes such as input, output, condition, and assignment, and the control flow between the boxes. The algorithm is explained in Example 7.4.

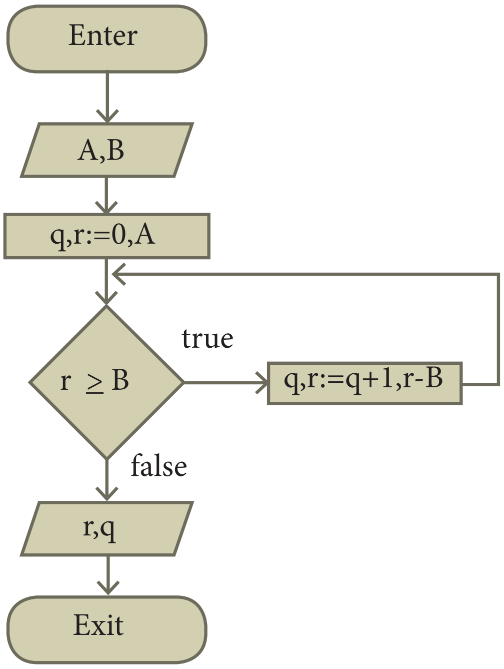


Figure 7.1: Flowchart for integer division

Flowcharts also have disadvantages. (1) Flowcharts are less compact than representation of algorithms in programming language or pseudo code. (2) They obscure the basic hierarchical structure of the algorithms. (3) Alternative statements and loops are disciplined control flow structures. Flowcharts do not restrict us to disciplined control flow structures.

7.2 Composition

A statement is a phrase that commands the computer to do an action. We have already seen assignment statement. It is a simple statement, used to change the values of variables. Statements may be composed of other statements, leading to hierarchical structure of algorithms. Statements composed of other statements are known as compound statements.

Control flow statements are compound statements. They are used to alter the control flow of the process depending on the state of the process. There are three important control flow statements:

- Sequential
- Alternative
- Iterative

When a control flow statement is executed, the state of the process is tested, and depending on the result, a statement is selected for execution.

7.2.1 Sequential statement

A sequential statement is composed of a sequence of statements. The statements in the sequence are executed one after another, in the same order as they are written in the algorithm, and the control flow is said to be sequential. Let S1 and S2 be statements. A sequential statement composed of S1 and S2 is written as

S1
S2

In order to execute the sequential statement, first do S1 and then do S2.

The sequential statement given above can be represented in a flowchart as shown in in Figure 7.2. The arrow from S1 to S2 indicates that S1 is executed, and after that, S2 is executed.

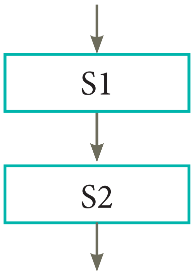


Figure 7.2: Sequential control flow

Let the input property be P, and the input-output relation be Q, for a problem. If statement S solves the problem, it is written as

1. -- P
2. S
3. -- Q

If we decompose the problem into two components, we need to compose S as a sequence of two statements S1 and S2 such that the input-output relation of S1, say R, is the input property of S2.

1. -- P
2. S1
3. -- R
4. S2
5. -- Q

Example 7.1. Let us solve the Farmer, Goat, Grass, and Wolf problem of Example 6.12. We decided to represent the state of the process by four variables farmer, goat, grass, and wolf, representing the sides of the farmer, goat, grass and wolf, respectively. In the initial state, all four variables have the value L (Left side). In the final state, all four variables should have the value R (Right side). The goal is to construct a statement S so as to move from the initial state to the final state.

1. **-- farmer, goat, grass, wolf = L, L, L, L**
2. **S**
3. **-- farmer , goat , grass , wolf = R, R, R, R**

We have to compose S as a sequence of assignment statements such that in none of the intermediate states

1. goat and wolf have the same value but farmer has the opposite value, or
2. goat and grass have the same value but farmer has the opposite value.Subject to these constraints, a sequence of assignments and the state after each assignment are shown in Figure 7.3.

1. -- farmer, goat, grass, wolf = L, L, L, L
2. farmer, goat := R, R
3. -- farmer , goat , grass , wolf = R, R, L, L
4. farmer := L
5. farmer, goat, grass, wolf = L, R, L, L
6. farmer, grass := R, R
7. -- farmer , goat , grass , wolf = R, R, R, L
8. farmer, goat := L, L
9. -- farmer, goat, grass, wolf = L, L, R, L
10. farmer, wolf := R, R
11. -- farmer , goat , grass , wolf = R, L, R, R
12. farmer : = L
13. -- farmer , goat , grass , wolf = L, L, R, R
14. farmer , goat : = R, R
15. -- farmer , goat , grass , wolf = R, R, R, R

Figure 7.3: Sequence of assignments for goat, grass and wolf problem

Other than lines (1) and (15), in line (7), goat and grass have the same value, but farmer also has the same value as they. In line (9), goat and wolf have the same value, but farmer also has the same value as they. Thus, the sequence has achieved the goal state, without violating the constraints.

7.2.2 Alternative statement

A condition is a phrase that describes a test of the state. If C is a condition and both

S1 and S2 are statements, then

```
if  C
    S1
else
    S2
```

is a statement, called an alternative statement, that describes the following action:

1. Test whether C is true or false.
2. If C is true, then do S1; otherwise do S2.

In pseudo code, the two alternatives S1 and S2 are indicated by indenting them from the keywords if and else, respectively. Alternative control flow is depicted in the flowchart of Figure 2.4. Condition C has two outgoing arrows, labeled true and false. The true arrow points to the S1 box. The false arrow points to the S2 box. Out going arrows of S1 and S2 point to the same box, the box after the alternative statement.

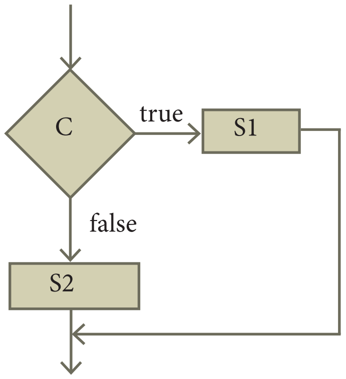


Figure 7.4: Alternative control flow

Conditional statement: Sometimes we need to execute a statement only if a condition is true and do nothing if the condition is false. This is equivalent to the alternative statement in which the else-clause is empty. This variant of alternative statement is called a conditional statement. If C is a condition and S is a statement, then

```
if  C
    S
```

is a statement, called a conditional statement, that describes the following action:

1. Test whether C is true or false.
2. If C is true then do S; otherwise do nothing.

The conditional control flow is depicted in the flowchart of Figure 2.5.

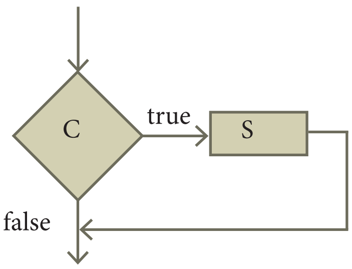


Figure 7.5: Conditional control flow

Example 7.2. Minimum of two numbers: Given two numbers a and b, we want to find the minimum of the two using the alternative statement. Let us store the minimum in a variable named result. Let a ↓ b denote the minimum of a and b (for instance, 4 ↓ 2 = 2, —5 ↓ 6 = -5). Then, the specification of algorithm minimum is

```
minimum(a, b)
-- input s : a , b
-- outputs: result = a ↓ b
```

Algorithm minimum can be defined as

```
minimum(a, b)
  -- a, b
  if a < b
    result : = a
  else
    result = b
  -- result = a ↓ b
```

7.2.3 Case analysis

Alternative statement analyses the problem into two cases. Case analysis statement generalizes it to multiple cases. Case analysis splits the problem into an exhaustive set of disjoint cases. For each case, the problem is solved independently. If C1, C2, and C3 are conditions, and S1, S2, S3 and S4 are statements, a 4-case analysis statement has the form,

```
case  C1
    S1
case  C2
    S2
case  C3
    S3
else
    S4
```

The conditions C1, C2, and C3 are evaluated in turn. For the first condition that evaluates to true, the corresponding statement is executed, and the case analysis statement ends. If none of the conditions evaluates to true, then the default case S4 is executed.

1. The cases are exhaustive: at least one of the cases is true. If all conditions are false, the default case is true.
2. The cases are disjoint: only one of the cases is true. Though it is possible for

more than one condition to be true, the case analysis always executes only one case, the first one that is true. If the three conditions are disjoint, then the four cases are (1) C1, (2) C2, (3) C3, (4) (not C1) and (not C2) and (not C3).

Example 7.3. We want an algorithm that compares two numbers and produces the result as

$$\text{compare }(a, b) = \begin{cases} 1- & \text{if } a < b \\ 0 & \text{if } a = b \\ 1 & \text{if } a > b \end{cases}$$

We can split the state into an exhaustive set of 3 disjoint cases: a < b, a = b, and a> b. Then we can define compare() using a case analysis.

```
1.  compare(a, b)
2.    case a < b
3.        result := -1
4.    case a = b
5.        result := 0
6.    else -- a > b
7.        result : = 1
```

7.2.4 Iterative statement

An iterative process executes the same action repeatedly, subject to a condition C. If C is a condition and S is a statement, then

```
while C
    S
```

is a statement, called an iterative statement, that describes the following action:

1. Test whether C is true or false.
2. If C is true, then do S and go back to step 1; otherwise do nothing.

The iterative statement is commonly known as a loop. These two steps, testing C and executing S, are repeated until C becomes false. When C becomes false, the loop ends, and the control flows to the statement next to the iterative statement. The condition C and the statement S are called the loop condition and the loop body, respectively. Testing the loop condition and executing the loop body once is called an iteration. not C is known as the termination condition.

Iterative control flow is depicted in the flowchart of Figure 7.6. Condition C has two outgoing arrows, true and false. The true arrow points to S box. If C is true, S box is executed and control flows back to C box. The false arrow points to the box after the iterative statement (dotted box). If C is false, the loop ends and the control flows to the next box after the loop.

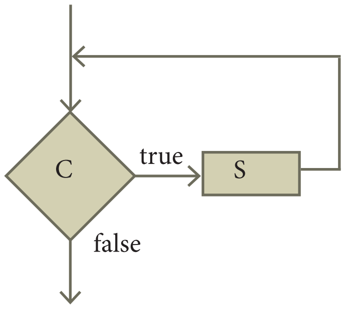


Figure 7.6: Iterative control flow

Example 7.4. Construct an iterative algorithm to compute the quotient and remainder after dividing an integer A by another integer B.

We formulated the specification of the algorithm in Example 6.6 as

```
divide (A , B)
-- inputs:    A is an integer and B ≠ 0
-- outputs :  q and r such that A = q X B
     + r and
--            0 ≤ r < B
```

Now we can construct an iterative algorithm that satisfies the specification.

```
        divide (A , B)
-- inputs: A is an integer and B ≠ 0
-- outputs : q and r such that A = q X B
      + r and
--              0 < r < B
        q, r : = 0, A
        while r ≥ B
           q, r := q + 1, r - B
```

The algorithm is presented as a flowchart in Figure 7.1.

We can execute the algorithm step-by-step for a test input, say, (A, B) = (22, 5). Each row of Table 7.1 shows one iteration — the evaluation of the expressions and the values of the variables at the end of an iteration. Note that the evaluation of the expression uses the values of the variables from the previous row. Output variables q and r change their values in each iteration. Input variables A and B do not change their values. Iteration 0 shows the values just before the loop starts. At the end of iteration 4, condition (r ≥ B) = (2 ≥ 5) is false, and hence the loop ends with (q, r) = (4, 2).

iteration	q	q+1	r	r-B	A	B
0	0		22		22	5
1	1	0+1	17	22-5		
2	2	1 + 1	12	17-5		
3	3	2+1	7	12-5		
4	4	3+1	2	7-5		

Table 7.1: Step by step execution of divide (22, 5)

Example 7.5. In the Chameleons of Chromeland problem of Example 1.3, suppose two types of chameleons are equal in number. Construct an algorithm that arranges meetings between these two types so that they change their color to the third type. In the end, all should display the same color.

Let us represent the number of chameleons of each type by variables a, b and c, and their initial values by A, B and C, respectively. Let a = b be the input property. The input-output relation is a = b = 0 and c = A+B+C. Let us name the algorithm monochromatize. The algorithm can be specified as

```
    monochromatize(a, b, c)
    -- inputs: a=A, b=B, c=C, a=b
    -- outputs : a = b = 0 , c = A+B+C
```

In each iterative step, two chameleons of the two types (equal in number) meet and change their colors to the third one. For example, if A, B, C = 4, 4, 6, then the series of meetings will result in

iteration	a	b	c
0	4	4	6
1	3	3	8
2	2	2	10
3	1	1	12
4	0	0	14

Table 7.2: Series of meetings between two types of chameleons equal in number.

In each meeting, a and b each decreases by 1, and c increases by 2. The solution can be expressed as an iterative algorithm.

```
monochromatize(a, b, c)
    -- inputs: a=A, b=B, c=C, a=b
    -- outputs: a = b = 0, c = A+B+C
          while a > 0
               a, b, c := a-1, b-1, c+2
```

The algorithm is depicted in the flowchart of Figure 7.7.

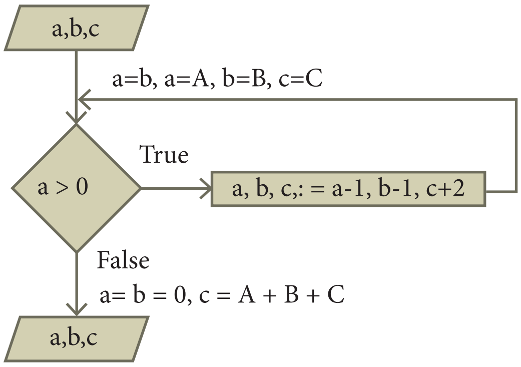


Figure 7.7: Algorithm monochromatize

7.3 Decomposition

Problem decomposition is one of the elementary problem-solving techniques. It involves breaking down a problem into smaller and more manageable problems, and combining the solutions of the smaller problems to solve the original problem. Often, problems have structure. We can exploit the structure of the problem and break it into smaller problems. Then, the smaller problems can be further broken until they become sufficiently small to be solved by other simpler means. Their solutions are then combined together to construct a solution to the original problem.

7.3.1 Refinement

After decomposing a problem into smaller subproblems, the next step is either to refine the subproblem or to abstract the subproblem.

1. Each subproblem can be expanded into more detailed steps. Each step can be further expanded to still finer steps, and so on. This is known as refinement.
2. We can also abstract the subproblem. We specify each subproblem by its input property and the input-output relation. While solving the main problem, we only need to know the specification of the subproblems. We do not need to know how the subproblems are solved.

Example 7.6. Consider a school goer's action in the morning. The action can be written as

1 Get ready for school

We can decompose this action into smaller, more manageable action steps which she takes in sequence:

1 Eat breakfast
2 Put on clothes
3 Leave home

We have refined one action into a detailed sequence of actions. However, each of these actions can be expanded into a sequence of actions at a more detailed level, and this expansion can be repeated. The action "Eat breakfast" can be expanded as

1 -- Eat breakfast
2 Eat idlis
3 Eat eggs
4 Eat bananas

The action "Put on clothes" can be expanded as

1 -- Put on clothes
2 Put on blue dress
3 Put on socks and shoes
4 Wear ID card

and "Leave home" expanded as

1 -- Leave home
2 Take the bicycle out
3 Ride the bicycle away

Thus, the entire action of "Get ready for school" has been refined as

1 -- Eat breakfast
2 Eat idlis
3 Eat eggs
4 Eat bananas

```

-- Put on clothes
Put on blue dress
Put on socks and shoes
Wear ID card

-- Leave home
Take the bicycle out
Ride the bicycle away
```

Refinement is not always a sequence of actions. What the student does may depend upon the environment. How she eats breakfast depends upon how hungry she is and what is on the table; what clothes she puts on depends upon the day of the week. We can refine the behaviour which depends on environment, using conditional and iterative statements.

```
-- Eat breakfast
if hungry and idlis on the table
    Eat idlis
if hungry and eggs on the table
    Eat eggs
if hungry and bananas on the table
    Eat bananas

-- Put on clothes
if Wednesday
    Put on blue dress
else
    Put on white dress
Put on socks and shoes
Wear the ID card

-- Leave home
Take the bicycle out
Ride the bicycle away
```

The action "Eat idlis" can be further refined as an iterative action:

```
-- Eat idlis
Put idlis on the plate
Add chutney
while idlis in plate
    Eat a bite of idli
```

How "Get ready for school" is refined in successive levels is illustrated in Figure 2.8.

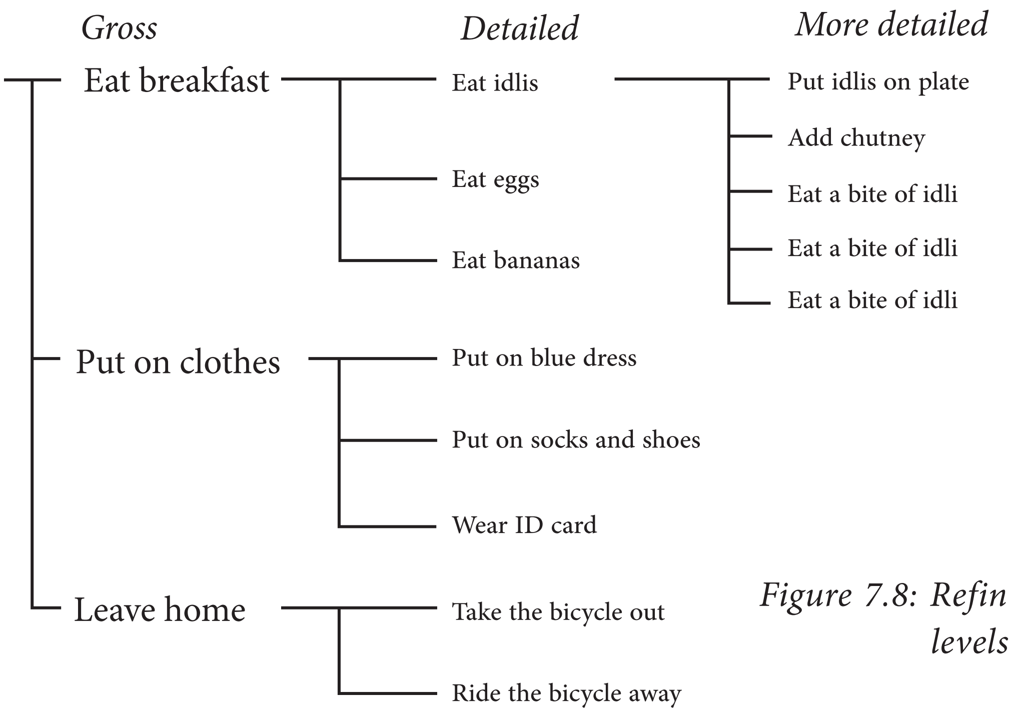


Figure 7.8: Refinement at various levels of details

The action "Eat breakfast" is depicted in a flowchart shown in Figure 2.9.

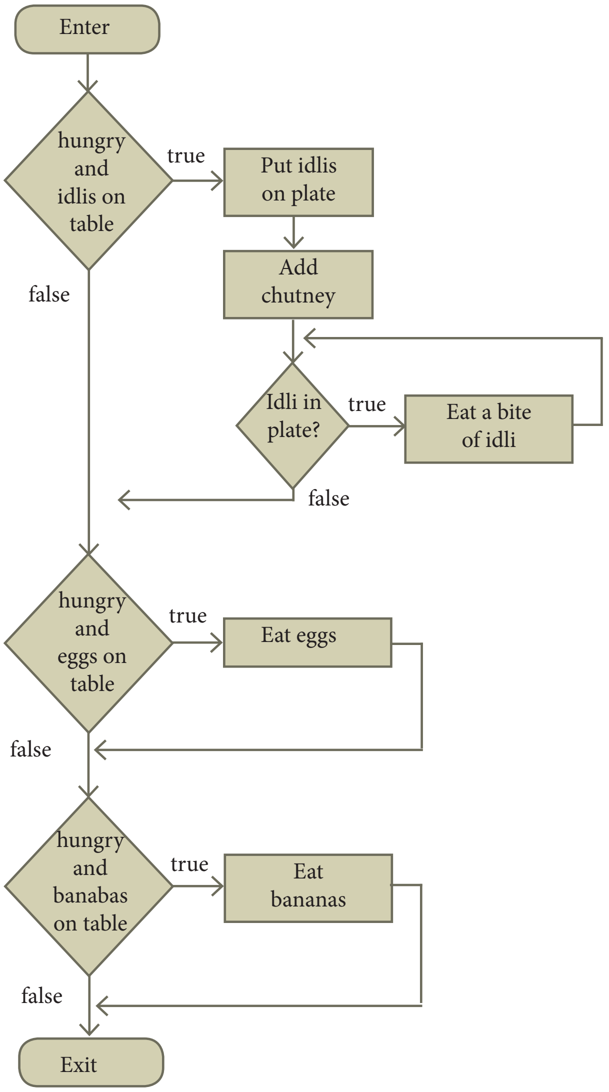


Figure 7.9: Flowchart for Eat breakfast

Note that the flowchart does not show the hierarchical structure of refinement.

7.3.2 Functions

After an algorithmic problem is decomposed into subproblems, we can abstract the subproblems as functions. A function is like a sub-algorithm. Similar to an algorithm, a function is specified by the input property, and the desired input-output relation.

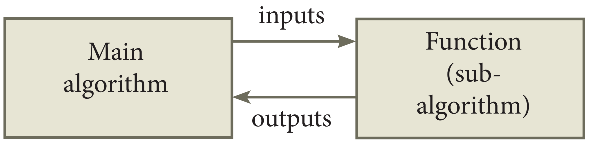


Figure 7.10: Function definition

To use a function in the main algorithm, the user need to know only the specification of the function — the function name, the input property, and the input-output relation. The user must ensure that the inputs passed to the function will satisfy the specified property and can assume that the outputs from the function satisfy the input-output relation. Thus, users of the function need only to know what the function does, and not how it is done by the function. The function can be used a a "black box" in solving other problems.

Ultimately, someone implements the function using an algorithm. However, users of the function need not know about the algorithm used to implement the function. It is hidden from the users. There is no need for the users to know how the function is implemented in order to use it.

An algorithm used to implement a function may maintain its own variables. These variables are local to the function in the sense that they are not visible to the user of the function. Consequently, the user has fewer variables to maintain in the main algorithm, reducing the clutter of the main algorithm.

Example 7.7. Consider the problem of testing whether a triangle is right-angled, given its three sides a, b, c, where c is the longest side. The triangle is right-angled, if

$$c^2 = a^2 + b^2$$

We can identify a subproblem of squaring a number. Suppose we have a function square(), specified as

square(y)

-- **inputs :** y

-- **outputs :** y^2

we can use this function three times to test whether a triangle is right-angled. square() is a "black box" — we need not know how the function computes the square. We only need to know its specification.

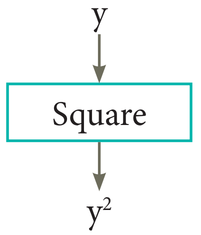


Figure 7.11: square function

```
right_angled(a, b, c)
-- inputs: $c \geq a, c \geq b$
-- outputs: result = true if $c^2 = a^2 + b^2$;
--            result = false , otherwise
if square (c) = square (a) + square (b)
    result := true
else
    result := false
```

Points to Remember

- Compound statements are composed of sequential, alternative and iterative control flow statements.
- The value of a condition is true or false, depending on the values of the variables.
- Alternative statement selects and executes exactly one of the two statements,depending on the value of the condition.
- Conditional statement is executed only if the condition is true. Otherwise, nothing is done.
- Iterative statement repeatedly evaluates a condition and executes a statement as long as the condition is true.
- Programming language, pseudo code, and flowchart are notations for expressing algorithms.
- Decomposition breaks down a problem into smaller subproblems and combine their solutions to solve the original problem.
- A function is an abstraction of a subproblem, and specified by its input property, and its input-output relation.
- Users of function need to know only what the function does, and not how it is done.
- In refinement, starting from high level, each statement is repeatedly expanded into more detailed statements in the subsequent levels.

Evaluation

SECTION – A

Choose the correct answer

1. Suppose u, v = 10 ,5 before the assignment. What are the values of u and v after the sequence of assignments?

```
u := v
v := u
```

(a) u, v = 5 ,5 (c) u, v = 10 ,5

(b) u, v = 5 ,10 (d) u, v = 10 ,10

2. Which of the following properties is true after the assignment (at line 3?

```
1 --i, j = 0, 0
2 i, j := i+1, j-1
3 -- ?
```

(a) i+j >0 (b) i+j < 0 (c) i+j =0 (d) i = j

3. If C1 is false and C2 is true, the compound statement

```
1 if C1
2     S1
3 else
4 if  C2
5     S2
6 else
7     S3
```

executes

C4D5J8

(a) S1 (b) S2 (c) S3 (d) none

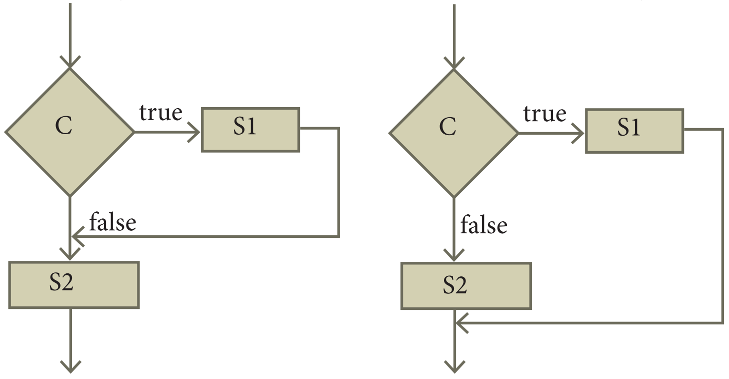
4. If C is false just before the loop, the control flows through

```
1 S1
2 while C
3 S2
4 S3
```

(a) S1 ; S3 (b) S1 ; S2 ; S3

(c)S1 ; S2 ; S2 ; S3 (d) S1 ; S2 ; S2 ; S2 ; S3

5. If C is true, S1 is executed in both the flowcharts, but S2 is executed in



(1) (2)

(a) (1) only (b) (2) only

(c) both (1) and (2) (d) neither (1) nor (2)

6. How many times the loop is iterated?

```
i := 0
while i ≠ 5
  i := i + 1
```

(a) 4 (b) 5 (c) 6 (d) 0

SECTION-B

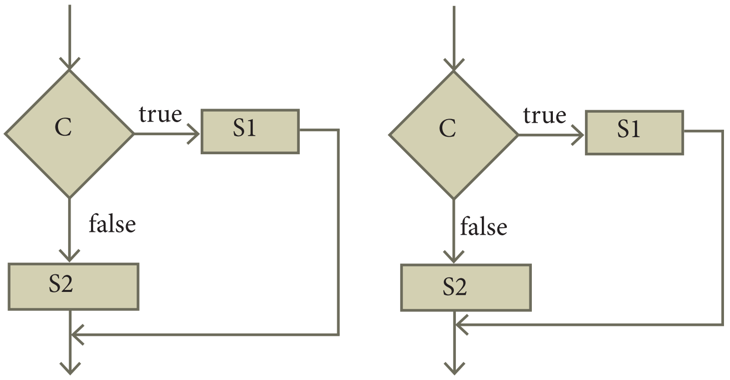
Very Short Answers

1. Distinguish between a condition and a statement.
2. Draw a flowchart for conditional statement.
3. Both conditional statement and iterative statement have a condition and a statement. How do they differ?
4. What is the difference between an algorithm and a program?
5. Why is function an abstraction?
6. How do we refine a statement?

SECTION-C

Short Answers

1. For the given two flowcharts write the pseudo code.



2. If C is false in line 2, trace the control flow in this algorithm.

```
1   S1
2   -- C is false
3   if C
4   S2
5   else
6   S3
7   S4
```

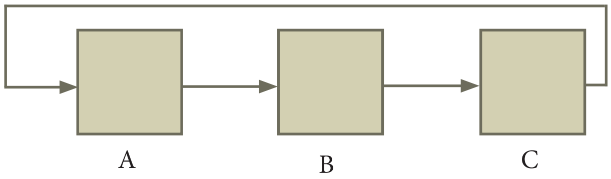
3. What is case analysis?
4. Draw a flowchart for -3case analysis using alternative statements.
5. Define a function to double a number in two different ways: (1) n + n, (2) 2 x n

SECTION - D

Explain in detail

1. Exchange the contents: Given two glasses marked A and B. Glass A is full of apple drink and glass B is full of grape drink. Write the specification for exchanging the contents of glasses A and B, and write a sequence of assignments to satisfy the specification.

2. Circulate the contents: Write the specification and construct an algorithm to circulate the contents of the variables A, B and C as shown below: The arrows indicate that B gets the value of A, C gets the value of B and A gets the value of C.



3. Decanting problem. You are given three bottles of capacities 5 ,8, and 3 litres. The 8L bottle is filled with oil, while the other two are empty. Divide the oil in 8L bottle into two equal quantities. Represent the state of the process by appropriate variables. What are the initial and final states of the process? Model the decanting of oil from one bottle to another by assignment. Write a sequence of assignments to achieve the final state.

4. Trace the step-by-step execution of the algorithm for factorial(4).

```
factorial(n)
-- inputs : n is an integer , n ≥ 0
-- outputs : f = n!
    f, i := 1 ,1
    while i ≤ n
      f, i := f × i, i+1
```

Unit II Algorithmic Problem Solving

CHAPTER 8

Iteration and recursion

Learning Objectives

After learning the concepts in this chapter, the students will be able

- To know the concepts of variants and invariants used in algorithmic techniques.
- Apply algorithmic techniques in iteration and recursion process.

There are several problems which can be solved by doing the same action repeatedly. Both iteration and recursion are algorithm design techniques to execute the same action repeatedly. What is the use of repeating the same action again and again? Even though the action is the same, the state in which the action is executed is not the same. Each time we execute the action, the state changes. Therefore, the same action is repeatedly executed, but in different states. The state changes in such a way that the process progresses to achieve the desired input-output relation.

Iteration: In iteration, the loop body is repeatedly executed as long as the loop condition is true. Each time the loop body is executed, the variables are updated. However, there is also a property of the variables which remains unchanged by the execution of the loop body. This unchanging property is called the loop invariant. Loop invariant is the key to construct and to reason about iterative algorithms.

Recursion: Recursion is another algorithm design technique, closely related to iteration, but more powerful.Using recursion, we solve a problem with a given input, by solving the same problem with a part of the input, and constructing a solution to the original problem from the solution to the partial input.

DO YOU KNOW?

E W Dijkstra was one of the most influential pioneers of Computing Science. He made fundamental contributions in diverse areas such as programming language design, operating systems, and program design. He coined the phrase "structured programming" which helped lay the foundations for the discipline of software engineering. In 1972, he was awarded ACM Turing Award, considered the highest distinction in computer science. Dijkstra is attributed to have said "Computer science is no more about computers than astronomy is about telescopes."

8.1 Invariants

Example 8.1. Suppose the following assignment is executed with (u, v) = (20,15). We can annotate before and after the assignment.

```
-- before: u, v = 20, 15
u, v :=u+5,v-5
-- after: u, v = 25, 10
```

After assignment (u, v) = (25, 10). But what do you observe about the value of the function u + v?

before: u + v = 20 + 15 = 35

after: u + v = 25 + 10 = 35

The assignment has not changed the value of u + v. We say that u + v is an invariant of the assignment. We can annotate before and after the assignment with the invariant expression.

```
-- before: u + v = 35
u, v : = u + 5, v - 5
-- after : u + v = 35
```

We can say, u + v is an invariant: it is 35 before and after. Or we can say u + v =35 is an invariant: it is true before and after.

Example 8.2. If we execute the following assignment with (p, c = 10, 9), after the assignment, (p, c) = (11, 10).

```
-- before : p, c = 10 , 9
p, c := p + 1, c+1
-- after: p, c = 11 , 10
```

Can you discover an invariant? What is the value of p - c before and after?

before: p — c = 10 — 9 = 1

after: p — c = 11 — 10 = 1

We find that p - c = 1 is an invariant.

In general, if an expression of the variables has the same value before and after an assignment, it is an invariant of the assignment. Let P(u, v) be an expression involving variables u and v. P(u, v)[u, v:= el, e2] is obtained from P(u, v) by replacing u by el and v by e2 simultaneously. P(u, v) is an invariant of assignment u, v := el, e2 if

P(u,v) [u,v := el, e2] = P(u,v)

Example 8.3. Show that p - c is an invariant of the assignment

p, c := p + 1, c + 1

Let P(p, c) = p - c. Then

P (p, c) [p, c := p + 1, c + l]

= p — c [p, c := p + 1, c + l]

= (p + 1) — (c + 1)

= p — c

= P(P , c)

Since (p - c)[p, c := p+l, c+l] = p - c, p - c is an invariant of the assignment p, c := p + 1, c + 1.

Example 8.4. Consider two variables m and n under the assignment

m, n := m + 3, n - 1

Is the expression m + 3n an invariant?

Let P(m, n) = m + 3n. Then

P(m, n) [m, n := m + 3, n — l]

= m + 3n [m, n := m + 3, n — l]

= (m + 3) + 3(n — l)

= m + 3 + 3n — 3

= m + 3n

= P(m, n)

Since (m + 3n) [m, n : = m + 3, n - 1] = m + 3n, m + 3n is an invariant of the assignment m, n := m + 3, n - l.

8.2 Loop invariant

In a loop, if L is an invariant of the loop body B, then L is known as a loop invariant.

```
while C
-- L
B
-- L
```

The loop invariant is true before the loop body and after the loop body, each time. Since L is true at the start of the first iteration, L is true at the start of the loop also (just before the loop). Since L is true at the end of the last iteration, L is true when the loop ends also (just after the loop). Thus, if L is a loop variant, then it is true at four important points in the algorithm, as annotated in the algorithm and shown in Figure 3.1.

1. at the start of the loop (just before the loop)
2. at the start of each iteration (before loop body)
3. at the end of each iteration (after loop body)
4. at the end of the loop (just after the loop)

```
1.  -- L, start of loop
      while
         C
2.    -- L, start of iteration
         B
3.       -- L, end of iteration
4.  -- L, end of loop
```

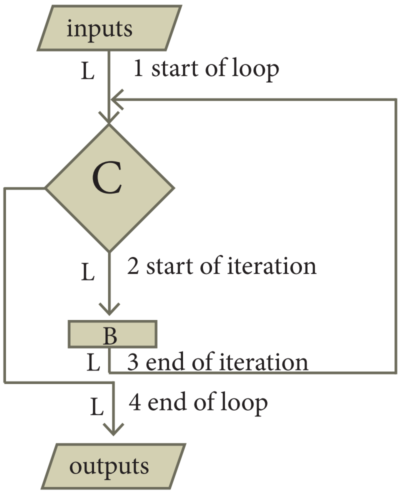


Figure 8.1: The points where the loop invariant is true

To construct a loop,

1. Establish the loop invariant at the start of the loop.
2. The loop body should update the variables, so as to progress toward the end, and maintain the loop invariant, at the same time.
3. When the loop ends, the termination condition and the loop invariant should establish the input-output relation.

8.3 Invariants — Examples

The loop invariant is true in four crucial points in a loop. Using the loop invariant, we can construct the loop and reason about the properties of the variables at these points.

Example 8.5. Design an iterative algorithm to compute a^n. Let us name the algorithm power(a, n). For example,

power(10, 4) = 10000

power (5, 3) = 125

power (2, 5) = 32

Algorithm power(a, n) computes a^n by multiplying a cumulatively n times.

$$a^n = \underbrace{a x a x \ldots x a}_{\text{n times}}$$

The specification and the loop invariant are shown as comments.

```
power (a, n)
-- inputs: n is a positive integer
-- outputs: $p = a^n$
p, i := 1, 0
while $i \neq n$
   -- loop invariant: $p = a^i$
   p, i :=p X a, i+1
```

The step by step execution of power (2, 5) is shown in Table 8.1. Each row shows the values of the two variables p and i at the end of an iteration, and how they are calculated. We see that $p = a^i$ is true at the start of the loop, and remains true in each row. Therefore, it is a loop invariant.

iteration	p	p x a	i	i+1	a^i
0	1		0		2^0
1	2	1x2	1	0 + 1	2^1
2	4	2x2	2	1 + 1	2^2
3	8	4x2	3	2 + 1	2^3
4	16	8x2	4	3 + 1	2^4
5	32	16x2	5	4+1	2^5

Table 8.1: Trace of power (2, 5)

When the loop ends, $p = a^1$ is still true, but i = 5. Therefore, $p = a^5$. In general, when the loop ends, $p = a^n$. Thus, we have verified that power(a, n) satisfies its specification.

Example 8.6. Recall the Chocolate bar problem of Example 6.11. How many cuts are needed to break the bar into its individual squares?

We decided to represent the number of pieces and the number of cuts by variables p and c respectively. Whenever a cut is made, the number of cuts increases by one and the number of pieces also increases by one. We decided to model it by an assignment.

p, c := p + 1, c+1

The process of cutting the bar can be modeled by a loop. We start with one piece and zero cuts, p = 1 and c = 0. Let n be the number of individual squares. When the number of pieces p equals the number of individual squares n, the process ends.

```
p, c : = 1 , 0
while p ≠ n
    p, c := p + 1, c+1
```

We have observed (in Example 8.2) that p - c is an invariant of the assignment p, c := p + 1, c + 1. Let p - c = k, where k is a constant. The points in the algorithm where p - c = k is true are shown in the algorithm below, and in the flowchart of Figure 8.2.

```
      p, c : = 1 , 0
1.  -- p - c = k
    while p ≠ n
2.      -- p - c = k
        p, c := p+1, c+1
3.      -- p - c = k
4.  --p-c=k,p=n
```

The loop invariant p- c = k is True at the start of the loop (line 1). Moreover, at the start of the loop, p- c = 1. Therefore, k = 1, and the loop invariant is p - c = 1

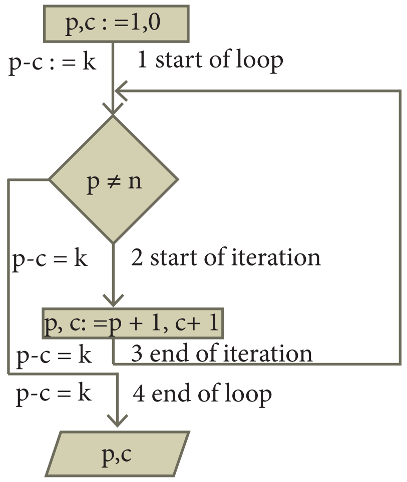


Figure 8.2: The points where the loop invariant is true

When the loop ends (line 4), the loop invariant is still true (p - c = 1). Moreover, the loop condition is false (p = n). From p - c = 1 and p = n,

1. **p — c = 1 loop invariant**
2. **p = n   end of the loop**
3. **n — c = 1 from 1, 2**
4. **c = n — 1 from 3**

When the process ends, the number of cuts is one less than the number of squares.

Example 8.7. There are 6 equally spaced trees and 6 sparrows sitting on these trees,one sparrow on each tree. If a sparrow flies from one tree to another, then at the same time, another sparrow flies from its tree to some other tree the same distance away, but in the opposite direction. Is it possible for all the sparrows to gather on one tree?

Let us index the trees from 1 to 6. The index of a sparrow is the index of the tree it is currently sitting on. A pair of sparrows flying can be modeled as an iterative step of a loop. When a sparrow at tree i flies to tree i + d, another sparrow at tree j flies to tree j — d. Thus, after each iterative step, the sum S of the indices of the sparrows remains invariant. Moreover, a loop invariant is true at the start and at the end of the loop.

At the start of the loop, the value of the invariant is

$$\mathbf{S = 1 + 2 + 3 + 4 + 5 + 6 = 21}$$

When the loop ends, the loop invariant has the same value. However, when the loop ends, if all the sparrows were on the same tree, say k, then S = 6k.

S = 21,	loop invariant at the start of the loop
S = 6k,	loop invariant at end of the loop
6k= 21,	loop invariant has the same value at the start and the end
21 is a multiple of 6	

It is not possible — 21 is not a multiple of 6. The desired final values of the sparrow indices is not possible with the loop invariant. Therefore, all the sparrows cannot gather on one tree.

Example 8.8. Consider the Chameleons of Chromeland of Example 6.3. There are 13 red, 15 green, and 17 blue chameleons on Chromeland. When two chameleons of different colors meet they both change their color to the third one (for example, if a red and a green meet, both become blue). Is it possible to arrange meetings that result in all chameleons displaying blue color?

Let r, g, and b be the numbers of red, green and blue chameleons. We can model the meetings of two types as an iterative process. A meeting changes (r, g, b) into (r-1, g-1, b+2) or (r-1, g+2, b-1) or (r+2, g-1, b-1). Consider, for example, the meeting of a red and a green chameleon.

r, g, b := r-1, g-1, b+2

The difference in the numbers of any two types either do not change or changes by 3. This is an invariant.

$$\mathbf{r - 1 - (g - 1) = r - g}$$
$$\mathbf{r - 1 - (b + 2) = (r - b) - 3}$$
$$\mathbf{g - 1 - (b + 2) = (g - b) - 3}$$

This is true for all three cases. If any two types differ in number by a multiple of 3 at the start of the iterative process, the difference can be reduced in steps of 3, to 0, when the iterative process ends. However, at the start,

$$\mathbf{r - g \quad = 13 - 15 \qquad = -2}$$

g - b = 15 - 17 = -2

b - r = 17 - 13 = 4

No two colors differ in number by a multiple of 3. Therefore, all the chameleons cannot be changed to a single color.

Example 8.9. Jar of marbles: You are given a jar full of two kinds of marbles, white and black, and asked to play this game. Randomly select two marbles from the jar. If they are the same color, throw them out, but put another black marble in (you may assume that you have an endless supply of spare marbles). If they are different colors, place the white one back into the jar and throw the black one away. If you knew the original numbers of white and black marbles, what is the color of the last marble in the jar?

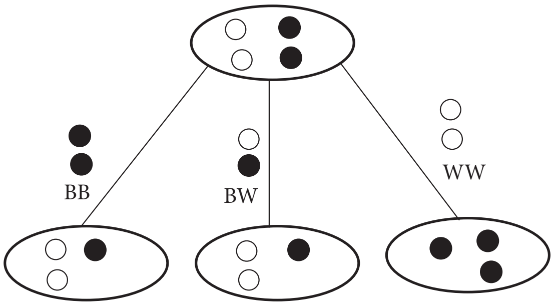


Figure 8.3: State changes in the jar marbles

The number of white and black marbles in the jar can be represented by two variables w and b. In each iterative step, b and w change depending on the colors of the two marbles taken out: Black Black, Black White or White White. It is illustrated in Figure 8.3 and annotated in the algorithm below.

```
while at least two marbles in jar
    -- b , w
    take out any two marbles
    case both are black -- BB
        throw away both the marbles
        put a black marble back
        -- b = b '-1, w = w',  b+w = b'+w' -1
    case both are white  --WW
        throw away both the marbles
        put a black marble back
        --b = b'+1, w = w'-2, --b+w = b'+w'-1
    else                    --BW
        throw away the black one
        put the white one back
        -- b = b'-1, w = w', b+w = b'+w'-1
```

For each case, how b, w and b+w change is shown in the algorithm, where b' and w' are values of the variables before taking out two marbles. Notice the way w changes. Either it does not change, or decreases by 2. This means that the parity of w, whether it is odd or even, does not change. The parity of w is invariant.

Suppose, at the start of the game, w is even. When the game ends, w is still even. Moreover, only one marble is left, w+b = 1.

1	w + b = 1	end of the loop
2	w = 0orw = 1	from 1
3	w is even	loop invariant
4	w = 0	from 2,3
5	b = 1	from 1,4

Last marble must be black. Similarly, if at the start of the game, there is an odd number of whites, the last marble must be white.

One last question: do we ever reach a state with only one marble? Yes, because the total number of marbles b+w always decreases by one at each step, it will eventually become 1.

8.4 Recursion

Recursion is an algorithm design technique, closely related to induction. It is similar to iteration, but more powerful. Using recursion, we can solve a problem with a given input, by solving the instances of the problem with a part of the input.

Example 8.10. Customers are waiting in a line at a counter. The man at the counter wants to know how many customers are waiting in the line.

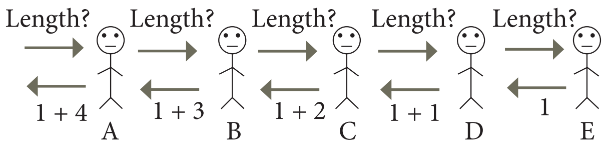


Figure 8.4: Length of a line

Instead of counting the length himself, he asks customer A for the length of the line with him at the head, customer A asks customer B for the length of the line with customer B at the head, and so on. When the query reaches the last customer in the line, E, since there is no one behind him, he replies 1 to D who asked him. D replies 1+1 = 2 to C, C replies 1+2 = 3 to B, B replies 1+3 = 4 to A, and A replies 1+4= 5 to the man in the counter

8.4.1 Recursive process

Example 8.10 illustrates a recursive process. Let us represent the sequence of 5 customers A, B, C, D and E as

[A,B,C,D,E]

The problem is to calculate the length of the sequence [A,B,C,D,E]. Let us name our solver length. If we pass a sequence as input, the solver length should output the length of the sequence.

length [A,B,C,D,E] = 5

Solver length breaks the sequence [A,B,C,D,E] into its first customer and the rest of the sequence.

first [A ,B,C,D,E] = A

rest [A ,B,C,D,E] = [B ,C,D,E]

To solve a problem recursively, solver length passes the reduced sequence [B,C,D,E] as input to a sub-solver, which is another instance of length. The solver assumes that the sub-solver outputs the length of [B,C,D,E], adds 1, and outputs it as the length of [A,B,C,D,E].

length [A,B,C,D,E] = 1 + length [B,C,D,E]

Each solver

1. receives an input,
2. passes an input of reduced size to a sub-solver,
3. receives the solution to the reduced input from the sub-solver, and produces the solution for the given input

as illustrated in Figure 8.5.

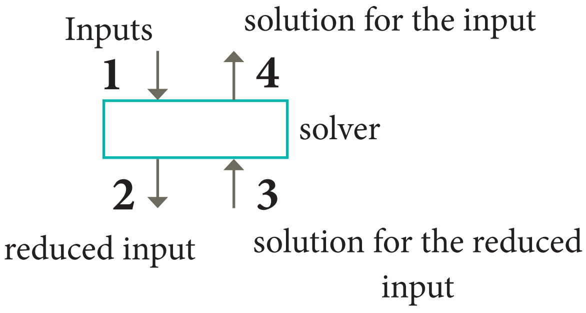


Figure 8.5: One instance of a solver in a recursive process

Figure 8.6 shows the input received and the solution produced by each

solver for Example 8.10. Each solver reduces the size of the input by one and passes it on to a sub-solver, resulting in 5 solvers. This continues until the input received by a solver is small enough to output the solution directly. The last solver received [E] as the input. Since [E] is small enough, the solver outputs the length of [E] as 1 immediately, and the recursion stops.

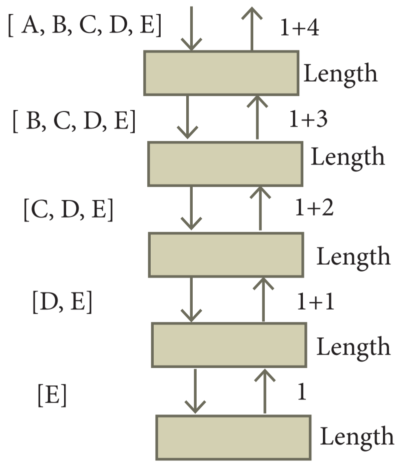


Figure 8.6: Recursive process with solvers and sub-solvers

The recursive process for length [A,B,C,D,E] is shown in Figure 8.7.

1 length [A,B,C,D,E]

2 = 1 + length [B,C,D,E]

3 = 1 +1 + length [C,D,E]

4 = 1 + 1+ 1 + length [D,E]

5 = 1 + 1+ 1 + 1 + length [E]

6 = 1 + 1 + 1 + 1 +1

7 = 1 +1 +1 + 2

8 = 1 + 1 + 3

9 = 1 + 4

10 = 5

Figure 8.7: Recursive process for computing the length of a sequence

8.4.2 Recursive problem solving

Each solver should test the size of the input. If the size is small enough, the solver should output the solution to the problem directly. If the size is not small enough, the solver should reduce the size of the input and call a sub-solver to solve the problem with the reduced input. For Example 8.10, solver's algorithm can be expressed as

$$\text{length of sequence} = \begin{cases} 1 & \text{if sequence has only one customer} \\ 1 + \text{length of tail,} & \text{otherwise} \end{cases}$$

To solve a problem recursively, the solver reduces the problem to sub-problems, and calls another instance of the solver, known as sub-solver, to solve the sub-problem. The input size to a sub-problem is smaller than the input size to the original problem. When the solver calls a sub-solver, it is known as recursive call. The magic of recursion allows the solver to assume that the sub-solver (recursive call) outputs the solution to the sub-problem. Then, from the solution to the sub-problem, the solver constructs the solution to the given problem.

As the sub-solvers go on reducing the problem into sub-problems of smaller sizes, eventually the sub-problem becomes small enough to be solved directly, without recursion. Therefore, a recursive solver has two cases:

1. **Base case:** The problem size is small enough to be solved directly. Output the solution. There must be at least one base case.
2. **Recursion step:** The problem size is not small enough. Deconstruct the problem into a sub-problem, strictly smaller in size than the given problem. Call a sub-solver to solve the sub-problem. Assume that the sub-solver outputs the solution to the sub-

problem. Construct the solution to the given problem.

This outline of recursive problem solving technique is shown below.

```
solver (input)
  if input is small enough
  construct solution
else
  find sub_problems of reduced
  input
  solutions to sub_problems =
  solver  for each sub_problem
  construct solution to the
  problem from
solutions to the sub_problems
```

Whenever we solve a problem using recursion, we have to ensure these two cases: In the recursion step, the size of the input to the recursive call is strictly smaller than the size of the given input, and there is at least one base case.

8.4.3 Recursion — Examples

Example 8.11. The recursive algorithm for length of a sequence can be written as

```
length (s)
-- inputs : s
-- outputs : length of s
  if s has one customer -- base case
    1
  else
    1 + length(tail(s)) -- recursion step
```

Example 8.12. Design a recursive algorithm to compute a^n. We constructed an iterative algorithm to compute a^n in Example 8.5. a^n can be defined recursively as

$$a^n = \begin{cases} 1 & \text{if } n = 0 \\ a \times a^{n-1} & \text{otherwise} \end{cases}$$

The recursive definition can be expressed as a recursive solver for computing power(a, n).

```
power (a, n)
-- inputs: n is an integer , n ≥ 0
-- outputs : a^n
if n = 0 -- base case
  1
else    --recursion step
        a × power (a, n-1)
```

The recursive process with solvers for calculating power(2, 5) is shown in Figure 8.8.

