

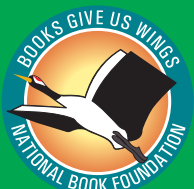
Textbook of

COMPUTER SCIENCE

GRADE

12

National Book Foundation
as
Federal Textbook Board
Islamabad



COMPUTER SCIENCE

Grade

12



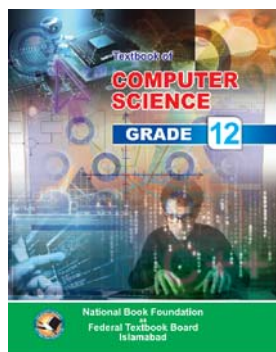
National Book Foundation
as
Federal Textbook Board
Islamabad

OUR MOTTO

• Standards • Outcomes • Access • Style

© 2018 National Book Foundation as Federal Textbook Board, Islamabad.
All rights reserved. This volume may not be reproduced in whole or in part in any form (abridged, photo copy, electronic etc.) without prior written permission from the publisher.

Textbook of
Computer Science Grade - 12



Content Authors	:	Mr. Mohammad Sajjad Heder, Mr. Mohammad Khalid
Pedagogical Author	:	Dr. Rahat Hussain Bokhari
Elaborator / Designer	:	Hafiz Rafiuddin, Shahzad Ahmad
Editor	:	Mr. Majeed-ur-Rehman Malik
Desk Officer	:	Dr. Zulfiqar Ali Cheema (EO) CD & TP Wing
Editor	:	Majeed-ur-Rehman Malik
Management of	:	Mr. Ishtiaq Ahmed Malik
Supervision of	:	Prof. Dr. Attash Durrani (T.I., S.I.), Advisor FTBB (NBF), Member National Council, Localization Consultant of Microsoft, UNICODE Contributor.

First Edition	:	2018 Qty: 24000
Price	:	Rs. 140/-
Code	:	STE: 575
ISBN	:	978-969-37-1102-8
Printer	:	ST Printers Rawalpindi.

for Information about other National Book Foundation Publications,
visit our Web site <http://www.nbf.org.pk> or call **92-51-9261125**
or Email us at: books@nbf.org.pk

PREFACE

The textbook of COMPUTER SCIENCE for GRADE - 12 has been developed according to the National Curriculum 2009. This book consists of seven chapters. It is based on C Programming Language, Computer Logic and Gates, and HTML.

Computers are changing our world and driving our economy. We are living in a world controlled by computer software. Students must learn how to design and write computer programs to solve problems related with any field. No matter what students' future plan may be, learning computer programming is essential for them. It enables them to succeed in the technology-driven world of 21st century. Programming allows putting ideas into reality in any industry. It provides skills about how to make use of programming language to solve various computational problems. There will be high demand of programmers in tomorrow's world to build useful applications and websites.

This textbook is developed with due care using simple language, supporting pictures to make it understandable by secondary school students. Keen efforts have been made to minimize the errors and omissions. This will surely enhance the confidence and comprehension of the students and general readers in the subject. Yet there is always room for improvement and any suggestion with regard to the quality of work will be warmly welcomed at our end.

Quality of Standards, Pedagogical Outcomes, Taxonomy Access and Actualization of Style is our motto. With this slogan the textbook is presented for use as per SOP's 2010, NCC Standards 2016 and National Curriculum Framework 2017. It will be revised accordingly in the light of feedback.

Dr. Inam ul Haq Javeid
(Pride of Performance)
Managing Director
National Book Foundation

TABLE OF CONTENTS

CHAPTER 1: OPERATING SYSTEM.....	08
1.1 Introduction	09
1.1.1 Operating System	09
1.1.2 Commonly Used Operating Systems	09
1.1.3 Types of Operating System	13
1.1.4 Single-User and Multi-User Operating Systems	14
1.2 Operating System Functions.....	15
1.3 Process Management	17
1.3.1 Process Definition.....	17
1.3.2 Various States of a Process.....	18
1.3.3 Thread and Process.....	18
1.3.4 Multithreading	19
1.3.5 Multitasking.....	19
1.3.6 Multiprogramming	20
CHAPTER 2: SYSTEM DEVELOPMENT LIFE CYCLE	24
2.1 System Development Life Cycle	25
2.1.1 A System	25
2.1.2 System development life cycle (SDLC) and its importance	25
2.1.3 Objectives of SDLC	25
2.1.4 Stakeholders of SDLC	26
2.1.5 SDLC Phases / Steps	26
1. Defining Phase	27
2. Planning Phase.....	27
3. Feasibility.....	27
4. Analysis Phase	28
5. Requirement Engineering	28
6. Design Phase	29
7. Construction / Coding	31
8. Testing / Verification	31
9. Deployment / Implementation	32
10. Maintenance / Support.....	33
2.1.6 Personal involved in SDLC and their Role	33
CHAPTER 3: OBJECT ORIENTED PROGRAMMING IN C++	40
3.1 Introduction	41
3.1.1 Computer Program	41
3.1.2 Header File and Reserved Words.....	41
3.1.3 Structure of A C++ Program	42
3.1.4 Statement Terminator	43
3.1.5 Comments in C++ Program	44
3.2 C++ Constants and Variables.....	45
3.2.1 Constants and Variables.....	45
3.2.2 Rules for Specifying Variable Names.....	45
3.2.3 C++Data Types	46
3.2.4 The Const Qualifier.....	47
3.2.5 Variable Declaration and Initialization	47

3.2.6	Type Casting	47
3.3	Input / Output Handling.....	48
3.3.1	The cout Statement	49
3.3.2	The cin Statement.....	49
3.3.3	The getch(), gets() and puts() Functions	50
3.3.4	The Escape Sequence	51
3.3.5	Commonly Used Escape Sequences.....	52
3.3.6	Programming with I/O Handling Functions	53
3.3.7	The manipulators endl and Setw.....	54
3.4	Operators in C++	57
3.4.1	Types of Operators	57
3.4.2	Unary, Binary and Ternary Operators	64
3.4.3	Order of Precedence of Operators	64
3.4.4	Expressions in C++.....	65
CHAPTER 4: CONTROL STRUCTURES		70
4.1	Decision Structures	71
4.1.1	The If Statement	71
4.1.2	The If-Else Statement	73
4.1.3	The Else-If Statement	74
4.1.4	The Switch Statement.....	77
4.1.5	Difference Between If-Else If and Switch Statements	80
4.2	LOOPS	81
4.2.1	The For Loop	82
4.2.2	The While Loop	84
4.2.3	The Do While Loop	86
4.2.4	Difference Between While Loop and Do-While Loop.....	87
4.2.5	The Break and Continue Statements	87
4.2.6	The Exit Function.....	89
4.2.7	Nested Loop	89
CHAPTER 5: ARRAYS AND STRINGS.....		96
5.1	Introduction to Arrays.....	97
5.1.1	Concept of an Array	97
5.1.2	Declaring and Array	98
5.1.3	Initialization of Array	98
5.1.4	Using Arrays in Programs	98
5.1.5	The sizeof() Function	102
5.2	Two Dimensional Arrays.....	103
5.2.1	Introduction to Two Dimensional Arrays.....	103
5.2.2	Defining and Initializing A Two Dimensional Array	103
5.2.3	Accessing and Writing in a Two Dimensional Array	105
5.3	STRINGS	107
5.3.1	Introduction to Strings	107
5.3.2	Defining a String	107
5.3.3	Initializing Strings.....	108
5.4	Commonly Used Strings Functions	108
CHAPTER 6: FUNCTIONS.....		116
6.1	Functions.....	117

6.1.1	Types of Functions	117
6.1.2	Advantages of Functions	120
6.1.3	Function Signature.....	120
6.1.4	Function Components.....	120
6.1.5	Scope of Variables in Functions.....	122
6.1.6	Parameters.....	125
6.1.7	Local and Global Functions	126
6.2	Passing Arguments and Returning Values	129
6.2.1	Passing Arguments.....	129
6.2.2	Default Arguments.....	134
6.2.3	Return Statement.....	135
6.3	Function Overloading	136
6.3.1	Advantages of Function Overloading	136
6.3.2	Use of Function Overloading	137
CHAPTER 7: POINTERS		142
7.1	POINTERS.....	142
7.1.1	Pointer Variable	142
7.1.2	Memory Addresses.....	143
7.1.3	Reference Operator (&)	143
7.1.4	Dereference Operator (*)	144
7.1.5	Declaring Variables of Pointer Types	145
7.1.6	Pointer Initialization.....	146
CHAPTER 8: OBJECTS AND CLASSES		150
8.1	Classes and Objects	151
8.1.1	Class and Object	151
8.1.2	Member of a Class.....	153
8.1.3	Access Specifiers	155
8.1.4	Data Hiding.....	159
8.1.5	Constructor and Destructor	159
8.1.6	Declaration of Objects for Accessing Members of a Class	165
8.1.7	Inheritance and Polymorphism	167
CHAPTER 9: FILE HANDLING		174
9.1	File Handling.....	174
9.1.1	Types of Files	175
9.1.2	Opening File	176
9.1.3	bof() and eof().....	181
9.1.4	Stream.....	182
9.1.5	Types of Streams.....	182





1

OPERATING SYSTEM



After completing this lesson, you will be able to:

- Define operating system
- Describe commonly used operating systems
- Explain the following types of operating systems
 - Batch Processing Operating System
 - Multiprogramming Operating System
 - Multitasking Operating System
 - Time-sharing Operating System
 - Real-time Operating System
 - Multiprocessor Operating System
 - Parallel Processing Operating System
 - Distributed Operating System
 - Embedded Operating System
- Define Single-user and Multi-user Operating Systems
- Describe the following main functions of operating system
 - Process Management
 - Memory Management
 - File Management
 - I/O System Management
 - Secondary Storage Management
 - Network Management
 - Protection System
 - Command Interpreter
- Define Process
- Describe new, running, waiting/blocked, ready and terminated states of a process
- Differentiate between the following.
 - Thread and process
 - Multi-threading and multitasking
 - Multitasking and multiprogramming.



1.1 INTRODUCTION

An **operating system** is the most important software that runs on a computer. It manages the computer's memory and processes, as well as all of its software and hardware. It also allows users to communicate with the computer. Without an operating system, a computer user cannot run any program on the computer. It automatically loads in RAM when the computer is turned on. Operating systems exist from the very first computer generation and keep evolving with time.

1.1.1 OPERATING SYSTEM

An Operating System (OS) is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.

Every desktop computer, tablet, and smartphone includes an operating system that provides basic functionality for the device. Common desktop operating systems include Windows, OS X, and Linux.

Operating system performs the following tasks.

- Loads application/system software into main memory and executes it.
- Controls the operation of main memory and external storage devices.
- Manages files and folders on storage devices such as hard disk, USB flash drive, etc.
- Manages the operations of all the input/output devices.
- Allows multitasking to handle several tasks at the same time such as running a spreadsheet software and a word-processor simultaneously.
- Performs network operations which enable a number of users to communicate with each other in a network environment and share computer resources such as CPU, main memory, hard disk, printer, Internet, etc.
- Detects hardware failures.
- Provides security through username and password.

1.1.2 COMMONLY USED OPERATING SYSTEMS

The commonly used operating systems are DOS, WINDOWS, Macintosh's OS X/OS2 and UNIX/LINUX.



Teacher Point

Before starting the chapter, the students could be encouraged to explain what they understand about Operating system.



DOS

DOS stands for Disk Operating System. It was developed in 1970s when microcomputer was introduced. It was called Disk Operating System because the entire operating system was stored on a single floppy disk. It had text-based (also known as command-line) user interface. The user had to type commands to interact with the computer.

The following are some DOS commands.

RENAME	For renaming a file
CD	For changing directory (called folder in Windows)
DIR	To display directories and files in a directory
DEL	To delete one or more files
COPY	To copy files from one drive/directory to another
FORMAT	To format a disk

The user had to learn the basic commands to operate the computer effectively. DOS was not a user-friendly operating system. DOS commands were difficult to learn, memorize and use for novice computer users. DOS had been used successfully on microcomputers for many years but it was replaced by a more user-friendly operating system called Windows in the early 1990s. DOS interface is shown in Figure 1.1.

```
C:\>dir

Volume in drive C is MS-DOS 5_0
Volume Serial Number is 446B-2781
Directory of C:\

COMMAND  COM      47845  11-11-91   5:00a
          1 file(s)      47845 bytes
                               10280960 bytes free

C:\>ver

MS-DOS Version 5.00

C:\>
```

Figure 1.1 DOS Interface



Teacher Point

Explain the importance of Operating System for a computer.



