केन्द्रीय विद्यालय संगठन

Kendriya Vidyalaya Sangathan

STUDY MATERIAL
(Computer Science)

CLASS-XI
2014-15

KENDRIYA VIDYALAYA SANGATHAN
GURGAON REGION
SECTOR-14, OLD DELHI GURGAON ROAD, GURGAON
(HARYANA)- 122001

STUDY MATERIAL
CLASS XI (COMPUTER SCIENCE)

CHIEF PATRON:
Sh. AVINASH DIKSHIT
(COMMISSIONER, KVS)

PATRON:
MR. C. MANI
(DEPUTY COMMISSIONER, GURGAON REGION)

GUIDE:
Dr. A. K. SHARMA, ASSISTANT COMMISSIONER, GURGAON REGION
Sh. B.L.MORODIA, ASSISTANT COMMISSIONER, GURGAON REGION
Sh. C.S AZAD, ASSISTANT COMMISSIONER, GURGAON REGION

CORDINATOR:
MRS. RAJNI H. UPPAL
PRINCIPAL KV, SEC. 8. ROHINI, NEW DELHI

SUBJECT CONTRIBUTORS:-
Mr. Lavendra Kumar Tyagi, PGT (Comp. Sc.) K. V. Sec. 8 Rohini, New Delhi
Mr. Omprakash, PGT (Comp. Sc.) K. V. Sec. 8 Rohini, New Delhi
Mr. Bhupesh Bhatt, PGT (Comp. Sc.) K. V. AFS Rajokri, New Delhi
Mr. Amit Saxena, PGT (Comp. Sc.) K. V. Sec. 3 Rohini , New Delhi
Mrs. Neelima , PGT (Comp. Sc.) K. V. Sec. 3, Rohini, New Delhi
Mrs. Bhawana Duggal, PGT (Comp. Sc.) K. V. Sec. 22, Rohini, New Delhi
Mr. Manoj Kulshrestha, PGT (Comp. Sc.) K. V. AFS Bawana, New Delhi

INDEX

Syllabus

Computer Science (083)

Class XI (Theory) C++

Duration: 3 hours | Total Marks: 70

Unit No.	Unit Name	Marks
1.	COMPUTER FUNDAMENTALS	10
2.	PROGRAMMING METHODOLOGY	12
3.	INTRODUCTION TO C++	14
4.	PROGRAMMING IN C++	34
		70

UNIT 1: COMPUTER FUNDAMENTALS (18 Theory + 6 Practical) Periods

Evolution of computers; Basics of computer and its operation; Functional Components and their interconnections, concept of Booting. Classification of Computers.

Software concepts: Types of Software - System Software, Utility Software and Application Software

System Software: Operating System, Complier, Interpreter and Assembler

Operating System: Need for Operating System, Functions of Operating System (Processor Management, Memory Management, File Management and Device Management), Types of Operating System-interactive (GUI based), Time Sharing, Real Time and Distributed, Commonly used operating system: UNIX, LINUX, Windows, Solaris, BOSS (Bharat Operating System Solutions); Mobile OS - Android, Symbian.

Utility Software: Anti Virus, File Management tools, Compression tools and Disk Management tools (Disk Cleanup, Disk Defragmenter, Backup).

Open Source Concepts: Open Source Software, Freeware, Shareware, Proprietary Software.

Application Software: Office Tools - Word Processor, Presentation Tool, Spreadsheet Package, Database Management System; Domain Specific tools - School Management System, Inventory Management System, Payroll System, Financial Accounting, Hotel Management, Reservation System and Weather Forecasting System.

Number System: Binary, Octal, Decimal, Hexadecimal and conversion between two different number systems.

Internal Storage encoding of Characters: ASCII, ISCII (Indian scripts Standard Code for Information Interchange), and UNICODE (for multilingual computing)

Microprocessor: Basic concepts, Clock speed (MHz, GHz), 16 bit, 32 bit, 64 bit processors; 128 bir processors; Types - CISC Processors (Complex Instruction set computing), RISC Processors (Reduced Instruction set Computing), and EPIC (Explicitly parallel Instruction computing).

Memory Concepts: Units: Byte, Kilo Byte, Mega Byte, Giga Byte, Tera Byte, Peta Byte, Exa Byte, Zetta Byte, Yotta Byte.

Primary Memory: Cache, RAM, ROM

Secondary Memory: Fixed and Removable storage - Hard Disk Drive, CD/DVD Drive, Pen Drive, Blue Ray Disk.

Input Output Ports/ Connections: Serial, Parallel and Universal Serial Bus, PS-2 port, Infrared port, Bluetooth, Firewire.

UNIT 2: PROGRAMMING METHODOLOGY (28 Theory + 10 Practical) Periods\

General Concepts: Modular Approach, Clarity and Simplicity of Expressions, Use of proper names for Identifiers, Comments, Indentation; Documentation and Program Maintenance; Running and Debugging programs, Syntax Errors, Run-Time Errors, Logical Errors

Problem solving Methodologies: Understanding of the problem, solution for the problem, identifying minimum number of inputs required for output, writing code to optimizing execution time and memory storage, step by step solution for the problem, breaking down solution into simple steps (modular approach), identification of arithmetic and logical operations required for solution; Control Structure-conditional control and looping (finite and infinite).

Problem Solving: Introduction to Algorithms/Flowcharts.

UNIT 3: INTRODUCTION TO C++ (44 Theory + 36 Practical) Periods

Getting Started: C++ character set, C++ Tokens (Identifiers, Keywords, Constants, Operators,), Structure of a C++ Program (include files, main function), Header files - iostream.h, iomanip.h, cout, cin; use of I/O operators (<<and>>), Use of endl and setw (), Cascading of I/O operators; Compilation, Error Messages and execution.

Data Types, Variables and Constants: Concept of Data types; Built-in Data types: char, int, float and double; Constants: Integer Constants, Character constants - \n, \t, \b), Floating Point Constants, String Constants; Access modifier; Variables of built-in-data types, Declaration/Initialization of variables, Assignment statement, Type modifier: signed, unsigned, long

Operator and Expressions: Operators: Arithmetic operators (-,+,*,/,%),Assignment operator(=),C++ shorthands (+=,- =,*=,/=,%=) Unary operator (-), Increment(++) and Decrement (--) Operators, Relation operator (>,>=,<=,=,!=), Logical operators (!,&&,II),Conditional operator; Precedence of Operators; Automatic type conversionin expressions, Type casting;

UNIT 4: **PROGRAMMING IN C++** **(50 Theory + 48 Practical) Periods**

Flow of control:

Conditional statements: if else, Nested if, switch..case..default, Nestedswitch..case, break statement (to be used in switch..case only); **Loops:** while, do - while, for and Nested loops

Inbuilt Functions

Header file Categorization	Header File	Function
Standard input/output functions	stdio.h	gets ( ), puts ( )
Character Functions	Ctype.h	isalnum ( ), isalpha ( ),isdigit ( ), islower ( ), isupper ( ), tolower ( ),toupper ( )
String Function	string.h	strcpy ( ), strcat ( ), strlen ( ), strcmp ( ), strcmpi ( ), strev ( ),strlen ( ), strupur ( ),strlwr ( )
Mathematical Functions	math.h	fabs ( ), pow ( ), sgrt ( ), sin ( ), cos ( ), abs ( )
Other Functions	stdlib.h	randomize ( ), random ( )

Introduction to user-defined function and its requirements.

Defining a function; function prototype, Invoking/calling a function, passing arguments to function, specifying argument data types, default argument, constant argument, call by value, call by reference, returning values from a function, scope rules; local and global variables.

Structured Data Type:

Arrays: Introduction to Array and its advantages.

One Dimensional Array : Declaration/initialization of One-dimensional array, inputting array elements, accessingarray elements, manipulation of array elements (sum of elements, product of elements, average of elements, linearsearch, finding maximum/minimum value)

Declaration / Initialization of a String, string manipulations (counting vowels/ consonants/ digits/ special characters, case conversion, reversing a string, reversing each word of a string

Two-dimensional Array:

Declaration/initialization of a two-dimensional array, inputting array elements accessing array elements, manipulationof array elements (sum of row element, column elements, diagonal elements, finding maximum / minimum values)

User-defined Data Types: Introduction to user defined data types.

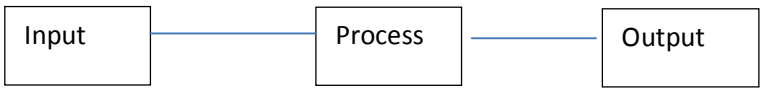
Structure

Defining a Structure, declaring structure variables, accessing structure elements, passing structureto functions as value and reference, function returning structure, array of structure

Defining a symbol name using **typedef** keyword and defining a macro using **#define** preprocessor directive.

Unit-1
Computer Fundamentals

What is Computer?
Computer is an advanced electronic device that takes raw data as input from the user and processes these data under the control of set of instructions (called program) and gives the result (output) and saves output for the future use. It can process both numerical and non-numerical (arithmetic and logical) calculations.
It works the principle of I-P-O Cycle



A computer has four functions:

a. accepts data	**Input**
b. processes data	**Processing**
c. produces output	**Output**
d. stores results	**Storage**

(While designing the Difference Engine and Analytical Engine Charles Babage has given the concept of these four units, Hence he is known as “Father of Computer”.)
Input (Data):
Input is the raw information or facts entered into a computer from the input devices. It is the collection of letters, numbers, images etc.
Process:
Process is the operation of data as per given instruction. It is totally internal process of the computer system.
Output:
Output is the processed data given by computer after data processing. Output is also called Result or information . We can save these results in the storage devices for the future use.

Computer System
The components of the Computer System are:-

1. Hardware
2. Software
3. Firmware
4. Liveware

COMPUTER SYSTEM = HARDWARE + SOFTWARE+ USER
• Hardware = Internal Devices + Peripheral Devices
All physical parts of the computer (or everything that we can touch) are known as Hardware.
• Software = Programs
Software gives "intelligence" to the computer.
• USER = Person, who operates computer.

Hardware
All the physical and tangible components of Computer are called Hardware. In other words all the components that we can touch come under the category of Hardware eg Keyboard, Mouse,

Software
Software is a set of instructions or a program that enables a hardware to run. Without the use of software a hardware cannot work.eg. Windows 8, Photoshop, MS Office etc.

Firmware
Instructions written/embedded on a hardware are known as firmware e.g., BIOS instruction on ROM chip are called Firmware.

Liveware
Persons or the users, using Computers in day to day activity are known as liveware.

Generations of computer:

First Generation (1940-56):
The first generation computers used **Vacuum tubes & Machine language** was used for giving the instructions. These computer were large in size & their programming was difficult task. The electricity consumption was very high. Some computers of this generation are ENIAC, EDVAC, EDSAC& UNIVAC-1.

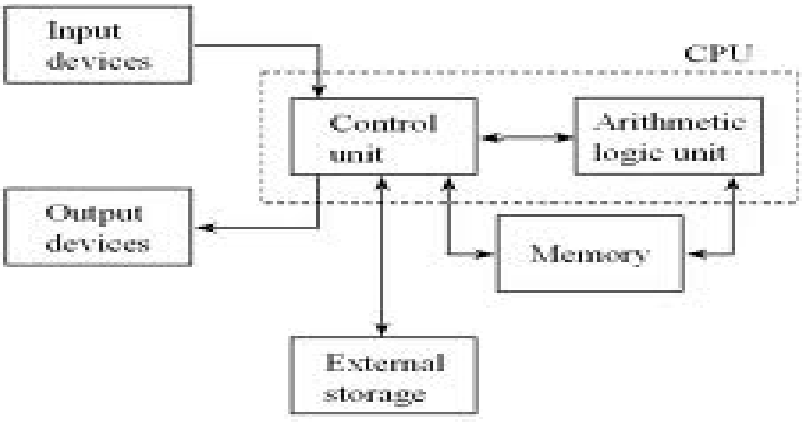
Second Generation(1956-63):
In 2nd generation computers, **Vacuum tubes were replaced by Transistors**. They required only 1/10 of power required by **Vacuum** tubes. This generation computers generated less heat & were reliable. The first operating system developed in this generation.

The Third Generation(1964-71):
The 3rd generation computers replaced transistors with Integrated circuit known as chip. From Small scale integrated circuits which had 10 transistors per chip, technology developed to MSI circuits with 100 transistors per chip. These computers were smaller, faster & more reliable. High level languages invented in this generation.

The fourth Generation(1972- present):
LSI & VLSI were used in this generation. As a result microprocessors came into existence. The computers using this technology known to be Micro Computers. High capacity hard disk were invented. There is great development in data communication.

The Fifth Generation (Present & Beyond):
Fifth generation computing devices, based on artificial intelligence, are still in development, though there are some applications, such as voice recognition, that are being used today. The use of parallel processing and superconductors is helping to make artificial intelligence a reality. Quantum computation and molecular and nanotechnology will radically change the face of computers in years to come.

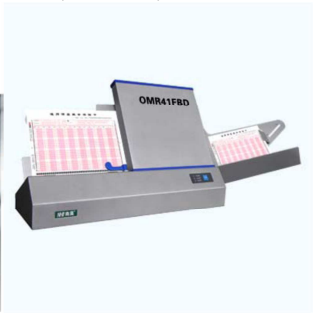
ARCHITECTURE OF COMPUTER



Input Devices: Those devices which help to enter data into computer system. Eg. Keyboard, Mouse, Touch screen,Bar Code Reader, Scanner, MICR, OMR etc.

Bar code Reader

MICR used in Bank



OMR

Output Devices: Those devices which help to display the processed information.Eg. Monitor, Printer, Plotter, Projector

Printer

Plotter

Projector

CENTRAL PROCESSING UNIT (CPU)

The main component to make a computer operate is the computer chip or microprocessor. This is referred to as the Central Processing Unit (CPU). It is also known as Brain of computer. It performs arithmetic and logic operations. The CPU (Central Processing Unit) is the device that interprets and executes instructions.

Processor

Memory: It facilitates the remembrance power to computer system. It refers to the physical devices used to store programs (sequences of instructions) or data (e.g. program state information) on a temporary or permanent basis for use in a computer or other digital electronic device.
Memory can be of two types:-

1. Primary Memory
2. Secondary Memory

The term primary memory is used for the information in physical systems which are fast (i.e. RAM), as a distinction from secondary memory, which are physical devices for program and data storage which are slow to access but offer higher memory capacity. Primary memory stored on secondary memory is called virtual memory.
Primary Memory can be categorized as:-

1. Volatile Memory (RAM)
2. Non-Volatile Memory(ROM)

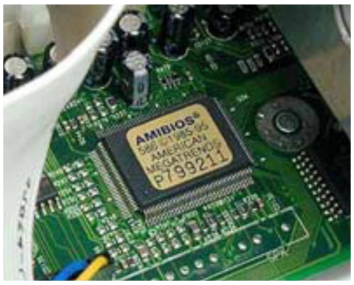
Volatile memory (RAM)
Volatile memory is computer memory that requires power to maintain the stored information. RAM stands for Random Access Memory. The data is primarily stored on RAM. This is also known as Read-Write memory as both the operation can take place on it. It is volatile in nature because as soon as the power is off its contents are also removed. It can be of two types:-

1. Static RAM or SRAM.
2. Dynamic RAM or DRAM

SRAM retains its contents as long as the power is connected and is easy to interface to but uses six transistors per bit.

Dynamic RAM is more complicated to interface to and control and needs regular refresh cycles to prevent its contents being lost. However, DRAM uses only one transistor and a capacitor per bit, allowing it to reach much higher densities and, with more bits on a memory chip, be much cheaper per bit. SRAM is not worthwhile for desktop system memory, where DRAM dominates, but is used for their cache memories..

Non Volatile Memory (ROM)
Non-volatile memory is computer memory that can retain the stored information even when not powered. ROM stands for Read Only Memory. As the name suggests we can perform only read operation on ROM. It is permanent in nature. In ROM booting instructions for computer in the form of firmware are stored



Other examples of non-volatile memory are flash memory and PROM/EPROM/EEPROM memory.

Cache Memory:

Cache memory is an intermediate between RAM and processor. It is very fast. Cache memory is random access memory (RAM) that a computer microprocessor can access more quickly than it can access regular RAM. As the microprocessor processes data, it looks first in the cache memory and if it finds the data there (from a previous reading of data), it does not have to do the more time-consuming reading of data from larger memory.

Secondary Memory:

A. Hard Disk (Local Disk)
B. Optical Disks: CD-R, CD-RW, DVD-R, DVD-RW
C. Floppy Disks
D Memory Cards
E. External Hard Disk
F. Blu Ray Disk

Blu-Ray Disk:

Blu-ray (not Blue-ray) also known as Blu-ray Disc (BD), is the name of a new optical disc format. The format offers more than five times the storage capacity of traditional DVDs and can hold up to 25GB on a single-layer disc and 50GB on a dual-layer disc. While current optical disc technologies such as DVD, DVD±R, DVD±RW, and DVD-RAM rely on a red laser to read and write data, the new format uses a blue-violet laser instead, hence the name Blu-ray.

Units of Memory:

The smallest unit is bit, which mean either 0 or 1.

1 bit	= 0 or 1
1 Byte	= 8 bit
1 Nibble	= 4 bit
1 Kilo Byte	= 1024 Byte= 2^{10} Byte
1 Mega Byte	= 1024 KB= 2^{10} KB
1 Gega Byte	= 1024 MB= 2^{10} MB
1 Tera Byte	= 1024 GB= 2^{10} GB
1 Peta Byte	=1024 TB= 2^{10} TB
1 Exa Byte	=1024 PB= 2^{10} PB
1 Zetta Byte	= 1024 EB= 2^{10} EB
1 Yotta Byte	= 1024 ZB= 2^{10} ZB

Types of Computer

On the basis of working principle

a) Analog Computer

An analog computer is a form of computer that uses continuous physical phenomena such as electrical, mechanical, or hydraulic quantities to model the problem being solved.

Eg: Thermometer, Speedometer, Petrol pump indicator, Multimeter

b) Digital Computer

A computer that performs calculations and logical operations with quantities represented as digits, usually in the binary number system.

c) Hybrid Computer (Analog + Digital)

A combination of computers those are capable of inputting and outputting in both digital and analog signals. A hybrid computer system setup offers a cost effective method of performing complex simulations. The instruments used in medical science lies in this category.

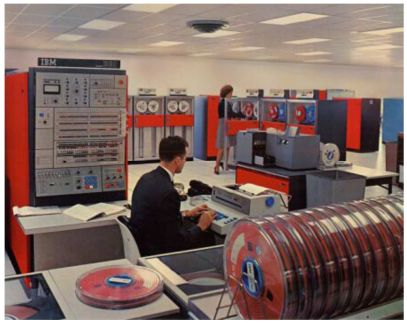
On the basis of Size

a) Super Computer

The fastest type of computer. Supercomputers are very expensive and are employed for specialized applications that require immense amounts of mathematical calculations. For example, weather forecasting requires a supercomputer. Other uses of supercomputers include animated graphics, fluid dynamic calculations, nuclear energy research, and petroleum exploration. PARAM, Pace & Flosolver are the supercomputer made in India.

b) Mainframe Computer

A very large and expensive computer capable of supporting hundreds, or even thousands, of users simultaneously. In the hierarchy that starts with a simple microprocessor (in watches, for example) at the bottom and moves to supercomputers at the top, mainframes are just below supercomputers. In some ways, mainframes are more powerful than supercomputers because they support more simultaneous programs. But supercomputers can execute a single program faster than a mainframe.



c) Mini Computer

A midsized computer. In size and power, minicomputers lie between *workstations* and *mainframes*. In the past decade, the distinction between large minicomputers and small mainframes has blurred, however, as has the distinction between small minicomputers and workstations. But in general, a minicomputer is a multiprocessing system capable of supporting from 4 to about 200 users simultaneously. Generally, servers are comes in this category.

d) Micro Computer

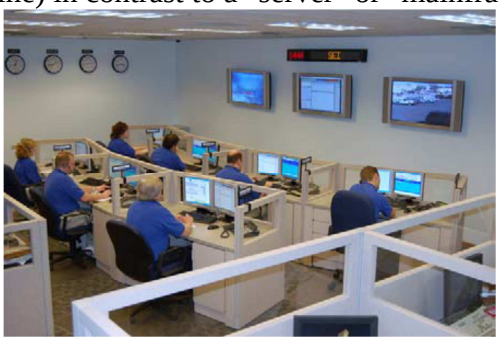
i. **Desktop Computer:** a personal or micro-mini computer sufficient to fit on a desk.

ii. **Laptop Computer:** a portable computer complete with an integrated screen and keyboard. It is generally smaller in size than a desktop computer and larger than a notebook computer.

iii. **Palmtop Computer/Digital Diary /Notebook /PDAs:** a hand-sized computer. Palmtops have no keyboard but the screen serves both as an input and output device.

e) Workstations

A terminal or desktop computer in a network. In this context, workstation is just a generic term for a user's machine (client machine) in contrast to a "server" or "mainframe."



Software

As specified earlier Software, simply are the computer programs. The instructions given to the computer in the form of a program is called Software. Software is the set of programs, which are used for different purposes. All the programs used in computer to perform specific task is called Software.