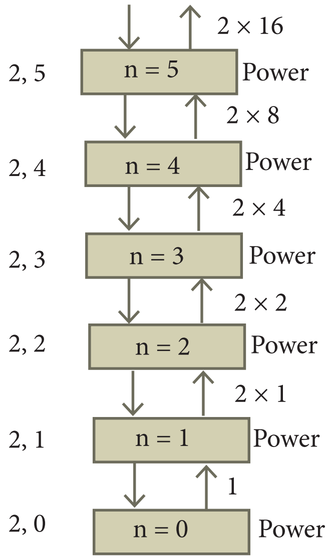


Figure 8.8: Recursive process with solvers for calculating power(2, 5)

The recursive process resulting from power(2, 5) is shown in Figure 8.9.

power (2,5)
= 2 × power (2,4)
= 2 × 2 × power(2,3)
= 2 × 2 × 2 × power(2, 2)
= 2 × 2 × 2 × 2 × power (2,1)
= 2 × 2 × 2 × 2 × 2 × power (2,0)
= 2 × 2 × 2 × 2 × 2 × 1
= 2 × 2 × 2 × 2 × 2
= 2 × 2 × 2 × 4
= 2 × 2 × 8
= 2 × 16
= 32

Figure 8.9: Recursive process for power(2, 5)

Example 8.13. A corner-covered board is a board of $2^n \times 2^n$ squares in which the square at one corner is covered with a single square tile. A triominoe is a L-shaped tile formed with three adjacent squares (see Figure 8.10). Cover the corner-covered board with the L-shaped triominoes without overlap. Triominoes can be rotated as needed.

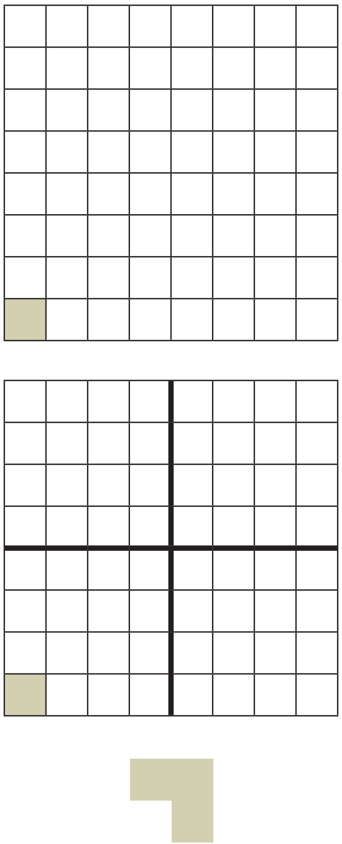


Figure 8.10: Corner-covered board and triominoe

The size of the problem is n (board of size $2^n \times 2^n$). We can solve the problem by recursion. The base case is n = 1. It is a 2 × 2 corner-covered board. We can cover it with one triominoe and solve the problem. In the recursion step, divide the corner-covered board of size $2^n \times 2^n$ into 4 sub-boards, each of size $2^{n-1} \times 2^{n-1}$, by drawing horizontal and vertical lines through the centre of the board. Place a triominoe at the center of the entire board so as to not cover the corner-covered sub-board, as shown in the left-most board of Figure 8.11. Now, we have four corner-covered boards, each of size $2^{n-1} \times 2^{n-1}$.

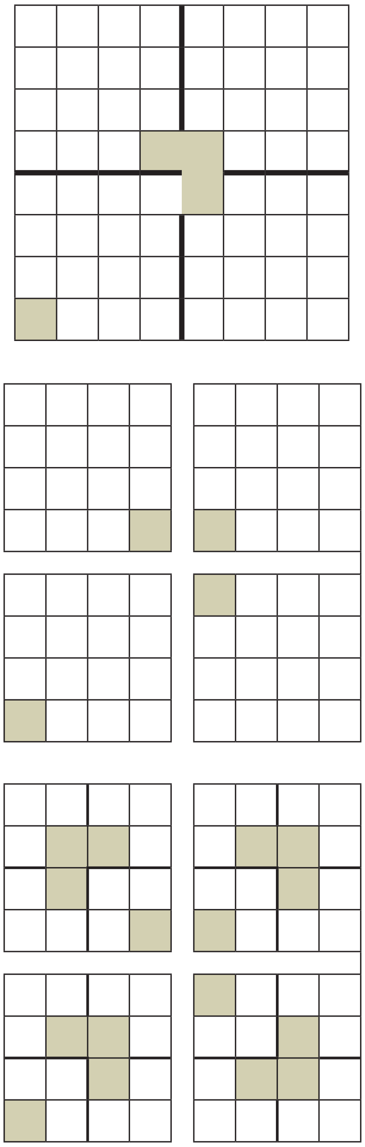
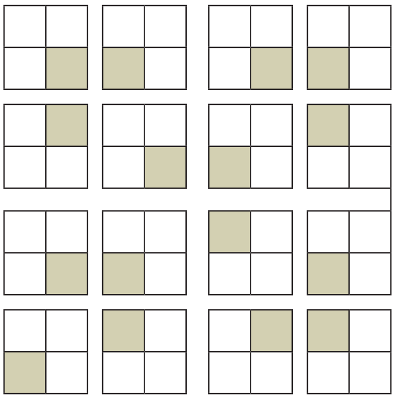
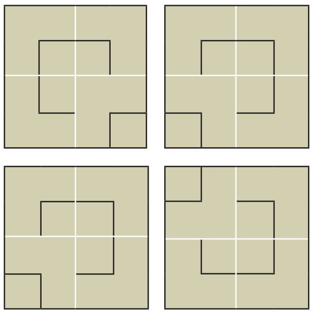


Figure 8.11: Recursive process of covering a corner-covered board of size 2 x 2^3





We have 4 sub-problems whose size is strictly smaller than the size of the given problem. We can solve each of the sub-problems recursively.

```
tile corner_covered board of size n
 if n = 1 -- base case
        cover the 3 squares with one triominoe
else    -- recursion step
        divide board into 4 sub_boards of size n-1
        place a triominoe at centre of board ,
        leaving out the corner_covered sub -board
        tile each sub_board of size n-1
```

The resulting recursive process for covering a 2^3 x 2^3 corner-covered board is illustrated in *Figure 8.11.*

Points to Remember

- Iteration repeats the two steps of evaluating a condition and executing a statement, as long as the condition is true.
- An expression involving variables, which remains unchanged by an assignment to one of these variables, is called an invariant of the assignment.
- An invariant for the loop body is known as a loop invariant.
- A loop invariant is true.
- (a) at the start of the loop (just before the loop)
- (b) at the start of each iteration (before loop body)
- (c) at the end of each iteration (after loop body)
- (d) at the end of the loop (just after the loop)
- When a loop ends, the loop invariant is true. In addition, the termination condition is also true.
- Recursion must have at least one base case.
- Recursion step breaks the problem into sub-problems of smaller size, assumes solutions for sub-problems are given by recursive calls, and constructs solution to the given problem.
- In recursion, the size of input to a sub-problem must be strictly smaller than the size of the given input.

Evaluation

SECTION – A

Choose the correct answer

1. A loop invariant need not be true

   (a) at the start of the loop. (b) at the start of each iteration

   (c) at the end of each iteration (d) at the start of the algorithm

2. We wish to cover a chessboard with dominoes, □□ the number of black squares and the number of white squares covered by dominoes, respectively, placing a domino can bemodeled by

   (a) b := b + 2 (b) w := w + 2 (c) b, w := b+1, w+1 (d) b := w

3. If m x a + n x b is an invariant for the assignment a, b : = a + 8, b + 7, the values of m and n are

   (a) m = 8, n = 7 (b) m = 7, n = -8 (c) m = 7, n = 8 (d) m = 8, n = -7

4. Which of the following is not an invariant of the assignment?

   m, n := m+2, n+3

   (a) m mod 2 (b) n mod 3 (c) 3 X m - 2 X n (d) 2 X m - 3 X n

5. If Fibonacci number is defined recursively as

$$\mathbf{F(n)} = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F(n-1) + F(n-2) & \text{otherwise} \end{cases}$$

   to evaluate F(4), how many times F() is applied?

   (a) 3 (b) 4 (c) 8 (d) 9

6. Using this recursive definition

$$\mathbf{a}^{n} = \begin{cases} 1 & \text{if } n = 0 \\ a \text{ x } a^{n-1} & \text{otherwise} \end{cases}$$

   how many multiplications are needed to calculate a^{10}?

   (a) 11 (b) 10 (c) 9 d) 8

SECTION-B

Very Short Answers

1. What is an invariant?
2. Define a loop invariant.
3. Does testing the loop condition affect the loop invariant? Why?
4. What is the relationship between loop invariant, loop condition and the input- output recursively

5. What is recursive problem solving?

6. Define factorial of a natural number recursively.

SECTION-C

Short Answers

1. There are 7 tumblers on a table, all standing upside down. You are allowed to turn any 2 tumblers simultaneously in one move. Is it possible to reach a situation when all the tumblers are right side up? (Hint: The parity of the number of upside down tumblers is invariant.)

2. A knockout tournament is a series of games. Two players compete in each game; the loser is knocked out (i.e. does not play any more), the winner carries on. The winner of the tournament is the player that is left after all other players have been knocked out. Suppose there are 1234 players in a tournament. How many games are played before the tournament winner is decided?

3. King Vikramaditya has two magic swords. With one, he can cut off 19 heads of a dragon, but after that the dragon grows 13 heads. With the other sword, he can cut off 7 heads, but 22 new heads grow. If all heads are cut off, the dragon dies. If the dragon has originally 1000 heads, can it ever die? (Hint:The number of heads mod 3 is invariant.)

SECTION - D

Explain in detail

1. Assume an 8 × 8 chessboard with the usual coloring. "Recoloring" operation changes the color of all squares of a row or a column. You can recolor re-peatedly. The goal is to attain just one black square. Show that you cannot achieve the goal. (Hint: If a row or column has b black squares, it changes by (|8 - b) - b|).

2. Power can also be defined recursively as

$$\mathbf{a}^{\mathbf{n}} = \begin{cases} 1 & \text{if } n = 0 \\ a \times a^{n-1} & \text{if } n \text{ is odd} \\ a^{n/2} \times a^{n/2} & \text{if } n \text{ is even} \end{cases}$$

Construct a recursive algorithm using this definition. How many multiplications are needed to calculate a^{10}?

3. A single-square-covered board is a board of 2^n x 2^n squares in which one square is covered with a single square tile. Show that it is possible to cover the this board with triominoes without overlap.

Unit III Introduction to C++

CHAPTER 9

Introduction to C++

F9DFGC

Learning Objectives

After the completion of this chapter, the student will be able to

- Understand the basic building blocks of C++ programming language
- Able to construct simple C++ programs
- Execute and debug C++programs

9.1 Introduction

C++ is one of the most popular programming language which supports both procedural and Object Oriented Programming paradigms. Thus, C++ is called as a **hybrid language.** C++ is a superset (extension) of its predecessor C language. Bjarne Stroustrup named his new language as "C with Classes". The name C++ was coined by Rick Mascitti where ++ is the C language increment operator.

Bjarne Stroustrup

Inventor of C++ Programming Language

Bjarne is a Danish Computer Scientist born on 30th December 1950. He has a Master degree in Mathematics and Computer Science in 1975 from Aarhus University, Denmark and Ph.D in Computer Science in 1979 from the University of Cambridge, England.

History of C++

C++ was developed by Bjarne Stroustrup at AT & T Bell Laboratory during 1979. C++ is originally derived from C language and influenced by many languages like Simula, BCPL, Ada, ML, CLU and ALGOL 68. Till 1983, it was referred "New C" and "C with Classes". In 1983, the name was changed as C++ by Rick Mascitti.

Benefits of learning C++

- C++ is a highly portable language and is often the language of choice for multi-device, multi-platform app development.
- C++ is an object-oriented programming language and includes **classes**, **inheritance**, **polymorphism**, **data abstraction** and **encapsulation**.
- C++ has a rich function library.
- C++ allows exception handling, inheritance and function overloading which are not possible in C.
- C++ is a powerful, efficient and fast language. It finds a wide range of applications – from GUI applications to 3D graphics for games to real-time mathematical simulations.

9.2 Character set

Character set is a set of characters which are used to write a C++ program. A character represents any alphabet, number or any other symbol (special characters) mostly available in the keyboard. C++ accepts the following characters.

<table>
<tr><td>Alphabets</td><td>A …. Z, a ….. z</td></tr>
<tr><td>Numeric</td><td>0 …. 9</td></tr>
<tr><td>Special Characters</td><td>+ - * / ~ ! @ # $ % ^&amp; [ ] ( ) { } = &gt;&lt; _ \ | ? . , : ‘ “ ;</td></tr>
<tr><td>White space</td><td>Blank space, Horizontal tab (→), Carriage return (↵), Newline, Form feed</td></tr>
<tr><td>Other characters</td><td>C++ can process any of the 256 ASCII characters as data.</td></tr>
</table>

9.3 Lexical Units (Tokens):

C++ program statements are constructed by many different small elements such as commands, variables, constants and many more symbols called as operators and punctuators. These individual elements are collectively called as **Lexical units** or **Lexical elements** or **Tokens**. C++ has the following tokens:

- Keywords
- Identifiers
- Literals
- Operators
- Punctuators

TOKEN:

The smallest individual unit in a program is known as a Token or a Lexical unit

9.3.1 Keywords

Keywords are the reserved words which convey specific meaning to the C++ compiler. They are the essential elements to construct C++ programs. Most of the keywords are common to C, C++ and Java.

C++ is a case sensitive programming language so, all the keywords must be in lowercase.

Table 9.1 C++ Keywords

asm	auto	break	case	catch
char	class	const	continue	default
delete	do	double	else	enum
extern	float	for	friend	goto
if	inline	int	long	new
operator	private	protected	public	register
return	short	signed	sizeof	static
struct	switch	template	this	throw
try	typedef	union	unsigned	virtual
void	volatile	while		

- With revisions and additions, the recent list of keywords also includes:

using, namespace, bal, static_cast, const_cast, dynamic_cast, true, false

- Identifiers containing a double underscore are reserved for use by C++ implementations and standard libraries and should be avoided by users.

9.3.2 Identifiers

Identifiers are the user-defined names given to different parts of the C++ program viz. variables, functions, arrays, classes etc., These are the fundamental building blocks of a program. Every language has specific rules for naming the identifiers.

Rules for naming an identifier:

- The first character of an identifier must be an alphabet or an underscore (_).
- Only alphabets, digits and underscore are permitted. Other special characters are not allowed as part of an identifier.
- C++ is case sensitive as it treats upper and lower-case characters differently.
- Reserved words or keywords cannot be used as an identifier name.

As per ANSI standards, C++ places no limit on its length and therefore all the characters are significant.

Identifiers	Valid / Invalid	Reason for invalid
Num	Valid	
NUM	Valid	
_add	Valid	
total_sales	Valid	
tamilMark	Valid	
num-add	Invalid	Contains special character (-)
this	Invalid	This is one of the keyword. Keyword cannot be used as identifier names.
2myfile	Invalid	Name must start begins with an alphabet or an underscore

- You may use an underscore in variable names to separate different parts of the name (eg: ***total_sales*** is a valid identifier where as the variable called ***total sales*** is an invalid identifier).
- You may use capital style notation, such as ***tamilMark*** ie. capitalizing the first letter of the second word.

9.3.3 Literals (Constants)

Literals are data items whose values do not change during the execution of a program. Therefore Literals are called as Constants. C++ has several kinds of literals:

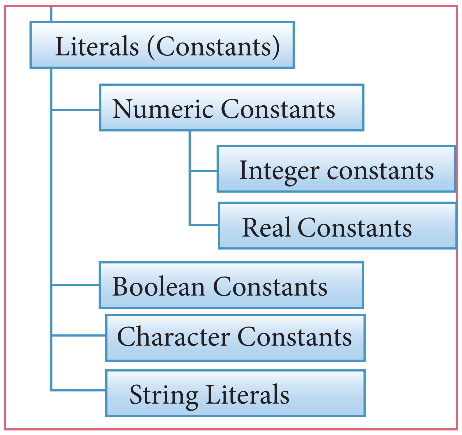


Figure 9.1 Types of Constants

Numeric Constants:

As the name indicates, the numeric constants are numeric values, which are used as constants. Numeric constants are further classified as:

1. Integer Constants (or) Fixed point constants.
2. Real constants (or) Floating point constants.

(1) Integer Constants (or) Fixed point constants

Integers are whole numbers without any fractions. An integer constant must have at least one digit without a decimal point. It may be signed or unsigned. Signed integers are considered as negative, commas and blank spaces are not allowed as part of it. In C++, there are three types of integer constants: (i) Decimal (ii) Octal (iii) Hexadecimal

(i) Decimal

Any sequence of one or more digits (0 …. 9)

Valid	Invalid
725	7,500 (Comma is not allowed)
-27	66 5(Blank space is not allowed)
4.56	9$ (Special Character not allowed)

If you assign **4.56** as an integer decimal constant, the compiler will accept only the integer portion of **4.56** ie. **4**. It will simply ignore **.56.**

Notes

If a Decimal constant declared with fractions, then the compiler will take only the integer part of the value and it will ignore its fractional part. This is called as "Implicit Conversion". It will be discussed later.

(ii) Octal

Any sequence of one or more octal values (0 …. 7) that begins with 0 is considered as an Octal constant.

Valid	Invalid
012	05,600(Commas is not allowed)
-027	04.56 (Decimal point is not allowed)**
+0231	0158 (8 is not a permissible digit in octal system)

Notes

** When you use a fractional number that begins with 0, C++ considers the number as an integer not an Octal.

(iii) Hexadecimal

Any sequence of one or more Hexadecimal values (0 …. 9, A …. F) that starts with 0x or 0Xis considered as an Hexadecimal constant.

Valid	Invalid
0x123	0x1,A5 (Commas is not allowed)
0X568	0x.14E (Decimal point is not allowed like this)

The suffix L or l and U or u added with any constant forces it to be represented as a long or unsigned constant respectively.

(2) Real Constants (or) Floating point constants

A real or floating point constant is a numeric constant having a fractional component. These constants may be written in fractional form or in exponent form.

Fractional form of a real constant is a signed or unsigned sequence of digits including a decimal point between the digits. It must have at least one digit before and after a decimal point. It may be prefixed with + or - sign. A real constant without any sign will be considered as positive.

Exponent form of real constants consists of two parts: **(1) Mantissa** and **(2) Exponent.** The mantissa must be either an integer or a real constant. The mantissa followed by a letter E or e and the exponent, should also be an integer.

For example, 58000000.00 may be written as 0.58×10^8 or 0.58E8.

Mantissa (Before E)	Exponent (After E)
0.58	8

Example:

5.864 E1	$10^1 \times 5.864$ →	58.64 →	
5864 E-2	$10^{-2} \times 5864$ →	58.64 →	
0.5864 E2	$10^2 \times 0.5864$ →	58.64 →	

Boolean Literals

Boolean literals are used to represent one of the Boolean values (True or false). Internally true has value 1 and false has value 0.

Character constant

A character constant in C++ is any valid single character enclosed within single quotes.

Valid character constants : 'A', '2', '$'

Invalid character constants : "A"

The value of a single character constant has an equivalent ASCII value. For example, the value of 'A' is 65.

Escape sequences (or) Non-graphic characters

C++ allows certain non-printable characters represented as character constants. Non-printable characters are also called as non-graphic characters. Non-printable characters are those characters that cannot be typed directly from a keyboard during the execution of a program in C++, for example: backspace, tabs etc. These non-printable characters can be represented by using escape sequences. An escape sequence is represented by a backslash followed by one or two characters.

Table 9.2 Escape Sequences

Escape sequence	Non-graphical character
\a	Audible or alert bell
\b	Backspace
\f	Form feed
\n	Newline or linefeed
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\’	Single quote
\”	Double quote
\?	Question Mark
\On	Octal number
\xHn	Hexadecimal number
\0	Null

Even though an escape sequence contains two characters, they should be enclosed within single quotes because, C++ consider escape sequences as character constants and allocates one byte in ASCII representation.

DO YOU KNOW?

ASCII (American Standard Code for Information Interchange) was first developed and published in 1963 by the X3 committee, a part of the American Standards Association (ASA).

Evaluate Yourself

1. What is meant by literals? How many types of integer literals are available in C++?
2. What kind of constants are following?
   i) 26 ii) 015 iii) 0xF iv) 014.9
3. What is character constant in C++?
4. How are non graphic characters represented in C++?
5. Write the following real constants into exponent form:
   i) 32.179 ii) 8.124 iii) 0.00007
6. Write the following real constants in fractional form:
   i) 0.23E4 ii) 0.517E-3 iii) 0.5E-5
7. What is the significance of null (\0) character in a string?

String Literals

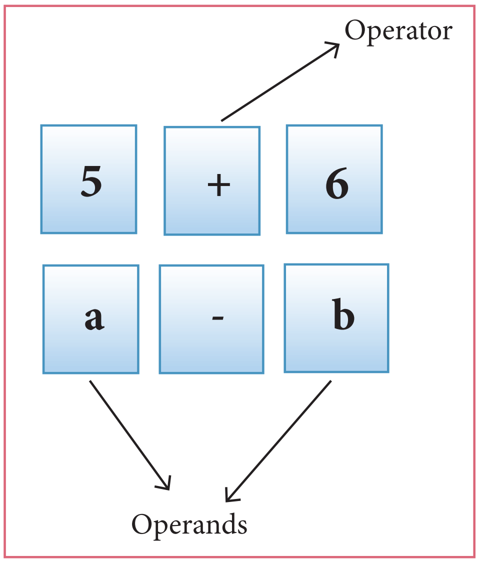
Sequence of characters enclosed within double quotes are called as String literals. By default, string literals are automatically added with a special character **‘\0’ (Null)** at the end. Therefore, the string “welcome” will actually be represented as “welcome\0” in memory and the size of this string is not 7 but 8 characters i.e., inclusive of the last character \0.

Valid string Literals : “A”, “Welcome” “1234”

Invalid String Literals : ‘Welcome’, ‘1234’

9.3.4 Operators

The symbols which are used to do some mathematical or logical operations are called as **“Operators”.** The data items or values that the oper ators act upon are called as **“Operands”.**



In C++, The operators are classified on the basis of the number of operands.

(i) **Unary Operators** - Require only one operand

(ii) **Binary Operators** - Require two operands

(iii) **Ternary Operators** - Require three operands

C++ Binary Operators are classified as:

(1) Arithmetic Operators
(2) Relational Operators
(3) Logical Operators
(4) Assignment Operators
(5) Conditional Operator

(1) Arithmetic Operators

Arithmetic operators perform simple arithmetic operations like addition, subtraction, multiplication, division etc.,

Operator	Operation	Example
+	Addition	10 + 5 = 15
-	Subtraction	10 – 5 = 5
*	Multiplication	10 * 5 = 50
/	Division	10 / 5 = 2 (Quotient of the division)
%	Modulus (To find the remainder of a division)	10 % 3 = 1(Remainder of the division)

- The above mentioned arithmetic operators are binary operators which requires minimum of two operands.

Increment and Decrement Operators

++ (Plus, Plus) Increment operator

-- (Minus, Minus) Decrement operator

An increment or decrement operator acts upon a single operand and returns a new value. Thus, these operators are unary operators. The increment operator adds 1 to its operand and the decrement operator subtracts 1 from its operand. For example,

- **x++ or ++ x** is the same as **x = x+1;**

  It adds 1 to the present value of x

- **x -- or -- x** is the same as to **x = x–1;**

  It subtracts 1 from the present value of x

The ++ or -- operators can be placed either as prefix (before) or as postfix (after) to a variable. With the prefix version, C++ performs the increment / decrement before using the operand.

(2) Relational Operators

Relational operators are used to determine the relationship between its operands. When the relational operators are applied on two operands, the result will be a Boolean value i.e 1 or 0 to represents True or False respectively. C++ provides six relational operators. They are,

Operator	Operation	Example
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b
==	Equal to	a == b
!=	Not equal	a != b

- In the above examples, the operand *a* is compared with *b* and depending on the relation, the result will be either 1 or 0. i.e., 1 for true, 0 for false.
- All six relational operators are binary operators.

(3)Logical Operators

A logical operator is used to evaluate logical and relational expressions. The logical operators act upon the operands that are themselves called as logical expressions. C++ provides three logical operators.

Table 9.3 Logical Operators

Operator	Operation	Description
&&	AND	The logical AND combines two different relational expressions in to one. It returns 1 (True), if both expression are true, otherwise it returns 0 (false).

||	**OR**	The logical OR combines two different relational expressions in to one. It returns 1 (True), if either one of the expression is true. It returns 0 (false), if both the expressions are false.
!	**NOT**	NOT works on a single expression / operand. It simply negates or inverts the truth value. i.e., if an operand / expression is 1 (true) then this operator returns 0 (false) and vice versa

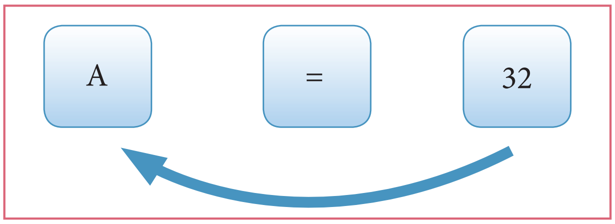
- AND, OR both are binary operators where as NOT is an unary operator.

Example: a = 5, b = 6, c = 7;

Expression	Result
(a<b) && (b<c)	1 (True)
(a>b) && (b<c)	0 (False)
(a<b) || (b>c)	1 (True)
!(a>b)	1 (True)

(4)Assignment Operator:

Assignment operator is used to assign a value to a variable which is on the left hand side of an assignment statement. = (equal to) is commonly used as the assignment operator in all computer programming languages. This operator copies the value at the right side of the operator to the left side variable. It is also a binary operator.



C++ uses different types of assignment operators. They are called as Shorthand assignment operators.

Operator	Name of Operator	Example

+=	Addition Assignment	a = 10;
		c = a += 5;
		(ie, a = a + 5)
		c = 15

-=	Subtraction Assignment	a = 10;
		c = a -= 5;
		(ie. a = a – 5)
		c = 5

*=	Multiplication Assignment	a = 10;
		c = a *= 5;
		(ie. a = a * 5)
		c = 50

/=	Division Assignment	a = 10;
		c = a /= 5;
		(ie. a = a / 5)
		c = 2

%=	Modulus Assignment	a = 10;
		c = a %= 5;
		(ie. a = a % 5)
		c = 0

Discuss the differences between = and == operators

(5) Conditional Operator:

In C++, there is only one conditional operator. **?:** is a conditional Operator which is also known as Ternary Operator. This operator is used as an alternate to if … else control statement. We will learn more about this operator in later chapters along with if …. else structure.

Other Operators:

The Comma operator	Comma (,) is an operator in C++ used to string together several expressions. The group of expression separated by comma is evaluated from left to right.
Sizeof	This is called as compile time operator. It returns the size of a variable in bytes.
Pointer	* Pointer to a variable & Address of

Component selection	. Direct component selector -> Indirect component selector
Class member operators	:: Scope access / resolution .* Dereference ->* Dereference pointer to class member

Precedence of Operators:

Operators are executed in the order of precedence. The operands and the operators are grouped in a specific logical way for evaluation. This logical grouping is called as an Association.

The order of precedence:

( ) [ ]	Operators within parenthesis are performed first	Higher
++, --	Postfix increment / decrement	↓
++, --	Prefix increment / decrement	
*, /, %	Multiplication, Division, Modulus	
+, -	Addition, Subtraction	
<, <=, >, >=	Less than, Less than or equal to, Greater than, Greater than or equal to	
==, !=	Equal to, Not equal to	
&&	Logical AND	
||	Logical OR	
?:	Conditional Operator	
=	Simple Assignment	
+=, -=, *=, /=	Shorthand operators	
,	Comma operator	Lower

9.3.5 Punctuators

Punctuators are symbols, which are used as delimiters, while constructing a C++ program. They are also called as **"Separators"**. The following punctuators are used in C++; most of these symbols are very similar to C and Java.

Separator	Description	Example

Curly braces { }	Opening and closing curly braces indicate the start and end of a block of code. A block of code containing more than one executable statement. These statements together are called as "compound statement"	int main ( )
		{
		int x=10, y=20, sum;
		sum = x + y;
		cout << sum;
		}

Parenthesis ( )	Opening and closing parenthesis indicate function calls and function parameters.	clrscr( ); int main ( )

Square brackets [ ]	It indicates single and multidimensional arrays.	int num[5];
		char name[50];

Comma ,	It is used as a separator in an expression	int x=10, y=20, sum;

Semicolon ;	Every executable statement in C++ should terminate with a semicolon	int main ( )
		{
		  int x=10, y=20, sum;
		  sum = x + y;
		  cout << sum;
		}

Colon :	It is used to label a statement.	private:

Comments
//
/*   */
	Any statement that begins with // are considered as comments. Comments are simply ignored by compilers. i.e., compiler does not execute any statement that begins with a //
	// Single line comment
	/* **……….. */ Multiline comment**
		/* **This is written by me to learn CPP** */
		int main ( )
		{
		    int x=10, y=20, sum;
		// **to sum x and y**
		  sum = x + y;
		  cout << sum;
		}

DO YOU KNOW? In C++, one or two operators may be used in different places with different meaning.

For example: Asterisk (*) is used for multiplication as well as for pointer to a variable.

Evaluate Yourself

1. What is the use of operators?
2. What are binary operators? Give examples of arithmetic binary operators.
3. What does the modulus operator % do?
4. What will be the result of 8.5 % 2?
5. Give that i = 8, j = 10, k = 8, What will be result of the following expressions?
   (i) i < k   (ii) i < j   (iii) i > = k   (iv) i = = j   (v) j ! = k
6. What will be the order of evaluation for the following expressions?
   (i) i + 3 >= j - 9   (ii) a +10 < p - 3 + 2 q
7. Write an expression involving a logical operator to test, if marks are 75 and grade is 'A'.

9.4 I/O Operators

9.4.1 Input operator

C++ provides the operator >> to get input. It extracts the value through the keyboard and assigns it to the variable on its right; hence, it is called as **"Stream extraction"** or **"get from"** operator.

It is a binary operator i.e., it requires two operands. The first operand is the pre-defined identifier cin (pronounced as C-In) that identifies keyboard as the input device. The second operand must be a variable.

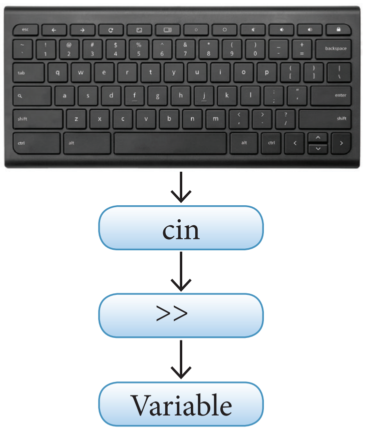


Figure 9.4 Working process of cin

To receive or extract more than one value at a time, >> operator should be used for each variable. This is called cascading of operator.

Example:

cin >> num;	Pre-defined object cin extracts a value typed on keyboard and stores it in variable **num.**

cin >>x >> y;	This is used to extract two values. cin reads the first value and immediately assigns that to variable x; next, it reads the second value which is typed after a space and assigns that to y. Space is used as a separator for each input.

9.4.2 Output Operator

C++ provides << operator to perform output operation. The operator << is called the **"Stream insertion"** or **"put to"** operator. It is used to send the strings or values of the variables on its right to the object on its left. << is a binary operator.

The first operand is the pre-defined identifier cout (pronounced as C-Out) that identifies monitor as the standard output object. The second operand may be a constant, variable or an expression.

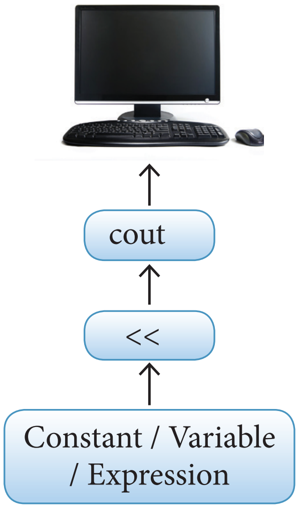


Figure 9.5 Working process of cout

To send more than one value at a time, << operator should be used for each constant/variable/expression. This is called **cascading of operator**.

Example:

cout << "Welcome";	Pre-defined object cout sends the given string "Welcome" to screen.
cout << "The sum = " << sum;	First, cout sends the string "The Sum = " to the screen and then sends the value of the variable sum; Usually, cout sends everything specified within double quotes or single quotes i.e., string or character constants, except non-graphic characters.
cout <<"\n The Area: " <<3.14*r*r;	First, cout sends everything specified within double quotes except \n to the screen, and then it evaluates the expression 3.14*r*r and sends the result to the screen. \n – is a non graphic character constant to feed a new line.
cout << a + b ;	cout sends the sum of a and b to the output console (monitor)

9.4.3. Cascading of I/O operators

The multiple use of input and output operators such as >> and << in a single statement is known as cascading of I/O operators.

Cascading cout:

```
int Num=20;

cout << "A=" << Num;
```

The Figure 9.6 is used to understand the working of Cascading cout statement

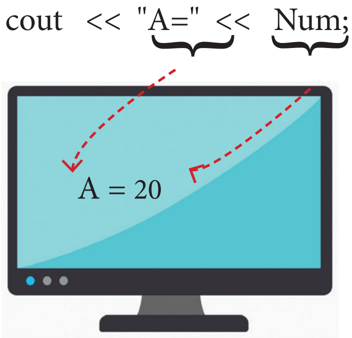


Figure 9.6 Cascading cout

Cascading cin - Example:

cout >> "Enter two number: ";

cin >> a >> b;

The Figure 9.7 is used to understand the working of Cascading cin statement

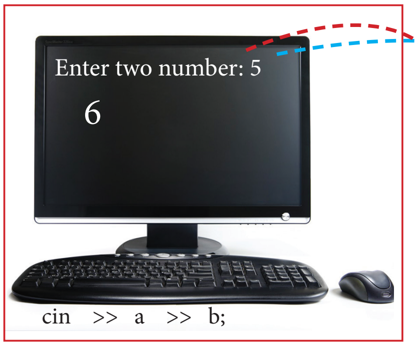
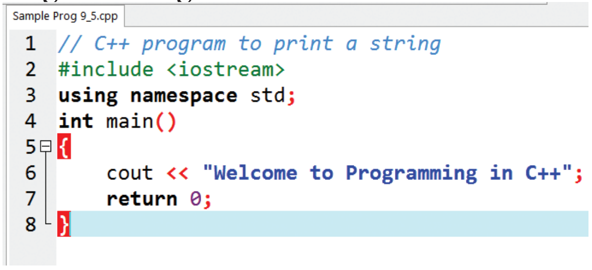


Figure 9.7 Cascading cin

9.5 Sample program – A first look at C++ program

Let us start our first C++ program that prints a string "Welcome to Programming in C++" on the screen.



```
1  // C++ program to print a string
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      cout << "Welcome to Programming in C++";
7      return 0;
8  }
```

The above program produces, the following output:

```
Welcome to Programming in C++
```

This is very simple C++ program which includes the basic elements that every C++ program has. Let us have a look at these elements:

1 // C++ program to print a string

This is a comment statement. Any statement that begins with // are considered as comments. Compiler does not execute any comment as part of the program and it simply ignores. If we need to write multiple lines of comments, we can use /* ……. */.

2 # include <iostream>

Usually all C++ programs begin with include statements starting with a # (hash / pound). The symbol # is a directive for the preprocessor. That means, these statements are processed before the compilation process begins.

#include <iostream> statement tells the compiler's preprocessor to include the header file "iostream" in the program.

The header file iostream should included in every C++ program to implement input / output functionalities.

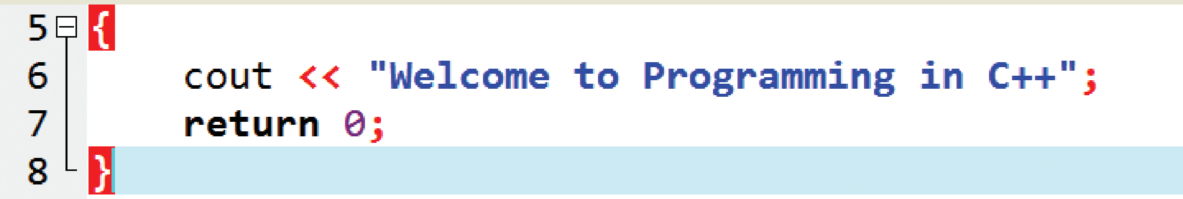
In simple words, **iostream header file contains the definition of its member objects cin and cout.** If you fail to include iostream in your program, an error message will occur on cin and cout; and we will not be able to get any input or send any output.

3 using namespace std;

The line using namespace std; tells the compiler to use standard namespace. Namespace collects identifiers used for class, object and variables. Namespaces provide a method of preventing name conflicts in large projects. It is a new concept introduced by the ANSI C++ standards committee.

4 int main ( )

C++ program is a collection of functions. Every C++ program must have a main function. The main( ) function is the starting point where all C++ programs begin their execution. Therefore, the executable statements should be inside the main( ) function.



```
5 {
6       cout << "Welcome to Programming in C++";
7       return 0;
8 }
```

The statements between the curly braces (Line number 5 to 8) are executable statements. This is actually called as a block of code. In line 6, cout simply sends the string constant "Welcome to Programming in C++" to the screen. As we discussed already, every executable statement must terminate with a semicolon. In line 7, return is a keyword which is used to return the value what you specified to a function. In this case, it will return 0 to main( ) function.

9.6 Execution of C++ program:

For creating and executing a C++ program, one must follow four important steps.

(1) Creating Source code

Creating includes typing and editing the valid C++ code as per the rules followed by the C++ Compiler.

(2) Saving source code with extension .cpp

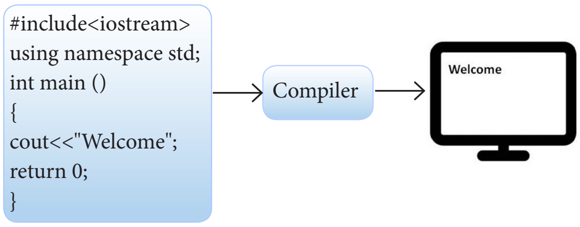
After typing, the source code should be saved with the extension .cpp

(3) Compilation

This is an important step in constructing a program. In compilation, compiler links the library files with the source code and verifies each and every line of code. If any mistake or error is found, it will throw error message. If there are no errors, it translates the source code into machine readable object file with an extension .obj

(4) Execution

This is the final step of a C++ Program. In this stage, the object file becomes an executable file with extension .exe. Once the program becomes an executable file, the program has an independent existence. This means, you can run your application without the help of any compiler or IDE.



```
#include<iostream>
using namespace std;
int main ()
{
cout<<"Welcome";
return 0;
}
```

Figure 9.8 Execution

9.7 C++ Development Environment

There are lot of IDE programs available for C++. IDE makes it easy to create, compile and execute a C++ program. Most of the IDEs are open source applications (ie.) they are available free of cost.

9.7.1 Familiar C++ Compilers with IDE

Table 9.4 Open Source Compilers

Compiler	Availability
Dev C++	Open source
Geany	Open source
Code::blocks	Open source
Code Lite	Open source

Net Beans	Open source
Digital Mars	Open source
Sky IDE	Open source
Eclipse	Open source

9.7.2 Working with Dev C++

Among the dozens of IDEs, we take "Dev C++" compiler to create C++ programs. Programming techniques and illustrated programs of this book are based on "Dev C++" compiler.

Dev C++ is an open source, cross platform (alpha version available for Linux), full featured Integrated Development Environment (IDE) distributed with the GNU General Public License for programming in C and C++. It is written in Delphi. It can be downloaded from ***http://www.bloodshed.net/dev/devcpp.html***

1. After installation **Dev C++** icon is available on the desktop. Double click to open IDE. Dev C++ IDE appears as given below.

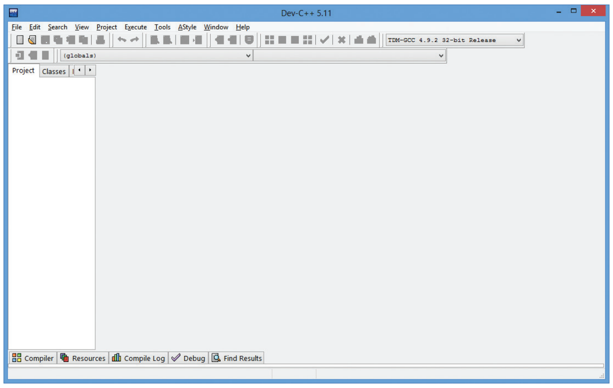
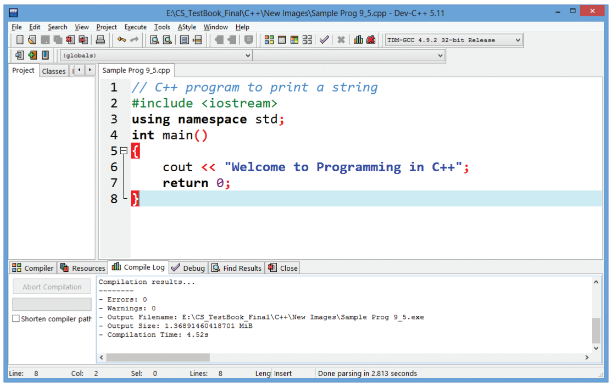


Figure 9.9 Dev C++ opening Window

2. To create a source file, Select **File → New → Source file** or **Press Ctrl + N.**

3. In the screen that appears, type your C++ program, and save the file by clicking **File → Save** or Pressing **Ctrl + S.** It will add .cpp by default at the end of your source code file. No need to type .cpp along with your file name.



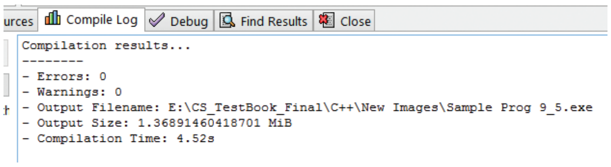
```
// C++ program to print a string
#include <iostream>
using namespace std;
int main()
{
    cout << "Welcome to Programming in C++";
    return 0;
}
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: E:\CS_TestBook_Final\C++\New Images\Sample Prog 9_5.exe
- Output Size: 1.36891460418701 MiB
- Compilation Time: 4.52s
```

Figure 9.10 Dev C++ IDE with a program

4. **After save, Click Execute → Compile and Run or press F11 key.**

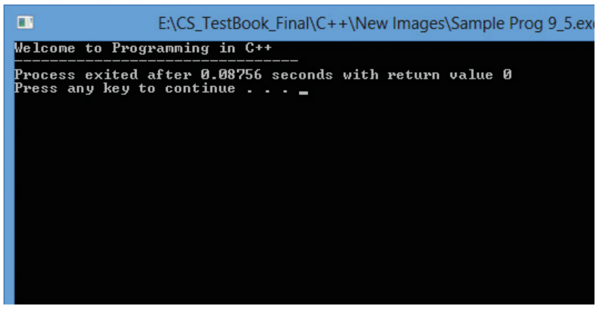
   If your program contains any error, it displays the errors under **compile log**. If your program is without any error, the display will appear as follows.



```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: E:\CS_TestBook_Final\C++\New Images\Sample Prog 9_5.exe
- Output Size: 1.36891460418701 MiB
- Compilation Time: 4.52s
```

Figure 9.11 Dev C++ Compile Log

5. **After successful compilation, output will appear in output console, as follows**



```
Welcome to Programming in C++
--------------------------------
Process exited after 0.08756 seconds with return value 0
Press any key to continue . . . _
```

Figure 9.12 Dev C++ Output Window

9.8 Types of Errors

Some common types of errors are given below:

Type of Error	Description
Syntax Error	• Syntax is a set of grammatical rules to construct a program. Every programming language has unique rules for constructing the sourcecode. • Syntax errors occur when grammatical rules of C++ are violated. • Example: if you type as follows, C++ will throw an error. **cout << "Welcome to Programming in C++"** • As per grammatical rules of C++, every executable statement should terminate with a semicolon. But, this statement does not end with a semicolon.
Semantic Error	• A Program has not produced expected result even though the program is grammatically correct.It may be happened by wrong use of variable / operator / order of execution etc. This means, program is grammatically correct, but it contains some logical error. So, Semantic error is also called as **"Logic Error"**.
Run-time error	• A run time error occurs during the execution of a program. It occurs because of some illegal operation that takes place. • For example, if a program tries to open a file which does not exist, it results in a run-time error

Points to Remember:

- C++ was developed by Bjarne Stroustrup at AT & T Bell Labs during the year 1979.
- Character set is the set of characters which are allowed to write C++ programs.
- Individual elements are collectively called as Lexical units or Lexical elements or Tokens.
- Keywords are the reserved words that convey specific meaning to the C++ compiler.
- Identifiers are user-defined names given to different parts of the C++ program viz. variables, functions, arrays, classes etc.,
- Literals are data items whose values do not change during the execution of a program. Therefore, Literals are called as Constants.
- There are different kinds of literals used in C++ (Integer, Float, Character, String)
- The symbols which are used to do some mathematical, logical operations are called as "Operators".
- Punctuators are symbols, which are used as delimiters in constructing C++ programs. They are also called as "Separators".
- Extraction operator(>>) and Insertion operator (<<) are used to get input and send output in C++.

Hands on practice:

- Type the following C++ Programs in Dev C++ IDE and execute. if compiler shows any errors, try to rectify it and execute again and again till you get the expected result.

1. C++ Program to find the total marks of three subjects

```
#include <iostream>
using namespace std;
int main()
{
        int m1, m2, m3, sum;
        cout << "\n Enter Mark 1: ";
        cin >> m1;
        cout << "\n Enter Mark 2: ";
        cin >> m2;
        cout << "\n Enter Mark 3: ";
        cin >> m3;
        sum = m1 + m2 + m3;
        cout << "\n The sum = " << sum;
}
```

- Make changes in the above code to get the average of all the given marks.

2. C++ program to find the area of a circle

```
#include <iostream>
using namespace std;
int main()
{
        int radius;
        float area;
        cout << "\n Enter Radius: ";
        cin >> radius;
        area = 3.14 * radius * radius;
        cout << "\n The area of circle = " << area;
}
```

3. point out the errors in the following program:

```
Using namespace std;
int main( )
{
cout << “Enter a value ”;
cin << num1 >> num2
num+num2=sum;
cout >> “\n The Sum= ” >> sum;
```

4. point out the type of error in the following program:

```
#include <iostream>
using namespace std;
int main()
{
        int h=10; w=12;
        cout << "Area of rectangle " << h+w;
}
```

Evaluation

SECTION – A

Choose the correct answer:

1. Who developed C++?

   (a) Charles Babbage　　(b) Bjarne Stroustrup

   (c) Bill Gates　　(d) Sundar Pichai

2. What was the original name given to C++?

   (a) CPP　　(b) Advanced C　　(c) C with Classes　　(d) Class with C

3. Who coined C++?

   (a) Rick Mascitti　　(b) Rick Bjarne　　(c) Bill Gates　　(d) Dennis Ritchie

4. The smallest individual unit in a program is:

   (a) Program　　(b) Algorithm　　(c) Flowchart　　(d) Tokens

5. Which of the following operator is extraction operator in C++?

   (a) >>　　(b) <<　　(c) <>　　(d) ^^

6. Which of the following statements is not true?

   (a) Keywords are the reserved words which convey specific meaning to the C++ compiler.

   (b) Reserved words or keywords can be used as an identifier name.

   (c) An integer constant must have at least one digit without a decimal point.

   (d) Exponent form of real constants consist of two parts

7. Which of the following is a valid string literal?

   (a) 'A'　　(b) 'Welcome'　　(c) 1232　　(d) "1232"

8. A program written in high level language is called as

   (a) Object code　　(b) Source code　　(c) Executable code　　(d) All the above

9. Assume a=5, b=6; what will be result of a&b?

   (a) 4　　(b) 5　　(c) 1　　(d) 0

10. Which of the following is called as compile time operators?

    (a) sizeof　　(b) pointer　　(c) virtual　　(d) this

SECTION-B

Very Short Answers

1. What is meant by a token? Name the token available in C++.
2. What are keywords? Can keywords be used as identifiers?
3. The following constants are of which type?
   (i) 39 (ii) 032 (iii) 0XCAFE (iv) 04.14
4. Write the following real constants into the exponent form:
   (i) 23.197 (ii) 7.214 (iii) 0.00005 (iv) 0.319
5. Assume n=10; what will be result of n++ and --n;?
6. Match the following

A	B
(a) Modulus	(1) Tokens
(b) Separators	(2) Remainder of a division
(c) Stream extraction	(3) Punctuators
(d) Lexical Units	(4) get from

SECTION-C

Short Answers

1. Describe the differences between keywords and identifiers?
2. Is C++ case sensitive? What is meant by the term "case sensitive"?
3. Differentiate "=" and "==".
4. What is the use of a header file?
5. Why is main function special?

SECTION - D

Explain in detail

1. Write about Binary operators used in C++.
2. What are the types of Errors?

References:

(1) Object Oriented Programming with C++ (4th Edition), Dr. E. Balagurusamy, Mc.Graw Hills.

(2) The Complete Reference C++ (Forth Edition), Herbert Schildt.Mc.Graw Hills.

(3) Computer Science with C++ (A text book of CBSE XI and XII), Sumita Arora, Dhanpat Rai & Co.

9.10 Introduction

Every programming language has two fundamental elements, viz., data types and variables. They are very essential elements to write even the most elementary programs. C++ provides a predefined set of data types for handling the data items. Such data types are known as fundamental or built-in data types. Apart from the built-in data types, a programmer can also create his own data types called as User-defined data types. In this chapter, we are going to learn about built-in data types.

9.11 Concept of Data types

Let us look at the following example,

Name = Ram

Age = 15

Average_Mark = 85.6

In the above example, Name, Age, Average_mark are the fields which hold the values such as Ram, 15, and 85.6 respectively.

In a programming language, **fields** are referred as **variables** and the **values** are referred to as **data**. Each data item in the above example looks different. That is, "Ram" is a sequence of alphabets and the other two data items are numbers. The first value is a whole number and the second one is a fractional number. In real-world scenarios, there are lots of different kinds of data we handle in our day-to-day life. The nature or type of the data item varies, for example distance (from your home to school), ticket fare, cost of a pen, marks, temperature, etc.,

In C++ programming, before handling any data, it should be clearly specified to the language compiler, regarding what kind of data it is, with some predefined set of data types.

9.12 C++ Data types

In C++, the data types are classified as three main categories

(1) Fundamental data types

(2) User-defined data types and

(3) Derived data types.

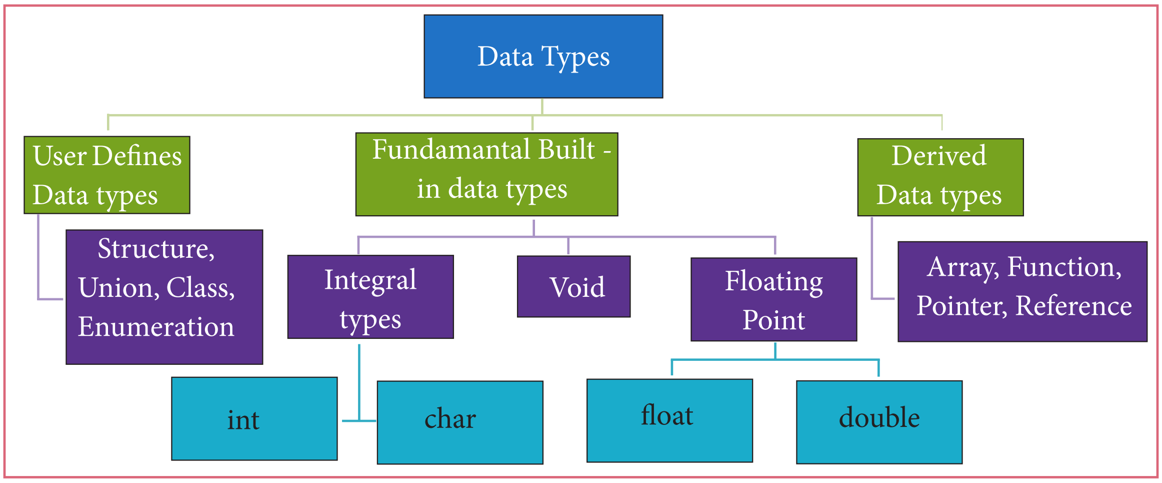

Data Types
User Defines Data types
Fundamantal Built - in data types
Derived Data types
Structure, Union, Class, Enumeration
Integral types
Void
Floating Point
Array, Function, Pointer, Reference
int
char
float
double

Figure 9.13 Data types in C++

In this chapter, we are going to learn about only the Fundamental data types.