WINDOWS OPERATING SYSTEM

Windows operating system was developed in mid 1980s by Microsoft Corporation. It provides a Graphical User Interface (GUI) which is user-friendly. The user does not have to memorize commands like DOS. It allows user to give commands to computer through icons, menus, and buttons etc. Today, it is the most commonly used operating system on PCs and laptop computers all over the world. Microsoft has released many versions of Windows over the years to enhance its user interface in computer technology. Some popular versions of Windows in the past were Windows 95, Windows 98, Windows Millennium, Windows XP and Windows Vista etc. Windows 10 interface is shown in Figure 1.2.



Figure 1.2 Windows 10 Interface

Mac OS

Mac OS is a series of operating systems developed by Apple Corporation. Mostly it is installed on all the Apple computers. The latest version is known as OS X. It is the tenth major release of the Mac operating systems. It is a more secure operating system compared to Windows. Mac hardware and software works together very well with minimum flaws. Mac computer is of high quality but more expensive than IBM compatible computers. A large variety of application software is easily available for Windows operating system whereas the OS X has very limited application software. The OS X is not a widely used operating system like the Windows. Mac OS X interface is shown in Figure 1.3.



Teacher Point

Teacher may assist his/her students in the installation of common operating systems in computer and other devices.



Figure 1.3 Mac OS X Interface

UNIX

UNIX operating system was developed in early 1970s at Bell Laboratories research center by Ken Thompson and Dennis Ritchie. It was developed in C language. It provides greater processing power and better security than Windows operating system. Computers running UNIX operating system rarely have malware attack. It is available for a wide range of computer systems from microcomputers to mainframes. It is less popular on microcomputers on which Windows is pre-installed when they are sold. UNIX OS/390 interface is shown in Figure 1.4.

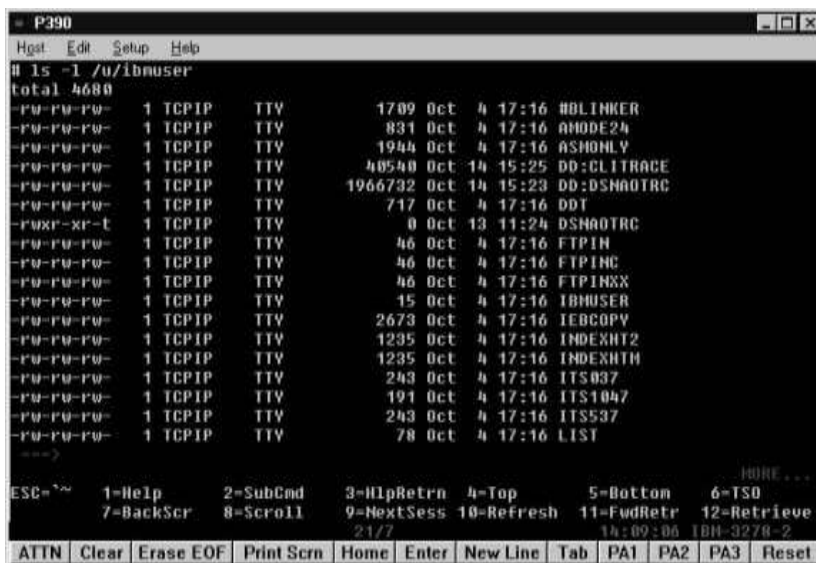


Figure 1.4 UNIX OS/390 Interface



Teacher Point

Teacher could also introduce the operating systems like iOS and Android.



1.1.3 TYPES OF OPERATING SYSTEMS

The following are the important types of operating systems that are commonly used on various computer systems.

Batch Processing Operating System

A batch processing operating system is a software that groups together same type of jobs in batches and automatically executes them one by one. It performs the same type of task on all the jobs in a batch in the sequence in which they appear. It provides an easy and efficient way of processing the same type of jobs. For example, at the end of month, banks print statement for each account holder. A batch processing system can easily and efficiently print each account holder's statement one by one.

Multiprogramming Operating System

A multiprogramming operating system is a software that loads one or more programs in main memory and executes them using a single CPU (Central Processing Unit). In fact, the CPU executes only one program at a time while other programs are waiting in queue. In multiprogramming system when one program is busy with input/output operation, the CPU executes another program that is in queue. In this way, multiprogramming operating system uses the CPU time and other resources of computer to improve the performance of computer.

Multitasking Operating System

A multitasking operating system is a software that performs multiple tasks at the same time on a computer that has a single CPU. The CPU executes only one program at a time but it rapidly switches between multiple programs and it appears as if all the users' programs are being executed at the same time.

Time-sharing Operating System

A time-sharing operating system is a software that shares the CPU time between multiple programs that are loaded in main memory. A time-sharing operating system gives a very short period of CPU time to each program one by one. This short period of time is called time slice or quantum. Since the CPU is switched between the programs at extremely fast speed, all the users get the impression of having their own CPU. It is used in mini and mainframe computers that support large number of users in big organization such as airline, bank, university, etc.

Real-time Operating System

A real-time operating system is a software that runs real-time applications that must process data as soon as it comes and provides immediate response. Real-time operating system executes special applications within specified time with high reliability. It is commonly used in space research programs, real-time traffic control and to control industrial processes such as oil refining.

Multiprocessor Operating System

A multiprocessor operating system is a software that controls the operations of two or more CPUs within a single computer system. All the CPUs of computer share the same main memory and input/output devices. Multiprocessing operating systems are used to obtain very



high speed to process large amount of data. It executes a single program using many CPUs at the same time to improve processing speed. Computers that support multiprocessing have sophisticated architecture which is difficult to design.

Parallel Processing Operating System

A parallel processing operating system is a software that executes programs developed in a parallel programming language. It uses many processors at the same time. In a parallel processing system, the task of a program that requires many calculations is divided into many smaller tasks and these are processed by multiple processors at the same time. Parallel processing operating systems are used in supercomputers that have thousands processors.

Distributed Operating System

A distributed operating system is a software that manages the operation of a distributed system. A distributed system allows execution of application software on different computers in a network. In a distributed system, user programs may run on any computer in the network and access data on any other computer. The users of distributed system do not know on which computer their programs are running. Distributed operating system automatically balances the load on different computers in the network and provides fast execution of application software.

Embedded Operating System

An embedded operating system is a built-in operating system which is embedded in the hardware of the device. It controls the operation of devices such as microwave oven, TV, camera, washing machine, games, etc. It runs automatically when the device is turned on and performs specific task.

1.1.4 SINGLE-USER AND MULTI-USER OPERATING SYSTEMS

Operating systems are divided into single-user and multi-user operating systems based on the number of users they can support.

Single-user Operating System

The operating system that allows only one person to operate the computer at a time is known as single-user operating system. Commonly used single-user operating systems are DOS and Windows.

Multi-user Operating System

The operating system that allows many users on different terminals or microcomputers to use the resources of single central computer (server) in a network is known as multi-user operating system. It is used on servers in business and offices where many users have to access the same application software and other resources. Some examples of multi-user operating systems are UNIX, Linux, Windows 2000 onward and Max OS X.



Teacher Point

Students may be taken to some organizations like Electric supply companies, Sui gas companies, Airlines, etc. to show the working of different types of operating systems, like Batch processing, Multiprogramming, Time sharing and Real time O.S. etc.



1.2 OPERATING SYSTEM FUNCTIONS

The following are the functions performed by operating system.

- Process Management
- Memory Management
- File Management
- I/O Management
- Secondary Storage Management
- Network Management
- Protection System
- Command-interpreter

Process Management

A process is a program in execution. Process management is the part of operating system that manages allocation of computer resources (like CPU time) to various processes in main memory. Process management actually describes the state and resource ownership of each process.

Example: In this example there are three processes **A**, **B** and **C** ready for execution. The OS will manage the CPU time as follows.

Process A has CPU cycle ($t_a = 5$ milli sec)

Process B has CPU cycle ($t_b = 2$ milli sec)

Process C has CPU cycle ($t_c = 1$ milli sec)

Case 1: When the 3 processes become ready in the order of ABC, the total execution time will be:

$$\tau = (5 + 7 + 8)/3 = 6.67 \text{ milli sec}$$

Case 2: When the 3 processes become ready in the order of BCA, the total execution time will be:

$$\tau = (2 + 3 + 8)/3 = 4.33 \text{ milli sec}$$

In the above example, in Case2, the OS is managing the processes more efficiently. The execution time in Case 2 is less as compare to Case 1.

Memory Management

Memory management is the part of operating system that controls and manages the operation of main memory during the operation of computer. It allocates space to programs that are loaded in main memory for execution. It keeps track of freed memory when a program is closed and updates the memory status.



Example: In this example the OS is managing memory for two processes P0 and P1. P1 is being loaded (swap in) and P0 is being taken out (swap out) from the main memory (RAM). The whole process is shown in Figure 1.5.

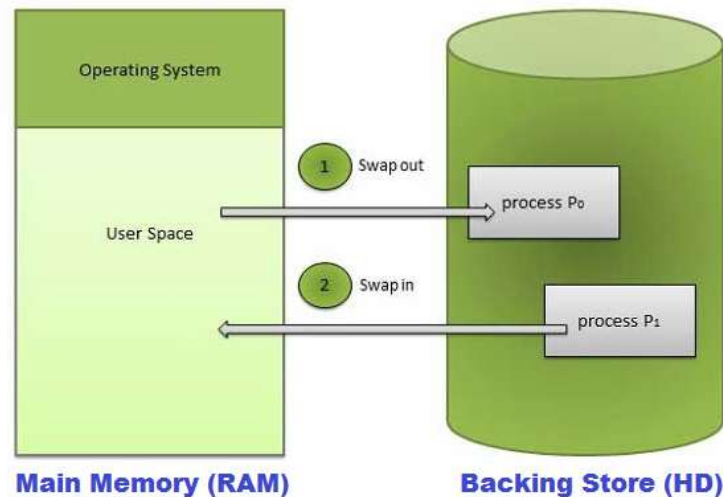


Figure 1.5 Memory Management

File Management

File management is the part of operating system that manages files and folders on storage devices such as hard disk, USB flash drive and DVD. It allows computer user to perform operations such as creating, copying, moving, renaming, deleting, and searching files and folders. It also allows the user to perform read, write, open and close operations on files and folders. Figure 1.6 shows the management of files in various folders by OS.

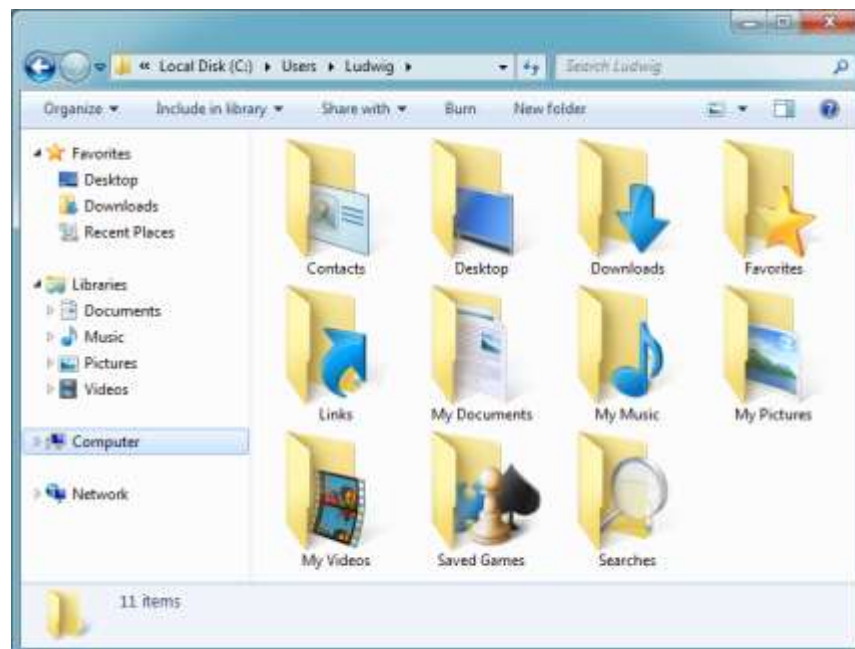


Figure 1.5 File Management



I/O Management

I/O management is the part of operating system that controls all the input/output operations during program execution. It manages all the input/output operations of input/output and storage devices. Efficient I/O management improves the performance of computer.

Example: There are three programs A, B and C which are using the printer. Now the OS will decide which program to use the printer first. A queue will be set by the OS and each program will get the printer by its turn.

Secondary Storage Management

Secondary storage management is the part of operating system that manages free space and storage allocation of user programs and data on secondary storage devices.

Example: Program 'A' is ready to be stored in Harddisk. Now OS will look for any free space in the Harddisk and assign proper address to it. If space is not available, OS will prompt the user to empty some space.

Network Management

Network management is the part of network operating system that monitors and manages the resources of a network. It allows to create user groups and assigns privileges to them. It shares the network resources among users and detects and fixes network problems.