Types of software

1. **System software:**
   - **a) Operating System Software**
     DOS, Windows XP, Windows Vista, Unix/Linux, MAC/OS X etc.
   - **b) Utility Software**
     Windows Explorer (File/Folder Management), Compression Tool, Anti-Virus Utilities, Disk Defragmentation, Disk Clean, BackUp, WinZip, WinRAR etc…
   - **c) Language Processors**
     Compiler, Interpreter and Assembler
2. **Application software:**
   - **a) General Application Software**
     Ms. Office 2003, Ms. Office 2007, Macromedia (Dreamweaver, Flash, Freehand), Adobe (PageMaker, PhotoShop)
   - **b) Tailored or Customized Software**
     School Management system, Inventory Management System, Payroll system, financial system etc.

Operating system

Operating system is an interface between hardware and user which is responsible for the management and coordination of activities and the sharing of the resources of a computer. It hosts the several applications that run on a computer and handles the operations of computer hardware.

Functions of operating System:

- Processor Management
- Memory Management
- File Management
- Device Management

Types of Operating System:

- **Real-time Operating System:** It is a multitasking operating system that aims at executing real-time applications. Example of Use: e.g. control of nuclear power plants, oil refining, chemical processing and traffic control systems, air

- **Single User Systems:** Provides a platform for only one user at a time. They are popularly associated with Desk Top operating system which run on standalone systems where no user accounts are required. Example: DOS.
- **Multi User Systems:** Provides regulated access for a number of users by maintaining a database of known users. Refers to computer systems that support two or more simultaneous users. Another term for multi-user is time sharing. Ex: All mainframes are multi-user systems. Example: Unix
- **Multi-tasking and Single-tasking Operating Systems:** When a single program is allowed to run at a time, the system is grouped under the single-tasking system category, while in case the operating system allows for execution of multiple tasks at a time, it is classified as a multi-tasking operating system.
- **Distributed Operating System:** An operating system that manages a group of independent computers and makes them appear to be a single computer is known as a distributed operating system. Distributed computations are carried out on more than one machine. When computers in a group work in cooperation, they make a distributed system.

Commonly used operating system

UNIX: Pronounced u*oo-niks*, a popular multi-user, multitasking operating system developed at Bell Labs in the early 1970s. UNIX was one of the first operating systems to be written in a high-level programming language, namely C. This meant that it could be installed on virtually any computer for which a C compiler existed.

LINUX: Pronounced *lee-nucks* or *lih-nucks*. A freely-distributable open source operating system that runs on a number of hardware platforms. The Linux kernel was developed mainly by Linus Torvalds and it is based on Unix. Because it's free, and because it runs on many platforms, including PCs and Macintoshes, Linux has become an extremely popular alternative to proprietary operating systems.

Windows: Microsoft Windows is a series of graphical interface operating systems developed, marketed, and sold by Microsoft. Microsoft introduced an operating environment named Windows on November 20, 1985 as an add-on to MS-DOS in response to the growing interest in graphical user interfaces (GUIs). Microsoft Windows came to dominate the world's personal computer market with over 90% market share, overtaking Mac OS, which had been introduced in 1984.The most recent client version of Windows is Windows 7; the most recent server version is Windows Server 2008 R2; the most recent mobile version is Windows Phone 7.5.

SOLARIS: Solaris is a Unix operating system originally developed by Sun Microsystems. It superseded their earlier SunOS in 1993. **Oracle Solaris**, as it is now known, has been owned by Oracle Corporation since Oracle's acquisition of Sun in January 2010.

BOSS: BOSS (Bharat Operating System Solutions) GNU/Linux distribution developed by C-DAC (Centre for Development of Advanced Computing) derived from Debian for enhancing the use of Free/ Open Source Software throughout India. This release aims more at the security part and comes with an easy to use application to harden your Desktop.

Mobile OS: A mobile operating system, also called a mobile OS, is an operating system that is specifically designed to run on mobile devices such as mobile phones, smartphones, PDAs, tablet computers and other handheld devices. The mobile operating system is the software platform on top of which other programs, called application programs, can run on mobile devices.

- **Android:** Android is a Linux-based mobile phone operating system developed by Google. Android is unique because Google is actively developing the platform but giving it away for free to hardware manufacturers and phone carriers who want to use Android on their devices.

- **Symbian:** Symbian is a mobile operating system (OS) targeted at mobile phones that offers a high-level of integration with communication and personal information management (PIM) functionality. Symbian OS combines middleware with wireless communications through an integrated mailbox and the integration of Java and PIM functionality (agenda and contacts). The Symbian OS is open for third-party development by independent software vendors, enterprise IT departments, network operators and Symbian OS licensees.

LANGUAGE PROCESSORS: Since a computer hardware is capable of understanding only machine level instructions, So it is necessary to convert the HLL into Machine Level Language. There are three Language processors:

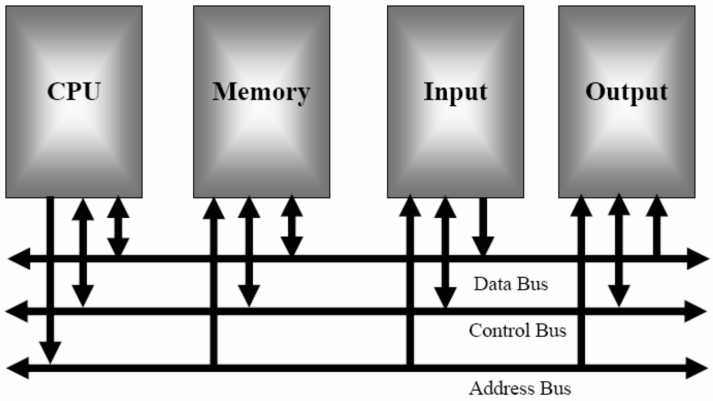
A. **Compiler:** It is translator which converts the HLL language into machine language in one go. A Source program in High Level Language get converted into Object Program in Machine Level Language.
B. **Interpreter:** It is a translator which converts and executes the HLL language code line by line. It takes one statement of HLL and converts it into machine code which is immediately executed. It eliminate the need of separate compilation/run. However, It is slow in processing as compare to compiler.
C. **Assembler:** It translate the assembly language into machine code.

Microprocessor:
A microprocessor is a semiconductor chip, which is manufactured using the Large Scale integration (LSI) or Very Large Scale Integration (VLSI), which comprises Arithmetic Logic Unit, Control unit and Central Processing Unit (CPU) fabricated on a single chip.

Terminologies:
Registers: A register is a very small amount of very fast memory that is built into the CPU (central processing unit) in order to speed up its operations by providing quick access to commonly used values. All data must be represented in a register before it can be processed. For example, if two numbers are to be multiplied, both numbers must be in registers, and the result is also placed in a register.

Bus:
A collection of wires through which data is transmitted from one part of a computer to another. You can think of a bus as a highway on which data travels within a computer. When used in reference to personal computers, the term bus usually refers to internal bus. This is a bus that connects all the internal computer components to the CPU and main memory. All buses consist of two parts -- an address bus and a data bus. The data bus transfers actual data whereas the address bus transfers information about where the data should go. The control bus is used by the CPU to direct and monitor the actions of the other functional areas of the computer. It is used to transmit a variety of individual signals (read, write, interrupt, acknowledge, and so forth) necessary to control and coordinate the operations of the computer.

The size of a bus, known as its width, is important because it determines how much data can be transmitted at one time. For example, a 16-bit bus can transmit 16 bits of data, whereas a 32-bit bus can transmit 32 bits



Clock speed: Also called *clock rate*, the speed at which a microprocessor executes instructions. Every computer contains an internal clock that regulates the rate at which instructions are executed and synchronizes all the various computer components. The CPU requires a fixed number of clock ticks (or *clock cycles*) to execute each instruction. The faster the clock, the more instructions the CPU can execute per second.
Clock speeds are expressed in megahertz (MHz) or gigahertz ((GHz).
16 bit Microprocessor: It indicates the width of the registers. A 16-bit microprocessor can process data and memory addresses that are represented by 16 bits. Eg. 8086 processor
32 bit Microprocessor: It indicates the width of the registers. A 32-bit microprocessor can process data and memory addresses that are represented by 32 bits. Eg. Intel 80386 processor, Intel 80486
64 bit Microprocessor: It indicates the width of the registers; a special high-speed storage area within the CPU. A 32-bit microprocessor can process data and memory addresses that are represented by 32 bits. e.g. Pentium dual core, Core 2 duo.
128 bit Microprocessor: It indicates the width of the registers. A 128-bit microprocessor can process data and memory addresses that are represented by 128 bits. e.g. Intel core i7

Difference between RISC & CISC architecture

RISC *(Reduced Instruction Set Computing):*

1. RISC system has reduced number of instructions.
2. Performs only basic functions.
3. All HLL support is done in software.
4. All operations are register to register.

CISC (Complex *Instruction Set Computing):*

1. A large and varied instruction set.
2. Performs basic as well as complex functions.
3. All HLL support is done in Hardware.
4. Memory to memory addressing mode

EPIC (Explicitly Parallel Instruction Computing):

It is a 64-bit microprocessor instruction set, jointly defined and designed by Hewlett Packard and Intel, that provides up to 128 general and floating point unit registers and uses speculative loading, predication, and explicit parallelism to accomplish its computing tasks. By comparison, current 32-bit CISC and RISC microprocessor architectures depend on 32-bit registers, branch prediction, memory latency, and implicit parallelism, which are considered a less efficient approach in micro architecture design.

PORTS: A port is an interface between the motherboard and an external device. Different types of port are available on motherboard as serial port, parallel port, PS/2 port, USB port, SCSI port etc.

Serial port(COM Port): A serial port transmit data one bit at a time. Typically on older PCs, a modem, mouse, or keyboard would be connected via serial ports. Serial cables are cheaper to make than parallel cables and easier to shield from interference. It is also called communication port.
Parallel Port (LPT ports): It supports parallel communication i.e. it can send several bits simultaneously. It provides much higher data transfer speed in comparison with serial port. It is also called Line Printer Port.

USB (Universal Serial Bus): It is a newer type of serial connection that is much faster than the old serial ports. USB is also much smarter and more versatile since it allows the "daisy chaining" of up to 127 USB peripherals connected to one port. It provides plug & play communication.

PS/2 Port : PS/2 ports are special ports for connecting the keyboard and mouse to some PC systems. This type of port was invented by IBM

FireWire Port : The IEEE 1394 interface, developed in late 1980s and early 1990s by Apple as FireWire, is a serial bus interface standard for high-speed communications and isochronous real-time data transfer. The 1394 interface is comparable with USB and often those two technologies are considered together, though USB has more market share.

Infrared Port: An IR port is a port which sends and receives infrared signals from other devices. It is a wireless type port with a limited range of 5-10ft.
Bluetooth: Bluetooth uses short-range radio frequencies to transmit information from fixed and mobile devices. These devices must be within the range of 32 feet, or 10 meters for Bluetooth to effectively work. A Bluetooth port enables connections for Bluetooth-enabled devices for synchronizing. Typically there are two types of ports: incoming and outgoing. The incoming port enables the device to receive connections from Bluetooth devices while the outgoing port makes connections to Bluetooth devices.

Internal Storage encoding of Characters:

ASCII (American standards code for information interchange): ASCII code is most widely used alphanumeric code used in computers. It is a 8- bit code, and so it has 2^8 =256 possible code groups. It represents all of the standard keyboard characters as well as control functions such as Return & Linefeed functions.

ISCII (Indian standards code for information interchange) : To use the Indian language on computers, ISCII codes are used. It is an 8-bit code capable of coding 256 characters. ISCII code retains all ASCII characters and offers coding for Indian scripts also.

Unicode: It is a universal coding standard which provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language. Unicode is a 16-bit code capable of representing more than 65000 characters. The coding for ASCII characters remain the same in Unicode. It can represent almost all the languages of the world.

UNIT-2

PROGRAMMING METHODOLOGY

Stylistic Guidelines:

Writing good program is a skill. This can be developed by using the following guidelines .

1. **Meaningful Names for identifiers:** A programmer should give the meaningful names to each section of the program so that it can help him to identify the variable used for specific purpose. This helps him to execute the right elements during the complex run of a program.
2. **Ensure clarity of expression**: Expression carry out the specified action. Thus they must be clearly understood by the users. There should not be any compromise with the clarity of expression.
3. **Use of comments and indentations**: Comments generally are used as the internal documentation of a program .if comments are used in the program they will guide the program while debugging and checking. While indentation is the proper way of writing to avoid the confusion regarding the flow of program. These highlights nesting of groups of control statements.
4. **Insert blank lines and blank spaces**: Blank lines should be used to separate long, logically related blocks of code. Specifically Normally in programming the standard for the use of spaces is to follow normal English rules. This means that: Most basic symbols in C++ (e.g., "=", "+", etc.) should have at least one space before and one space after them.
5. **Statements**: Each statement should appear on a separate line. The opening brace following a control statement such as if or while should appear on the line after the if or while, lined up with the left of the control statement, and the closing brace should appear on its own line, lined up with the left of the control statement. The opening and closing braces for a function should be lined up in the same way. The statements within a {_____} pair are indented relative to the braces.

Characteristics of a Good Program:

Following are the characteristics of a good program.

1. **Effective and efficient:** The program produces correct results and is faster, taking into account the memory constraints.
2. **User friendly:** The program should be user friendly. The user should not be confused during the program execution . The user should get correct direction and alerts when he is going through the program.
3. **Self documenting code:** A good program must have self documenting code. This code will help the programmer to identify the part of the source code and clarify their meaning in the program.
4. **Reliable:** The good program should be able to cope up from any unexpected situations like wrong data or no data.
5. **Portable:** The program should be able to run on any platform, this property eases the use of program in different situations.
6. **Robustness**: Robustness is the ability of the program to bounce back an error and to continue operating within its environment

PROBLEM SOLVING METHODOLOGY AND TECHNIQUES:

To develop an efficient and effective programs we should adopt a proper problem solving methodology and use appropriate techniques. Following are some of the methods and techniques to develop a good program.

1. **Understand the problem well**: for a good program one should understand the problem well . one should know what exactly is expected from the problem. Knowing the problem well is the half way done.
2. **Analyze the program. :** Analyzing the problem involves identifying the program specification and defining each program's minimum number of inputs required for output and processing components.
3. **Design** : In this phase of design a Model is developed which look alike a program . This phase gives the face to the program. Outputs are designed in this phase.

4. **Code program :**This step is the actual implementation of the program. In this program algorithm is translated into programming language. in this it is essential to decide which technique or logical will be more appropriate for coding.
5. **Test and Debug program.:** Once the solution algorithm is coded the next step is to test and debug the program. Testing is the process of finding errors in a program and debugging is of correcting the errors. The developed program is put to test in different conditions and verified at different level for its proper and efficient working.
6. **Implementation & Documentation:** After successful execution of the program it is implemented. Documentation refers to written descriptions specification, design code and comments, internal and external to program which makes more readable and understandable.

Uses of documentation:

1. This becomes an useful interface between a technical personnel and non technical personnel.
2. This is very useful for upkeep and maintenance.
3. Documentation makes ease for any technical emergencies.
4. Very useful in operating for learners and trainers.

Algorithm :-Algorithm is a step-by-step process of solving a well-defined computational problem. An algorithm is an effective method expressed as a finite list of well-defined instructions for calculating a function, starting from an initial state and initial input. Thus, a step-by-step procedure to solve the problem is called algorithm.

Example: Let us take one simple day-to-day example by writing algorithm for making „Maggi Noodles" as a food.

Step 1: Start
Step 2: Take pan with water
Step 3: Put pan on the burner
Step 4: Switch on the gas/burner
Step 5: Put maggi and masala
Step 6: Give two minutes to boil
Step 7: Take off the pan
Step 8: Take out the magi with the help of fork/spoon
Step 9: Put the maggi on the plate and serve it
Step 10: Stop.

Further, the way of execution of the program shall be categorized into three ways: (i) sequence statements; (ii) selection statements; and (iii) iteration or looping statements. This is also called as „control structure".

Sequence statements: In this program**,** all the instructions are executed one after another.
Example : Write an algorithm to print „Good Morning‟.
Step 1: Start
Step 2: Print „Good Morning‟
Step 3: Stop
Example: Write an algorithm to find area of a rectangle.
Step 1: Start
Step 2: Take length and breadth and store them as L and B?
Step 3: Multiply by L and B and store it in area
Step 4: Print area
Step 5: Stop

In the above mentioned two examples (Example II and III), all the instructions are executed one after another. These examples are executed under sequential statement. **Selective Statements:** In this program, some portion of the program is executed based upon the conditional test. If the conditional test is true, compiler will execute some part of the program, otherwise it will execute the other part of the program.

Example : Write an algorithm to check whether a person is eligible to vote? (Age more than or equal to 18 years makes one eligible to vote).
Step 1: Start
Step 2: Take age and store it in age
Step 3: Check age value, if age >= 18 then go to step 4 else step 5
Step 4: Print "Eligible to vote" and go to step 6
Step 5: Print "Not eligible to vote"
Step 6: Stop

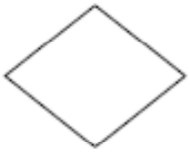
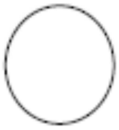
Example : Write an algorithm to check whether given number is +ve, -ve or zero.
Step 1: Start
Step 2: Take any number and store it in n.
Step 3: Check n value, if n > 0 then go to step 5 else go to step 4
Step 4: Check n value, if n < 0 then go to step 6 else go to step 7
Step 5: Print "Given number is +ve" and go to step 8
Step 6: Print "Given number is -ve" and go to step 8
Step 7: Print "Given number is zero"
Step 8: Stop
In the above mentioned examples IV and V, all the statements are not executed, but based upon the input, some portions of the algorithm are executed, because we have „true‟ or „false‟ situation in the program.

Iterative statements: In some programs, certain set of statements are executed again and again based upon conditional test. i.e. executed more than one time. This type of execution is called „looping or iteration‟.
Example: Write an algorithm to print all natural numbers up to „n‟.
Step 1: Start
Step 2: Take any number and store it in n.
Step 3: Store 1 in I
Step 4: Check I value, if I<=n then go to step 5 else go to step 8
Step 5: Print I
Step 6: Increment I value by 1
Step 5: Go to step 4
Step 8: Stop

In the above example, steps 4, 5, 6 and 7 are executed more than one time.

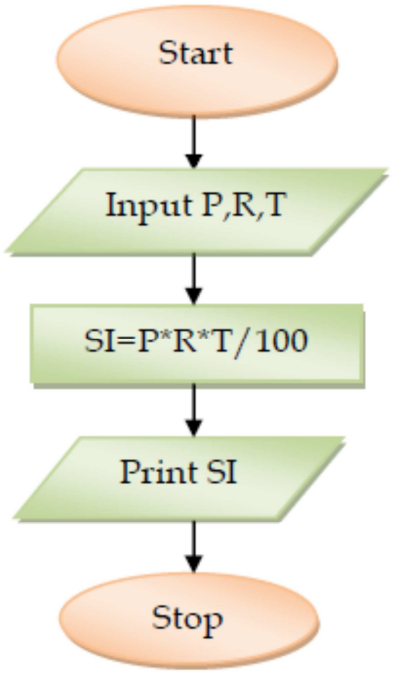
Flowchart :-We can also show steps of an algorithm in graphical form by using some symbols. This is called flowcharting. Thus, Flowchart is a pictorial view of algorithm.

Flowchart Symbols:-Some of the standard symbols along with respective function(s) that are used for making flowchart are as follows:

Symbols	Functions
1. 	Start/stop
2. 	Input/output
3. 	Processing
4. 	Decision Box
5.	Flow of control
6. 	Connector

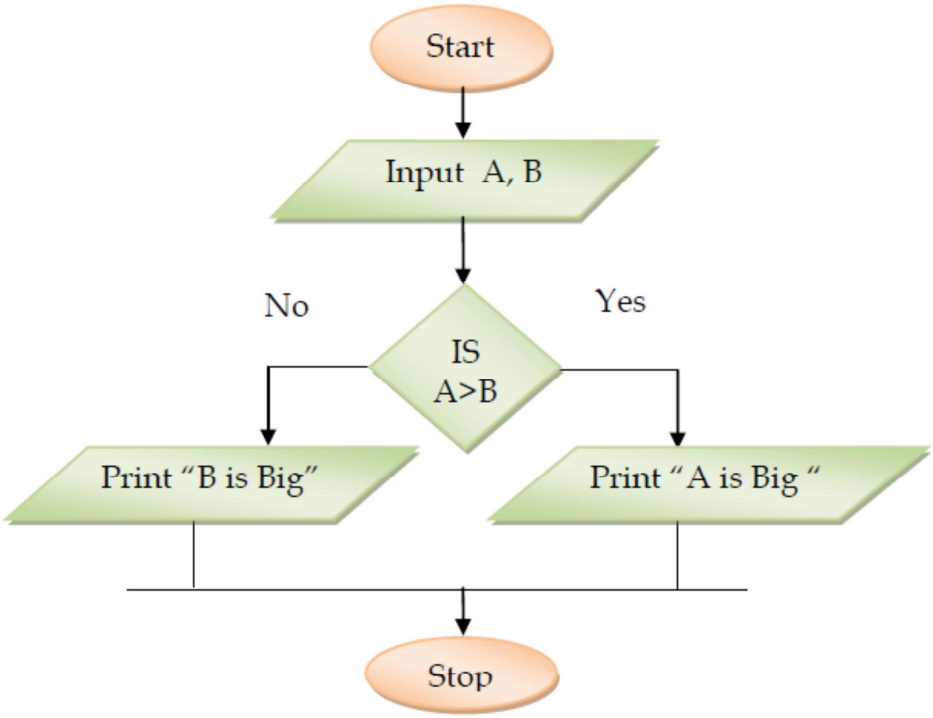
The following flowchart is an example of a sequential execution.