In order to understand the working of data types, we need to know about variables. The variables are the named memory locations to hold values of specific data types. In C++, the variables should be declared explicitly with their data types before they are actually used.

Syntax for declaring a variable:

<data type> <variable name>;

Example:

int num1;

To declare more than one variable which are of the same data type using a single statement, it can be declared by separating the variables using a comma.

Example:

int num1, num2, sum;

For example, to store your computer science marks first you should declare a variable to hold your marks with a suitable data type. Choosing an appropriate data type needs more knowledge and experience. Usually, marks are represented as whole numbers. Thus, the variable for storing the computer science marks should be of integer data type.

Example:

int comp_science_marks;

Now, one variable named **comp_science_marks** is ready to store your marks.

We will learn more about variables later in this chapter.

9.12.1 Introduction to fundamental Data types:

Fundamental (atomic) data types are predefined data types available with C++. There are five fundamental data types in C++: **char, int, float, double** and **void**. Actually, these are the keywords for defining the data types.

(1) int data type:

Integer data type accepts and returns only integer numbers. If a variable is declared as an **int**, C++ compiler allows storing only integer values into it. If you try to store a fractional value in an int type variable it will accept only the integer portion and the fractional part will be ignored.

For Example
int num=12;

num1 variable is declared as integer types. So, it can store integer value

(2) char data type:

Character data type accepts and returns all valid ASCII characters. Character data type is often said to be an integer type, since all the characters are represented in memory by their associated **ASCII Codes.** If a variable is declared as char, C++ allows storing either a character or an integer value.

Example 1:-

```
char c='A';

cout<<ch ;
```

In the above code, **ch** is declared as a char type variable to hold a character. It displays the character A

Example 2:-

```
char ch='A'

cout<<ch+1;
```

In the above statements, the value of **ch** is incremented by 1 and the new value is stored back in the same variable **ch**. (Remember that, arithmetic operations are carried out only on the numbers not with alphabets) so it displays B

Another program illustrates how **int** and **char** data types are working together.

Illustration 9.3: C++ program to get an ASCII value and display the corresponding character

```
#include <iostream>
using namespace std;
int main ()
{
        int n;
        char ch;
        cout << "\n Enter an ASCII code (0 to 255): ";
        cin >> n;
        ch = n;
        cout << "\n Equivalent Character: " << ch;
}
```

The output

```
Enter an ASCII code (0 to 255): 100
Equivalent Character: d
```

In the above program, variable **n** is declared as an int type and another variable **ch** as a char type. During execution, the program prompts the user to enter an ASCII value. If the user enters an ASCII value as an integer, it will be stored in the variable **n**. In the statement **ch = n;** the value of **n** is assigned into **ch**. Remember that, **ch** is a char type variable.

For example, if a user enters 100 as input; initially, 100 is stored in the variable **n**. In the next statement, the value of **n** i.e., 100 is assigned to **ch**. Since, **ch** is a char type; it shows the corresponding ASCII character as output. (Equivalent ASCII Character for 100 is d).

(3) float data type:

If a variable is declared as float, all values will be stored as floating point values.

There are two advantages of using float data types.

(1) They can represent values between the integers.

(2) They can represent a much greater range of values.

At the same time, floating point operations takes more time to execute compared to the integer type ie., floating point operations are slower than integer operations. This is a disadvantage of floating point operation.

For Example

float num=13.4;

In the above example, num variable is declared as float type .

(4) double data type:

This is for double precision floating point numbers. (precision means significant numbers after decimal point). The double is also used for handling floating point numbers. But, this type occupies double the space than float type. This means, more fractions can be accommodated in double than in float type. The double is larger and slower than type float. double is used in a similar way as that of float data type.

(5) void data type:

The literal meaning for void is 'empty space'. Here, in C++, the void data type specifies an empty set of values. It is used as a return type for functions that do not return any value.

Evaluate Yourself

1. What do you mean by fundemantal data types?
2. The data type char is used to represent characters. then why is it often termed as an integer type?
3. What is the advantage of floating point numbers over integers?
4. The data type double is another floating point type. Why is it treated as a distinct data type?
5. What is the use of void data type?

9.12.2 Memory representation of Fundamental Data types:

One of the most important reason for declaring a variable as a particular data type is to allocate appropriate space in memory. As per the stored program concept, every data should be accommodated in the main memory before they are processed. So, C++ compiler allocates specific memory space for each and every data handled according to the compiler's standards.

The following Table 9.5 shows how much of memory space is allocated for each fundamental data type. Remember that, every data is stored inside the computer memory in the form of binary digits (See Unit I Chapter 2).

Table 9. 5 Memory allocation for Fundamental data types

Data type	Space in memory		Range of value
	in terms of bytes	in terms of bits	
char	1 byte	8 bits	-128 to 127
int	2 bytes	16 bits	-32,768 to 32,767
float	4 bytes	32 bits	3.4×10^{-38} to 3.4×10^{38} -1
double	8 bytes	64 bits	1.7×10^{-308} to 1.7×10^{308}-1

9.12.3 Data type modifiers:

Modifiers are used to modify the storing capacity of a fundamental data type except void type. Usually, every fundamental data type has a fixed range of values to store data items in memory. For example, int data type can store only two bytes of data. In reality, some integer data may have more length and may need more space in memory. In this situation, we should modify the memory space to accommodate large integer values. **Modifiers can be used to modify (expand or reduce) the memory allocation of any fundamental data type. They are also called as Qualifiers.**

There are four modifiers used in C++. They are:

(1) signed (2) unsigned (3) long (4) short

These four modifiers can be used with any fundamental data type. The following Table 9.6 shows the memory allocation for each data type with and without modifiers.

Integer type

Table 9.6 Memory allocation for Data types

Data type		Space in memory		Range of value
		in terms of bytes	in terms of bits	
short	short is a short name for **short int**	2 bytes	16 bits	-32,768 to 32,767
unsigned short	an integer number without minus sign.	2 bytes	16 bits	0 to 65535
signed short	An integer number with minus sign	2 bytes	32 bits	-32,768 to 32,767
Both short and signed short are similar				
int	An integer may or may not be signed	2 bytes	16 bits	-32,768 to 32,767

unsigned int	An integer without any sign (minus symbol)	2 bytes	16 bits	0 to 65535
signed int	An integer with sign	2 bytes	16 bits	-32,768 to 32,767
Both short and int are similar				
long	long is short name for long int	4 bytes	32 bits	-2147483648 to 2147483647
unsigned long	A double spaced integer without any sign	4 bytes	32 bits	0 to 4,294,967,295
signed long	A double spaced integer with sign	4 bytes	32 bits	-2147483648 to 2147483647

The above table clearly shows that an integer type accepts only 2 bytes of data whereas a long int accepts data that is double this size i.e., 4 bytes of data. So, we can store more digits in a long int. (long is a modifier and int is a fundamental data type)

char type

Table 9.7 Memory allocation for char Data types

Data type		**Space in memory**		**Range of value**
		in terms of bytes	**in terms of bits**	
char	Signed ASCII character	1 byte	8 bits	-128 to 127
unsigned char	ASCII character without sign	1 byte	8 bits	0 to 255
signed char	ASCII character with sign	1 byte	8 bits	-128 to 127

Floating point type

Table 9.8 Memory allocation for floating point Data types

Data type		**Space in memory**		**Range of value**
		in terms of bytes	**in terms of bits**	
float	signed fractional value	4 bytes	32 bits	3.4×10^{-38} to $3.4\times10^{38}-1$
double	signed more precision fractional value	8 bytes	64 bits	1.7×10^{-308} to $1.7\times10^{308}-1$
long double	signed more precision fractional value	10 bytes	80 bits	3.4×10^{-4932} to $1.1\times10^{4932}-1$

Memory allocation is subjected to vary based on the type of compiler that is being used. Here, the given values are as per the **Turbo C++** compiler. **Dev C++** provides some more space to **int** and **long** double types. Following Tables 9.9 shows the difference between Turbo C++ and Dev C++ allocation of memory.

Table 9.9 Memory allocation of Turbo C++ and Dev C++

Data type	Memory size in bytes	
	Turbo C++	**Dev C++**
short	2	2
unsigned short	2	2
signed short	2	2
int	2	**4**
unsigned int	2	**4**
signed int	2	**4**
long	4	4
unsigned long	4	4
signed long	4	4
char	1	1
unsigned char	1	1
signed char	1	1
float	4	4
double	8	8
long double	10	**12**

Since, Dev C++ provides 4 bytes to int and long, any one of these types can be used to handle bigger integer values while writing programs in Dev C++.

Note: sizeof( ) is an operator which gives the size of a data type.

Number Suffixes in C++

There are different suffixes for integer and floating point numbers. Suffix can be used to assign the same value as a different type. For example, if you want to store 45 in int, long, unsigned int and unsigned long int, you can use suffix letter **L** or **U** (either case) with 45 i.e. **45L** or **45U**. This type of declaration instructs the compiler to store the given values as long and unsigned. 'F' can be used for floating point values, example: 3.14F

9.13 Variables

Variables are user-defined names assigned to specific memory locations in which the values are stored. Variables are also identifiers; and hence, the rules for naming the identifiers should be followed while naming a variable. These are called as symbolic variables because these are named locations.

There are two values associated with a symbolic variable; they are **R**-value and **L**-value.

- R-value is data stored in a memory location
- L-value is the memory address in which the R-value is stored.

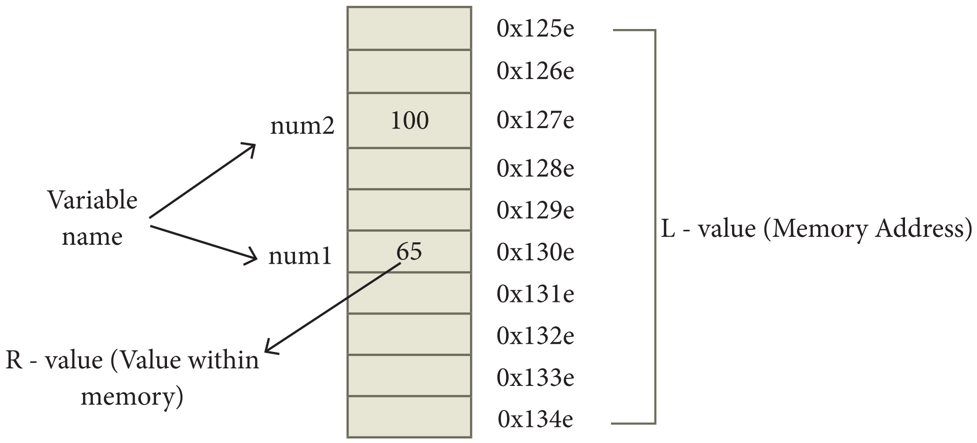


Figure 9.14 Memory allocation of a variable

Remember that, the memory addresses are in the form of Hexadecimal values

9.13.1 Declaration of Variables

Every variable should be declared before they are actually used in a program. Declaration is a process to instruct the compiler to allocate memory as per the type that is specified along with the variable name. For example, if you declare a variable as int type, in Dev C++, the compiler allocates 4 bytes of memory. Thus, every variable should be declared along with the type of value to be stored.

Declaration of more than one variable:

More than one variable of the same type can be declared as a single statement using a comma separating the individual variables.

Syntax:

<data type> <var1>, <var2>, <var3> …… <var_n>;

Example:

int num1, num2, sum;

In the above statement, there are three variables declared as int type. Which means, in **num1, num2** and **sum**, you can store only integer values.

For the above declaration, the C++ compiler allocates 4 bytes of memory (i.e. 4 memory boxes) for each variable.

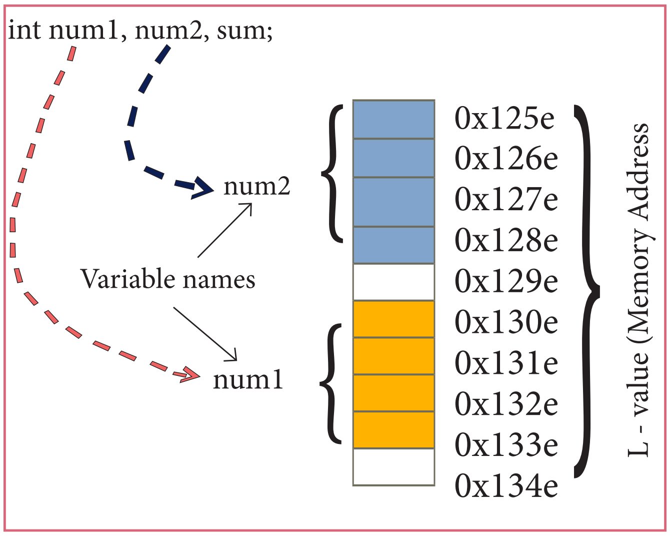


Figure 9.15 Memory allocation of int type variables

If you declare a variable without any initial value, the memory space allocated to that variable will be occupied with some unknown value. These unknown values are called as **"Junk"** or **"Garbage"** values.

```
#include <iostream>
using namespace std;
int main()
{
        int num1, num2, sum;
        cout << num1 << endl;
        cout << num2 << endl;
        cout << num1 + num2;
}
```

In the above program, some unknown values will be occupied in memory that is allocated for the variables **num1** and **num2** and the statement c**out << num1 + num2;** will display the sum of those unknown junk values.

9.13.2 Initialization of variables

Assigning an initial value to a variable during its declaration is called as "Initialization".

Examples:

int num = 100;

float pi = 3.14;

double price = 231.45;

Here, the variables num, pi, and price have been initialized during the declaration. These initial values can be later changed during the program execution.

Illustration 9.6 C++ Program to find the Curved Surface Area of a cylinder (CSA) (CSA = 2 pi r * h)

```
#include <iostream>
using namespace std;
int main()
{
  float pi = 3.14, radius, height, CSA;
  cout << "\n Curved Surface Area of a cylinder";
  cout << "\n Enter Radius (in cm): ";
  cin >> radius;
  cout << "\n Enter Height (in cm): ";
  cin >> height;
  CSA = (2*pi*radius)*height;
  system("cls");
  cout << "\n Radius: " << radius <<"cm";
  cout << "\n Height: " << height << "cm";
  cout << "\n Curved Surface Area of a Cylinder is " << CSA <<" sq. cm.";
}
```

Output:

```
 Curved Surface Area of a cylinder
 Enter Radius (in cm): 7
 Enter Height (in cm): 20
 Radius: 7cm
Height: 20cm
Curved Surface Area of a Cylinder is 879.2 sq. cm.
```

Variables that are of the same type can be initialized in a single statement.

Example:

int x1 = -1, x2 = 1, x3, n;

9.13.3 Dynamic Initialization

A variable can be initialized during the execution of a program. It is known as "**Dynamic initialization**". For example,

int num1, num2, sum;

sum = num1 + num2;

The above two statements can be combined into a single one as follows:

int sum = num1+num2;

This initializes sum using the known values of num1 and num2 during the execution.

Illustration 9.7 C++ Program to illustrate dynamic initialization

```
#include <iostream>
using namespace std;
int main()
{
        int num1, num2;
        cout << "\n Enter number 1: ";
        cin >> num1;
        cout << "\n Enter number 2: ";
        cin >> num2;
        int sum = num1 + num2; // Dynamic initialization
        cout << "\n Average: " << sum /2;
}
```

Output:

```
Enter number 1: 78
Enter number 2: 65
Average: 71
```

In the above program, after getting the values of num1 and num2, sum is declared and initialized with the addition of those two variables. After that, it is divided by 2.

Illustration 9.8: C++ program to find the perimeter and area of a semi circle

```
#include <iostream>
using namespace std;
int main()
{
        int radius;
        float pi = 3.14;
        cout << "\n Enter Radius (in cm): ";
        cin >> radius;
        float perimeter = (pi+2)*radius;   // dynamic initialization
        float area = (pi*radius*radius)/2; // dynamic initialization
        cout << "\n Perimeter of the semicircle is " << perimeter << " cm";
        cout << "\n Area of the semicircle is " << area << " sq.cm";
}
```

Output:

```
Enter Radius (in cm): 14
Perimeter of the semicircle is 71.96 cm
Area of the semicircle is 307.72 sq.cm
```

9.13.4 The Access modifier const

const is the keyword used to declare a constant. You already learnt about constant in the previous chapter. const keyword modifies / restricts the accessibility of a variable. So, it is known as Access modifier.

For example,

int num = 100;

The above statement declares a variable num with an initial value 100. However, the value of num can be changed during the execution. If you modify the above definition as **const int num = 100;** the variable num becomes a constant and its value will remain 100 throughout the program, and it can never be changed during the execution.

```
#include <iostream>
using namespace std;
int main()
{
        const int num=100;
        cout << "\n Value of num is = " << num;
        num = num + 1; // Trying to increment the constant
        cout << "\n Value of num after increment " << num;
}
```

In the above code, an error message will be displayed as **"Cannot modify the const object"** in Turbo compiler and **"assignment of read only memory num"** in Dev C++.

9.13.5 References

A reference provides an alias for a previously defined variable. Declaration of a reference consists of base type and an & (ampersand) symbol; reference variable name is assigned the value of a previously declared variable.

Syntax:

<type> <& reference_variable> = <original_variable>;

Illustration 9.9: C++ program to declare reference variable

```
#include <iostream>
using namespace std;
int main()
{
        int num;
        int &temp = num; //declaration of a reference variable temp
        num = 100;
cout << "\n The value of num = " << num;
cout << "\n The value of temp = " << temp;
}
```

The output of the above program will be

```
The value of num = 100
The value of temp = 100
```

Evaluate Yourself

1. What are modifiers? What is the use of modifiers?
2. What is wrong with the following C++ statement:

   long float x;
3. What is a variable ? Why is a variable called symblolic variable?
4. What do you mean by dynamic initialization of a variable? Give an exmple.
5. What is wrong with the following statement?

   const int x;

9.14 Formatting Output

Formatting output is very important in the development of output screens for easy reading and understanding. Manipulators are used to format the output of any C++ program. Manipulators are functions specifically designed to use with the insertion (<<) and extraction(>>) operators.

C++ offers several input and output manipulators for formatting. Commonly used manipulators are: **endl, setw, setfill, setprecision and setf**. In order to use these manipulators, you should include the appropriate header file. **endl** manipulator is a member of iostream header file. **setw, setfill, setprecision and setf** manipulators are members of iomanip header file.

endl (End the Line)

endl is used as a line feeder in C++. It can be used as an alternate to '\n'. In other words, endl inserts a new line and then makes the cursor to point to the beginning of the next line. There is a difference between endl and '\n', even though they are performing similar tasks.

- endl – Inserts a new line and flushes the buffer (Flush means – clean)
- '\n' - Inserts only a new line.

Example:

```
cout << "\n The value of num = " << num;
cout << "The value of num = " << num <<endl;
```

Both these statements display the same output.

setw ( )

setw manipulator sets the **width of the field** assigned for the output. The field width determines the minimum number of characters to be written in output.

Syntax:

setw(number of characters)

Illustration 9.10: Program to Calculate Net Salary

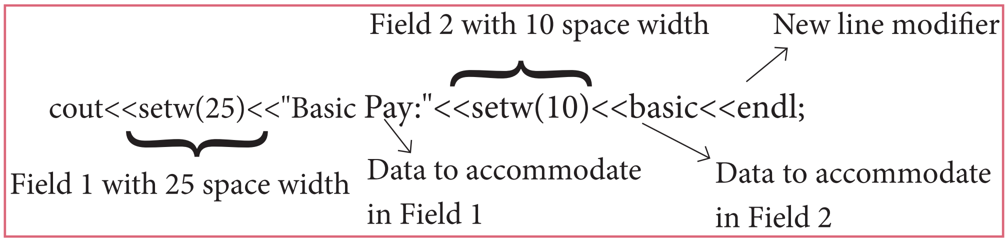
```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        float basic, da, hra, gpf, tax, gross, np;
        char name[30];
        cout << "\n Enter Basic Pay: ";
        cin >> basic;
        cout << "\n Enter D.A : ";
        cin >> da;
        cout << "\n Enter H.R.A: ";
        cin >> hra;
        gross = basic+da+hra; // sum of basic, da and hra
        gpf = (basic+da) * 0.10; // 10% 0f basic and da
        tax = gross * 0.10; //10% of gross pay
        np = gross - (gpf+tax); //netpay = earnings - deductions
        cout << setw(25) << "Basic Pay : " << setw(10)<< basic<< endl;
        cout << setw(25) << "Dearness Allowance : "<< setw(10)<<da<< endl;
        cout<<setw(25)<<"House Rent Allowance : "<<setw(10)<< hra<<endl;
        cout << setw(25) << "Gross Pay : " << setw(10) << gross << endl;
        cout << setw(25) << "G.P.F : " << setw(10) << gpf << endl;
        cout << setw(25) << "Income Tax : " << setw(10)<< tax << endl;
        cout << setw(25) << "Net Pay : " << setw(10) << np << endl;
}
```

The output will be,

```
Enter Basic Pay: 12000
Enter D.A : 1250
Enter H.R.A : 1450
                Basic Pay  :      12000
       Dearness Allowance  :       1250
     House Rent Allowance  :       1450
                Gross Pay  :      14700
                    G.P.F  :       1325
               Income Tax  :       1470
                  Net Pay  :      11905
```

(HOT: Try to make multiple output statements as a single cout statement)

In the above program, every output statement has two setw( ) manipulators; first setw (25) creates a filed with 25 spaces and second setw(10) creates another field with 10 spaces. When you represent a value to these fields, it will show the value within the field from right to left.


Field 2 with 10 space width
New line modifier
cout<<setw(25)<<"Basic Pay:"<<setw(10)<<basic<<endl;
Field 1 with 25 space width
Data to accommodate in Field 1
Data to accommodate in Field 2

In field1 and field 2, the string "Basic Pay: " and the value of basic pay are shown as given in Figure 9.16 below.

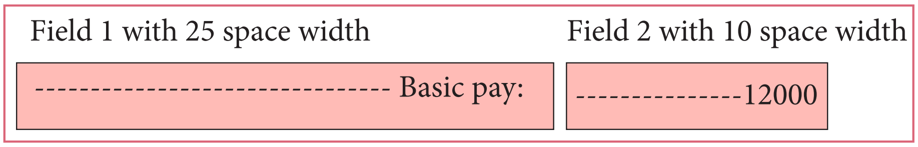

Field 1 with 25 space width
Field 2 with 10 space width
------------------------------ Basic pay:
--------------12000

Figure 9.16 setw( ) function

setprecision ( )

This is used to display numbers with fractions in specific number of digits.

Syntax:

setprecision (number of digits);

Example:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{       float hra = 1200.123;
        cout << setprecision (5) << hra; }
```

In the above code, the given value 1200.123 will be displayed in 5 digits including fractions. So, the output will be **1200.1**

setprecision ( ) prints the values from left to right. For the above code, first, it will take 4 digits and then prints one digit from fractional portion.

setprecision can also be used to set the number of decimal places to be displayed. In order to do this task, you will have to set an ios flag within **setf()** manipulator. This may be used in two forms: (i) **fixed** and (ii) **scientific**

These two forms are used when the keywords fixed or scientific are appropriately used before the setprecision manipulator.

Example:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{       cout.setf(ios::fixed);

        cout << setprecision(2)<<0.1; }
```

In the above program, ios flag is set to **fixed** type; it prints the floating point number in fixed notation. So, the output will be, 0.10

cout.setf(ios::scientific);

cout << setprecision(2) << 0.1;

In the above statements, ios flag is set to **scientific type;** it will print the floating point number in scientific notation. So, the output will be, 1.00e-001

9.15 Expression

An expression is a combination of operators, constants and variables arranged as per the rules of C++. It may also include function calls which return values. (Functions will be learnt in upcoming chapters).

An expression may consist of one or more operands, and zero or more operators to produce a value. In C++, there are seven types of expressions, and they are:

(i) Constant Expression (ii) Integer Expression
(iii) Floating Expression (iv) Relational Expression
(v) Logical Expression (vi) Bitwise Expression
(vii) Pointer Expression

SN	Expression	Description	Example
1	Constant Expression	Constant expression consist only constant values	int num=100;

2	Integer Expression	The combination of integer and character values and/or variables with simple arithmetic operators to produce integer results.	sum=num1+num2;
			avg=sum/5;

3	Float Expression	The combination of floating point values and/or variables with simple arithmetic operators to produce floating point results.	Area=3.14*r*r;

4	Relational Expression	The combination of values and/or variables with relational operators to produce bool(true means 1 or false means 0) values as results.	x>y;
			a+b==c+d;

5	Logical Expression	The combination of values and/or variables with Logical operators to produce bool values as results.	(a>b)&& (c==10);

6	Bitwise Expression	The combination of values and/or variables with Bitwise operators.	x>>3;
			a<<2;

7	Pointer Expression	A Pointer is a variable that holds a memory address. Pointer variables are declared using (*) symbol.	int *ptr;

Table 9.10 : Types of Expressions

9.16 Type Conversion

The process of converting one fundamental data type into another is called as "Type Conversion". C++ provides two types of conversions.

(1) Implicit type conversion

(2) Explicit type conversion.

(1) Implicit type conversion:

An Implicit type conversion is a conversion performed by the compiler automatically. So, implicit conversion is also called as **"Automatic conversion".**

This type of conversion is applied usually whenever different data types are intermixed in an expression. If the type of the operands differ, the compiler converts one of them to match with the other, using the rule that the "smaller" type is converted to the "wider" type, which is called as **"Type Promotion".**

For example:

```
#include <iostream>
using namespace std;
int main()
{
        int a=6;
        float b=3.14;
        cout << a+b;
}
```

In the above program, operand **a** is an int type and **b** is a float type. During the execution of the program, int is converted into a float, because a float is wider than int. Hence, the output of the above program will be: **9.14**

The following Table 9.11 shows you the conversion pattern.

RHO LHO	**char**	**short**	**int**	**long**	**float**	**double**	**long double**
char	int	int	int	long	float	double	long double
short	int	int	int	long	float	double	long double
int	int	int	int	long	float	double	long double
long	long	long	long	long	float	double	long double
float	float	float	float	float	float	double	long double
double	double	double	double	double	double	double	long double
long double	long double	long double	long double	long double	long double	long double	long double

(RHO – Right Hand Operand; LHO – Left Hand Operand)

Table 9.11: Implicit conversion of mixed operands

(2) Explicit type conversion

C++ allows explicit conversion of variables or expressions from one data type to another specific data type by the programmer. It is called as **"type casting".**

Syntax:

(type-name) expression;

Where type-name is a valid C++ data type to which the conversion is to be performed.

Example:

```
#include <iostream>
using namespace std;
int main( )
{
        float varf=78.685;
        cout << (int) varf;
}
```

In the above program, variable **varf is** declared as a **float** with an initial value 78.685. The value of **varf** is explicitly converted to an **int** type in cout statement. Thus, the final output will be 78.

During explicit conversion, if you assign a value to a type with a greater range, it does not cause any problem. But, assigning a value of a larger type to a smaller type may result in loosing or loss of precision values.

S.No	Explicit Conversion	Problem
1	double to float	Loss of precision. If the original value is out of range for the target type, the result becomes undefined
2	float to int	Loss of fractional part. If original value may be out of range for target type, the result becomes undefined
3	long to short	Loss of data

Table 9.12 – Explicit Conversion Problems

Example:

```
#include <iostream>
using namespace std;
int main()
{
        double varf=178.25255685;
        cout << (float) varf << endl;
        cout << (int) varf << endl;
}
```

Output:

```
178.253
178
```

Evaluate Yourself

1. What is meant by type conversion?
2. How implicit conversion is different from explicit conversion?
3. What is the difference between endl and \n?
4. What is the use of references?
5. What is the use of setprecision ( ) ?

Hands on practice:

1. Write C++ programs to interchange the values of two variables.
   a. Using the third variable
   b. Without using third variable
2. Write C++ programs to do the following:
   a. To find the perimeter and area of a quadrant.
   b. To find the area of triangle.
   c. To convert the temperature from Celsius to Fahrenheit.
3. Write a C++ to find the total and percentage of marks you secured from 10th Standard Public Exam. Display all the marks one-by-one along with total and percentage. Apply formatting functions.

Points to Remember

- Every programming language has two fundamental elements, viz., data types and variables.
- In C++, the data types are classified as three main categories (1) Built-in data types (2) User-defined data types (3) Derived data types.
- The variables are the named space to hold values of certain data type.
- There are five fundamental data types in C++: char, int, float, double and void.
- C++ compiler allocates specific memory space for each and every data handled according to the compiler's standards.
- Variables are user-defined names assigned to a memory location in which the values are stored.
- Declaration is a process to instruct the compiler to allocate memory as per the type specified along with the variable name.
- Manipulators are used to format output of any C++ program. Manipulators are functions specifically designed to use with the insertion (<<) and extraction(>>) operators.
- An expression is a combination of operators, constants and variables arranged as per the rules of C++.
- The process of converting one fundamental data type into another is called as "Type Conversion". C++ provides two types of conversions (1) Implicit type conversion and (2) Explicit type conversion.

Evaluation

SECTION – A

F9H7JM

Choose the correct answer

1. How many categories of data types are available in C++?
   (a) 5 (b) 4 (c) 3 (d) 2
2. Which of the following data types is not a fundamental type?
   (a) signed (b) int (c) float (d) char
3. What will be the result of following statement?
   char ch= 'B';
   cout << (int) ch;
   (a) B (b) b (c) 65 (d) 66
4. Which of the character is used as suffix to indicate a floating point value?
   (a) F (b) C (c) L (d) D
5. How many bytes of memory is allocated for the following variable declaration if you are using Dev C++? short int x;
   (a) 2 (b) 4 (c) 6 (d) 8
6. What is the output of the following snippet?
   char ch = 'A';
   ch = ch + 1;
   (a) B (b) A1 (c) F (d) 1A
7. Which of the following is not a data type modifier?
   (a) signed (b) int (c) long (d) short
8. Which of the following operator returns the size of the data type?
   (a) sizeof( ) (b) int ( ) (c) long ( ) (d) double ( )
9. Which operator is used to access reference of a variable?
   (a) $ (b) # (c) & (d) !
10. This can be used as alternate to endl command:
   (a) \t (b) \b (c) \0 (c) \n

SECTION-B

Very Short Answers

1. Write a short note on const keyword with an example.
2. What is the use of setw( ) format manipulator?
3. Why is char often treated as integer data type?
4. What is a reference variable? What is its use?
5. Consider the following C++ statement. Are they equivalent?
   char ch = 67; char ch = 'C';

6. What is the difference between 56L and 56?

7. Determine which of the following are valid constant? And specify their type.

   (i) 0.5 (ii) 'Name' (iii) '\t' (iv) 27,822

8. Suppose x and y are two double type variable that you want add as integer and assign to an integer variable. Construct a C++ statement to do the above.

9. What will be the result of following if num=6 initially.

   (a) cout << num;

   (b) cout << (num==5);

10. Which of the following two statements are valid? Why? Also write their result.

    (i) int a; a = 3,014; (ii) int a; a=(3,014);

SECTION-C

Short Answers

1. What are arithmetic operators in C++? Differentiate unary and binary arithmetic operators. Give example for each of them.

2. How relational operators and logical operators are related to one another?

3. Evaluate the following C++ expressions where x, y, z are integers and m, n are floating point numbers. The value of x = 5, y = 4 and m=2.5;

   (i) n = x + y / x;

   (ii) z = m * x + y;

   (iii) z *= x * m + x;

Reference:

(1) Object Oriented Programming with C++ (4th Edition), Dr. E. Balagurusamy, Mc.Graw Hills.

(2) The Complete Reference C++ (Forth Edition), Herbert Schildt. Mc.Graw Hills.

(3) Computer Science with C++ (A text book of CBSE XI and XII), Sumita Arora, Dhanpat Rai & Co.

Flow of Control

Learning Objectives

After learning this chapter, the students will be able to

- Understand the different kinds of statements.
- Construct different flow of control statements in C++.

FA63LA

10.1 Introduction

In the previous chapters you learnt the basic concepts of C++ programming such as variables, constants, operators, data types etc. Generally a program executes its statements sequentially from beginning to end. However, such a strict sequential ordering is restrictive and less useful. There are lot of situations where it is useful to decide the code block executed on the basis of a certain condition. In such situations, the flow of control jumps from one part of the code to another segment of code. Program statements that cause such jumps are called as **"Control flow"**. This chapter deals with the basics of control structures such as "Selection", "Iteration" and "Jump" statement.

10.2 Statements

A computer program is a set of statements or instructions to perform a specific task. These statements are intended to perform specific action. The action may be of variable declarations, expression evaluations, assignment operations, decision making, looping and so on.

There are two kinds of statements used in C++.

(i) Null statement

(ii) Compound statement

10.2.1 Null statement

The "**null** or **empty statement**" is a statement containing only a semicolon. It takes the flowing form:

; // it is a null statement

Null statements are commonly used as placeholders in iteration statements or as statements on which to place labels at the end of compound statements or functions.

10.2.2 Compound (Block) statement

C++ allows a group of statements enclosed by pair of braces {}. This group of statements is called as a compound statement or a block.

The general format of compound statement is:

```
{
        statement1;
        statement2;
        statement3;
}
```

For example

```
{
        int x, y;
        x = 10;
        y = x + 10;
}
```

The compound statement or block is a treated as a single unit and may appear anywhere in the program.

10.3 Control Statements

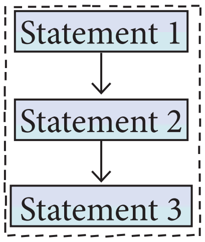
Control statements are statements that alter the sequence of flow of instructions.

In a program, statements may be executed sequentially, selectively or iteratively. Every programming languages provide statements to support sequence, selection (branching) and iteration.

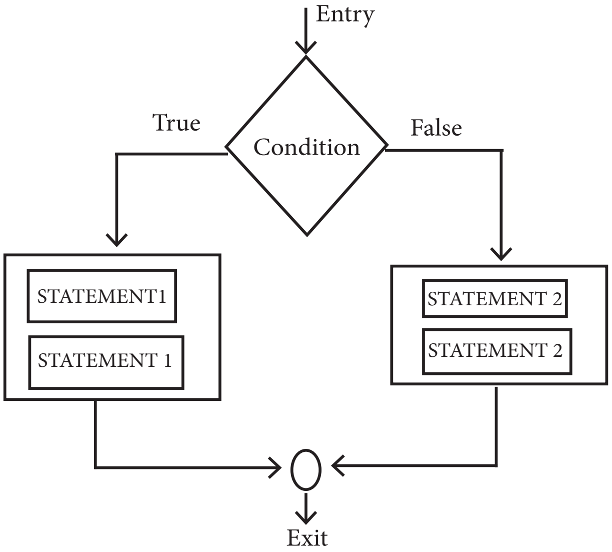
If the statements are executed sequentially, the flow is called as sequential flow. In some situations, if the statements alter the flow of execution like branching, iteration, jumping and function calls, this flow is called as control flow.

Sequence statement

The **sequential statement** are the statements, that are executed one after another only once from top to bottom. These statements do not alter the flow of execution. These statements are called as sequential flow statements. They always end with a semicolon (;).

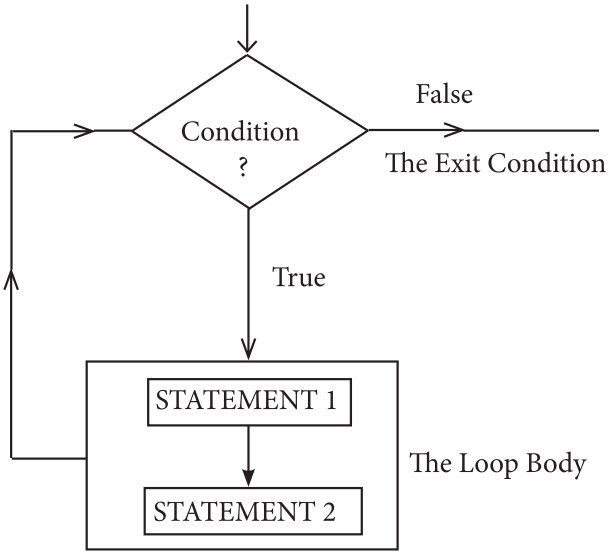


Selection statement



The selection statement means the statement (s) executed depend upon a condition. If a condition is true, a true block (a set of statements) is executed otherwise a false block is executed. This statement is also called **decision statement** or **selection statement** because it helps in making decision about which set of statements are to be executed.

Iteration statement



The **iteration statement** is a set of statement that are repetitively executed based upon a conditions. If a condition evaluates to true, the set of statements (true block) is executed again and again. As soon as the condition becomes false, the repetition stops. This is also known as **looping statement** or iteration statement.

The set of statements that are executed again and again is called the **body of the loop.**The condition on which the execution or exit from the loop is called **exit-condition** or **test-condition.**

Generally, all the programming languages support this type of statements to write programs depending upon the problem. C++ also supports this type of statements. These statements will be discussed in coming sections.

Note

In C++, any **non zero is treated as true including negative numbers and zero is treated as false.**

Selection statements and iteration statements are executed depending upon the conditional expression. The conditional expression evaluates either true or false.

10.4 Selection statements

In a program a decision causes a one time jump to a different part of a program. Decisions in C++ are made in several ways, most importantly with if .. else … statement which chooses between two alternatives. Another decision statement, switch creates branches for multiple alternatives sections of code, depending on the value of a single variable.

10.4.1 if statement

The if statement evaluates a condition, if the condition is true then a true-block (a statement or set of statements) is executed, otherwise the true-block is skipped.The general syntax of the if statement is:

```
if (expression)
        true-block;
statement-x;
```

In the above syntax, **if** is a keyword that should contain expression or condition which is enclosed within parentheses. If the expression is true (nonzero) then the true-block is executed and followed by statement-x are also executed, otherwise, the control passes to statement-x. The true-block may consists of a single statement, a compound statement or empty statement. The control flow of **if** statement and the corresponding flow chart is shown below.

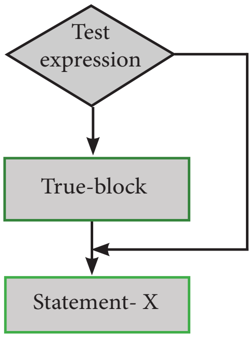


Illustration 10.1 C++ program to check whether a person is eligible to vote using if statement

```
#include <iostream>
using namespace std;
int main()
{
        int age;
        cout<< "\n Enter your age: ";
        cin>> age;
        if(age>=18)
                cout<< "\n You are eligible for voting ....";
        cout<< "This statement is always executed.";
        return 0;
}
```

The pair of braces is not required because if condition followed by only one statement

Output

```
        Enter your age: 23
        You are eligible for voting….
        This statement is always executed.
```

10.4.2 if-else statement

In the above examples of **if**, you have seen that, a block of statements are excecuted only if the condition evaluates to true. What if there is another course of action to be followed if the condition evaluates to false. There is another form of **if** that allows for this kind of either or condition by providing an else clause. The syntax of the if-else statement is given below:

```
if ( expression)
{
        True-block;
}
else
{
        False-block;
}
Statement-x
```

In if-else statement, first the expression or condition is evaluated to either true of false. If the result is true, then the statements inside true-block is executed and false-block is skipped. If the result is false, then the statement inside the false-block is executed i.e., the true-block is skipped.

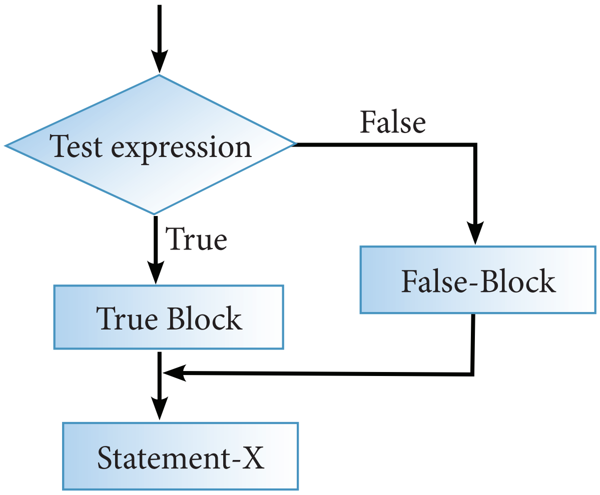


Illustration 10.2 C++ program to find whether the given number is even number or odd number using if-else statement

```
#include <iostream>
using namespace std;
int main()
{
        int num, rem;
        cout<< "\n Enter a number: ";
        cin>>num;
        rem = num % 2;
        if (rem==0)
                cout<< "\n The given number" <<num<< " is Even";
        else
                cout<< "\n The given number "<<num<< " is Odd";
        return 0;
}
```

Output

```
Enter number: 10
The given number 10 is Even
```

In the above program, the remainder of the given number is stored in rem. If the value of rem is zero, the given number is inferred as an even number otherwise, it is inferred as on odd number.

10.4.3 Nested if

An if statement which contains another if statement is called nested if. The nested can have one of the following three forms.

1. If nested inside if part
2. If nested inside else part
3. If nested inside both if part and else part

The syntax of the nested if:

If nested inside if part

```
if (expression-1)
{
    if (expression-2)
    {
      True_Part_Statements;
    }
    else
    {
      False_Part_Statements;
    }
}
else
    body of else part;
```

If nested inside else part

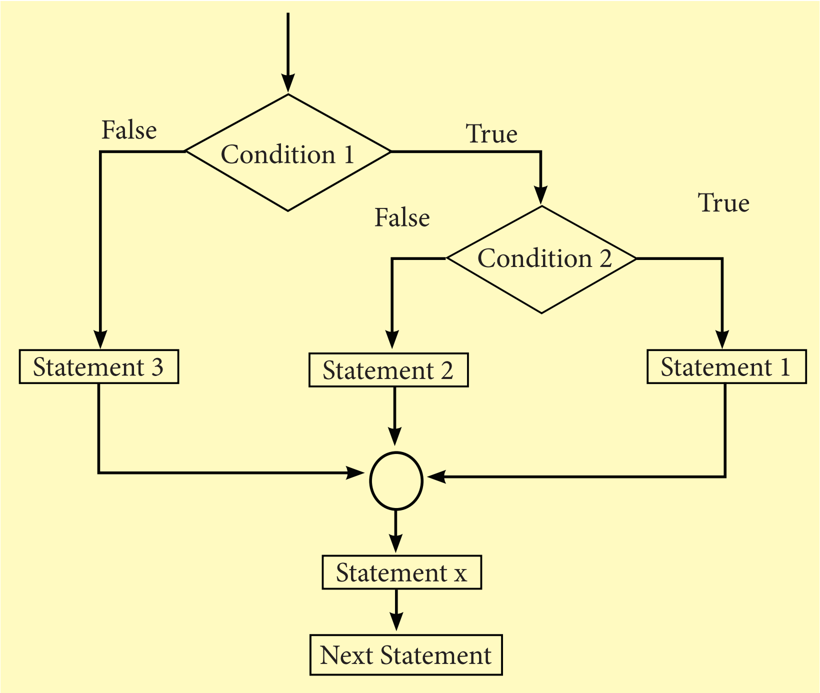
```
if (expression-1)
{
    body of true part;
}
else
{
    if (expression-2)
    {
      True_Part_Statements;
    }
    else
    {
      False_Part_Statements;
    }
}
```

If nested inside both if part and else part

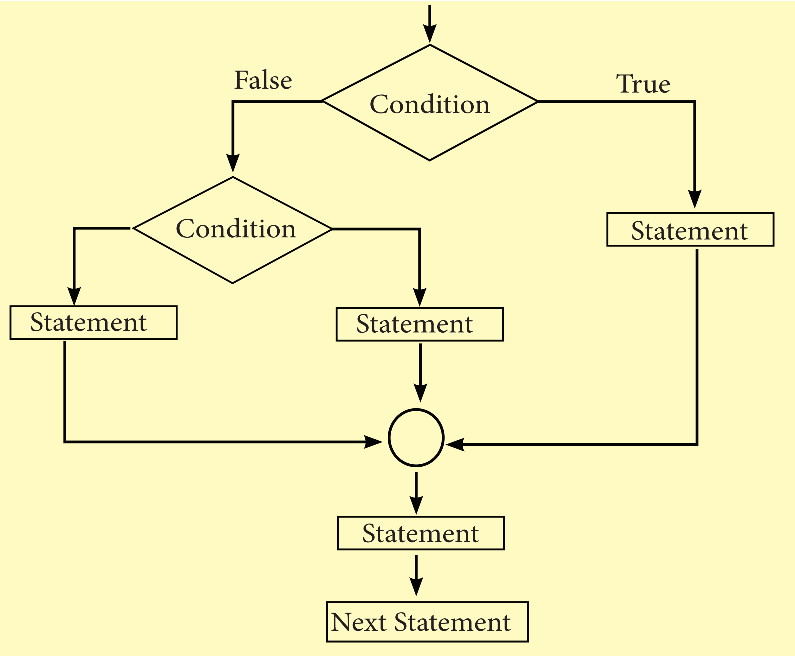
```
if (expression)
{
    if (expression)
    {
      True_Part_Statements;
    }
    else
    {
      False_Part_Statements;
    }
}
else
{
    if (expression)
    {
      True_Part_Statements;
    }
    else
    {
      False_Part_Statements;
    }
}
```

In the first syntax of the nested if mentioned above the expression-1 is evaluated and the expression result is false then control passes to statement-m. Otherwise, expression-2 is evaluated,if the condition is true, then Nested-True-block is executed, next statement-n is also executed. Otherwise Nested-False-Block, statement-n and statement-m are executed.

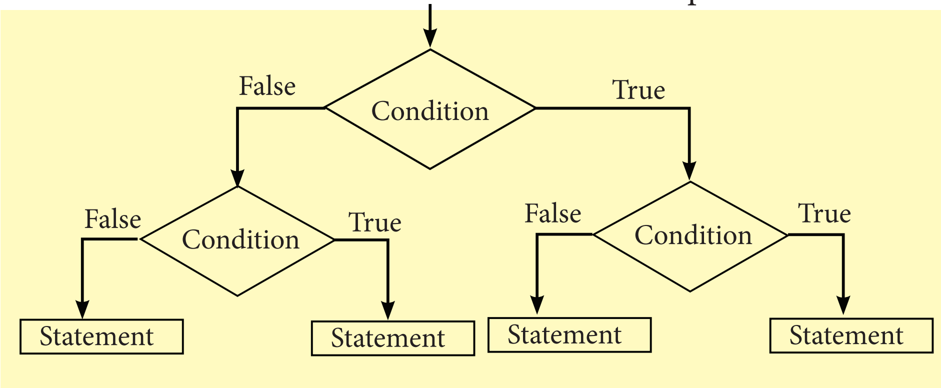
The working procedure of the above said if..else structures are given as flowchart below:



Flowchart 10.1 if nested inside if Part



Flowchart 10.2 If nested inside else part



Flowchart 10.3 If nested inside both if part and else part

Illustration 10.3 – C++ program to calculate commission according to grade using nested if statement

```
#include <iostream>
using namespace std;
int main()
{
        int sales, commission;
        char grade;
        cout << "\n Enter Sales amount: ";
        cin >> sales;
        cout << "\n Enter Grade: ";
        cin >> grade;
        if (sales > 5000)
        {
                commission = sales * 0.10;
                cout << "\n Commission: " << commission;
        }
        else
        {
                commission = sales * 0.05;
                cout << "\n Commission: " << commission;
        }
        cout << "\n Good Job ..... ";
        return 0;
}
```

Output:

```
Enter Sales amount: 6000
Enter Grade: A
Commission: 600
Good Job .....
```

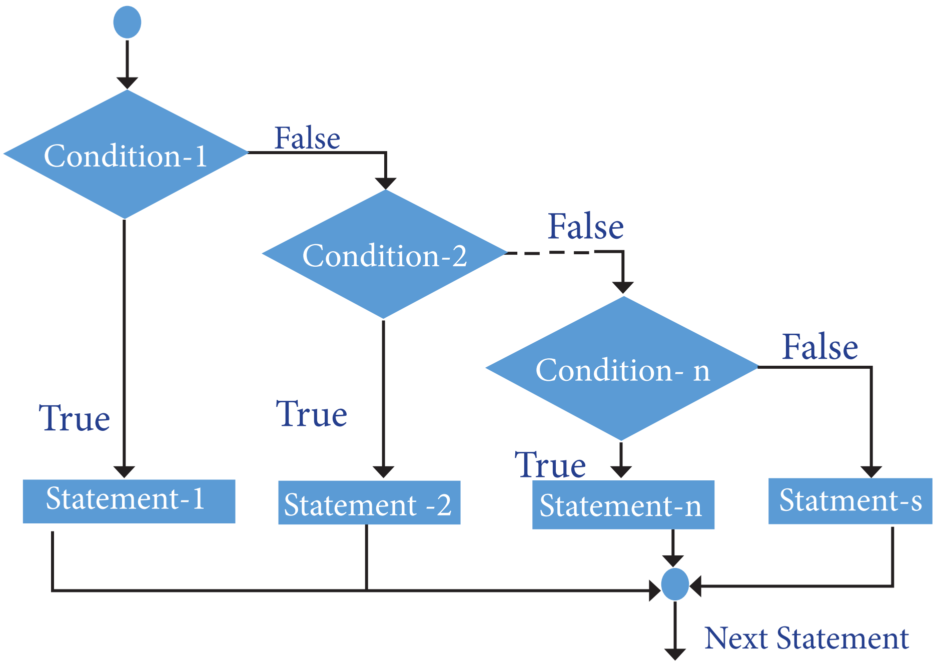
10.4.4 if -else-if ladder

The if-else ladder is a multi-path decision making statement. In this type of statement '**if**' is followed by one or more **else if** statements and finally end with an **else** statement.

The syntax of if-else ladder:

```
if (expression 1)
{
  Statement-1
}
else
        if( expression 2)
        {
                Statement-2
        }
        else
                if ( expression 3)
                {
                        Statement-3
                }
                else
                {
                        Statement-4
                }
```

When the respective expression becomes true, the statement associated with block is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final else statement will be executed.



Flowchart 10.4 if-else ladder flow chart

Illustration 10.4 C++ program to find your grade using if-else ladder.

```
#include <iostream>
using namespace std;
int main ()
{
int marks;
cout<<" Enter the Marks  :";
cin>>marks;
if( marks >= 60 )
          cout<< "Your grade is 1st class !!" <<endl;
                    else if( marks >= 50 && marks < 60)
                              cout<< "your grade is 2nd class !!" <<endl;
                                        else if( marks >= 40 && marks < 50)
                                                  cout<< "your grade is 3rd class !!" <<endl;
else
          cout<< "You are fail !!" <<endl;
return 0;
}
```

Output

```
Enter the Marks  :60
Your grade is 1st class !!
```

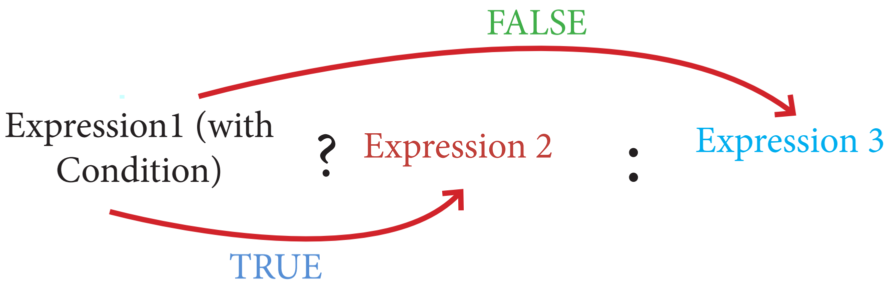
When the marks are greater than or equal to 60, the message **"Your grade is 1st class !!"** is displayed and the rest of the ladder is bypassed. When the marks are between 50 and 59, the message **"Your grade is 2nd class !!"** is displayed, and the other ladder is bypassed. When the mark between 40 to 49, the message **"Your grade is 3rd class !!"** is displayed, otherwise, the message "You are fail !!" is displayed.