Protection System

Protection system is the part of operating system that ensures that each resource of computer is used according to the privileges given to users by the system administrator. It creates account for each user and gives privileges to prevent misuse of the system. It provides password to all the users to maintain network security.

Command-Interpreter

Command-Interpreter is the part of operating system that provides interface between user and the computer system. It is a file in operating system that reads and executes user commands entered as text through keyboard. For example, Windows operating system uses the **cmd.exe** file as command-interpreter.

1.3 PROCESS MANAGEMENT

Process management is an important task of operating system. It allocates systems resources to various processes so that they can run efficiently.

1.3.1 PROCESS

A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates a binary code. The original code and Binary code, both are programs. When we actually run the binary code, it becomes a process. Process is a part of program under execution that is scheduled and controlled by operating system. When a program is loaded in memory for execution, it becomes a process. A program is an executable code that is stored in disk as a text file whereas a process is a dynamic instance of a program during its execution in RAM. It represents basic unit of work. It uses various resources of computer such as CPU time, files, I/O devices, memory, etc.



1.3.2 VARIOUS STATES OF A PROCESS

There are five states of a process which are new, ready, running, waiting and terminated as shown in Figure 1.5.

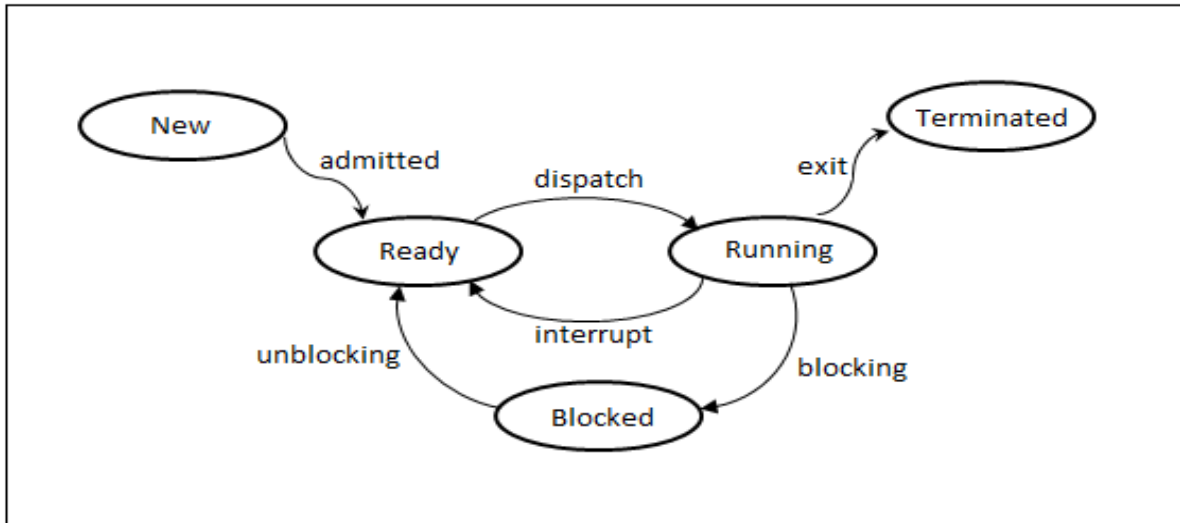


Figure 1.5 States of a Process

New State

This is the first state of a process when it is created. Any new operation or service that is requested by a program for execution by the processor is known as new state of process.

Ready State

A process is said to be in ready state when it is ready for execution but it is waiting to be assigned to the processor by the operating system.

Running State

A process is said to be in running state when it is being executed by the processor. A process is assigned to a processor for execution by operating system.

Blocked State/Waiting State

A process is in blocked or waiting state when it is not under execution. It is waiting for a resource to become available.

Terminated State

A process is in terminated state when it completes its execution.

1.3.3 THREAD AND PROCESS

In programming, there are two basic units of execution: processes and threads. They both execute a series of instructions. A **Process** is an instance of a program that is being executed. A process may be made up of multiple threads. A **Thread** is a basic ordered sequence of instructions within a process that can be executed independently. The threads are made of and



exist within a process; every process has at least one thread. Multiple threads can also exist in a process and share resources.

Comparison between Process and Thread

	Process	Thread
1	An executing instance of a program is called a process.	A thread is a subset of the process.
2	It has its own copy of the data segment of the parent process.	It has direct access to the data segment of its process.
3	Any change in the process does not affect other processes.	Any change in the thread may affect the behavior of the other threads of the process.
4	Processes run in separate memory spaces.	Threads run in shared memory spaces.
5	Process is controlled by the operating system.	Threads are controlled by programmer in a program.
6	Processes are independent.	Threads are dependent.

1.3.4 MUTLITHREADING

The process of executing multiple threads simultaneously is known as multithreading. Multithreading is an execution method of a program that allows a single process to run multiple threads at the same time. Multithreading allows multiple threads to exist within a single process and these threads can execute independently. The main purpose of multithreading is to provide simultaneous execution of two or more parts of a program to maximum utilize the CPU time.

Some examples of multithreading are:

- A user is typing a paragraph on MS word. But in background one more thread is running and checking the spelling mistakes. As soon as user is doing a typing work the other thread notifies the user about the spelling mistakes.
- Web servers use multithreading all the time, every request is handled by a different thread.

1.3.5 MULTITASKING

Multitasking is the function of operating system that loads multiple (programs, processes, tasks, threads) in main memory and executes them at the same time by rapidly switching the



Teacher Point

Teacher may also use presentations or animations or videos to explain the concepts of operating systems.



CPU among them. The operating system is able to keep track of where the users are in these tasks and go from one to the other without losing information. Each running task takes only a fair quantum of the CPU time.

1.3.6 MULTIPROGRAMMING

In multiprogramming many programs are loaded in memory but the CPU only executes one program at a time. Other programs wait until the previous program is executed out or blocked.

For example, when a user loads program 1 (say MS-Word) and program 2 (say C-language compiler). The CPU is able to execute only one program i.e. MS-Word or C-language compiler.

The advantage of multiprogramming is that it saves user's time in loading the programs to main memory and runs the programs quickly. The only drawback is, the system requires more main memory as it is occupied by many programs. Sometimes bigger programs cannot fully load in main memory and thus programs run slowly.

1.3.7 MULTIPROCESSING

Multiprocessing is the ability of an operating system to execute more than one process simultaneously on a multi-processor machine (having more than one CPUs). In this, a computer uses more than one CPU at a time.

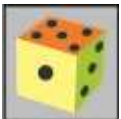


Key Points

- An Operating System (OS) is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.
- A batch processing operating system is a software that groups together same type of jobs in batches and automatically executes them one by one.
- A multiprogramming operating system is a software that loads one or more programs in main memory and executes them using a single CPU (Central Processing Unit).
- A multitasking operating system is a software that performs multiple tasks at the same time on a computer that has a single CPU.
- A time-sharing operating system is a software that shares the CPU time between multiple programs that are loaded in main memory. A time-sharing operating system gives a very short period of CPU time to each program one by one.
- A real-time operating system is a software that runs real-time applications that must process data as soon as it comes and provides immediate response.
- A multiprocessor operating system is a software that controls the operations of two or more CPUs within a single computer system. All the CPUs of computer share the same main memory and input/output devices.
- A parallel processing operating system is a software that executes programs developed in a parallel programming language. It uses many processors at the same time.



- A distributed operating system is a software that manages the operation of a distributed system. A distributed system allows execution of application software on different computers in a network.
- An embedded operating system is a built-in operating system which is embedded in the hardware of the device.
- The operating system that allows only one person to operate the computer at a time is known as single-user operating system.
- The operating system that allows many users on different terminals or microcomputers to use the resources of single central computer (server) in a network is known as multi-user operating system.
- Process management is the part of operating system that manages allocation of computer resources such as CPU to various programs in main memory.
- Memory management is the part of operating system that controls and manages the operation of main memory during the operation of computer.
- File management is the part of operating system that manages files and folders on storage devices such as hard disk, USB flash drive and DVD.
- I/O management is the part of operating system that controls all the input/output operations during program execution. It manages all the input/output operations of input/output and storage devices.
- Secondary storage management is the part of operating system that manages free space and storage allocation of user programs and data on secondary storage devices.
- Network management is the part of network operating system that monitors and manages the resources of a network.
- Protection system is the part of operating system that ensures that each resource of computer is used according to the privileges given to users by the system administrator.
- Command-Interpreter is the part of operating system that provides interface between user and the computer system. It is a file in operating system that reads and executes user commands entered as text through keyboard.
- Process is a part of program under execution that is scheduled and controlled by operating system.



Exercise

Q1. Select the best answer for the following MCQs.

- i. In which operating system, same types of jobs are grouped together and executed one by one?
 - a) Multiprogramming operating system
 - b) Batch processing operating system
 - c) Real-time operating system
 - d) Time-sharing operating system



- ii. In _____ operating system CPU is rapidly switched between programs so that all the programs are executed at the same time.
 - a) Multiprogramming operating system
 - b) Batch processing operating system
 - c) Real-time operating system
 - d) Time-sharing operating system
- iii. Which operating system runs applications with very precise timing and provides immediate response to avoid safety hazards?
 - a) Real-time operating system
 - b) Multitasking operating system
 - c) Multiprocessing operating system
 - d) Distributed operating system
- iv. _____ operating system divides a task into many subtasks and processes them independently using many processors.
 - a) Real-time operating system
 - b) Distributed operating system
 - c) Parallel processing operating system
 - d) Multitasking operating system
- v. Which operating system is used in home appliances?
 - a) Time-sharing operating system
 - b) Distributed operating system
 - c) Parallel processing operating system
 - d) Embedded operating system
- vi. Which of the following manages allocation of computer resources during program execution?
 - a) Memory management
 - b) Process management
 - c) I/O management
 - d) File management
- vii. Which of the following creates user groups and assigns privileges to them?
 - a) Process management
 - b) I/O management
 - c) File management
 - d) Network management
- viii. In which state, a process is waiting to be assigned to the processor by the operating system scheduler?
 - a) New state
 - b) Ready state
 - c) Waiting state
 - d) Running state



Q2. Write short answers of the following questions.

- i. What is the purpose of operating system in a computer?
- ii. What is Graphical User Interface (GUI)?
- iii. Mention three advantages of UNIX operating system.
- iv. Differentiate between multiprogramming and time-sharing operating systems.
- v. Why multiprocessing operating systems have been developed?
- vi. Differentiate between single-user and multiuser operating system.
- vii. Why memory management is required in a computer?
- viii. Why protection system is required in a computer?
- ix. What is a thread?
- x. Differentiate between multiprogramming and multithreading by giving one example of each.

Q3. Write long answers of the following questions.

- i. Mention the tasks performed by operating system.
- ii. Compare DOS with Windows operating system.
- iii. Describe the following types of operating systems.
 - a. Real-time operating system
 - b. Parallel processing operating system
 - c. Embedded operating system
- iv. Define the following terms.
 - a. File management
 - b. I/O management
 - c. Network management
 - d. Command-Interpreter
- v. Describe the five states of process with diagram.



Lab Activities

1. Find out which type of operating systems are installed in your computer lab.
2. Observe the installation procedure of common types of operating system through animation/video.
3. If you have visited any organization to see the working of different types of operating systems then prepare a report on it.



2

SYSTEM DEVELOPMENT LIFE CYCLE



After completing this lesson, you will be able to:

- Define a system.
- Explain System Development Life Cycle and its importance.
- Describe objectives of SDLC.
- Describe stakeholder of SDLC and their roles.
- Explain the following phases of SDLC:
 - Planning
 - Feasibility
 - Analysis
 - Requirement engineering
 - Requirement gathering
 - ❖ Functional requirements
 - ❖ Non-functional requirements
 - Requirement validation
 - Requirement management
 - Design (algorithm, flowchart and pseudo code)
 - Coding
 - Testing/verification
 - Deployment/implementation
 - Maintenance/support
- Explain the role of the following in SDLC:
 - Management in SDLC
 - Project manager
 - System analyst
 - Programmer
 - Software tester
 - Customer



2.1 SYSTEM DEVELOPMENT LIFE CYCLE

The **Systems Development Life Cycle (SDLC)**, in Software Engineering, is the process of creating or altering information systems. In other words these are the models and methodologies that experts use to develop these systems. Software engineering is an engineering approach for software development. In software engineering the SDLC concept reinforces many kinds of software development techniques. These techniques form the framework for planning and controlling the creation of an information system.

2.1.1 A System

A system is a set of components (hardware and software) for collecting, creating, storing, processing, and distributing information.

A system can be developed by applying a set of methods, procedures and routines in a proper sequence to carry out some specific task.