Example : Draw a flowchart to find the simple interest. (Sequence)
Solution:

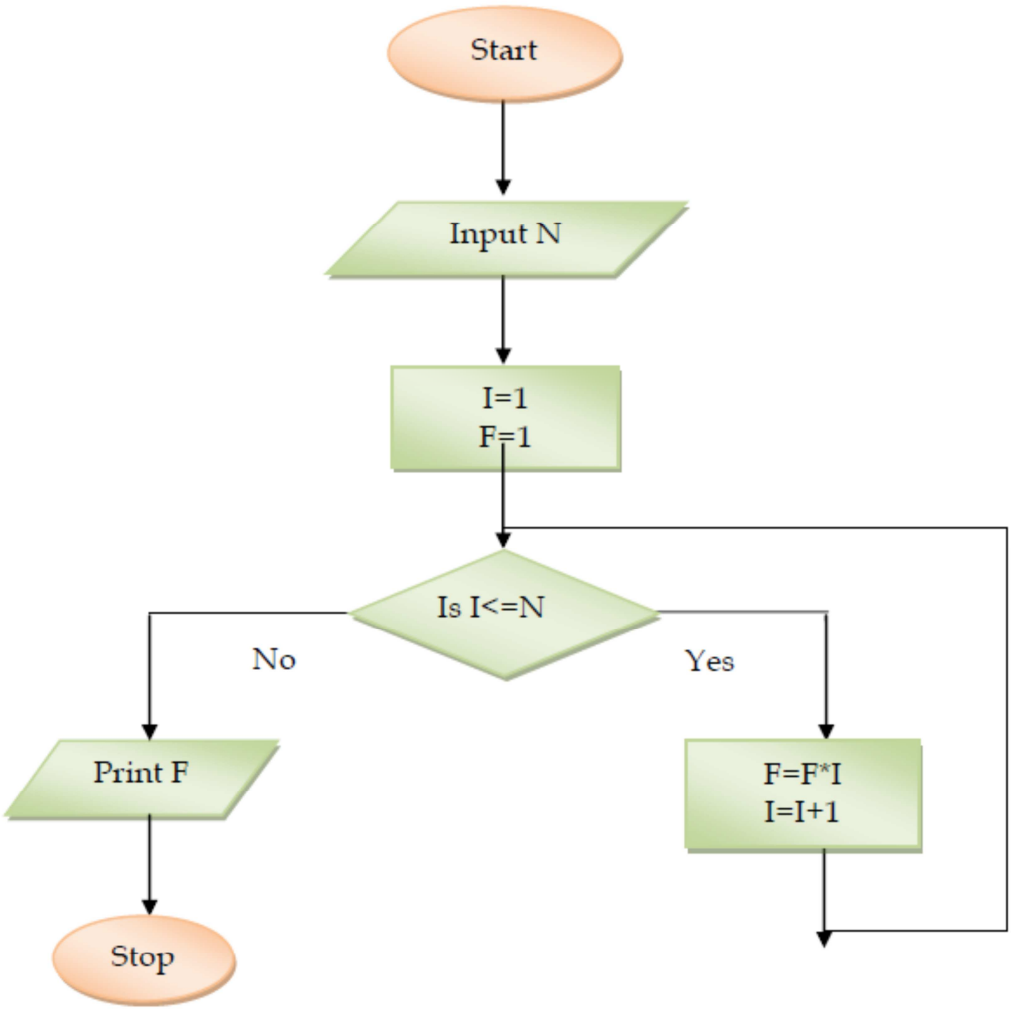


The following flowchart is an example of a selective execution.

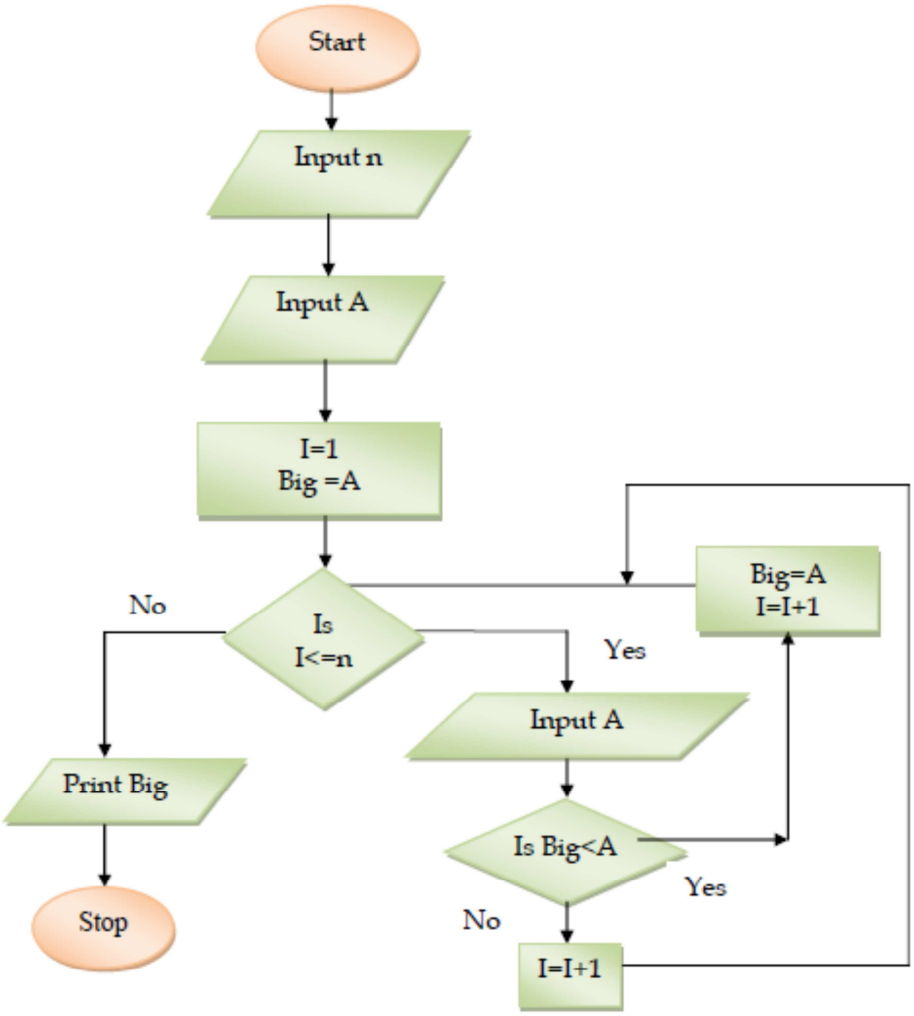
Example : Draw a flowchart to find bigger number among two numbers (selective)
Solution:



Following are the examples of an iterative execution.
Example : Draw a flow chart to find factorial of any number.
Solution:



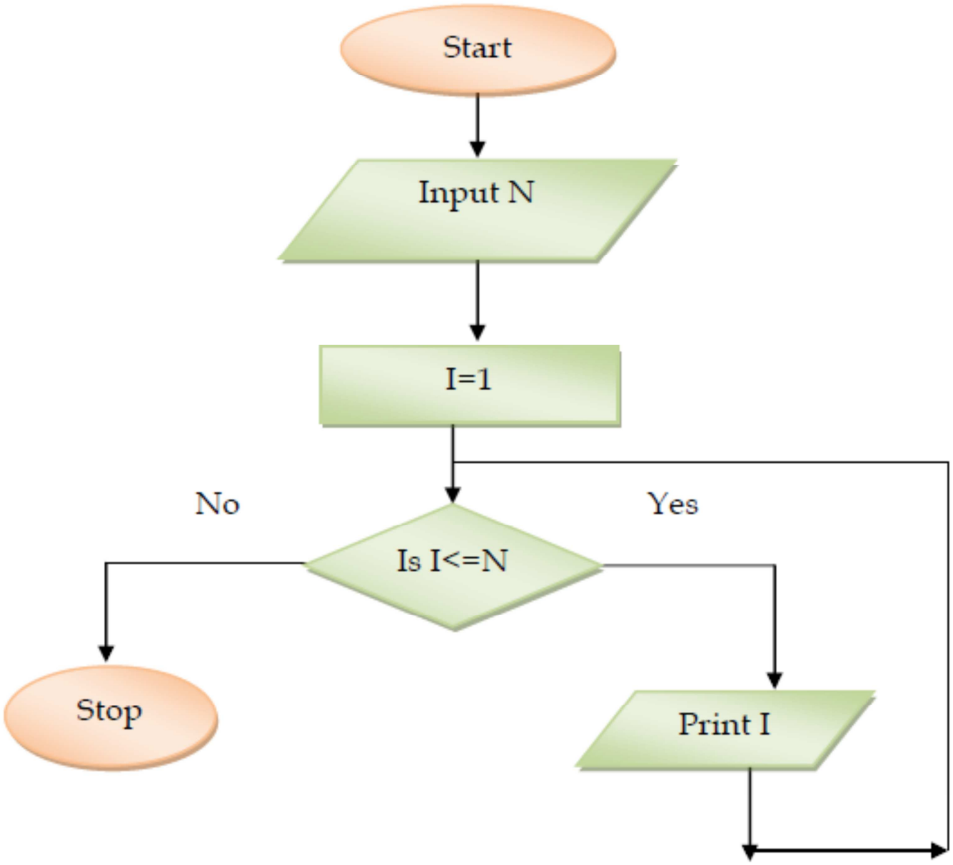
Example: Draw a flow chart to find biggest number among "n" numbers.
Solution:



Finite and Infinite loop: In looping statements, if some set of statements are executed n times (fixed number of times), then it is called finite loop. At the same time, if some set of statements are executed again and again without any end (infinite times), then it is called infinite loop . For example (X), if we are not incrementing I (index) value, then we will get endless (infinite) loop. Set of statements is executed again and again without any end is called infinite loop. The following is an example of infinite

Example : Draw a flow chart to describe an infinite loop.

Solution:



In the above example value of I is not incremented, so it will create endless loop.
This is also called infinite loop.

Type of errors: There are three types of errors generally occur during compilation and running a program. They are (i) Syntax error; (ii) Logical error; and (iii) Runtime error.

Syntax error: Every programming language has its own rules and regulations (syntax). If we overcome the particular language rules and regulations, the syntax error will appear (i.e. an error of language resulting from code that does not conform to the syntax of the programming language). It can be recognized during compilation time.

Example

```
int a = 0;
while (a < 10)
{
a = a + 1
cout<<a;
}
```

In the above statement, the fourth line is not correct. Since the statement does not end with **;**. This will flash a syntax error.

Logical error: Programmer makes errors while writing program that is called logical error. It is an error in a program's source code that results in incorrect or unexpected result. It is a type of runtime error that may simply produce the wrong output or may cause a program to crash while running. The logical error might only be noticed during runtime, because it is often hidden in the source code and are typically harder to find and debug.

```
int a = 100;
while (a < 10)
{
a = a + 1;
cout<< a;
}
```

In the above example, the while loop will not execute even a single time, because the initial value of a is 100.

Runtime error: A runtime error is an error that causes abnormal termination of program during run time. In general, the dividend is not a constant but might be a number typed by you at runtime. In this case, **division by zero** is illogical. Computers check for a "division by zero" error during program execution, so that you can get a "division by zero" error message at runtime, which will stop your program abnormally.

UNIT- 3
Introduction to C++

C++ Character Sets:

Letters	A-Z , a-z
Digits	0-9
Special Symbols	Space + - * / ^ \ ( ) [ ] { } = != <> . „ “ $, ; : % ! & ? _ # <= >= @
White Spaces	Blank spaces, horizontal tab, carriage return
Other Characters	Any of the 256 ASCII character

Token:-The smallest individual unit in a program is known as token. Tokens used in C++ are:

KEYWORDS :

Keywords are the certain reserved words that convey a special meaning to the compiler. These are reserve for special purpose and must not be used as identifier name.eg for , if, else , this , do, etc.

IDENTIFIERS:

Identifiers are programmer defined names given to the various program elements such as variables, functions, arrays, objects, classes, etc.. It may contain digits, letters and underscore, and must begin with a letter or underscore. C++ is case sensitive as it treats upper and lower case letters differently. The following are some valid identifiers:

Pen time580 s2e2r3 _dos _HJI3_JK

LITERALS:

The data items which never change their value throughout the program run. There are several kinds of literals:

- Integer literals
- Character literals
- Floating literals
- String literals

Integer literals :

Integer literals are whole numbers without any fractional part. An integer literal must have at least one digit and must not contain any decimal point. It may contain either + or - sign. A number with no sign is assumed as positive. C++ allows three types of integer literals:

(i) Decimal Integer Literals:- An integer literal without leading 0 (zero) is called decimal integer literals e.g., 123, 786 , +97 , etc.

(ii) Octal Integer Literals:- A sequence of octal digit starting with 0 (zero) is taken to be an octal integer literal (zero followed by octal digits). e.g., 0345, 0123 , etc.

(iii) Hexadecimal Integer Literals :- Hexadecimal Integer Literals starts with 0x or 0X followed by any hexa digits. e.g., 0x9A45, 0X1234, etc.

Character literals:

Any single character enclosed within single quotes is a character literal.

e.g ‘ A’ , ‘3’

Floating literals:

Numbers which are having the fractional part are referred as floating literals or real literals. It may be a positive or negative number. A number with no sign is assumed to be a positive number.

e.g 2.0, 17.5, -0.00256

String Literals:

It is a sequence of character surrounded by double quotes. e.g., "abc" , "23".

PUNCTUATORS:

The following characters are used as punctuators which are also known as separators in C++

[ ] { } ( ) , ; : * …….. = #

Punctuator	Name	Function
[ ]	Brackets	These indicates single and multidimensional array subscripts
()	Parenthesis	These indicate function calls and function parameters.
{ }	Braces	Indicate the start and end of compound statements.
;	Semicolon	This is a statement terminator.
,	Comma	It is used as a separator.
:	Colon	It indicates a labeled statement
*	Asterisk	It is used as a pointer declaration
…	Ellipsis	These are used in the formal argument lists of function prototype to indicate a variable number of arguments.
=	Equal to	It is used as an assigning operator.
#	Pound sign	This is used as preprocessor directives.

OPERATORS:

An operator is a symbol or character or word which trigger some operation (computation) on its operands.

(i) **Unary operators**: Those which require only one operand to operate upon. e.g. unary - , unary + , ++ , - - ! .

(ii) Binary operators: Binary operators require two operands to operate upon. e.g. +, *, /, -, etc.

(iii) Ternary Operator : Ternary operator require three operands to operate upon. Conditional operator (? :) is a ternary operator in C++.

Structure of a C++ program:

Structure of a C++ program	A C++ program to prints a string on the screen

#include<iostream.h>
void main ()
{
Statement_1;
Statement_2;
:
:
}
	#include<iostream.h>
	void main ()
	{
	cout<< "Kendriya Vidyalaya";
	}
	
	The program produces following output:
	Kendriya Vidyalaya

The above program includes the basic elements that every C++ program has. Let us check it line by line

#include<iostream.h> : This line includes the preprocessor directive include which includes the header file iostream.h in the program. The header file iostream.h is included in every C++ program to implement input/ output facilities.

Preprocessor Directives:

#include is the preprocessor directive used in C++ programs. This statement tells the compiler to include the specified file into the program. This line is compiled by the processor before the compilation of the program.

e.g #include<iostream.h>

the above line include the header file ***iostream*** into the program for the smooth running of the program.

void main () : This line indicate the beginning of the main( ) function. The main( ) function is the point by where all C++ program begin their execution**. void** is the keyword used to specify the return type of any function (when the function has no return values).

cout<< "Kendriya Vidyalaya"; : This statement prints the sequence of string "Kendriya Vidyalaya" into this output stream i.e. on monitor.

Every executable statement in C++ will be terminated by a semicolon (;) which specifies the end of statement.

COMMENTS IN A C++ PROGRAM.:

Comments are the pieces of code that compiler ignores to compile. There are two types of comments in C++.

1. **Single line comment:** The comments that begin with // are single line comments. The Compiler simply ignores everything following // in the same line.

2. **Multiline Comment :** The multiline comment begin with /* and end with */ . This means everything that falls between /* and */ is consider a comment even though it is spread across many lines. e.g

```
// A sample program
#include<iostream.h>
void main ()
{
cout<< "  hello world";
/* this is the program to print hello world
      For demonstration of comments */
}
```

Input Output (I/O) operations In C++: Input & Output operations are supported by the istream (input stream) and ostream (output stream) classes. The predefined stream objects for input, output are :

(i) The cout Object:

The identifier cout is a predefined object of ostream class that represents the standered output stream in C++ and tied to slandered output. cout stands for console output . cout sends all output to the standard output device i.e. monitor.

The syntax of **cout** is as follows:

cout<< data;

Where data may be a variable or constant or string etc. e.g.

cout<< a ; *(here a can be any variable)*

Output Operator (<<): The output operator (<<) is also known as 'stream insertion' or 'put to' operator. It directs the contents of the variable (or value) on its right to the object on its left (i.e., cout).

(ii) The cin Object :

The cin object is an istream class object tied to slandered input. cin stands for console input.

cin object used to get input from the keyboard. When a program reaches the line with cin, the user at the keyboard can enter values directly into variables.

The syntax of **cin** is as follows:

cin>> variablename;

e.g

cin>> ch; (here ch can be any variable)

input Operator (>>): The input operator (>>) is also known as extraction or 'get from' operator . It extracts (or takes) the value from the keyboard and assign it to the variable on its right.

CASCADING OF OPERATOR:

When input or output (>>or <<) are used more than one time in a single statement then it is called as cascading of operators.

e.g cout<< roll<< age<< endl;

DATA TYPES IN C++:

Data types are means to identify the types of data and associated operations of handling it. Data types in C++ are of two types:

1. Fundamental or Built-in data types .
2. Derived data types.

1. Fundamental or Built-in data types: These data types are already known to compiler. These are the data types those are not composed of other data types. There are following fundamental data types in C++:

(i) **int data type (for integer)** :- int data type is used for integer value. An identifiers declare as int cannot have fractional part.

(iii) **char data type (for characters):-** An identifiers declare as char can store a character.

(iv) **float data type (for floating point numbers):-** An identifier declare as float can hold a floating point number.

(v) **double data type (for double precision floating point numbers):-** The double data type is also used for handling floating point numbers but it occupies twice as much memory as float and store numbers with much larger range and precision.

Data Type Modifiers:-There are following four data type modifiers in C++ , which may be used to modify the fundamental data types to fit various situations more precisely:

(i) signed

(ii) unsigned

(iii) long

(iv) short

signed, unsigned, long, short data type modifiers may be apply to char & int data types. However you may also apply long to double

Data Type	Size (in Bytes)	Range
char	1	-128 to 127
unsigned char	1	0 to 255
Signed char	1	Same as char
int	2	-32768 to 32767
unsigned int	2	0 to 65535
signed int	2	Same as int
short (or short int)	2	-32768 to 32767
long (or long int)	4	-2147483648 to 2147483647
float	4	3.4×10^{-38} to $3.4 \times 10^{38} - 1$ (upto 7 digits of precision)
double	8	1.7×10^{-308} to $1.7 \times 10^{308} - 1$ (upto 15 digits of precision)
long double	10	3.4×10^{-4932} to $1.1 \times 10^{4932} - 1$ (upto 19 digits of precision)

2**. Derived Data Types**:- These are the data types that are composed of fundamental data types. e.g., array, class, structure, etc.

Variables:-A named memory location, whose contains can be changed with in program execution is known as variable. OR

A variable is an identifier that denotes a storage location, which contains can be varied during program execution.

Declaration of Variables:- All variables must be declared before they are used in executable statements. Variable declaration reserves memory required for data storage and associates it with a name. Syntax for variable declaration is:

datatypes variable_name1, variable_name2, variable_name3,……. ;

e.g.,

```
int num;
int num, sum, avg;
```

We can also initialize a variable at the time of declaration by using following syntax:

datatypes variable_name = value;

e.g.,

```
int num = 0;
```

In C++ both the declaration and initialization of a variable can be done simultaniouly at the place where the variable is used first time this feature is known as dynamic initialization. e.g.,

```
float avg;
avg = sum/count;
```

then above two statements can be combined in to one as follows:

```
float avg = sum/count;
```

Constant:- A named memory location, whose contains cannot be changed with in program execution is known as constant. OR

A constant is an identifier that denotes a storage location, which contains cannot be varied during program execution.

Syntax for constant declaration is:

const datatypes constant_name = value ;

e.g.,

```
const float pi = 3,14f ;
```

Formatted Output (Manipulators) :

Manipulators are the operators used with the insertion operator << to format the data display. The most commonly used manipulators are **endl** and **setw.**

1. **The endl manipulator :**The endl manipulator, when used in a output statement , causes a line feed to be inserted. It has same effect as using new line character "\n". e.g.,

```
cout<< " Kendriya Vidyalaya Sangathan"<<endl;
cout<< " Human Resource and Development";
```

The output of the above code will be

Kendriya Vidyalaya Sangathan
Human Resource and development

2. **The setw( ) Manipulator** : The **setw( )** manipulator sets the width of the field assign for the output. It takes the size of the field (in number of character) as a parameter. The output will be right justified e.g., the code :

```
cout<<setw(6)<<"R" ;
```

Generates the following output on the screen (each underscore represent a blank space)

_ _ _ _ _ R

In order to use these manipulator , it is must to include header file iomanip.h

Operators:-

Arithmetic operators :-Those operators are operates only on numeric data types operands are known as arithmetic operators.

Operator	Operation	Example
Unary -	Result is the negation of operand's value (Reverse the sign of operand's value)	If a=5 then – a means -5. If a = - 4 then – a means 4.
Unary +	The result is the value of its operand	If a=5 then +a means 5. If a = - 4 then +a means -4.
+ (Addition Operator)	Addition (it adds two numbers)	4+20 results is 24. If a = 5 then a + 5 results 10.
- (Subtraction Operator)	Subtraction (Subtract the second operand from first)	14 – 3 evaluates to 12. If a = 2 , b = 3 then b – a evaluates 1.
* (Multiplication Operator)	Multiplies the values of its operands	3*4 evaluates to 12. If a=2, b=3 then a*b evaluates to 6.
/ (Division Operator)	Divides its first operand by the second	100/5 evaluates 20. If a =10 , b = 5 then a/b evaluates 2
% (Modulus Operator)	It produce the remainder of dividing the first operand by second	19%6 evaluates to 1. If a = 14 , b = 3 then a%b evaluates to 2. Modulus operator requires that both operands be integer and second operand be non-zero.

Increment and Decrement Operators (++ , - -) :

The increment operator (++) adds 1 to its operand and decrement operator (--) subtract one from its operand. In other word

a = a + 1; is same as ++a; or a++;

& a = a – 1 ; is same as --a; or a--;

Both the increment & decrement operators comes in two version :

(i) Prefix increment/decrement :- When an increment or decrement operator precedes its operand, it is called prefix increment or decrement (or pre-increment / decrement). In

prefix increment/decrement , C++ perform the increment or decrement operation before using the value of the operand. e.g.,
If sum = 10 and count =10 then
Sum = sum +(++count);
First count incremented and then evaluate sum = 21.

(ii) Postfix increment/decrement :- When an increment or decrement operator follows its operand, it is called postfix increment or decrement (or post-increment / decrement). In postfix increment/decrement , C++ first uses the value of the operand in evaluating the expression before incrementing or decrementing the operand's value. e.g.,
If sum = 10 and count =10 then
Sum = sum +(count++);
First evaluate sum = 20 , and then increment count to 11.

Relational Operator: These operators are used to compare two values. If comparison is true, the relational expression results into the value 1 and if the comparison is false its result will be 0. The six relational operators are:

Operator	Meaning
= =	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Logical Operators : In addition to the relational operator, C++ contains three logical operators. Relational operators often are used with logical operators to construct more complex decision making expressions.

Operators	Use	Return True if
&& (Logical AND)	op1 && op2	op1 and op2 are both true
|| (Logical OR)	op1 || op2	Either op1 or op2 is true
! (Logical NOT)	!op1	op1 is false (it is unary operator)

Assignment Operator: C++ offers an assignment operator (=) to assign a value to an identifier. The assignment statement that make use of this operator are written in the form :

var = expression ;

where var generally represents a variable and expression may be a constant or a variable or an expression.

C++ offers special shorthand operators that simplify the coding of a certain type of assignment statement . e.g.,

a = a + 10 ; can be written as a+=10 ;

This shorthand works for all binary arithmetic operators. The general form of this shorthand is

Var = var operator expression ; is same as
var operator = expression ;

Following are some examples of C++ shorthands:

x -=10 ;	equivalent to	x = x -10 ;
x*=3 ;	equivalent to	x = x * 3 ;
x/=2 ;	equivalent to	x = x/2 ;
x%=z	equivalent to	x = x % z ;

Conditional operator (? :)

The conditional operator (? :) is a ternary operator i.e., it require three operands. The general form of conditional operator is:

expression1? expression2: expression3 ;

Where expression1 is a logical expression , which is either true or false.
If expression1 evaluates to true i.e., 1, then the value of whole expression is the value of expression2, otherwise, the value of the whole expression is the value of expression3. For example

min = a<b? a : b ;

Here if expression (a<b) is true then the value of a will be assigned to min otherwise value of b will be assigned to min.

Comma operator (,)

The comma operator (,) is used to separate two or more expressions that are included where only one expression is expected. When the set of expressions has to be evaluated for a value, only the rightmost expression is considered.

For example, the following code:

a = (b =3 , b +2);

Would first assign the value 3 to b, and then assign b+2 to variable a. So, at the end, variable a would contain the value 5 while variable b would contain value 3.

sizeof()

This operator returns the size of its operand in bytes. The operand may be an expression or identifier or it may be a data type.

a= sizeof (char);

This will assign the value 1 to a because char is a one-byte long type.

Expressions:-An expression in C++ is any valid combination of operators, constants, and variables**.**

Pure Expressions:-If an expression have all operand of same data types then it is called a pure expression.

Mixed Expressions :-If an expression have operands of two or more different data types then it is called a mixed expression.

Arithmetic Expressions:-Arithmetic expression can either be integer expressions or real expressions. Sometimes a mixed expression can also be formed which is a mixture of real and integer expressions.

Integer Expressions:- Integer expressions are formed by connecting all integer operands using integer arithmetic operators.

Real Expressions:- Real expressions are formed by connecting real operands by using real arithmetic operators.

Logical Expressions:- The expressions which results evaluates either 0 (false) or 1 (true) are called logical expressions. The logical expressions use relational or Logical operators.

Type Conversion:-The process of converting one predefined data type into another is called type conversion.

C++ facilitates the type conversion in two forms:

(i) **Implicit type conversion**:- An implicit type conversion is a conversion performed by the compiler without programmer's intervention. An implicit conversion is applied generally whenever different data types are intermixed in an expression. The C++ compiler converts all operands upto the data type of the largest data type's operand, which is called type promotion.

(ii) **Explicit type conversion** :- An explicit type conversion is user-defined that forces an expression to be of specific data type.

Type Casting:- The explicit conversion of an operand to a specific type is called type casting.

Type Casting Operator - (type) :-Type casting operators allow you to convert a data item of a given type to another data type. To do so , the expression or identifier must be preceded by the name of the desired data type , enclosed in parentheses . i. e.,

(data type) expression

Where data type is a valid C++ data type to which the conversion is to be done. For example , to make sure that the expression (x+y/2) evaluates to type float , write it as:

(float) (x+y/2)

Precedence of Operators:- Operator precedence determines which operator will be performed first in a group of operators with different precedence. For instance

5 + 3 * 2 is calculated as 5 + (3 * 2), giving 11

From greatest to smallest priority, C++ operators are evaluated in the following order:

Level	Precedence group	Operator	Description	Grouping
1	Scope	::	scope qualifier	Left-to-right
2	Postfix (unary)	++ --	postfix increment / decrement	Left-to-right
		()	functional forms	
		[]	subscript	
		. ->	member access	
3	Prefix (unary)	++ --	prefix increment / decrement	Right-to-left
		~ !	bitwise NOT / logical NOT	
		+ -	unary prefix	
		& *	reference / dereference	
		new delete	allocation / deallocation	
		sizeof	parameter pack	
		(*type*)	C-style type-casting	
4	Pointer-to-member	.* ->*	access pointer	Left-to-right
5	Arithmetic: scaling	* / %	multiply, divide, modulo	Left-to-right
6	Arithmetic: addition	+ -	addition, subtraction	Left-to-right
7	Bitwise shift	<<>>	shift left, shift right	Left-to-right
8	Relational	<><= >=	comparison operators	Left-to-right
9	Equality	== !=	equality / inequality	Left-to-right
10	And	&	bitwise AND	Left-to-right
11	Exclusive or	^	bitwise XOR	Left-to-right
12	Inclusive or	|	bitwise OR	Left-to-right
13	Conjunction	&&	logical AND	Left-to-right
14	Disjunction	||	logical OR	Left-to-right
15	Assignment-level expressions	= *= /= %= += -= >>= <<= &= ^= |=	assignment / compound assignment	Right-to-left
		?:	conditional operator	
16	Sequencing	,	comma separator	Left-to-right

When an expression has two operators with the same precedence level, *grouping* determines which one is evaluated first: either left-to-right or right-to-left.

Practice Questions:

1. What is the name of the function that should be present in all c++ program?

Ans: main()

2. What are C++ comments?

Ans: comments are internal documentation of a program which helps the program for many purposes.

3. What is indentation of a program?

Ans: It is the systematic way of writing the program which makes it very clear and readable.

4. What is #include directives?

Ans :it instructs the compiler to include the contents of the file enclosed within the brackets into the source file.

5. What is role of main() in c++ program?

Ans: This is the first line that a C++ compiler executes. Program starts and end in this function.

6. What is a header file?

Ans: Header file provide the declaration and prototypes for various token in a program.

7. What is the purpose of comments and indentation?

Ans: The Main purpose of comments and indentation is to make program more readable and understandable.

8. What are console input /output functions?

Ans: Console I/O functions are cout and cin.

9. Write an appropriate statement for each of the following:
    1. Write the values for a & b in one un separated by blanks and value of after two blanks lines.

    Ans: cout<<a<<b<<endl<<endl<<c;

    2. Read the values for a,b and c.

    Ans: cin>>a>>b>>c;

    3. Write the values for a and b in one line, followed by value of c after two blank lines.

    Ans: cout<a<<b<<'\n\n'<<c;

10. What type of errors occurs while programming?

Ans: There are three types of errors generally occur are:

1. Syntax error
2. Semantic error
3. Type error.

11. How '/' operator is different from '%' operator?

Ans: '/' operator is used to find the quotient whereas % operator is used to find the remainder.

12. Which type of operator is used to compare the values of operands?

Ans: Relational operators.

13. How will you alter the order of evaluation of operator?

Ans: We can use parentheses to alter the order of evaluation of an equation.

14. What is the unary operator? Write 2 unary operator .

Ans : The operator which needs only one operand is called as unary operator .The '++' (increment) and '_ _'(decrement) operators.

15. What is output operator and input operator?

Ans: The output operator ("<<") is used to direct a value to standard output. The input operator (">>") is used to read a value from standard input.

16. What will be the output of following code:

```
void main()
{
int j=5;
cout<<++j<<j++<<j;  // in cascading processing starts from right to left
}
```
Ans. 7 5 5

17. What will be the output of following code:
```
void main()
{
int j=5;
cout<<++j + j++ +j++;  // values will be: 6 6 7 (From left to right)
}
```
Ans. 19

18. What will be the output of following code:
```
void main()
{
int j=5, k;
k= a++ +a+ ++a;
cout<<k;
}
```
Ans. 18 (Because in evaluation of expression first of all prefix are evaluated, then it's value is assigned to all occurrences of variable)

UNIT-4

Programming in C++

Statements:-Statements are the instructions given to the Computer to perform any kind of action.

Null Statement:-A null statement is useful in those case where syntax of the language requires the presence of a statement but logic of program does not give permission to do anything then we can use null statement. A null statement is nothing only a ;. A null (or empty statement have the following form:

```
;        // only a semicolon (;)
```

Compound Statement :-A compound statement is a group of statements enclosed in the braces { }. A Compound statement is useful in those case where syntax of the language requires the presence of only one statement but logic of program have to do more thing i.e., we want to give more than one statement in place of one statement then we can use compound statement.

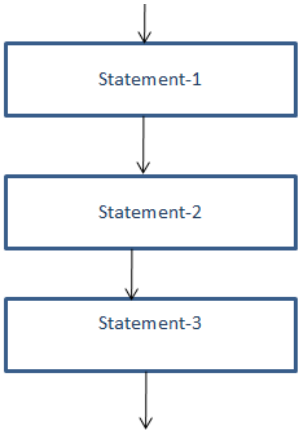
```
{
        St-1;
        St-2;
        :
        :
}
```

Statement Flow Control:-In a program , statements may be executed sequentially, selectively, or iteratively.

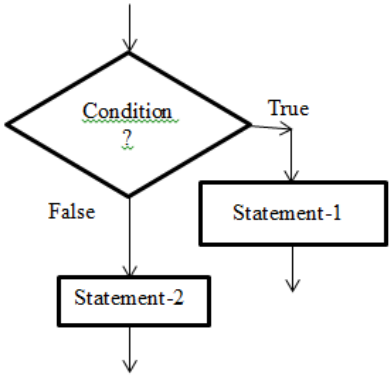
Every programming language provides three constructs:

1. Sequence Constructs
2. Selection Constructs
3. Iteration Constructs

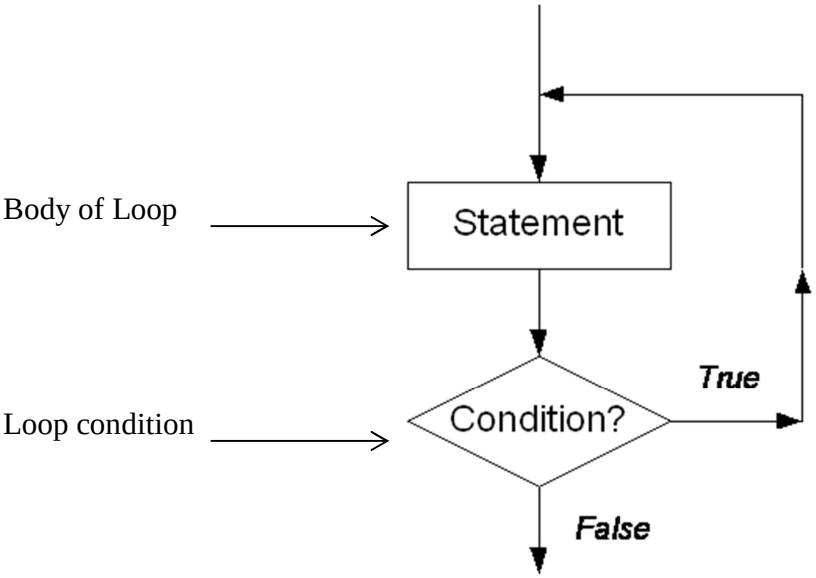
Sequence Construct:-The sequence construct means the statements are being executed sequentially. It represents the default flow of statements**.**



Selection Construct:- The selection construct means the execution of statement(s) depending on a condition. If a condition is true, a group of statements will be execute otherwise another group of statements will be execute.

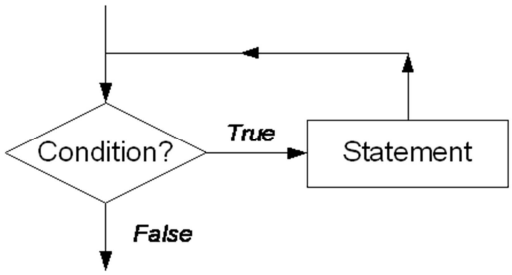


Looping or Iteration Statements:-Looping the iteration construct means repetition of set of statements depending upon a condition test. The iteration statements allow a set of instructions to be performed repeatedly until a certain condition is true.

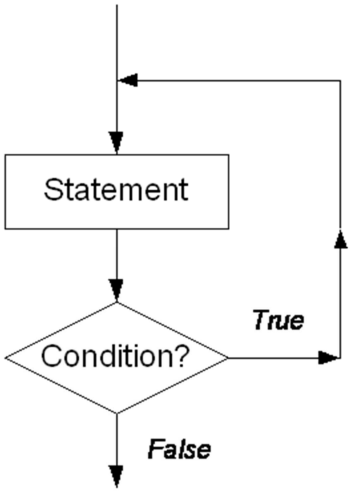


There are two types of loops:-

1. **Entry-controlled loop :-**In entry-controlled loop first of all loop condition is checked and then body of loop is executed is condition is true. If loop condition is false in the starting the body of loop is not executed even once.



2. **Exit-controlled loop :-**In exit-controlled loop first body of loop is executed once and then loop condition is checked. If condition is true then the body of loop will be executed again.
   It means in this type of loop, loop body will be executed once without checking loop condition.

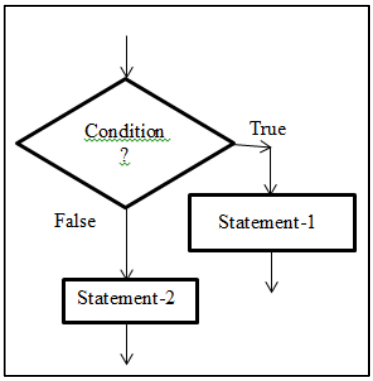


Selection Statements :-There are two types of selection statements in C++ :

1. if statement
2. switch statement

1. if Statement : If statement have three forms

(a) if … else Statement :- It is useable, when we have to performs an action if a condition is True and we have to perform a different action if the condition is false. The syntax of if…else statement is:




```
if ( < conditional expression > )
{
< statement-1 or block-1>;
    // statements to be executed when conditional expression is true.
            }
else
{
< statement-2 or block-2>;
    // statements to be executed when conditional expression is false.
}
```

If the <conditional expression> is evaluated to true then the < statement-1 or block-1> (statement under if ( ) block) will be executed otherwise the <statement-2 or block-2> (statements under else block) would be executed. if there exists only one program statement under if( ) block then we may omit curly braces { }.

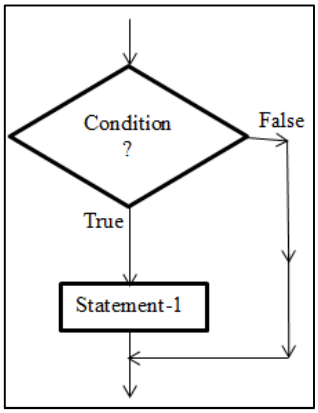
For example, a program to check whether a given number is greater than 100 or not is given below:

```
#include <iostream.h>
void  main()
{
int x;
cout<< "\nEnter a number: ";
cin>> x;
if( x > 100 )
cout<< "That number is greater than 100\n";
else
cout<< "That number is not greater than 100\n";
}
```

In this program if the conditional expression (x>100) in the if statement is true, the program prints one message; if it isn't, it prints the other.

Here's output from two different invocations of the program:

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



(b) **Simple if statement:-** The else part in if … else statement is optional, if we omit the else part then it becomes simple if statement. This statement is usable, when we have to either perform an action if a condition is True or skips the action if the condition is false. The syntax of simple if statement is:

```
if ( < conditional expression > )
{
    < statement-1 or block-1>;
    // statements to be executed when conditional expression is true.
}
```

Here <statement-1 or block-1> will be executed only if <conditional expression > evaluates true. if there exists only one program statement under if( ) block then we may omit curly braces { }. eg.,

```
#include <iostream.h>
void  main()
{
int x;
cout<< "Enter a number: ";
cin>> x;
if( x > 100 )
cout<< "That number is greater than 100\n";

}
```

In the above example, program's output when the number entered by the user is greater than 100:

```
Enter a number: 2000
That number is greater than 100
```

If the number entered is not greater than 100, the program will terminate without printing the second line.

(c) **The if-else-if ladder :-**This statement allows you to test a number of mutually exclusive cases and only execute one set of statements for which condition evaluates true first.

The syntax is:

```
if ( <condition -1> )
        statement-1;        // do something if condition-1 is satisfied (True)
else if ( <condition – 2 >)
        statement-3 ;    // do something if condition -2 is satisfied (True)
else if (<condition – 3 >)
        statement-3 ;        // do something if condition- 3 is satisfied (True)
                                :
                            :    // many more n-1 else - if ladder may come
                            :
else if( < condition – n >)
        statement-n ;            // do something if condition – n is satisfied (True)
else
        statement-m ;            // at last do here something when none of the
                                 // above  conditions gets satisfied (True)
}
```

<>	in syntax is known as a place holder, it is not a part of syntax, do not type it while writing program. It only signifies that anything being kept there varies from program to program.
[ ]	is also not a part of syntax , it is used to mark optional part of syntax i.e. all part of syntax between [ ] is optional.

In the above syntax there are ladder of multiple conditions presented by each if( ) , all of these conditions are mutually exclusive. If one of them would evaluates true then the statement followed that condition will be executed and all the conditions below it would not be evaluated (checked).

Say suppose if condition-3 gets satisfy (i.e. evaluates true value for the condition), then statement-3 gets executed and all other conditions below it would be discarded.

If none of the n if ( ) conditions gets satisfied then the last else part always gets executed. It is not compulsory to add an else at the last of the ladder.

We can also write a group of statement enclosed in curly braces { } (as a compound statement) in place of any statement (statement-1. Statement-2,……., statement-n) if required in above syntax.

For example, a program which accept number of week's day (1-7) and print is equivalent name of week day (Monday for 1,……, Sunday for 7). using the if-else-if ladder is given below:

```
#include< iostream.h>
#include<conio.h>
void main()
{
        int day;
        cout<<" Enter a  number (between 1 and 7):"<< endl;
        cin>>day;
        if (day= =1)
                cout<<" Monday"<< endl;
        else if (day= =2)
                cout<<" Tuesday"<<endl;
        else if( day= =3)
                cout<<" Wednesday"<<endl;
        else if(day= =4)
                cout<<" Thursday"<<endl;
        else if(day= =5)
                cout<<" Friday"<< endl;
        else if(day= =6)
                cout<<" Saturday"<< endl;
        else if(day= =7)
                cout<<" Sunday"<< endl;

        else
                cout<<" You enter a wrong number"<< endl;
        getch();
}
```

Nested if Statement:- If an if statement is written in the if or else clause of another if statement then it is known as nested if. Some possible syntax of nested if statements given below:

Syntax 1:-

```
if ( <outer- condition > )
 {
        if ( <inner-condition> )
        {

                //some statements to be executed
                // on satisfaction of  inner  if ( ) condition.

        } // end of scope of inner if( )
        //some statements to be executed
        // on satisfaction of outer  if ( ) condition.

} // end of the scope of outer if( )
```

Syntax 2:-

```
if ( <outer- condition > )
 {
        if ( <inner-condition> )
        {

                //some statements to be executed
                // on satisfaction of  inner  if ( ) condition.

        }
        else
        {
                // statements on failure of inner if( )
        }

                //some statements to be executed
                // on satisfaction of outer  if ( ) condition.

        }
 else
 {

                // statements on failure of outer if( )
 }
```

2. switch Statement :-This is multi-branching statement. Syntax of this statement is as follows:

```
        switch (expression/variable)
        {       case value_1:  statement -1;
                               break;
                case value_2:  statement -2;
                               break;
                :
                :
                case value_n:  statement -n;
                               break;
                [  default:  statement -m    ]
        }
```

Note: expression/variable should be integer or character type only.

When the switch statement is executed, the expression/variable is evaluated and control is transferred directly to the statement whose case label value matches the value of expression/ variable. If none of the case label value matches the value of expression/variable then only the statement following the default will be executed. If no default statement is there and no match is found then no action take place. In this case control is transferred to the statement that follows the switch statement.

For example, a program which accept number of week's day (1-7) and print is equivalent name of week day (Monday for 1,……, Sunday for 7). using switch-case statement is given below:

```
#include <iostream.h>
#include<conio.h>
void main()
{
```

```
        int day;
        cout<<" Enter a number (between  1 and 7):"<< endl;
        cin>>day;
        switch(day)
        {
        case  1:
                cout<< " Monday"<< endl;
                break;
        case 2:
                cout<<" Tuesday"<< endl;
                break;
        case 3:
                cout<<" Wednesday"<< endl;
                break;
        case 4:
                cout<<" Thursday"<< endl;
                break;
        case 5:
                cout<<" Friday" << endl;
                break;
        case 6:
                cout<<" Saturday" << endl;
                break;
        case 7:
                cout<<" Sunday"<< endl;
                break;
        default:
                Cout<<" You enter a wrong number "<< endl;
        }
                getch();
}
```

Loops in C++:-There are three loops or iteration statements are available in C++

1. for loop
2. while loop
3. do…. while loop

1. **The for Loop**:For loop is an entry control loop the syntax of for loop is :

```
for(initialization_expression(s); loop_Condition; update_expression)
        {
                Body of loop
        }
```

Working of the for Loop:-

1. The *initialization_expression* is executed once, before anything else in the for loop.
2. The *loop condition* is executed before the body of the loop.
3. If loop condition is true then body of loop will be executed.
4. The *update expression* is executed after the body of the loop
5. After the *update expression* is executed, we go back and test the *loop condition* again, if loop_condition is true then body of loop will be executed again, and it will be continue until loop_condition becomes false.

Example:

```
for (int i = 0; i < 7; i++)
cout<< i * i << endl;
```

Interpretation:

An int i is declared for the duration of the loop and its value initialized to 0. i^2 is output in the body of the loop and then i is incremented. This continues until i is 7.

Example: A C++ Program to Print a table of factorials (from 0 to 9).

```
#include <iostream.h>
void main( )
{
        int factorial =1;
        for(int i=0; i<10; i++)
        {
                if (i!=0)
                factorial*=i;
                cout<<i<<"\t"<<factorial;
        }
}
```

2. **while Loop:-** while loop is also an entry controlled loop. The syntax of while loop is :

```
while (loop_condition)
        {
                Loop_body
        }
```

Where the Loop_body may contain a single statement, a compound statement or an empty statement.

The loop iterates (Repeatedly execute) while the loop_condition evaluates to true. When the loop_condition becomes false, the program control passes to the statement after the loop_body. In while loop , a loop control variable should be initialized before the loops begins. The loop variable should be updated inside the loop_body.

For example the program:

```
const int MAX_COUNT = 10;
count = 0;
while (count < MAX_COUNT)
        {
                cout<< count << " ";
                count ++;
                }
```

Will give the output:

0 1 2 3 4 5 6 7 8 9

3. **do-while loop:-** do-while loop is an exit-controlled loop i.e. it evaluates its loop_condition at the bottom of the loop after executing its loop_body statements. It means that a do-while loop always executes at least once. The syntax of do-while loop is:

```
do
        {
                Loop_body
        }while (loop_condition);
```

In do-while loop first of all loop_body will be executed and then loop_condition will be evaluates if loop_condition is true then loop_body will be executed again, When the

loop_condition becomes false, the program control passes to the statement after the loop_body.

For example the program:

```
const int MAX_COUNT = 10;
count = 0;
do
{
cout<< count << " ";
count ++;
} while (count < MAX_COUNT) ;
```

Will give the output:

0 1 2 3 4 5 6 7 8 9

Nested Loops :-Any looping construct can also be nested within any other looping construct . Let us look at the following example showing the nesting of a for( ) loop within the scope of another for( ) loop :

➢

```
for(int i = 1 ; i<=2 ; i++)  ← Outer for( ) loop
{
    for( int j = 1 ; j<=3 ; j++)  ← Inner for( ) loop
        {
            cout<< i * j <<endl ;
        }
    }
```

For each iteration of the outer for loop the inner for loop will iterate fully up to the last value of inner loop iterator. The situation can be understood more clearly as :

1st Outer Iteration
i= 1
1st Inner Iteration
j = 1 , output : 1 * 1 = 1
2nd Inner Iteration
j = 2 , output : 1 * 2 = 2
3rd Inner Iteration
j = 3 , output : 1 * 3 = 3

2nd Outer Iteration
i= 2
1st Inner Iteration
j = 1 , output : 2 * 1 = 1
2nd Inner Iteration
j = 2 , output : 2 * 2 = 4
3rd Inner Iteration
j = 3 , output : 2 * 3 = 6

You can observe that j is iterated from 1 to 2 every time i is iterated once.

Jump Statements:-These statements unconditionally transfer control within function . In C++ four statements perform an unconditional branch :

1. return
2. goto
3. break
4. continue

1. **return Statement:-** The return statement is used to return from a function. It is useful in two ways:

(i) An immediate exit from the function and the control passes back to the operating system which is main's caller.

(ii) It is used to return a value to the calling code.

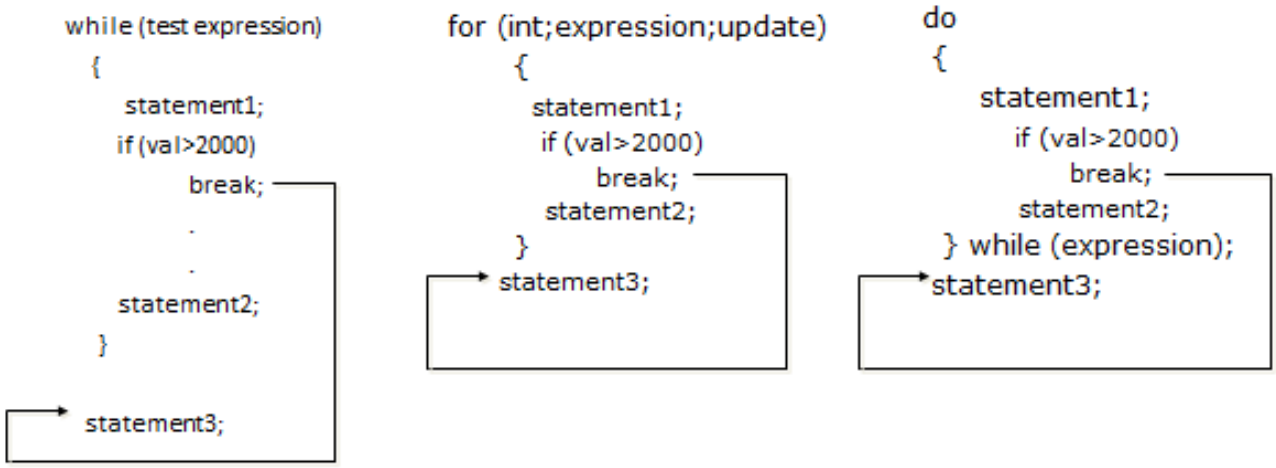
2. **goto statement :-** A goto Statement can transfer the program control anywhere in the program. The target destination of a goto statement is marked by a *label*. The target label and goto must appear in the same function. The syntax of goto statement is:

```
goto label;
        :
label :
```

Example :

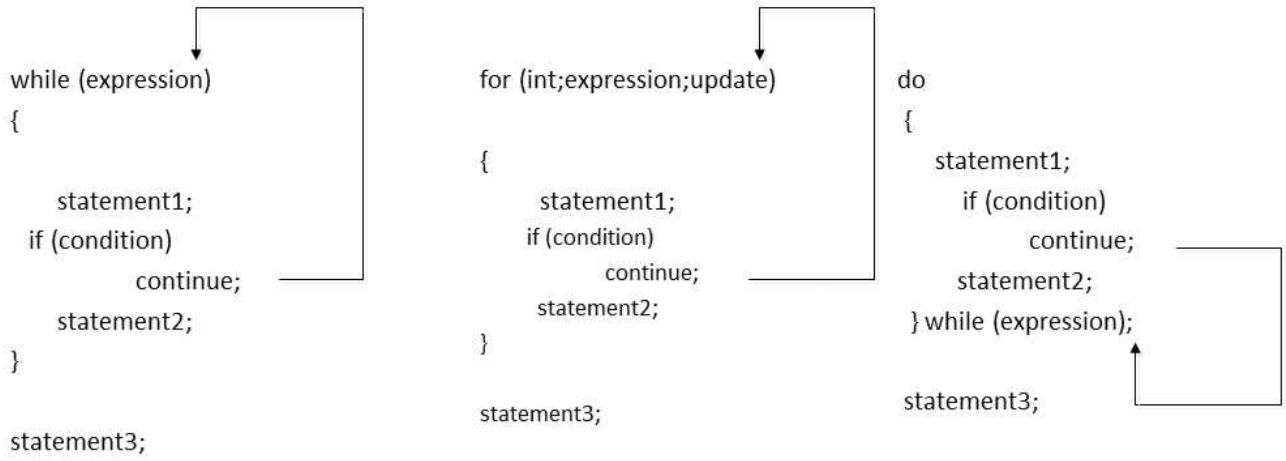
```
a= 0;
start :
        cout<<"\n" <<++a;
if(a<50) goto start;
```

3. **break Statement :-** The break statement enables a program to skip over part of the code. A break statement terminates the smallest enclosing while, do-while, for or switch statement. Execution resumes at the statement immediately following the body of the terminated statement. The following figure explains the working of break statement:



The Working of a Break Statement

4. **continue Statement**:- The continue is another jump statement like the break statement as both the statements skip over a part of the code. But the continue statement is somewhat different from break. Instead of forcing termination, it forces the next iteration of the loop to take place, skipping any code in between. The following figure explains the working of continue statement:



The Working of Continue Statement

Functions :- Function is a named group of programming statements which perform a specific task and return a value.

There are two types of functions:-

1. Built-in (Library)functions
2. User defined functions

Built-in Functions (Library Functions) :- The functions, which are already defined in C++ Library (in any header files) and a user can directly use these function without giving their definition is known as built-in or library functions. e.g., sqrt( ), toupper( ), isdigit( ) etc.

Following are some important Header files and useful functions within them :

stdio.h (standard I/O function)	gets( ) , puts( )
ctype.h (character type function)	isalnum( ) , isalpha( ), isdigit ( ), islower (), isupper ( ), tolower ( ), toupper( )
string.h (string related function)	strcpy ( ), strcat ( ), strlen( ), strcmp( ) , strcmpi( ) , strrev( ),strupr( ) , strlwr( )
math.h (mathematical function)	fabs ( ), pow ( ), sqrt ( ), sin ( ), cos ( ), abs ( )
stdlib.h	randomize ( ), random ( )

The above list is just few of the header files and functions available under them , but actually there are many more. The calling of library function is just like User defined function , with just few differences as follows:

i) We don't have to declare and define library function.

ii) We must include the appropriate header files , which the function belongs to, in global area so as these functions could be linked with the program and called.

Library functions also may or may not return values. If it is returning some values then the value should be assigned to appropriate variable with valid datatype.

gets( ) and puts( ) : these functions are used to input and output strings on the console during program run-time.

gets( ) accept a string input from user to be stored in a character array.
puts() displays a string output to user stored in a character array.

Program: Program to use gets( ) and puts( )

```
#include<iostream.h>
#include<stdio.h>  // must include this line so that gets( ) , puts( ) could be linked and
                   //      called
void main( )
{
        char myname[25] ;   //declaring a character array of size 25

        cout<<"input your name : ";
        gets(myname)   ;   // just pass the array name into the parameter of the function.
        cout<< "You have inputted your name as : " ;
        puts(myname);
}
```

i**salnum( ) , isalpha( ), isdigit( ) :** checks whether the character which is passed as parameter to them are alphanumeric or alphabetic or a digit ('0' to '9') . If checking is true functions returns 1.

Program : Program to use isalnum( ) , isalpha( ), isdigit( )

```
#include<iostream.h>
#include<ctype.h>
void main( )
{
        char ch;
        cout<<"Input a character";
        cin>>ch;
        if( isdigit(ch) = = 1)
          cout<<"The inputed character is a digit";
        else if(isalnum(ch) = = 1)
          cout<<"The inputed character is an alphanumeric";
        else if(isalpha(ch) = = 1)
          cout<<"The inputed character is an alphabet.
}
```

islower ( ),isupper ( ), tolower ( ), toupper( ) : islower( ) checks whether a character is lower case , isupper( ) check whether a character is upper case . tolower( ) converts any character passed to it in its lower case and the toupper( ) convert into upper case.

Program: Program to use islower ( ),isupper ( ), tolower ( ), toupper( )

```
#include<iostream.h>
#include<ctype.h>
void main( )
{
        char ch;
        cout<<"Input a character";
        cin>>ch;
        if( isupper(ch) = = 1)    // checks if character is in upper case converts the character
                                  //  to lower case

        {
        tolower(ch);
          cout<<ch;
        }
        else if(islower(ch) = = 1)  // checks if character is in lower case converts the
                                    // character to uppercase

        {
          toupper(ch);
          cout<<ch;
        }
}
```

fabs ( ), pow ( ), sqrt ( ), sin ( ), cos ( ), abs ( ) :

Program 3.4

```
#include <iostream.h>
#include <math.h>
#define PI 3.14159265 // macro definition PI will always hold 3.14159265

void  main ()
{

cout<<"The absolute value of 3.1416 is : "<<fabs (3.1416) ;
                                                    // abs( ) also acts similarly but only on int data
cout<<"The absolute value of -10.6 is "<< fabs (-10.6) ;
cout<<"7.0 ^ 3 = " <<pow (7.0,3);
cout<<"4.73 ^ 12 = " <<pow (4.73,12);
cout<<"32.01 ^ 1.54 = "<<pow (32.01,1.54);

double param, result;
param = 1024.0;
result = sqrt (param);
cout<<"sqrt() = "<<result ;
result = sin (param*PI/180);   // in similar way cos( ) , tan() will be called.
cout<<"The sine of " <<param<<" degrees is : "<< result
}
```

randomize ( ), random ( ) :
The above functions belongs to header file stdlib.h . Let us observe the use of these functions :

randomize( ) : This function provides the seed value and an algorithm to help random( ) function in generating random numbers. The seed value may be taken from current system's time.

random(<int>) : This function accepts an integer parameter say x and then generates a random value between 0 to x-1
for example : random(7) will generate numbers between 0 to 6.
To generate random numbers between a lower and upper limit we can use following formula
random(U – L +1) + L
where U and L are the Upper limit and Lower limit values between which we want to find out random values.
For example : If we want to find random numbers between 10 to 100 then we have to write code as :
random(100 -10 +1) + 10 ; // generates random number between 10 to 100
User-defined function :- The functions which are defined by user for a specific purpose is known as user-defined function. For using a user-defined function it is required, first define it and then using.
Function Prototype :- Each user define function needs to be declared before its usage in the program. This declaration is called as function prototype or function declaration. Function prototype is a declaration statement in the program and is of the following form :
Return_type function_name(List of formal parameters) ;
Declaration of user-defined Function: In C++ , a function must be defined, the general form of a function definition is :

```
Return_type  function_name(List of formal parameters)
    {
       Body of the function
     }
```

Where Return_type is the data type of value return by the function. If the function does not return any value then void keyword is used as return_type.
List of formal parameters is a list of arguments to be passed to the function. Arguments have data type followed by identifier. Commas are used to separate different arguments in this list. A function may be without any parameters, in which case , the parameter list is empty.
statements is the function's body. It is a block of statements surrounded by braces { }.
Function_name is the identifier by which it will be possible to call the function.

e.g.,

```
int addition (int a, int b)
        {
                int r ;
                r=a+b ;
                return (r) ;
        }
```

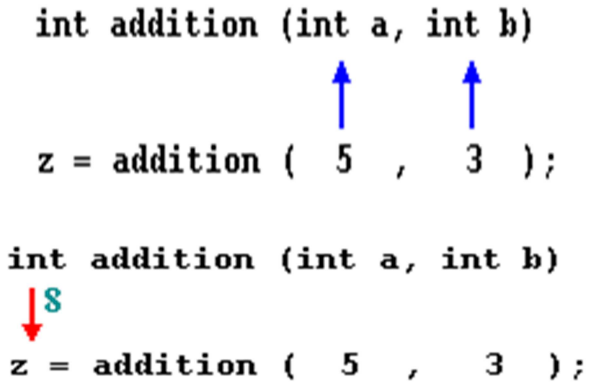
Calling a Function:- When a function is called then a list of actual parameters is supplied that should match with formal parameter list in number, type and order of arguments.
Syntax for calling a function is:

function_name (list of actual parameters);

e.g.,

```
#include <iostream>
 int addition (int a, int b)
{  int r;
   r=a+b;
return (r); }
void main ( )
{  int z ;
   z = addition (5,3);
cout<< "The result is " << z;
  }
```

The result is 8



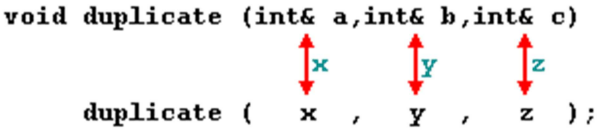
Call by Value (Passing by value) :- The call by value method of passing arguments to a function copies the value of actual parameters into the formal parameters , that is, the function creates its own copy of argument values and then use them, hence any chance made in the parameters in function will not reflect on actual parameters . The above given program is an example of call by value.

Call by Reference (Passing by Reference) :- The call by reference method uses a different mechanism. In place of passing value to the function being called , a reference to the original variable is passed . This means that in call by reference method, the called function does not create its own copy of original values , rather, its refers to the original values only by different names i.e., reference . thus the called function works the original data and any changes are reflected to the original values.

// passing parameters by reference

```
#include <iostream.h>
void duplicate (int& a, int& b, int& c)
{
        a*=2;
        b*=2;
        c*=2;
}

void main ()
        {
                int x=1, y=3, z=7;
                duplicate (x, y, z);
                cout <<"x="<< x <<", y="<< y <<", z="<< z;
        }
```



output :x=2, y=6, z=14

The ampersand (&) (address of) is specifies that their corresponding arguments are to be passed *by reference* instead of *by value*.

Constant Arguments:-In C++ the value of constant argument cannot be changed by the function. To make an argument constant to a function , we can use the keyword const as shown below:

int myFunction(const int x , const int b);

The qualifier const tell the compiler that the function should not modify the argument. The compiler will generate an error when this condition is violated.

Default Arguments :- C++ allows us to assign default value(s) to a function's parameter(s) which is useful in case a matching argument is not passed in the function call statement. The default values are specified at the time of function definition. e.g.,

float interest (float principal, int time, float rate = 0.70f)

Here if we call this function as:

si_int= interest(5600,4);

then rate =0.7 will be used in function.

Formal Parameters:- The parameters that appear in function definition are formal parameters.

Actual Parameters :- The parameters that appears in a function call statement are actual parameters.

Functions with no return type (The use of void):- If you remember the syntax of a function declaration:

Return_type function_name(List of formal parameters)

you will see that the declaration begins with a type, that is the type of the function itself (i.e., the data type of value that will be returned by the function with the return statement). But what if we want to return no value?

Imagine that we want to make a function just to show a message on the screen. We do not need it to return any value. In this case we should use the void type specifier for the function. This is a special specifier that indicates absence of type.

The return Statement :- The execution of return statement, it immediately exit from the function and control passes back to the calling function (or, in case of the main( ), transfer control back to the operating system). The return statement also returns a value to the calling function. The syntax of return statement is:

return (value);

Scope of Identifier :- The part of program in which an identifier can be accessed is known as scope of that identifier. There are four kinds of scopes in C++

(i) Local Scope :- An identifier declare in a block ({ }) is local to that block and can be used only in it.

(ii) Function Scope :- The identifier declare in the outermost block of a function have function scope.

(iii) File Scope (Global Scope) :- An identifier has file scope or global scope if it is declared outside all blocks i.e., it can be used in all blocks and functions.

(iv) Class Scope :- A name of the class member has class scope and is local to its class.

Lifetime :- The time interval for which a particular identifier or data value lives in the memory is called Lifetime of the identifier or data value.

ARRAYS :-Array is a group of same data types variables which are referred by a common name.

An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier.

Need for Arrays:- Suppose we have to store roll number & marks of 50 students in a program , then in all we need 100 variable each having a different name. Now remembering and managing these 100 variables is not easy task and it will make the program a complex and not-understandable program.

But if we declare two array each having fifty elements one for roll numbers and another for marks. Now, we have only two names to remember. Elements of these array will be referred to as Arrayname[n], where n is the element number in in the array . Therefore arrays are very much useful in a case where quit many elements of the same data types need to be stored and processed. The element number s in parenthesis are called subscripts or index. The subscript, or index of an element designates its position in the array's ordering.

Types of Array :-Arrays are of two types:

1. One-dimensional Arrays
2. Multi-dimensional Arrays (But we have to discussing only Two-dimensional arrays in our syllabus)

One –dimensional Arrays :- The simplest form of an array is one - dimensional. The array itself is given a name and its elements are referred to by their subscripts. In C++ , an array must be defined before it can be used to store information. The general form of an array declaration is:

Data_type Array_name[size];

For example the following statement declares an array marks of int data type , which have 50 elements marks[0] to marks[49].

int marks[50] ;

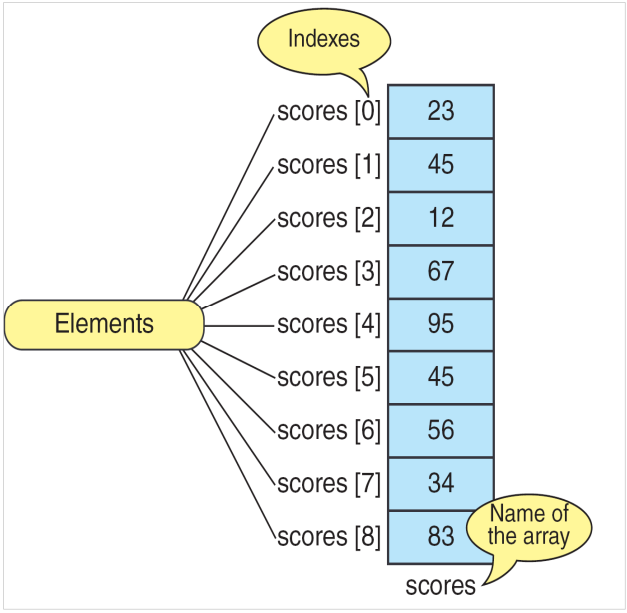
Referencing Array Elements:- We can reference array elements by using the array's name & subscript (or index). The first element has a subscript of 0. The last element in an array of length *n* has a subscript of *n-1*.

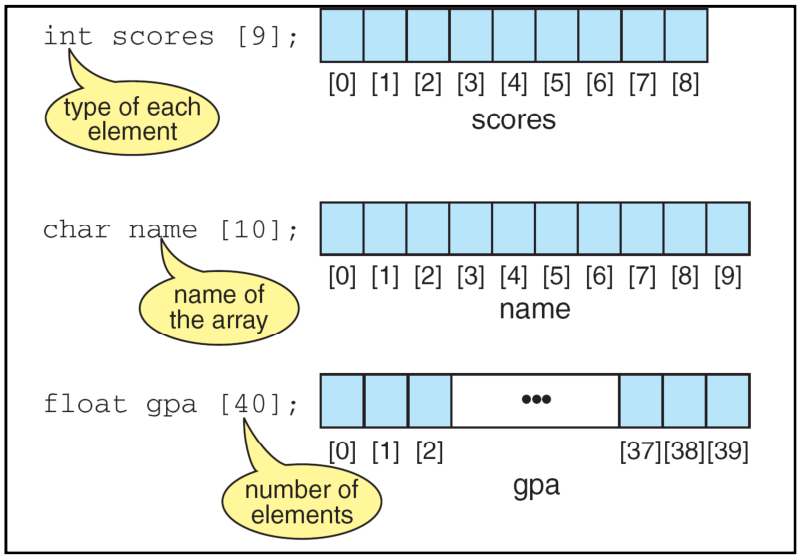
If we declare an array:

int scores[9];

then its 10 element will be referred as:

scores[0], scores[1], ……., scores[8]

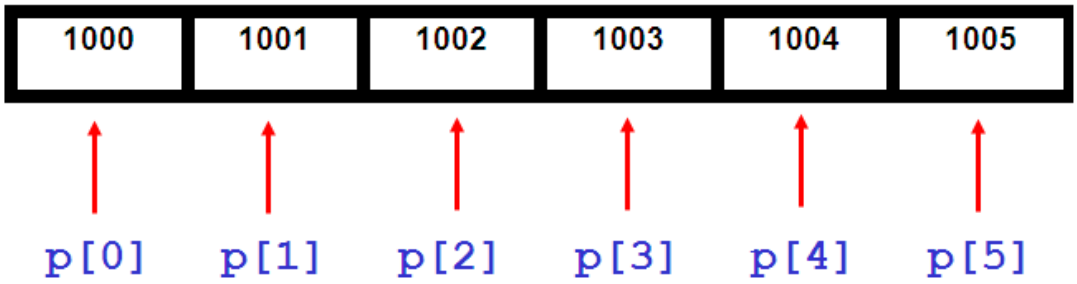




Memory Representation of Single Dimensional Arrays:-Elements of single dimensional array are stored in contiguous memory location in their index order. For example , an array p of type char with 6 elements declared as:

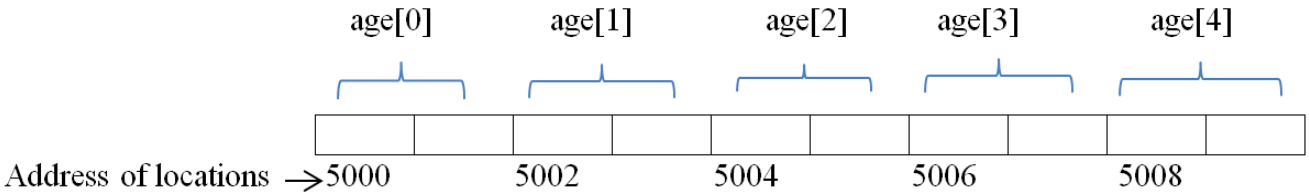
char p[6];

will be represented in memory as shown below:

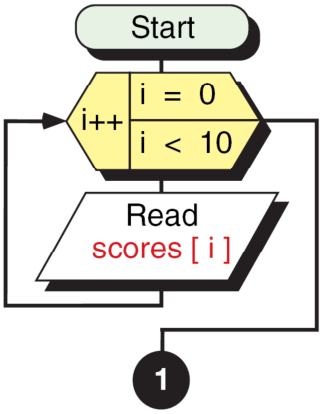


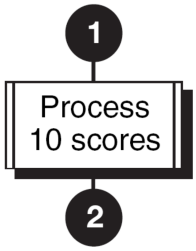
If we declare an int array age with 5 elements as:

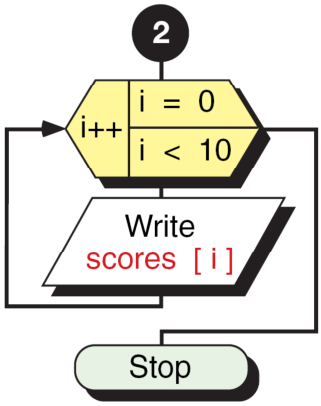
int age[5];



Accepting Data in Array from User (Inputting Array elements) and Printing Array elements:- Since we can refer to individual array elements using numbered indexes, it is very common for programmers to use **for** (or any) loops when processing arrays.







General form of for loop for Reading elements of array (1-D)	Generally processing part may be include with in the loop of reading or printing, otherwise a same type separate loop may be used for processing	General form of for loop for printing elements of array (1-D)

for (int i=0; i< size; i++)
{
cout<<"Enter Array Element "<<i+1;
cin>>Array_Name[i];
}
		for (int i=0; i< size; i++)
		{
		cout<<Array_Name[i]<< ", ";
		}

Program : A C++ program to read an array of 10 elements and print the sum of all elements of array:

```
#include <iostream.h>
void  main()
{
int a[10], sum=0;
for(int j=0; j<10; j++)                    //loop for reading
{
cout<< "Enter Element "<< j+1<<":";
cin>> a[j]
sum=sum+a[j]`;
}
cout<<"Sum : "<<sum;
}
```

Program: Here's another example of an array this program , invites the user to enter a series of six values representing sales for each day of the week (excluding Sunday), and then calculates the average sales.

```
#include <iostream.h>
void  main()
{
const int SIZE = 6;                        //size of array
double sales[SIZE];                        //array of 6 elements
double total = 0;
cout<< "Enter sales for 6 days\n";
for(int j=0; j<SIZE; j++)                  //put figures in array
cin>> sales[j];
for(j=0; j<SIZE; j++)                      //read figures from array
total += sales[j];                         //to find total
double average = total / SIZE;             // find average
cout<< "Average = " << average << endl;
}
```

Output:-

```
Enter sales for 6 days
352.64
867.70
781.32
867.35
746.21
189.45
Average = 634.11
```

Linear Search:- In linear search, each element of the array is compared with the given item to be searched for , one by one either the desired element is found or the end of the array is reached.

```
#include<iostream.h>
void main( )
{
int size, i, n, c=0;
cout<<"Enter the size of array:"
cin>>size;
int a[size];
cout<<"Enter the elements of Array: ";
```

```
for(i-0; i<size; i++)
{
cin>>a[i];
}
cout<<" Enter the number to be search : ";
cin>>n ;
for (i=0; i<size; i++)
{   if (a[i]= =n)
    {
       c=1;
break;
     }
}
if (c= =0)
     cout<<" The number is not in the array.";
else
cout<<" The Number is found and location of element is: "<<i+1:
}
```

Initializing arrays.

When we declare an array, we have the possibility to assign initial values to each one of its elements by enclosing the values in braces { }. Syntax for initialization of one-dimensional array is:

Data_type Array_name[sise] = { value list } ;

For example:

int num [5] = { 16, 2, 77, 40, 12071 };

This declaration would have created an array like this:

num

0	1	2	3	4
16	2	77	40	12071

The amount of values between braces { } must not be larger than the number of elements that we declare for the array between square brackets [ ].

C++ allows us to skip the size of the array in an array initialization statement (by leaving the square brackets empty [ ]). In this case, the compiler will assume a size for the array equal to the number of values given between braces { }:

int num [ ] = { 16, 2, 77, 40, 12071 };

After this declaration, the size of array num will be 5, since we have provided 5 initialization values.

Program : Program to find the maximum and minimum of an array :

```
#include<iostream.h>
void  main( )
 {
int arr[ ] = {10, 6 , -9 , 17 , 34 , 20 , 34 ,-2 ,92 ,22 };
int max = min = arr[0];
for(int i = 0 ; i<10 ; i++)
  {
                if(arr[i] < min)
                    min= arr[i];
```

```
                    if(arr[i] > max)
                      max = arr[i];
    }
  cout<<"The minimum of all is : " << min<<"and the maximum is : " <<max;
}
```

String as an array:- C++ does not have a string data type rather it implements string as single dimensional character array. A string is defined as a character array terminated by a null character '\0'. Hence to declare an array strg that holds a 10 character string, you would write :

```
char  strg[11];
```

Program : Program to count total number of vowels present in a string :

```
#include<iostream.h>
    #include<stdio.h>
    void  main( )
    {
      char string[35]; int count = 0;
      cout<<"Input a string";
      gets(string);                    // library function in stdio.h to input a string
      for(int i = 0 ; string[i] != '\0' ; i++)
      {
 if( string[i] = = 'a' || string[i] = = 'e' || string[i] = = 'o' || string[i] = = 'u' || string[i] = = 'i' ||
string[i] = = 'A' ||  string[i] = = 'E' || string[i] =  = 'O' || string[i] = = 'U' ||  string[i] == 'I' )
        {
            count++;
        }
      }
    }
```

Two dimensional array : A two dimensional array is a continuous memory location holding similar type of data arranged in row and column format (like a matrix structure).

Declaration of 2-D array:- The general syntax used for 2-Darray declaration is:

Data_type Array_name [R][C] ;

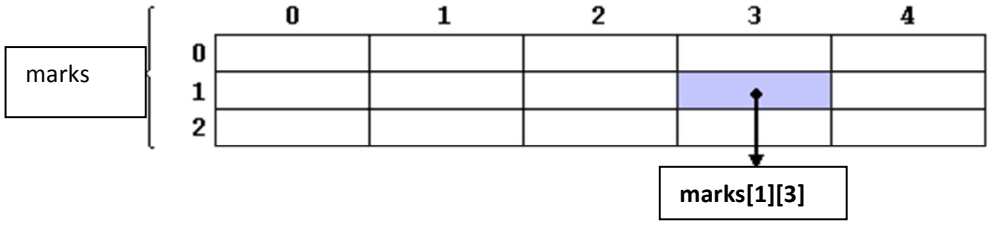
Where R represent number of rows and C represent number of columns in array.

For example,

int marks [3][5];

and, for example, the way to reference the second element vertically and fourth horizontally in an expression would be:

marks[1][3]



Initialization of 2-D Array:- Two dimensional arrays are also initialize in the same way as single dimensional array . e. g.,

int cube[5][2] = {1,1, 2,8,3,27,4,64,4,125 };

will initialize cube[0][0]=1, cube[0][1]=1, cube[1][0]=2, cube[1][1]=8 , cube[2][0]=3 and so on.

2-D array can also be initialized in unsized manner. However the first index can be skiped , the second index must be given. e.g.,

```
int cube[ ][2] = {1,1, 2,8,3,27,4,64,4,125 };
```

is also valid.

Accepting Data in 2-D Array from User (Inputting 2-D Array elements) and Printing 2-D Array elements:- Since We must reference both, a row number and a column number to input or output values in a two-dimensional array. So, we use a nested loop (generally nested for loop) , when processing 2-D arrays.

General form of for loop for Reading elements of 2-D array	Generally processing part may be include within the loop of reading or printing, otherwise a same type separate nested loop may be used for processing	General form of for loop for printing elements of 2-D array

for (int i=0; i< R; i++)
{ cout<<"Enter Row "<<i+1;
for (int j=0; j<C ; j++)
cin>>Array_Name[i][j];
}
		for (int i=0; i< R; i++)
		{ for (int j=0; j<C ; j++)
		cout<<Array_Name[i][j]
		<<'\t';
		cout<<'\n';
		}

Where R represent number of rows and C represent number of columns in array.

Program : Program to subtract two 3x3 matrices and then print the resultant matrix.

```
#include <iostream.h>
#include<conio.h>
main()
{
int m, n, c, d, first[10][10], second[10][10], sub[10][10];
cout<<"Enter the elements of first matrix\n";

for ( i = 0 ; j < 3 ; i++ )
for ( j = 0 ; j <3 ; j++ )
cin>>first[i][j];

cout<<"Enter the elements of second matrix\n";

for ( i = 0 ; j < 3 ; i++ )
for ( j = 0 ; j <3 ; j++ )
cin>>second[i][j];
cout<<"Subtraction of entered matrices:-\n";
for ( i = 0 ; j < 3 ; i++ )
  {
for ( j = 0 ; j <3 ; j++ )
        {
                sub[i][j] = first[i][j] - second[i][j];
                cout<<sub[i][j]<<'\t';
        }
  cout<<'\n' ;
}
getch();
}
```

Array of Strings:- An array of string is a two-dimensional character array. The size of first index determines the number of strings and size of second index determines maximum length of each string. The following statement declares an array of 10 strings , each of which can hold maximum 50 valid characters.

```
char string[10][51] ;
```

Notice that the second index has been given 51, because 1 extra size is given for store null character '\0'.

User Defined Data Types:-

The C++ language allows you to create and use data types other than the fundamental data types. These types are called user-defined data types.

Structures:- In C++, a structure is a collection of variables that are referenced by a single name. The variable can be of different data types. They provide a convenient means of keeping related information together.

Defining Structure :-

A Structure is defined by using the Keyword struct. The General Syntax for defining a Structure is :

Syntax :

```
struct< Name of Structure >
{
<datatype>< data-member 1>;
<datatype>< data-member 2>;
<datatype>< data-member 3>;
        …
        …
<datatype>< data-member n>;
} ;
```

Example :

A Proper example following the previous syntax could be :

```
struct  WORKER                  Structure name
{
char name[45];
char gender ;                   data-member
int age ;
float rate ;

} ;
```

Declaring Structure Variable :-

This is similar to variable declaration. We can declare the variable of structure type using two different ways either with structure definition or after structure definition.

The following syntax shows the declaration of structure type variables with structure definition:

```
struct< Name of Structure >
{
<datatype>< data-member 1>;
```

```
<datatype>< data-member 2>;
<datatype>< data-member 3>;
                …
                …
<datatype>< data-member n>;
} var1, var2,….., varn ;
```

We can declare the structure type variables separately (after defining of structure) using following syntax:

```
Structure_name  var1, var2, …… ….., var_n;
```

Accessing Structure Elements :- To access structure element , dot operator is used. It is denoted by (.). The general form of accessing structure element is :

```
Structure_Variable_Name.element_name
```

Initializing of Structure elements:- You can initialize members of a structure in two ways. You can initialize members when you declare a structure or you can initialize a structure with in the body of program. For Example:

First Method:-

```
#include <iostream.h>
#include <conio.h>
void main()
{
        // Declaring structure at here
        struct Employee
        {
                int empcode;
                float empsalary;
        } ;
        Employee emp1 = {100, 8980.00} ;          // emp1 is the structure variable ,
                                                  // which is also initialize with declaration
        clrscr();
        int i;          // declares a temporary variable for print a line
                        // Printing the structure variable emp1 information to the screen
        cout<< "Here is the employee information : \n";
        for (i = 1; i <= 32; i++)
                cout<< "=";
        cout<< "\nEmployee code : " << emp1.empcode;
        cout<< "\nEmployee salary : " << emp1.empsalary;
}
```

Second Method:-

```
#include <iostream.h>
#include <conio.h>
void main()
{
        // Declaring structure  here
        struct Employee
        {
                int empcode;
                float empsalary;
```

```
        } emp1;          // emp1 is the structure variable
        clrscr();
        int i;           // declares a temporary variable for print a line
        // Initialize members here
        emp1.empcode = 100;
        emp1.empsalary = 8980.00;
        // Printing the structure variable emp1 information to the screen
        cout<< "Here is the employee information : \n";
        for (i = 1; i <= 32; i++)
                cout<< "=";
        cout<< "\nEmployee code : " << emp1.empcode;
        cout<< "\nEmployee salary : " << emp1.empsalary;
}
```

Structure variable assignments

We know that every variable in C++ can be assigned any other variable of same data type i,e :

if int a = 7 ; b = 3;

we can write :

a = b ; // assigning the value of b to variable a

Similar is the case with Structure variables also , i,e :

if WORKER w1 = {“Ramu” , ‘M’ , 17 , 100.00};

WORKER w2;

we can write :

w2 = w1 ; // assigning the corresponding individual // data member values of w1 to Worker w2;

or

WORKER w2 = w1;

Note : Both structure variables must be of same type (i,e WORKER in this case)

There is a member wise copying of member-wise copying from one structure variable into other variable when we are using assignment operator between them.

So, it is concluded that :

```
Writing : strcpy (w1. name,w2.name) ;
w1.gender = w2.gender ;
w1.age = w2.age ;
w1.rate = w2.rate ;
```

is same as :

```
w1 = w2 ;
```

Array of Structure :- We can define an array of structure by using following syntax :

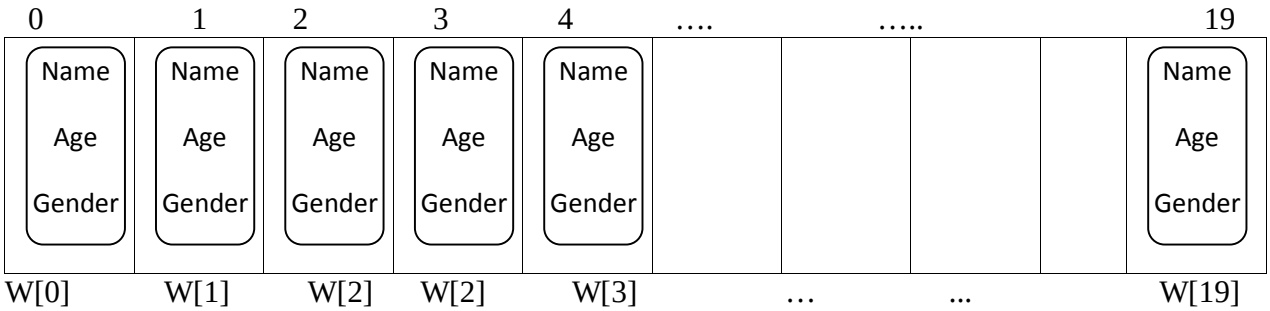
<structure_name><array_name>[size];

where : structure_name is the name of the structure which you have created.
array_name is any valid identifier
size is a positive integer constant.

Example : to create array of 20 Workers we can have :

Worker W[20];

The above structure could be visualized as :



Each of the elements of the array is itself a structure hence each of them have all the four components.

Function with Structure variable:-

We can also pass a structure variable as function parameter / arguments. The benefit is that the structure carries a bundled information to the structure. The prototype of such a function would be

<return_type> function_name(<structure_name><var> , … , …);

Let us understand the concept with following function ,which increases the wage of a female worker by passed percent

```
void increaseWage(   Worker  & w ,  float incr )
{
        if( w.gender == 'F' || w.gender == 'f' )
        {
                w.wage + = w.wage* (incr /100) ;
        }
}
```

Look at the highlighted parameter, we have passed formal structure variable as reference, so that the increment is reflected back in actual parameter.

Similarly, if we don't want to pass our structure variable as reference we can do so , but then we have to return the modified structure variable back. This can be achieved in above function, if we take return type as structure name. Let us modify the function so that it returns a structure :

```
Worker increaseWage( Worker w , float incr)
{
        if( w.gender == 'F' || w.gender == 'f' )
        {
                w.wage + = w.wage* (incr /100) ;
        }
 return w ;
}
```

Nested structure :- A Structure within another structure is known as nested structure. A structure containing an another structure as valid element is known as nested structure.

e.g.

```
struct address
{
int houseno;
char city[20];
char area[20];
long int pincode;
} ;

struct employee
{
int empno;
char name[30];
char design[20];
char department[20];
address ad; // nested structure
}
```

Declaration:

employee e;

Input /Output :

```
cin>>e.ad.houseno;          // members are accessed using dot(.) operator.
cout<<e.ad.houseno;
```

typedef :- The typedef keyword allows to create a new names for data types. the syntax is:

typedef existing_data_type new_name ;

e.g.,

```
typedef int num;
```

#**define Preprocessor Directive :-**The # define directive create symbolic constant, constants that are represent as symbols and macros (operation define as symbols). The syntax is:

```
#define identifier replacement_text ;
```

When this line appears in a file, all subsequent occurances of identifier in that file will be replaced by replacement_text automatically before the program is compiled. e.g.,

Program: Program to calculate area of a circle by using #define preprocessor directive.

```
#include <iostream.h>
#define PI 3.14159
#define CIRCLE_AREA(x) (PI * (x) * (x))
void main()
{
    float area;
    int r;
    cout<< "Enter the radius : ";
    cin>> r;
    area = CIRCLE_AREA(r);
    cout<< "Area is : " << area;
}
```

Sample Paper Class – I
Subject – Computer Science

Time: 3Hours **Maximum Marks: 70**

Note. (i) All questions are compulsory.

1 a) What are the different functions of operating system? 2
b) How the information can be used as a data explain ? 2
c)What do you mean by unary 2
d) What are the different parts of CPU? Explain every part in brief. 2
e) Define System Software and what are its two main types? Give examples. 2
f) What is Booting? 1
g) Which of the following are hardware and software? 1
(i) Capacitor (ii) Internet Explorer (iii) Hard disk (iv) UNIX

2. Explain the following term: (Give answer any six) 6

i) Variable
ii) Token
iii) Array
iv) Debugging
v) Comment
vi) Keyword

3 a) What is the difference b/w **"while"**& "do **while**" loop? 2
b) What are data types? What are all predefined data types in c++? 2
c) What will be the size of following constants? 1
'\v',"\v",
d) Write the corresponding C++ expressions for the following mathematical expressions: 1
i) $\sqrt{(a^2+b^2)}$ (ii) $(a+b)/(p+q)^2$
e) Evaluate the following, where p, q are integers and r, f are floating point numbers.
The value of p=8, q=4 and r=2.5
(i) f = p * q + p/q
(ii) r = p+q + p % q 2

4 a) What is the output of the following? 2
i) # include<iostream.h>

```
void main ( )
        {
        int i=0;

                cout<<i++<<" "<<i++<<" "<<i++<<endl;
                 cout<<++i<<" "<<++i<<" "<<++i<<endl
        }
```

ii) # include<iostream.h>

```
        void main( )
                {
                a=3;
                a=a+1;
                if (a>5)
                        cout<<a;
                else
                        cout<<(a+5);
                }
```

2
iii) What will be the output of the following program segment? 3
If input is as: (a) g (b) b (c) e (d) p

```
cin>>ch;
switch (ch)
          { case ‘g’: cout<<”Good”;
            case ‘b’: cout<<”Bad”;
break;
            case ‘e’: cout<<” excellent ”;
break;
            default: cout<<” wrong choice”;
          }
```

iv) Determine the output: 2

```
for(i=20;i<=100;i+=10)
         {
          j=i/2;
          cout<<j<<””;
         }
```

v) What output will be the following code fragment produce?

```
void main( )
                    {
                    int val, res, n=1000;
                    cin>>val;
          res = n+val >1750 ? 400:200;
          cout<<res;
                     }
```

(i) if val=2000 (ii) if val=1000(iii) if val=500 3

5 a) Find the error from the following code segment and rewrite the corrected code underlining the correction made. 2

```
  # include(iostream.h)
void main ( )
int X,Y;
cin>>>X;
for(Y=0,Y<10, Y++)
if X= =Y
cout<<Y+X;
else
cout>>Y;      }
```

b) Convert the following code segment into switch case construct. 3

```
     int ch;
     cin>>ch;
     If(ch = = 1)
     {      cout<<“ Laptop”;
     }
else If(ch = = 2)
            {
            cout<<“Desktop ”;
            }    else if(ch= = 3)
                {
cout<<“Notebook”;
               } else
                        {
cout<<“Invalid Choice”;
                           }
            }
     }
```

c) Convert the following code segment into do-while loop. 3

```
#include<iostream.h>
void main()
    {  int i;
for(i=1;i<=20;++i)
        cout<<"\n"<<i;
    }
```

d) Given the following code fragment

```
int ch=5;
    cout << ++ch<< "\n"<<ch<<"\n";
```

i) What output does the above code fragment produce?
ii) What is the effect of replacing ++ ch with ch+1? 2

6 a) Which header files are required for the following?
(i) frexp()((ii) sqrt( ) (iii) rand( ) (iv) isupper() 2

b) Evaluate: 4
i) $(12)_{10} = (X)_2$
ii) $(347)_8 = (X)_{10}$
iii) $(896)_{16} = (X)_8$
iv) $(100)_{10} = (X)_2$

7 a) Write a C++ program to check a year for leap year or not. 2
b) Write a C++ program to check a number for Armstrong or not. 4
c) Write a C++ program to calculate the factorial of any given number 4
d) Write a C++ program to print the Fibonacci series 4
e) Write a C++ program to print table a given number. 2

Answer key

Q.No.1

a. Major OS functions are listed below

1. Process Management
2. Storage Management
3. Device Management
4. Command Interpreter

b. The processed information can be used as a data again to produce a next level information. For example- total no. of students school wise can give the information that how students are there in one region again this information as a data ca be used to calculate that how many students are studying in KVS

c. unary operators are the operators, require one operand

d. ALU(Arithmetic logic unit), CU(control unit), MU(memory unit)

e. System software are the software that govern the operation of computer system and make the hardware run. These software can be classified into two categories. Operating System & Language Processor

f. Booting is a process through which operating system makes the computer system ready to perform user's task

g. Hardware- I&III, Software- II&IV

Q.No.2

i. variable is a name given to the memory location, whose value can be changed during run time.

ii. The smallest individual unit in a program is known as a token

iii. Array is a combination of similar data values. It is used to store more than one value under same name

iv. debugging is a way to correct the errors in the program during compilation

v. comments are non executable statements, used to give the information about the code for future use.

vi. Keywords are the reserved words, Which are reserve in compiler for a specific purpose. The keyword can not be taken as a name of identifier.

Q.No.3

a. While loop is entry control loop i.e. while loop first will test the condition and if condition is true then only the body of the loop will be executed. While do-while loop is exit control loop and even the condition is not true at least one time the body of the loop will be executed.

b. data types are means to identify the types of data and associated operation of handling it. The fundamental data types are- char, int, float , double and void .

c. one byte

d. i. sqrt(a*a+b*b) & ii. ((a+b)/((p+q)*(p+q))

e. students do yourself

Q.No.4

a. i. 0 1 2
4 5 6 , ii. 9 , iii. For g- good & bad/ for b – bad / for e – excellent / for – p wrong choice

iv. 10,15,20,25,30,35,40,45,50 v. 400, 400, 200

Q.No.5

a. Errors – if x==y (correct- if(x==y)) & cout>>y(correct cout<<y)

b.

```
int ch; cin>>ch;
      switch(ch)
           {
              case 1 :  cout<<" Laptop"; break;
              case 2:  cout<<"Desktop "; break;
              case 3:  cout<<"Notebook";break;
          default : cout<<"Invalid Choice";
          }
```

c. #include<iostream.h>
void main()
{ int i;
i=1
do
{ cout<<”\n”<<i;
++i
}while (i<=20);
}
d. In both condition output will be 6 5

Q.No.6

a. math.h , math.h , stdlib.h , ctype.h
b. 1100, (232) , (4226), (1100100)

Q.No.7

a.

```
#include<iostream.h>
#include<conio.h>

void main()
{
clrscr();
int year;
cout<<"Enter Year(ex:1900):";
cin>>year;
if(year%100==0)
{
if(year%400==0)
cout<<"\nLeap Year";
}
else
if(year%4==0)
cout<<"\nLeap Year";
else
cout<<"\nNot a Leap Year";
getch();
}
```

b.

```
#include<iostream.h>
#include<conio.h>
void main()
{
intNumber,Temp,b=0;
cout<<endl<<"Enter any number to check";
cin>>Number;
Temp=Number;
intP;
while(Temp>0)
{
 P=Temp%10;
 b=b P*P*P;
 Temp=Temp/10;
}
if(b==Number)
```

```
{
 cout<<endl<<"Armstrong no";
}
else
{
 cout<<"Not an armstrong no";
}
getch();
}
```

c.

```
#include <iostream.h>
int factorial(int);

void main(void) {
int number;

cout << "Please enter a positive integer: ";
cin >> number;
if (number < 1)
cout << "That is not a positive integer.\n";
else
cout << number << " factorial is: " << factorial(number) << endl;
}

int factorial(int number) {

if(number <= 1) return 1;
else
return number * factorial(number - 1);
}
```

d.

```
#include<iostream.h>
#include<conio.h>
int main()
{
        clrscr();
        unsigned long first,second,third,n;
        int i;
        first=0;
        second=1;
        cout<<"how many elements(>5)? \n";
        cin>>n;
        cout<<"fibonacci series\n";
        cout<<first<<" "<<second;
        for(i=2;i<n;i++)
        {
                third=first+second;
                cout<<" "<<third;
                first=second;
                Second=third;
        }
        return 0;
        getch();}
```

e.

```
#include<iostream.h>
#include<stdio.h>
void main()
{
int r,m,i,n;
cout<<"Enter the number to generate its table";
cin>>n;
cout<<"Enter the number(table upto)";
cin>>m;
i=1;
while(i<=m)
{
r=n*i;
cout<<N<<"*"<<I<<"="<<R<<endl;
}

}
```

Sample Paper - II
Subject – Computer Science

Max Marks 70 Duration 3 hrs

Note:- All questions are compulsory

Q1)
a) What is significance of My Computer ? 2
b) Explain different types of operating systems . 1
c) What is an Operating System? Explain its any two functions 2
d) How is a compiler different from interpreter? 2
e) Convert
(i) $(10.10)_{10} = (?)_2$
(ii) $(101011.1110)_2 = (?)_{10}$ 2

Q2)
a) What do you mean by run time error? 1
b) what is the deference between if and if – else statement 2
c) Mention and explain briefly any three characteristics of a good program 2
d) How can you give a single line and multiline comments in C++ explain with suitable examples 2
e) what do you mean by header files? What is the difference between #include <iostream.h> and #include"iostream.h" 2

Q3)
a) Classify the following variable names of c++ into valid and invalid category 2
(i) 1no (ii) num 1 (iii) num (iv) num1num (v) num+1 (vi) num.1
b) Give output of following code. 3

```
#include<iostream.h>
int m=5; void check();
void main( )
{  int m=20;
        {
        int m=10*::m;
        cout<<"m="<<m<<"::m="<<::m<<endl;
        } check();
        cout<<"m="<<m<<"::m="<<::m<<endl;
        check(); cout<<"::m="<<::m<<"m="<<m<<endl;
        }
        void check()
        {  ++m;

    }
```

c) What will be result of following statements if a=5 , b=5 initially 2
(i) ++a<=5 (ii) b++<=5

d)Name the header file(s) that shall be needed for successful compilation of the following C++ code. 2

```
        void main( )
        {
```

```
char name[40];
strcpy(name,"India");
puts(name); }
```

e) Explain conditional operator (?) with example in c++ . 2
f) What is difference between '/' and '%' operators in c++ ? explain with a example 2

Q4)

a) What do you mean by function prototype in C++ 2
b) Explain Break and Continue statement in C++ with example. 2
c) Find syntax error(s) if any in following program (Assume all header files are present) 2

```
 main<>
{  int c;
  switch( c );
case 1.5:  { cout<<" India is great\n";
             }  break;
'case' 2:  {  cout<<" hello\n";
             }  break;
}  // end of main
} // end of switch
```

d) How will you declare and define 1
i)Array named mark with 10 integer values
ii)array named avg with 8 float values

e) Convert following while loop to for loop 2

```
int   x=0;
while(x<=100)
{  cout<<" the value of x is \n"<<x;
   cout<<"done \n";
  x+=2;
}
```

f) Define token ? 1

g) What is the difference between call by value and call by reference explain with suitable example 2

h)Find the output of the following program; **3**

```
#include<iostream.h>
#include<ctype.h>
void main( )
{ char Text[ ] = "Comp@uter!";
for(int I=0; Text[I]!='\0';I++)
{ if(!isalpha(Text[I]))
Text[I]='*';
else if(isupper(Text[I]))
Text[I]=Text[I]+1;
else
Text[I] = Text[I+1]; }
cout<<Text; }
```

i) What are differences between for and do- while loop ? explain with example 2

j) How can you define Global Variable and Local Variable? Also, give a suitable C++ code to illustrate both. 2

Q5 : a) Write a program to print the left and right diagonal element of an NXN matrix 4

b) b. Write a program to find the factorial of a number recursive function. 4

c) Write a program to find the total number of characters, lines and words in a paragraph of text. 4

d) Write a program to sort an array on N numbers in ascending order. Avoid duplication of elements. 4

e) Write a program to find the roots of a quadratic equation. 4

Answer Key

Q.No.1

a. My Computer is used to viewing the contents of a single folder or drive. Also the things we have on our computer – Your Programs, Documents and data files, for example- are all accessible from form one place called My Computer.

b. Types of operating system -

i. Single User ii. Multiuser iii. Time Sharing iv. Real Time v. Multiprocessing vi. Interactive

c. An operating system is a program which acts as an interface between a user and the Hardware.

Functions- i. Process Management and Storage(Memory) Management.

Note. Students has to explain both in brief

d. An interpreter converts source program(HLL) into object program(LLL) line by line whereas compiler converts source program into object program in one go and once the program is error free it can be executed later .

Q.No.2

a. An error in logic or arithmetic that must be detected at run time is called run time error. For example any number is divided by zero (c=a/0) is run time error

b. if statement can perform only true condition and there is no option for false condition while if-else statement can perform both true as well as false condition . Example,

if(a<b) cout<<" a is greater than b" – in this code for false condition no statement is given

if(a<b) cout<<" a is greater than b"

else

cout<<" b is greater than a" – here for both option either true or false output will come

c. Given on page No. 19

d. Comments in c++ can be given in two way

Single line comment – which can be given by using the symbol - //

Example- // Program to find the sum of two numbers

And Multi line comment – which can be given by using the symbol /*…….*/

Example -- /* this is the way

To give multiline Comment */

e. Header files are the files in which some pre – programed program will be stored by the developer of the language which will help the user of to develop their program. If the header file is written in angular bracket(<>) compiler will search it in C++ library only while in case of " " the header file will be searched throughout the database.

Q.No.3

a. i , ii ,v , vi – are invalid & iii, iv are valid declaration

b. m=5 ::m=5

m=20 ::m=6

m=7 ::m=20

c. True and false

d. String.h and stdio.h

e. The conditional operator (**? :**) is a ternary operator (it takes three operands). The conditional operator works as follows:

- The first operand is implicitly converted to **bool**. It is evaluated and all side effects are completed before continuing.
- If the first operand evaluates to **true** (1), the second operand is evaluated.
- If the first operand evaluates to **false** (0), the third operand is evaluated.

The result of the conditional operator is the result of whichever operand is evaluated — the second or the third. Only one of the last two operands is evaluated in a conditional expression

Ex.- // expre_Expressions_with_the_Conditional_Operator.cpp

// compile with: /EHsc

// Demonstrate conditional operator

```
#include <iostream>
using namespace std;
void main() {
  int i = 1, j = 2;
  cout << ( i > j ? i : j ) << " is greater." << endl;
}
```

f. Operator / is used to find the quotient while % is used to find the remainder in the c++ expression for ex.
int a=5,b=2,c; if c=a/b the result of c will be 2 and if c=a%b; the value of c will be 1

Q.No.4
a. A function prototype is a declaration of the function that tells the program about the type of the value returned by the function and the number and the type of arguments.
The prototype declaration looks just like a function definition except that it has no body.
Ex. void abc();
b. Given on Page No. 48-49..

e. to j . Do yourself.

Ans 5 :
a)

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main( )
{
        int A[10][10];
        int i,j,N;
        clrscr( );
        cout<<"\nHow many rows and columns required for matrix: ";
        cin>>N;
        cout<<"\nEnter "<<N*N<<" elements: ";
        for(i=0;i<N;i++)
        {
                cout<<"Enter the elements into Row "<<i+1<<": ";
                for(j=0;j<N;j++)
                  cin>>A[i][j];
        }
        clrscr( );
}

cout<<"\nThe entered elements in the matrix are: \n";
for(i=0;i<N;i++)
{
        for(j=0;j<N;j++)
        cout<<A[i][j]<<"\t";
        cout<<endl;
}
cout<<"\n\n\nThe elements which are belongs to only diagonals...\n";
for(i=0;i<N;i++)
{
  for(j=0;j<N;j++)
    if((i==j)||((i+j)==(N-1)))
        cout<<setw(6)<<A[i][j];
```

```
            else
                  cout<<" ";
                  cout<<endl;
          }
          getch( );
}
```

b**)**

```
#include<iostream.h>
#include<conio.h>
long f =1;
long factorial(int n)
{
  if (n==0)
    return f;
else
    f=n*factorial(n-1);
}
void main( )
{
    clrscr( );
    long num;
    cout<<"\nEnter the number to which you want to find factorial: ";
    cin>>num;
    cout<<"\nThe factorial of the number = "<<factorial(num);
    getch( );
}
```

c)

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
void main( )
{
  char str[300];
  int i,charcount=0,words=1,lines=1;
  clrscr();
  cout<<"\nEnter the Paragraph ie message: \n";
  gets(str);
  for(i=0;str[i]!='\0';i++)
   {
       charcount++;
if(str[i]==' ')
         words++;
      if (charcount%80==0)
         lines++;
   }
 cout<<"\nNumber of Characters in the entered message: "<<charcount;
 cout<<"\nNumber of Words in the entered message: "<<words;
 cout<<"\nNumber of Lines in the entered message: "<<lines;
 getch( );
}
```

d)

```
#include<iostream.h>
#include<conio.h>
void main( )
{
        clrscr( );
        int A[20],N,i,j,temp;
        cout<<"\nEnter the number of elements:";
        cin>>N;
        for(i=0;i<N;i++)
        cin>>A[i];
        //Bubble sort technique
        for(i=0;i<N;++i)
          for(j=0;j<(N-1)-i ;j++)
            if(A[j]>A[j+1])
              {
                Temp=A[j];
                A[j]=A[j+1];
                A[j+1]=Temp;
                }
cout<<"The Elements in the array after sorting…. ";
for(i=0;i<N;i++){
cout<<A[i]<<'\t';
}
```

e)

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void main( )
{
  clrscr( );
  double d1,d2,b,a,c,d;
  cout<<"\nEnter the value of b,a and c: ";
  cin>>b>>a>>c;
 d=(b*b-sqrt(4*a*c));
 if(d==0)
        cout<<"\nRoots are equal or distinct";
 else if(d>=0)
        cout<<"\nRoots are Real";
 else
 {
     cout<<"\nRoots are complex..ie Imaginary";
     d1=(-b+d)/(2*a);
     d2=(b+d)/(2*a);
     cout<<"\nD1: "<<d1;
     cout<<"\nD2: "<<d2;
     getch( );
  }
}
```

Sample Paper -III

Subject – Computer Science
Class – XI

Full Marks : 70 *Duration : 3 Hrs.*

GENERAL INSTRUCTIONS :

a) All questions are compulsory.
b) Programs should be written using the syntax of C++ Language.
c) Proper comments are expected with each program.

Q1. a) Define the term Booting. **(1)**

b) What are comments and Indentations? Why are they important? **(2)**

c) Find the output of the code given below : **(3)**

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
void main()
{
        int i = 5;
        int j = 10;
        clrscr();
        cout<< i/j<<"\n";
        cout<< i%j<<"\n";
        i += j;
        cout<< ++i+j << j--<<"\n";
        cout<<i<<j<<"\n";
        cout<<++i+j<<++j<<"\n";
        cout<<i<<j<<"\n";
        getch();
}
```

d) Write a function pattern( ) having two arguments a character ch and integer i. When user calls this function it should print a pattern. For example if User calls the pattern as pattern('@' , 5) it should print the following pattern : **(4)**

```
@
@ @
@ @ @
@ @ @ @
@ @ @ @ @
```

and if it is called like : pattern('%' , 3); it should print :

```
%
% %
% % %
```

Q2. a) Define the term Modular approach of programming. **(1)**

b) Write the names of the individual header files to which the following functions belongs : i) gets() , ii) pow() , iii) toupper() , iv) log10() **(2)**

c) Find the output of the following code snippet : **(3)**

```
#include<iostream.h>
#include<conio.h>
#include<dos.h>
int jumpingBit[ ] = {8,40,20,60};
void main()
{
        clrscr();
        int i = 0, j = 1;
        while( jumpingBit[0] > 0)
        {
                jumpingBit[ j] += 1;
                jumpingBit[i] -=1;
                for(int k = i ; k<= 3 ; k++)
                cout<<jumpingBit[k]<< " ";
                cout<<"\n";
                if(j < 3)
                j+=1;
                else
                j =1;
        }
        getch();
}
```

d) Consider the structure called Point having following format : **(4)**

```
struct Point {
        int xCoord;
        int yCoord;
};
```

write a function called getLine( ) which accepts two points as parameters and then returns the length of line being drawn between them using distance formula.
[the distance formula beween points P and Q gives square of the distance as :
$(PQ)^2 = (x_2 – x_1)^2 + (y_2 – y_1)^2$]

Q3. a) What is the difference between a++ and ++a according to C++ syntax, if a is a variable? **(1)**
b) Convert $(243)_{10}$ into its octal notation and 10110011 into its decimal form. **(2)**
c) What do you mean by an infinite loop? What category of error it belongs to? Explain with proper example. **(3)**
d) Write a program to convert a temperature inputted in Celsius into Fahrenheit and using only ternary operator (conditional Operator). **(4)**

Q4. a) What is a random() function ? Under which header file is it defined? **(1)**
b) What is an array? How is it different from normal variables? **(1)**
c) Find the error in the following code snippet , assuming all necessary header files included : **(2)**

```
structure ComplexNumber
{
        int realPart = 0;
        int imgPart = 0;
}
void main( )
{
        int ComplexNumber C[  ];
```

```
        cout<<"Input the real part :";
        cin>>ComplexNumber[0].realPart ;
        cout<<"Input the imaginary part";
        cin>>ComplexNumber[0].imgPart;
        cout<<"The number is : "<<ComplexNumber[0].realPart << " +" <<
                        ComplexNumber[0].imgPart << "i" ;
}
```

d) Differentiate between Cache memory , RAM memory and ROM memory. **(3)**

e) A two dimensional array would be called a Sparse matrix if its maximum number of elements are zero (0). Write a program which checks whether an inputted 3x3 2-D array is a sparse matrix or not. **(3)**

Q5. a) What is the difference between signed and unsigned datatypes? **(1)**

b) What is a reference variable? Declare one. **(1)**

c) After completing the only one line of code for the function isvowel() find the output of the whole program : **(2)**

```
#include<stdio.h>
#include<ctype.h>
#include<conio.h>
#include<string.h>
int isvowel(char );
void main( )
{
        char str[ ] = "Kamal is great";
        for(int i = 0 ; i<= strlen(str) -1 ; i++)
        {
                if( isalpha(str[i]))
                  if(islower(str[i]))
                    if(isvowel(str[i]))
                        str[ i ] = '#';
        }
        puts(str);
        getch();
}
int isvowel(char ch)
{
        _____ ;   // code line 1 (don't write more than two
        _____ ;   // lines of code.
}
```

d) Find the error(s) in the given code explaining the reasons and then rectify them : **(3)**

```
#include<iostream.h>
#include<conio.h>

void KamalsTrick ( int = 10 , int = 20 , int =30);
void KamalsTrick (int , int);

void main( )
{
        KamalsTrick(1 , 2) ;
}
void  KamalsTrick( int x , int y , int z)
```

```
{
        cout<<endl << x << endl << y<< endl << z ;
}
void KamalsTrick(int x , int y)
{
        cout<<endl << x<< endl << y ;
}
```

e) Write a function to check whether a string passed to it is a Toggled String or not. A Toggled String is a string whose alphabetic characters are alternately of changed cases. For Example : "WeRe ArE yOu" is a toggled string similarly "wErE aRe YoU" is also a toggled string. The function assumes that the inputted string has only alphabets, and no special symbols. **(3)**

Q6. a) What are #define directives? Give a example. **(1)**

b) What is prototype mismatch error? Why it happens? **(1)**

c) Write a program to find the length of the hypotenuse of a given right angled triangle if its other two sides are inputted from user. **(2)**

d) What are utility Software? Why do we use them? Name any two of them, and describe any one of them. **(4)**

e) What is the difference between RISC and CISC. Discuss. **(2)**

Q7. a) TCP/IP are a type of protocols which are needed for network transmission of data through a communication channel. TCP (Transmission Control Protocol) breaks the data to be transferred in "packets" where a packet has following data members :

```
source address :        long    // the source IP address as a long value
destination address:    long    // the destination IP address as long value
data                    :a string // contains the content to be transferred as text
parity bit              :       int
```

Now, considering the above description code for following :

i) create a structure called TcpPacket which has the mentioned data members. **(1)**

ii) create a function called sendPacket( ) which inputs all the data member values of a TCP packet and returns a TCP packet. The sendPacket() function also have an overloaded version which inputs an TCP packet as parameter and it simply returns it. **(2)**

iii) create a function called receivePacket( ) which inputs a TCP packet and a current IP address (long type) as parameter. If the current IP address matches with destination address then the receivePacket( ) shows the data part of the packet is parity bit is true else it simply calls the second version of the overloaded sendPacket( ) function, to propagate the packet to its correct destination. **(4)**

b) Write a program which inputs the eye color of a person as integer , for e.g. 1 for black , 2 for blue , 3 for green , 4 for brown , and then prints the following statement for each of them : **(3)**

```
for black   :   "You are almost same as me"
for blue    :   "I will be drown in the deep sea of your eyes"
for green   :   "You are one in million, I am lost in your eyes"
for brown   :   "brown a democratic color"
```

You must use switch case statement for the above program.

MARKING SCHEME :

Q1. a) Definition / working of booting process. (1 marks)

b) comment description and its use - 1 mark + indentation description and use 1 mark

c) 0 (each correct line output ½ marks = 6 x ½ = 3)
5
2510
169
2710
1710

d) i) correct prototype of code : - ½ mark
ii) correct logic - 2 marks
iii) correct syntax - ½ marks
iv) correct output/return value - 1 marks

Q2. a) Modular approach definition / description - 1 marks.

b) Each correct header file – ½ marks

c) output :

7	41	20	60
6	41	21	60
5	41	21	61
4	42	21	61
3	42	22	61
2	42	22	62
1	43	22	62
0	43	23	62

half correct output - 1 ½ marks and next half correct output – 1 ½ marks

d) Correct function prototype declaration – ½ marks
correct type of use of C++ expression – 2 mark
correct use of logic – ½ mark
correct output / return 1 marks

Q3. a) one correct working difference – 1mark.

b) 1 marks for octal conversion + 1 marks for decimal conversion.

c) 1 mark for Infinite loop definition/description + 1 mark for describing type of error it belongs + 1 mark for example expalantion

d) 1 marks for correct variable declaration and syntax + 1 marks for correct logic used + 1 mark for correct use of ternary operator + 1 mark for correct output produced.

Q4. a) ½ mark for correct definition / prototype with description of random() function + ½ mark for correct header file.

b) ½ mark for definition/ correct description of arrays + ½ mark for difference from normal variable.

c)
```
structure ComplexNumber     // ( ½ mark)
{
        int realPart = 0;
        int imgPart = 0;          // ½ mark
}
void main( )
{
        int ComplexNumber C[ ];              // ½ mark
        cout<<"Input the real part :";
```

```
        cin>>ComplexNumber[0].realPart ;   // ½ mark
        cout<<"Input the imaginary part";
        cin>>ComplexNumber[0].imgPart;
        cout<<"The number is : "<<ComplexNumber[0].realPart << " +" <<
                        ComplexNumber[0].imgPart << "i" ;
}
```

d) 2 points of difference between the memory modules will carry (1 ½ + 1 ½)

e) 1 mark for Correct declaration and input of a 3x3 2-D array + 1 mark for correct logic for evaluation of sparse matrix + 1 mark for valid and correct result/output.

Q5. a) 1 mark for difference between signed and unsigned variables with example.

b) ½ marks for description pf reference variable + ½ marks for declaration.

c)

```
#include<stdio.h>
#include<ctype.h>
int isvowel(char );
void main( )
{
        char str[ ] = "Kamal is great";
        for(int i = 0 ; i<= strlen(str) -1 ; i++)
        {
            if( isalpha(str[i]))
              if(islower(str[i]))
                if(isvowel(str[i]))
                    str[ i ] = "#";
        }
        puts(str);
}
int isvowel(char ch)
{
    int a = (ch =='A' || ch =='a' || ch =='E' || ch == 'e' || ch == 'I' || ch == 'i ' ||
        ch == 'O' || ch == 'o' || ch == 'u' || ch == 'U' ) ? : 1 : 0 ;
                                                    //  1st line  ½ marks
    return a;                                       // 2nd line ½ marks

        //OR ALTERNATIVELY
    if (ch =='A' || ch =='a' || ch =='E' || ch == 'e' || ch == 'I' || ch == 'i ' ||
        ch == 'O' || ch == 'o' || ch == 'u' || ch == 'U' )
        return 1;                                   // 1st line
    else
        return 0;                                   // 2nd line
}
```

output : K#m#l #s gr##t (1 marks)

d) Error : Ambiguous call for function prototype KamalIsGreat(int , int) (**1 mark**)

cause : The ambiguity comes from the conflict between the normal function call KamalIsGreat(int , int) and with its overloaded default parameterized version KamalIsGreat(int = 10 , int = 20 , int = 30). Compiler is unable to resolve, which version is to be called , because parameter of both the version gives exact match to the function call. (or any alternative answer having same meaning as explained. **(1 mark)**

rectification : omission of one of the conflicting prototypes from the program. **(1 mark)**

e) ½ marks for correct function prototype. + ½ marks for correct variable and input + 1 ½ marks for correct logic used to find the result + ½ marks for correct output / return value. (we may neglect header file declarations)

Q 6. a) ½ mark for Definition/description of #include directive. + ½ mark for correct example.

b) ½ marks for prototype mismatch error + ½ marks for accurate reason.

c) ½ marks for correct variable declaration + ½ marks for correct inputting + 1 marks for correct C++ expression and output.

d) 1 mark for definition/description of utility software + 1 mark for use of them + 1 marks for examples of few of them + 1 mark for explaining any one of them

e) two/three correct differences between RISC and CISC. (2marks)

Q7. a) i)

```
struct TcpPacket {                                          // (1marks )

        long source;
        long destination ;
        char data [ ]    ;
        int parity_bit ;
};
```

ii)

```
TcpPacket  sendPacket( )                    // 1st overloaded version
{
    TcpPacket  dataGram;                                    // (3 marks )
    cout<<"Input the source IP address as a long value";
    cin>> dataGram.source ;
    cout<<:Input the destination IP address as a long value";
    cin>> dataGram.destination ;
    cout<<"Input the data as text"  ;
    gets(dataGram.data);
    cout>>"input the status of parity bit checker [ 1 for true / 0 for false] ";
    cin>>parity_bit ;
    return dataGram ;
}

    TcpPacket sendPacket( TcpPacket T )   // 2nd overloaded version
    {
        return  T ;
    }
```

iii)

```
TcpPacket receivePacket ( TcpPacket  incomingPack  , long currIP )
{
        if ( incomingPack.destination == currIP)          // ( 3 marks )
        {
           if ( incomingPack.parity_bit)
                puts( incomingPack.data);
        }
        else
                sendPacket( incomingPack) ;
}
```

b) 1 mark for correct use of variable declaration and inputs + 1 mark for correct use of switch case statement (with break statement) + 1 marks for over-all correctness and output syntaxes.

SAMPLE PAPER-IV

CLASS XI (COMPUTER SCIENCE)

Max Marks : 70 Time Allowed : 3:00 Hrs

General Instructions :

- *All questions are compulsory.*
- *Programming Language: C++*

Q1 Answer the following questions –

(a) Give two major innovations each of first and second generation computers. 2

(b) What are the main categories of software? 2

(c) Distinguish CPU and ALU 1

(d) What is booting? 1

Q2 Answer the following –

(a) What is an interpreter? 1

(b) What is a guard code? 1

(c) Explain Object Oriented Programming. 2

(d) Differentiate the two compile-time errors. 2

(e) Mention the steps you would follow while writing a program. 2

(f) What is Abstraction and Encapsulation? How are these two terms interrelated? 2

Q3 Answer these questions-

(a) What is the difference between 'm' and "m" in C++? 1

(b) What is a keyword? 1

(c) Explain the role of header file *iostream.h* in C++. 2

(d) What is the purpose of stream insertion and stream extraction operators in C++? 2

(e) What is wrong with the following statements?

(i) const int discount**;**

(ii) enum {stop, look, go}**;** 2

(f) Evaluate the following C++ expression –

```
int a, b =3, c= 8;
a = b * 5 / 7 + c / 2 - b + 5 ;
```
2

Q4Answer the following questions -

(a) Rewrite the following statement using if-else:

```
n = ((a < b ) ? a : b ) ;
```
1

(b) What will be the output of the following program?

```
#include<iostream.h>
#inlcude<conio.h>
void main()
{
        int x = 9, y = 5 ;
        clrscr( ) ;
        z = ++x  +  y-- ;
        cout << z ;
}
```
2

(c) Write a program in C++ to input a number. If the number is even, print its square, otherwise print its cube. 2

(d) Write a program to input three integers and print the largest of the three. 3

(e) Write a C++ program to find the sum of digits of a number entered by the user. 4

Q5Answer the following -

(a) What output does the following code fragment produce?

```
        :
        :
int stock [] = {10, 22, 15, 12, 18};
int total = 0 ;
for (int i = 0 ; i < 5 ; ++ i)
        if (stock [i] > 15)
                total += stock [i] ;
        cout << "Total = " << total << endl ;
```

2

(b) How does C++ view a string as? Which character marks the end of a string? 2

(c) Correct the following code so that it is functional, underline the corrections.

```
value = 4 ;
do ;
{
        total += value ;
        cout << total ;
while value <= 8;
```

2

(d) Write a C++ 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? 4

Q6 Answer these questions -

(a) What is a function? Write the steps in using a function. 3

(b) Given the following code fragment :

```
int main()
{
        float addnum (float , float) ;
        :
        :
}
void calculate (void)
{
        float x, y, s ;
        cin >> x >> y ;
        s = addnum (x , y);
}
```

Why will the above given function not work? What should be done to make it work? 2

(c) Write a function that takes an **int** argument and doubles it. The function does not return any value. Also write the main function to invoke this function. 3

(d) Write a program to add to matrices A[3][4] and B[3][4]. 4

Q7Answer the following –

(a) Expand the term ASCII 1

(b) What is a port? 1

(c) Find the one's complement of -14 1

(d) What is the difference between RAM and ROM? 2

(e) What are the two categories of printers? Which type of printer is faster? 2

(f) Convert the following numbers to binary –

(i) $(B6A)_{16}$ (ii) $(6214)_8$ (iii) $(97)_{10}$ 3

MARKING SCHEME

Max Marks : 70 Time Allowed : 3:00 Hrs

General Instructions :

- *All questions are compulsory.*
- *Programming Language: C++*

Q1

(a) Two innovations of First Generation Computers	1
Two innovations of Second Generation Computers	1
(b) Correct names of categories of software	½
Brief description of each type ½ mark each	1 ½
(c) Correct difference or description of CPU and ALU	1
(d) Definition of booting.	1

Q2

(a) Definition or purpose of interpreter.	1
(b) Definition of guard code.	1
(c) Definition of OOP	1
Any two important features of OOP	1
(d) Description or definition of syntax error	1
Description or definition of semantic error	1
(e) Steps in writing a program – (½ mark each for atleast 4 steps)	
(i) Analyze the problem	
(ii) Design program.	
(iii) Code program.	
(iv)Test & debug program.	
(v) Complete the documentation.	
(vi) Maintain the program	2
(f) Description of Abstraction and Encapsulation	1
Relation between Abstraction and Encapsulation	1

Q3

(a) Correct reason (‘m’ = character & “m” = string)	1
(b) Definition of keyword.	1
(c) Two correct purposes / roles of *iostream.h* (1 mark each)	2
(d) Purpose of stream insertion operator (<<).	1
Purpose of stream extraction operator (>>).	1
(e) (i) constant variable not initialized	1
(ii) identifier missing.	1
(f) Correct value of variable	2

Q4

(a) Syntax of if-else	1
Correct logic / code	1
(b) Correct output	2
(c) Correct syntax	1
Program logic (conditional expression)	1
(d) For correct declarations	½
For input of values	½
For correct use of conditional construct	1 ½
For correct output statement	½
(e) For correct declarations & input	1

For correct use of iteration 1 ½
For output statement ½

Q5

(a) For correct output 2
(b) String is viewed as an array of characters 1
'\0' (null character) marks the end of a string 1
(c) value = 4 **;**

```
do ; // semi-colon to be  removed                    ½
{
        total += value ;
        cout << total ; // missing parenthesis of do loop    ½
                        // update statement for " value"     ½
while value <= 8;       // incorrect conditional expression  ½
```

(d) For including header files ½
For correct declarations & input 1
For correct use of nested conditional statement 2
For output statement ½

Q6

(a) Definition of function. 1
Steps in function (prototype, function call, function definition) 2
(b) Reason – placement of prototype declaration inside main() 1
For correction of code 1
(c) For correct return type of function ½
For function parameter ½
For function call ½
Overall Logic and code organization 1 ½
(d) For including header files, declarations ½
For initialization of the arrays 1
For correct logic to add the two matrices 2
For output statement ½

Q7

(a) Full form – American Standard Code for Information Interchange 1
(b) Ports – connecting points pf various computer devices 1
(c) For correct calculation 1
(d) For description of RAM 1
For description of ROM 1
(e) For describing Impact printer ½
For describing Non-Impact printer ½
Non- impact printers are faster 1

Sample Paper-V

Subject : Computer Science

Time Allowed : 3 hours ***Class: XIth*** ***Max. Marks : 70***

Note : Attempt all questions.

1. Answer the following questions:
   (a) Convert the following numbers to equivalent decimal numbers: 2
   (i) $(11000111000)_2$ (ii) $(11.011)_2$
   (b) Convert the following numbers to equivalent Binary numbers: 3
   (i) $(256)_{10}$ (ii) $(8AF)_{16}$ (iii) $(7621)_8$
   (c) Differentiate between Impact and Non-impact printers. 2
   (d) Differentiate between Dynamic RAM and Static RAM. 2
   (e) What is the purpose of Accumulator Register (AC) in the ALU. 1
2. Answer the following questions:
   (a) Differentiate between Compiler & Interpreter. 2
   (b) What are the different types of digital computers based on their performance? 1
   (c) What is an operating System? 1
   (d) How you create a folder in MS- Windows? 1
   (e) How you rename a File/folder in MS-Windows? 1
   (f) Write the names of the Header Files to which following Functions belong: 2
   (i) getchar( ) (ii) strlen( ) (iii) isdigit( ) (iv) strcat ( )
   (g) What is an Event-Driven Programming? 2
3. Answer the following questions:
   (a) Differentiate between Run-time & Compile-time errors. 2
   (b) What is debugging ? 1
   (c) What do you mean by Encapsulation? 2
   (d) Write an algorithm to find whether a given number is odd or even. 2
   (e) Draw a flowchart to find out the factorial of a given integer. 3
4. Answer the following questions:
   (a) Write a C++ program that inputs a student's marks in three subjects (out of 100) and prints the percentage marks. 2½
   (b) What is an identifier? Which of the following are invalid identifiers? 2½
   23we, my.file, sum_r, ravi
   (c) What is constant. How you declare a constant in C++? (write syntax) 2
   (d) What will be the result of following expression if age = 65
   Age>65?350 : 100 1½
   (e) Write an equivalent C++ expression for the following expressions: 1½
   $ut + ½ ft^2$

5. Answer the following questions:

(a) Explain the use of break and continue statement in C++, with example. 3

(b) Rewrite the following code using switch : 2

```
if (a == 0)
        cout << “Zero”;
if (a == 1 )
        cout << “One”;
if (a == 2 )
        cout << “ Two”;
if (a == 3)
        cout << “ Three”;
```

(c) What will be the output of the following code fragment : 2

(i)
```
for (i = 0; i >3; i++)
cout<< “Hello” ;
```

(ii)
```
for (int I =0; I <10; I++ )
  {     if (I%2== 0)
        cout<< I;
   }
```

(d) How many times is the following loop executed ? . 1

```
int s = 0 , I = 0;
do
{ s += I;
} while (I < 5);
```

(e) Write a C++ program to print the following pattern : 2

```
*
* *
* * *
* * * *
* * * * *
```

6. Answer the following questions:

(a) What is an array ? How you declare a single dimensional array in C++ ? 3

(b) Write a program that subtracts two matrix X[3][3] and Y[3][3]. 5

(c) Write a C++ function that take two int arguments and return the largest of given numbers. 2

7. Answer the following questions:

(a) Differentiate between CALL by reference and CALL by value. 2

(b) What is meant by scope ? what kinds of scope is supported by C++ ? 3

(c) Write a C++ program to find the sum of n terms of following series:

$(1^1/1!) + (2^2/2!) + (3^3/3!) + (4^4/4!) + \ldots\ldots\ldots$ 5