10.4.5 The ?: Alternative to if- else

The conditional operator (or Ternary operator) is an alternative for ‘if else statement’. The conditional operator that consists of two symbols (?:). It takes three arguments. The control flow of conditional operator is shown below:

The syntax of the conditional operator is:

```
expression 1? expression 2 : expression 3
```



In the above syntax, the expression 1 is a condition which is evaluated, if the condition is true (Non-zero), then the control is transferred to expression 2, otherwise, the control passes to expression 3.

Illustration 10.5 – C++ program to find greatest of two numbers using conditional operator

```
#include <iostream>
using namespace std;
int main()
{
        int a, b, largest;
        cout << "\n Enter any two numbers: ";
        cin >> a >> b;
         largest = (a>b)? a : b;
        cout << "\n Largest number : " << largest;
        return 0;
}
```

Output:

```
 Enter any two numbers: 12 98
 Largest number : 98
```

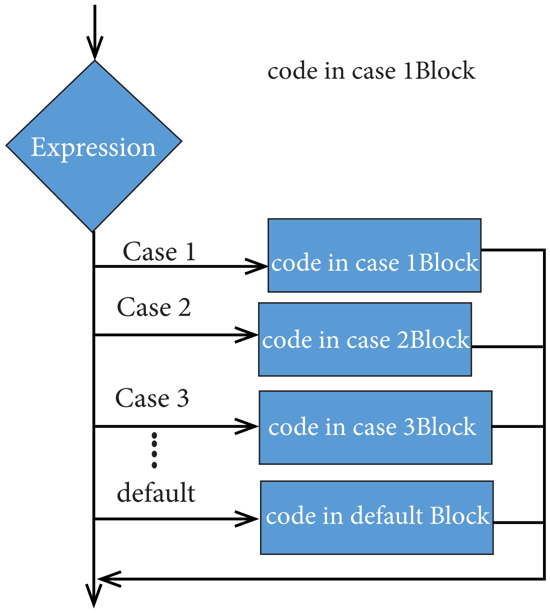
10.4.6 Switch statement

The switch statement is a multi-way branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression. The switch statement replaces multiple if-else sequence.

The syntax of the switch statement is;

```
switch(expression)
{
        case constant 1:
                statement(s);
                break;
        case constant 2:
                statement(s);
                break;
        .
        .
        .
        .
        default:
                statement(s);
}
```

In the above syntax, the expression is evaluated and if its value matches against the constant value specified in one of the case statements, that respective set of statements are executed. Otherwise, the statements under the default option are executed. The workflow of switch statement and flow chart are shown below.



Flowchart10.5: workflow of switch and flow chart

Rules:

1. The expression provided in the switch should result in a constant value otherwise it would not be valid.
2. Duplicate case values are not allowed.
3. The default statement is optional.
4. The break statement is used inside the switch to terminate a statement sequence. When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.

5. The break statement is optional. If omitted, execution will continue on into the next case. The flow of control will fall through to subsequent cases until a break is reached.
6. Nesting of switch statements is also allowed.

Illustration 10.6 – C++ program to demonstrate switch statement

```
#include <iostream>
using namespace std;
int main()
{
        int num;
        cout << "\n Enter week day number: ";
        cin >> num;
        switch (num)
        {
                case 1 : cout << "\n Sunday"; break;
                case 2 : cout << "\n Monday"; break;
                case 3 : cout << "\n Tuesday"; break;
                case 4 : cout << "\n Wednessday"; break;
                case 5 : cout << "\n Thursday"; break;
                case 6 : cout << "\n Friday"; break;
                case 7 : cout << "\n Saturday"; break;
                default: cout << "\n Wrong input....";
        }
}
```

Output:

```
Enter week day number: 6
Friday
```

10.4.7 Switch vs if-else

"if-else" and "switch" both are selection statements. The selection statements, transfer the flow of the program to the particular block of statements based upon whether the condition is "true" or "false". However, there are some differences in their operations. These are given below:

Key Differences Between if-else and switch

S.No	if-else	Switch
1	Expression inside if statement decide whether to execute the if block or under else block.	expression inside switch statement decide which case to execute.
2	An if-else statement uses multiple statements for multiple choices	switch statement uses single expression for multiple choices.
3	If-else statement checks for equality as well as for logical expression.	switch checks only for equality.
4	The **if** statement evaluates integer, character, pointer or floating-point type or Boolean type.	switch statement evaluates only character or a integer data type.
5	If the condition is false the else block statements will be executed	If the condition is false then the default statements are executed.

The if statement is more flexible than switch statement.

10.5 Iteration statements

An iteration (or looping) is a sequence of one or more statements that are repeatedly executed until a condition is satisfied. These statements are also called as control flow statements. It is used to reduce the length of code, to reduce time, to execute program and takes less memory space. C++ supports three types of iteration statements;

- for statement
- while statement
- do-while statement

All looping statements repeat a set statements as long as a specified condition is remains true. The specified condition is referred as a loop control. For all three loop statements, a true condition is any nonzero value and a zero value shows a false condition.

10.5.1 Parts of a loop

Every loop has four elements that are used for different purposes. These elements are

- Initialization expression
- Test expression
- Update expression
- The body of the loop

Initialization expression(s): The control variable(s) must be initialized before the control enters into loop. The initialization of the control variable takes place under the initialization expressions. The initialization expression is executed only once in the beginning of the loop.

Test Expression: The test expression is an expression or condition whose value decides whether the loop-body will be execute or not. If the expression evaluates to true (i.e., 1), the body of the loop gets executed, otherwise the loop is terminated.

In an entry-controlled loop, the test-expression is evaluated before the entering into a loop whereas in an exit-controlled loop, the test-expression is evaluated before exit from the loop.

Update expression: It is used to change the value of the loop variable. This statement is executed at the end of the loop after the body of the loop is executed.

The body of the loop: A statement or set of statements forms a body of the loop that are executed repetitively. In an entry-controlled loop, first the test-expression is evaluated and if it is nonzero, the body of the loop is executed otherwise the loop is terminated. In an exit-controlled loop, the body of the loop is executed first then the test-expression is evaluated. If the test-expression is true the body of the loop is repeated otherwise loop is terminated

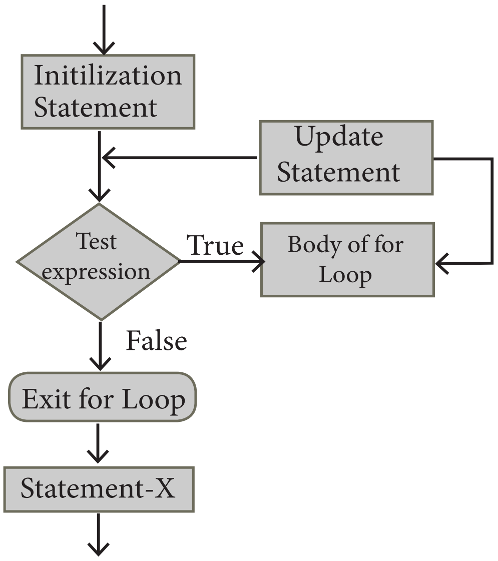
10.5.2 for loop

The for loop is a entry- controlled loop and is the easiest looping statement which allows code to be executed repeatedly. It contains three different statements (initialization, condition or test-expression and update expression(s)) separated by semicolons.

The general syntax is:

```
for (initialization(s); test-expression; update expression(s))
{
        Statement 1;
        Statement 2;
        ………….
}
Statement-x;
```

The initialization part is used to initialize variables or declare variable which are executed only once, then the control passes to test-expression. After evaluation of test-expression, if the result is false, the control transferred to statement-x. If the result is true, the body of the for loop is executed, next the control is transferred to update expression. After evaluation of update expression part, the control is transferred to the test-expression part. Next the steps 3 to 5 is repeated. The workflow of for loop and flow chart are shown below.



Flowchart 10.6: Workflow of for loop and flow chart

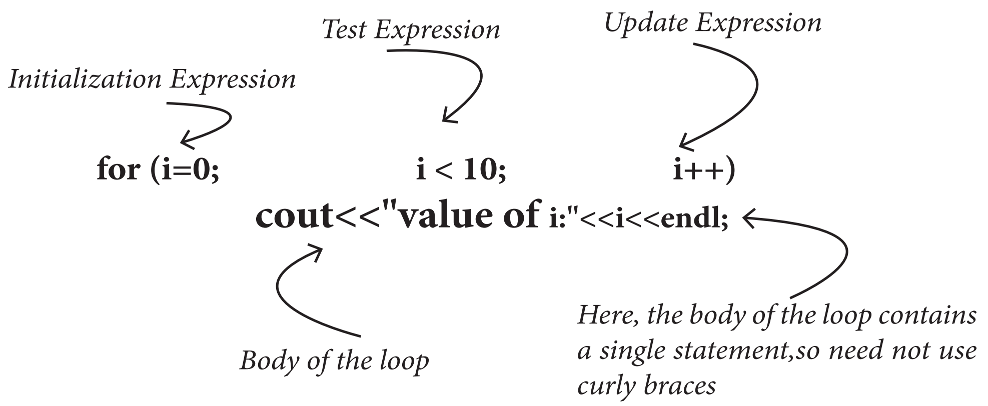
Illustration 10.7 C++ program to display numbers from 0 to 9 using for loop

```
#include <iostream>
using namespace std;
int main ()
{
int i;
for(i = 0; i< 10; i ++ )
        cout<< "value of i : " <<i<<endl;
return 0;
}
```

Output

```
value of i : 0
value of i : 1
value of i : 2
value of i : 3
value of i : 4
value of i : 5
value of i : 6
value of i : 7
value of i : 8
value of i : 9
```

The following lines describes the working of the above given for loop:



Test Expression

Update Expression

Initialization Expression

```
for (i=0;          i < 10;          i++)
     cout<<"value of i:"<<i<<endl;
```

Body of the loop

Here, the body of the loop contains a single statement,so need not use curly braces

In the above program, first the variable i is initialized, next i is compared with 10, if i is less than ten, the value of i is incremented. In this way, the numbers 0 to 9 are displayed. Once i becomes 10, it is no longer < 10. So, the control comes out of the for loop.

Illustration 10.8 C++ program to sum the numbers from 1 to 10 using for loop

```
#include <iostream>
using namespace std;
int main ()
{
int i,sum=0;
for(i=1; i<=10;i++)
  {
          sum=sum+i;
  }
cout<<"The sum of 1 to 10 is "<<sum;
return 0;
}
```

Output

```
The sum of 1 to 10 is 55
```

Variations of for loop

The **for** is one of the most important looping statement in C++ because it allows a several variations. These variations increase the flexibility and applicability of **for** loop. These variations will be discussed below:

Multiple initialization and multiple update expressions

Multiple statements can be used in the initialization and update expressions of **for** loop. These multiple initialization and multiple update expressions are separated by commas. For example,

```
#include<iostream>
using namespace std;
int main()
{
        int i, j;
                for(i=0, j=10 ;  i<j  ;  i++,j--)
                {
                cout<<"\nThe value of i is"<<i<<" The value of j is "<<j;
```

Multiple initialization expressions (separated by commas)

Multiple update expressions (separated by commas)

```
}
return 0;
}
```

Output

```
The value of i is 0  The value of j is 10
The value of i is 1  The value of j is 9
The value of i is 2  The value of j is 8
The value of i is 3  The value of j is 7
The value of i is 4  The value of j is 6
```

In the above example, the initialization part contains two variables i and j and update expression contains i++ and j++. These two variables are separated by commas which is executed in sequential order i.e., during initialization firstly i=0 followed by j=10. Similarly, in update expression, firstly i++ is evaluated followed by j++ is evaluated.

Prefer prefix operator over postfix

Generally, the update expression contains increment/decrement operator (++ or --). In this part, always prefer prefix increment/decrement operator over postfix when to be used alone. The reason behind this is that when used alone, prefix operators are executed faster than postfix.

Optional expressions

Generally, the for loop contains three parts, i.e., initialization expressions, test expressions and update expressions. These three expressions are optional in a for loop.

Illustration 10.9 C++ program to sum the numbers from 1 to n

```
#include <iostream>
using namespace std;
int main ()
{
int i, sum=0, n;
cout<<"\n Enter The value of n";
cin>>n;
i =1;
for ( ; i<=n; )
 {
        sum += i;
        ++i;
 }
cout<<"\n The sum of 1 to " <<n<<"is "<<sum;
return 0;
}
```

Output

```
Enter the value of n 5
The sum of 1 to 5 is 15
```

In the above code, the update expression is not given, but a semicolon is necessary before the update expression.

```
for (    ;        i<=n;   )
```

Initialization expression and update expressions are skipped

In the above code, neither the initialization nor the update expression is given in the for loop. If both or any one of expressions are absent then the control is transferred to conditional part.

infinite loop

An infinite loop will be formed if a test-expression is absent in a for loop. For example,

test - expression is skipped

```
for( i=0 ;  ;    ++i)
  cout<<"\n Welcome";
```

This statement is displayed infinitely

Similarly, the following for loop also forms an infinite loop.

All three expressions are skipped

```
for(    ;        ;        )
     cout<<"\n Welcome";
```

This statement is displayed infinitely

Empty loop

Empty loop means a loop that has no statement in its body is called an empty loop. Following for loop is an empty loop:

```
for( i=0        ;        i<=5;   +=i);
```

The body of for loop contains a null statement

In the above code, the for loop contains a null statement, it is an empty loop.

Similarly, the following for loop also forms an empty loop.

```
int i;
for( i=0        ;        i<=5;   ++i);
{
          cout<<"\nWe are Indians";
}
```

The body of for loop contains a null statement

The body of for loop is not executed because semicolon(;) is given at the end of for loop.

In the above code, the body of a for loop enclosed in braces is not executed because a semicolon is given after the for loop.

Declaration of variable in a for loop

In C++, the variables can also be declared within a for loop. For instance,

```
int main ()
{
  int sum = 0;

  for(int i=0;  i<=5;  ++i )
  {
       sum = sum + i;
  }
  cout<<"\nThe variable i cannot be accessed here";
  cout<<"\n The variable sum can be accessed here";
}
```

Variable (i)is declared within the for loop.

The variable i can be accessed only within the body of loop.

A variable declared inside the block of main() can be accessed anywhere inside main() i.e., the scope of variable in main()

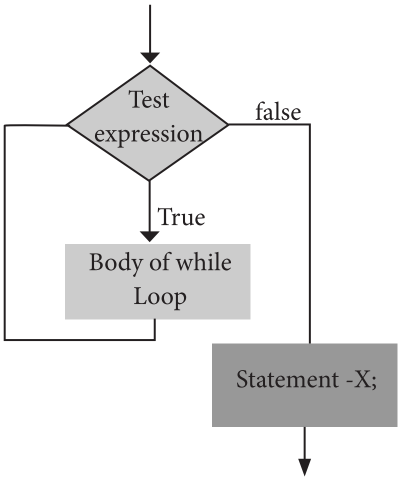
10.5.3 While loop

A while loop is a control flow statement that allows the loop statements to be executed as long as the condition is true. The while loop is an entry-controlled loop because the test-expression is evaluated before entering into a loop.

The while loop syntax is:

```
while ( Test expression )
{
        Body of the loop;
}
Statement-x;
```

The control flow and flow chart of the while loop is shown below.



Flowchart 10.7: while loop control flow and flowchart

In while loop, the test expression is evaluated and if the test expression result is true, then the body of the loop is executed and again the control is transferred to the while loop. When the test expression result is false the control is transferred to statement-x.

Illustration 10.10 C++ program to sum numbers from 1 to 10 using while loop

```
#include <iostream>
using namespace std;
int main ()
{
int i=1,sum=0;
while(i<=10)
{
        sum=sum+i;
        i++;
}
cout<<"The sum of 1 to 10 is "<<sum;
return 0;
}
```

Output

```
The sum of 1 to 10 is 55
```

In the above program, the integer variable i is initialized to 1 and the variable sum to 0. The while loop checks the condition, i < 10, if the condition is true, the value of i, which is added to sum and i is incremented by 1. Again, the condition i < 10 is checked. Since 2 < 10, 2 is added to the earlier value of sum. This continues until i becomes 11. At this point in time, 11 < 10 evaluates to false and the while loop terminates. After the loop termination, the value of sum is displayed.

Illustration 10.11 C++ program to find sum and average of 5 numbers using while loop

```
#include <iostream>
using namespace std;
int main ()
{
int i=1,num,avg,sum=0;
while(i<=5)
{
        cout<<"Enter the number : ";
        cin>>num;
        sum=sum+num;
        i++;
}
avg=sum/5;
cout<<"The sum is "<<sum<<endl;
cout<<"The average is "<<avg;
return 0;
}
```

Output

```
Enter the number : 1
Enter the number : 2
Enter the number : 3
Enter the number : 4
Enter the number : 5
The sum is 15
The average is 3
```

In the above program, integer variables **num** and **avg** are declared and variable i is initialized to 1 and sum to 0. The while loop checks the condition, since i <= 5 the condition is true, a number is read from the user and this is added to sum and i is incremented by 1. Now, the condition is i <= 5 is again checked. Since 2 <=5, the second number is obtained from the user and it is added to sum. This continues, until i becomes 6, at which point the while loop terminates. After the loop termination, the avg is computed and both sum and avg are displayed.

While loop variation

A while loop may contain several variations. It can be an empty loop or an infinite loop. An empty while loop does not have any statement inside the body of the loop except null statement i.e., just a semicolon.
For example

```
int main()
{
     int i=0;
     while(++i < 10000 )
                          ──────────→  This is an empty loop because the while
                                       loop does not contain any statement
     return 0;
}
```

In the above code, the loop is a time delay loop. A time delay loop is useful for pausing the program for some time.

A while loop may be infinite loop when no update statement is given inside the body of the loop. For example,

```
int main()
{
     int i = 0;
     while(i < =10)
        cout <<"The value of i is "<<i;  ──→  This statement will be displayed
                                              infinitely because no update
                                              statement inside the body of the loop
         i++;  ──┐
                 │
      return 0;  └──→  This is not a part of the while loop statement
                       because of missing curly braces
}
```

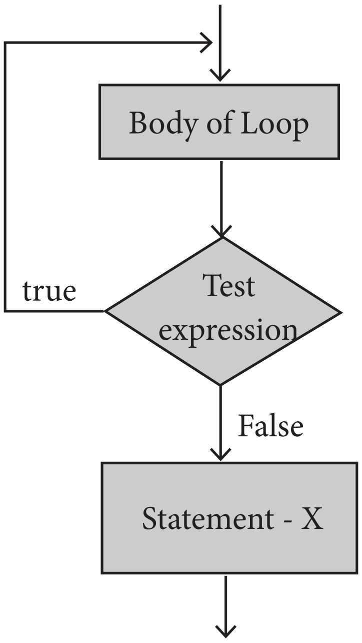
10.5.4 do-while loop

The do-while loop is an exit-controlled loop. In do-while loop, the condition is evaluated at the bottom of the loop after executing the body of the loop. This means that the body of the loop is executed at least once, even when the condition evaluates false during the first iteration. The do-while loop syntax is:

```
do
{
        Body of the loop;

} while(condition);
```

The flow control and flowchart for do-while loop is shown below



Flowchart 10.8 : do-while loop control flow and flowchart

Illustration 10.12 C++ program to display number from 10 to 1 using do-while loop

```
#include <iostream>
using namespace std;
int main ()
{
int n = 10;
do
{
        cout<<n<<", ";
        n--;
}while (n>0) ;
}
```

Output

```
10, 9, 8, 7, 6, 5, 4, 3, 2, 1
```

In the above program, the integer variable **n** is initialized to **10**. Next the value of **n** is displayed as **10** and **n** is decremented by **1**. Now, the condition is evaluated, since **9 > 0**, again **9** is displayed and **n** is decremented to **8**. This continues, until **n** becomes equal to **0**, at which point, the condition **n > 0** will evaluate to false and the do-while loop terminates.

10.5.5 Nesting of loops

A loop which contains another loop is called as a nested loop.
The syntax is given below:

```
for (initialization(s); test-expression; update expression(s))
{
        for (initialization(s); test-expression; update expression(s)
        {
        statement(s);
         }
statement(s);
}
```

```
while(condition)
{
        while(condition)
        {
        statement(s);
        }
statement(s);
}
```

```
do
{
statement(s);
        do
        {
        statement(s);
        }while(condition);
} while( condition );
```

Illustration 10.13 C++ program to display matrix multiplication table using nested for loop

```
#include<iostream>
using namespace std;
int main(void)
{
   cout<< "A multiplication table:" <<endl <<"  1\t2\t3\t4\t5\t6\t7\t8\t9" <<endl<< "" <<endl;
   for(int c = 1; c < 10; c++)
   {
   cout<< c << "| ";
   for(int i = 1; i< 10; i++)
   {
   cout<<i * c << '\t';
   }
   cout<<endl;
   }
return 0;
}
```

Output

```
A multiplication table:
1       2     3     4     5     6     7     8     9
1| 1    2     3     4     5     6     7     8     9
2| 2    4     6     8     10    12    14    16    18
3| 3    6     9     12    15    18    21    24    27
4| 4    8     12    16    20    24    28    32    36
5| 5    10    15    20    25    30    35    40    45
6| 6    12    18    24    30    36    42    48    54
7| 7    14    21    28    35    42    49    56    63
8| 8    16    24    32    40    48    56    64    72
9| 9    18    27    36    45    54    63    72    81
```

10.6 Jump statements

Jump statements are used to interrupt the normal flow of program. Types of Jump Statements are

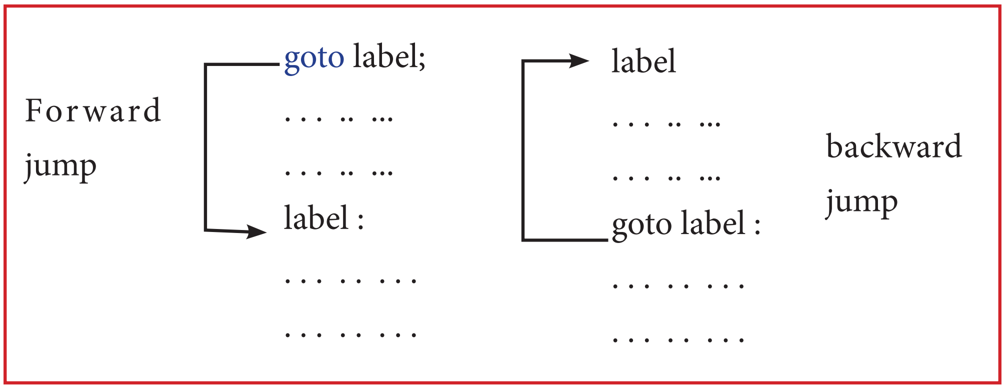
- goto statement
- break statement
- continue statement

10.6.1 goto statement

The goto statement is a control statement which is used to transfer the control from one place to another place without any condition in a program.

The syntax of the goto statement is;

Syntax1	Syntax2
goto label;	**label:**
--------------	-------------
--------------	-------------
--------------	-------------
label:	**goto label;**



In the syntax above, label is an identifier. When goto label; is encountered, the control of program jumps to label: and executes the code below it.

Illustration 10.14 C++ program to display the first five odd numberts using goto statement

```
# include <iostream>
using namespace std;
int main()
{
int n=1;
jump:
{
  if(n<10)
  { // Control of the program move to jump:
            cout<<n<<'\t';
            n+=2;
            goto jump;
  }
  else
            return 0;
}  }
```

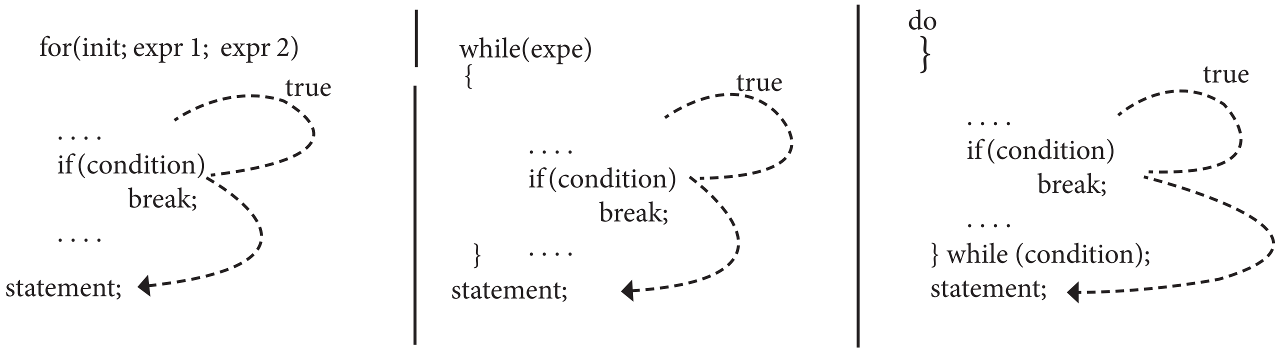
Output

```
1       2       5       7       9
```

In the above program the first five odd numbers are displayed.if n is is less than 10 goto transfers the control to jump statement .If n is greater than 10 the control comes out of the loop.

10.6.2 break statement

A break statement is a jump statement which terminates the execution of loop and the control is transferred to resume normal execution after the body of the loop. The following Figure. shows the working of break statement with looping statements;



```
for(init; expr 1;  expr 2)

  . . . .
  if(condition)
       break;
  . . . .

statement;
```

```
while(expe)
{

      . . . .
      if(condition)
           break;
  }   . . . .
statement;
```

```
do
 }

   . . . .
   if(condition)
        break;
   . . . .
} while (condition);
statement;
```

break statement in for, while and do-while loop

Illustration 10.15 C++ program to count N numbers using break statement

```
#include <iostream>
using namespace std;
int main ()
{
int count = 1;
do
{
cout<< "Count : " << count <<endl;
if( count > 3)
 {
          break;
  }
count ++;
}while( count < 20 );
return 0;
}
```

Output

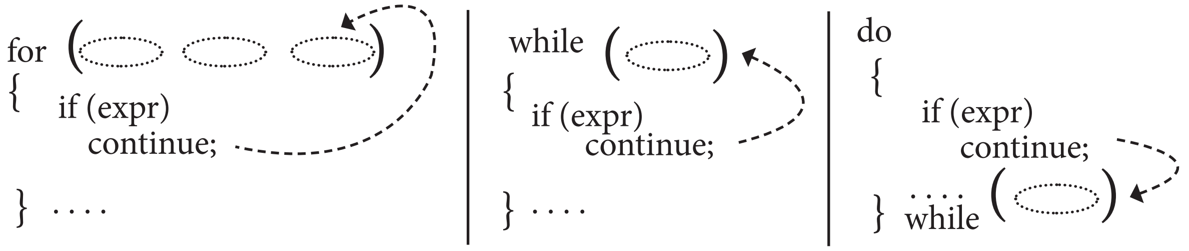
```
Count : 1
Count : 2
Count : 3
```

In the above example, the while loop will iterate for 20 times, but as soon as the count reaches 3, the loop is terminated, because of the break statement.

10.6.3 continue statement

The continue statement works quite similar to the break statement. Instead of terminating the loop (break statement), continue statement forces the loop to continue or execute the next iteration. When the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped and next iteration of the loop will begin.

The following Figure describes the working flow of the continue statement



The workflow of the continue statement

In the above example, the loop will iterate 10 times but, if i reaches 6, then the control is transferred to for loop, because of the continue statement.

Illustraion 10.16 C++ program to display numbers from 1 to 10 except 6 using continue statement

```
#include <iostream>
using namespace std;
 int main()
{
for (int i = 1; i<= 10; i++) {
if (i == 6)
continue;
else
cout<<i<< " ";
}
return 0;
}
```

Output

```
1 2 3 4 5 7 8 9 10
```

Difference between Break and Continue

Break	Continue
Break is used to **terminate** the execution of the loop.	Continue is not used to terminate the execution of loop.
It **breaks** the iteration.	It **skips** the iteration.
When this statement is executed, control will come out from the loop and executes the statement immediate after loop.	When this statement is executed, it will not come out of the loop but moves/jumps to the next iteration of loop.
Break is used with loops as well as switch case.	Continue is only used in loops, it is not used in switch case.

Hands on practice:

Write C++ program to solve the following problems :

1. Program to input a character and to print whether a given character is an alphabet, digit or any other character.

2. Program to print whether a given character is an uppercase or a lowercase character or a digit or any other character. use ASCII codes for it. The ASCII codes are as given below:

Characters	ASCII Range
0' - '9'	48 - 57
'A' - 'Z'	65 - 90
'a' - 'z'	97 - 122
other characters	0- 255 excluding the above mentioned codes.

3. Program to calculate the factorial of an integer.

4. Program to print fibonacci series i.e., 0 1 1 2 3 5 8......

5. Programs to produce the following design using nested loops

(a)	(b)
A	5 4 3 2 1
A B	5 4 3 2
A B C	5 4 3
A B C D	5 4
A B C D E	5
A B C D E F	

Points to Remember:

- A computer program is a set of statements or instructions to perform a specific task.
- There are two kinds of statements used in C++, viz Null and Compound Statement.
- Control Statement are statements that alter the sequence of flow of instaructions.
- There are three kinds of control statement used in C++. (1) Sequence Statement (2) Selection Statement (3) Iteration Statement
- *If* and *Switch* are Selection Statements.
- The Conditional Operator is an alternative for 'if else Statement'.
- The Switch Statment is a multi-way branching statement.
- Iteration Statement (looping) is use to execute a set of statements repeatedly until a condition is satisfied.
- There are three kinds Iteration Statements supported. (1) *for* (2) *While* (3) *do-While.*
- In C++ three Jump Statment are used (1) *goto* (2) *break* (3) *continue*

Evaluation

SECTION – A

FAEYMX

Choose the correct answer

1. What is the alternate name of null statement?
(A) No statement
(B) Empty statement
(C) Void statement
(D) Zero statement

2. In C++, the group of statements should be enclosed within:
(A) { } (B) [ ]
(C) ( ) (D) < >

3. The set of statements that are executed again and again in iteration is called as:
(A) condition (B) loop
(C) statement (D) body of loop

4. The multi way branch statement:
(A) if (B) if … else
(C) switch (D) for

5. How many types of iteration statements?
(A) 2 (B) 3
(C) 4 (D) 5

6. How many times the following loop will execute? for (int i=0; i<10; i++)
(A) 0 (B) 10
(C) 9 (D) 11

7. Which of the following is the exit control loop?

   (A) for (B) while

   (C) do…while (D) if…else

8. Identify the odd one from the keywords of jump statements:

   (A) break (B) switch

   (C) goto (D) continue

9. Which of the following is called entry control loop?

   (A) do-while (B) for

   (C) while (D) if-else

10. A loop that contains another loop inside its body:

    (A) Nested loop (B) Inner loop

    (C) Inline loop (D) Nesting of loop

SECTION-B

Very Short Answers

1. What is a null statement and compound statement?
2. What is selection statement? write it's types?
3. Correct the following code sigment:

```
if (x=1)
  p= 100;
else
  p = 10;
```

4. What will be the output of the following code:

```
int year;
cin >> year;
  if (year % 100 == 0)
          if ( year % 400 == 0)
                   cout << "Leap";
  else
          cout << "Not Leap year";
```

If the input given is (i) 2000 (ii) 2003 (iii) 2010?

5. What is the output of the following code?

```
for (int i=2; i<=10 ; i+=2)
  cout << i;
```

6. Write a for loop that displays the number from 21 to 30.
7. Write a while loop that displays numbers 2, 4, 6, 8.......20.
8. Compare an if and a ? : operator.

SECTION-C

Short Answers

1. Convert the following if-else to a single conditional statement:

```
if (x >= 10)
          a = m + 5;
else
          a = m;
```

2. Rewrite the following code so that it is functional:

```
v = 5;
do;
{
  total += v;
  cout << total;
while v <= 10
```

3. Write a C++ program to print multiplication table of a given number.
4. Write the syntax and purpose of switch statement.
5. Write a short program to print following series:

   (a) 1 4 7 10...... 40

SECTION - D

Explain in detail

1. Explain control statement with suitable example.
2. What is an entry control loop? Explain any one of the entry controlled loop with suitable example.

3. Write a program to find the LCM and GCD of two numbers.

4. Write programs to find the sum of the following series:

   (a) $x-\frac{x^2}{2!}+\frac{x^3}{3!}-\frac{x^4}{4!}+\frac{x^5}{5!}-\frac{x^6}{6!}$

   (b) $x+\frac{x^2}{2}+\frac{x^3}{3}+....+\frac{x^n}{n}$

5. Write a program to find sum of the series

   $S = 1 + x + x^2 +..... + x^n$

Reference:

(1) Object Oriented Programming with C++ (4th Edition), Dr. E. Balagurusamy, Mc.Graw Hills.

(2) The Complete Reference C++ (Forth Edition), Herbert Schildt. Mc.Graw Hills.

(3) Computer Science with C++ (A text book of CBSE XI and XII), Sumita Arora, Dhanpat Rai & Co.

Unit III | Introduction to C++

CHAPTER 11

Functions

Learning Objectives

After learning this chapter, the students will be able to

- Understand the Definition of Functions and uses of Functions
- Understand the Types of Functions – pre-defined and user-defined functions
- Apply mathematical functions for solving problems.
- Use String and Character functions for the manipulation of String and Character data
- Implement modular programming by creating functions
- Understand the role of arguments and compare different methods of the arguments
- Recognizes the scope of variables and functions in a program.

11.1 INTRODUCTION

A large program can be split into small sub-programs (blocks) called as functions where each sub-program can perform some specific functionality. Functions reduce the size and complexity of a program, makes it easier to understand, test, and check for errors. The functions which are available by default are known as **"Built-in"** functions and user can create their own functions known as **"User-defined"** functions.

- Built-in functions – Functions which are available in C++ language standard library.
- User-defined functions – Functions created by users.

11.2 Need for Functions

To reduce size and complexity of the program we use Functions. The programmers can make use of sub programs either writing their own functions or calling them from standard library.

1. Divide and Conquer

- Complicated programs can be divided into manageable sub programs called functions.
- A programmer can focus on developing, debugging and testing individual functions.
- Many programmers can work on different functions simultaneously.

2. Reusability

- Few lines of code may be repeatedly used in different contexts. Duplication of the same code can be eliminated by using functions which improves the maintenance and reduce program size.
- Some functions can be called multiple times with different inputs.

11.3 Types of Functions

Functions can be classified into two types,

1. Pre-defined or Built-in or Library Functions
2. User-defined Function.

C++ provides a rich collection of functions ready to be used for various tasks. The tasks to be performed by each of these are already written, debugged and compiled, their definitions alone are grouped and stored in files called **header files**. Such ready-to-use sub programs are called **pre-defined functions or built-in functions.**

C++ also provides the facility to create new functions for specific task as per user requirement. The name of the task and data required (arguments) are decided by the user and hence they are known as **User-defined functions.**

11.4 C++ Header Files and Built-in Functions

Header files provide function prototype and definitions for library functions. Data types and constants used with the library functions are also defined in them. A header file can be identified by their file extension **.h.** A single header file may contain multiple built-in functions.

For example: **stdio.h** is a header file that contains pre-defined **"standard input/ output"** functions.

11.4.1 Standard input/output (stdio.h)

This header file defines the standard I/O predefined functions **getchar(), putchar(), gets(), puts()** and etc.

11.4.1.1 getchar() and putchar() functions

The predefined function **getchar()** is used to get a single character from keyboard and **putchar()** function is used to display it.

Program 11.1 C++ code to accept a character and display it

```
#include<iostream>
#include<stdio.h>
using namespace std;
int main()
{
        cout<<"\n Type a Character : ";
        char ch = getchar();
        cout << "\n The entered Character is: ";
        putchar(ch);
        return 0;
}
```

Output:

```
Type a Character : T
The entered Character is: T
```

11.4.1.2. gets() and puts() functions

Function gets() reads a string from standard input and stores it into the string pointed by the variable. Function **puts()** prints the string read by **gets()** function in a newline.

Program 11.2 C++ code to accept and display a string

```
#include<iostream>
#include<stdio.h>
using namespace std;
int main()
{
        char str[50];
        cout<<"Enter a string : ";
        gets(str);
        cout<<"You entered: "
        puts(str);
        return(0);
}
```

Output :

```
Enter a string : Computer Science
You entered: Computer Science
```

11.4.2 Character functions (ctype.h)

This header file defines various operations on characters. Following are the various character functions available in C++. The header file **ctype.h** is to be included to use these functions in a program.

11.4.2.1.isalnum()

This function is used to check whether a character is **alphanumeric or not**. This function returns non-zero value if c is a digit or a letter, else it returns 0.

General Form:

```
int isalnum (char c)
```

Example :

```
int r = isalnum(‘5’);

cout << isalnum('A') <<’\t’<<r;
```

But the statements given below assign 0 to the variable n, since the given character is neither an alphabet nor a digit.

```
char c = '$';

int n = isalnum(c);

cout<<c;
```

Output:

```
0
```

Program 11.3

```
#include<iostream>
#include<stdio.h>
#include<ctype.h>
using namespace std;
int main()
{
        char ch;
        int r;
        cout<<"\n Type a Character :";
        ch = getchar();
        r = isalnum(ch);
        cout<<"\nThe Return Value of isalnum(ch) is :"<<r;
}
```

Output-1:

```
        Type a Character :A
        The Return Value of isalnum(ch) is :1
```

Output-2:

```
        Type a Character :?
        The Return Value of isalnum(ch) is :0
```

11.4.2.2. isalpha()

The isalpha() function is used to check whether the given character is an alphabet or not.

General Form:

isalpha(char c)

This function will return 1 if the given character is an alphabet, and 0 otherwise 0. The following statement assigns 0 to the variable n, since the given character is not an alphabet.

int n = isalpha('3');

But, the statement given below displays 1, since the given character is an alphabet.

cout << isalpha('a');

Program 11.4

```
#include<iostream>
#include<stdio.h>
#include<ctype.h>
using namespace std;
int main()
{
        char ch;
        cout << "\n Enter a charater: ";
        ch = getchar();
        cout<<"\n The Return Value of isalpha(ch) is :" << isalpha(ch) ;
}
```

Output-1:

```
        Enter a charater: A
        The Return Value of isalpha(ch) is :1
```

Output-2:

```
        Enter a charater: 7
        The Return Value of isalpha(ch) is :0
```

11.4.2.3 isdigit()

This function is used to check whether a given character is a digit or not. This function will return 1 if the given character is a digit, and 0 otherwise.

General Form:

isdigit(char c)

Program 11.5

```
using namespace std;
#include<iostream>
#include<ctype.h>
int main()
{
        char ch;
        cout << "\n Enter a Character: ";
        cin >> ch;
        cout<<"\n The Return Value of isdigit(ch) is :" << isdigit(ch) ;
}
```

Output-1

```
Enter a Character: 3
The Return Value of isdigit(ch) is :1
```

Output-2

```
Enter a Character: A
The Return Value of isdigit(ch) is :0
```

***Return 0; (Not Compulsory in latest compilers)**

11.4.2.4. islower()

This function is used to check whether a character is in lower case (small letter) or not. This functions will return a non-zero value, if the given character is a lower case alphabet, and 0 otherwise.

General Form:

islower(char c)

After executing the following statements, the value of the variable **n** will be **1** since the given character is in lower case.

```
char ch = 'n';
int n = islower(ch);
```

But the statement given below will assign 0 to the variable n, since the given character is an uppercase alphabet.

```
int n = islower('P');
```

11.4.2.5. isupper()

This function is used to check the given character is uppercase. This function will return 1 if true otherwise 0.

General Form:

isupper(char c)

For the following examples value **1** will be assigned to **n** and **0** for m.

```
int n=isupper('A');
int m=isupper('a');
```

11.4.2.6. toupper()

This function is used to convert the given character into its uppercase. This function will return the upper case equivalent of the given character. If the given character itself is in upper case, the output will be the same.

General Form:

char toupper(char c);

The following statement will assign the character constant **'K'** to the variable c.

```
char c = toupper('k');
```

But, the output of the statement given below will be **'B'** itself.

```
cout <<toupper('B');
```

11.4.2.7. tolower()

This function is used to convert the given character into its lowercase. This function will return the lower case equivalent of the given character. If the given character itself is in lower case, the output will be the same.

General Form:

char tolower(char c)

The following statement will assign the character constant '**k**' to the variable c.

```
char c = tolower('K');
```

But, the output of the statement given below will be **'b'** itself.

```
cout <<tolower('b');
```

11.4.3 String manipulation (string.h)

The library **string.h** (also referred as cstring) has several common functions for dealing with strings stored in array of characters. The string.h header file is to be included before using any string function.

11.4.3.1 strcpy()

General Form:

strcpy(Target String, Source String)

The **strcpy()** function takes two arguments: target and source. It copies the character string pointed by the source to the memory location pointed by the target. The null terminating character (**\0**) attached to the string is also copied.

Program 11.6

```
#include <string.h>
#include <iostream>
using namespace std;
int main()
{
        char source[] = "Computer Science";
        char target[20]="target";
        cout<<"\n String in Source Before Copied :"<<source;
        cout<<"\n String in Target Before Copied :"<<target;
        strcpy(target,source);
        cout<<"\n String in Target After strcpy function Executed :"<<target;
        return 0;
}
```

Output:

```
        String in Source Before Copied :Computer Science
        String in Target Before Copied :target
        String in Target After strcpy function Executed :Computer Science
```

11.4.3.2 strlen()

The **strlen()** takes a null terminated string as its argument and returns its length. The length does not include the null(\0) character.

General Form:

strlen(string)

Program 11.7

```
#include <string.h>
#include <iostream>
using namespace std;
int main()
{
        char source[ ] = "Computer Science";
        cout<<"\n Given String is "<<source<<" its Length is "<<strlen(source);
        return 0;
}
```

Output:

```
        Given String is Computer Science its Length is 16
```

11.4.3.3 strcmp()

The **strcmp()** function takes two arguments: string1 and string2. It compares the contents of string1 and string2 lexicographically.

General Form:

strcpy(String1, String2)

The strcmp() function returns a:

- Positive value if the first differing character in string1 is greater than the corresponding character in string2. (ASCII values are compared)
- Negative value if the first differing character in string1 is less than the corresponding character in string2.
- 0 if string1 and string2 are equal.

Program 11.8

```
#include <string.h>
#include <iostream>
using namespace std;
int main()
{
        char string1[] = "Computer";
        char string2[] = "Science";
        int result;
        result = strcmp(string1,string2);
        if(result==0)
        {
        cout<<"String1 : "<<string1<<" and String2 : "<<string2 <<"Are Equal";
        }
        if (result<0)
        {
        cout<<"String1 :"<<string1<<" and String2 : "<<string2 <<" Are Not Equal";
        }
}
```

Output

```
        String1 : Computer and String2 : Science Are Not Equal
```

11.4.3.4 strcat()

The **strcat()** function takes two arguments: target and source. This function appends copy of the character string pointed by the source to the end of string pointed by the target.

General Form:

strcat(Target, source)

Program 11.9

```
#include <string.h>
#include <iostream>
using namespace std;
int main()
{
        char target[50] = "Learning C++ is fun";
        char source[50] = " , easy and Very useful";
        strcat(target, source);
        cout << target ;
        return 0;
}
```

Output

```
        Learning C++ is fun , easy and Very useful
```

11.4.3.5 strupr()

The **strupr()** function is used to convert the given string into Uppercase letters.

General Form:

strcat(string)

Program 11.10

```
using namespace std;
#include<iostream>
#include<ctype.h>
#include<string.h>
int main()
{
        char str1[50];
        cout<<"\nType any string in Lower case :";
        gets(str1);
        cout<<"\n Converted the Source string “<<str1<<into Upper Case is "<<strupr(str1);
        return 0;
}
```

Output:

```
        Type any string in Lower case : computer science
Converted the Source string computer science into Upper Case is COMPUTER SCIENCE
```

11.4.3.6 strlwr()

The **strlwr()** function is used to convert the given string into Lowercase letters.

General Form:

strlwr(string)

Program 11.11

```
using namespace std;
#include<iostream>
#include<ctype.h>
#include<string.h>
int main()
{
        char str1[50];
        cout<<"\nType any string in Upper case :";
        gets(str1);
        cout<<"\n Converted the Source string “<<str1<<into Lower Case is "<<strlwr(str1);
}
```

Output:

```
        Type any string in Upper case : COMPUTER SCIENCE
Converted the Source string COMPUTER SCIENCE into lower Case is computer science
```

11.4.4 Mathematical functions (math.h)

Most of the mathematical functions are defined in math.h header file which includes basic mathematical functions.

11.4.4.1 cos() function

The cos() function takes a single argument in radians. The cos() function returns the value in the range of [-1, 1]. The returned value is either in double, float, or long double.

Program 11.12

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
        double x = 0.5, result;
        result = cos(x);
        cout << "COS("<<x<<")= "<<result;
}
```

Output:

```
        COS(0.5)= 0.877583
```

11.4.4.2 sqrt() function

The **sqrt()** function returns the square root of the given value. The **sqrt()** function takes a single non-negative argument. If a **negative value** is passed as an argument to **sqrt()** function, a **domain error occurs**.

Program 11.13

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
        double x = 625, result;
        result = sqrt(x);
        cout << "sqrt("<<x<<") = "<<result;
        return 0;
}
```

Output:

```
        sqrt(625) = 25
```

11.4.4.3 sin() function

The **sin()** function takes a single argument in radians. The **sin()** function returns the value in the range of [-1, 1]. The returned value is either in double, float, or long double.

11.4.4.4 pow() function

The **pow()** function returns base raised to the power of an exponent. If any argument passed to **pow()** is long double, the return type is promoted to long double. If not, the return type is double. The **pow()** function takes two arguments:

- **base** - the base value
- **exponent** - exponent of the base

Program 11.14

```
#include <iostream>
#include <math.h>
using namespace std;
int main ()
{
        double base, exponent, result;
        base = 5;
        exponent = 4;
        result = pow(base, exponent);
        cout << "pow("<<base << "^" << exponent << ") = " << result;
        double x = 25;;
        result = sin(x);
        cout << "\nsin("<<x<<")= "<<result;
        return 0;
}
```

Output:

```
        pow(5^4) = 625
        sin(25)= -0.132352
```

11.5 User-defined Functions

11.5.1 Introduction

We can also define new functions to perform a specific task. These are called as **user-defined functions**. User-defined functions are created by the user. A function can optionally define input parameters that enable callers to pass arguments into the function. A function can also optionally return a value as output. Functions are useful for encapsulating common operations in a single reusable block, ideally with a name that clearly describes what the function does.

11.5.2 Function Definition

In C++, a function must be defined before it is used anywhere in the program. The general syntax of a function definition is:

Return_Data_Type Function_name(parameter list)

{

Body of the function

}

Note:

1. The Return_Data_Type is any valid data type of C++.
2. The Function_name is a user-defined identifier.
3. The parameter list, which is optional, is a list of parameters, i.e. a list of variables preceded by data types and separated by commas.
4. The body of the function comprises C++ statements that are required to perform the intended task of this function.

11.5.3 Function Prototype

C++ program can contain any number of functions. But, it must always have **only one main() function** to begin the program execution. We can write the definitions of functions in any order as we wish. We can define the main() function first and all other functions after that or we can define all the needed functions prior to main(). Like a variable declaration, a function must be declared before it is used in the program. The declaration statement may be given outside the main() function.

long fact (int, double)

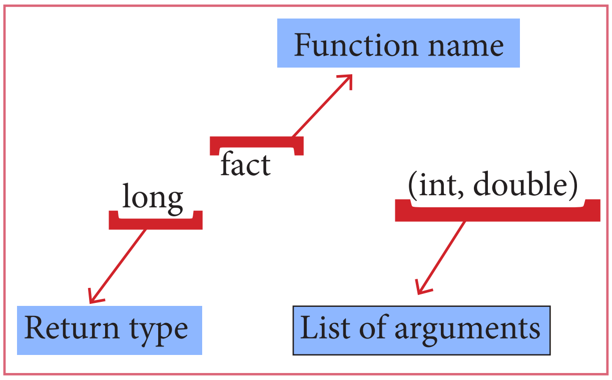


Figure 11.1

The prototype above provides the following information to the compiler:

- The return value of the function is of type long.
- **fact** is the name of the function.
- the function is called with two arguments:
  The first argument is of int **data** type.
  The second argument is of **double** data type.
  int display(int, int) // function prototype//

The above function prototype provides details about the return data type, name of the function and a list of formal parameters or arguments.

11.5.4 Use of void command

void type has two important purposes:

- To indicate the function does not return a value
- To declare a generic pointer.

Notes

void data type indicates the compiler that the function does not return a value, or in a larger context void indicates that it holds nothing.

For Example:

void fun(void)

The above function prototype tells compiler that the function **fun()** neither receives values from calling program nor return a value to the calling program.

11.5.5 Accessing a function

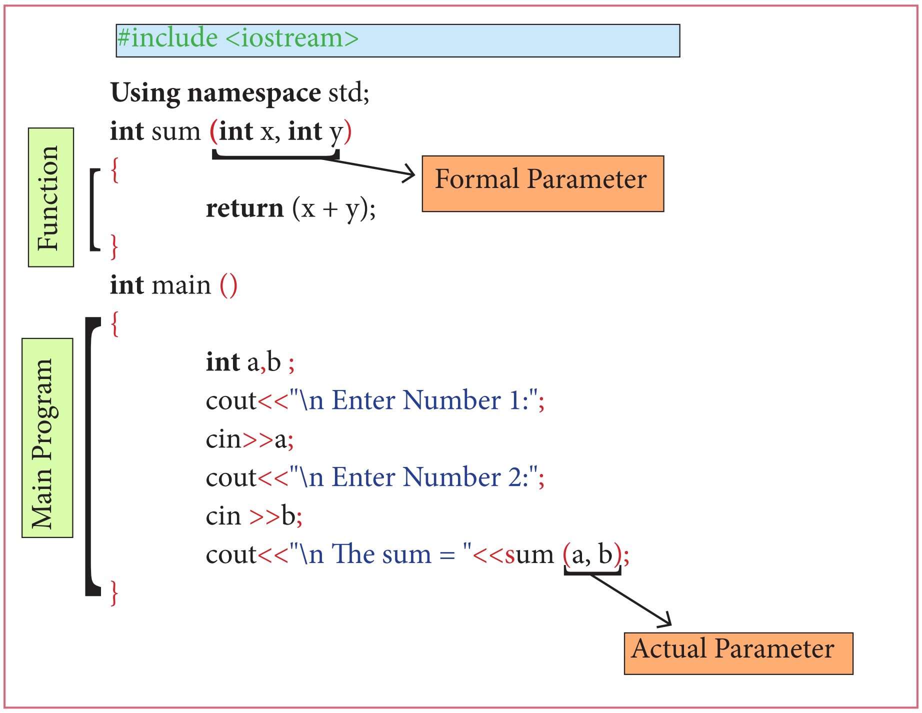
The user-defined function should be called explicitly using its name and the required arguments to be passed. The compiler refers to the function prototype to check whether the function has been called correctly. If the argument type does not match exactly with the data type defined in the prototype, the compiler will perform type conversion, if possible. If type conversion is impossible, the compiler generates an error message.