2.1.2 System development life cycle (SDLC) and its importance

System developing life cycle (SDLC) is a problem-solving process through which a series of steps or phases helps to produce a new computer system.

Importance of System Development Life Cycle

The basic purpose of System development life cycle is to develop a system in a systematic way in the perfect manner.

- It delivers quality software which meet the system requirements.
- It ensures that the requirements for the development of the software system are well defined and subsequently satisfied.
- It delivers cost-effective system.
- It maximizes the productivity.

2.1.3 Objectives of SDLC

A systems development lifecycle (SDLC) has three primary objectives:

- Ensure that high quality systems are delivered
- Provide strong management controls over the projects
- Maximize the productivity

These objectives are summarized as follows:



Teacher Point

Before starting the chapter, the students could be asked few questions about the term “System” to explain what they understand about it.



- One of the major objectives of SDLC is to establish an appropriate level of management authority to direct, coordinate, control, review, and approve the software development project.
- SDLC should identify the potential project risks in advance so that proper planning should be done in early.

2.1.4 Stakeholders of SDLC

Stakeholders of SDLC are those entities or groups which are either within the organization or outside of the organization that sponsor, plan, develop or use a project. Stakeholders may be users, managers and developers. It is the duty of the project management team to identify the stakeholders, determine their requirements, expectations and manage their influence in relation to the requirements to ensure a successful project.

2.1.5 SDLC PHASES/STEPS

The following are phases/steps in SDLC. These phases are shown in Figure 2.1.

- Defining Problem
- Planning
- Feasibility Study
- Analysis
- Requirement Engineering
- Design
- Coding
- Testing / Verification
- Deployment / Implementation
- Maintenance / Support

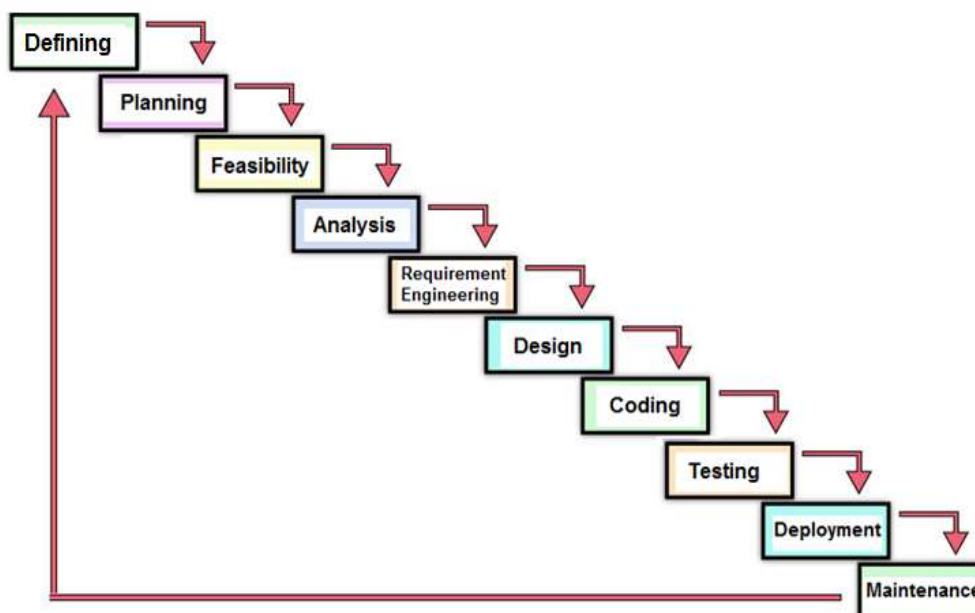


Figure 2.1: Phases of System Development Life Cycle



1. DEFINING PHASE

In this phase the problem to be solved or system to be developed is clearly defined. All the requirements are documented and approved from the customer or the company which consists of all the product requirements to be designed and developed during the development life cycle.

Example: Students' Examination System Development

Defining the problem: A Students' Examination System is needed to be developed that covers all the aspects from Examination taking to the Students' results generations.

2. PLANNING PHASE

During the planning phase, the objective of the project is determined and the requirements to produce the product are considered. An estimate of resources, such as personnel and costs, is prepared, along with a concept for the new product. All of the information is analyzed to see if there is an alternative solution to creating a new product. If there is no other viable alternative, the information is assembled into a project plan and presented to management for approval.

Example: In the **Students' Examination System Development** project planning will be made to set the ultimate goals and an estimate of resources, such as personnel and costs, is prepared.

3. FEASIBILITY

Feasibility study is used to assess the strengths and weaknesses of a proposed software/system and present directions of activities which will improve a project and achieve desired results. The nature and components of feasibility studies depend primarily on the areas in which analyzed projects are implemented.

Feasibility study is the analysis and evaluation of a proposed project/system, to determine, whether it is technically, financially/economically, legally and operationally feasible within the estimated cost and time. Feasibility study is one of the important steps in SDLC. It is divided into the following types/forms.

- Technical feasibility
- Economic feasibility
- Operational feasibility
- Legal feasibility
- Schedule feasibility

Example: The **Students' Examination System Development** project is assessed for all types of feasibilities and presented to management for final approval.



4. ANALYSIS PHASE

During the analysis phase the project team determines the end-user requirements. Often this is done with the assistance of client focus groups, which provide an explanation of their needs and what their expectations are for the new system and how it will perform.

In this phase, the in-charge of the project team must decide whether the project should go ahead with the available resources or not. Analysis is also looking at the existing system to see what and how it is doing its job. The project team asks the following questions during the analysis.

- Can the proposed software system be developed with the available resources and budget?
- Will this system significantly improve the organization?
- Does the existing system even need to be replaced etc.?

Example: The **Students' Examination System Development** project is analysed for development. The project team will visit the School/College to study the existing system and will suggest the possible improvements.

5. REQUIREMENT ENGINEERING

It is the process of determining user expectations for a new or modified system/software. Requirements engineering is a set of activities used to identify and communicate the purpose of a software system, and the framework in which it will be used. Requirement engineering consists of the following steps.

- Requirement gathering
- Requirement validation
- Requirements management

i. Requirement Gathering

Requirement gathering is usually the first part of any software/system development process. In this, meetings with the customers are arranged, the market requirements and features that are in demand are analyzed.

These requirements are of two types:

- Functional requirements
- Non-Functional Requirements

Functional requirements: Functional requirements specify the software functionality that the developers must build into the product to enable users to accomplish their tasks.

Non-functional requirements: Non-functional requirements specify criteria for the judgment of the operations of a system. It describes that how well the system performs its duties.

ii. Requirements Validation

Requirement validation is concerned with examining the requirements to certify that they meet the intentions of the stakeholders. The validation differs from verification in the sense that



verification occurs after requirements have been accepted. In requirements validation, the requirements elicited are reviewed to check that requirements are complete and accurate.

iii. Requirements Management

Requirements management is performed to ensure that the software continues to meet the expectations of the acquirer and users. Requirements management needs to gather new requirements that arise from changing expectations, new regulations, or other sources of change.

Example: In the **Students' Examination System Development** project, the development team will gather the necessary information by 'Interviewing the users', 'Giving questionnaire' or 'by reading existing documents'.

6. DESIGN PHASE

The design phase is the “architectural” phase of system design. The flow of data processing is developed into charts, and the project team determines the most logical design and structure for data flow and storage. For the user interface, the project team designs mock-up screen layouts that the developers use to write the code for the actual interface.

The design phase normally consists of two different structures. These are:

- Algorithms
- Flow chart

i. Algorithms

An algorithm is a specific step by step procedure for carrying out the solution of a problem.

Example: In the **Students' Examination System Development** project the following algorithm will find the result of a student on percentage marks.

1. **Start**
2. **Read Marks**
3. **If Marks ≥ 40 Then Print “ Pass” Else Print “Fail”**
4. **End**

ii. Flowcharts

A flowchart is a type of diagram that represents an algorithm or a process. It shows the steps of the algorithms with the help of boxes and their order by arrows connecting the boxes. This diagrammatic representation of the algorithm gives a step-by-step solution to a given problem. The operations are represented in these boxes, and the flow of control on the arrows. These flowcharts are used for analyzing, designing, documenting or managing a process or program in various fields.



Some most commonly used flowchart symbols are shown in Figure 2.3.



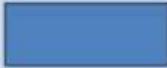



Symbol	Representations
	Terminal: Symbol for "Start "and "End" of algorithm
	Input and Output operation
	Process : indicate process such as mathematical computation, variable assignment
	Flow line: connect one symbol to other symbol
	Diamond: represent decision structure
	Connector: joining two kinds of algorithm (flow)

Figure 2.2 Flowchart Symbols

Example: In the **Students' Examination System Development** project, the Flowchart for the above algorithm will be as follows.

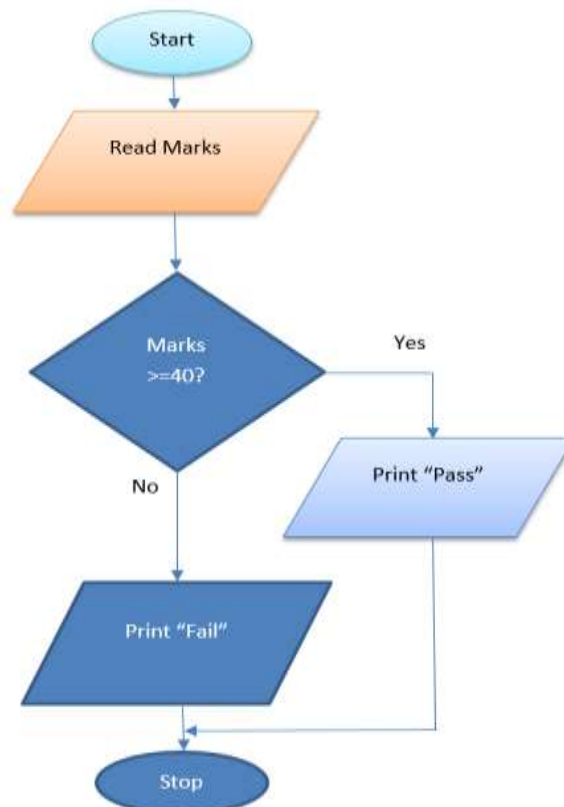


Figure 2.3 Flowchart



7. CONSTRUCTION/CODING

During the construction phase developers execute the plans laid out in the design phase. The developers design the database, generate the code for the data flow process and design the actual user interface screens. During the construction phase, test data is prepared and processed as many times as necessary to refine the code. This code is written in programming languages. Coding is also called computer programming.

Example: In the **Students' Examination System Development** project, the coding style of the C++ programming language is shown as follows.

```
/* program to display result s of the students*/
#include <iostream.h>
#include<conio.h>
int main ()
{
float Percentage;
cout<< "Enter Percentage of the student"<<endl;
cin>> Percentage;
if (Percentage >=40.0)
cout<< "Pass";
else
cout<< "Fail";
getch();
return 0;
}
```

8. TESTING/VERIFICATION

During the test phase all aspects of the system are tested for functionality and performance. The execution of a programming modules to find errors is called testing. Here, the bugs are identified in the programmed modules. The purpose of testing is to evaluate an attribute or capability of a program or system and determine that whether it meets its required results. Testing/verification the software is actually operating the software under controlled conditions. It is the process of checking the items for consistency by evaluating the results against pre-specified requirements.



Example: In the **Students' Examination System Development** project the above programming module is tested for errors. The result of the program module is given as follows.

Test 1:

Enter Percentage of the student

70.0

Pass

Test 2:

Enter Percentage of the student

20.0

Fail

9. DEPLOYMENT / IMPLEMENTATION

Software deployment is a set of activities that are used to make the software/system available for use. The deployment is also called implementation.

The main activities that are involved during deployment/implementation phase are:

- Installation and activation of the hardware and software.
- In some cases the users and the computer operation personals are trained on the developed software system.
- Conversion: The process of changing from the old system to the new one is called conversion.

Deployment/Implementation Methods

The following four methods/techniques are used for system deployment/implementation. (Figure 2.4)

- Direct Implementation:** This method involves the old system being completely dropped and the new system being completely implemented at the same time. The old system is no longer available.
- Parallel:** The parallel method of implementation involves operating both systems together for a period. This allows any major problems with the new system to be encountered and corrected without the loss of data.
- Phased:** The phased method of implementing from an old system to a new system involves a gradual introduction of the new system. The old system is progressively discarded.



- iv. **Pilot:** With the Pilot method of implementation, the new system is installed for a small number of users. These users learn, use and evaluate the new system. Once the new system is deemed to be performing satisfactorily then the system is installed and used by all.