Example :

1	display()	calling the function without a return value and without any argument
2	display (x, y)	calling the function without a return value and with arguments
3	x = display()	calling the function with a return value and without any argument
4	x = display (x, y)	calling the function with a return value and with arguments

11.5.5.1 Formal Parameters and Actual Parameters or Arguments

Arguments or parameters are the means to pass values from the calling function to the called function. The variables used in the function definition as parameters are known as formal parameters. The constants, variables or expressions used in the function call are known as actual parameters.



```
#include <iostream>
Using namespace std;
int sum (int x, int y)
{
        return (x + y);
}
int main ()
{
        int a,b ;
        cout<<"\n Enter Number 1:";
        cin>>a;
        cout<<"\n Enter Number 2:";
        cin >>b;
        cout<<"\n The sum = "<<sum (a, b);
}
```

Figure 11.2 Formal and Actual Parameters

11.5.5.2 Default arguments

In C++, one can assign default values to the formal parameters of a function prototype. The Default arguments allows to omit some arguments when calling the function.

When calling a function,

- For any missing arguments, complier uses the values in default arguments for the called function.
- The default value is given in the form of variable initialization.

Example : **void defaultvalue(int n1=10, n2=100);**

- The default arguments facilitate the function call statement with partial or no arguments.

Example : **defaultvalue(x,y);**

defaultvalue(200,150);

defaultvalue(150);

defaultvalue(x,150);

- The default values can be included in the function prototype from right to left, i.e., we cannot have a default value for an argument in between the argument list.

Example : **void defaultvalue(int n1=10, n2);**//invalid prototype

void defaultvalue(int n1, n2 = 10);//valid prototype

11.5.5.3 Constant Arguments

The constant variable can be declared using **const** keyword. The **const** keyword makes variable value stable. The constant variable should be initialized while declaring. The **const** modifier enables to assign an initial value to a variable that cannot be changed later inside the body of the function.

Syntax :

<returntype><functionname> (const <datatype variable=value>)

Example:

- int minimum(const int a=10);
- float area(const float pi=3.14, int r=5);

Program 11.16

```
#include <iostream>
using namespace std;
double area(const double r,const double pi=3.14)
{
        return(pi*r*r);
}
int main ()
{
        double rad,res;
        cout<<"\nEnter Radius :";
        cin>>rad;
        res=area(rad);
        cout << "\nThe Area of Circle ="<<res;
        return 0;
}
```

Output:

```
Enter Radius :5
The Area of Circle =78.5
```

If the variable value “r” is changed as **r=25;** inside the body of the function **“area”** then compiler will throw an error as **“assignment of read-only parameter 'r'”**

```
double area(const double r,const double pi=3.14)
{
          r=25;
          return(pi*r*r);
}
```

11.6 Methods of calling functions

In C++, the arguments can be passed to a function in two ways. Based on the method of passing the arguments, the function calling methods can be classified as **Call by Value method** and **Call by Reference or Address method**.

11.6.1 Call by value Method

This method copies the value of an actual parameter into the formal parameter of the function. In this case, changes made to formal parameter within the function will have no effect on the actual parameter.

Program 11.17

```
#include<iostream>
using namespace std;
void display(int x)
{
          int a=x*x;
          cout<<"\n\nThe Value inside display function (a * a):"<<a;
}
int main()
{
            int a;
          cout<<”\nExample : Function call by value:”;
            cout<<"\n\nEnter the Value for A :";
            cin>>a;
          display(a);
          cout<<"\n\nThe Value inside main function "<<a;
          return(0);
}
```

Output :

```
Example : Function call by value
Enter the Value for A : 5
The Value inside display function (a * a) : 25
The Value inside main function 5
```

11.6.2 Call by reference or address Method

This method copies the address of the actual argument into the formal parameter. Since the address of the argument is passed ,any change made in the formal parameter will be reflected back in the actual parameter.

Program 11.18

```
#include<iostream>
using namespace std;
void display(int &x) //passing address of a//
{
           x=x*x;
          cout<<"\n\nThe Value inside display function (n1 x n1) :"<<x ;
 }
int main()
{
int n1;
cout<<"\nEnter the Value for N1 :";
cin>>n1;
cout<<"\nThe Value of N1 is inside main function Before passing : "<< n1;
display(n1);
cout<<"\nThe Value of N1 is inside main function After passing (n1 x n1)  : "<< n1; return(0);
}
```

Output :

```
Enter the Value for N1 :45
The Value of N1 is inside main function Before passing : 45
The Value inside display function (n1 x n1) :2025
The Value of N1 is inside main function After passing (n1 x n1)  : 2025
```

Note that the only change in the **display()** function is in the function header. The **&** symbol in the declaration of the parameter **x** means that the argument is a reference variable and hence the function will be called by passing reference. Hence when the argument **n1** is passed to the **display()** function, the variable **x** gets the address of **n1** so that the location will be shared. In other words, the variables **x** and **n1** refer to the same memory location. We use the name **n1** in the main() function, and the name **x** in the **display()** function to refer the same storage location. So, when we change the value of **x**, we are actually changing the value of **n1**.

11.6.3 Inline function

Normally the call statement to a function makes a compiler to jump to the functions (the definition of the functions are stored in STACKS) and also jump back to the instruction following the call statement. This reduces the speed of program execution. Inline functions can be used to reduce the overheads like STACKS for small function definition.

An **inline** function looks like normal function in the source file but inserts the function's code directly into the calling program. To make a function inline, one has to insert the keyword **inline** in the function header.

Syntax :

inline returntype functionname(datatype parameter 1, … datatype parameter n)

Advantages of inline functions:

- Inline functions execute faster but requires more memory space.
- Reduce the complexity of using STACKS.

Program 11.19

```
#include <iostream>
using namespace std;
inline int add (int a , int b)
{
        int c=a+b;
        return(c);
}
int main ()
{
        int x,y,z;
        cout<<"\nEnter the First Number :";
        cin>>x;
        cout<<"\nEnter the second Number    :";
        cin>>y;
        z=add(x,y);
        cout << "\n sum of "<<x<<"+"<<y<<"="<<z;
        return 0;
}
```

Output:

```
Enter the First Number :10
Enter the second Number    :20
sum of 10+20=30
```

Though the above program is written in the normal function definition format during compilation the function code **a+b** will be directly inserted in the calling statement i.e. **z=add(x,y);** this makes the calling statement to change as **z = a+b;**

11.7 Different forms of User-defined Function declarations

11.7.1 A Function without return value and without parameter

The following program is an example for a function with no return and no arguments passed .

The name of the function is **display()**, its return data type is void and it does not have any argument.

Program 11.20

```
#include<iostream>
using namespace std;
void display()
{         cout<<"First C++ Program with Function"; }
int main()
{         display(); // Function calling statement//
          return(0);
}
```

Output :

```
          First C++ Program with Function
```

11.7.2 A Function with return value and without parameter

The name of the function is **display()**, its return type is int and it does not have any argument. The **return** statement returns a value to the calling function and transfers the program control back to the calling statement.

Program 11.21

```
#include<iostream>
using namespace std;
int display()
{
          int a=10, b=5, s;
          s=a+b;
          return s;
}
int main()
{         int m=display();
          cout<<"\nThe Sum="<<m;
          return(0);
}
```

Output :

```
          The Sum=15
```

11.7.3 A Function without return value and with parameter

The name of the function is **display()**, its return type is void and it has two parameters or arguments **x** and **y** to receive two values. The **return** statement returns the control back to the calling statement.

Program 11 .22

```
#include<iostream>
using namespace std;
void display(int x, int y)
{
        int s=x+y;
        cout<<"The Sum of Passed Values: "<<s;
}
int main()
{
        int a=50,b=45;
          display(a,b);
        return(0);
}
```

Output :

```
The Sum of Passed Values: 95
```

11.7.4 A Function with return value and with parameter

The name of the function is display(), its return type is int and it has two parameters or arguments **x** and **y** to receive two values. The return statement returns the control back to the calling statement.

Program 11.23

```
#include<iostream>
using namespace std;
int display(int x, int y)
{
        int s=x+y;
        return s;
}
int main()
{
        int a=45,b=20;
        int s=display(a,b);
        cout<<"\nExample:Function with Return Value and  with Arguments";
        cout<<"\nThe Sum of Passed Values: "<<s;
        return(0);
}
```

Output :

```
Example: Function with Return Value and  with Arguments
The Sum of Passed Values: 65
```

11.8 Returning from function

Returning from the function is done by using the **return** statement.

The **return** statement stops execution and returns to the calling function. When a **return** statement is executed, the function is terminated immediately at that point.

11.8.1 The return statement

The **return** statement is used to return from a function. It is categorized as a jump statement because it terminates the execution of the function and transfer the control to the called statement. A **return** may or may not have a value associated with it. If return has a value associated with it, that value becomes the return value for the calling statement. Even for void function return statement without parameter can be used to terminate the function.

Syntax:

return expression/variable;

Example : return(a+b); return(a);

return; // to terminate the function

11.8.2 Returning values:

The functions that return no value is declared as void. The data type of a function is treated as **int**, if no data type is explicitly mentioned. For example,

For Example :

int add (int, int);

add (int, int);

In both prototypes, the return value is int, because by default the return value of a function in C++ is of type **int** when no return value is explicitly given. Look at the following examples:

Sl.No	Function Prototype	Return type
1	int sum(int, float)	int
2	float area(float, float)	float
3	char result()	char
4	double fact(int n)	double

Returning Non-integer values

A string can also be returned to a calling statement.

Program 11.24

```
#include<iostream>
#include<string.h>
using namespace std;
char *display()
{         return ("chennai");   }
int main()
{
          char s[50];
          strcpy(s,display());
          cout<<"\nExample:Function with Non Integer Return"<<s;
          return(0);}
```

Output :

```
Example: Function with Non Integer Return Chennai
```

11.9 Recursive Function

A function that calls itself is known as recursive function. And, this technique is known as recursion.

Example 1: Factorial of a Number Using Recursion

Program 11.25

```
#include <iostream>
using namespace std;
int factorial(int); // Function prototype //
int main()
{
          int no;
          cout<<"\nEnter a number to find its factorial: ";
          cin >> no;
          cout << "\nFactorial of Number " << no <<" = " << factorial(no);
          return 0;
}
int factorial(int m)
{
          if (m > 1)
          {
          return m*factorial(m-1);
          }
          else
          {
          return 1;
}
}
```

Output :

```
Enter a number to find its factorial: 5
Factorial of Number 5 = 120
```

Note: Function prototype is mandatory since the function factorial() is given after the main() function.

11.10 Scope Rules of Variables

Scope refers to the accessibility of a variable. There are four types of scopes in C++. They are: **Local scope, Function scope, File scope** and **Class scope**.

11.10.1 Introduction

A scope is a region or life of the variable and broadly speaking there are three places, where variables can be declared,

- Inside a block which is called local variables.
- Inside a function is called function variables.
- Outside of all functions which is called global variables.
- Inside a class is called class variable or data members.

11.10.2 Local Scope:

- A local variable is defined within a block. A block of code begins and ends with curly braces { }.
- The scope of a local variable is the block in which it is defined.
- A local variable cannot be accessed from outside the block of its declaration.
- A local variable is created upon entry into its block and destroyed upon exit.

11.10.3 Function Scope:

- The scope of variables declared within a function is extended to the function block, and all sub-blocks therein.
- The life time of a function scope variable, is the life time of the function block. The scope of formal parameters is function scope.

11.10.4 File Scope:

- A variable declared above all blocks and functions (including main ( )) has the scope of a file. The life time of a file scope variable is the life time of a program.
- The file scope variable is also called as **global variable**.

Program 11.26

```
//Demo to test all Scopes//
#include<iostream>
using namespace std;
int file_var=20;          //Declared within File - file scope variable
void add(int x)
{
        int m;            //Declaration of  variable m in add () - Function scope variable
        m=x+30+file_var;
        cout<<"\n The Sum = "<<m;
}
```

```
int main ( )
{
int a ;
a = 10;
if(a>b)
{
          int t;              // local to this if block - Local variable
          t=a+20;
 }
cout<<t;
add(a);
cout<<m;
cout<<"\nThe File Variable = "<<file_var;
return(0);
}
```

Error
In function 'int main()':
[Error] 't' was not declared in main()

On compilation the Program 11.28, the compiler prompts an error message: The variable **t** is not accessible. Because the life time of a local variable is the life time of a block in its state of execution.

[Error] 'm' was not declared in this scope

The variable **m is** not accessible. Because the life time of the function scope variable is the life time of a block in its state of execution.

11.10.5 Class Scope:

- A class is a new way of creating and implementing a user defined data type. Classes provide a method for packing together data of different types.

- Data members are the data variables that represent the features or properties of a class.

class student
{
  private :
int mark1, mark2, total;
};
	The class student contains mark1, mark2 and total are data variables. Its scope is within the class student only.

Note: The class scope will be discussed later in chapter **"Classes and Object"**.

11.10.6 Scope resolution operator

The scope operator reveals the hidden scope of a variable. The scope resolution operator (::) is used for the following purposes.

- To access a Global variable when there is a Local variable with same name. An example using Scope Resolution Operator.

Program 11.27

```
// Program to show that we can access a global variable
// using scope resolution operator :: when there is a local
// variable with same name //
#include<iostream>
using namespace std;
int x=45;  // Global Variable x
int main()
{
  int x = 10; // Local Variable x
  cout << "\nValue of global x is " << ::x;
  cout << "\nValue of local x is " << x;
  return 0;
}
```

Output:

```
Value of global x is 45
Value of local x is 10
```

Points to Remember:

- A large program can typically be split into smaller sized blocks called as functions.
- Functions can be classified into Pre-defined or Built-in or Library Functions and User-defined Functions.
- User-defined functions are created by the user.
- The void function tells the compiler that the function returns nothing.
- The return statement returns a value to the calling function and transfers the program control back to the calling function.
- The default return type of a function in C++ is of type int.
- A function that calls itself is known as recursive function.
- Scope refers to the accessibility of a variable.
- There are four types of Scopes. They are: Local scope, Function scope, File scope and Class scope.
- The scope operator (::) reveals the hidden scope of a variable.

Hands on practice:

Write C++ program to solve the following problems :

1. Program that reads two strings and appends the first string to the second. For example, if the first string is entered as Tamil and second string as nadu, the program should print Tamilnadu. Use string library header.
2. Program that reads a string and converts it to uppercase. Include required header files.
3. Program that checks whether a given character is an alphabet or not. If it is an alphabet, whether it is lowercase character or uppercase character? Include required header files.
4. Write definition for a function sumseries ( ) in c++ with two arguments/ parameters - double x and int n. The function should return a value of type double and it should perform sum of the following series:

   x-x2 /3! + x3 / 5! - x4 / 7! + x5 / 9! -... upto n terms.
5. Program that invokes a function calc ( ) which intakes two integers and an arithmetic operator and prints the corresponding result.

Evaluation

FAXQR8

SECTION – A

Choose the correct answer

1. Which of the following header file defines the standard I/O predefined functions ?

   A) stdio.h B) math.h C) string.h D) ctype.h
2. Which function is used to check whether a character is alphanumeric or not.

   A) isalpha() B) isdigit() C) isalnum() D) islower()
3. Which function begins the program execution ?

   A) isalpha() B) isdigit() C) main() D) islower()
4. Which of the following function is with a return value and without any argument ?

   A) x=display(int, int) B) x=display() C) y=display(float) D) display(int)
5. Which is return data type of the function prototype of add(int, int); ?

   A) int B) float C) char D) double
6. Which of the following is the scope operator ?

   A) > B) & C) % D) ::

SECTION-B

Very Short Answers

1. Define Functions.
2. Write about strlen() function.
3. What are importance of void data type.
4. What is Parameter and list its types?
5. Write a note on Local Scope.

SECTION-C

Short Answers

1. What is Built-in functions ?
2. What is the difference between isupper() and toupper() functions ?
3. Write about strcmp() function.
4. Write short note on pow() function in C++.
5. What are the information the prototype provides to the compiler ?
6. What is default arguments ? Give example.

SECTION - D

Explain in detail

1. Explain Call by value method with suitable example.
2. What is Recursion? Write a program to find the factorial of the given number using recursion.
3. What are the different forms of function return? Explain with example.
4. Explain scope of variable with example.
5. Write a program to accept any integer number and reverse it.

Unit III Introduction of C++

CHAPTER 12

Arrays and Structures

Learning Objectives

After learning this chapter, the students will be able to

FB7LSV

- Know the structured data type using arrays.
- Know the types of arrays.
- Writing programs to manuplates different types of arrays.

12.1 Introduction

The variables are used to store data. These variables are the one of the basic building blocks in C++. A single variable is used to store a single value that can be used anywhere in the memory. In some situations, we need to store multiple values of the same type. In that case, it needs multiple variables of the same data type. All the values are stored randomly anywhere in the memory.

For example, to store the roll numbers of the 100 students, it needs 100 variables named as roll1, roll2, roll3,……. roll100 . It becomes very difficult to declare 100 variables and store all the roll numbers. In C++, the concept of Array helps to store multiple values in a single variable. Literally, the meaning of **Array is "More than one". In other words, array is an easy way of storing multiple values of the same type referenced by a common name". An array is also a derived data type in C++.**

"An array is a collection of variables of the same type that are referenced by a common name". In an array, the values are stored in a fixed number of elements of the same type sequentially in memory. Therefore, an integer array holds a sequence of integers; a character array holds a sequence of characters, and so on. The size of the array is referred to as its dimension.

12.2 Types of Arrays:

There are different types of arrays used in C++. They are:

- One-dimensional arrays
- Two-dimensional arrays
- Multi-dimensional arrays

12.2.1 One-dimensional array

This is the simplest form of an array. A one dimensional array represents values that are stored in a single row or in a single column.

Declaration

Syntax:

<data type><array_name> [<array_size>];

data_type declares the basic type of the array, which is the type of each element in the array.

array_name specifies the name with which the array will be referenced.

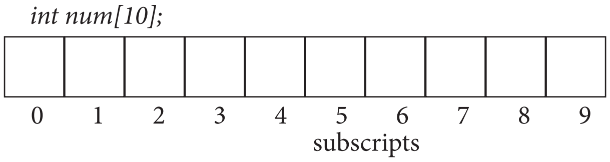
array_size defines how many elements the array will hold. Size should be specified with square brackets [ ].

Example:

```
int num[10];
```

In the above declaration, an array named “num” is declared with 10 elements (memory space to store 10 different values) as integer type.

For the above declaration, the compiler allocates 10 memory locations (boxes) referenced by a common name “num” as given below



Each element (Memory box) has a unique index number starting from 0 which is known as “subscript”. The subscript always starts with 0 and it should be an unsigned integer value. Each element of an array is referred by its name with subscript index within the square bracket. For example, num[3] refers to the 4th element in the array.

Some more array declarations with various data types:

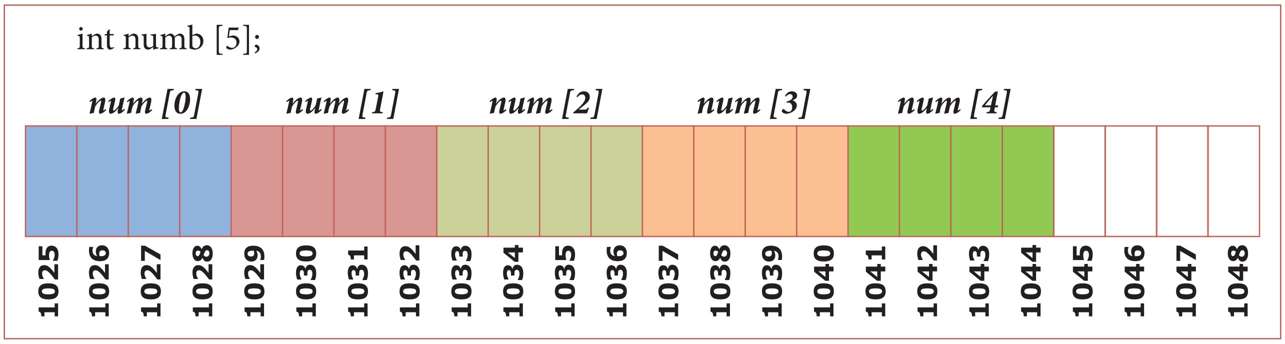
char emp_name[25]; // character array named emp_name with size 25

float salary[20]; // floating-point array named salary with size 20

int a[5], b[10], c[15]; // multiple arrays are declared of type int

Memory representation of an one dimensional array

The amount of storage required to hold an array is directly related with type and size. The following figure shows the memory allocation of an array with five elements.



The above figure clearly shows that, the array num is an integer array with 5 elements. As per the Dev-C++ compiler, 4 bytes are allocated for every int type variable. Here, there are totally 5 elements in the array, where for each element, 4 bytes will be allocated. Totally, 20 bytes will be allocated for this array.

Datatype	Turbo C++	Dev C++
char	1	1
int	2	4
float	4	4
long	4	4
double	8	8
long double	10	10

The memory space allocated for an array can be calculated using the following formula:

Number of bytes allocated for type of array × Number of elements

Initialization

An array can be initialized at the time of its declaration. Unless an array is initialized, all the array elements contain garbage values.

Syntax:

<datatype> <array_name> [size] = {value-1,value-2,…………… ,value-n};

Example

int age[5]={19,21,16,1,50};

In the above example, the array name is 'age' whose size is 5. In this case, the first element 19 is stored in age[0], the second element 21 is stored in age[1] and so on as shown in figure 12.1

int age [5]={19,21,16,1,50};

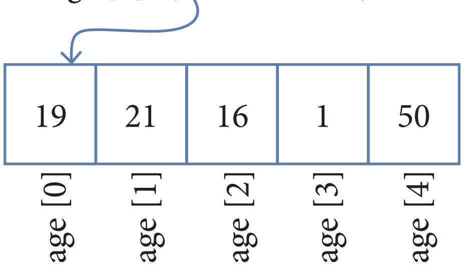


Figure 12.1

While declaring and initializing values in an array, the values should be given within the curly braces ie. { ….. }

The size of an array may be optional when the array is initialized during declaration.

Example:

int age[]={ 19,21,16,1,50};

In the above initialization, the size of the array is not specified directly in the declaration with initialization. So, the size is determined by compiler which depends on the total number of values. In this case, the size of the array is five.

More examples of array initialization:

float x[5] = {5.6, 5.7, 5.8, 5.9, 6.1};

char   vowel[6] = {'a', 'e', 'i', 'o', 'u', '\0'};

Accepting values to an array during run time :

Multiple assignment statements are required to insert values to the cells of the array during runtime. The for loop is ideally suited for iterating through the array elements.

// Input values while execution

```
#include <iostream>
using namespace std;
int main()
{
        int num[5];
        for(int i=0; i<5; i++)
        {
                cout<< "\n Enter value " << i+1 << "= ";
                cin>>num[i];
        }
}
```

In the above program, a for loop has been constructed to execute the statements within the loop for 5 times. During each iteration of the loop, *cout* statement prompts you to "Enter value ……" and *cin* gets the value and stores it in num[i];

The following table shows the execution of the above code block.

Iteration	i <5	cout << "\n Enter value " << i+1 << "= ";	cin>>num [i];	Received value stored in memory		i++ (i=i+1)
1	5 > 0 (T)	Enter value 1 =	num[0] = 5	num[0]	5	1
2	5 > 1 (T)	Enter value 2 =	num[1] = 10	num[1]	10	2
3	5 > 2 (T)	Enter value 3 =	num[2] = 15	num[2]	15	3
4	5 > 3 (T)	Enter value 4 =	num[3] = 20	num[3]	20	4
5	5 > 4 (T)	Enter value 4 =	num[25 = [4	num[4]	25	5
6	5 > 5 (F)	Exit from Loop				

Note

In for loop, the index ***i*** is declared with an initial value 0 (zero). Since in most of the cases, the initial value of the loop index will be used as the array subscript representation.

Accessing array elements

Array elements can be used anywhere in a program as we do in case of a normal variable. The elements of an array are accessed with the array name followed by the subscript index within the square bracket.

Example:

```
cout<<num[3];
```

In the above statement, num[3] refers to the 4th element of the array and *cout* statement displays the value of num[3].

Note

The subscript in bracket can be a variable, a constant or an expression that evaluates to an integer.

// Accessing array elements

```
#include <iostream>
using namespace std;
int main()
{
        int num[5] = {10, 20, 30, 40, 50};
        int t=2;
        cout<<num[2] <<endl; // S1
        cout<<num[3+1] <<endl; // S2
        cout<<num[t=t+1]; // S3
}
```

output:

```
30
50
40
```

In the above program, statement **S1** displays the value of the 3rd element (subscript index 2). **S2** will display the value of the 5th element (ie. Subscript value is 3+1 = 4). In the same way statement **S3** will display the value of the 4th element.

C++ program to inputs 10 values and count the number of odd and even numbers

```
#include <iostream>
using namespace std;
int main()
{
        int num[10], even=0, odd=0;
        for (int i=0; i<10; i++)
         {
                cout<< "\n Enter Number " << i+1 <<"= ";
                cin>>num[i];
                if (num[i] % 2 == 0)
                 ++even;
                else
                 ++odd;
         }
         cout << "\n There are "<< even <<" Even Numbers";
         cout << "\n There are "<< odd <<" Odd Numbers";
}
```

Output:

```
Enter Number 1= 78
Enter Number 2= 51
Enter Number 3= 32
Enter Number 4= 66
Enter Number 5= 41
Enter Number 6= 68
Enter Number 7= 27
Enter Number 8= 65
Enter Number 9= 28
Enter Number 10= 94
There are 6 Even Numbers
There are 4 Odd Numbers
```

(HOTS : Rewrite the above program using the conditional operator instead of if)

Searching in a one dimensional array:

Searching is a process of finding a particular value present in a given set of numbers. The linear search or sequential search compares each element of the list with the value that has to be searched until all the elements in the array have been traversed and compared.

Program for Linear Search

```
#include <iostream>
using namespace std;
int main()
{
          int num[10], val, id=-1;
          for (int i=0; i<10; i++)
          {
          cout<< "\n Enter value " << i+1 <<"= ";
          cin>>num[i];
          }
          cout<< "\n Enter a value to be searched: ";
          cin>>val;
      for (int i=0; i<size; i++)
      {
              if (arr[i] == value)
               {    id= i;
                    break;
               }
      }
          if(id==-1)
          cout<< "\n Given value is not found in the array..";
          else
          cout<< "\n The value is found at the position" << id+1;
          return 0;
}
```

The above program reads an array and prompts for the values to be searched. It compares each element of the list with the value that has to be searched until all the elements in the array have been traversed and compared.

Strings

A string is defined as a sequence of characters where each character may be a letter, number or a symbol. Each element occupies one byte of memory. Every string is terminated by a null ('\0', ASCII code 0) character which must be appended at the end of the string. In C++, there is no basic data type to represent a string. Instead, it implements a string as an one-dimensional character array. When declaring a character array, it also has to hold a null character at the end, and so, the size of the character array should be one character longer than the length of the string.

Character Array (String) creation

To create any kind of array, the size (length) of the array must be known in advance, so that the memory locations can be allocated according to the size of the array. Once an array is created, its length is fixed and cannot be changed during run time. This is shown in figure12.2

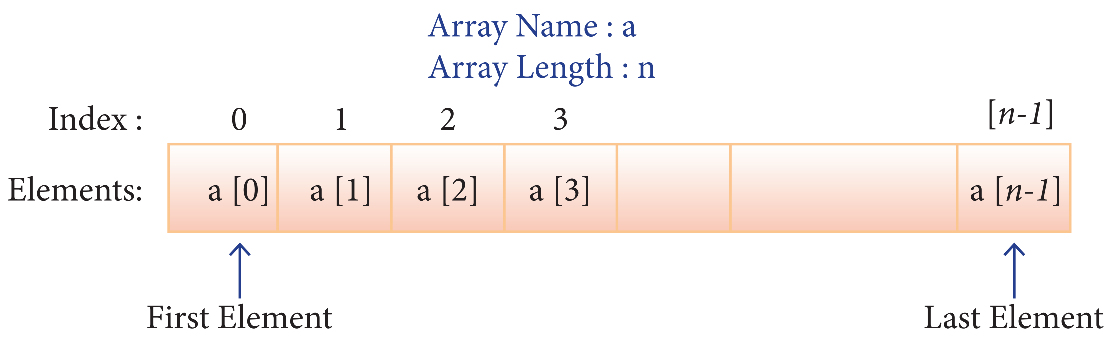


Figure 12.2

Syntax

Array declaration is:

char array_name[size];

In the above declaration, the size of the array must be an unsigned integer value.

For example,

char country[6];

Here, the array reserves 6 bytes of memory for storing a sequence of characters. The length of the string cannot be more than 5 characters and one location is reserved for the null character at the end.

//Program to demonstrate a character array.

```
#include <iostream>
using namespace std;
int main()
 {
          char country[6];
          cout<< "Enter the name of the country: ";
          cin>>country;
          cout<<" The name of the country is "<<country;
}
```

OUTPUT

```
Enter country the name: INDIA
The country name is INDIA
```

Initialization

The character array can be initialized at the time of its declaration. The syntax is shown below:

char array_name[size]={ list of characters separated by comma or a string } ;

For example,

char country[6]="INDIA";

In the above example, the text "INDIA" has 5 letters which is assigned as initial value to array country. The text is enclosed within double quotes. The memory representation is shown in Figure 13.3

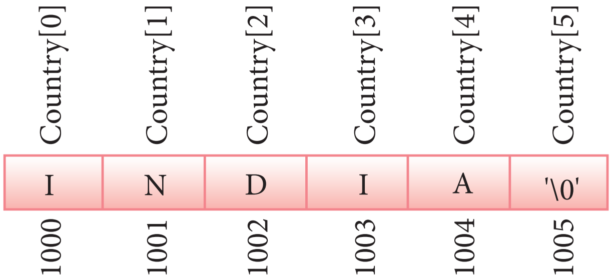


Figure 12.3

In the above memory representation, each character occupies one byte in memory. At the end of the string, a null character is automatically added by the compiler. **C++ also provides other ways of initializing the character array:**

char country[6]={'I', 'N', 'D', 'I', 'A', '\0'};

char country[]="INDIA";

char country[]={'I', 'N', 'D', 'I', 'A', '\0'};

If the size of the array is not explicitly mentioned, the compiler automatically calculate the size of the array based on the number of elements in the list and allocates space accordingly.

In the initialization of the string, if all the characters are not initialized, then the rest of the characters will be filled with NULL.

Example:

char str[5]={'5','+','A'};

str[0]; ---> 5

str[1]; ---> +

str[2]; ---> A

str[3]; ---> NULL

str[4]; ---> NULL

Note

During initialization, the array of elements cannot be initialized more than its size.

For example

char str[2]={'5','+','A','B'}; // Invalid

In the above example, the compiler displays "initialize-string for array of chars is too long" error message.

Write a Program to check palindrome or not

```
#include<iostream>
using namespace std;
int main( )
{
          int i, j, len, flag =1;
          char a [20];
          cout<<"Enter a string:";
          cin>>a;
          for(len=0;a[len]!='\0';++len)
          for(!=0,j=len-1;i<len/2;++i,--j)
          {
          if(a[j]!=a[i])
                    flag=0;
          }
```

```
        if(flag==1)
                cout<<"\n The String is palindrome";
        else
                cout<<"\n The String is not palindrome";
        return 0;
}
```

Output:

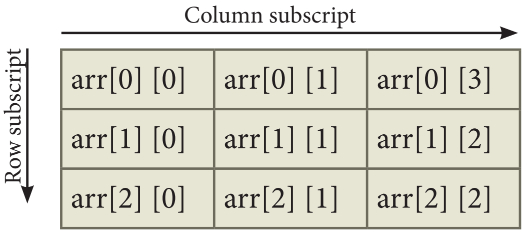
```
        Enter a string : madam
        The String is palindrome
```

12.3 Two-dimensional array

Two-dimensional (2D) arrays are collection of similar elements where the elements are stored in certain number of rows and columns. An example m × n matrix where m denotes the number of rows and n denotes the number of columns is shown in Figure12.4

int arr[3][3];

2D array conceptual memory representation



arr[0] [0]	arr[0] [1]	arr[0] [3]
arr[1] [0]	arr[1] [1]	arr[1] [2]
arr[2] [0]	arr[2] [1]	arr[2] [2]

The array arr can be coneptually viewed in matrix form with 3 rows and 3 coloumns. The point to be noted here is since the subscript starts with 0 arr [0][0] represents the first element.

Figure 12.4

12.3.1 Declaration of 2-D array

The declaration of a 2-D array is

data-type array_name[row-size][col-size];

In the above declaration, data-type refers to any valid C++ data-type, array_name refers to the name of the 2-D array, row-size refers to the number of rows and col-size refers to the number of columns in the 2-D array.

For example

int A[3][4];

In the above example, A is a 2-D array, 3 denotes the number of rows and 4 denotes the number of columns. This array can hold a maximum of 12 elements.

Note

Array size must be an unsigned integer value which is greater than 0. In arrays, column size is compulsory but row size is optional.

Other examples of 2-D array are:

```
int A[3][3];
float x[2][3];
char name[5][20];
```

12.3.2 Initialization of Two-Dimensional array

The array can be initialized in more than one way at the time of 2-D array declaration.

For example

```
int matrix[4][3]={
{10,20,30},// Initializes row 0
{40,50,60},// Initializes row 1
{70,80,90},// Initializes row 2
{100,110,120}// Initializes row 3
};
int matrix[4][3]={10,20,30,40,50,60,70,80,90,100,110,120};
```

Array's row size is optional but column size is compulsory.

For example

```
int matrix[][3]={

{10,20,30},// row 0

{40,50,60},// row 1

{70,80,90},// row 2

{100,110,120}// row 3

};
```

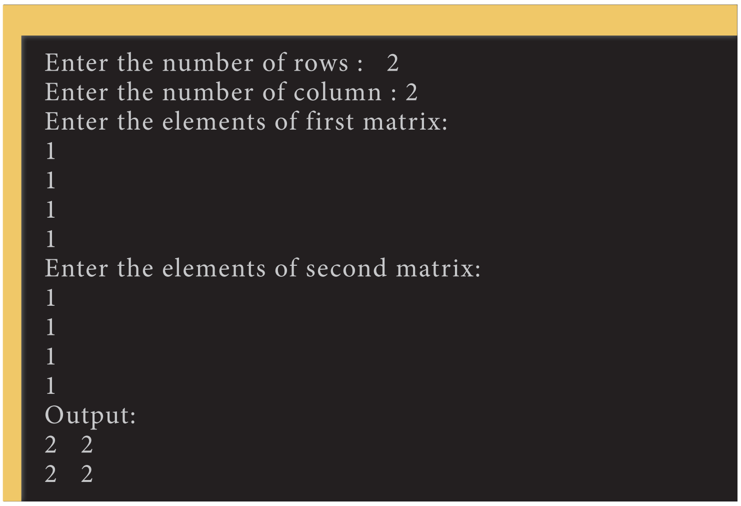
12.3.3 Accessing the two-dimensional array

Two-dimensional array uses two index values to access a particular element in it, where the first index specifies the row value and second index specifies the column value.

```
matrix[0][0]=10;// Assign 10 to the first element of the first row
matrix[0][1]=20;// Assign 20 to the second element of the first row
matrix[1][2]=60;// Assign 60 to the third element of the second row
matrix[3][0]=100;// Assign 100 to the first element of the fourth row
```

Write a program to perform addition of two matrices

```
#include<iostream>
#include<conio>
using namespace std;
int main()
{
        int row, col, m1[10][10], m2[10][10], sum[10][10];
        cout<<"Enter the number of rows : ";
        cin>>row;
        cout<<"Enter the number of columns : ";
        cin>>col;
        cout<< "Enter the elements of first matrix: "<<endl;
        for (int i = 0;i<row;i++ )
        for (int j = 0;j <col;j++ )
        cin>>m1[i][j];
        cout<< "Enter the elements of second matrix: "<<endl;
        for (int i = 0;i<row;i++ )
        for (int j = 0;j<col;j++ )
        cin>>m2[i][j];
        cout<<"Output: "<<endl;
        for (int i = 0;i<row;i++ )
        for (int j = 0;j<col;j++ )
        {
        sum[i][j]=m1[i][j]+m2[i][j];
        cout<<sum[i][j]<<" ";
        }
        cout<<endl<<endl;
        }
getch();
return 0;
}
```



```
Enter the number of rows :   2
Enter the number of column : 2
Enter the elements of first matrix:
1
1
1
1
Enter the elements of second matrix:
1
1
1
1
Output:
2   2
2   2
```

12.3.4 Memory representation of 2-D array

Normally, the two-dimensional array can be viewed as a matrix. The conceptual view of a 2-D array is shown below:

int A[4][3];

A[0][0]	A[0][1]	A[0][2]
A[1][0]	A[1][1]	A[1][2]
A[2][0]	A[2][1]	A[2][2]
A[3][0]	A[3][1]	A[3][2]

In the above example, the 2-D array name A has 4 rows and 3 columns.

Like one-dimensional, the 2-D array elements are stored in continuous memory.

There are two types of 2-D array memory representations. They are:

- Row-Major order
- Column-Major order

For example

int A[4][3]={ { 8,6,5}, { 2,1,9}, {3,6,4}, {4,3,2} }

Row Major order

In row-major order, all the elements are stored row by row in continuous memory locations, that is, all the elements in first row, then in the second row and so on. The memory representation of row major order is as shown below;

8	6	5	2	1	9	3	6	4	4	3	2
1000	1004	1008	1012	1016	1020	1024	1028	1032	1036	1040	1044
← Row 0 →			← Row 1 →			← Row 2 →			← Row 3 →		

Column Major order

8	2	3	4	6	1	6	3	5	9	4	2
1000	1004	1008	1012	1016	1020	1024	1028	1032	1036	1040	1044
← Col 0 →				← Col 1 →				← Col 2 →			

12.4 Array of strings

An array of strings is a two-dimensional character array. The size of the first index (rows) denotes the number of strings and the size of the second index (columns) denotes the maximum length of each string. Usually, array of strings are declared in such a way to accommodate the null character at the end of each string. For example, the 2-D array has the declaration:

char Name[6][10];

In the above declaration, the 2-D array has two indices which refer to the row size and column size, that is 6 refers to the number of rows and 10 refers to the number of columns.

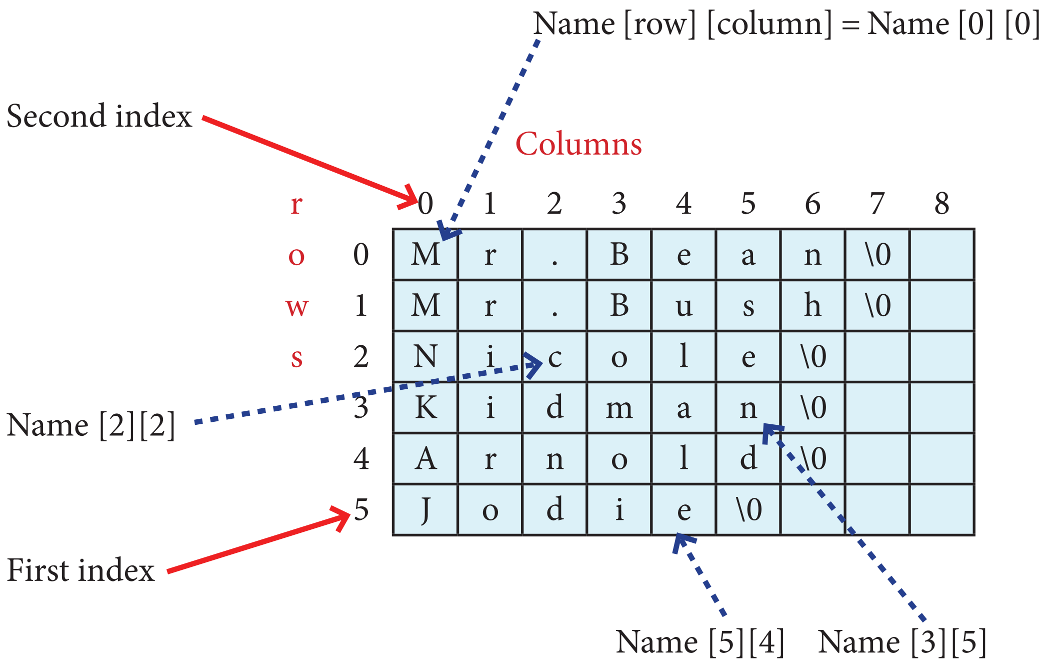
12.4.1 Initialization

For example

char Name[6][10] = {"Mr. Bean", "Mr.Bush", "Nicole", "Kidman", "Arnold", "Jodie"};

In the above example, the 2-D array is initialized with 6 strings, where each string is a maximum of 9 characters long, since the last character is null.

The memory arrangement of a 2-D array is shown below and all the strings are stored in continuous locations.



	0	1	2	3	4	5	6	7	8
0	M	r	.	B	e	a	n	\0	
1	M	r	.	B	u	s	h	\0	
2	N	i	c	o	l	e	\0		
3	K	i	d	m	a	n	\0		
4	A	r	n	o	l	d	\0		
5	J	o	d	i	e	\0			

C++ program to demonstrate array of strings using 2-D character array

```
#include<iostream>
using namespace std;
int main( )
{
        // initialize 2d array
        char colour [4][10]={"Blue","Red","Orange",
                           "yellow"};

        // printing strings stored in 2d array
        for (int i=0;  i <4; i++)
        cout  <<  colour [i] << "\n";
}
```

Output:

```
        Blue
        Red
        Orange
        Yellow
```

Case Study:

(1) Write a program to accept the marks of 10 students and find the average, maximum and minimum marks.

Structures

12.5 Structures Introduction

Structure is a user-defined which has the combination of data items with different data types. This allows to group variables of mixed data types together into a single unit.

12.5.1 Purpose of Structures

In any situation when more than one variable is required to represent objects of uniform data-types, array can be used. If the elements are of different data types,then array cannot support. If more than one variable is used, they can be stored in memory but not in adjacent locations. It increases the time consumption while searching. The structure provides a facility to store different data types as a part of the same logical element in one memory chunk adjacent to each other.

12.5.2 Declaring and defining structures

Structure is declared using the keyword 'struct'. The syntax of creating a structure is given below.

```
struct structure_name {
            type member_name1;
            type member_name2;
      } reference_name;
```

Objects declared along with structure definition are called global objects

An optional field reference_name can be used to declare objects of the structure type directly.

Example:

```
struct Student
{
      long rollno;
      int age;
      float weight;
} ;
```

In the above declaration of the struct, three variables rollno,age and weight are used. These variables(data element)within the structure are called members (or fields). In order to use the Student structure, a variable of type Student is declared and the memory allocation is shown in figure 12.5

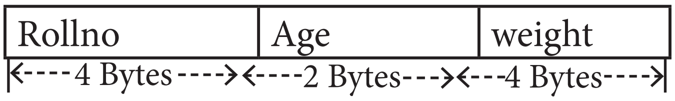


Fig 12.5 Memory Allocation

struct Student balu; // create a Student structure for Balu

This defines a variable of type Student named as Balu. Similar to normal variables, struct variable allocates memory for that variable itself. It is possible to define multiple variables of the same struct type:

struct Student frank; // create a structure for Student Frank.

For example, the structure objects balu and frank can also be declared as the structure data type as:

```
struct Student
{
longrollno;
int age;
float weight;
} balu, frank;
```

12.5.3 Referencing Structure Elements

Once the two objects of student structure type are declared (balu and frank),their members can be accessed directly. The syntax for that is using a dot (.) between the object name and the member name. For example, the elements of the structure Student can be accessed as follows:

```
balu.rollno
balu.age
balu.weight
frank.rollno
frank.age
frank.weight
```

(Anonymous Structure Vs Named Structure)

A structure without a name/tag is called anonymous structure.

```
struct
{
long rollno;
          int age;
          float weight;
} student;
```

The student can be referred as reference name to the above structure and the elements can be accessed like student.rollno, student.age and student.weight .

12.5.4 Initializing structure elements

Values can be assigned to structure elements similar to assigning values to variables.

Example

```
        balu.rollno= “702016”;
        balu.age= 18;
        balu.weight= 48.5;
```

Also, values can be assigned directly as similar to assigning values to Arrays.

balu={702016, 18, 48.5};

12.5.5 Structure Assignment

Structures can be assigned directly instead of assigning the values of elements individually.

Example

If Mahesh and Praveen are same age and same height and weight then the values of Mahesh can be copied to Praveen

```
struct Student
{
        int  age;
        float height, weight;
}mahesh;
```

Structure assignment is possible only if both structure variables/ objects are same type.

The age of Mahesh is 17 and the height and weights are 164.5 and 52.5 respectively.The following statement will perform the assignment.

```
mahesh = {17, 164.5, 52.5};
praveen =mahesh;
```

will assign the same age, height and weight to Praveen.

Examples:

The following C++ program reads student information through keyboard and displays the same

```
#include <iostream>
using namespace std;
struct Student
{
        int  age;
        float height, weight;
} mahesh;
void main( )
{
        cout<< " Enter the age:"<<endl;
        cin>>mahesh.age;
        cout<< "Enter the height:"<<endl;
        cin>>mahesh.height;
        cout<< "Enter the weight:"<<endl;
        cin>>mahesh.weight;
        cout<< "The values entered  for Age, height and weight are"<<endl;
        cout<<mahesh.age<<     "\t"<<mahesh.height<<     "\t"<<Mahesh.
weight;
}
```

Output:

```
Enter the age:
18
Enter the height:
160.5
Enter the weight:
46.5
The values entered  for Age, height and weight are
18      160.5          46.5
```

Points to Remember:

- Structure is a user-defined which has the combination of data items with different data types
- Structure is declared using the keyword 'struct'
- Structure elements are referenced using its object name followed by dot(.) operator and then the member name
- A structure without a name/tag is called anonymous structure.
- The structure elements can be initialized either by using separate assignment statements or at the time of declaration by surrounding its values with braces.
- A structure object can also be assigned to another structure object only if both the objects are of same structure type.
- The structure declared within another structure is called a nested structure
- A structure can contain array as its member element.
- Array of structure variable can also be created.

Evaluation

SECTION – A

FBQDH6

Choose the correct answer

1. Which of the following is the collection of variables of the same type that an referenced by a common name ?

   a) int b) float c) Array d) class

2. int age[]={6,90,20,18,2}; How many elements are there in this array?

   a) 2 b) 5 c) 6 d) 4

3. cin>>n[3]; To which element does this statement accept the value?

   a) 2 b) 3 c) 4 d) 5

4. By default, a string ends with which character?

   a)\o b) \t c) \n d) \b

5. Structure definition is terminated by

   (a) : (b) } (c) ; (d) ::

6. What will happen when the structure is declared?

   (a) it will not allocate any memory (b) it will allocate the memory

   (c) it will be declared and initialized (d) it will be only declared

7. A structure declaration is given below.

```
struct Time
{
int hours;
int minutes;
int seconds;
}t;
```

   Using above declaration which of the following refers to seconds.

   (a) Time.seconds (b) Time::seconds (c)seconds (d) t. seconds

8. Which of the following is a properly defined structure?

   (a) struct {int num;} (b) struct sum {int num;}

   (c) struct sum int sum; (d)struct sum {int num;};

9. A structure declaration is given below.

```
struct employee
{
int empno;
char ename[10];
}e[5];
```

Using above declaration which of the following statement is correct.

(a) cout<<e[0].empno<<e[0].ename; (b) cout<<e[0].empno<<ename;

(c)cout<<e[0]->empno<<e[0]->ename; (d) cout<<e.empno<<e.ename;

10. When accessing a structure member ,the identifier to the left of the dot operator is the name of

(a) structure variable (b) structure tag

(c) structure member (d) structure function

SECTION-B

Very Short Answers

1. What is Traversal in an Array?
2. What is Strings?
3. What is the syntax to declare two – dimensional array.
4. Define structure .What is its use?
5. What is the error in the following structure definition.

```
struct   employee{ inteno;charename[20];char dept;}
Employee e1,e2;
```

SECTION-C

Short Answers

1. Define an Array ? What are the types?
2. Write note an Array of strings.
3. The following code sums up the total of all students name starting with 'S' and display it. Fill in the blanks with required statements.

```
struct student {int exam no,lang,eng,phy,che,mat,csc,total;char name[15];};
int main()
{
student s[20];
for(int i=0;i<20;i++)
{       …………………….. //accept student details          }
for(int i=0;i<20;i++)
{
…………………….. //check for name starts with letter "S"
…………………….   // display the detail of the checked name
}
return 0;
}
```

4. How to access members of a structure?Give example.

5. What is called anonymous structure .Give an example

SECTION - D

Explain in detail

1. Write a C++ program to find the difference between two matrix.

2. Write a C++ program to add two distances using the following structure definition

```
struct Distance{
int feet;
float inch;
}d1 , d2, sum;
```

3. Write the output of the following c++ program

```
#include<iostream>
#include<stdio>
#include <string>
#include<conio>
using namespace std;
struct books {
char name[20], author[20];
} a[2];
int main()
{  cout<< "Details of Book No " << 1 << "\n";
cout<< "------------------------\n";
cout<< "Book Name :"<<strcpy(a[0].name,"Programming  ")<<endl;
cout<< "Book Author :"<<strcpy(a[0].author,"Dromy")<<endl;
cout<< "\nDetails of Book No " << 2 << "\n";
cout<< "------------------------\n";
cout<< "Book Name :"<<strcpy(a[1].name,"C++programming" )<<endl;
cout<< "Book Author :"<<strcpy(a[1].author,"BjarneStroustrup ")<<endl;
cout<<"\n\n";
cout<< "================================================\n";
cout<< " S.No\t| Book Name\t|author\n";
cout<< "====================================================";
for (int i = 0; i < 2; i++) {
```

```
cout<< "\n  " << i + 1 << "\t|" << a[i].name << "\t| " << a[i].author;
}
cout<< "\n=================================================";
return 0;
}
```

4. Write the output of the following c++ program

```
    #include <iostream>
    #include <string>
    using namespace std;
    struct student
    {
            introll_no;
            char name[10];
            long phone_number;
    };
    int main(){
    student p1 = {1,"Brown",123443},p2;
    p2.roll_no = 2;
    strcpy(p2.name ,"Sam");
    p2.phone_number = 1234567822;
    cout<< "First Student" <<endl;
    cout<< "roll no : " << p1.roll_no <<endl<< "name : " << p1.name <<endl;
    cout<< "phone no : " << p1.phone_number <<endl;
    cout<< "Second Student" <<endl;
    cout<< "roll no : " << p2.roll_no <<endl<< "name : " << p2.name <<endl;
    cout<< "phone no : " << p2.phone_number <<endl;
    return 0;
}
```

5. Debug the error in the following program

```
#include <istream.h>
structPersonRec
{
```

```
            charlastName[10];
            chaefirstName[10];
    int age;
    }
    PersonRecPeopleArrayType[10];
    void main()
    {
            PersonRecord people;
    for (i = 0; i < 10; i++)
    {
            cout<<people.firstName<< ‘ ‘ <<people.lastName  <<people.age;
    }
    for (int i = 0; i < 10; i++)
    {
            cout<< "Enter first name: ";            cin<<peop[i].firstName;
            cout<< "Enter last name: ";             cin>>peop[i].lastName;
            cout<< "Enter age: ";           cin>> people[i].age;}
}
```

References:

1. Object Oriented Programming with C++ (4th Edition), Dr. E. Balagurusamy, Mc.Graw Hills.
2. The Complete Reference C++ (Forth Edition), Herbert Schildt.Mc.Graw Hills.
3. Computer Science with C++ (A text book of CBSE XI and XII), SumitaArora, DhanpatRai& Co.
4. The C++ Programming Language, Bjarne Stroustrup
5. https://www.tutorialspoint.com
6. http://www.cs.princeton.edu
7. https://www.programiz.com

Unit IV | Object Oriented Programming with C++

CHAPTER 13

Introduction to Object Oriented Programming Techniques

FBZ9XT

Learing Objectives

After learning this chapter, the students will be able to

- Understand the concept of **OOPS**
- Know the difference between *Procedural*, *Modular* and *Object Oriented Programming*.
- Understand the advantages and disadvantages of Object Oriented Programming.

13.1 Introduction

Object-Oriented Programming (OOP) is the term used to describe a programming approach based on classes and objects. The object-oriented paradigm allows us to organize software as a collection of objects that consist of both data and behaviour. This is in contrast to conventional functional programming practice, that loosely connects data and behaviour.

Since 1980's the word **'object'** has appeared in relation to programming languages, with almost all languages developed since 1990 having object-oriented features. This chapter introduces general OOP concepts.

13.2 Programming Paradigms

Paradigm means organizing principle of a program. It is an approach to programming. There are different approaches available for problem solving using computer. They are Procedural programming, Modular Programming and Object Oriented Programming

13.2.1 Procedural programming

Procedural means a list of instructions were given to the computer to do something. Procedural programming aims more at procedures. This emphasis on doing things.

Important features of procedural programming

- Programs are organized in the form of subroutines or sub programs
- All data items are global
- Suitable for small sized software application
- Difficult to maintain and enhance the program code as any change in data type needs to be propagated to all subroutines that use the same data type. This is time consuming.
- Example: **FORTRAN** and **COBOL**.

13.2.2 Modular programming:-

Modular programming consist of a list of instructions that instructs the computer to do something. But this **Paradigm consists of multiple modules, each module has a set of functions of related types. Data is hidden under the modules.** Arrangement of data can be changed only by modifying the module

Important features of Modular programming

- Emphasis on algorithm rather than data
- Programs are divided into individual modules
- Each modules are independent of each other and have their own local data
- Modules can work with its own data as well as with the data passed to it.
- Example: **Pascal** and **C**

13.2.3 Object Oriented Programming:-

Object Oriented Programming paradigm emphasizes on the data rather than the algorithm. It implements programs using **classes** and **objects.**

Class: A Class is a construct in C++ which is used to bind data and its associated function together into a single unit using the encapsulation concept. Class is a user defined data type. Class represents a group of similar objects.

It can also be defined as a template or blueprint representing a group objects that share common properties and relationship.

Objects: Represents data and its associated function together into a single unit. Objects are the basic unit of OOP. Basically an object is created from a class. They are instances of class also called as class variables

An identifiable entity with some characteristics and behaviour is called object.

Important features of Object oriented programming

- Emphasizes on data rather than algorithm
- Data abstraction is introduced in addition to procedural abstraction
- Data and its associated operations are grouped in to single unit
- Programs are designed around the data being operated
- Relationships can be created between similar, yet distinct data types
- Example: **C++, Java, VB.Net, Python etc.**

13.3 Basic Concepts of OOP

The Object Oriented Programing has been developed to overcome the drawbacks of procedural and modular programming. It is widely accepted that object-oriented programming is the most important and powerful way of creating software.

The Object-Oriented Programming approach mainly encourages:

- **Modularisation:** where the program can be decomposed into **modules**.
- **Software re-use:** where a program can be composed from existing and new modules.

Main Features of Object Oriented Programming

- Data Abstraction
- Encapsulation

- Modularity
- Inheritance
- Polymorphism

13.3.1 Encapsulation

The mechanism by which the data and functions are bound together into a single unit is known as **Encapsulation.** It implements abstraction.

Encapsulation is about binding the data variables and functions together in class. It can also be called **data binding.**

Encapsulation is the most striking feature of a class. The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it. These functions provide the interface between the object's data and the program. **This encapsulation of data from direct access by the program is called data hiding or information hiding.**

13.3.2 Data Abstraction

Abstraction refers to showing only the essential features without revealing background details. Classes use the concept of abstraction to define a list of abstract attributes and function which operate on these attributes. They encapsulate all the essential properties of the object that are to be created. The attributes are called **data members** because they hold information. The functions that operate on these data are called **methods or member function.**

13.3.3 Modularity

Modularity is designing a system that is divided into a set of functional units (named modules) that can be composed into a larger application.

13.3.4 Inheritance

Inheritance is the technique of building new classes **(derived class)** from an existing Class **(base class)**. The most important advantage of inheritance is **code reusability.**

13.3.5 Polymorphism

Polymorphism is the ability of a message or function to be displayed in more than one form.

13.4 Advantages of OOP

Re-usability:

"Write once and use it multiple times" you can achieve this by using class.

Redundancy:

Inheritance is the good feature for data redundancy. If you need a same functionality in multiple class you can write a common class for the same functionality and inherit that class to sub class.

Easy Maintenance:

It is easy to maintain and modify existing code as new objects can be created with small differences to existing ones.

Security:

Using data hiding and abstraction only necessary data will be provided thus maintains the security of data.

13.5 Disadvantages of OOP

Size:

Object Oriented Programs are much larger than other programs.

Effort:

Object Oriented Programs require a lot of work to create.

Speed:

Object Oriented Programs are slower than other programs, because of their size.

Points to Remember:

- Paradigm means organizing principle of a program.It is an approach to programming.
- Procedural or Modular programming means a list of instructions were given and each instructions tell the computer to do something.
- Procedural programming aims more ate procedures. In this Programs are organized in the form of subroutines or sub programs
- Modular programming combines related procedures in a module and hides data under modules.
- Object Oriented programming Paradigm emphasizes on the data rather than the algorithm. It implements programs using classes and objects
- Class is a user defined data type. Class represents a group of similar objects.
- Objects are the basic unit of OOP.It represents data and associated function together in to a single unit.
- The mechanism by which the data and functions are bound together into a single unit is known as ENCAPSULATION. It implements abstraction .
- Abstraction refers to showing only the essential features without revealing background details
- Modularity is designing a system that is divided into a set of functional units that can be composed into a larger application.
- Polymorphism is the ability of a message or function to be displayed in more than one form.
- Inheritance is the technique of building new classes (derived class) from an existing class. The most important advantage of inheritance is code reusability.Inheritance is transitive in nature.

Evaluation

FC95ZG

SECTION – A

Choose the correct answer

1. The term is used to describe a programming approach based on classes and objects is
(A) OOP (B) POP (C) ADT (D) SOP

2. The paradigm which aims more at procedures.
(A) Object Oriented Programming (B)Procedural programming
(C) Modular programming (D)Structural programming

3. Which of the following is a user defined data type?
(A) class (B) float (C) int (D) object

4. The identifiable entity with some characteristics and behaviour is.
(A) class (B) object (C) structure (D) member

5. The mechanism by which the data and functions are bound together into a single unit is known as
(A) Inheritance (B) Encapsulation
(C) Polymorphism (D) Abstraction

6. Insulation of the data from direct access by the program is called as
(A) Data hiding (B) Encapsulation
(C) Polymorphism (D) Abstraction

7. Which of the following concept encapsulate all the essential properties of the object that are to be created?

   (A) class (B) Encapsulation

   (C) Polymorphism (D) Abstraction

8. Which of the following is the most important advantage of inheritance?

   (A) data hiding (B) code reusability

   (C) code modification (D) accessibility

9. “Write once and use it multiple time” can be achieved by

   (A) redundancy (B) reusability

   (C) modification (D) composition

10. Which of the following supports the transitive nature of data?

    (A) Inheritance (B) Encapsulation

    (C) Polymorphism (D) Abstraction

SECTION-B

Very Short Answers

1. How is modular programming different from procedural programming paradigm?
2. Differentiate classes and objects.
3. What is polymorphism?
4. How is encapsulation and abstraction are interrelated?
5. Write the disadvantages of OOP.

SECTION-C

Short Answers

1. What is paradigm ?Mention the different types of paradigm.
2. Write a note on the features of procedural programming.
3. List some of the features of modular programming
4. What do you mean by modularization and software reuse?
5. Define information hiding.

SECTION - D

Explain in detail

1. Write the differences between Object Oriented Programming and procedural programming
2. What are the advanatges of OOPs?

3 Write a note on the basic concepts that suppors OOPs?

Reference:

(1) Object Oriented Programming with C++ (4th Edition), Dr. E. Balagurusamy, Mc.Graw Hills.

(2) The Complete Reference C++ (Forth Edition), Herbert Schildt.Mc.Graw Hills.

(3) Computer Science with C++ (A text book of CBSE XI and XII), SumitaArora, DhanpatRai& Co.

(4) A text book of CBSE XI and XII computer science by PreetiArora and Pinky Gupta.

(5) Computer Science with C++ Reeta shoo and Gagansahoo

(6) The C++ Programming Language,BjarneStroustrup

(7) C++ Primer (5th Edition) by S. B. Lippman, J. Lajoie

Unit IV

Object Oriented Programming with C++

CHAPTER 14

Classes and objects

FCI224

Learining Objectives

After learning this chapter, the students will be able to

- Understand the purpose of classes, objects Constructors and Destructors
- able to construct C++ programs using classes with Constructors and Destructors
- Execute and debug class programs with Constructors and Destructors

14.1 Introduction to Classes

The most important feature of C++ is the “Class”. It is significance is highlighted by the fact that Bjarne Stroustrup initially gave the name 'C with classes'. C++ offers classes, which provide the four features commonly present in OOP languages: **Abstraction, Encapsulation**, **Inheritance**, and **Polymorphism**.

14.1.1 Need for Class

Class is a way to bind the data and its associated functions together. Classes are needed to represent real world entities that not only have data type properties but also have associated operations. It is used to create user defined data type

14.1.2 Declaration of a class

A class is defined in C++ using the keyword **class** followed by the name of the class. The body of the class is defined inside the curly brackets and terminated either by a semicolon or a list of declarations at the end.

Note

The only difference between structure and class is the members of structure are by default **public** where as it is **private in class**.

The General Form of a class definition

```
class class-name
{
private:
        variable declaration;
        function declaration;
protected:
        variable declaration;
        function declaration;
public:
        variable declaration;
        function declaration;
};
```

- The class body contains the declaration of its members (Data member and Member functions).

- The class body has three access specifiers (visibility labels) viz., private , public and protected.

14.1.3 Class Access Specifiers

Data hiding is one of the important features of Object Oriented Programming which allows preventing the functions of a program to access directly the internal representation of a class type. The access restriction to the class members is specified by public, private, and protected sections within the class body. The keywords public, private, and protected are called access specifiers. The default access specifier for members is private.

The Public Members

A public member is accessible from anywhere outside the class but within a program.You can set and get the value of public data members even without using any member function.

The Private Members

A private member cannot be accessed from outside the class. Only the class member functions can access private members.By default all the members of a class would be private.

The Protected Members

A protected member is very similar to a private member but it provides one additional benefit that they can be accessed in child classes which are called derived classes (inherited classes).

Example

```
      Keyword class intimates the compiler that it is a class definition
            Class name or tag name acts as a user defined data type. Using this,
            object of the same class type will be created.
class student
{
private:                                These are private access specifier members
        char name [10];               } That means these members cannot be accessed
        int rollno, mark1, mark2, total; } from outside
protected:                              These are protected access specifier members
        void accept();                } These members also cannot be accessed from
        void compute();               } outside
public:
        void display();  ------------->  Members under this specifier can be accessed
};                                        from outside
```

Note

If all members of the class are defined as private ,then the object of the class can not access anything from the class.

Activity 1

State the reason for the invalidity of the following code fragment

(i)	(ii)

class count
{
int first;
int second;
public:
int first;
};
	class item
	{
	int prd;
	};
	item int prdno;

14.1.4 Definition of class members

Class comprises of members. Members are classified as Data Members and Member functions. Data members are the data variables that represent the features or properties of a class. Member functions are the functions that perform specific tasks in a class. Member functions are called as methods, and data members are also called as attributes.

Example

```
Class result
{
Private;
          char name [10];
          int rollno,mark1, mark2, total;

Public:
          void accept();
          void display();
};
```

Data members

Member functions

Note

Classes also contain some special member functions called as constructors and destructors.

14.1.5 Defining methods of a class

Without defining the methods (functions), class definition will become incomplete. The member functions of a class can be defined in two ways.

(1) Inside the class definition

(2) Outside the class definition

(1) Inside the class definition

When a member function is defined inside a class, it behaves like inline functions. These are called Inline member functions.

Note

If a function is inline, the compiler places a copy of the code at each point where the function is called at compile time.

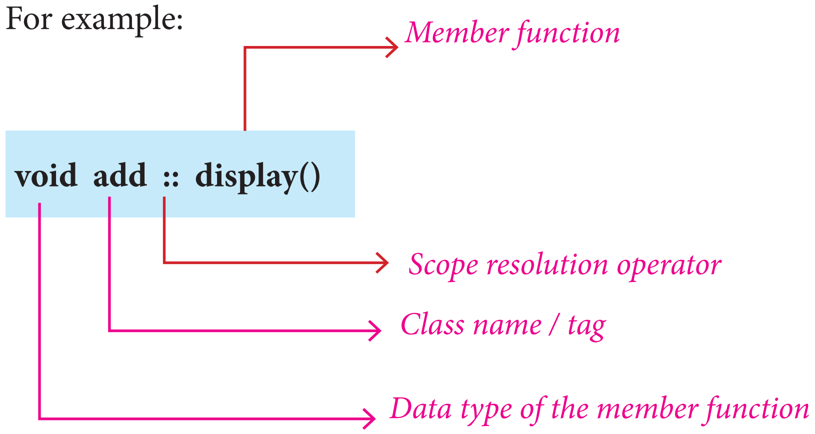
(2) Outside the class definition

When Member function defined outside the class just like normal function definition (Function definitions you are familiar with) then it is be called as **outline member function or non-inline member function. Scope resolution operator (::) is used for this purpose.** The syntax for defining the outline member function is

Syntax

```
return_type class_name :: function_name (parameter list)
{
        function definition
}
```

For example:



```
void  add  ::  display()
```

Illustration 14.1 Inline and Outline member function

Absence of access specifier means that members are private by default..

```
# include <iostream>
using namespace std;
class Box
{
        double width;          // no access specifier mentioned
public:
        double length;
        void printWidth( )          //inline member function definition
        {
                cout<<"\n The width of the box is..."<<width;
        }
        void  setWidth( double w );          //prototype of the function
};
void Box :: setWidth(double w)      // outline member function definition
{
        width=w;
}
int main( )
{
Box b;          // object for class Box
b.setWidth(10.0);      // Use member function to set the width.
b.printWidth( );               //Use member function to print the width.
return 0;
}
```

Output:

```
The width of the box is... 10
```

Note

Declaring a member function having looping construct, switch or goto statement as inline is not advisable.

14.2 Creating Objects

A class specification just defines the properties of a class. To make use of a class, the variables of that class type have to be declared. The class variables are called *object*. Objects are also called as *instance* of class.

For example

student s;

In the above statement **s** is an instance of the class **student**.

Objects can be created in two methods,

(1) Global object

(2) Local object

(1) Global Object

If an object is declared outside all the function bodies or by placing their names immediately after the closing brace of the class declaration then it is called as *Global object*. These objects can be used by any function in the program

(2) Local Object

If an object is declared with in a function then it is called *local object*. It cannot be accessed from outside the function.

Illustration 14.2 The use of local and global object

```
# include <iostream>
# include <conio>
using namespace std
class add
{
   int a,b;
   public:
     int sum;
   void getdata()
   {
     a=5;
     b=10;
     sum = a+b;
   }
} a1;                        //global object
add a2;                      //global object
int main()
{
add a3;          // Local object
a1.getdata();
a2.getdata();
a3.getdata();
cout<<a1.sum;
cout<<a2.sum;
cout<<a3.sum;
return 0;
}
```

Output:

```
151515
```

ACTIVITY 2

Identify the error in the following code fragment

```
class A
{
          float x;
          void init()
          {
          A a1;
          X1.5=1;
          }};
void main()
{     A1.init(); }
```

14.3 Memory allocation of objects

The member functions are created and placed in the memory space only when they are defined as a part of the class specification. Since all the objects belonging to that class use the same member function, **no separate space is allocated for member functions when the objects are created. Memory space required for the member variables are only allocated separately for each object** because the member variables will hold different data values for different objects

Illustration 14.3 Memory allocation for objects

```
# include <iostream>
using namespace std;

class product
{
        int code, quantity;
        float price;
        public:
        void assignData();
        void Print();
};
int main()
{
        product p1, p2;
        cout<<"\n Memory allocation for object p1 " <<sizeof(p1);
        cout<<"\n Memory allocation for object p2 " <<sizeof(p2);
        return 0;
}
```

Output:

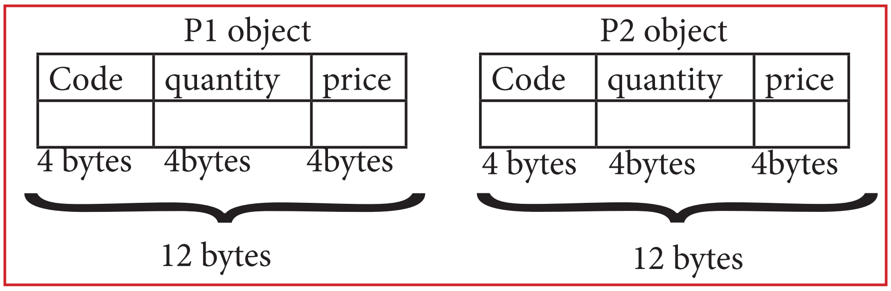
```
Memory allocation for object p1  12
Memory allocation for object p2  12
```

Member functions assignData( ) and Print( ) belong to the common pool in the sense both the objects p1 and p2 will have access to the code area of the common pool.

Note

The members will be allocated with memory space only after the creation of the class type object

Memory for Objects for p1 and p2 is illustrated:



ACTIVITY 3

What is the size of the objects s1, s2?

```
class sum
{
        int n1,n2;
        public:
        void add(){int n3=10;n1=n2=10;}
}  s1,s2;
```

14.4 Referencing class members

The members of a class are referenced (accessed) by using the object of the class followed by the dot (membership) operator and the name of the member.

The general syntax for calling the member function is:

Object_name . function_name(actual parameter);

For example consider the following illustration

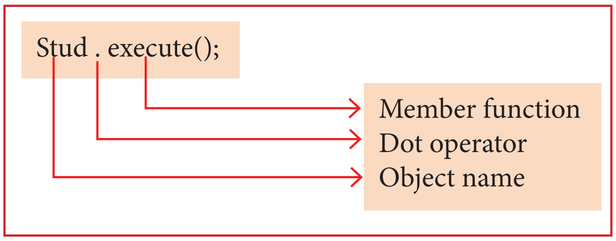


Illustration 14.4 C++ program to illustrate the communication of object:

```
#include<iostream>
using namespace std;
class compute
{
 int n1,n2;      //private by default
 public :
 int n;
 int add (int a, int b)   //inline member  function
 {
 int c=a+b;      //int c ; local variable for this function
 return c;
 }

}c1,c2;
int main()
{
 c1.n =c1.add(12,15); //member function is called
 c2.n =c2.add(8,4);
 cout<<"\n Sum of object-1  "<<c1.n;
 cout<<"\n Sum of object-2  "<<c2.n;
 cout<<"\n Sum of the two objects are  "<<c1.n+c2.n;
        return 0;
}
```

Output:

```
Sum of object-1  27
Sum of object-2  12
Sum of the two objects are  39
```

Note

Even an array of objects can be created for a class. It is declared and defined in the same way as any other type of array.

Example :

student s[10];

Where student is the class name and s[10] is 10 objects created for the student class.

14.5 Introduction to Constructors

The definition of a class only creates a new user defined data type. The instances of the class type should be instantiated (created and initialized) . Instantiating object is done using constructor.

14.5.1 Need for Constructors

An array or a structure in c++ can be initialized during the time of their declaration.

For example

```
struct sum
   {
          int n1,n2;
   };
class add
  {
          int num1,num2;
  };
int  main()
   {
          int arr[]={1,2,3};        //declaration and initialization of array
          sum s1={1,1};            //declaration and initialization of structure object
          add  a1={0,0};          // class object declaration and initialization throws
                                                                       compilation error
    }
```

Member function of a class can access all the members irrespective of their associated access specifier.

The initialization of class type object at the time of declaration similar to a structure or an array is not possible because the class members have their associated access specifiers (private or protected or public). Therefore Classes include special member functions called as *constructors*. The constructor function initializes the class object.

14.6 Declaration and Definition

When an instance of a class comes into scope, a special function called the *constructor* gets executed. The constructor function name has the same name as the class name. The constructors return nothing. They are not associated with any data type. It can be defined either inside class definition or outside the class definition.

Example 1:

Illustration 14.5 A constructor defined inside the class specification.

```
#include<iostream>
using namespace std;
class Sample
{
        int i,j;
        public :
          int k;
          Sample()
          {
             i=j=k=0;   //constructor defined inside the class
          }
};
```

14.6.1 Functions of constructor

As we know now that the constructor is a special initialization member function of a class that is called automatically whenever an instance of a class is declared or created. The main function of the constructor is

1) To allocate memory space to the object and

2) To initialize the data member of the class object

There is an alternate way to initialize the class objects but in that case we have to explicitly call the member function.

14.7 Types of constructors

There are different types of constructors.

- **Default Constructors**

A constructor that accepts no parameter is called default constructor. For example in the class Data, Data ::Data() is the default constructor . Using this constructor Objects are created similar to the way the variables of other data types are created. If a class does not contain an explicit constructor (user defined constructor) the compiler automatically generate a default constructor.

- **Parameterized Constructors**

A constructor which can take arguments is called parameterized constructor .This type of constructor helps to create objects with different initial values. This is achieved by passing parameters to the function.

Example :

```
Data :: Data(int,int);
```

- **Copy Constructors**

A constructor having a reference to an already existing object of its own class is called copy constructor. It is usually of the form Data (Data&), where Data is the class name.

A copy constructor can be called in meny ways:

1) When an object is passed as a parameter to any of the member functions

Example void Data::putdata(Data x);

2) When a member function returns an object

Example Data getdata() { }

3) When an object is passed by reference to an instance of its own class

For example, Data d1, d2 (d1); // d2(d1) calls copy constructor

Illustration 14.6 Types of constructor

```
#include<iostream>
using namespace std;
class Data
 {
         int i, j;
      public:
         int k;
          Data()
           {
                  cout<<"Non Parametrerized constructor";
                   i=0;
                    j=0'
             }
           Data(int a,int b)
           {
                  cout<<"Parametrerized constructor";
                   i=a;
                    j=b'
             }
          Data(Data &a)
           {
                  cout<<"Copy constructor";
                   i=a.i;
                    j=b.j'
             }

             void display()              //member function
              {
                  cout<< i<<j;
               }
         };
int main()
{
             Data  d1,d2(10,20),d3(d2);
             d1.display();
             d2.display();
             d3.display();
              return 0;

}
```

14.8 Invocation of constructors

There are two ways to create an object using parameterized constructor

- Implicit call
- Explicit call

14.8.1 Implicit call

In this method ,the parameterized constructor is invoked automatically whenever an object is created. For example simple s1(10,20); in this for creating the object s1 parameterized constructor is automatically invoked.

14.8.2 Explicit call

In this method ,the name of the constructor is explicitly given to invoke the parameterized constructor so that the object can be created and initialized .

For example

```
simple s1=simple(10,20);        //explicit call
```

Explicit call method is the most suitable method as it creates a temporary object ,the chance of data loss will not arise.A temprory object lives in memory as long as it is being used in an expression.After this it get destroyed.

14.9 Dynamic initialization of Objects

When the initial values are provided during runtime then it is called dynamic initialization.

Illustration14.7 to illustrate dynamic initialization

```
#include<iostream>
using namespace std;
class X
{   int n;
    float avg;
    public:
    X(int p,float q)
    {   n=p;
        avg=q;  }
    void disp()
    {     cout<<"\n Roll numbe:- " <<n;
          cout<<"\nAverage :- "<<avg;         } };
int main()
{
int a ; float b;
          cout<<"\nEnter the Roll Number";
          cin>>a;
          cout<<"\nEnter the Average";
          cin>>b;
          X x(a,b);       // dynamic initialization
          x.disp();
          return 0;
}
```

Output:

```
Enter the Roll Number 1201
Enter the Average 98.6
 Roll numbe:- 1201
Average :- 98.6
```

14.10 Characteristics of Constructors

- The name of the constructor must be same as that of the class
- No return type can be specified for constructor
- A constructor can have parameter list
- The constructor function can be overloaded
- They cannot be inherited but a derived class can call the base class constructor
- The compiler generates a constructor, in the absence of a user defined constructor.
- Compiler generated constructor is public member function
- The constructor is executed automatically when the object is created
- A constructor can be used explicitly to create new object of its class type

14.11 Destructors

When a class object goes out of scope, a special function called the *destructor* gets executed. The destructor has the same name as the class tag but prefixed with a ~**(tilde)**. Destructor function also return nothing and it does not associated with anydata type.

14.11.1 Need of Destructors

The purpose of the destructor is to free the resources that the object may have acquired during its lifetime. A destructor function removes the memory of an object which was allocated by the constructor at the time of creating a object.

14.11.2 Declaration and Definition

A *destructor* is a special member function that is called when the lifetime of an object ends and destroys the object constructed by the constructor. Normally declared under public.

Illustration14.8 To illustrate destructor function in a class

```
#include<iostream>
using namespace std;
class simple
{
private:
int a, b;
public:
simple()
{
a= 0 ;
b= 0;
cout<< "\n Constructor of class-simple ";
}
void getdata()
{
cout<<"\n Enter values for a and b ";
cin>>a>>b;
}
void putdata()
{
cout<<"\nThe two integers are .. ";
cout<<<<a<<'\t'<< b<<endl;
cout<<"\n The sum = "<<a+b;
}
~simple()
{    cout<<”\n Destructor is executed ”;}
};
int main()
{
simple s;
s.getdata();
s.putdata();
return 0;
}
```

Output:

```
Constructor of class-simple
Enter values for a and b  6 7
The two integers are .. 6      7
The sum = 13
Destructor is executed
```

14.12 Characteristics of Destructors

- The destructor has the same name as that class prefixed by the tilde character '~'.
- The destructor cannot have arguments
- It has no return type
- Destructors cannot be overloaded
- In the absence of user defined destructor, it is generated by the compiler
- The destructor is executed automatically when the control reaches the end of class scope to destroy the object
- They cannot be inherited

Points to Remember:

- A class binds data and associated functions together.
- A class in C++ makes a user defined data type using which objects of this type can be created.
- While declaring a class data members , member functions ,access specifiers and class tag name are given.
- The member functions of a class can either be defined within the class (inline) definition or outside the class definition.
- The public members of the class can be accessed outside the class directly by using object of this class type.
- A class binds data and associated functions together.
- A class in C++ makes a user defined data type using which objects of this type can be created.
- While declaring a class data members , member functions, access specifiers and class tag name are given.
- The member functions of a class can either be defined within the class (inline) definition or outside the class definition.
- The public members of the class can be accessed outside the class directly by using object of this class type.
- A class supports OOP features ENCAPSULATION by binding data and functionsassociated together.
- A class supports Data hiding by hiding the information from the outside world through private and protected members.
- When a member function is called by another member function of the same class , it is calledas nesting of member functions.
- The scope resolution operator (::), when used with the class name depicts that the members belong to that class as in class_ name :: function_name and only used with the variable name as in :: s variable –name , depicts the global variable.(the one with file scope).
- When an instance of a class comes into scope, a special function called the constructor gets executed.
- The constructor function allocates memory and initializes the class object.
- When an instance of a class comes into scope, a special function called the constructor gets executed.
- When a class object goes out of scope, a special function called the destructor gets executed.
- The constructor function name and the destructor have the same name as the classtag.
- A constructor without parameters is called as default constructor.
- A constructor with default argument is equivalent to a default constructor
- Both the constructors and destructor return nothing. They are not associated with any data type.
- Objects can be initialized dynamically .

Hands on practice:

1 Define a class Employee with the following specification

private members of class Employee

empno- integer

ename – 20 characters

basic – float

netpay, hra, da, - float

calculate () – A function to find the basic+hra+da with float return type

public member functions of class employee

havedata() – A function to accept values for empno, ename, basic, hra, da and

call calculate() to compute netpay

dispdata() – A function to display all the data members on the screen

Evaluation

SECTION – A

Choose the correct answer

1. The variables declared inside the class are known as

   (A) data (B) inline

   (C) method (D) attributes

2. Which of the following statements about member functions are True or False?

   i) A member function can call another member function directly with using the dot operator.

   ii) Member function can access the private data of the class.

   (A) i)True, ii)True (B) i)False, ii)True

   (C) i)True, ii)False (D) i)False,ii)False

3. A member function can call another member function directly, without using the dot operator called as

   (A) sub function

   (B) sub member

   (C) nesting of member function

   (D) sibling of member function

4. The member function defined within the class behave like .... functions

   (A) inline (B) Non inline

   (C) Outline (D) Data

5. Which of the following access specifier protects data from inadvertent modifications?

   (A) Private (B) Protected

   (C) Public (D) Global

6. class x

```
 {
      int y;
      public:
      x(int z){y=z;}
      } x1[4];
 int main()
 {  x  x2(10);
    return 0;}
```

How many objects are created for the above program

(A) 10 (B) 14 (C) 5 (D) 2

7. State whether the following statements about the constructor are True or False.

   i) constructors should be declared in the private section.

   ii) constructors are invoked automatically when the objects are created.

   (A) True, True (B) True, False

   (C) False, True (D) False, False

8. Which of the following constructor is executed for the following prototype ?

   add display(add &);   // add is a class name

   (A) Default constructor

   (B) Parameterized constructor

   (C) Copy constructor

   (D) Non Parameterized constructor

SECTION-B

Very Short Answers

1. What are called members?
2. Differentiate structure and class though both are user defined data type.
3. What is the difference between the class and object in terms of oop?
4. Why it is considered as a good practice to define a constructor though compiler can automatically generate a constructor ?
5. Write down the importance of destructor.

SECTION-C

Short Answers

1. Rewrite the following program after removing the syntax errors if any and underline the errors:

```
#include<iostream>
$include<stdio>
class mystud
{       int studid =1001;
        char name[20];
    public
        mystud( )     { }
        void register ( )
        {cin>>stdid;  gets(name);   }
void display ( )
{cout<<studid<<": "<<name<<endl;}
}
int main( )
{ mystud MS;
  register.MS( );
  MS.display( );
}
```

2. Write with example how will you dynamically initialize objects?

3. What are advantages of declaring constructors and destructor under public accessability?

4. Given the following C++ code, answer the questions (i) & (ii).

```
class TestMeOut
{
 public:
 ~TestMeOut()        //Function 1
  {cout<<"Leaving the exam hall"<<endl;}
  TestMeOut()        //Function 2
  {cout<<"Appearing for exam"<<endl;}
   void MyWork()     //Function 3
  {cout<<"Answering"<<endl;}     };
```

(i) In Object Oriented Programming, what is Function 1 referred as and when doesit get invoked / called ?

(ii) In Object Oriented Programming, what is Function 2 referred as and when doesit get invoked / called ?

SECTION - D

Explain in detail

1. Mention the differences between constructor and destructor

2. Define a class RESORT with the following description in C++ :

   Private members:

   Rno // Data member to store room number

   Name //Data member to store user name

Charges //Data member to store per day charge

Days //Data member to store the number of days

Compute( )/*A function to calculate total amount as Days * Charges and if the total amount exceeds 11000 then total amount is 1.02 * Days *Charges */

Public member:

GetInfo( ) /* Function to Read the information like name , room no, charges and days*/

DispInfo( )/* Function to display all entered details and total amount calculated using COMPUTE function*/

3. Write the output of the following

```
#include<iostream>
using namespace std;
class student
{
        int rno, marks;
        public:
        student(int r,int m)
        { cout<<"Constructor "<<endl;
          rno=r;
          marks=m;
        }
        void printdet()
        {
          marks=marks+30;
          cout<<"Name: Bharathi"<<endl;
          cout<<"Roll no : "<<rno<<"\n";
          cout<<"Marks : "<<marks<<endl;
          }
};
int main()
{
        student s(14,70);
        s.printdet();
        cout<< "Back to Main";
        return 0;
}
```

Reference:

(1) Object Oriented Programming with C++ (4th Edition), Dr. E. Balagurusamy,
Mc.Graw Hills.

(2) The Complete Reference C++ (Forth Edition), Herbert Schildt.Mc.Graw Hills.

Unit IV | Object Oriented Programming with C++

CHAPTER 15

Polymorphism

FD1T5E

Learning Objectives

After learning this chapter, the students will be able to

- Understand the purpose of overloading
- Construct C++ programs using function, constructor and operator overloading
- Execute and debug programs which contains the concept of polymorphism

15.1 Introduction

The word polymorphism means many forms (poly – many, morph – shapes) Polymorphism is the ability of a message or function to be displayed in more than one form. In C++, polymorphism is achieved through function overloading and operator overloading. The term overloading means a name having two or more distinct meanings. Thus an **'overloaded function'** refers to a function having more than one distinct meaning.

15.2 Function overloading

The ability of the function to process the message or data in more than one form is called as function overloading. In other words function overloading means two or more functions in the same scope share the same name but their parameters are different. In this situation, the functions that share the same name are said to be overloaded and the process is called function overloading . The number and types of a function's parameters are called the **function's signature.** When you call an overloaded function, the compiler determines the most appropriate definition to use, by comparing the argument types you have used to call the function with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function or operator is called **overload resolution**

15.2.1 Need For Function overloading

Sometimes it's hard to find many different meaningful names for a single action.

Consider the situation to find the area of circle ,triangle and rectangle the following function prototype is given

```
float area_circle(float radius)                          // to calculate the area of a circle
float area_triangle(float half,floatbase,float height)   // to calculate the area of a triangle
float area_rectangle(float length , float breadth)       // to calculate the area of a rectangle
```

This can be rewritten using a single function header in the following manner

float area (float radius);

float area (float half, float base, float height);

float area (float length , float breadth);

Illustration 15.1 C++ Program to demonstrate function overloading

```
#include <iostream>
using namespace std;
void print(int i)
 {cout<< " It is integer " << i <<endl;}
void print(double  f)
{  cout<< " It is float " << f <<endl;}
void print(string c)
{  cout<< " It is string " << c <<endl;}
int main() {
        print(10);
        print(10.10);
        print("Ten");
        return 0;
}
```

Output:

```
It is integer 10
 It is float 10.1
It is string Ten
```

Tip Notes

Function overloading is not only implementing polymorphism but also reduces the number of comparisons in a program and makes the program to execute faster. It also helps the programmer by reducing the number of function names to be remembered.

15.2.2 Rules for function overloading

1. The overloaded function must differ in the number of its arguments or data types
2. The return type of overloaded functions are not considered for overloading same data type

3. The default arguments of overloaded functions are not considered as part of the parameter list in function overloading.

Illustration 15.2 C++ Program to demonstrate function overloading

```
#include <iostream>
using namespace std;
long add(long, long);
long add(long,long,long);
float add(float, float);
int main()
{
 long a, b, c,d;
 float e, f, g;
 cout << "Enter three integers\n";
cin >> a >> b>>c;
d=add(a,b,c);          //number of arguments different  but  same data type
cout << "Sum of 3 integers: " << d << endl;
cout << "Enter two integers\n";
cin >> a >> b;
c = add(a, b);         //two arguments with same data type
cout << "Sum of 2 integers: " << c << endl;
cout << "Enter two floating point numbers\n";
 cin >> e >> f;
 g = add(e, f);        //two arguments with different data type
cout << "Sum of floats: " << g << endl;
}
long add(long c, long g)
{
 long sum;
 sum = c + g;
 return sum;
}
float add(float c, float g)
{
 float sum;
 sum = c + g;
 return sum;
}
long add(long c, long g,long h)
{
 long sum;
 sum = c + g+h;
 return sum;
}
```

Output

```
Enter three integers
3 4 5
Sum of 3 integers: 12
Enter two integers
4 6
Sum of 2 integers: 10
Enter two floating point numbers
2.1 3.1
Sum of floats: 5.2
```

15.3 Constructor overloading

Function overloading can be applied for constructors, as constructors are special functions of classes .A class can have more than one constructor with different signature. Constructor overloading provides flexibility of creating multiple type of objects for a class.

Illustration 15.3 Constructor overloading

Compiler identifies a given member function is a constructor by its name and the return type.

```
#include<iostream>
using namespace std;
class add
{ int num1, num2, sum;
public:
add()
{          cout<<"\n Constructor without parameters.. ";
num1= 0;  num2= 0;  sum = 0; }
add ( int s1, int s2 )
{ cout<<"\n Parameterized constructor... ";
 num1= s1;  num2=s2; sum=0; }
add (add &a)
{ cout<<"\n Copy Constructor ... ";
 num1= a.num1;
 num2=a.num2;
 sum = 0; }
void getdata()
{ cout<<"\nEnter data ... "; cin>>num1>>num2; }
void addition()
{ sum=num1+num2; }
void putdata() {
cout<<"\n The numbers are..";
cout<<num1<<'\t'<<num2;
cout<<"\n The sum of the numbers are.. "<< sum; } };
int main() {
add a, b (10, 20) , c(b);
a.getdata();
a.addition();
b.addition();
c.addition();
cout<<"\n Object a : ";
a.putdata();
cout<<"\n Object b : ";
b.putdata();
cout<<"\n Object c.. ";
c.putdata();
return 0; }
```

Output

```
Constructor without parameters..
 Parameterized constructor...
 Copy Constructor ...
Enter data ...  20 30
 Object a :
 The numbers are..20    30
 The sum of the numbers are.. 50
 Object b :
 The numbers are..10    20
 The sum of the numbers are.. 30
 Object c..
 The numbers are..10    20
 The sum of the numbers are.. 30
```

Note

Since, there are multiple constructors present, argument to the constructor should also be passed while creating an object.

15.4 Operator overloading

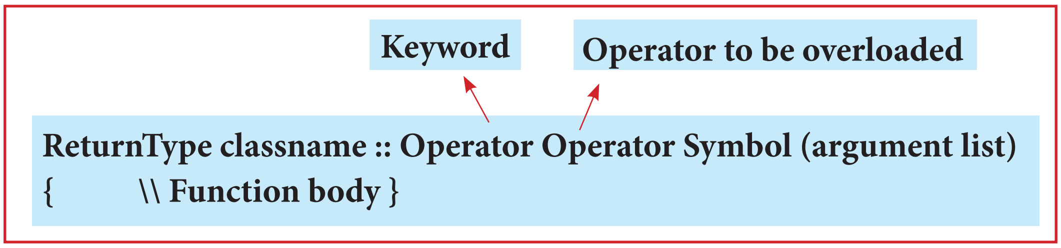
The term Operator overloading, refers to giving additional functionality to the normal C++ operators like +,++,-,—,+=,-=,*.<,>. It is also a type of polymorphism in which an operator is overloaded to give user defined meaning to it .

For example '+' operator can be overloaded to perform addition on various data types, like for Integer, String(concatenation) etc.

Almost all operators can be overloaded in C++. However there are few operator which can not be overloaded. Operator that are not overloaded are follows

- Scope operator (**::**)
- Sizeof
- Member selector (**.**)
- Member pointer selector (⁎)
- Ternary operator (**?:**)

Operator Overloading Syntax



ReturnType classname :: Operator Operator Symbol (argument list)
{ \\ Function body }

15.4.1 Restrictions on Operator Overloading

Following are some restrictions to be kept in mind while implementing operator overloading.

1. Precedence and Associativity of an operator cannot be changed.
2. No new operators can be created, only existing operators can be overloaded.
3. Cannot redefine the meaning of an operator's procedure. You cannot change how integers are added.Only additional functions can be given to an operator
4. Overloaded operators cannot have default arguments.
5. When binary operators are overloaded, the left hand object must be an object of the relevant class

Illustration 15.5 Binary operator overloading using '+'in Complex number addition

```
#include<iostream>
using namespace std;
class complex
{  int real,img;
   public:
void read()
{
cout<<"\nEnter the REAL PART : ";
cin>>real;
cout<<"\nEnter the IMAGINARY PART : ";
cin>>img;
}
complex operator +(complex c2)
{
    complex c3;
    c3.real=real+c2.real;
    c3.img=img+c2.img;
    return c3;
}
    void display()
{  cout<<real<<"+"<<img<<"i";  }
};
int main()
{
    complex c1,c2,c3;
    int choice, cont;
    cout<<"\n\nEnter the First Complex Number";
    c1.read();
    cout<<"\n\nEnter the Second Complex Number";
    c2.read();
    c3=c1+c2;  // binary + overloaded
    cout<<"\n\nSUM = ";
    c3.display();
    return 0;
 }
```

Output

```
Enter the First Complex Number
Enter the REAL PART : 3
Enter the IMAGINARY PART : 4
Enter the Second Complex Number
Enter the REAL PART : 5
Enter the IMAGINARY PART : 8
SUM = 8+12i
```

Illustration 15.6 Concatenation of string using operator overloading

```
#include<string.h>
#include<iostream>
using namespace std;
class strings
{
        public:
        char s[20];
        void getstring(char str[])
{
        strcpy(s,str);
  }
        void operator+(strings);
};
void strings::operator+(strings ob)
{
        strcat(s,ob.s);
        cout<<"\nConcatnated String is:"<<s;
}
int main()
{
        strings ob1, ob2;
        char string1[10], string2[10];
        cout<<"\nEnter First String:";
        cin>>string1;
        ob1.getstring(string1);
        cout<<"\nEnter Second String:";
        cin>>string2;
        ob2.getstring(string2);
        //Calling + operator to Join/Concatnate strings
        ob1+ob2;
        return 0;
}
```

Output

```
Enter First String:COMPUTER
Enter Second String:SCIENCE
Concatnated String is:COMPUTERSCIENCE
```

Points to Remember

- In C++, polymorphism is achieved through function overloading and operator overloading.
- The term overloading means a name having two or more distinct meanings.
- Overloaded function' refers to a function having more than one distinct meaning.
- Overloaded functions have same name but different signatures (Number of argument and type of argument)
- A function's argument list is known as a function signature
- Two function cannot be overloaded when the only difference is that one takes a reference parameter and the other takes a normal, call-by-value parameter.
- Ordinary functions as well member functions can be overloaded
- A class can have overloaded constructors where as destructor function cannot be overloaded.
- The mechanism of giving special meaning to an operator is known as operator overloading.
- Operator overloading provides new definitions for most of the C++ operators.
- Even user defined types (objects) can be overloaded.
- The definition of the overloaded operator is given using the keyword 'operator' followed by an operator symbol.
- We can overload all the C++ operators except the following:
- Scope resolution operator (::), sizeof (), Conditional operator (?:), Member selection(.) and Member pointer selector (*) operator

Evaluation

FDAP72

SECTION – A

Choose the correct answer

1. Which of the following refers to a function having more than one distinct meaning?

   (A) Function Overloading (B) Member overloading

   (C) Operator overloading (D) Operations overloading

2. Which of the following reduces the number of comparisons in a program ?

   (A) Operator overloading (B) Operations overloading

   (C) Function Overloading (D) Member overloading

3.
```
void dispchar(char ch='$',int size=10)
    {
        for(int i=1;i<=size;i++)
        cout<<ch;
    }
```

How will you invoke the function dispchar() for the following input?

To print $ for 10 times

(A) dispchar(); (B) dispchar(ch,size);

(C) dispchar($,10); (D)dispchar('$',10 times);

4. Which of the following is not true with respect to function overloading?

(A) The overloaded functions must differ in their signature.

(B) The return type is also considered for overloading a function.

(C) The default arguments of overloaded functions are not considered for Overloading.

(D) Destructor function cannot be overloaded.

5. Which of the following is invalid prototype for function overloading

(A) void fun (intx);
void fun (char ch) ;

(B) void fun (intx);
void fun (inty);

(C) void fun (double d);
void fun (char ch);

(D) void fun (double d);
void fun (inty);

SECTION-B

Very Short Answers

1. What is function overloading?
2. List the operators that cannot be overloaded.
3. class add{int x; public: add(int)}; Write an outline definition for the constructor.
4. Does the return type of a function help in overloading a function?
5. What is the use of overloading a function?

SECTION-C

Short Answers

1. What are the rules for function overloading?
2. How does a compiler decide as to which function should be invoked when there are many functions? Give an example.
3. What is operator overloading? Give some examples of operators which can be overloaded.
4. Discuss the benefits of constructor overloading ?
5. class sale (int cost, discount ;public: sale(sale &); Write a non inline definition for constructor specified;

SECTION - D

Explain in detail

1. What are the rules for operator overloading?

2. Answer the question (i) to (v) after going through the following class.

```
class Book {
int BookCode ; char Bookname[20];float fees;
public:
Book( )                        //Function 1
{ fees=1000;
  BookCode=1;
  strcpy(Bookname,"C++");  }
void display(float C)          //Function 2
{   cout<<BookCode<<":"<<Bookname<<":"<<fees<<endl;   }
~Book( )                       //Function 3
{  cout<<"End of Book Object"<<endl;   }
      Book (intSC,char S[ ],float F)  ;          //Function 4
};
```

(i) In the above program, what are Function 1 and Function 4 combined together referred as?

(ii) Which concept is illustrated by Function3? When is this function called/ invoked?

(iii) What is the use of Function3?

(iv) Write the statements in main to invoke function1 and function2

(v) Write the definition for Function4 .

3. Write the output of the following program

```
include<iostream>
using namespace std;
class Seminar
{ int Time;
public:
Seminar()
 {     Time=30;cout<<"Seminar starts now"<<endl;   }
void Lecture()
{ cout<<"Lectures in the seminar on"<<endl; }
Seminar(int Duration)
{   Time=Duration;cout<<"Welcome to Seminar "<<endl; }
Seminar(Seminar &D)
{   Time=D.Time;cout<<"Recap of Previous Seminar Content "<<endl;}
~Seminar()
{cout<<"Vote of thanks"<<endl; }  };
int main()
{       Seminar s1,s2(2),s3(s2);
        s1.Lecture();
        return 0;
}
```

4. Answer the questions based on the following program

```
#include<iostream>
#include<string.h>
using namespace std;
class comp {
public:
char s[10];
void getstring(char str[10])
   { strcpy(s,str);    }
void operator==(comp);
};
void comp::operator==(comp ob)
{ if(strcmp(s,ob.s)==0)
 cout<<"\nStrings are Equal";
else
 cout<<"\nStrings are not Equal"; }
int main()
{ comp ob, ob1;
char string1[10], string2[10];
cout<<"Enter First String:";
cin>>string1;
ob.getstring(string1);
cout<<"\nEnter Second String:";
cin>>string2;
ob1.getstring(string2);
ob==ob1;
return 0; }
```

(i) Mention the objects which will have the scope till the end of the program.

(ii) Name the object which gets destroyed in between the program

(iii) Name the operator which is over loaded and write the statement that invokes it.

(iv) Write out the prototype of the overloaded member function

(v) What types of operands are used for the overloaded operator?

(vi) Which constructor will get executed in the above program? Write the output of the program

CASE STUDY

Suppose you have a Kitty Bank with an initial amount of Rs500 and you have to add some more amount to it. Create a class 'Deposit' with a data member named 'amount' with an initial value of Rs500. Now make three constructors of this class as follows:

1. without any parameter - no amount will be added to the Kitty Bank
2. having a parameter which is the amount that will be added to the Kitty Bank
3. whenever amount is added an additional equal amount will be deposited automatically

Create an object of the 'Deposit' and display the final amount in the Kitty Bank.

Reference:

1. Object Oriented Programming with C++ (4th Edition), Dr. E. Balagurusamy, Mc.Graw Hills.
2. The Complete Reference C++ (Forth Edition), Herbert Schildt. Mc.Graw Hills.
3. The C++ Programming Language,BjarneStroustrup

Unit IV Object Oriented Programming with C++

CHAPTER 16

Inheritance

Learning Objectives

After the completion of this chapter, the student will be able to

- Understand the purpose of Inheritance
- Construct C++ programs using Inheritance

FDJK8P

- Execute and debug programs which contains the concept of Inheritance

16.1 Introduction to Inheritance

Inheritance is one of the most important features of Object Oriented Programming. In object-oriented programming, inheritance enables new class and its objects to take on the properties of the existing classes. A class that is used as the basis for creating a new class is called a superclass or base class. A class that inherits from a superclass is called a subclass or derived class

16.2 Need for Inheritance

Inheritance is an important feature of object oriented programming used for code reusability. It is a process of creating new classes called derived classes, from the existing or base classes. Inheritance allows us to inherit all the code (except declared as private) of one class to another class. The class to be inherited is called base class or parent class and the class which inherits the other class is called derived class or child class. The derived class is a power packed class, as it can add additional attributes and methods and thus enhance its functionality.

Notes

The main advantage of inheritance is

- It represents real world relationships well
- It provides reusability of code
- It supports transitivity

16.3 Types of Inheritance

There are different types of inheritance viz., Single Inheritance, Multiple inheritance, Multilevel inheritance, hybrid inheritance and hierarchical inheritance.

1. Single Inheritance

When a derived class inherits only from one base class, it is known as single inheritance

2. Multiple Inheritance

When a derived class inherits from multiple base classes it is known as multiple inheritance

3. Hierarchical inheritance

When more than one derived classes are created from a single base class , it is known as Hierarchical inheritance.

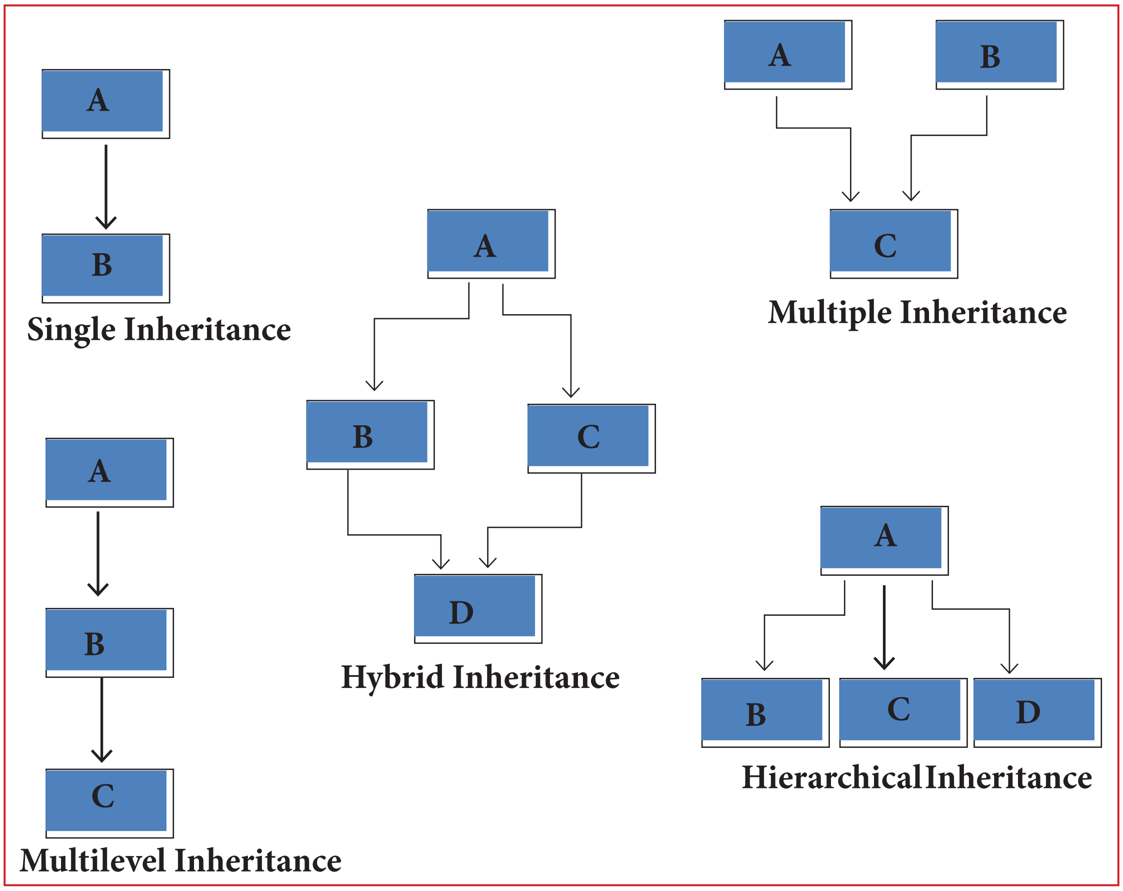
4. Multilevel Inheritance

The transitive nature of inheritance is reflected by this form of inheritance. When a class is derived from a class which is a derived class – then it is referred to as multilevel inheritance.