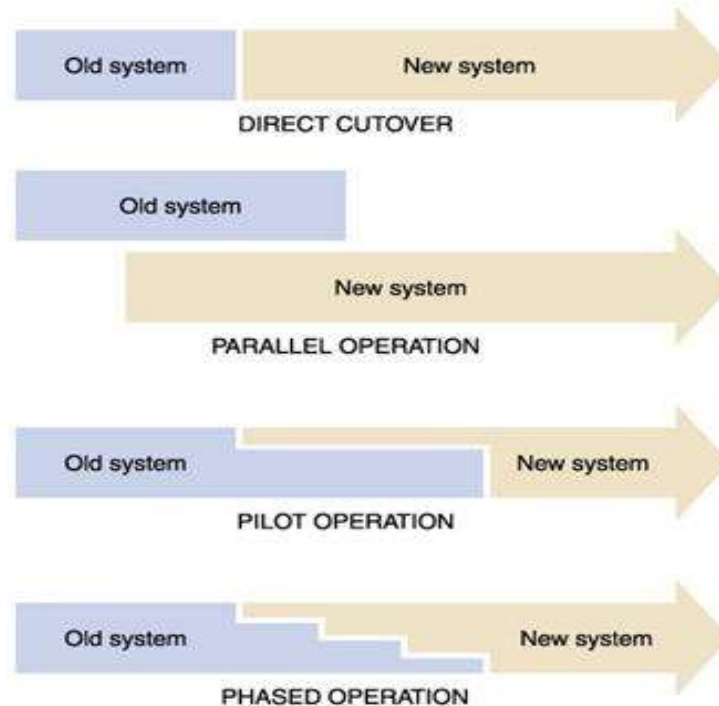


Figure 2.4 Deployment/Implementation

10. MAINTENANCE / SUPPORT

In SDLC, the system maintenance is an ongoing process. The system is monitored continually for performance in accordance with user requirements and needed system modifications are incorporated. When modifications are identified, the system may reenter the planning phase. This process continues until a complete solution is provided to the customer. Maintenance can be either be repairing or modification or some enhancement in the existing system.

2.1.6 Personnel involved in SDLC and their Role

SDLC activities are performed by different groups of people and individuals called personnel. These personnel are professionals in performing their particular jobs. These include:

- Management Personnel
- Project Manager



Teacher Point

Take one system, (other than the computer system), and explain how all the steps of SDLC could be applied to develop this system.



- System Analyst
- Programmer
- Software Tester
- Customer

a. Management Personnel/Team

A strong management team has the ability to satisfy the customers and acquirers of the software system. Also, a proposed project or a product will only meet its objectives if managed properly, otherwise, will result in failure.

The roles of a good management team is to:

- i. provide consistency of success of the software with regard to Time, Cost, and Quality objectives.
- ii. ensure that customer expectations are met.
- iii. collect historical information and data for future use.
- iv. provide a method of thought for ensuring all requirements are addressed through a comprehensive work definition process.
- v. reduce risks associated with the project.

b. Project Manager

A project manager is a professional responsible for planning, execution, and closing of any project. Apart from management skills, a software project manager will typically have an extensive background in software development. He/She is also expected to be familiar with the whole software development life cycle process. The key roles of a project manager are:

- i. Developing the project plan
- ii. Managing the project budget
- iii. Managing the project stakeholders
- iv. Managing the project team
- v. Managing the project risk
- vi. Managing the project schedule
- vii. Managing the project conflicts



Teacher Point

Teacher may also use presentations or animations or videos to explain the steps of SDLC. Take any computer system as example.



c. System Analyst

A systems analyst is a professional in the field of software development that studies the problems, plans solutions for them, recommends software systems, and coordinates development to meet business or other requirements. System analyst has expertise in a variety of programming languages, operating systems, and computer hardware platforms. The general roles and responsibilities of an analyst are defined below.

- i. Plan a system flow.
- ii. Interact with customers to learn and document requirements that are then used to produce business requirements documents.
- iii. Define technical requirements.
- iv. Interact with designers to understand software limitations.
- v. Help programmers during system development phase.
- vi. Manage system testing.
- vii. Document requirements and contribute to user manuals.

d. Programmer

A programmer is a technical person that writes computer programs in computer programming languages to develop software. A programmer writes, tests, debugs, and maintains the detailed instructions that are executed by the computer to perform their functions. The responsibilities of a programmer includes:

- i. Writing, testing, and maintaining the instructions of computer programs.
- ii. Updating, modifying and expanding existing programs.
- iii. Testing the code by running to ensure its correctness.
- iv. Preparing graphs, tables and analytical data displays which show the progress of a computer program.

e. Software Tester

A software tester is a computer programmer having specialty in testing the computer programs using different testing techniques. Software tester is responsible for understanding requirements, creating test scenarios, test scripts, preparing test data, executing test scripts and reporting defects and reporting results.



Teacher Point

Teacher should give some home assignments to the students at the end of the chapter.



f. Customer

A Customer is an individual or an organization that is a current or potential buyer or user of the software product. Customers usually purchase software from software manufacturer companies (software houses), users groups and individuals. Customers are also called clients but the only difference between the two is that the customers purchase the software products and the clients purchase services. Customers are the real evaluators of a software product by using it and identifying its merits and demerits.



Key Points

- The Systems Development Life Cycle (**SDLC**), is the process of creating or altering information systems, and the models and methodologies that experts use to develop these systems.
- The term '**System**' is a Greek word meaning to "place together." It can be defined as a set of interrelated components having a clearly defined boundary that work together to achieve a common set of objectives.
- The basic purpose of System development life cycle is to develop a system in a systematic way in the perfect manner.
- A systems development life cycle (SDLC) has three primary objectives: ensure that high quality systems are delivered, provide strong management controls over the projects, and maximize the productivity.
- **Stakeholders** of SDLC are those entities or groups which are either within the organization or outside of the organization that sponsor, plan, develop or use a project.
- The **planning phase** of SDLC determines the objective of the project and considers the requirements to produce the product.
- **Feasibility study**, in SDLC, is used to assess the strengths and weaknesses of a proposed project and present directions of activities which will improve a project and achieve desired results.
- During the **analysis phase** of SDLC the project team determines the end-user requirements.
- In SDLC, **requirements engineering**, also called requirements analysis is the process of determining user expectations for a new or modified system/software.
- The **design phase**, in SDLC, is the "architectural" phase of system design. The flow of data processing is developed into charts, and the project team determines the most logical design and structure for data flow and storage.



- An **algorithm** is a specific set of instructions for carrying out a procedure or solving a problem.
- A **flowchart** is a type of diagram that represents an algorithm or a process.
- During the **construction phase** of SDLC, developers execute the plans laid out in the design phase.
- The **test phase** of SDLC is used to test the functionality and performance of the system/program.
- **Software deployment**, in SDLC, is a set of activities that are used to make the software system available for use.
- In SDLC, keeping a system in its proper working condition is called **maintenance**.
- '**Management**' is the organization, coordination and controlling the activities of a software development by the managers and executives in accordance with certain standard procedures.
- A **project manager** is a professional responsible for planning, execution, and closing of any project.
- A **systems analyst** is a professional in the field of software development that studies the problems, plans solutions for them, recommends software systems, and coordinates development to meet business or other requirements.
- A **programmer** is a technical person that writes computer programs in computer programming languages to develop software.
- A **software tester** is a computer programmer having specialty in testing the computer programs using different testing techniques.
- A **customer** is an individual or an organization that is a current or potential buyer or user of the software product.



Exercise

Q1. Select the best answer for the following MCQs.

- i. The first step in the system development life cycle is:
 - a) Analysis
 - b) Design
 - c) Problem Identification
 - d) Development and Documentation
- ii. The organized process or set of steps that needs to be followed to develop an information system is known as:
 - a) Analytical cycle
 - b) Design cycle
 - c) Program specifications
 - d) System development life cycle
- iii. Enhancements, upgradation and bugs fixation are done during the _____ step in the SDLC.
 - a) Maintenance and Evaluation
 - b) Problem Identification
 - c) Design
 - d) Development and Documentation
- iv. The _____ determines whether the project should go forward.
 - a) Feasibility
 - b) Problem identification
 - c) System evaluation
 - d) Program specification
- v. _____ spend most of their time in the beginning stages of the SDLC, talking with end-users, gathering requirements, documenting systems and proposing solutions.
 - a) Project managers
 - b) System analysts
 - c) Network engineers
 - d) Database administrators
- vi. The entities having a positive or negative influence in the project completion are known as _____.
 - a) Stakeholders
 - b) Stake supervisors
 - c) Stake owners
 - d) None of the above
- vii. System maintenance is performed in response to _____.
 - a) Business changes
 - b) Hardware and software changes
 - c) All of the above
 - d) User's requests for additional features

Q2. Write short answers of the following questions.

- i. What is a system?
- ii. Name different phases of SDLC.
- iii. What are the objectives of SDLC?
- iv. Give some activities of planning phase.
- v. Differentiate between functional and non-functional requirements.



- vi. Design phase is considered as the “architectural” phase of SDLC. Give reasons.
- vii. Explain flowchart symbols.
- viii. What is the purpose of Testing/verification phase of SDLC?
- ix. Give main activities of the Implementation phase.

Q3. Write long answers of the following questions.

- i. Define software development life cycle (SDLC). What are its objectives?
- ii. What is system? Where exactly the Testing activities begin in SDLC?
- iii. Why software development life cycle is important for the development of software?
- iv. Who are stakeholders of SDLC? Describe their responsibilities.
- v. What is feasibility study? Explain its different types.
- vi. What is the purpose of Requirement Engineering phase? Explain its various steps in detail.
- vii. Which personnel are involved in SDLC? Explain their role briefly.



Lab Activities

Prepare a chart to show all the phases of SDLC.



3

OBJECT ORIENTED PROGRAMMING IN C++



After completing this lesson, you will be able to:

- Define program
- Define header file and reserved words
- Describe the structure of C++ program
- Know the use of statement terminator
- Explain the purpose of comments and their syntax
- Explain the difference between constant and variable
- Explain the rules for specifying variable names
- Know data types used in C++
- Define constant qualifier – const
- Explain the process of declaring and initializing variables
- Use type casting
- Explain the use of cout statement
- Explain the use of cin statement
- Define getch(), gets() and puts() functions
- Define escape sequence
- Use of escape sequence in programs
- Use I/O handling functions
- Use manipulators endl and setw
- Define operators and know their use in programs
- Use unary, binary and ternary operators in programs
- Define expression
- Define and explain the order of precedence of operators
- Define and explain compound expression



3.1 INTRODUCTION

C++ is a general purpose programming language that supports various computer programming models such as object-oriented programming (oops) and generic programming. It was created by Bjarne Stroustrup in early 1980s at Bell Laboratories. Its main purpose was to make writing good programs easier and more pleasant for the individual programmer.

C++ supports modern programming techniques. It is commonly used for developing high performance commercial software, games and graphics related programs.

3.1.1 COMPUTER PROGRAM

A computer program is a set of instructions that performs a specific task when executed by a computer. It tells the computer what to do. Everything a computer does is controlled by computer program. For example, Microsoft Word is a program that allows computer users to create documents and Skype is a program used to make calls free of charge to other people who are on Skype. Generally, programs are written in English oriented high level languages such as Visual Basic, Pascal, Java, C++, etc. A computer can only understand instruction in machine language which consists of 0s and 1s. Therefore, a program written in a high level language must be translated into machine language before execution. This task is achieved by software known as compiler. A program written in a high language is called source code and its equivalent program in machine language is called object code.

3.1.2 HEADER FILE AND RESERVED WORDS

Header File

The C++ contains many header files. Header files contain information that is required by the program in which these are used. It has *.h* extension. Some examples of header files are *iostream.h*, *conio.h* and *math.h*. These files are included in the standard library of C++ compiler.

Preprocessor directive is used to include a header file at the beginning of program. Preprocessor directive is not a normal program instruction to be executed by the CPU. It is a code for the compiler to include a header file.

The syntax of preprocessor directive to include a header file in a program is:

```
# include <name of header file>
```

It begins with a *#* sign, followed by the word *include* and then the name of header file is written within angled brackets.

For example, to include the header file *iostream.h* in a program, the code is

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



Teacher Point

Before starting the chapter, the students could be asked few questions about the “Programming Language” to explain what they understand about it.



The effect of this line is to copy all the contents of *iostream.h* header file into the program and make them available for use. Almost all the C++ programs perform input/output operations. Therefore, generally this header file is included in the programs.

The following is another example for including *math.h* header file in a program.

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

The *math.h* header file contains code for performing mathematical functions such as finding the square root of a number.

Reserved Words

Reserved words are special words which are reserved by a programming language for specific purpose in program. These cannot be used as a variable names. Some examples of reserved words of C++ are *if*, *void*, *break*, *while*, *case* and *char*. All the reserved words are written in lower-case letters. There are about 80 reserved words in C++ but this may vary depending on the version being used.

The reserved words of C++ are:

and	and_eq	asm	auto	bitand
bitor	bool	break	case	catch
char	class	const	const_cast	continue
default	delete	do	double	dynamic_cast
else	enum	explicit	export	extern
false	float	for	friend	goto
if	inline	int	long	mutable
namespace	new	not	not_eq	operator
or	or_eq	private	protected	public
register	reinterpret_cast	return	short	signed
sizeof	static	static_cast	struct	switch
template	this	throw	true	try
typedef	typeid	typename	union	unsigned
using	virtual	void	volatile	wchar_t
while	xor	xor_eq		

3.1.3 STRUCTURE OF C++ PROGRAM

A C++ program has the following structure.

preprocessor directives



Teacher Point

Teacher should recommend a suitable C++ compiler to the students. All the parts of the compiler editor should also be explained.



```
void main()
{
    body of main() function
}
```

A C++ program starts with preprocessor directives, followed by the line ***void main()*** function and then the body of the *main()* function is written within curly brackets ({ and }). Body of *main()* function consists of executable statements. These statements perform a specific task when the program is executed by the CPU. There is no restriction on the number of statement that can be written in the body of *main()* function.

To understand the structure of program, consider the following program.

```
#include <iostream.h>
void main()
{
    cout<<"Information Technology";
}
```

Explanation:

#include<iostream.h>

This is the first line of the program and it is a preprocessor directive. This program prints a message on the screen. Therefore, *iostream.h* header file is included which contains the code for performing input/output operations.

void main()

This line identifies the main function of the C++ program. All C++ programs must have *main()* function. If a program consists of more than one function, the location of *main()* does not matter. In C++ program the *main()* function is always executed first.

cout<<"Information Technology";

This is a C++ statement. The meaning of this statement is to print the message "Information Technology" on the screen. This statement is written within the curly brackets and is called body of the *main()* function.

3.1.4 STATEMENT TERMINATOR (;)

In C++, semicolon is a statement terminator. It marks the end of a statement. All the C++ statements must end with a semicolon.

The following statement was used in the previous program, notice that it ends with a semicolon.



Teacher Point

Demonstrate how a program can be written, saved, compiled and executed in C++.



```
cout<<"Information Technology";
```

If ';' is missing the compiler will give syntax error number and also the message that ';' is missing. The error number and message may vary depending on the compiler used.

3.1.5 COMMENTS IN C++ PROGRAM

All the programming languages allow comments in programs. Comments are explanatory statements that help the reader in understanding source code. Comments can be entered at any location in the program. Comments are ignored during program execution which means they are not executable statements.

There are two types of comments in C++. These are single-line comments and multiple-line comments.

The Single-line Comments (//)

The single-line comments start with // (double slash) and continue until the end of the line.

The following program demonstrates the use of single-line comments.

```
// This is a very simple C++ program.
# include <iostream.h>
void main()
{
    cout<<"Information Technology"; // It prints a message on the screen.
}
```

The Multiple-line Comments (/* and */)

This type of comment is used for entering multiple line comments in a program. The /* is used at the beginning of comments and */ ends it.

The following program demonstrates the use of multiple-line comments.

```
/* This is my first C++ program.
   It demonstrates the
   structure of C++ program.
*/
# include <iostream.h>
void main()
{
    cout<<"Information Technology";
}
```

The /* and */ type comments can also be used to enter comments on a single line as shown below.

```
/* This is my first C++ program. */
```




3.2 C++ CONSTANTS AND VARIABLES

Constants and variables are used in programs to perform calculations of various types. Therefore, it is important to understand how they are used in computer programs.

3.2.1 CONSTANTS AND VARIABLES

Constant

In computer programming, a constant is a value that does not change during execution of program. A constant can be a number, a character or a character string. A character string is a sequence of any number of characters.

Some examples of constants are 42, 7.25, 's' and "Computer" respectively. In C++, a single character constant is written within single quotes and a string constant within double quotes.

Variable

A variable is a name of memory location where data is stored. Variables are used in computer programs to store values of different data types. The data stored in a variable may change during program execution.

3.2.2 RULES FOR SPECIFYING VARIABLE NAMES

The following are the rules for naming a variable.

- i. The first character of a variable name must be alphabet or underscore.
- ii. The characters allowed in a variable name are:
 - Underscore (_)
 - Digits (0 to 9)
 - Upper-case letters (A to Z)
 - Lower-case letters (a to z)

An upper-case letter is considered different from a lower-case letter. For example, the variable *SUM* is different from *Sum* or *sum*. The underscore is generally used to improve readability. For example, the variable ***overtime*** may also be written as ***over_time***.

- iii. Special symbols such as \$, @, %, #, etc. are not allowed.
- iv. Blank space or comma is not allowed.
- v. Reserved words of C++ are not allowed to be used as a variable name.



Teacher Point

Teacher should also explain few more related programs according to the chapter topics.



3.2.3 C++ DATA TYPES

Data types are declarations of variables for storing various types of data. Data types have different storage capacities. In C++ programs data type of a variable must be defined before assigning it a value.

The data types used in C++ programming are integer, floating-point, double precision and character.

Integer

It is a data type that is used to define numeric variables to store whole numbers, such as, -3, 0, 367, +2081, etc. Integers represent values that are counted, such as number of students in a class. The short form of integer is *int*. Numbers that have fractional part, such as, 3.84 cannot be stored in an integer variable.

The following table shows the integer types, the number of bytes it takes in memory to store the value and the range of numbers it can store.

Integer Type	No. of Bytes	Range of Numbers
int	4 bytes	-2147483648 to 2147483647
unsigned int	4 bytes	0 to 4294967295
short int	2 bytes	-32768 to 32767
unsigned short int	2 bytes	0 to 65535
long int	4 bytes	-2147483648 to 2147483647
unsigned long int	4 bytes	0 to 4294967295

Floating-point

It is a data type that is used to define variables that can store numbers that have fractional part such as 3.75, -2.1, 388.80, etc. These numbers are also known as real numbers. The short form of floating-point is *float*.

The following table shows the floating-point types, the number of bytes it takes in memory to store the value and the range of real numbers it can store.

Float Type	No. of Bytes	Range of Numbers
float	4 bytes	-3.4^{38} to 3.4^{38} (with 6 digits of precision)
double	8 bytes	-1.7^{308} to 1.7^{308} (with 15 digits of precision)

The *float* type variables might occupy different number of bytes and their range might also be different depending on the computer and the compiler being used.

Character

It is a data type that is used to define variables that can store only a single character. One byte of memory is set aside in memory to store a single character. The short form of character is *char*. A variable of type *char* can store a lower-case letter, an upper-case letter, a digit or a special character. Some examples of characters that can be stored in a variable of type *char* are 'a', '+', '%' and '5'. Note that characters are written within single quotation marks.



3.2.4 THE CONSTANT QUALIFIER (const)

In C++ programming language, *const* defines a variable whose value cannot be changed throughout the program. When the *const* qualifier is used with a variable, it no longer remains a variable because its value will not be changed. A variable defined with the *const* qualifier must be assigned some value.

It has the following syntax.

```
const data_type variable = constant;
```

For example,

```
const int AGE = 34;
```

```
const float LENGTH = 7.5;
```

The variables of type *const* are generally written in upper-case letters.

3.2.5 VARIABLE DECLARATION AND INITIALIZATION

In C++, all the variables that are going to be used in a program must be declared before use. Declaring a variable means specifying the data type of a variable. It allows the compiler to decide how many bytes should be set aside in memory for storage of value that is going to be assigned to the variable in the program.

The following are some examples of declaring variables.

```
int x,y,z;
```

```
float length,breadth,sum;
```

```
char ch;
```

Here the variables, *x*, *y* and *z* are declared as of type *int*, *length* and *breadth* as of type *float* and *ch* as of type *char*.

A variable may be initialized at the beginning of a program when it is declared. Initializing a variable means assigning it an initial value.

The following are some examples of initializing variables in declaration statements.

```
int x=4,y=5,z;
```

```
float length=12.5,breadth=15.25,sum=0.0;
```

```
char ch='a';
```

In the first declaration statement, the variable *x* is initialized to integer constant 4 and *y* to 5. The second statement initializes *length* to 12.5, *breadth* to 15.25 and *sum* to 0.0. The last statement initialized the character variable *ch* to 'a'.

3.2.6 TYPE CASTING

Type casting is used in C++ to convert data type from one type to another. There are two types of type casting, implicit and explicit type casting.

Implicit Type Casting

Implicit type casting automatically converts a data type to another. This is explained by the following example.



Suppose variable q is declared as of type *float* and the following calculation is to be performed.

```
q = 15/6;
```

When this integer division is performed, the result will also become an integer value 2 which will be implicitly converted to floating-point value 2.0 and assigned to the variable q .

If one or both of the integer constants are converted to floating-point constant (14.0 or 6.0), this will perform division using floating-point mathematics. In this case, the result produced will be 2.5 and it will be assigned to q .

Explicit Type Casting

In explicit type casting, a special operator is used to convert one data type into another.

The general form for conversion is

(type) expression

Here, expression can be an arithmetic expression or a variable. This is explained by the following example.

Suppose, a and b are variables of type *int* and q is of type *float*. The integer value 15 is stored in a and 6 in b and the following division is to be performed.

```
q = a/b;
```

When this division is performed, integer math will be used and the result produced will be 2 which will be assigned to the variable q .

To obtain correct result, type casting should be used to convert at least one of the operands to type *float* as shown below.

```
q = (float)a/b;
```

Now, first the value stored in a will be converted to type *float* and then the division will be performed. The floating-point math will be used and the correct result 2.5 will be produced which will be assigned to q .

Similarly, the *int* type can also be used to convert a floating-point value stored in a floating-point variable into integer type by truncating the fractional part of the number.

3.3 INPUT/OUTPUT HANDLING

In computer programming, providing data into a program from outside source is known as input and output means to display some data on screen or save in a file on a storage device.

In C++, input/output (I/O) is performed by using streams. A stream is a sequence of data that flows in and out of the program. The ***cin*** and ***cout*** are the standard statements that use the *iostream.h* header file for performing I/O operations. Therefore, it must be included at the beginning of program.

The functions *getch()*, *gets()* and *puts()* are also used for handling input/output operations.



3.3.1 THE `cout` STATEMENT

The `cout` statement is used to output text or values on the screen.

The following is the syntax of `cout` statement.

`cout<<character string/variable;`

The keyword `cout` is used with insertion operator on its right side which is two less than signs (`<<`), followed by a character string or a variable. The insertion operator displays the contents of the character string or the value stored in the variable on the screen. It directs the output to the standard output device which is monitor. Note that the statement ends with semicolon. The insertion operator may be used more than once in a single statement.

The following program demonstrates the use of `cout` statement.

```
#include <iostream.h>    // this header file is used for I/O functions
void main()
{
    int a;
    a=12;
    cout<<"The value stored in a is ";
    cout<<a;
}
```

The output of the program will be

The value stored in a is 12

The first output statement will print the text and the second will print the value stored in variable `a`. The output of the two `cout` statements will appear on the same line.

3.3.2 THE `cin` STATEMENT

The `cin` statement is used to input data from the keyboard and assign it to one or more variables.

The following is the syntax of `cin` statement.

`cin>>variable;`

The keyword `cin` is used with the extraction operator on the right side which is two greater than signs (`>>`), followed by a variable. When this input statement is executed, it causes the program to wait for the user to input data. The data entered by the user is assigned to the variable.

The following program demonstrates the use of `cin` statement.

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



```
int n;
cout<<"Enter an integer:";
cin>>n;
cout<<"The number you typed is "<<n;
}
```

When this program is executed, the first *cout* statement prompts the user to enter an integer. The *cin* statement causes the typed number to be assigned to variable *n*. The last statement prints the number stored in *n* on the screen.

The execution of program is shown below.

Enter an integer:7

The number you typed is 7

More than one extraction operator can be used in a single *cin* statement to input more than one data values as shown below.

```
cin>>a>>b;
```

The values for variables *a* and *b* should be input with space between them.

3.3.3 THE *getch()*, *getche()*, *gets()* and *puts()* FUNCTIONS

These three functions are also used for handling I/O operations.

The *getch()* Function

In some situations in programming, it is required to read a single character the instant it is typed without waiting for Enter key to be pressed. For example, in a game we might want an object to move each time we press one of the arrow keys. It would be awkward to press the Enter key each time we pressed an arrow key.

The *getch()* function is used for this purpose. The *get* means it gets something from an input device and *ch* means it gets a character. This function uses the *conio.h* header file. Therefore, it must be included in the program.

The following program demonstrates the use of *getch()* function to read a character from the keyboard and displays it on the screen.

```
#include<iostream.h>
#include<conio.h>    // this header file is used for console (screen) input/output
{
    char ch;
    cout<<"Enter a character:";
    ch=getch;
    cout<<"The character you typed is "<<ch;
}
```

The execution of the program is shown below with the input character *a*.

Enter a character:



The character you typed is a

Note that, when this function is used, the input character *a* is read and stored in variable *ch* but it is not displayed on the screen.

The *getche()* Function

If the user wants the typed character to be displayed on the screen then another similar function *getche()* is to be used. In this function the letter *e* is added which means to echo or display the input character on the screen.

When executing a program, some C++ compilers display the output for a very short time and immediately returns to the editing window. The user is not able to see the program output. Therefore, very often, *getch()* function is used at the end of the program so that the user is able to see the program output and has to enter any character to return to the editing window. Since the character entered is not used in program, there is no need to store it in a variable. This is shown below.

```
#include<iostream.h>
#include<conio.h>
{
    char ch;
    cout<<"Enter a character:";
    ch=getche();
    cout<<"The character you typed is "<<ch;
    getch();
}
```

The execution of the program is shown below with the input character 'h'.

Enter a character:

The character you typed is h

Note that, when this function is used, the input character 'h' is read and stored in variable *ch* and displayed on the screen.

The *gets()* and *puts()* Functions

These functions are used to handle input/output of character strings. The *gets()* function is used to input a string from the keyboard and the *puts()* is used to display it on the screen. In C++ strings are handled as arrays. These functions will be explained in Chapter 5 in which arrays are discussed.

3.3.4 THE ESCAPE SEQUENCE

Escape sequences are special characters used to control the output look on output devices. These characters are not printed. These are used inside the output statement.

An escape sequence begins with a backslash (\) followed by a code character. These are called escape sequences because the backslash causes an 'escape' from the normal way characters are interpreted in C++ programming language. Escape sequences are used for



special purposes in programming such as to begin printing on the next line, to issue a tab, to print special characters, etc.

3.3.5 COMMONLY USED ESCAPE SEQUENCES

The commonly used escape sequences are `\a`, `\b`, `\n`, `\r`, `\t`, `\\`, `\'` and `\"`.

Suppose we want to print the following message on the screen.

```
cout<< "There are many versions of";
cout<< "Windows operating system.";
```

When the above two statements are executed, the output will appear on a single line as shown below.

There are many versions of Windows operating system.

If it is desired to display the output in two lines then `\n` escape sequence can be used in various ways to move cursor to the beginning of next line.

```
cout<< "\nThere are many versions of";
cout<< "\nWindows operating system.";
```

The same output can also be obtained by the following two statements.

```
cout<< " \nThere are many versions of\n";
cout<< "Windows operating system.";
```

A single output statement can also be used.

```
cout<< "\nThere are many versions of\nWindows operating system.";
```

Similarly, the `\t` escape sequence can also be used to tab over eight characters as shown in the following statement.

```
cout<< "C++\tis\ta\thigh\tleve\tlanguage";
```

The output of this statement will be

C++ is a high level language

A list of commonly used escape sequences is given in the following table with their meanings.

Escape Sequence	Meaning
<code>\a</code>	Produces alert (beep) sound
<code>\b</code>	Moves cursor backward by one position
<code>\n</code>	Moves cursor to the beginning of next line
<code>\r</code>	Moves cursor to the beginning of current line
<code>\t</code>	Moves cursor to the next horizontal tabular position
<code>\\</code>	Produces a backslash
<code>\'</code>	Produces a single quote
<code>\"</code>	Produces a double quote



The following statement uses the `\` escape sequence to print the word “Pakistan” in double quotation marks.

```
cout<<“This will print the word \“Pakistan\” in double quotation marks.”;
```

The escape sequences `\` and `\\` are also used in the same way to print a single quote or a backslash.

3.3.6 PROGRAMMING WITH I/O HANDLING FUNCTIONS

Program 1: The following program reads three integers and prints their sum and product.

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int x,y,z,sum,prod;
    cout<<“\nEnter first number:”;
    cin>>x;
    cout<<“\nEnter second number:”;
    cin>>y;
    cout<<“\nEnter third number:”;
    cin>>z;
    sum=x+y+z;
    prod=x*y*z;
    cout<<“\nSum=”<<sum;
    cout<<“\nProduct=”<<prod;
    getch();
}
```

The execution of the program is shown below.

```
Enter first number:3
Enter second number:4
Enter third number:5
Sum=12
Product=60
```

When this program is executed, it prompts the user to enter three numbers. The user enters the numbers, 3,4 and 5 separated by space and presses the Enter key. The program calculates the sum and product and displays on the screen.

Program 2: The following program reads the base and height of a triangle and prints area using floating-point values.



```
#include<iostream.h>
#include<conio.h>
void main()
{
    float base,height,area;
    cout<< "\nEnter the base:";
    cin>>base;
    cout<< "\nEnter the height:";
    cin>>height;
    area=(base*height)/2;
    cout<< "\nThe area of triangle is "<<area;
    getch();
}
```

In this program, two separate input statements are used for reading the values for variables *base* and *height*. The execution of program is shown is below.

Enter the base:4.5;

Enter the height 6.2;

The area of triangle is 13.95

3.3.7 THE MANIPULATORS *endl* AND *setw*

A manipulator is a command in C++ that is used for formatted output. It modifies the output in various ways. There are many manipulators available in C++. The most commonly used manipulators are ***endl*** and ***setw***. The *iomanip.h* header file should be included when manipulators are used in a program. Only the *endl* manipulator can be used without using *iomanip.h* file.

The *endl* Manipulator

The *endl* manipulator has the same function as the *\n* escape sequence. It causes a linefeed in the *cout* statement so that the subsequent text is displayed on the next line.

The following program demonstrates the use of *endl* manipulator.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
    cout<< "\nI am a student."<<endl;
    cout<< "I was born in 2001.";
```



```
    getch();  
}
```

The output of the program will be:

I am a student.

I was born in 2001.

A single *cout* statement can also be used as shown below to obtain the same output.

```
cout<< "I am a student."<<endl<< "I was born in 2001.";
```

The *setw* Manipulator

The *setw* manipulator is used in output statement to set the minimum field width.

It has the general form:

setw(n)

Here, *n* is an integer value that causes the number or text that follows to be printed within a field width of *n* characters. The number or text is right justified within the set field width. It is commonly used in programs to align numbers or text on output.

The following program demonstrates the use of *setw* manipulator in a program.

```
#include<iostream.h>  
#include<conio.h>  
#include<iomanip.h>  
void main()  
{  
    int price1=8540,price2=325,price3=27800;  
    cout<< "Product      "<<setw(10)<< "Price"<<endl;  
    cout<< "Hard disk    "<<setw(10)<<price1<<endl;  
    cout<< "Mouse        "<<setw(10)<<price2<<endl;  
    cout<< "Computer     "<<setw(10)<<price3<<endl;  
    getch();  
}
```

This program will print the values following the *setw* manipulator within a field width of 10 characters and they will be right justified.

The output of the program is given below.

Product	Price
Hard disk	8540
Mouse	325
Computer	27800



The same program can also be written using a single `cout` statement as shown in the next program.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
    int price1=8540,price2=325,price3=27800;
    cout<< "Product      "<<setw(10)<<"Price"<<endl
         << "Hard disk    "<<setw(10)<<price1<<endl
         << "Mouse        "<<setw(10)<<price2<<endl
         << "Computer     "<<setw(10)<<price3<<endl;
    getch();
}
```

If `setw` manipulator is not used, the output will be:

Product	Price
Hard disk	8540
Mouse	325
Computer	27800

The commonly used header files are listed in the following table with their purpose.

Header file	Purpose
iostream.h	Provides basic input/output operations such as <code>cin</code> and <code>cout</code> .
conio.h	Stands for console input/output. It manages input/output on console based applications. Console applications take input from keyboard and display output on monitor.
math.h	Provides basic mathematical operations such as <code>sqrt()</code> , <code>pow()</code> , etc.
string.h	Provides several functions to manipulate strings such as <code>strcmp</code> (compare string), <code>strcpy</code> (copy string), etc.
iomanip.h	Provides manipulator function such as <code>setw()</code> , <code>setprecision()</code> , etc.
time.h	Provides date and time operations



3.4 OPERATORS IN C++

An operator is a symbol that tells the computer to perform a specific computing task. For example the '+' operator is used to add two numbers.

3.4.1 TYPES OF OPERATORS

The following types of operators are commonly used in C++.

- Assignment operators
- Arithmetic operators
- Arithmetic assignment operators
- Increment/Decrement operators
- Relational operators
- Logical operators
- Ternary operator

Assignment Operator

Assignment operator is equal sign (=). It is used to assign value of an expression to a variable. It has the general form

variable = expression;

where expression may be a constant, another variable to which a value has previously been assigned or a formula to be evaluated. For example:

z == x + y;

When this statement is executed, the values stored in variables x and y will be added and the resulting answer will be stored in variable z.

Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations that include addition, subtraction, multiplication, division and to find the remainder of integer division.

The types of arithmetic operators used in C programming are described with their operations in the table given below.

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo Operator

The modulo operator (%) gives the remainder after division of one number by another. For example,

a = 20 % 6;



will give the result 2 which will be assigned to the variable *a* because 6 will divide 20 by 3 with a remainder of 2.

Some more examples of modulo operator are given below.

13 % 4 will give the result 1
 15 % 4 will give the result 3
 10 % 5 will give the result 0
 4 % 5 will give the result 4

Arithmetic Assignment Operators

In addition to equal (=) assignment operator, there are a number of assignment operators unique to C++ which are known as arithmetic assignment operators. These include +=, -=, *=, /= and %=.

Suppose *op* represents an arithmetic operator. Then, the arithmetic assignment operator has the following general form to assign value of an expression to a variable.

variable op = expression;

For example:

a += *b*;

It is equivalent to:

a = *a* + *b*;

The effect is exactly the same but the expression is more compact. The arithmetic assignment operators are described in the following table.

Operator	Operation	Example
+=	Adds the right side operand to the left side operand and assigns the result to the left side operand	<i>a</i> += <i>b</i> It is equivalent to <i>a</i> = <i>a</i> + <i>b</i>
-=	Subtracts the right side operand from the left side operand and assigns the result to the left side operand	<i>k</i> -= <i>n</i> It is equivalent to <i>k</i> = <i>k</i> - <i>n</i>
*=	Multiplies the right side operand with the left side operand and assigns the result to the left side operand	<i>prod</i> *= <i>n</i> It is equivalent to <i>prod</i> = <i>prod</i> * <i>n</i>
/=	Divides the left side operand by the right side operand and assigns the result to the left side operand	<i>n</i> /= <i>m</i> It is equivalent to <i>n</i> = <i>n</i> / <i>m</i>
%=	Takes modulus and assigns the result to the left side operand	<i>x</i> %= <i>y</i> It is equivalent to <i>x</i> = <i>x</i> % <i>y</i>

Increment and Decrement Operators

The increment operator is ++ and it is used to add one to the value stored in a variable. The decrement operator is -- and it subtracts one from the value stored in a variable. The purpose of using these operators is simply to shorten the expression.

**Examples:**

`++x` and `x++` are both equivalent to `x = x + 1`

`--x` and `x--` are both equivalent to `x = x - 1`

When increment or decrement operator is written before the variable, it is known as *prefix* and when it is written after the variable, it is known as *postfix*.

In certain situations, `++x` and `x++` have different effect. This is because `++x` increments `x` before using its value whereas `x++` increments `x` after its value is used.

As an example, suppose `x` has the value 3. The statement

```
y = ++x;
```

will first increment `x` and then assigns the value 4 to `y`. But the statement

```
y = x++;
```

will first assign the value 3 to `y` and then increments `x` to 4. In both cases `x` is assigned the value 4 but `y` is assigned different value.

The same rule applies to `--x` and `x--` as well.

Example of prefix increment:

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int x,y;
    y=10;
    x=++y;
    cout<<"x: "<<x;
    cout<<"y: "<<y;
    getch();
}
```

Output:

x: 11

y: 11

Example of postfix increment:

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



```
{
    int x,y;
    y=10;
    x=y++;
    cout<<"x: "<<x;
    cout<<"y: "<<y;
    getch();
}
```

Output:

x: 10

y: 11

Relational Operators

Relational operators are used to compare two values of the same type. These operators are very helpful in computer programming when a flow of program is based on a condition. After evaluation of a relational expression, the result produced is **True** or **False**.

Six types of relational operators are available in C language. These are described in the following table.

Operator	Definition	Example
= =	equal to	a==b
!=	not equal to	n!=m
<	less than	a<b+c
>	greater than	z>y
<=	less than or equal to	z<=(x+y)/2
>=	greater than or equal to	f>=a+12

The following are some examples of relational operators.

x > y

x == y

z > x + y

z <= x + y

x!=10

If x has the value 12 and y has the value 7, then the condition in first line will become true since 12 is greater than 7.

The condition of second expression will become false since 12 is not equal to 7.



In the third expression, if z has the value 15, then the condition will become false since 15 is not greater than the sum of x and y which is 19.

In the fourth expression, 15 is less than 19. Therefore, the condition will become true.

In the last expression if x is any number other than 10 then the expression will be true. It will only be false when x is equal to 10.

Note: In C language true is represented by the integer 1 and false is represented by the integer 0.

Logical Operators

Logical operators are used in programming when it is required to take some action based on more than one condition. When two or more conditions are combined, it is called compound condition.

There are three types of logical operators. These are described below.

Operator	Definition
&&	AND
	OR
!	NOT

- **Logical AND (&&) Operator**

It is used to form compound condition in which two relational expressions are evaluated. One relational expression is to the left and the other to the right of the operator. If both of the relational expressions (conditions) are true then the compound condition is considered true otherwise it is false.

The following is a compound condition that uses the && operator.

$(x \geq 1) \ \&\& \ (x \leq 10)$

When this compound condition is evaluated, it will produce the result true if x is greater than or equal to 1 and less than or equal to 10. In other words, the result will be true if both conditions are true that is x is in between 1 and 10 otherwise it will be false.

The following compound condition will check whether the character stored in character variable ch is a lowercase letter or not.

$(ch \geq 'a') \ \&\& \ (ch \leq 'z')$

The following truth table shows all the possible results of AND (&&) logical operator for two expressions.

Expression-1	Expression-2	Expression-1 && Expression-2
False	False	False
False	True	False
True	False	False
True	True	True



- **Logical OR (||) Operator**

Logical OR operator is also used to form a compound condition. Just like the logical AND operator, one relational expression is to the left and the other to the right of the OR operator. The compound condition is true if either of the conditions is true or both are true. It is considered false only if both of the conditions are false.

The following is an example of compound condition that uses || operator.

`(x>y) || (z==8)`

This compound condition will produce the result true, if one of the conditions is true, that is, if x is greater than y or z is equal to 8. The result will only be false when both of the conditions are false, that is, x is not greater than y and z is not equal to 8.

The following truth table shows all the possible results of OR (||) logical operator for two expressions.

Expression-1	Expression-2	Expression-1 Expression-2
False	False	False
False	True	True
True	False	True
True	True	True

Logical NOT (!) Operator

The logical NOT operator is used with a single expression (condition) and evaluates to true if the expression is false and vice versa. In other words, it reverses the result of a single expression.

For example, the expression

`!(z<10)`

will be true if z is not less than 10. In other words, the condition will be true if z is greater than or equal to 10. Some programmers may prefer to write the above expression as given below which is easy to understand and has the same effect.

`(z>=10)`

The following truth table shows all the possible results of NOT (!) logical operator for one expression.

Expression	! Expression
False	True
True	False

Ternary Operator (? :) / Conditional Operator

The ? : operator is known as ternary or conditional operator. It returns one of two values depending on the result of a condition. Therefore, it is also known as conditional operator. It is very useful in situation where the programmer needs to choose one of two options depending on a single condition.



The general form of ternary operator is

Condition? Expression1 : Expression2;

The condition is evaluated. If it is true then Expression1 is evaluated otherwise Expression2 is evaluated.

The following is an example of using ternary operator.

$(x > y) ? x + y : x - y$

When this code is executed, the condition $x > y$ is evaluated. If the result is true then the value $x + y$ is evaluated otherwise the value $x - y$ is evaluated. It allows to execute different code based on the result of condition $x > y$.

The result of ternary operator can be assigned to a variable as shown below.

$k = (x > y) ? x + y : x - y;$

This is demonstrated in the following program.

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int x,y,k;
    x=15; y=10;
    k=(x>y)? x + y: x - y;
    cout<< "The value of k is "<<k<<endl;
    getch();
}
```

The value of x is greater than y. Therefore, the condition $(x > y)$ is true and k will be assigned the sum of x and y.

The output of the program will be

The value of k is 25

The ternary operator also allows to output text as shown below.

$(x > y) ? \text{cout} << \text{"x is greater than y"} : \text{cout} << \text{"x is smaller than y"};$

This is demonstrated in the following program.

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int x,y,k;
    x=3; y=10;
    (x>y)? cout<< "x is greater than y"<<endl;
```



```

        cout<< "x is smaller than y"<<endl;
    getch();
}

```

In this program the value of x is smaller than y. Therefore, the condition (x>y) is false and the second output statement will be executed.

The output of the program will be

x is smaller than y

3.4.2 UNARY, BINARY AND TERNARY OPERATORS

There are three types of operators in C++ which are unary, binary and ternary operators.

The operator that works on a single operand is known as unary operators. Unary operators are -, ++, -- and the logical operator !.

Some examples of unary operators are

```

a= -b;
k++;
- -x;

```

The operators that work on two operands are known as binary operators. Binary operators are -, +, *, /, % and logical operators && and ||.

Some examples of binary operators are

```

a=b+c;
z=x*y;
k=d%e;

```

The conditional operator (? :) is known as ternary operator since it has three parts. These three parts are a condition and two expressions. The condition is evaluated and based on its result one of the two expressions is executed.

3.4.3 ORDER OF PRECEDENCE OF OPERATORS

Order of precedence of operators describes the rules according to which operations are to be performed in an expression.

In the following table, the operator that has the highest precedence is written at the top and the one with the lowest precedence is written at the bottom.

Precedence	Operator	Description
1.	*, /, %	Multiplication, Division and Remainder
2.	+, -	Addition and Subtraction
3.	<, <=, >, >=	Relational Operators



4.	=, !=	Equal to and Not Equal to
5.	!	Logical NOT
6.	&&	Logical AND
7.		Logical OR
8.	=, *=, /=, %=, +=, -=	Assignment Operators

For example, in the expression

$$7 + 3 * 2$$

multiplication operator has higher precedence than the addition operator and thus will be evaluated first.

Parentheses are used in expression to change the order of evaluation of operators specified by operator precedence.

For example, in the above example, if it is required to perform addition before multiplication then parentheses can be used as shown below.

$$(7 + 3) * 2$$

When two operators of same precedence occur in an expression then they are evaluated from left to right as they occur.

When an expression contains arithmetic, relational and logical operators, the arithmetic operators are evaluated first, relational operators next and logical are evaluated last. Assignment operators are always applied at the end.

3.4.4 EXPRESSIONS IN C++

An expression is a combination of constants, variables and operators. Constants and variables are operands and operators tell the computer what types of action to perform on the operands.

There are three types of expressions in C++. These are arithmetic, relational and compound or logical expression.

An expression that contains constants, variables and arithmetic operators is called arithmetic expression.

For example, in the expression

$$a + 2 * b$$

a , b and 2 are operands and $+$ and $*$ are arithmetic operators. When this expression is evaluated,



Teacher Point

Teacher should give some home assignments to the students at the end of the chapter.

the values stored in variables a and b are substituted and the result produces another value.

The following are some more examples of arithmetic expressions.

$$5*(a - b)$$



$$(a + b)/(a - b) + a*b$$

$$3*a + 8 - b + (b + c)/2$$

An expression that contains a relational operator to compare values of same type is called relational expression. Relational expressions are used in programming to create conditions based on which computer takes different path during program execution.

An expression that combines two or more conditions using logical operators, && or || is called compound expression.

The following is an example of compound expression.

```
((ch>='a')&&(ch<='z'))||((ch>='A')&&(ch<='Z'))
```

The first compound condition ((ch>='a')&&(ch<='z')) checks whether the character stored in variable ch is a lower-case letter or not and the second compound condition ((ch>='A')&&(ch<='Z')) checks whether it is an upper-case letter or not. If one of the two compound conditions is true, the compound expression will be true because the two compound conditions are combined with an || (OR) operator.

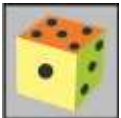


Key Points

- A computer program is a set of instructions to perform a specific task.
- Header files contain information that is required by the program in which they are used. It has .h extension.
- Reserved words are special words which are reserved by a programming language for specific purpose in program.
- In C++, semicolon is a statement terminator. It marks the end of a statement. All the C++ statements must end with a semicolon.
- Comments are explanatory statements that help the reader in understanding source code.
- A constant is a value that does not change during execution of program.
- A variable is a name of memory location where data is stored. Variables are used in computer programs to store values of different data types.
- Escape sequences are special characters used to control printing on the output device. These characters are not printed. These are used inside the output statement.
- Order of precedence of operators describes the rules according to which operations are to be performed in an expression.
- An operator is a symbol that tells the computer to perform a specific computing task.
- Assignment operator is equal sign (=). It is used to assign value of an expression to a variable.
- Arithmetic operators are used to perform arithmetic operations that include addition, subtraction, multiplication, division and to find the remainder of integer division.



- Relational operators are used to compare two values of the same type.
- Logical operators are used in programming when it is required to take some action based on more than one condition.
- An expression is a combination of constants, variables and operators. Constants and variables are operands and operators tell the computer what types of action to perform on the operands.
- An expression that combines two or more conditions using logical operators, && or || is called compound expression.



Exercise

Q1. Select the best answer for the following MCQs.

- Which of the following is ignored during program execution?
 - Reserved word
 - Constant
 - Comment
 - const qualifier
- What is the range of unsigned short integer?
 - 2147483648 to 2147483647
 - 0 to 4294967295
 - 32768 to 32767
 - 0 to 65535
- In C++ the expression `sum=sum+n` can also be written as:
 - `sum+=n`
 - `sum=n++`
 - `sum=+n`
 - `n+=sum`
- Which of the following is equal to operator?
 - `+=`
 - `==`
 - `=`
 - `==+`
- Which of the following is an arithmetic operator?
 - `&&`
 - `%`
 - `<=`
 - `++`
- Which of the following operators is used to form compound condition?
 - Arithmetic operator
 - Assignment operator
 - Relational operator
 - Logical operator
- The number of bytes reserved for a variable of data type 'float' is:
 - 2
 - 4
 - 6
 - 8
- How cursor is moved to the next tabular position for printing data?
 - By using reserved word
 - By using manipulator
 - By using escape sequence
 - By using header file

Q2. Write short answers of the following questions.

- Define reserved words and give three examples.



- ii. What is the purpose of using header file in program?
- iii. State whether the following variable names are valid or invalid. State the reason for invalid variable names.

a) a123	b) _abcd	c) 5hml	d) tot.al
e) f3ss1	f) c\$avg	g) net_weight	h) cout
- iv. Why escape sequence is used? Give three examples with explanation.
- v. Differentiate between relational and logical operators.
- vi. What will be the output of the following statements?
 - a) `cout<< "17/3 is equal to "<<17/2;`
 - b) `cout<< "10.0/4 is equal to "<<10.2/4;`
 - c) `cout<< "40/5%3*7 is equal to "<<(40/5%3*7);`
- vii. Evaluate the following integer expressions.

a) $3+4*5$	b) $4*5/10+8$	c) $3*(2+7*4)$
d) $20-2/6+3$	e) $(20-2)/(6+3)$	f) $25\%7$
- viii. Evaluate the following expressions that have integer and floating-point data types.

a) $10.0+15/2+4.3$	b) $10.0+15.0/2+4.3$	c) $4/6*3.0 +6$
--------------------	----------------------	-----------------
- ix. What will be the output of the following program?

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int num1,num2,total;
    num1=13;
    num2=20;
    total=num1+num2;
    cout<< "The total of "<<num1<< " and "
        <<num2<< " is "<<total<<endl;
    getch();
}
```

- x. What will be the output of the following program?

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



```
n=10;
cout<< "The initial value of n is "<<n<<endl;
n++;
cout<< "The value of n is now "<<n<<endl;
n++; n++;
cout<< "The value of n is now "<<n<<endl;
n--;
cout<< "The value of n is now "<<n<<endl;
getch();
}
```

Q3. Write long answers of the following questions.

- Define variable and write the rules for specifying variable names.
- Why type casting is used? Explain the types of type casting with an example of each type.
- Define *endl* and *setw* manipulators and give an example of each.
- What is meant by precedence of operators? Write the operators with the highest precedence at the top and the lowest at the bottom.



Lab Activities

- Practice all the Example programs given in the chapter.
- Write a program that reads four integers and prints their sum, product and average.
- Write a program that reads length and breadth of a rectangle and prints its area.
- Write a program that reads temperature in Fahrenheit and prints its equivalent temperature in Celsius using the following formula.

$$c = 5/9(f - 32)$$