5. Hybrid inheritance

When there is a combination of more than one type of inheritance, it is known as hybrid inheritance. Hence, it may be a combination of Multilevel and Multiple inheritance or Hierarchical and Multilevel inheritance or Hierarchical, Multilevel and Multiple inheritance.

The following diagram represents the different types of inheritance



16.4 Derived Class and Base class

While defining a derived class, the derived class should identify the class from which it is derived.The following points should be observed for defining the derived class.

i The keyword class has to be used

ii The name of the derived class is to be given after the keyword class

iii A single colon (:)

iv The type of derivation (the visibility mode), namely private, public or protected. If no visibility mode is specified ,then by default the visibility mode is considered as private.

v The name of the base class(parent class), if more than one base class, then it can be given separated by comma.

class derived_class_name :visibility_mode base_class_name

{ // members of derivedclass };

The following are some of the examples for different forms of inheritance

16.4.1 Single Inheritance

Illustration 16.1 Single inheritance

```
# include <iostream>
using namespace std;
class student          //base class
{ private :
          char name[20];
          int rno;
   public:
          void  acceptname()
          { cout<<"\n Enter roll no and name .. ";    cin>>rno>>name;    }
          void  displayname()
          { cout<<"\n Roll no :-"<<rno;
            cout<<"\n Name :-"<<name<<endl;      }  };
class exam : public student          //derived class with single base class
 {
      public:
        int  mark1,mark2,total;
        void acceptmark()
        {  cout<<"\n Enter mark1 and mark2.... ";    cin>>mark1>>mark2; }
void  displaymark()
{ cout<<"\n\t\t Marks Obtained ";
  cout<<"\n Subject1.. "<<mark1;
  cout<<"\n Subject2 .. "<<mark2;
  cout<<"\n Total .. "<<mark1+mark2;     }  };
int main()
{   exam e1;
 e1.acceptname();    //calling base class function using derived class object
 e1.acceptmark();
 e1.displayname();             //calling base class function using derived class object
 e1.displaymark();
 return 0;
}
```

Output

```
Enter roll no and name .. 1201 KANNAN
Enter lang,eng,phy,che,csc,mat marks.. 100 100 100 100 100 100
Roll no :-1201
Name :-KANNAN
                Marks Obtained
 Language.. 100
English .. 100
Physics .. 100
 Chemistry.. 100
Comp.sci.. 100
Maths ..   100
```

In the above program the derived class “exam” inherits all the members of the base class “student”. But it has access privilege only to the non private members of the base class.

16.4.3 Multilevel Inheritance

Illustration 16.2 Multilevelw inheritance

```
# include <iostream>
using namespace std;
class student              //base class
{
private :
char name[20];
int rno;
public:
void acceptname()
{ cout<<"\n Enter roll no and name .. "; cin>>rno>>name;
}
void displayname()
{ cout<<"\n Roll no :-"<<rno;
cout<<"\n Name :-"<<name<<endl;   }};
 class exam : public student            //derived class with single base class
                      {
                                 total=mark1+mark2+mark3;
                                 cout<<"\nTOTAL MARK SCORED    :  "<<total;
              }
 };
```

```
{
          public:
          int  mark1, mark2 ,mark3;
          void acceptmark()
          {         cout<<"\n Enter 3 subject marks.. ";
                    cin>>mark1>>mark2>>mark3;     }
void displaymark(){
cout<<"\n\t\t Marks Obtained ";
cout<<"\n Subject1... "<<mark1;
cout<<"\n Subject2... "<<mark2;
cout<<"\n Subject3... "<<mark3;  }  };
class result : public exam
{
int total;
public:
void showresult()
{
total=mark1+mark2+mark3;
cout<<"\nTOTAL MARK SCORED : "<<total;
}
};
int main()
{
          result r1;
r1.acceptname();                //calling base class function using derived class object
r1.acceptmark();                //calling base class function which itself is a derived
                                // class  function using its derived class object
r1.displayname();               //calling base class function using derived class object
r1.displaymark();               /*calling base class function which itself is a derived
                                class  function using its derived class object*/
r1.showresult();                //calling the child class function
return 0;
}
```

Output:

```
Enter roll no and name .. 1201 Lohit Sarathi
 Enter 3 subject marks.. 98 100 100
 Roll no :-1201
 Name :-Lohith Sarathi
             Marks Obtained
Subject1 ...  98
Subject 2 ...  100
Subject 3... 100
TOTAL MARK SCORED    :  298
```

In the above program class “result “ is derived from class “exam” which itself is derived from class student.

Note

In multilevel inheritance the level of inheritance can be extended to any number of level depending upon the relation. Multilevel inheritance is similar to relation between grandfather, father and child.

A class without any declaration will have 1 byte size.class x{}; X occupies 1 byte.

In the above program the derived class “result” has acquired the properties of class “detail” and class “exam” which is derived from “student”. So this inheritance is a combination of multi level and multiple inheritance and so it is called hybrid inheritance

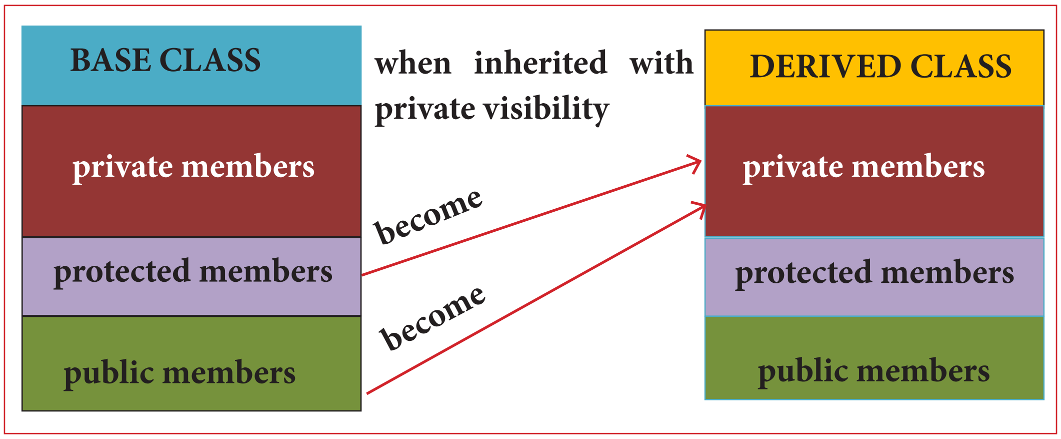
16.5 VISIBILITY MODES

An important feature of Inheritance is to know which member of the base class will be acquired by the derived class. This is done by using visibility modes.

The accessibility of base class by the derived class is controlled by visibility modes. The three visibility modes are private, protected and public. The default visibility mode is private. Though visibility modes and access specifiers look similar, the main difference between them is Access specifiers control the accessibility of the members with in the class where as visibility modes control the access of inherited members with in the class.

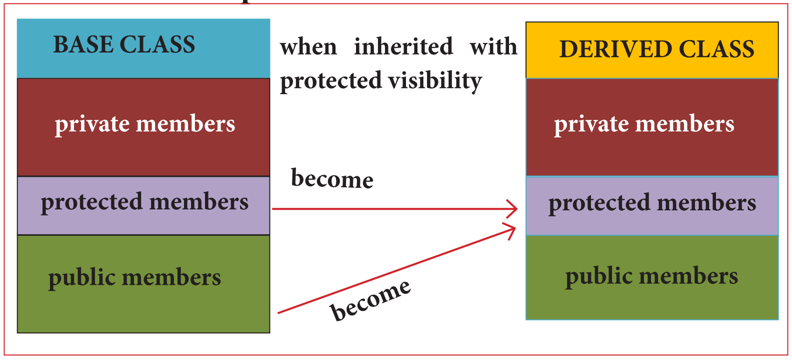
16.5.1 Private visibility mode

When a base class is inherited with **private** visibility mode the public and protected members of the base class become **‘private’** members of the derived class



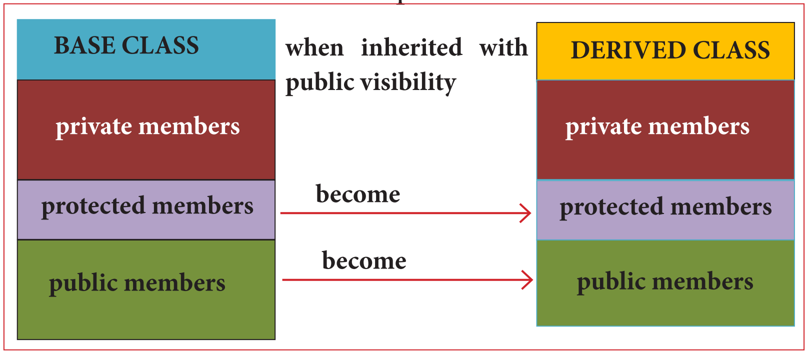
16.5.2 protected visibility mode

When a base class is inherited with **protected** visibility mode the protected and public members of the base class become '**protected**' members of the derived class



16.5.3 public visibility mode

When a base class is inherited with **public** visibility mode , the protected members of the base class will be inherited as **protected** members of the derived class and the **public** members of the base class will be inherited as public members of the derived class.



Tip Notes

When classes are inherited with public, protected or private the private members of the base class are not inherited they are only visible i.e continue to exist in derived classes, and cannot be accessed

Illustration 16.3 Explains the significance of different visibility modes.

```
//Implementation of Single Inheritance using public visibility mode
#include <iostream>
using namespace std;
class Shape
{   private:
          int count;
  protected:
          int width, height;
  public:
          void setWidth(int w)
          {       width = w;     }
void setHeight(int h)
{ height = h; } };
```

```
class Rectangle: public Shape
{
public:
int getArea()
{
return (width * height);
}
};
int main()
{
Rectangle Rect;
Rect.setWidth(5);
Rect.setHeight(7);
// Print the area of theobject.
cout<< "Total area: "<<Rect.getArea() <<endl;
return 0;
}
```

```
Output
Total area: 35
```

The following table contain the members defined inside each class before inheritance

MEMBERS of class	visibility modes		
	Private	Protected	Public

Shape(base class)	int count;	int width;
		int height;
			void setWidth(int)
			void setHeight(int)

Rectangle (derived class only with its defined members)			int getArea();

The following table contains the details of members defined after inheritance

MEMBERS of class	visibility modes –public for acquiring the properties of the base class		
	Private	Protected	Public

Shape(base class)	int count;	int width;
		int height;
			void setWidth(int)
			void setHeight(int)

Rectangle (derived class acquired the properties of base class with public visibility)	Private members of base classes are not directly accessible by the derived class	int width;
		int height;
			int getArea();
			void setWidth(int)
			void setHeight(int)

Suppose the class **rectangle is derived with protected visibility** then the properties of class rectangle will change as follows

MEMBERS of class	visibility modes –protected for acquiring the properties of the base class		
	Private	Protected	Public

Shape(base class)	int count;	int width;
		int height;
			void setWidth(int)
			void setHeight(int)

Rectangle (derived class acquired the properties of base class with protected visibility)	Private members of base classes are not directly accessible by the derived class	int width;
		int height;
		void setWidth(int)
		void setHeight(int)
			int getArea();

In case the class rectangle is derived with private visibility mode from its base class shape then the property of class rectangle will change as follows

MEMBERS of class	visibility modes –private for acquiring the properties of the base class		
	Private	Protected	Public
Shape(base class)	int count;		void setWidth(int) void setHeight(int)

Rectangle (derived class acquired the properties of base class with private visibility)	int width;
	int height;
	void setWidth(int)
	void setHeight(int)
			int getArea();

When you derive a class from an existing base class,it may inherit the properties of the base class based on its visibility mode.So one must give appropriate visibility mode depending on the need.

Private inheritance should be used when you want the features of the base class to be available to the derived class but not to the classes that are derived from the derived class.

Protected inheritance should be used when features of base class to be available only to the derived class members but not to the outside world.

Public inheritance can be used when features of base class to be available to the derived class members and also to the outside world.

16.6 Inheritance and constructors and destructors

When an object of the derived class is created ,the compiler first call the base class constructor and then the constructor of the derived class. This is because the derived class is built up on the members of the base class. When the object of a derived class expires first the derived class destructor is invoked followed by the base class destructor.

Illustration 16.4 The order of constructors and destructors

```
#include<iostream>
using namespace std;
class base
{
public:
base()
{ cout<<"\nConstructor of base class..."; }
~base()
{ cout<<"\nDestructor of base class.... "; }
};
class derived:public base
{
public :
derived()
{ cout << "\nConstructor of derived ...";  }
~derived()
{ cout << "\nDestructor of derived ..."; }};
class derived1 :public derived
{
public :
derived1()
{ cout << "\nConstructor of derived1 ...";}
~derived1()
{ cout << "\nDestructor of derived1 ...";}
};
int main()
{
derived1 x;
return 0;
}
```

Output:

```
Constructor of base class...
Constructor of derived ...
Constructor of derived1 ...
Destructor of derived1 ...
Destructor of derived ...
Destructor of base class....
```

Notes

- The constructors are executed in the order of inherited class i.e., from base constructor to derived. The destructors are executed in the reverse order.
- The size of derived class object=size of all base class data members + size of all data members in derived class.

16.7 Overriding / Shadowing Base class functions in derived class

In case of inheritance there are situations where the member function of the base class and derived classes have the same name. If the derived class object calls the overloaded member function it leads to confusion to the compiler as to which function is to be invoked. The derived class member function have higher priority than the base class member function. This shadows the member function of the base class which has the same name like the member function of the derived class. The scope resolution (::) operator resolves this problem.

Illustration 16.5 The use of scope resolution operator in derived class

```
#include<iostream>
#include<string>
using namespace std;
class Employee
{
    private:
        char name[50];
        int code;
    public:
        void getdata();
        void display();
};
class staff: public Employee
{
    private:
        int ex;
    public:
            void getdata();
        void display();
};
void Employee::display()
{
            cout<<"Name:"<<name<<endl;
            cout<<"Code:"<<code<<endl;
}
void staff::display()
{           Employee :: display();//overriding
            cout<<"Experience:"<<ex<<" Years"<<endl;
}
int main()
{
            staff s;
            cout<<"Enter data"<<endl;
            s.getdata();
            cout<<endl<<endl<<"\tDisplay Data"<<endl;
            s.display();
            return 0;
}
```

Output

```
Enter data
Name: SUGANYA
Code: 1201
Experience: 30
        Display Data
Name: SUGANYA
Code:1201
Experience:30 Years
```

In the above program getdata() and display() are defined both in base and in derived class. So when the derived class staff inherits the properties of Employee class it will have two getdata() and display() each. To differentiate the derived getdata() and display() from the defined getdata() and display() :: (scope resolution) operator is given along with the base class name to the base class members

Note

When a derived class member function has the same name as that of its base class member function ,the derived class member function shadows/hides the base class's inherited function .This situation is called function overriding and this can be resolved by giving the base class name followed by :: and the member function name.

Points to Remember:

- The mechanism of deriving new class from an existing class is called inheritance.
- The main advantage of Inheritance is it supports reusability of code.
- The derived class inherits all the properties of the base class. It is a power packed class, as it can add additional attributes and methods and thus enhance its functionality.
- The various types of Inheritance are Single inheritance, multiple inheritance, multilevel inheritance, hierarchical inheritance and hybrid inheritance
- When a derived class inherits only from one base class, it is known as single inheritance
- When a derived class inherits from multiple base classes itis known as multiple inheritance
- When a class is derived from a class which is a derived class itself – then this is referred to as multilevel inheritance. The transitive nature of inheritance is reflected by this form of inheritance.
- When more than one derived classes are created from a single base class , it is known as Hierarchical inheritance.
- When there is a combination of more than one type of inheritance, it is known as hybrid inheritance.
- In multiple inheritance, the base classes are constructed in the order in which they appear in the declaration of the derived class.
- A sub-class can derive itself publicly, privately or protectedly.
- The private member of a class cannot be inherited .
- In publicly derived class,the public members of the base class remain public and protected members of base class remain protected in derived class.
- In privately derived class, the public and the protected members of the base class become private in derived class
- When class is derived in protected mode, the public and protected members of base class become protected in derived class.
- constructors and destructors of the base class are not inherited but during the creation of an object for derived class the constructors of base class will automatically be invoked.
- The destructors are invoked in reverse order .The destructors of the derived classes are invoked first and then the base class.
- The size of derived class object=size of all base class data members + size of all data members in derived class
- overriding of the members are resolved by using Scope resolution operator(::).
- this pointer is used to refer to the current objects members

Evaluation

FDPTLF

SECTION – A

Choose the correct answer

1. Which of the following is the process of creating new classes from an existing class

   (a) Polymorphism (b) Inheritance
   (c) Encapsulation (d) super class

2. Which of the following derives a class student from the base class school

   (a) school: student
   (b) class student : public school
   (c) student : public school
   (d) class school : public student

3. The type of inheritance that reflects the transitive nature is

   (A) Single Inheritance
   (B) Multiple Inheritance
   (C) Multilevel Inheritance
   (D) Hybrid Inheritance

4. Which visibility mode should be used when you want the features of the base class to be available to the derived class but not to the classes that are derived from the derived class?

   (A) Private (B) Public
   (C) Protected (D) All of these

5. Inheritance is a process of creating new class from

   (A) Base class (B) abstract
   (C) derived class (D) Function

6. A class is derived from a class which is a derived class itself, then this is referred to as

   (A) multiple inheritance
   (B) multilevel inheritance
   (C) single inheritance
   (D) double inheritance

7. Which amongst the following is executed in the order of inheritance?

   (A) Destructor (B) Member function
   (C) Constructor (D) Object

8. Which of the following is true with respect to inheritance?

   (A) Private members of base class are inherited to the derived class with private
   (B) Private members of base class are not inherited to the derived class with private accessibility
   (C) Public members of base class are inherited but not visible to the derived class
   (D) Protected members of base class are inherited but not visible to the outsideclass

9. Based on the following class declaration answer the questions (from9.1 to 9.4)

```
class vehicle
{ int wheels;
public:
void input_data(float,float);
void output_data();
protected:
int passenger;
 };
class heavy_vehicle : protected vehicle {
int diesel_petrol;
protected:
        int load;
public:
void read_data(float,float)
void write_data(); };
class bus: private heavy_vehicle {
char Ticket[20];
public:
void fetch_data(char);
void display_data(); };
```

9.1. Which is the base class of the class heavy_vehicle?

(a) Bus (b) heavy_vehicle

(c) vehicle (d) both (a) and (c)

9.2. The data member that can be accessed from the function displaydata()

(a) passenger (b) load

(c) Ticket (d) All of these

9.3. The member function that can be accessed by an objects of bus Class is

(a) input_data(), output_data()

(b) read_data() ,write_data()

(c) fetch_data(), display_data()

(d) All of these

9.4. The member function that is inherited as public by Class Bus

(a) input_data(), output_data()

(b) read_data(), write_data()

(c) fetch_data(), display_data()

(d) none of these

SECTION-B

Very Short Answers

1. What is inheritance?
2. What is a base class?
3. Why derived class is called power packed class?
4. In what multilevel and multiple inheritance differ though both contains many base class?
5. What is the difference between public and private visibility mode?

SECTION-C

Short Answers

1. What are the points to be noted while deriving a new class?
2. What is difference between the members present in the private visibility mode and the members present in the public visibility mode
3. What is the difference between polymorphism and inheritance though are usedfor reusability of code?
4. What do you mean by overriding?
5. Write some facts about the execution of constructors and destructors in inheritance

SECTION - D

Explain in detail

1. Explain the different types of inheritance
2. Explain the different visibility mode through pictorial representation
3. **Consider the following c++ code and answer the questions**

```
class Personal
{
int Class,Rno;
char Section;
protected:
char Name[20];
public:
personal();
void pentry();
void Pdisplay(); };
class Marks:private Personal
{ float M{5};
protected:
char Grade[5];
public:
Marks();
void Mentry();
void Mdisplay(); };
class Result:public Marks
{
float Total,Agg;
public:
char FinalGrade, Commence[20];
Result();
void Rcalculate();
void Rdisplay();
};
```

3.1. Which type of Inheritance is shown in the program?

3.2. Specify the visibility mode of base classes.

3.3 Give the sequence of Constructor/ Destructor Invocation when object of class Result is created.

3.4. Name the base class(/es) and derived class (/es).

3.5 Give number of bytes to be occupied by the object of the following class:
(a) Personal (b) Marks
(c) Result

3.6. Write the names of data members accessible from the object of class Result.

3.7. Write the names of all member functions accessible from the object of class Result.

3.8 Write the names of all members accessible from member functionsof class Result.

4. Write the output of the following program

```
#include<iostream>
using namespace std;
class A
{       protected:
        int x;
        public:
        void show()
        {cout<<"x = "<<x<<endl;}
        A()
        { cout<<endl<<" I am class A "<<endl;}
        ~A()
        { cout<<endl<<" Bye ";} };
class B : public A
{protected:
  int y;
  public:
  B(int x1, int y1)
  { x = x1;
    y = y1;       }
B()
{   cout<<endl<<" I am class B "<<endl; }
~B()
{   cout<<endl<<" Bye "; }
void show()
{ cout<<"x = "<<x<<endl;
  cout<<"y = "<<y<<endl;   } };
int main()
{A objA;
B objB(30, 20);
objB.show();
return 0;  }
```

5. Debug the following program

```
%include(iostream.h)
#include<conio.h>
class A()
{ public;
int a1,a2:a3;
void getdata[]
{       a1=15;  a2=13; a3=13; }         }
class B:: public A()
{       PUBLIC
        voidfunc()
        { int b1:b2:b3;
          A::getdata[];
          b1=a1;
          b2=a2;
          a3=a3;
          cout<<b1<<’\t’<<b2<<’t\’<<b3; }
void main()
{       B der;
        der1:func(); }
```

CASE STUDY

Write a class for a class Stock

Each Stock has a data member which holds the net price, and a constructor which sets this price. Each Stock has a method get_Price(), which calculates and returns the gross price (the gross price includes VAT at 21%)

Reference:

Object Oriented Programming with C++ (4th Edition), Dr. E. Balagurusamy, Mc.Graw Hills.

Unit V COMPUTER ETHICS AND CYBER SECURITY

CHAPTER 17

Computer Ethics And Cyber Security

Learning Objectives

After learning this chapter, the students will be able to

- To know about cyber-crimes.
- To understand the guidelines and need for ethics in cyber-world.
- To understand issues related to cyber-crimes.
- To know the functionality of firewalls and proxy servers.
- To learn about encryption and decryption.
- To gain knowledge on IT Act.

FDTGAC

17.1 INTRODUCTION

Internet is a communication media which is easily accessible and open to all. Information Technology is widespread through computers, mobile phones and internet. There is a lot of scope and possibility for misuse of Information Technology.

Computer systems in general are vulnerable. They play an important role in the daily lives of individuals and businesses. Special care must be taken explicitly in order to ensure that the valuable data do not get into wrong hands. Hence, the data need to be protected.

A cyber-crime is a crime which involves computer and network.This is becoming a growing threat to society and is caused by criminals or irresponsible action of individuals who are exploiting the widespread use of Internet. It presents a major challenge to the ethical use of information technologies. Cyber-crime also poses threats to the integrity, safety and survival of most business systems.

Figure. 17.1 presents the types of cyber-crimes that happen across the world.

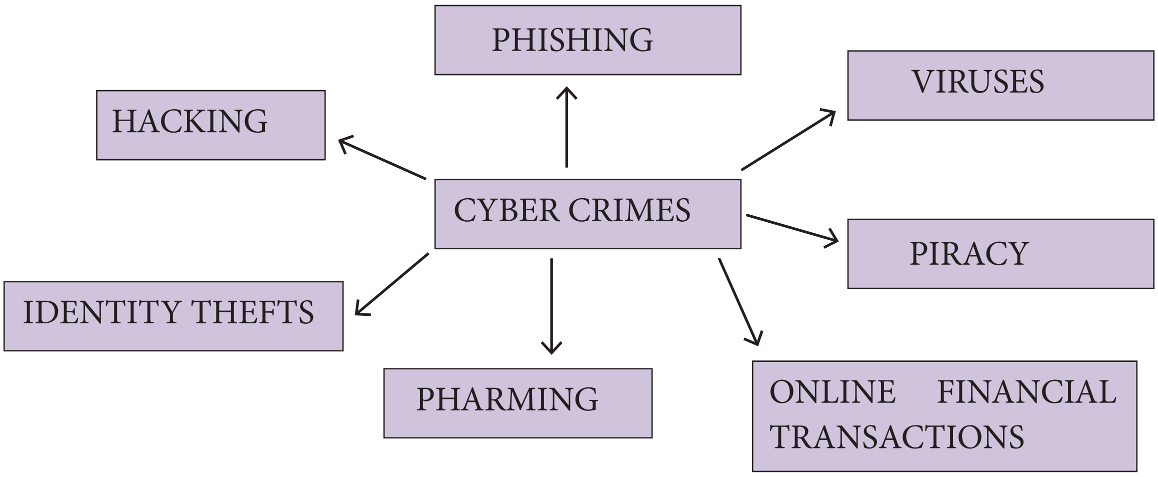


Figure 17.1 Types of cyber – crimes

ETHICS

Ethics means "What is wrong and What is Right". It is a set of moral principles that rule the behavior of individuals who use computers. An individual gains knowledge to follow the right behavior, using morals that are also known as ethics. Morals refer to the generally accepted standards of right and wrong in the society. Similarly, in cyber-world, there are certain standards such as

- Do not use pirated software
- Do not use unauthorized user accounts
- Do not steal others' passwords
- Do not hack

The core issues in computer ethics are based on the scenarios arising from the use of internet such as privacy, publication of copyrighted content, unauthorized distribution of digital content and user interaction with web sites, software and related services.

COMPUTER ETHICS

With the help of internet, world has now become a global village. Internet has been proven to be a boon to individuals as well as various organizations and businesses. e-Commerce is becoming very popular among businesses as it helps them to reach a wide range of customers faster than any other means.

Computer ethics deals with the procedures, values and practices that govern the process of consuming computer technology and its related disciplines without damaging or violating the moral values and beliefs of any individual, organization or entity.

GUIDELINES OF ETHICS

Generally, the following guidelines should be observed by computer users:

1. **Honesty:** Users should be truthful while using the internet.
2. **Confidentiality:** Users should not share any important information with unauthorized people.
3. **Respect:** Each user should respect the privacy of other users.

4. **Professionalism:** Each user should maintain professional conduct.

5. **Obey The Law:** Users should strictly obey the cyber law in computer usage.

6. **Responsibility:** Each user should take ownership and responsibility for their actions

> DO YOU KNOW?
>
> Ethics is a set of moral principles that govern the behavior of an individual in a society, and Computer ethics is set of moral principles that regulate the use of computers by users.

17.2 ETHICAL ISSUES

An Ethical issue is a problem or issue that requires a person or organization to choose between alternatives that must be evaluated as right (ethical) or wrong (unethical). These issues must be addressed and resolved to have a positive influence in society.

Some of the common ethical issues are listed below:

- Cyber crime
- Software Piracy
- Unauthorized Access
- Hacking
- Use of computers to commit fraud
- Sabotage in the form of viruses
- Making false claims using computers

CYBER CRIME

Cybercrime is an intellectual, white-collar crime. Those who commit such crimes generally manipulate the computer system in an intelligent manner.

For example – illegal money transfer via internet.

Examples of some Computer crimes and their functions are listed below in **Table 17.1:**

Table 17.1 Computer Crime

Crime	Function
Cyber Terrorism	Hacking, threats, and blackmailing towards a business or a person.
Cyber stalking	Harassing through online.
Malware	Malicious programs that can perform a variety of functions including stealing, encrypting or deleting sensitive data, altering or hijacking core computing functions and monitoring user's computer activity without their permission.
Denial of service attack	Overloading a system with fake requests so that it cannot serve normal legitimate requests.
Fraud	Manipulating data, for example changing the banking records to transfer money to an unauthorized account.
Harvesting	A person or program collects login and password information from a legitimate user to illegally gain access to others' account(s).
Identity theft	It is a crime where the criminals impersonate individuals, usually for financial gain.
Intellectual property theft	Stealing practical or conceptual information developed by another person or company.
Salami slicing	Stealing tiny amounts of money from each transaction.
Scam	Tricking people into believing something that is not true.
Spam	Distribute unwanted e-mail to a large number ofinternet users.
Spoofing	It is a malicious practice in which communication is send from unknown source disguised as a source known to the receiver.

SOFTWARE PIRACY

Software Piracy is about the copyright violation of software created originally by an individual or an institution. It includes stealing of codes / programs and other information illegally and creating duplicate copies by unauthorized means and utilizing this data either for one's own benefit or for commercial profit.

In simple words,Software Piracy is "unauthorized copying of software". **Figure 17.2** shows a diagrammatical representation of software piracy.

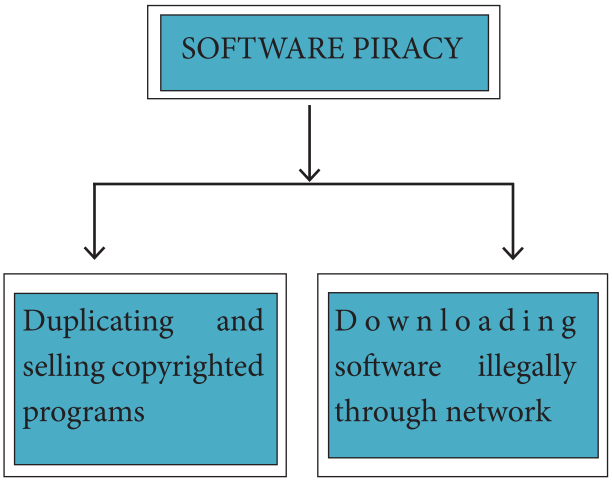

SOFTWARE PIRACY
Duplicating and selling copyrighted programs
Downloading software illegally through network

Figure 17.2- Diagrammatic representation of Software piracy

An entirely different approach to software piracy is called **Shareware**, this acknowledges the futility of trying to stop people from copying software and instead relies on people's honesty. Shareware publishers encourage users to give copies of programs to friends and colleagues but ask everyone who uses that program regularly to pay a registration fee to the program's author directly. Commercial programs that are made available to the public illegally are often called **Warez**.

UNAUTHORIZED ACCESS

Unauthorized access is when someone gains access to a website, program, server, service, or other system by breaking into a legitimate user account. For example, if someone tries guessing a password or username for an account that was not theirs until they gained access, it is considered an unauthorized access.

To prevent unauthorized access, Firewalls, **Intrusion Detection Systems** (IDS), Virus and Content Scanners, Patches and Hot fixes are used.

HACKING

Hacking is intruding into a computer system to steal personal data without the owner's permission or knowledge (like to steal a password). It is also gaining unauthorized access to a computer system, and altering its contents. It may be done in pursuit of a criminal activity or it may be a hobby. Hacking may be harmless if the hacker is only enjoying the challenge of breaking systems' defenses, but such ethical hacking should be practiced only as controlled experiments. **Figure 17.3** shows a diagrammatic representation of Hacking.

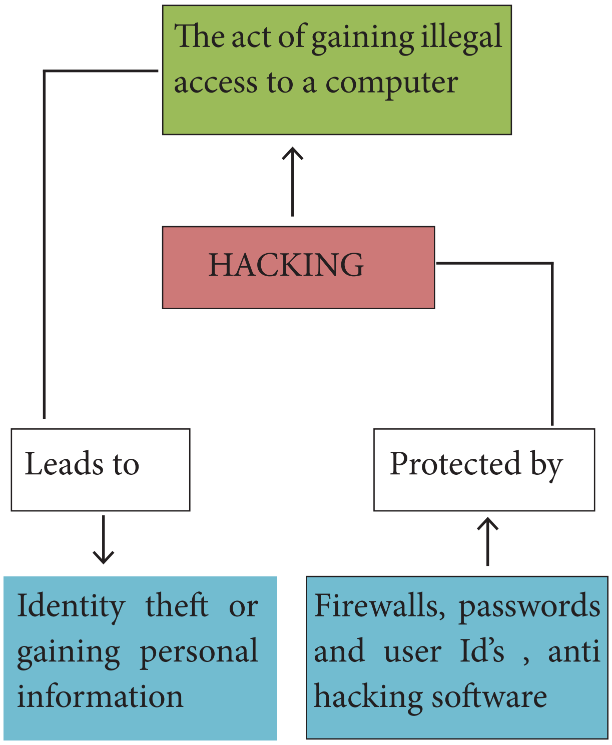

The act of gaining illegal access to a computer
HACKING
Leads to
Protected by
Identity theft or gaining personal information
Firewalls, passwords and user Id's , anti hacking software

Figure 17.3 Diagramatic representation of Hacking

CRACKING

Cracking is where someone edits a program source so that the code can be exploited or modified. A cracker (also called a black hat or dark side hacker) is a malicious or criminal hacker. “Cracking” means trying to get into computer systems in order to steal, corrupt, or illegitimately view data.

A cracker is someone who breaks into someone else's computer system, often on a network, bypassing passwords or licenses in computer programs.

They may send official e-mail requesting some sensitive information. It may look like a legitimate e-mail from bank or other official institution.

17.3 Cyber Security and Threats

Cyber attacks are launched primarily for causing significant damage to a computer system or for stealing important information from an individual or from an organization. Cyber security is a collection of various technologies, processes and measures that reduces the risk of cyber attacks and protects organizations and individuals from computer based threats.

TYPES OF CYBER ATTACKS

Malware is a type of software designed through which the criminals gain illegal access to software and cause damage. Various types of cyber-attacks and their functions are given in**Table 17.2.**

Pharming

Pharming is a scamming practice in which malicious code is installed on a personal computer or server, misdirecting users to fraudulent web sites without their knowledge or permission. Pharming has been called "**phishing without a trap**”. It is another way hackers attempt to manipulate users on the Internet. It is a cyber-attack intended to redirect a website's traffic to a fake site.

Table 17.2 – Cyber Attacks and Functions

Cyber Security Threats

In recent years, most of the individuals and enterprises are facing problems due to the weaknesses inherent in security systems and compromised organizational infrastructures. Different types of Cyber Security Threats are categorized as below:

Phishing

Phishing is a type of computer crime used to attack, steal user data, including login name, password and credit card numbers. It occurs when an attacker targets a victim into opening an e-mail or an instant text message. The attacker uses phishing to distribute malicious links or attachments that can perform a variety of functions, including the extraction of sensitive login credentials from victims.

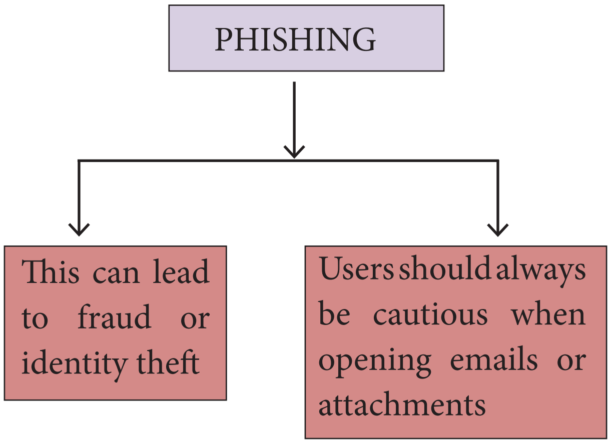

PHISHING
This can lead to fraud or identity theft
Users should always be cautious when opening emails or attachments

Figure 17.4 Diagrammatic representation of Phishing

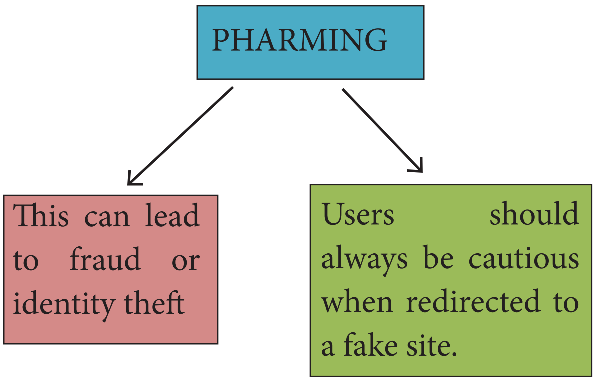

PHARMING
This can lead to fraud or identity theft
Users should always be cautious when redirected to a fake site.

Figure 17.5 Diagrammatic representation of Pharming

Man In The Middle (MITM)

Man-in-the-middle attack (MITM; also Janus attack) is an attack where the attacker secretly relays and possibly alters the communication between two parties who believe they are directly communicating with each other.

Example: Suppose Alice wishes to communicate with Bob. Meanwhile, Mallory wishes to intercept the conversation to overhear and optionally to deliver a false message to Bob.

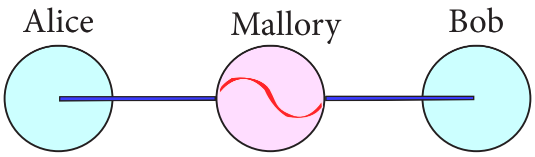


Figure 17.6 - An illustration of the Man-In-The-Middle attack

Cookies

A cookie (also called HTTP cookie, web cookie, Internet cookie, browser cookie, or simply cookie) is a small piece of data sent from a website and stored on the user's computer memory (Hard drive) by the user's web browser while the user is browsing internet. Cookies were designed to be a reliable mechanism for websites to remember stateful information (such as items added in the shopping cart in an online store) or to record the user's browsing activity (including clicking particular buttons, logging in etc.). They can also be used to remember arbitrary pieces of information that the user previously entered into form fields such as names, addresses, passwords, and credit card numbers. From the security point of view, if cookie data is not encrypted, any anonymous user (hacker) can access the cookie information and misuse it.

Web sites typically use cookies for the following reasons:

- To collect demographic information about who has visited the Web site.
- Sites often use this information to track how often visitors come to the site and how long they remain on the site.
- It helps to personalize the user's experience on the Web site.
- Cookies can help store personal information about users so that when a usersubsequently returns to the site, a more personalized experience is provided.

If you ever returned to a site and have seen your name mysteriously appear on the screen, it is because on a previous visit, you gave your name to the site and it was stored in a cookie. A good example of this is the way some online shopping sites will make recommendations to users based on their previous purchases. It helps to monitor advertisements. Cookies do not act maliciously on computer system. They are merely text files that can be deleted at any time.

Cookies cannot be used to spread viruses and they cannot access your hard drive. However, any personal information that you provide to a Web site, including credit card information, will most likely be stored in a cookie unless the cookie feature is explicitly turned off in your browser. This is the way in which cookies threaten privacy.

Firewall and Proxy Servers

A firewall is a computer network security based system that monitors and controls incoming and outgoing network traffic based on predefined security rules. A firewall commonly establishes a block between a trusted internal computer network and entrusted computer outside the network. **Figure 17.7** shows the working of firewall server.

A proxy server acts as an intermediary between the end users and a web server. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resources available from a different server. The proxy server examines the request, checks authenticity and grants the request based on that. Proxy servers typically keep the frequently visited site addresses in its cache which leads to improved response time. **Figure 17.8** shows the working of a proxy server.

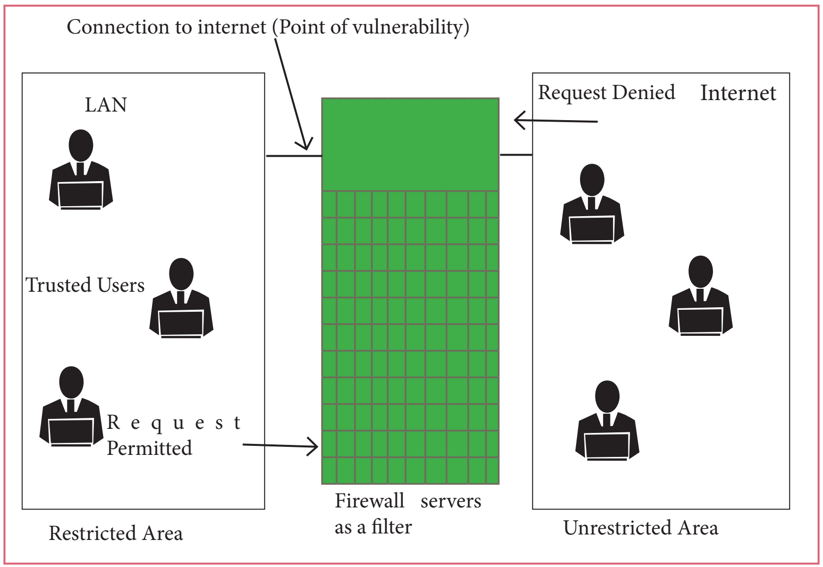


Figure 17.7 Firewall Server

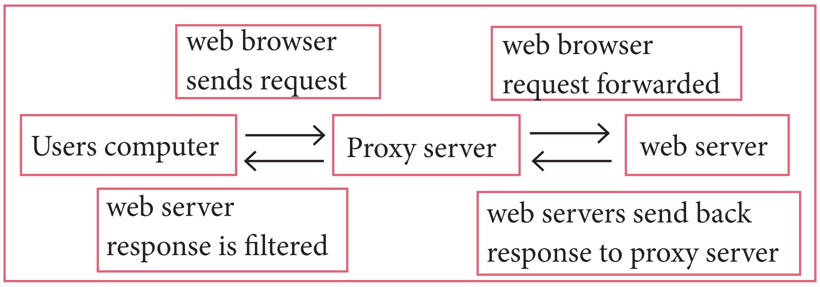

web browser sends request
web browser request forwarded
Users computer
Proxy server
web server
web server response is filtered
web servers send back response to proxy server

Figure 17.8 Working of Proxy server

Encryption and Decryption

Encryption and decryption are processes that ensure confidentiality that only authorized persons can access the information.

Encryption is the process of translating the plain text data (plaintext) into random and mangled data (called cipher-text).

Decryption is the reverse process of converting the cipher-text back to plaintext.Encryption and decryption are done by cryptography. In cryptography a key is a piece of information (parameter) that determines the functional output of a cryptographic algorithm.

Figure 17.9 shows the encryption and decryption process.

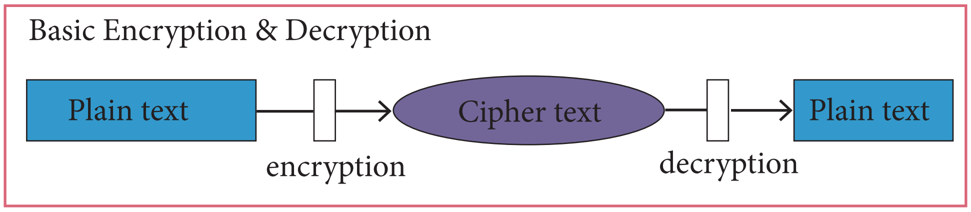


Figure 17.9 Encryption and Decryption

Encryption has been used by militaries and governments to facilitate secret communication. It is now commonly used in protecting information within many kinds of civilian systems. It is also used to protect data in communication system, for example data being transferred via networks (e.g. the Internet, ecommerce), mobile telephones, wireless microphones, wireless intercom systems, Bluetooth devices and bank automatic teller machines. There have been numerous reports of data in communication being intercepted in recent years. Data should also be encrypted when transmitted across networks in order to protect against the network traffic by unauthorized users.

17.4 INTRODUCTION TO INFORMATION TECHNOLOGY ACT

In the 21st century, Computer, Internet and ICT or e-revolution has changed the life style of the people. Today paper based communication has been substituted by e-communication. Accordingly we have new terminologies like cyber world, e-transaction, e-banking, e-return and e-contracts. Apart from positive side of e-revolution there is also negative side of computer, that is, the internet and ICT in the hands of criminals which has become a weapon of offence. Accordingly a new panel of members emerged to tackle the problems of cyber crimes in cyber space i.e. Cyber Law or Cyber Space Law or Information Technology Law or Internet Law.

In India Cyber law and IT Act 2000 , modified in 2008 are being articulated to prevent computer crimes. IT Act 2000 is an act to provide legal recognition for transactions carried out by means of **ElectronicData Interchange(EDI)** and other means of electronic communication. It is the primary law in India dealing with cybercrime and electronic commerce(e-Commerce). e-Commerce is electronic data exchange or electronic filing of information.

"Cyber law or Internet law is a term that encapsulates the legal issues related to use of the Internet.

PREVENTION

25% of cyber crime remains unsolved. To protect the information the following points are to be noted:

- Complex password setting can make your surfing secured.
- When the internet is not in use, disconnect it.
- Do NOT open spam mail or emails that have an unfamiliar sender.
- When using anti-virus software, keep it up-to-date.

- Awareness is the key to security.
- Information security is the immune system in the body of business.
- A check that does not bounce is called the Security Check. Do it every day before you leave!
- Do Your Part - Be Security Smart !!!
- Don't be Quick to Click… be wary when you shop online.
- Restart is Smart job
- Passwords are like toothbrushes. They are best when new and should never be shared.
- When you and your system part away, your system should be first off for the day.
- Your mind is a storage room of information, keep the door locked.
- _ a _ _word is not a PaSSword without Protect, Save and Secure!
- Link Link stop neglect….Think Think before connect…..

Evaluation

SECTION – A

Choose the correct answer

1. Which of the following deals with procedures, practices and values?
   a. piracy b. programs c. virus d. computer ethics

2. Commercial programs made available to the public illegally are known as
   a. freeware b. warez c. free software d. software

3. Which one of the following are self-repeating and do not require a computer program to attach themselves?
   a. viruses b. worms c. spyware d. Trojans

4. Which one of the following tracks a user visits a website?
   a. spyware b. cookies c. worms d. Trojans

5. Which of the following is not a malicious program on computer systems?
   a. worms d. Trojans c. spyware d. cookies

6. A computer network security that monitors and controls incoming and outgoing traffic is
   a. Cookies b.Virus c. Firewall d. worms

7. The process of converting cipher text to plain text is called
   a. Encryption b. Decryption c. key d. proxy server

8. e-commerce means
   a. electronic commerce b. electronic data exchange
   c. electric data exchange d. electronic commercialization.

9. Distributing unwanted e-mail to others is called.
   a. scam b. spam c. fraud d. spoofing

10. Legal recognition for transactions are carried out by
    a. Electronic Data Interchange b. Electronic Data Exchange
    c. Electronic Data Transfer d. Electrical Data Interchange

SECTION-B

Very Short Answers

1. What is harvesting?
2. What are Warez?
3. Write a short note on cracking.
4. Write two types of cyber attacks.
5. What is a Cookie?

SECTION-C

Short Answers

1. What is the role of firewalls?
2. Write about encryption and decryption.
3. Explain about proxy server.
4. What are the guidelines to be followed by any computer user?
5. What are ethical issues? Name some.

SECTION - D

Explain in detail

1. What are the various crimes happening using computer?
2. What is piracy? Mention the types of piracy? How can it be prevented?
3. Write the different types of cyber attacks.

Reference Books :

- Computer Network Security and Cyber Ethics by Joseph MiggaKizza
- "Investigating Cyber Law and Cyber Ethics: Issues, Impacts and Practices: 1" by Alfreda Dudley and James Braman

Unit V

COMPUTER ETHICS AND CYBER SECURITY

CHAPTER 18

Tamil Computing

18.1 Introduction

“ பிறநாட்டு நல்லறிஞர் சாத்திரங்கள்
தமிழ்மொழியிற் பெயர்த்தல் வேண்டும்;
இறவாத புகழுடைய புதுநூல்கள்
தமிழ்மொழியில் இயற்றல் வேண்டும்;
மறைவாக நமக்குள்ளே பழங்கதைகள்
சொல்வதிலோர் மகிமை இல்லை;
திறமான புலமையெனில் வெளிநாட்டோர்;
அதை வணக்கஞ் செய்தல் வேண்டும்.”

- மகாகவி பாரதி

Human civilization developed with the innovation of computer in the twentieth century. Computer development began as a early calculating tool and has now become a essential ingredient for gigantic growth for the existence of human life without computers.

It is true that any language will be outdated when it does not have the ability to adapt itself to the changing technologies. Tamil is a living language for thousands of years. Development of modern technologies, does not affect the growth of classical Tamil as it is ready to adopt the growing technological changes. **Tamil is not just a language, it is our identity, our life and our sense.**

“எங்கள் வாழ்வும், எங்கள் வளமும் மங்காத தமிழென்று சங்கே முழங்கு” – புரட்சி கவி.

18.2 Tamil in Internet

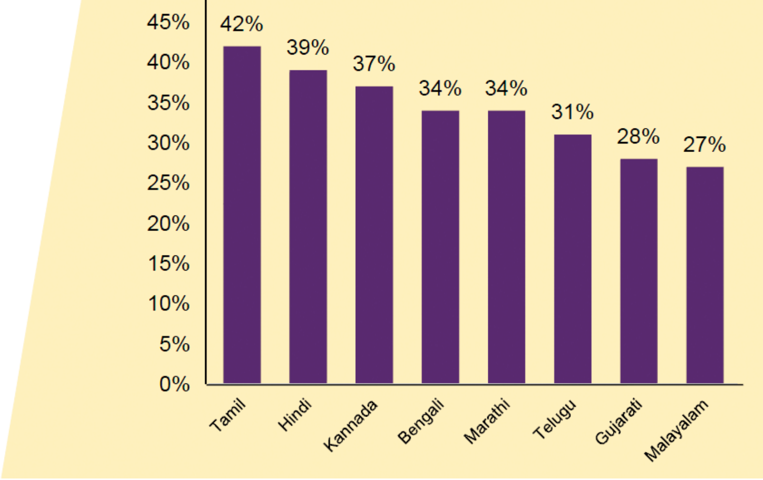
We know that the internet today plays a vital role in every man's life. Internet is the best information technological device, through which we get know information.

In 2017 a study conducted by KPMG a Singapore based organization along with google, reported that, Tamil topped the list, among the most widely used languages in India, where 42% are using the Internet in Tamil

68%[9] Internet users consider local language digital content to be more reliable than English

Currently, Tamil (42%[9]) has the highest internet adoption levels followed by Hindi and Kannada among the Indian language users

Language wise internet adoption levels for Indian language users – 2016[9]



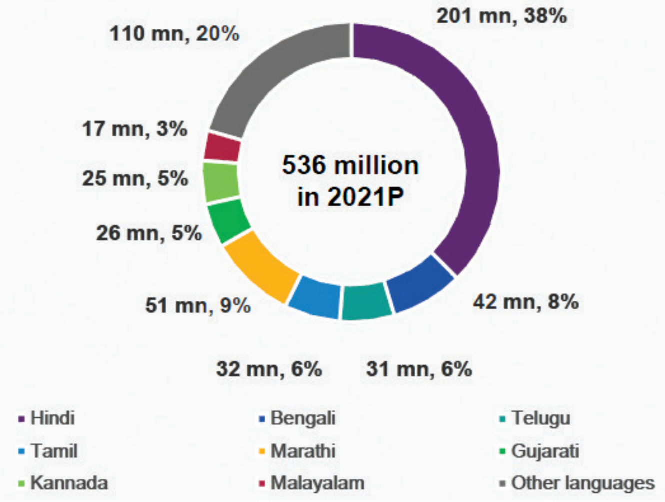
As per study, by 2021, 74% of people in India will access internet using Tamil and it will be in the top usage of Internet in India.

By 2021, the number of Hindi internet users will be more than English users

Marathi, Bengali, Tamil and Telugu internet users are expected to form 30%[12] of the total Indian language internet user base

Tamil and Kannada speakers have the highest propensity to adopt internet in future with the Indian language enablement of the ecosystem

Language wise internet users 2021P[12]



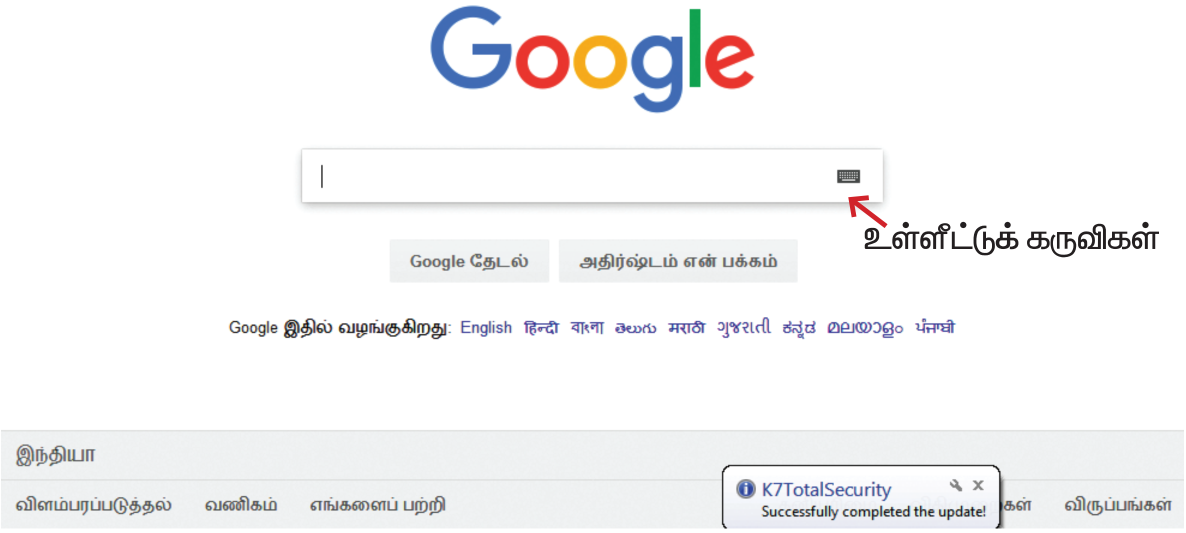
Language	Adoption propensity
Hindi	54%
Bengali	46%
Telugu	60%
Tamil	74%
Marathi	43%
Gujarati	43%
Kannada	74%
Malayalam	44%
Other languages	NA*

* Propensity to adopt internet in future has been ascertained for select 8 Indian languages

These statistical data will be useful to improve internet services in Tamil.

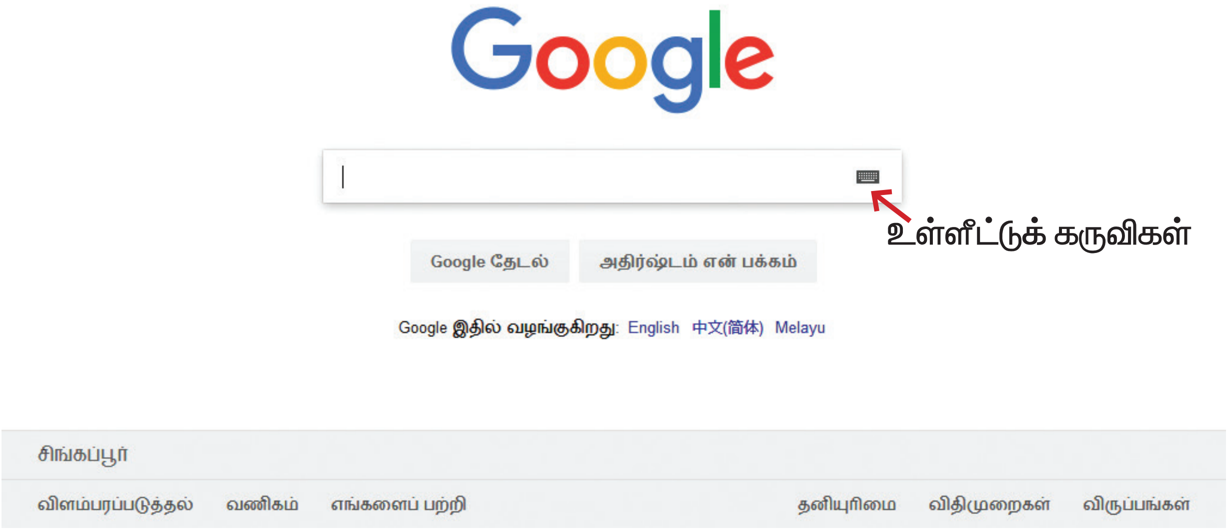
18.3 Search Engines in Tamil

The "Search Engines" are used to search any information from the cyber space. Although there are many search engines, but only a few of them are frequently in use. In the top ten search engines, Google, Bing and Yahoo take first three places respectively. Google and Bing provide searching facilities in Tamil, which means you can search everything through Tamil. The Google search engine gives you an inbuilt Tamil virtual keyboard.


Google
Google தேடல்
அதிர்ஷ்டம் என் பக்கம்
Google இதில் வழங்குகிறது: English हिन्दी বাংলা తెలుగు मराठी ગુજરાતી ಕನ್ನಡ മലയാളം ਪੰਜਾਬੀ
இந்தியா
விளம்பரப்படுத்தல்
வணிகம்
எங்களைப் பற்றி
K7TotalSecurity
Successfully completed the update!
கள்
விருப்பங்கள்

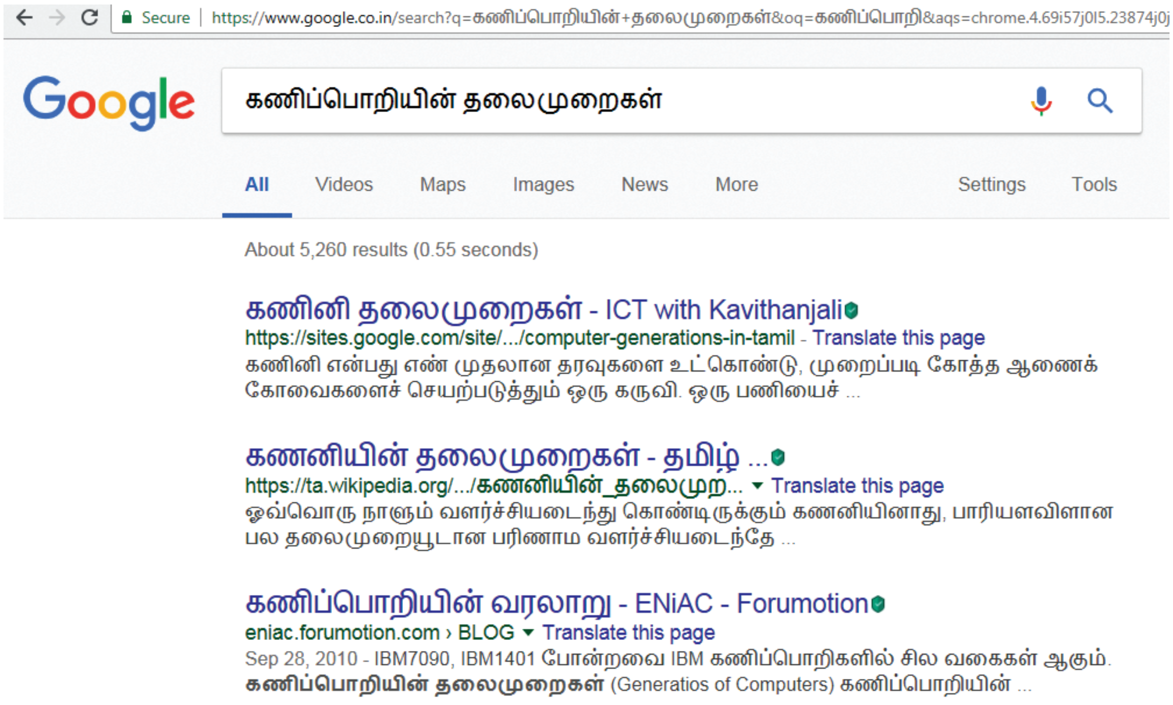
உள்ளீட்டுக் கருவிகள்

Figure 18.1(a) Google Search Engine (India)



உள்ளீட்டுக் கருவிகள்

Figure 18.1(b) Google Search Engine (Singapore)


Secure
https://www.google.co.in/search?q=கணிப்பொறியின்+தலைமுறைகள்&oq=கணிப்பொறி&aqs=chrome.4.69i57j0l5.23874j0j
Google
கணிப்பொறியின் தலைமுறைகள்
All
Videos
Maps
Images
News
More
Settings
Tools

About 5,260 results (0.55 seconds)

கணினி தலைமுறைகள் - ICT with Kavithanjali
https://sites.google.com/site/.../computer-generations-in-tamil - Translate this page
கணினி என்பது எண் முதலான தரவுகளை உட்கொண்டு, முறைப்படி கோத்த ஆணைக் கோவைகளைச் செயற்படுத்தும் ஒரு கருவி. ஒரு பணியைச் ...

கணனியின் தலைமுறைகள் - தமிழ் ...
https://ta.wikipedia.org/.../கணனியின்_தலைமுற... ▾ Translate this page
ஒவ்வொரு நாளும் வளர்ச்சியடைந்து கொண்டிருக்கும் கணனியினாது, பாரியளவிலான பல தலைமுறையூடான பரிணாம வளர்ச்சியடைந்தே ...

கணிப்பொறியின் வரலாறு - ENiAC - Forumotion
eniac.forumotion.com › BLOG ▾ Translate this page
Sep 28, 2010 - IBM7090, IBM1401 போன்றவை IBM கணிப்பொறிகளில் சில வகைகள் ஆகும்.
கணிப்பொறியின் தலைமுறைகள் (Generatios of Computers) கணிப்பொறியின் ...

Figure 18.2 Searching in Tamil

18.4 e – Governance:

Getting Government services through internet is known as e-Governance. Govt. of Tamilnadu has been giving its services through Internet. One can communicate with Govt. of Tamilnadu from any corner of the World. One can get important announcements, government orders, and government welfare schemes from the web portal of Govt. of. Tamilnadu.

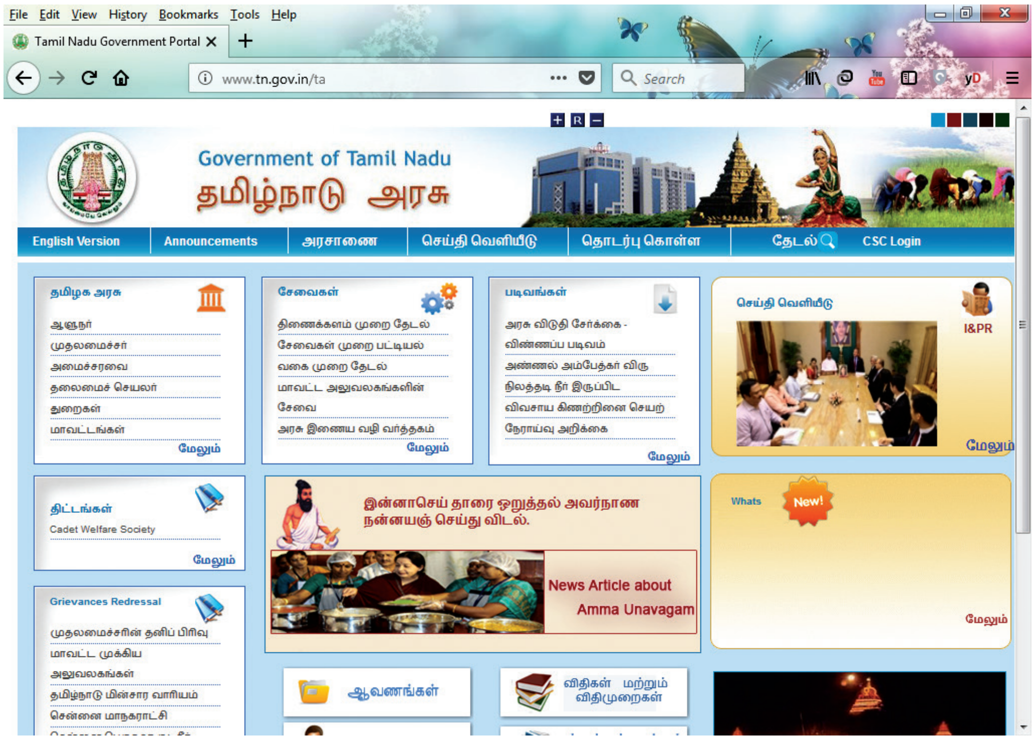


Figure 18.3 Official Website of Govt. of Tamilnadu

E-Governance through Tamil	Web Address
Official Website of Govt. of Tamilnadu	http://www.tn.gov.in/ta
Department of Agricultural Engineering	http://www.aed.tn.gov.in/
Department of Environment	http://www.environment.tn.nic.in/
Directorate of Govt. Examinations	http://www.dge.tn.nic.in/
Tamilnadu Health Department	http://www.tnhealth.org/
Tamilnadu Micro, Small and Medium Enterprises Department	http://www.msmeonline.tn.gov.in/
Rural Development and Panchayat Raj Department	http://www.tnrd.gov.in/
Backward, Most Backward and Minorities Welfare Department	http://www.bcmbcmw.tn.gov.in/
Tamilnadu Forest Department	https://www.forests.tn.gov.in/
Hindu Religious and Charitable Endowments Department.	http://www.tnhrce.org/
Tamil Nadu Public Service Commission (TNPSC)	http://www.tnpsc.gov.in/tamilversion/index.html
Official Website of Govt. of Srilanka	https://www.gov.lk/index.php

Outside India, Government of Srilanka provides all their services through the official website in Tamil.

18.5 e-Library

E-Libraries are portal or website of collection of e-books. Tamil e-Library services provide thousands of Tamil Books as ebooks mostly at free of cost. It is the most useful service to Tamil people who live far away from their home land.

Tamil e-Library	Website address
Tamilnadu School Education and Teacher Education Training Textbooks and Resource Books	http://www.textbooksonline.tn.nic.in/
Tamil Virtual Academy	http://www.tamilvu.org/library/libindex.htm
Connemara Public Library	http://connemarapubliclibrarychennai.com/Veettukku_oru_noolagam/index.html
Tamil Digital Library	http://tamildigitallibrary.in/
Chennai Library	http://www.chennailibrary.com/
Thamizhagam	http://www.thamizhagam.net/parithi/parithi.html

Project Madurai	http://www.projectmadurai.org/pmworks.html
Old Books and Manuscripts	http://www.tamilheritage.org/old/text/ebook/ebook.html
Noolaham	http://www.noolaham.org/wiki/index.php/
Anna Centenary Libraray	http://www.annacentenarylibrary.org/

18.6 Tamil Typing and Interface software

Tamil is mostly used to type documents in word processors and search information on the internet. Typing Tamil using Tamil interface software is a familiar one among the different methods of typing. This is the simplest method of typing Tamil in both Computer and Smart phones.

18.6.1 Familiar Tamil Keyboard Interface:

- NHM Writer, E-Kalappai and Lippikar – are familiar Tamil keyboard interfaces software that is used for Tamil typing which works on Tamil Unicode, using phonetics.
- Sellinam and Ponmadal – are familiar Tamil keyboard layouts that works on Android operating system in Smart phone using phonetics.

Version 3.0
eKalappai
தமிழா
http://thamizha.com
Created by: Mugunth, Sethu, Vijay Gupta, Arunan, Venkatesh, Gunasekar...

Figure 18.4 eKalappai Opening screen

18.7 Tamil Office Automation Applications

Famous Office automation software like Microsoft Office, Open Office etc., provides complete Tamil interface facility. These softwares are downloadable and installed in your computer. After installation, your office automation software environment will completely change to Tamil. Menu bars, names of icons, dialog boxes will be shown in Tamil. Moreover, you can save files with Tamil names and create folders with Tamil names.

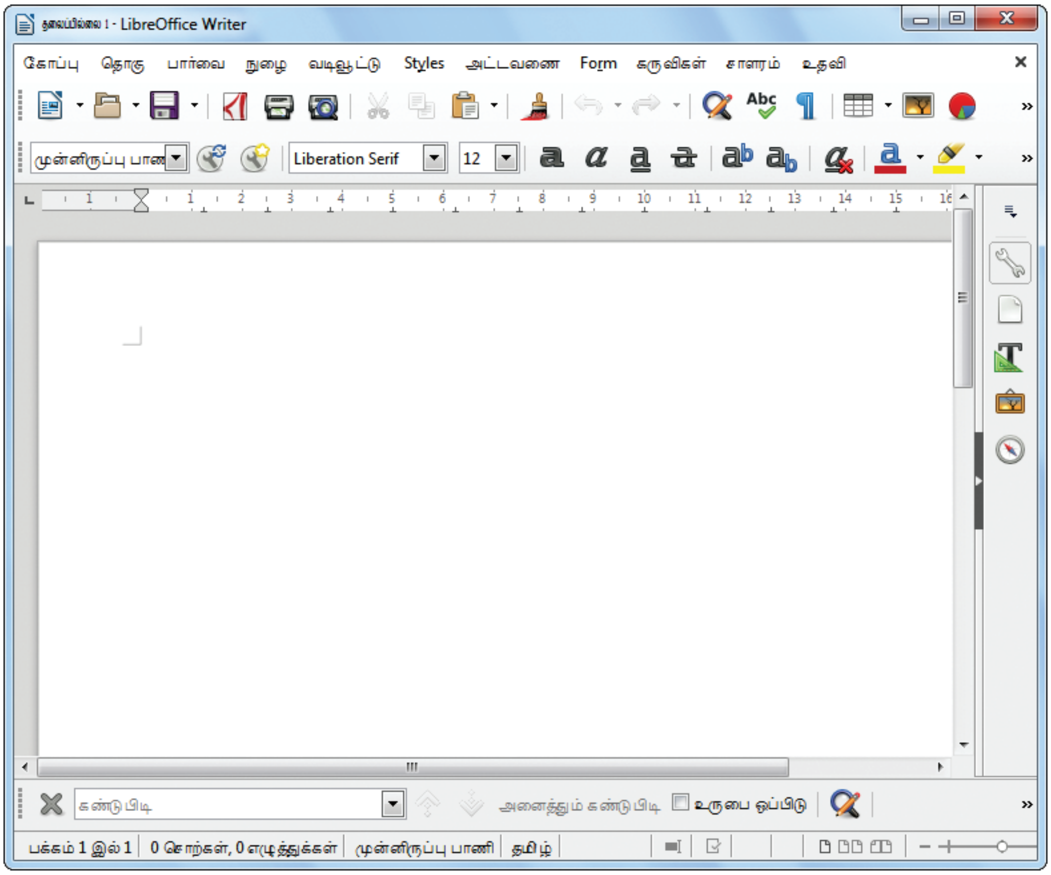


Figure 18.5 Libra Office Writer Environments in Tamil

Apart from that Tamil Libra Office, Tamil Open Office, Azhagi Unicode Editor, Ponmozhi, Menthamiz, Kamban, Vani are office automation software working exclusively for Tamil. You can use these applications to work completely in Tamil.

18.8 Tamil Translation Applications

Thamizpori (தமிழ்பொறி) is a Tamil tranlation application having more than 30000 Tamil words equalent to English words. Using this application, we can translate small english sentences into Tamil. Google also gives an online translation facility, using this online facility we can translate from Tamil to any other language and vice versa.

18.9 Tamil Programming Language

Programming languages to develop software in computers and smart phones are available only in English. Now, efforts are taken to develop programming languages in Tamil. Based on Python programming language, the first Tamil programming language "Ezhil" (எழில்) is designed. With the help of this programming language, you can write simple programs in Tamil.

18.10 Tamil Information Interchange Coding Systems

TSCII (Tamil Script Code for Information Interchange)

Computers handle data and information as binary system. Every data should be converted into binary when it is fed into a computer system. You have learnt about all these things in the first unit of this text book. Computers use ASCII encoding system to handle data and information. The ASCII encoding system is applicable only for handling English language. Therefore, TSCII (Tamil Script Code for Information Interchange) is the first coding system to handle our Tamil language in an analysis of an encoding scheme that is easily handled in electronic devices, including non-English computers. This encoding scheme was registered in IANA (Internet Assigned Numbers Authority) a unit of ICANN.

ISCII (Indian Script Code for Information Interchange)

This is one of the encoding schemes specially designed for Indian languages including Tamil. It was unified with Unicode.

Unicode:

Unicode is an encoding system, designed to handle various world languages, including Tamil. Its first version 1.0.0 was introduced in October 1991. When Unicode was introduced it could handle nearly 23 languages including Tamil. Among the various encoding scheme, Unicode is the best suitable to handle Tamil.

18.11 Tamil Operating System

An operating system is needed to access electronic systems such as computer and smart phone. Microsoft Windows is very popular operating system for personal computers. Linux is another popular open source operating system. Operating systems are used to access a computer easily. An operating system should be easy to work and its environment should be in an understandable form. Thus, all operating systems used in computers and smart phones are offered in Tamil environment.

Windows Tamil Environment interface should be downloaded and installed from the internet. It displays all window elements such as Taskbar, desktop elements, names of icons, commands in Tamil.

18.12 Organisation and projects to develop Tamil

Tamil Virtual Academy:

With the objectives of spreading Tamil to the entire world through internet, Tamil Virtual University was established on 17th February 2001 by the Govt. of Tamilnadu. Now, this organisation functions with the name of "Tamil Virtual Academy". It offers different courses in Tamil language, Culture, heritage etc., from kindergarten to under graduation level.

Website: http://www.tamilvu.org/index.php

Tamil Language Council, Singapore

With the objectives of promoting the awareness and greater use of Tamil among the Singaporeans, in 2001 the council of Tamil Language was formed by the ministry of Information Communications and Arts, Govt. of Singapore. The council is called as "வளர்தமிழ் இயக்கம்".

வளர்தமிழ் இயக்கம்
Tamil Language Council

Website: http://tamil.org.sg/ta

Madurai Project

Project Madurai is an open and voluntary initiative to collect and publish free electronic editions of ancient tamil literary classics. This means either typing-in or scanning old books and archiving the text is one of the most readily accessible formats for use on all popular computer platforms.

Since its launch in 1998, Project Madurai etexts released are in Tamil script form as per TSCII encoding. Since 2004 we have started releasing etexts in Tamil unicode as well.

Web Site: http://www.projectmadurai.org/

Tamil Wikipedia:

Wikipedia is a open source encyclopedia where any person can write an article about any subject. There are more than One lakh articles in Tamil Wikipedia.

Web Site: https://ta.wikipedia.org/

In order to make Tamil as a living language, it is the duty of every Tamilian to actively use Tamil in the development of technology.

Points to Remember:

- Tamil topped the list of the most widely used regional languages in India by the end of 2016, among 42% are using the Internet.
- Google and Bing provide searching facilities in Tamil.
- Getting Government services through internet is known as e-Governance.
- Tamil e-Library services provide thousands of Tamil Books as ebooks mostly at free of cost.
- Thamizpori (தமிழ்பொறி) is a Tamil tranlation application having more than 30000 Tamil words equalent to English words.
- The first Tamil programming language is "Ezhil" (எழில்)
- Unicode is an encoding system, designed to handle various world languages, including Tamil.
- Windows Tamil Environment interface should be downloaded and installed from internet.

Evaluation

Very Short Answers

1. List the search engines supported by Tamil language.
2. What are the keyboard layouts used in Android?
3. Write a short note about Tamil Programming Language.
4. What is TSCII?
5. Write a short note on Tamil Virtual Academy.

****

FE8KPQ

FEC8DH

GLOSSARY

WORD	*MEANING*
Paradigm	Organizing principle of a program.
Abstraction	Abstraction refers to showing only the essential features without revealing background details
Modularity	Designing a system that is divided into a set of functional units (named modules) that can be composed into a larger application.
Base class	A class whose properties are inherited by other newly created classes .Also called as parent class
Derived class	A class which inherits the properties of the base class. Also called as child class or subclass.
Class	Class represents a group of similar objects that share common properties
Object	Identifiable entity with some characteristics and behaviour
Encapsulation	Mechanism by which the data and function sare bound together into a single unit
Inheritance	Process of creating new classes called derived classes, from the existing or base classes.
Signature	Number of argument and type of argument
Polymorphism	many forms
Default argument	Initializing the argument with a value
Base Class:	A class from which another class inherits (Also called Super class or parent class)
Derived Class:	A class inheriting properties from another class. (Also called Sub class)
Inheritance	The process of one class to inherit properties from another class
Inheritance Hierarchy (Inheritance Path):	The chain depicting relationship between a base class and the derived class (Also called Derivation Hierarchy)
Visibility mode	The public, private or protected specifier that controls the visibility and availability of a member in a class
Vulnerability	The possibility of being attacked or harmed.
Ethics	Moral principles that govern a person's behaviour or the conducting of an activity.

Cyber	Characteristic of the culture of computers, information technology, and virtual reality.
Computer Crime	Computer crime is an intellectual crime to manipulate computer system.
Authenticity	The quality of being real or true.
Sabotage	Deliberately destroy, damage, or obstruct.
Perpetrator	A person who carries out a harmful, illegal, or immoral act.
Software Piracy	Software Piracy is the copyright violation of software created originally by one person and illegally used by someone else.
Hacking	Hacking is gaining unauthorized access to computer system without the owner's permission.
Cracking	Cracking is gaining unauthorized access to computer systems to commit a crime, such as stealing the code to make a copy-protected program run thus denying service to legitimate users.
Malicious	Intentionally doing harm.
Freeware	Freeware is a software available free of charge.
Shareware	Shareware is a software that is distributed free of charge on a trial basis for a limited time.
Phishing	Phishing is a term used to describe a malicious individual or group of individuals who scam users by sending e-mails or creating web pages that are designed to collect an individual's online bank, credit card, or other login information.
Fraudulent	Dishonest, cheating, swindling, corrupt, criminal, illegal, unlawful.
Anonymous	Unnamed, nameless, unidentified, unspecified.
Cookies	Cookies are messages that web servers pass to your web browser when you visit Internet sites
Tampering	Interfere in order to cause damage.
Immune	Resistant to a particular infection or toxin.
Firewall	A firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules.
Proxy server	A proxy server is a gateway between a local network and a larger-scale network such as the Internet. Proxy servers provide increased performance and security.
Encryption	Encryption is the process of encoding a message or information so that only authorized users can decrypt it
Decryption	Decryption is theprocess of decoding the encrypted text by converting it back into normal text.

Instructions:

1. Ten exercises from C++ are practiced in the practical classes
2. One question from C++ with internal choice
3. Distribution of Marks

I. Internal Assessment: 5 Marks	
Record Book	5 Marks

II. External Assessment: 15 Marks	
(a) C++ Program coding	10 Marks
(b) Execution & Output	5 Marks
Total	**20 Marks**

INDEX

CS1 - GROSS SALARY

CS-1 **Write a C++ program to input basic salary of an employee and calculate its Gross salary according to following:**

Basic Salary <25000 : HRA = 20%, DA = 80%
Basic Salary >= 25000 : HRA = 25%, DA = 90%
Basic Salary >= 40000 : HRA = 30%, DA = 95%

Coding

```
#include <iostream>
using namespace std;
int main()
{
float basic, gross, da,hra;
cout<<"Enter basic salary of an employee: ";
cin>>basic;
if (basic <25000)
{
da = basic *80/100;
hra= basic *20/100;
}
else if (basic >=25000 && basic<40000)
{
da = basic *90/100;
hra= basic *25/100;
}
else if (basic>=40000)
{
da = basic *95/100;
hra= basic *30/100;
}
gross= basic +hra+ da;
cout<< "\n\t Basic Pay ..............."<< basic<<endl;
cout<< "\t Dearness Allowance ......." << da <<endl;
cout<< "\t House Rent Allowance......"<< hra <<endl;
  cout<< "\t -----------------------------------------"<<endl;
  cout<< "\t Gross Salary.............."<<gross <<endl;
cout<< "\t ------------------------------------------"<<endl;
return 0;
}
```

Output

```
Enter basic salary of an employee: 25000
                      Basic Pay        : 25000
              Dearness Allowance       : 22500
           House Rent Allowance        :   6250
                                 ------------------------
                 Gross Salary          : 53750
                                 ------------------------
```

CS2 - PERCENTAGE

CS-2 **Write a C++ program to check percentage of a student and display the division (distinction, first, second, third or fail) scored using switch case**

Percentage	Division
>=80	Distinction
>=60 and <80	First division
>=50 and <60	Second Division
>=40 and <50	Third Division
<40	Fail

Coding

```
#include <iostream>
using namespace std;
int main()
{
float percent;
int x;
cout<<"Enter your percentage: ";
cin>>percent;
cout<<"You scored "<<percent<<"%"<<endl;
   x = percent/10;
switch (x)
{
case 10:
case 9:
```

```
case 8:
cout<<"You have passed with Distinction";
break;
case 7:
case 6:
cout<<"You have passed with First division";
break;
case 5:
cout<<"You have passed with Second division";
break;
case 4:
cout<<"You have passed with Third division";
break;
default:
cout<<"Sorry: You have failed";
}
return 0;
}
```

Output 1

```
Enter your percentage: 79
You scored 79%
You have passed with First division
```

Output 2

```
Enter your percentage: 39
You scored 39%
Sorry: You have failed
```

CS3 - PALINDROME

CS-3 **Write a C++ program to enter any number and check whether the number is palindrome or not using while loop.**

Coding

```
#include <iostream>
using namespace std;
int main()
{
int n,num, digit, rev =0;
cout<<"Enter a positive number: ";
cin>>num;
   n =num;
while (num)
{
digit=num%10;
rev=(rev *10)+ digit;
num=num/10;
}
cout<<" The reverse of the number is: "<< rev <<endl;
if (n == rev)
cout<<" The number is a palindrome";
else
cout<<" The number is not a palindrome";
return 0;
}
```

Output 1

```
Enter a positive number to reverse: 1234
 The reverse of the number is: 4321
 The number is not a palindrome
```

Output 2

```
Enter a positive number to reverse: 1221
The reverse of the number is: 1221
The number is a palindrome
```

CS4 - NUMBER CONVERSION

CS-4 Using do while loop create the following menu based C++ program

1.Convert a Decimal to binary number
2.Convert a binary number to Decimal
3. Exit

Depending on the choice accept the value and display the result .The program should continue till the userselect the third option

Coding

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int dec,d,i,temp,ch;
    long int bin;
do
{
    dec=bin=d=i=0;
    cout<<"\n\n\t\tMENU\n1. Decimal to Binary number\n2.Binary to Decimal number\
    n3.Exit\n";
    cout <<"Enter your choice(1/2/3)";
    cin>>ch;
    switch (ch)
    {
            case 1:  cout << "Enter a decimal number: ";  cin >> dec;
                    temp=dec;
    while (dec!=0)
    {
    d = dec%2;
    bin += d * pow(10,i);
    dec /= 2;
    i++;
}
    cout << temp << " in decimal = " << bin << " in binary" << endl ;break;
    case 2: cout << "Enter a binary number: ";    cin >> bin;
            temp=bin;
            while (bin!=0)
    {
    d = bin%10;
    dec += d*pow(2,i);
    bin /= 10;
    i++;
}
    cout << temp << " in binary = " <<dec << " in decimal";
    break;
    case 3: break;
    default : cout<<"Invalid choice";
 }
} while (ch!=3);
return 0;
}
```

Output 1

```
    MENU
1.Decimal to Binary number
```

```
2.Binary to Decimal number
3.Exit
Enter your choice(1/2/3)1
Enter a decimal number: 23
23 in decimal = 10111 in binary
   MENU
1.Decimal to Binary number
2.Binary to Decimal number
3.Exit
Enter your choice(1/2/3)2
Enter a binary number: 11001
11001 in binary = 25 in decimal
   MENU
1.Decimal to Binary number
2.Binary to Decimal number
3.Exit
Enter your choice(1/2/3)3
```

Output 2

```
   MENU
1.Decimal to Binary number
2.Binary to Decimal number
3.Exit
Enter your choice(1/2/3)4
Invalid choice
   MENU
1.Decimal to Binary number
2.Binary to Decimal number
3.Exit
Enter your choice(1/2/3)3
```

CS5 - FIBONACCI PRIME SERIES

CS-5 **Write a C++ program using a user defined function to generate the Fibonacci series till n terms and print if each term is prime or Composite.**

Coding

```
#include <iostream>
#include <stdlib.h>
```

```
using namespace std;
void Primechk (int a )
{ int j;
     if ( a == 0 || a == 1 )
    { cout<< "  NEITHER PRIME NOR COMPOSITE ";}
     else
{
    for (j = 2 ; j<a; j++)
    {           if (a%j==0)
    { cout<< "\tCOMPOSITE" ;
    break ;
}
    }
     if ( a==j )
     cout<< "\tPRIME" ;
    }
}
    void  fibo ( int n )
    { int a = -1 , b = 1 ,c=0 ;
    for ( int i = 1 ; i <= n ; i++)
    {
                cout<<endl;
    c = a + b ;
    cout<<c;
    Primechk(c);
    a = b;
    b = c ;
      }
    }
 int main ()
 {
    int n ;
    cout << " ENTER THE NUMBER OF REQUIRED FIBO TERMS " ;
    cin >> n ;
    cout<< "\n\tFIBONACCI SERIES\n " ;
    fibo (n) ;
return 0;
  }
```

Output

```
ENTER THE NUMBER OF TERMS 10
    FIBONACCI SERIES
0  NEITHER PRIME NOR COMPOSITE
1  NEITHER PRIME NOR COMPOSITE
1  NEITHER PRIME NOR COMPOSITE
2       PRIME
3       PRIME
5       PRIME
8       COMPOSITE
13       PRIME
21       COMPOSITE
34       COMPOSITE
```

CS6 - INSERT / DELETE ELEMENTS IN AN ARRAY

CS-6 **Write a menu driven C++ program to Insert and Delete elements in a single dimension array of integers and print the array after insertion or deletion**

Coding

```
#include<iostream>
using namespace std;
int a[20],b[20],c[40];
int m,n,p,val,i,j,key,pos,temp;
/*Function Prototype*/
void display();
void insert();
void del();
int main()
{
int choice;
cout<<"\nEnter the size of the array elements:\t";
cin>>n;
cout<<"\nEnter the elements for the array:\n";
for (i=0;i<n;i++)
```

```
{
cin>>a[i];
}
do {
cout<<"\n\n--------Menu-----------\n";
cout<<"1.Insert\n";
cout<<"2.Delete\n";
cout<<"3.Exit\n";
cout<<"-----------------------";
cout<<"\nEnter your choice:\t";
cin>>choice;
switch (choice)
{
    case 1: insert();
    break;
    case 2: del();
    break;
    case 3:break;
    default :cout<<"\nInvalid choice:\n";
}
} while (choice!=3);
return 0;
}
void display()//displaying an array elements
{
    int i;
    cout<<"\nThe array elements are:\n";
    for(i=0;i<n;i++)
{
    cout<<a[i]<<" ";
}
}//end of display()
    void insert()//inserting an element in to an array
{
    cout<<"\nEnter the position for the new element:\t";
    cin>>pos;
    cout<<"\nEnter the element to be inserted :\t";
    cin>>val;
    for (i=n; i>=pos-1; i--)
{
    a[i+1]=a[i];
}
a[pos-1]=val;
      n=n+1;
display();
}//end of insert()
void del()//deleting an array element
{
    cout<<"\n Enter the position of the element to be deleted:\t";
    cin>> pos;
    val= a [pos];
    for (i= pos;i<n-1;i++)
{
    a[i]=a[i+1];
}
    n=n-1;
cout<<"\nThe deleted element is = "<<val;
display();
}//end of delete()
```

Output

```
Enter the size of the array elements:   5
Enter the elements for the array:
1
2
3
4
5
--------Menu-----------
1.Insert
2.Delete
3.Exit
-----------------------
```

```
Enter your choice:     1
Enter the position for the new element: 3
Enter the element to be inserted :     26
The array elements are:
1  2  26  3  4  5
--------Menu-----------
1.Insert
2.Delete
3.Exit
-----------------------
Enter your choice:     2
Enter the position of the element to be deleted:      2
The deleted element is = 2
The array elements are:
1 3 26 4 5
--------Menu-----------
1.Insert
2.Delete
3.Exit
--------------------------
Enter your choice:     3
---------------------------
```

CS 7 - Boundary Element of a Matrix

CS-7 Write a C++ program to print boundary elements of a matrix

Coding

```
#include <iostream>
using namespace std;
void printBoundary (int a[][10], int m, int n)
{
for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++)
{
if (i==0|| j==0||i==m-1||j==n-1)
cout<<a[i][j]<<" ";
else
cout<<"  ";
}
cout <<endl ;
}
}
// Driver code
int main()
{
    int a[10][10] ,i,j,m,n;
    cout<<"Enter more than 3 number of rows and columns"<<endl;
    cin>>m>>n;
              for (i=0;i<m;i++)
    {
              for (j=0;j<n;j++)
    {
              cout<<"enter the value for array["<<i+1<<"]"<<"["<<j+1<<"] :";
              cin>>a[i][j];
     }
}
system("cls");
    cout<<"\n\nOriginal Array\n";
    for (i=0;i<m;i++)
{
    for (j=0;j<n;j++)
    {
              cout<<a[i][j]<<" ";
     }
    cout<<endl;
}
    cout<<"\n\n The Boundry element\n";
    printBoundary(a, m, n);
return 0;
}
```

Output

```
Enter more than 3 number of rows and columns
4 4
```

```
enter the value for array[1][1] :1
enter the value for array[1][2] :2
enter the value for array[1][3] :3
enter the value for array[1][4] :4
enter the value for array[2][1] :5
enter the value for array[2][2] :6
enter the value for array[2][3] :7
enter the value for array[2][4] :8
enter the value for array[3][1] :9
enter the value for array[3][2] :0
enter the value for array[3][3] :1
enter the value for array[3][4] :2
enter the value for array[4][1] :3
enter the value for array[4][2] :4
enter the value for array[4][3] :5
enter the value for array[4][4] :6
Original Array
1 2 3 4
5 6 7 8
9 0 1 2
3 4 5 6
The Boundary element
1   2   3   4
5           8
9           2
3   4   5   6
```

CS8 - ABC PUBLISHERS

CS-8 Define a class named Publisher in C++ with the following descriptions

private members

Bookno integer

Title 20 characters

Author 10 characters

price float

Totamt float

Define a member function called calculate() to calculate the number of copies and the price and return the total amount.

Public members

- A default constructor function to initialize all data members.The book number must be automatically generated staring from 1001
- Readdata() function to accept values for Title,Author and price.Get the number of copies from the user and invoke calculate().
- Display data () function display all the data members in the following format

```
                    ABC PUBLISHERS
                  ~~~~~~~~~~~~~~~~~
                       INVOICE
                      ~~~~~~~~~
==================================
Book Number          :
Title          :
Author Name          :
Price Per Book:
Total Amount         :
==================================
```

Coding

```
#include<iostream>
#include<stdlib.h>
using namespace std;
int id=1001;
class Publisher
{
int Bookno;
char Title[20];
char Author [10];
float Price;
float Totamt;
float calculate (int);
public:
Publisher()
```

```
{Bookno=id;
Title[0]='\0';
Author[0]='\0';
    Price=0;
Totamt=0;
id++;
}
void Readdata();
void Displaydata();
};
void Publisher::Readdata()
{
int nocopies;
cout<<"Enter the Title name ";cin>>Title;
cout<<"Enter the Author name ";cin>>Author;
cout<<"Enter the Price ";cin>>Price;
cout<<"Enter the Number of copies ";cin>>nocopies;
Totamt=calculate(nocopies);
}
float Publisher::calculate(int x)
{
   return x*Price;
}
void Publisher::Displaydata()
{
cout<<"\n\t\tABC PUBLISHERS\n";
cout<<"\t\t~~~~~~~~~~~~~~~~\n";
cout<<"\t\t  INVOICE\n";
cout<<"\t\t  ~~~~~~~~\n";
cout<<"\n===================================\n";
cout<<" Book Number : "<<Bookno<<endl;
cout<<"Title                    : "<<Title<<endl;
cout<<"Author Name   : "<<Author<<endl;
cout<<"Price Per Book : "<<Price<<endl;
cout<<"Total Amount  : "<<Totamt<<endl;
cout<<"\n===================================\n";
```

```
}
int main()
{
   int n,i;
   Publisher p[10];
   cout<<"Enter the number of object to be created";cin>>n;
   for (i=0;i<n;i++)
   p[i].Readdata();
   for (i=0;i<n;i++)
   p[i].Displaydata();
   return 0;
}
```

Output

```
Enter the number of object to be created2
Enter the Title name C++Programming
Enter the Author name Balaguru
Enter the Price 500
Enter the Number of copies 3
Enter the Title name CoreJava
Enter the Author name Xavier
Enter the Price 250
Enter the Number of copies 5
                                        ABC PUBLISHERS
                                        ~~~~~~~~~~~~~~
                                           INVOICE
                                           ~~~~~~~
===================================
Book Number   : 1001
Title                 : C++Programming
Author Name           : Balaguru
Price Per Book  : 500
Total Amount    : 1500
==================================
       ABC PUBLISHERS
       ~~~~~~~~~~~~~~
       INVOICE
```

```
            ~~~~~~~
====================================
Book Number    : 1002
Title                  : CoreJava
Author Name            : Xavier
Price Per Book  : 250
Total Amount    : 1250
====================================
```

CS9 - EMPLOYEE DETAILS USING CLASS

CS-9 **Create a C++ program to create a class employee containg the following members in public.**

Public members

eno	integer
name	20 characters
des	20 characters

member Function

void get() to accept values for all data members

Declare the derived class called Salary which contain the following details.

Public members

bp, hra, da, pf, np	float

member Function

void get1() to accept values for bp,hra,da and pf and invoke calculate()

calculate() calculate the np by adding bp,hra,da subtracting pf

display() Display all the details

Create the derived class object and read the number of employees.Call the function get(),get1() for each employee and display the details

Coding

```
#include<iostream>
using namespace std;
class emp{
public:
int eno;
char name[20], des[20];
void get(){
```

```
cout<<"Enter the employee number:";
cin>>eno;
cout<<"Enter the employee name:";
cin>>name;
cout<<"Enter the designation:";
cin>>des;
}
};
class salary :public emp
{
float bp,hra, da,pf,np;
public:
void get1()
{
cout<<"Enter the basic pay:";
cin>>bp;
cout<<"Enter the HouseRent Allowance:";
cin>>hra;
cout<<"Enter the Dearness Allowance :";
cin>>da;
cout<<"Enter the Provident Fund:";
cin>>pf;
}
void calculate()
{
np=bp+hra+ da -pf;
}
void display()
   {
cout<<eno<<"\t"<<name<<"\t"<<des<<"\t"<<bp<<"\t"<<hra<<"\t"<<da<<"\t"<<pf<<"\
t"<<np<<"\n";
}
};
int main(){
int i, n;
char ch;
salary s[10];
cout<<"Enter the number of employee:";
cin>>n;
for (i =0; i < n; i++){
```

```
s[i].get();
s[i].get1();
s[i].calculate();
}
    cout<<"\n\t\t\tEmployee Details\n";
    cout<<"\ne_no \t e_name\t des \t bp \t hra \t da \t pf \t np \n";
    for (i =0; i < n; i++){
    s[i].display();
}
return 0;
}
```

Output

```
Enter the number of employee:2
Enter the employee number:1201
Enter the employee name:Ramkumar
Enter the designation:Engineer
Enter the basic pay:50000
Enter the House Rent Allowance:10000
Enter the Dearness Allowance :5000
Enter the Provident Fund:1000
Enter the employee number:1202
Enter the employee name:Viswanathan
Enter the designation:Engineer-Tech
Enter the basic pay:40000
Enter the House Rent Allowance:9000
Enter the Dearness Allowance :4500
Enter the Provident  Fund:1000
                    Employee Details
e_noe_name                des              bp       hra         da       pf       np
1201   Ramkumar      Engineer       50000   10000          5000    1000    64000
1202   Viswanathan   Engineer-Tech          40000   9000          4500    1000    52500
```

CS10 -STUDENT DETAILS

CS-10 Write a C++ program to create a class called Student with the following details

Protected member

Rno   integer

Public members

void Readno(int);   to accept roll number and assign to Rno

void Writeno(); To display Rno.

The class Test is derived Publically from the Student class contains the following details

Protected member

Mark1   float

Mark2   float

Public members

void Readmark(float,float);   To accept mark1 and mark2

void Writemark();   To display the marks

Create a class called Sports with the following detail

Protected members

score   integer

Public members

void Readscore(int);   To accept the score

void Writescore();   To display the score

The class Result is derived Publically from Test and Sports class contains the following details

Private member

Total   float

Public member

void display()   assign the sum of mark1 ,mark2,score in total.

invokeWriteno(),Writemark() and Writescore() .Display the total also.

Coding

```
#include<iostream>
using namespace std;
class Student
{
   protected:
   int Rno;
   public:
   void Readno(int r)
   {
   Rno=r;
   }
   void Writeno()
   {
            cout<<"\nRoll no : "<<Rno;
   }
};
class Test :public Student
```

```
{
    protected:
    float Mark1,Mark2;
    public:
    void Readmark (float m1,float m2)
    {
            Mark1=m1;
            Mark2=m2;
    }
void Writemark()
{
cout<<"\n\n\tMarks Obtained\n ";
cout<<"\n Mark1       : "<<Mark1;
cout<<"\n Mark2       : "<<Mark2;
}
};
class Sports
{
protected:
int score;// score = Sports mark
public:
void  Readscore (int s)
{
score=s;
}
void  Writescore()
    {
cout<<"\n Sports Score : "<<score;
    }
};
class Result :public Test,public Sports
{
int Total;
public:
void display()
{
    Total = Mark1 + Mark2 + score;
Writeno();
Writemark();
Writescore();
cout<<"\n\n Total Marks Obtained    : "<< Total<<endl;
}
};
int main()
{
   Result stud1;
stud1.Readno(1201);
stud1.Readmark(93.5,95);
stud1.Readscore(80);
cout<<"\n\t\t\t HYBRID INHERITANCE PROGRAM\n";
stud1.display();
return 0;
}
```

Output

```
HYBRID INHERITANCE PROGRAM
Roll no : 1201
      Marks Obtained
 Mark1               : 93.5
 Mark2               : 95
 Sports Score        : 80
 Total Marks Obtained    : 268
```

COMPUTER SCIENCE – XI
List of Authors and Reviewers

Domain Experts

Dr. T.V.Gopal
Professor ,Dept. of Computer Science and Technology,
College of Engineering, Guindy, Anna University, Chennai.

Dr. Chitra Babu
Professor and Head of the Department,
Dept of Computer Science and Engineering, SSN College of Engineering , Chenna

Mrs. Bagyalakshmi P
Asst. Professor and Head of the Department, Dept of Computer
Applications, Queen Mary's College, Chennai

Mrs. Sasikala k
Associate Professor, Dept of Computer Science,
Queen Mary's College, Chennai

Dr. Radha P
Assistant Professor, Dept of Information Technology,
Govt. Arts & Science College (A), Coimbatore

Dr. Nester Jeyakumar M
Associate Professor and Head Of the Department,
Dept of Computer Science, Loyola College, Chennai

Dr. Srinivasan N
Professor, Dept of Computer Science and Engineering,
Sathyabama Institute of Science & Technology, Chennai

Dr. Chandra Mohan B
Associate Professor, School of Computer Science and Engineering,
Vellore Institute of Technology, Vellore

Mr. Sethuraman R
Assistant Professor, Dept of Computer Science and Engineering,
Sathyabama Institute of Science & Technology, Chennai

Mr. Sankar K
Assistant Professor, Dept of Computer Science,
RKM Vivekananda College, Mylapore, Chennai

Experts Co-ordinator

Mr. Ravikumar Arumugam
Deputy Director,
SCERT, Chennai

Art and Design Team

Layout

THY designers and computers
Chennai.

In-House

Gopu Rasuvel
Rajesh Thangapan
C. Prasanth
Adaikkala Stephen S

Cover Design

Kathir Arumugam

Co-ordination

Ramesh Munisamy

Reviewers

Dr. Ranjani Parthasarathi
Professor, Dept of Info Sci and Tech, College of Engineering, Guindy, Anna University, Chennai

Mr. Munivel E
Scientist/Engineer 'C' IT Group (Information Security), NIELIT
Calicut (MeitY, Govt. of India), NIT Campus, Calicut - KL (IN).

Authors

Mr. Kannan K
Post Graduate Teacher, Chennai Girls Hr Sec School,
Rotler street , Chennai

Mr. Ramakrishnan V G
Post Graduate Teacher, Karnataka Sangha Hr Sec School,
T Nagar, Chennai

Mrs. Bindhu Mohandas
Post Graduate Teacher, Vijayanta Model Hr Sec School,
H.V.F Estate , Avadi, Chennai

Mr. Gowrisankar N.V
Post Graduate Teacher, Chennai Girls Hr Sec School,
Nungambakkam, Chennai

Mr. Sreenivasan R
Post Graduate Teacher, Santhome Hr Sec School, Mylapore, Chennai

Mr. Lenin K
Post Graduate Teacher, Chennai Girls Hr Sec School, Saidapet, Chennai

Miss. Sangeetha A
Post Graduate Teacher, Govt. Hr Sec School, Rajanthangal,
Thiruvannamalai Dt

Dr. Valarmathi K E
Post Graduate Teacher, Velammal Vidhyashram, Surapet, Chennai

Mrs. Gajalakshmi R
Post Graduate Teacher, Jaigopal Garodia Hindu Vidyalaya Hr Sec School,
West Mambalam, Chennai

Dr.Mrs. Vidhya H.
Post Graduate Teacher,
DAV Boys Senior Secodnary School, Gopalapuram, Chennai.

Academic Coordinators

Mrs. Nevedha Selvaraj
Assistant Professor, SCERT, Chennai

Mrs. Tamil Selvi R
B.T. Assistant,
Government High School, Poonampalayam, Trichy District

QR Code Management Team

R. Jaganathan, SGT,
PUMS - Ganesapuram, Polur,
Thiruvannamalai.

S. Albert Valavan Babu, B.T. Asst,
GHS, Perumal Kovil, Paramakudi,
Ramanathapuram

M. Murugesan, B.T. Asst.,
PUMS., Pethavelankottagam,
Muttupettai, Thiruvarur.

This book has been printed on 80 G.S.M.
Elegant Maplitho paper.

Printed by offset at: