

PREScribed TEXTBOOK

COMPUTER SCIENCE

SECOND YEAR PUC

REVISED EDITION

2018

Department of Pre-University Education
Malleshwaram, Bengaluru - 560 012
www.pue.kar.nic.in

© Department of Pre-University
Education 2014-2015.

All Rights Are Reserved

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photo copying, recording or otherwise without the prior permission of the publisher.

Revised Edition - 2018

This book is sold subject to the condition that it shall not, by way of trade, be lent, resold, hired out or otherwise disposed of without the publisher's consent, in any form of binding or cover other than that in which it is published.

The correct price of this publication is the price printed on this page/cover page. Any revised price indicated by a rubber stamp or by a sticker or by any other means is incorrect and should be unacceptable.

Printed on 70 GSM Maplitho paper

Director's Message

Dear Students,

We at the Department of Pre-university Education, Karnataka strive to empower each student to dream big and equip them with the tools that enable them to reach new heights and successfully deal with the challenges of life. As Swami Vivekananda said, "**Real education is that which enables one to stand on one's own legs**".

The course contents in this book are designed with the objective of equipping you well for the next level of study.

We wish you well on your journey and look forward to you becoming a responsible citizen of the nation and give back to the betterment of the society.

With best wishes,

Sd/-

C. Shikha, IAS

Director

Department of Pre University Education
Bengaluru

Preface

Dr. P. Nagabhushan

BE,MTech,Phd,FIE,FIETE

Professor, Department of Studies in Computer Science

Chief Nodal Officer,

University of Mysore, Mysore-570 006

I am pleased to understand that Department of Pre-University Education, Govt. of Karnataka, took up an important academic exercise to revise the syllabi and accordingly came out with an appropriate text book, which is jointly authored by a committee of qualified and experienced faculty members drawn across Karnataka state.

The committee of authors consisted of Sri. Rajappa, Empress Govt. PU College, Tumkur, as the Chairperson and Sri. Santus Xavio B K, Stracey Memorial PU College, Bengaluru as the Co-ordinator and Sri Nagaraje Urs, Maharaja Govt. PU College, Mysore, Smt. M R Nagamani, SBRR Mahajana PU College Mysore, Sri Ravindra K V, Govt. PU College, Sagara, Smt. Sharon Mednora, Govt PU College, Malleshwaram, Bengaluru, Sri. Naveen Kumar B, Govt. PU College for Girls, Malleshwaram, Bangalore and Smt. Padmashree R, NMKRV PU college, Bengaluru, as the members. When the committee members met me for seeking my remarks on the draft syllabus proposed, I felt comfortable to see the balanced distribution of topics to serve the needs of Commerce as well as Science students of PU Education. The spread and the organisation of the syllabus is quite good and care has been taken to see that the contents in Second Year Pre-University Course is properly placed as an extension to First PUC. The committee has ensured that the entire syllabus is on par with NCERT.

The committee has put its best effort to provide a comprehensive coverage of both theoretical and practical aspects and also have paid careful attention to make the learners to acquire necessary skills, which makes the learners feel comfortable to work with the machine.

I am happy to understand that the draft book is thoroughly reviewed by Dr. Somashekara M T, Bengaluru University and also by prof. Mukundappa, Tumkur University. The exercise problems set by the authors at the end of every chapter are good enough to keep the interests of the learners alive.

I commend the excellent collective effort put by the committee of authors and also the two reviewers, and I am confident Second year PUC students of Karnataka would be greatly benefitted.

Computer Science Syllabus Review and Textbook Development
Committee

Department of Pre-University Education, Govt. of Karnataka

AUTHORS

Rajappa

Chairperson

Empress Govt. Pre-University College
Tumakuru

Co-ordinator:

Santus Xavio B K
Lecturer
Stracey Memorial Pre-University College
#52, St. Marks' Road, Bengaluru

Members:

Nagaraje Urs
Lecturer
Maharaja Govt. Pre-University College
JLB Road, Mysore

Sharon Mednora
Lecturer
Govt. Pre-University College
18th Cross, Malleshwaram
Bengaluru

Ravindra K V
Lecturer
Govt. Pre-University College
Sagara, Shimoga District

Nagamani M R
Lecturer
SBRR Mahajana Pre-University College
Jayalakshmipuram, Mysore

Padmashree R
Lecturer
NMKRV Pre-University College
Jayanagar, Bengaluru

Acknowledgements

I am very fortunate to have esteemed personality Dr. Nagabhushan P, Professor, Dept. of Computer Science, Manasagangothri, University of Mysore who guided us in shaping the syllabus. We are also grateful to our reviewers Dr. Somashekhar M T, Associate Professor, Dept. of Computer Science, Bengaluru University and Prof. Mukundappa , HOD, Dept. of Computer Science, University Science College, Tumkur. I whole heartedly thank them personally and on behalf of the committee.

I acknowledge Sri. Marulaiah B, Principal, Empress Govt. PU College, Tumkur for his outstanding encouragement. I also thank A B Jagadish, th Cross, Malleshwaram, Bengaluru who helped us by providing the Computer Science laboratory. I also thank the College Management, SBMM Mahajana PU College, Jayalakshmipuram, Mysore for their support. We thank Dr. Serkad Arunachalam Kribanandan, Principal, Stracey Memorial PU College, Bengaluru for his overall support rendered to us.

I proudly remember the service of Mr. Nagendrakumar K M, Lecturer, Govt. First Grade College, Tumkur who did fine DTP work amidst his busy engagements. I also thank Mr. Armstrong .M, former syllabus committee member, for his valuable suggestions and corrections.

I also recall the support of my wife Shylaja. Last but not least, I acknowledge the support of family and friends of our committee members who directly or indirectly helped us in bringing out this text book.

Rajappa
Chairperson

Computer Science Syllabus Review and
Textbook Development Committee
Department of Pre-University Education
Govt. of Karnataka

Government of Karnataka
Computer Science Syllabus Review Committee
Department of Pre-University Education, Bengaluru

Rajappa

Chairperson

Empress Govt. Pre-University College
Tumakuru

e-mail: rajappatumkur@gmail.com

Santus Xavio B K

Co-ordinator

Stracey Memorial PU College
Bengaluru

santus03@yahoo.com

Reviewers:

Dr. Nagabhushan P, Dept. of Computer Science, Manasagangothri, Mysore

Dr. Somashekhar M T, Dept. Computer Science, Gnanabharathi, Bengaluru

Prof. Mukundappa, HOD, Dept of Computer Science, University Science
College, Tumkur

Members:

Nagaraje Urs

Lecturer

Maharaja Govt. Pre-University College
JLB Road, Mysore

Sharon Mednora

Lecturer

Govt. Pre-University College
18th Cross, Malleshwaram, Bengaluru

Ravindra K V

Lecturer

Govt. Pre-University College
Sagara, Shimoga District

Naveen Kumar B

Lecturer

Govt. Pre-University College
13th Cross, Malleshwaram, Bengaluru

Nagamani M R

Lecturer

SBRR Mahajana Pre-University College
Jayalakshmipuram, Mysore

Padmashree R

Lecturer

NMKRV Pre-University College
Jayanagar, Bengaluru

Chapter No		Topics	Page No.
	UNIT A	BACKDROP OF COMPUTERS	35 Hrs
Chapter 1		Typical configuration of Computer system	1
	1.1	Introduction	2
	1.2	Motherboard	4
	1.2.1	Introduction to Motherboard	4
	1.2.2	Types of Motherboards	5
	1.2.3	Components of Motherboard	6
	1.3	Memory	15
	1.4	Power supply to the computer system	18
	1.5	Assembling the computer system	19
Chapter 2		Boolean Algebra	24
	2.1	Introduction	25
	2.2	Binary valued quantities-constants and variables	25
	2.3	Logical operations	26
	2.3.1	Logical functions or compound statements	26
	2.3.2	Logical operators	26
	2.4	Evaluation of Boolean expressions using truth table	29
	2.4.1	Basic logic gates	34
	2.5	Basic postulates of Boolean Algebra (with proof)	36
	2.5.1	Properties of 0 and 1	38
	2.5.2	Indempotence law	41
	2.5.3	Involution law	42
	2.5.4	Complementarity law	43
	2.5.5	Commutative law	44
	2.5.6	Associative law	46
	2.5.7	Distributive law-different forms	47
	2.5.8	Absorption law	48
	2.6	De Morgan's theorems	50
	2.6.1	De Morgan's I theorem	50
	2.6.2	De Morgan's II theorem	51
	2.6.3	Applications of De Morgan's theorems	53
	2.6.4	Basic duality of Boolean algebra	53
	2.7	Derivation of Boolean expressions	54
	2.7.1	Min terms	54
	2.7.2	Max terms	57
	2.7.3	Canonical expressions	57
	2.7.4	Minimization of Boolean expressions	64
	2.8	Simplification using Karnaugh map	65
	2.8.1	Sum-of-product reduction using Karnaugh map	66
	2.8.2	Product-of-sum reduction using Karnaugh map	76

Chapter 3	Logic gates	86
3.1	Introduction	87
3.1.1	Invertor (NOT gate)	87
3.1.2	OR gate	88
3.1.3	AND Gate	89
3.2	Derived Gates	90
3.2.1	NOR Gate	90
3.2.2	NAND Gate	91
3.2.3	XOR Gate	91
3.2.4	XNOR Gate	93
3.2.5	Circuit diagram	94
3.2.6	NAND,NOR as universal Gates	95
Chapter 4	DATA STRUCTURE	103
4.1	Introduction	104
4.2	Data representation	104
4.3	Classification of Data structures	105
4.3.1	Primitive Data structure	105
4.3.2	Operations on primitive data structures	106
4.3.3	Non-primitive Data structures	106
4.3.4	Linear data structure	107
4.3.5	Non-Linear data structure	107
4.4	Operations on linear data structures	107
4.5	Arrays	107
4.5.1	Types of array Memory representation of data	108
4.5.2	One dimensional array	109
4.5.3	Memory representation one dimensional array	109
4.5.4	Basic operations on one-dimensional array	109
4.5.5	Traversing using one dimension array	110
4.5.6	Searching an element	111
4.5.7	Insertion of an element	114
4.5.8	Deletion of an element	116
4.5.9	Sorting the elements	118
4.5.10	Two dimension Array	119
4.6	Stacks	123
4.6.1	Introduction	123
4.6.2	Representation of stacks in memory	124
4.6.3	Operations on stacks	127
4.6.4	Applications of Stacks	128
4.7	Queues	133
4.7.1	Introduction	133
4.7.2	Types of Queues	134

4.7.3	Operations on queues	136
4.7.4	Memory representation of queues	136
4.7.5	Applications of Queues	138
4.8	Linked lists	139
4.8.1	Introduction	139
4.8.2	Types of linked list	139
4.8.3	Operations on single linked lists	141
4.9	Non-Linear data structure	153
4.9.1	Introduction	153
4.9.2	Trees	153
4.9.3	Graphs	155
UNIT B	COMPUTING IN C++	45 Hrs
Chapter 5	Review of C++	158
5.1	Review of c++ language	158
5.2	Fundamentals of c++	160
5.3	Structure of c++ program	164
5.4	Libraray functions	164
5.5	Data types	165
5.6	Input and output operations	166
5.7	Control statements	167
5.8	Arrays	169
5.9	Functions	172
5.10	User-defined Functions	175
5.11	Structures	180
Chapter 6	Basic concepts of OOP	181
6.1	Introduction	182
6.2	Basic concepts of OOP	183
6.2.1	Objects	183
6.2.2	classses	183
6.2.3	Data Abstraction	184
6.2.4	Data Encapsulation	184
6.2.5	Inheritance	184
6.2.6	Overloading	185
6.2.7	Polymorphism	185
6.2.8	Dynamic Biding	185
6.2.9	Message passing	185
6.3	Advantages of OOP over earlier programming methods	186
6.4	Limitations of OOP	186
6.5	Applications of OOP	186

Chapter 7	Classes and objects		189
	7.1	Introduction	190
	7.2	Definition and declaration of classes and objects	191
	7.3	Access specifiers (scope of class & its members)	193
	7.3.1	Private	193
	7.3.2	Public	193
	7.3.3	Protected	194
	7.4	Members of the class	194
	7.5	Member functions	196
	7.5.1	Member functions inside class definition	196
	7.5.2	Member functions out side class definition	196
	7.6	Defining objects of a class	198
	7.7	Arrays as members of class	199
	7.8	Array of objects	200
	7.9	Objects as function arguments	202
	7.10	Differences between structures and classes in C++	204
Chapter 8	Function overloading		207
	8.1	Introduction	208
	8.2	Need for function overloading	208
	8.3	Definition and declaration of overloaded function	208
	8.4	Restrictions on overloaded function	209
	8.4.1	Calling over loaded functions	209
	8.5	Other functions in a class	210
	8.5.1	Inline function	211
	8.5.2	Friend function	212
Chapter 9	Constructor and Destructor		216
	9.1	Introduction	217
	9.2	Declaration and definition of constructor	218
	9.3	Types of constructors	219
	9.3.1	Default constructor	219
	9.3.2	Parameterized constructor	221
	9.3.3	Copy constructor	224
	9.4	Constructor overloading	227
	9.5	Destructor	228
Chapter 10	Inheritance(Extending classes)		232
	10.1	Introduction	233
	10.2	Base class	233
	10.3	Derived class	233
	10.3.1	Defining derived class	233
	10.3.2	Public derived class	234
	10.3.3	Private derived class	235

	10.3.4	Protected derived class	235
	10.4	Visibility modes	235
	10.4.1	Public inheritance	235
	10.4.2	Private inheritance	236
	10.4.3	Protected inheritance	236
	10.5	Levels of inheritance	236
	10.5.1	Single level inheritance	237
	10.5.2	Multilevel inheritance	237
	10.5.3	Multiple inheritance	238
	10.5.4	Hierarchical inheritance	238
	10.5.5	Hybrid inheritance	238
	10.6	Relationship between classes	240
	10.6.1	Virtual base classes	240
	10.6.2	Abstract classes	242
	10.6.3	Constructors in Derived classes	242
	10.6.4	Destructors in Derived classes	243
Chapter 11		Pointers	247
	11.1	Introduction	248
	11.2	Memory representation of pointers	248
	11.3	Declaration & initialization of pointers	249
	11.4	Address operator	249
	11.5	Pointer operator(indirection operator)	250
	11.6	Pointer arithmetic	250
	11.7	Pointer and arrays	251
	11.8	Arrays of pointers	252
	11.9	Pointers to strings	253
	11.10	Pointer as function parameters	253
	11.11	Pointer and structures	254
	11.12	Memory allocation of pointers(static and dynamic)	254
	11.12.1	Static allocation of memory	254
	11.12.2	Dynamic allocation of memory-new and delete	254
	11.13	Free store (heap memory)	256
	11.14	Memory leak	256
	11.15	Self Referential Structure	256
	11.16	Pointers and functions	257
	11.16.1	Invoking functions by passing the references	257
	11.16.2	Invoking functions by passing the pointers	258
	11.17	Memory comes and memory goes	260
	11.18	Pointer and objects	260
	11.19	this pointer	261

Chapter 12	Data file handling	266
12.1	Introduction	267
12.2	Header files(fstream.h)	268
12.2.1	Classes for file stream operation	269
12.3	Types of data files	270
12.3.1	Text file	270
12.3.2	Binary file	270
12.4	Opening & closing files	270
12.4.1	Opening file using constructor	270
12.4.2	Using open()	271
12.4.3	File modes -In ,out, app modes	272
12.4.4	closing files	273
12.5	Input and output operation in text files	273
12.6	Detecting end of file	275
12.7	File pointers -tellg(), tellp(), seekg(), seekp() functions	275
UNIT C	LARGE DATA, DATABASE & QUERIES	20 HRs
Chapter 13	Database Concepts	282
13.1	Introduction	283
13.2	Appllications of database	284
13.3	Origin of Data : Facts,data,information,features	284
13.4	Evolution of database	285
13.5	Data processing cycle	286
13.6	Data base terms	287
13.7	Data Types in DBMS	288
13.8	DBMS	289
13.9	Data abstraction	291
13.10	Data independence	293
13.11	Database Model	298
13.11.1	Hierachial data model	299
13.11.2	Network data Model	300
13.11.3	Relational Data model	301
13.12	Codd's Rules	302
13.13	Logical data concepts	304
13.13.1	Normalization	304
13.13.2	Entity-relationship Model	308
13.13.3	Cardinality	313
13.14	KEYS-Primary,Secondary,Candidate,Foreign, Alternate	315
13.15	Relational Algebra	318
13.16	Data warehousing	327
13.17	Data Mining	329

Chapter 14	Structured Query Language		333
	14.1	Introduction	334
	14.1.1	SQL Architecture	335
	14.2	SQL commands	337
	14.2.1	DDL	337
	14.2.2	DML	338
	14.3	Data types in SQL	339
	14.3.1	Exact Numeric data types	339
	14.3.2	Floating point Numeric data types	339
	14.3.3	Date and time data types	339
	14.3.4	Character and string data type	340
	14.4	Operators in SQL	340
	14.4.1	SQL arithmetic operators	340
	14.4.2	Comparison operators	341
	14.4.3	Logical operators	341
	14.5	SQL expressions	342
	14.5.1	SQL Boolean Expression	342
	14.5.2	SQL Numeric expression	343
	14.5.3	Date expression	343
	14.6	SQL constraints	344
	14.6.1	Primary key	344
	14.6.2	Foreign Key or Referential integrity	346
	14.6.3	Not NULL constraint	347
	14.6.4	Unique Key	347
	14.6.5	Check constraint	348
	14.7	Implementation of SQL Commands	348
	14.7.1	Create table statement	348
	14.7.2	Alter	349
	14.7.3	Insert Statement	350
	14.7.4	Select statement	351
	14.7.5	AND operator	353
	14.7.6	OR operator	354
	14.7.7	Update statement	354
	14.7.8	Delete Statement	356
	14.7.9	Order by	357
	14.7.10	Group by	357
	14.7.11	Distinct statement	359
	14.7.12	Join	361
	14.7.13	NULL	363
	14.8.1	Create View	365
	14.9.1	Commit	365

	14.10	DCL commands	365
	14.10.1	Grant command	365
	14.10.2	Revoke command	366
	14.11	Built-In Function	368
	14.11.1	Single row function	368
	14.11.2	Group function	368

UNIT D ADVANCED CONCEPTS IN COMMUNICATION TECHNOLOGY 20Hrs

Chapter 15	Networking Concepts		375
	15.1	Introduction	376
	15.1.1	Networking Goals	376
	15.1.2	Need of networking	376
	15.2.1	Arpanet	376
	15.2.2	OSI reference Model	377
	15.2.3	TCP/IP	378
	15.3.1	HTTP	380
	15.3.2	FTP	381
	15.3.3	SLIP	381
	15.4.1	Internet	381
	15.4.2	Interspace	382
	15.4.3	Elementary terminologies of networking	382
	15.4.4	Types of services	382
	15.4.5	Types of networking	383
	15.4.6	Networking Topologies	386
	15.4.7	Transmission medium	393
	15.4.8	Switching techniques	398
	15.4.9	Communication modes	399
	15.4.10	Networking devices	400
	15.5.1	Gateway	403
	15.6.1	SIM	404
	15.7.1	GPRS	406
	15.8.1	Applications of Networking	410
	15.8.2	Wi-fi	411
	15.9.1	Network security	411
	15.10.1	Cookies	413
	15.11.1	Virus	413
Chapter 16	Internet and Open source concepts		416
	16.1	Introduction	416
	16.1.2	Free software	417
	16.1.3	Open source software	417
	16.1.4	OSS and FLOSS	418

	16.1.5	GNU	419
	16.1.6	FSF	419
	16.2.1	OSI	419
	16.2.2	W3C	419
	16.2.3	Proprietary software	419
	16.2.4	www	420
	16.2.5	Telnet	420
	16.2.6	Web browser	420
	16.2.7	Webserver	420
	16.2.8	Webpage	421
	16.3	URL and domain	421
	16.4	E-Commerce	422
	16.4.1	Types of E-commerce	424
	16.4.2	Advantages of e-commerce	425
	16.5	IPR issues	426
Chapter 17		Web designing	428
	17.1	Introduction	429
	17.1.1	HTML structure	430
	17.2.1	Advanced HTML tags/commands	432
	17.2.2	Text formating	432
	17.2.3	Resizing text	432
	17.2.4	Example for resizing text	433
	17.2.5	Text layout	434
	17.2.6	Number listing	435
	17.2.7	Links	437
	17.2.8	Inserting images	438
	17.2.9	Background	439
	17.2.10	Background color and fixed images	440
	17.2.11	Tables	440
	17.2.12	Frames	442
	17.2.13	Forms	443
	17.2.14	Settings and text fields	444
	17.3.1	Web Hosting	447
	17.3.2	Domain registration	448
	17.4.1	Uploading HTML file	449
	17.5.1	XML	450
	17.6.1	DYNAMIC HTML	451
	17.7.1	Web scripting	453
		Model Question Paper	455

DESIGN OF QUESTION PAPER

CLASS: SECOND PUC

SUBJECT: COMPUTER SCIENCE (41)

Time : 3Hours 15 Minutes(of which minutes for reading the questions Paper).

Max.Marks:70

The weightage of the distribution of marks over different dimensions of the question paper shall be as follows:

Weightage to Objectives:

Objective	Weightage	Marks
Knowledge	30%	31
Understanding	40%	43
Application	20%	21
Skill	10%	10
Total	100%	105

Weight age to Content/Subject units: Computer Science(41)

Unit	Description	VSA(1 Mark)	SA(2 Marks)	LA(3 Marks)	E(5Marks)	Total Marks
<i>A 35 Hrs</i>	<i>BACKDROP OF COMPUTERS</i>	<i>3</i>	<i>2</i>	<i>3</i>	<i>3</i>	<i>31</i>
<i>B 45Hrs</i>	<i>COMPUTING IN C++</i>	<i>2</i>	<i>3</i>	<i>2</i>	<i>5</i>	<i>39</i>
<i>C 20Hrs</i>	<i>LARGE DATA, DATABASE & QUERIES</i>	<i>1</i>	<i>2</i>	<i>1</i>	<i>2</i>	<i>18</i>
<i>D 20Hrs</i>	<i>ADVANCED CONCEPTS IN COMMUNICATION TECHNOLOGY</i>	<i>4</i>	<i>1</i>	<i>2</i>	<i>1</i>	<i>17</i>
	<i>Total Marks</i>	<i>10</i>	<i>16</i>	<i>24</i>	<i>55</i>	<i>105</i>
<i>120 Hrs</i>	<i>Total No of Questions to be answered</i>	<i>$1 \times 10 = 10$</i>	<i>$2 \times 5/8 = 10$</i>	<i>$3 \times 5/8 = 15$</i>	<i>$5 \times 7/11 = 35$</i>	<i>70/37</i>

UNIT	DESCRIPTION	VSA (1 Mark)	SA (2 Marks)	LA (3 Marks)	E (5 Marks)	Total Marks
A 35 Hrs	BACKDROP OF COMPUTERS	3	2	3	3	31
Chapter 1 5 Hrs	Typical configuration of Computer system	1	-----	1	-----	4
Chapter 2 10 Hrs	Boolean algebra	-----	2	-----	1	09
Chapter 3 5 Hrs	Logic Gates	1	-----	1	-----	04
Chapter 4 15 Hrs	Data structures	1	-----	1	2	14
B 45Hrs	COMPUTING IN C++	2	3	2	5	39
Chapter 5 3 Hrs	Review of C++ covered in First PUC	-----	-----	-----	-----	---
Chapter 6 4 Hrs	OOP concepts	---	1	---	1	07
Chapter 7 6 Hrs	Classes and objects	1	-----	---	1	06
Chapter 8 3 Hrs	Function Overloading	-----	-----	-----	1	05
Chapter 9 8 Hrs	Constructors and Destructors	---	1	---	1	07
Chapter 10 8 Hrs	Inheritance	-----	-----	-----	1	05
Chapter 11 7 Hrs	Pointers	1	-----	1	-----	04
Chapter 12 6 Hrs	Data File handling	-----	1	1	-----	05
C 20Hrs	LARGE DATA, DATABASE & QUERIES	1	2	1	2	18
Chapter 13 8 Hrs	Database concepts	1	1	1	1	11
Chapter 14 12 Hrs	SQL commands	-----	1	---	1	07
D 20Hrs	ADVANCED CONCEPTS IN COMMUNICATION TECHNOLOGY	4	1	2	1	17
Chapter 15 10 Hrs	Networking Concepts	2	1	---	1	9
Chapter 16 5 Hrs	Internet and Open source concepts	1	---	1	---	4
Chapter 17 5 Hrs	Web Designing	1	-----	1	-----	4
	Total Marks	10	16	24	55	105
	Total No of Questions to be answered	1X10=10	2X5/8=10	3X5/8=15	5X7/11=35	70/37

List of programs to be conducted in practical sessions

Section A C++ and Data structure

1. Write a program to find the frequency of presence an element in an array.
2. Write a program to insert an element into an array at a given position.
3. Write a program to delete an element from an array from a given position
4. Write a program to sort the elements of an array in ascending order using insertion sort.
5. Write a program to search for a given element in an array using Binary search method.
6. Write a program to create a class with data members principle, time and rate. Create member functions to accept data values to compute simple interest and to display the result.
7. Write a program to create a class with data members a, b, c and member functions to input data, compute the discriminant based on the following conditions and print the roots.
 - If determinant=0, print the roots that are equal
 - If the discriminant is>0, print the real roots
 - If the discriminant<0, print that the roots are imaginary
8. Program to find the area of a square/rectangle/triangle using function overloading.
9. Program to find the cube of a number using inline functions.
10. Write a program to find the sum of the series $1 + x + x^2 + \dots + x^n$ using constructors.
11. Create a base class containing the data members roll number and name. Also create a member function to read and display the data using the concept of single level inheritance. Create a derived class that contains marks of two subjects and total marks as the data members.
12. Create a class containing the following data members register No., name and fees. Also create a member function to read and display the data using the concept of pointers to objects.
13. Write a program to perform push items into the stack.
14. Write a program to pop elements from the stack.
15. Write a program to perform enqueue and dequeue.
16. Write a program to create a linked list and appending nodes.

Section B SQL

17. Generate the Electricity Bill for one consumer
18. Create a student database and compute the result.
19. Generate the Employee details and compute the salary based on the department.
20. Create database for the bank transaction.

Section C HTML

21. Write a HTML program to create a study time-table.
22. Create an HTML program with table and Form.

Chapter 1**Typical Configuration of Computer system****Objectives**

- To understand various units of a computer system
- To recognize various components of the motherboard
- To analyze the configuration of todays' computer system
- Insight to assemble a computer system



1.1 Introduction:

The computer has evolved as a result of man's search for fast, accurate calculating devices. Computers have thus become an integral part of everyone's life and useful in many ways by increasing man's efficiency and enhancing his abilities.

Computers are used to perform various tasks in different fields like science, engineering, business, education, training, entertainment. The computer works at high speed, can handle large volumes of data with greater accuracy and have the ability to carry out a specified sequence of operations without human intervention.

The term **computer** basically includes a series of electrical and electronic circuits together to form a single unit to perform the required operations for the user. The computer has no intelligence of its own and thus cannot perform any task by itself. Thus it requires the hardware, software, data and users which form the computer system to perform the different operations.

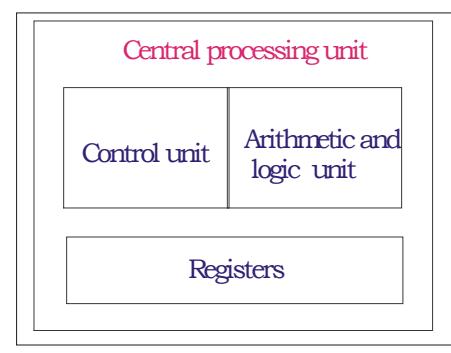
The related terms and definitions in the study of computer systems are:

- **Hardware** consists of physical devices of the computer such as keyboard, monitor, printer, processor and motherboard.
- **Software** consists of set of instructions called programs that instructs the computer the tasks to be performed and how it should be performed.
- **Data** are values or raw facts which are provided as input to the computer, then processed to generate some meaningful information.
- **Users** are people who write computer programs or interact with the computer.

Review of block diagram of Computer system

The computer system comprises of four main units which can be seen in the block diagram given below. They are,

1. Input Unit,
2. Central Processing Unit (CPU),
 - i. Control Unit (CU),
 - ii. Arithmetic Logic Unit (ALU),
 - iii. Registers,
3. Storage Unit, and
4. Output Unit



Central processing unit

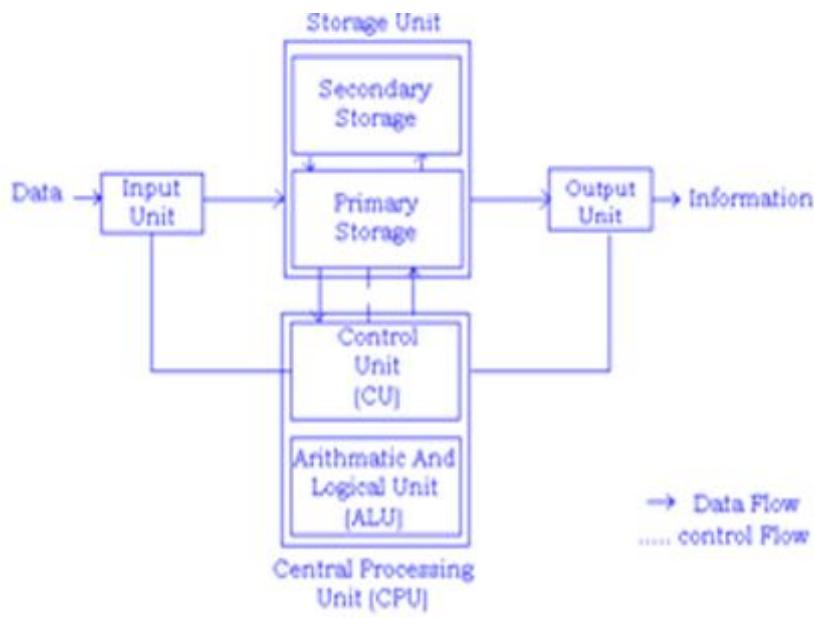


Figure 1.1 Block diagram of a computer

1. Input Unit

The user interacts with the computer via the input unit. The Input unit accepts data from the user and converts it into a form that is understandable by the computer. The input data can be characters, word, text, sound, images, document, etc. The input is provided to the computer using input devices like keyboard, mouse, joystick, trackball, microphone, scanner etc.

2. Central Processing Unit (CPU)

CPU controls, coordinates and supervises the operations of the computer. It is also responsible for processing of the input data. CPU consists of Arithmetic Logic Unit (ALU) and Control Unit (CU). CPU also has a set of **registers** for temporary storage of data, instructions addresses and intermediate results of calculation.

- a. **ALU** performs all the arithmetic and logic operations on the input data.
- b. **CU** controls the overall operations of the computer i.e. it checks the sequence of execution of instructions, controls and co-ordinates the overall functioning of the units of computer.
- c. **Registers** are high speed storage units within the CPU, but have least storage capacity. Registers are not referenced by their address, but are directly accessed and manipulated by the CPU during instruction execution. They are referred to as the **CPU's working memory** as they are used to store data, instructions, addresses and intermediate results of processing.

3. Storage Unit

There are two types of memory associated with storage unit. They are: **primary memory** and **secondary memory**.

The primary memory also called as the main memory of the computer, consists of RAM (Random Access Memory) and ROM (Read Only Memory) memories. Main memory stores the data, instructions, intermediate results and output, temporarily, during the processing of data. The input data that is to be processed is brought into the main memory before processing. The instructions required for processing of data, any intermediate results are also stored in the main memory. The output is stored in main memory before being transferred to the output device. CPU can work with the information stored in the main memory.

The **secondary memory** also called as the **external memory** of the computer stores permanently the data, programs and the output. Magnetic disks, magnetic tapes, optical disks and flash drives are examples of secondary memory.

4. Output Unit

The output unit provides the processed data i.e. the result generated after processing of data. The output may be in the form of text, sound, image, document, etc. The computer may display the output on a monitor, sends output to the printer or plotter for printing, and also sends sound output to the speaker, etc.

1.2 Motherboard

The computer is built up around a motherboard. The motherboard is the most important part of any computer. It is a large **Printed Circuit Board** (PCB) having many chips, ports, controllers and other electronic components mounted on it.

1.2.1 Introduction to Motherboard

The motherboard or the system board is the **main circuit board** inside a computer. Every component inside the computer has to communicate through the motherboard, either by directly plugging into it or by communicating through one of the motherboard ports. The motherboard provides a platform for all the components and peripherals to communicate with each other.

The electronic components mounted on the motherboard are processor, memory chips, interfaces, various ports and all expansion cards. The motherboard is the hub, which is used to connect all the necessary components of a computer. The RAM, hard drive, disk drives and optical drives are all plugged into interfaces on the motherboard.

The motherboard may be characterized by the form factor, chipset and type of processor socket used.

- **Form factor** refers to the motherboard's geometry, dimensions, arrangement and electrical requirements. Different standards have been developed to build motherboards, which can be used in different brands. Advanced Technology Extended (ATX) is the most common design of motherboard for desktop computers.
- **Chipset** controls the majority of resources of the computer. The function of chipset is to coordinate data transfer between the various components of the computer. As the chipset is integrated into the motherboard, it is important to choose a motherboard, which includes a recent chipset, in order to maximize the computer's upgradeability.
- The **processor socket** may be a rectangular connector into which the processor is mounted vertically, or a square shaped connector with many small connectors into which the processor is directly inserted.

1.2.2 Types of Motherboard

There are various types of motherboards available depending on the processors that are used. Some of them are XT, AT, Baby AT and ATX motherboards.

XT Motherboard:

XT stands for eXtended Technology. These are all old model motherboard. In these motherboards, we find old model processor socket LIF (Low Insertion Force) sockets, ram slots DIMM and ISA (Industry Standards Architecture) slots, 12 pin power connector and no ports.

They have slot type processors, DIMM memory modules, ISA slots for add-on card, and no ports. There are connectors and add-on cards for ports.

Example: Pentium-I, Pentium-MMX, Pentium -II and Pentium-II Processors.

AT Motherboard:

AT stands for Advanced Technology. Advanced Technology Motherboards have PGA (Pin Grid Array) Socket, SDRAM slots, 20 pin power connector PCI slots and ISA slots. We find the above components on AT motherboards.

Example: Pentium III Processors

Baby AT Motherboard:

Baby AT Motherboards have the combination of XT and AT. They have slot type processor sockets and PGA processor sockets, SDRAM slots and DDRRAM slots, PCI slots and ISA slots, 12 pin power connector and 20 pin power connector and ports.

Example: Pentium-III and Pentium-IV

ATX Motherboard:

ATX stands for **Advanced Technology eXtended**. Latest motherboards all are called as ATX motherboards, designed by ATX form factor. In this motherboard, we find MPGA processor sockets, DDRRAM slots, PCI slots, AGP slots, Primary and secondary IDE interfaces, SATA connectors, 20 pin and 24 pin ATX power connector and ports.

Example: Pentium-IV, Dual Core, Core 2 Duo, Quad Core, i3, i5 and i7 processors.

1.2.3 Components of Motherboard

The motherboard components are:

- Processor (CPU)
- BIOS
- CMOS
- Slots
- Disk Controllers
- I/O Ports and Interfaces
- BUS

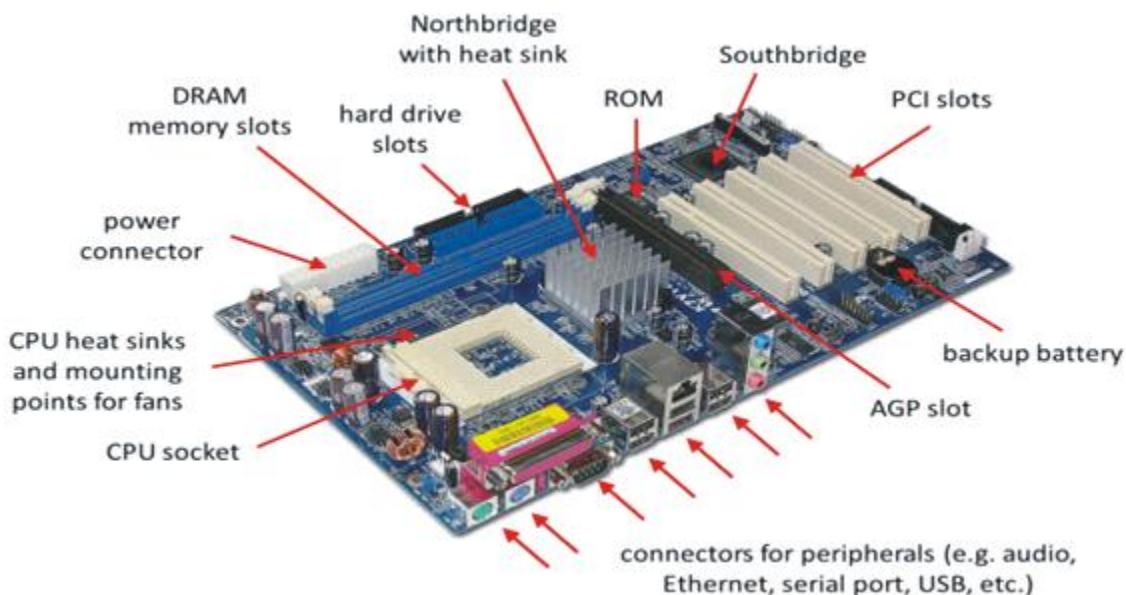


Figure 1.2 Motherboard components

Processor (CPU)

The processor or CPU is the main component on the motherboard and is called the brain of the computer. The CPU consists of Arithmetic Logic Unit (ALU) and Control Unit (CU). CPU also has a set of **registers** which are temporary storage areas for holding data, and instructions. ALU performs the arithmetic and logic operations on the data. CU is responsible for organizing the processing

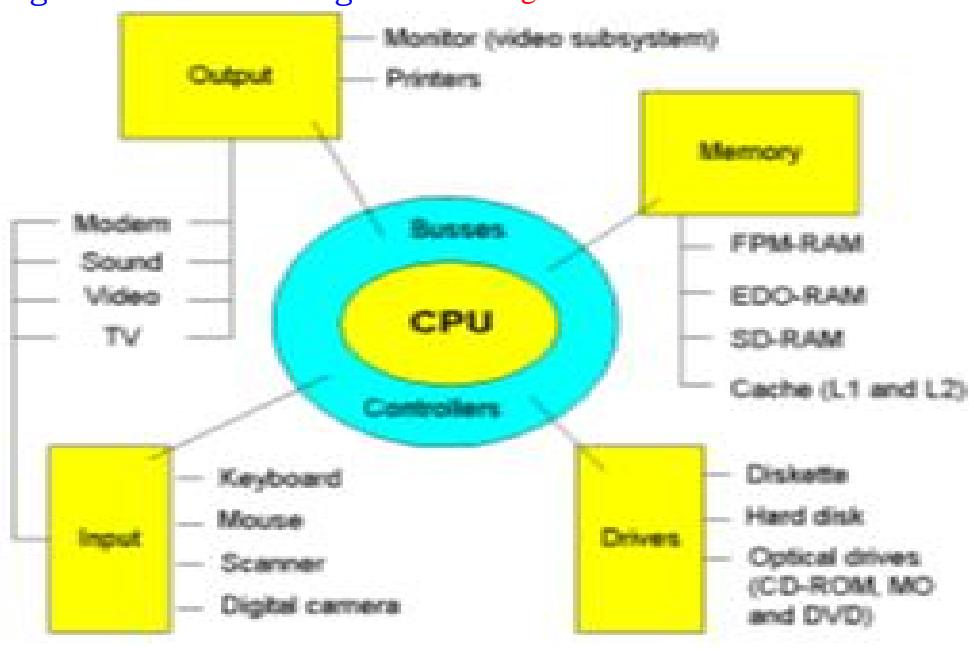
of data and instructions. CU controls and coordinates the activity of the other units of computer.

During processing the CPU gets data and instructions from the memory and interprets the program instructions and performs the arithmetic and logic operations required for the processing of data. It then sends the processed data to the memory.

The clock speed of a CPU is defined as the frequency with which a processor executes instructions or the data that is processed. Higher clock frequencies mean more clock ticks per second. The computer's operating speed is linked to the speed of the system clock. The clock frequency is measured in millions of cycles per second or megahertz (MHz) or gigahertz (GHz) which is billions of cycles per second. A CPU's performance is measured by the number of instructions executed per second, i.e. MIPS or BIPS. PCs presently come with a clock speed of more than 1GHz.

In Windows OS, the System Properties dialog box is selected to see the processor name and clock frequency.

The diagram of the CPU is given in Figure 1.3 CPU with Buses and Controllers



The CPU is fabricated as a single Integrated Circuit (IC) chip and is also known as the **microprocessor**. This tiny chip contains the entire computation engine. The microprocessor is plugged into the motherboard of the computer. **Intel** is one of the leading processor manufacturers in the world today.

General Structure of motherboard

The primary function of the processor is to execute the instructions given to it and to produce the results. It fetches instructions and data from the primary memory and performs the required operations. This movement of data between

the processor and memory is established by a communication path called **bus**. The processor contains number of special purpose **registers** in addition to **ALU** which is responsible for doing calculations. The different components inside the processor can be seen in the figure 1.4.

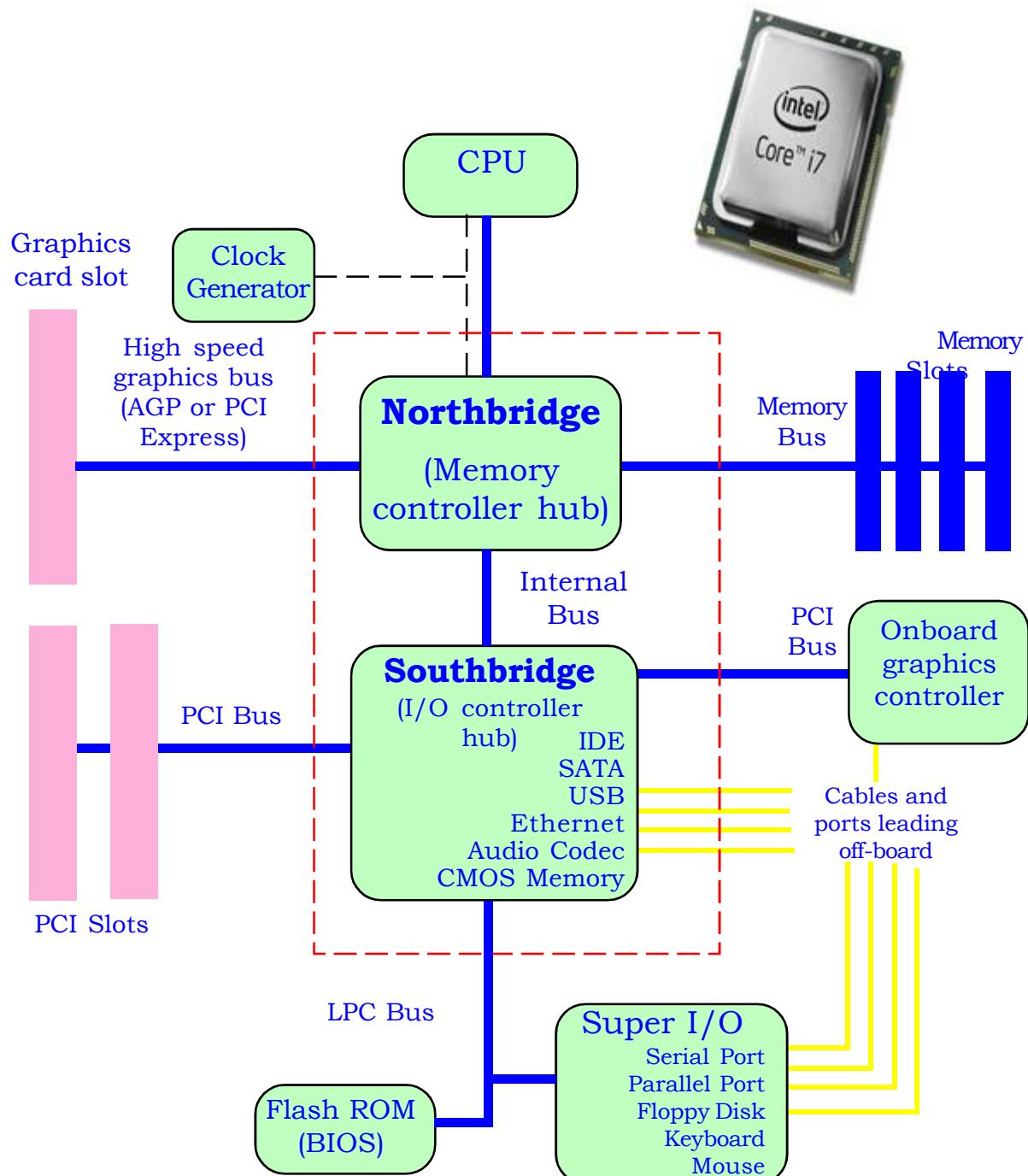


Figure 1.4 Schematic diagram of Motherboard

Figure 1.5 is the replica of the motherboard structure, except that it displays the actual devices connected to the CPU.

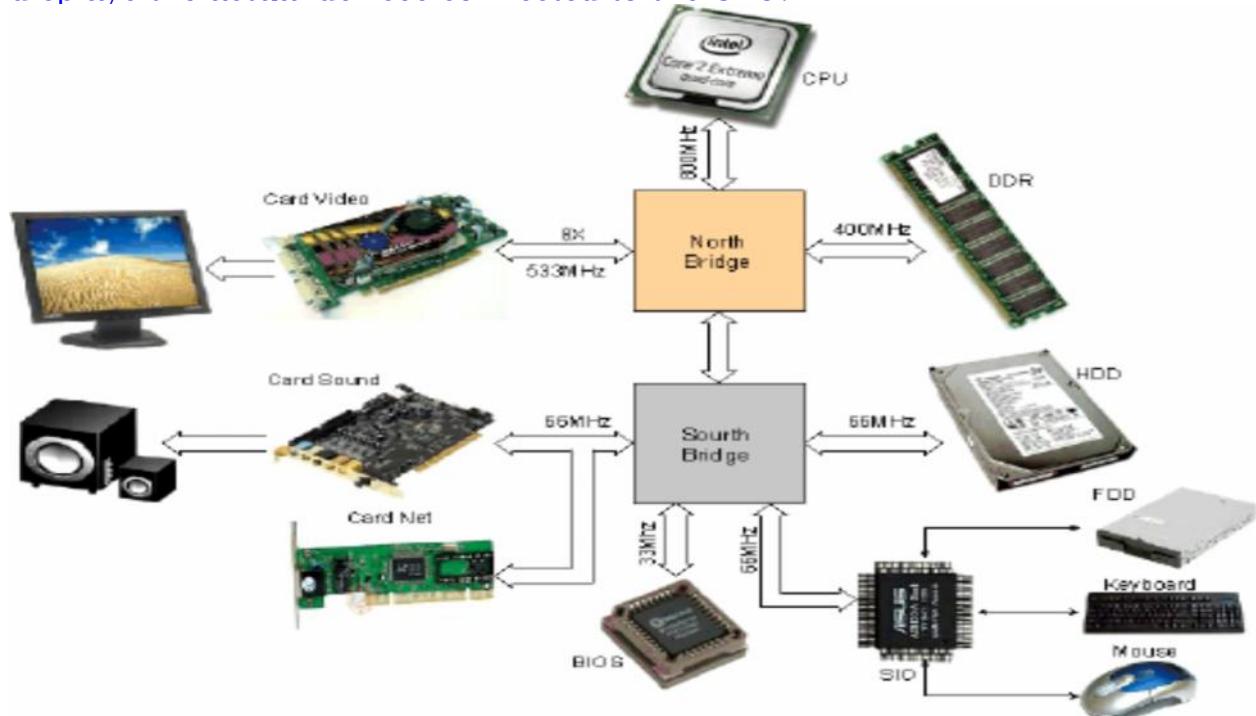


Figure 1.5 Overview of the motherboard structure

The **north bridge** or **host bridge** is one of the two chips in the core logic chipset on a PC motherboard, used to manage data communications between the CPU and motherboard. It is supposed to be paired with a second support chip known as a **south bridge**.

North Bridge or north Chipset is responsible for control of high speed components like CPU, RAM, and Video Card. Chipset BUS speed control and switch control data, ensuring data back and forth between the components is a smooth and continuous, fully exploit the speed of the CPU and RAM. It can be a chipset like the traffic in an intersection, as drivers switch traffic lights to allow each data stream passes through a period of time, while speed control is a BUS different directions of the intersection, the vehicle must run on a specified speed.

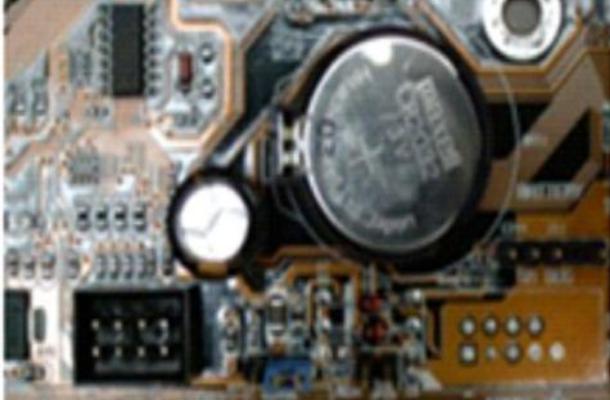
South Bridge or south Chipset is similar as north chipset, but the south bridge driver chipset components slower as: Sound Card, Net Card, hard disk, CD ROM drive, USB port, SIO and BIOS IC etc.

BIOS (Basic Input Output System)

BIOS is a small chip on the motherboard that holds a set of instructions to load the hardware settings required to activate various devices like keyboards, monitors or disk drives. The BIOS runs when the computer is switched ON. It performs a **Power On Self Test (POST)** that checks if the hardware devices are present and functioning properly. BIOS invoke the bootstrap loader to load the

operating system into memory. Most new PCs come with Flash BIOS-these BIOS can be software upgraded to support new devices.

CMOS (Complementary Metal Oxide Semiconductor)



CMOS is a type of memory chip to store the date, time and system setup parameters. These parameters are loaded every time the computer is started. That is why we observe, when the computer is turned ON, the system still displays the correct clock time. BIOS as well as CMOS are kept powered by a small lithium Ion battery located on the motherboard. It can be seen in the figure 1.6 below.

Figure 1.6 CMOS battery

Slots

A slot is an opening in a computer where you can insert a printed circuit board. Slots are often called **expansion slots** because they allow you to expand the capabilities of a computer.

- **Expansion Slots** These slots are located on the motherboard. The expansion cards are inserted in the expansion slots. These cards give the computer new features or increased performance. There are several types of slots:
- **ISA (Industry Standard Architecture) slot** – ISA slot is used to connect modem and input devices.
- **PCI (Peripheral Component Inter Connect) slot** – PCI slots are used to connect graphics accelerator cards, sound cards, internal modems or SCSI cards. They are much faster than ISA cards.
- **AGP (Accelerated Graphic Port) slot** – AGP slot is meant to provide faster access to a graphics accelerator card, thus enhancing the visual experience for the user. All Celeron and Pentium-III motherboards come with an AGP slot.
- **RAM slot** – RAM slot is used to install memory and is of two types. They are SIMM (Single Inline Memory Module) slot and DIMM (Dual Inline Memory Module) slot. The original Pentium systems typically have either four 72-pin SIMM slots, or two 168-pin DIMM slot to install memory.

- **Processor slot** – Processor slot is used to insert the processor chip which is the largest chip on the motherboard. It can be identified, as a heat sink or fan is located on top of it.
- **PCI Express slot** – It has faster bus architecture than AGP and PCI buses.
- **PC Card** – It is used in laptop computers. It includes Wi-Fi card, network card and external modem.

Disk Controllers

Disk controller is the circuit that enables the CPU to communicate with a hard disk, floppy disk or other kind of disk drive. Modern disk controllers are integrated into the disk drive.

Hard disk controller (HDC)

The hard disk controller is the interface that enables the computer to read and write information to the hard drive. Today, hard drives have the controller built on to them.

The first standard hard disk controller developed is the IDE standard drive also known as Advanced Technology Attachment (ATA). This drive is attached to the motherboard by means of 40-wire ribbon cable. The IDE standard also allows two drives to connect in a daisy-chain fashion. The enhanced IDE (EIDE) standard followed shortly. The EIDE standard is a specification that allows four drives to be connected to a dual channel controller.

Floppy disk controller (FDC)

A floppy-disk controller is the interface that directs and controls reading from and writing to a computer's floppy disk drive (FDD). The floppy disk controller usually performs data transmission in direct memory access (DMA) mode.

A single floppy-disk controller board supports a 33-wire ribbon cable and can connect up to four floppy disk drives to the motherboard. The controller is linked to the system bus of the computer and appears as a set of I/O ports to the CPU.

I/O Ports and Interfaces

The ports and interfaces are used to connect external devices like printers, keyboards or scanners to the computer, which gets connected to the computer's motherboard. These ports and interfaces are found on the rear side of a computer. There are several types of ports like serial port, parallel port, USB port, and AGP port etc. which is given in figure 1.7.

Serial port

Serial port is also known as communication (COM) ports or RS-232-c ports. They are used for connecting communication devices like mouse and modem. This port transfers data serially one bit at a time. It needs a single wire to transmit 1 bit of data. Hence it takes 8 times longer to transfer a byte. There are two varieties of Com ports, the 9-pin ports and 25-pin ports.

Parallel port:

Parallel ports are used to connect external input/output devices like printers or scanners. This port facilitates the parallel transfer of data, usually one byte (8-bits) at a time. Parallel ports come in the form of 25-pin connector.

IDE (Integrated Digital Electronics) port

IDE devices like CD-ROM drives or hard disk drives are connected to the motherboard through the IDE port.

USB (Universal Serial Bus) port

USB port gives a single, standardized, easy-to-use way to connect a variety of newer peripherals to a computer. These devices include printers, scanners, digital cameras, web cameras, speakers etc. USB supports a data speed of 12 megabits per second, supporting up to 127 devices. USB is a plug-and-play interface between a computer and add-on devices. i.e. a new device can be added to the computer without adding an adapter card or even turning the computer off. The figure 1.8 shows the symbol used to represent a USB port.

PS-2 port (Personal System-2 port)

The PS-2 port was developed by IBM to interface keyboards and pointing devices like mouse, trackballs and touch pads. This port is also called as mouse port as most computers now have a PS-2 port to connect a mouse. This port uses synchronous serial signals to communicate between the keyboard and a mouse to the computer.

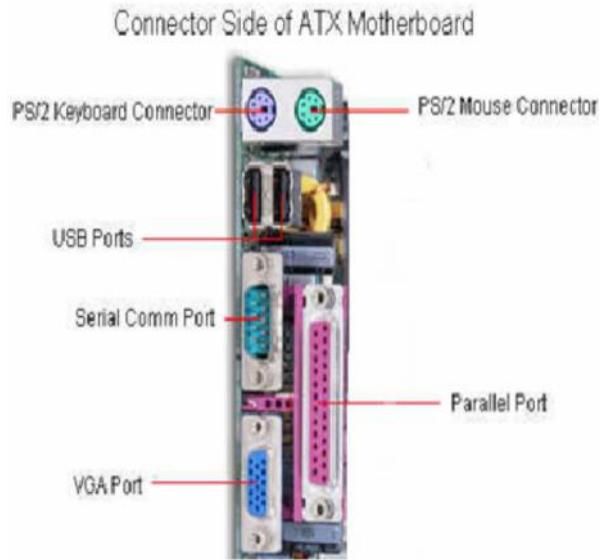


Figure 1.7 Types of ports in the motherboard



Figure 1.8 USB port

AGP (Accelerated Graphics Port) port

The AGP port is used to connect to graphic card that provides high-speed video performance typically required in games and other multimedia applications.

SCSI (Small Computer System Interface) port

This port is used for adding external devices such as high-speed hard-disks, high-end scanners and CD-ROM drives. It does fast data transfers and I-O operations. These ports are expensive, as they provide faster access at very high speeds and need separate dedicated adapters to function.

VGA (Visual Graphics Adaptor) port connects monitor to a computer's video card. It has 15 holes and is similar to serial port connector, but serial port connector has pins, this has holes.

Power Connector has three-pronged plug. It connects to the computer's power cable that plugs into a power bar or wall socket.

Firewire Port transfer large amounts of data at very fast speed. It connects camcorders and video equipment's to the computer. The data travels at 400 to 800 megabits per second. It is invented by Apple. The three variants of firewire port are 4-Pin firewire 400 connector, 6-Pin firewire 400 connector and 9-Pin firewire 800 connector

Modem (Modulator demodulator) connects a PC's modem to the telephone network.

Ethernet Port connects to a network and high speed Internet. It connects network cable to a computer. This port resides on an Ethernet Card. Data travels at 10 megabits to 1000 megabits per second depending upon the network bandwidth.

Game Port connects a PC to a joystick. It is now replaced by USB.

DVI (Digital Video Interface) port connects a Flat panel LCD monitor to the computer's high-end video graphic cards. It is very popular among video card manufacturers.



Sockets are used to connect microphone, speakers to sound card of the computer.

MIDI (Musical Instrument Digital Interface) port is a system designed to transmit information between electronic musical instruments. A MIDI musical keyboard can be attached to a computer and allow a performer to play music that is captured by the computer system as a sequence of musical notes with

Figure 1.9 MIDI port

electronic musical instruments. A MIDI musical keyboard can be attached to a computer and allow a performer to play music that is captured by the computer system as a sequence of musical notes with

the associated timing (instead of recording digitized sound waves). The port and interface are required for connectivity.

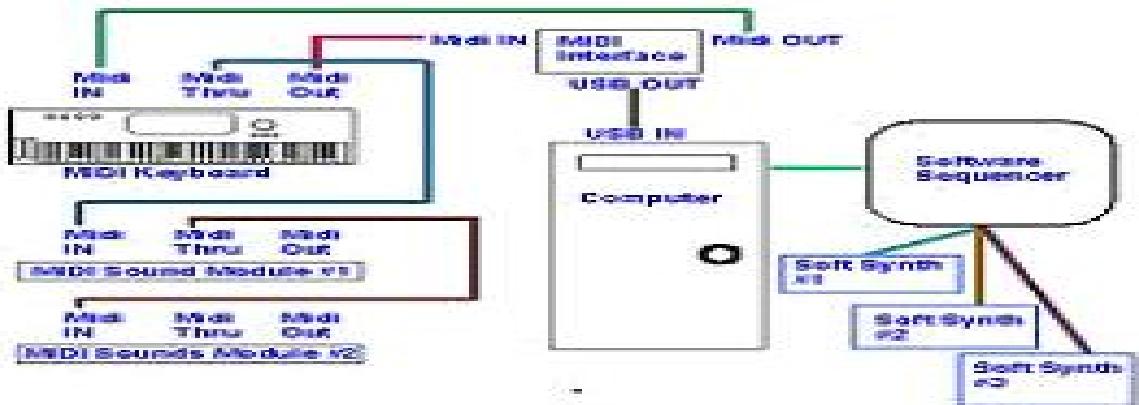


Figure 1.10 MIDI interface

BUS

The different components of computer, i.e. CPU, I/O unit, and memory unit are connected to each other by a **bus**. The data, instructions and the signals are carried between the different components via a bus.

A bus is a collection of parallel wires that form a pathway to carry address, data and control signals

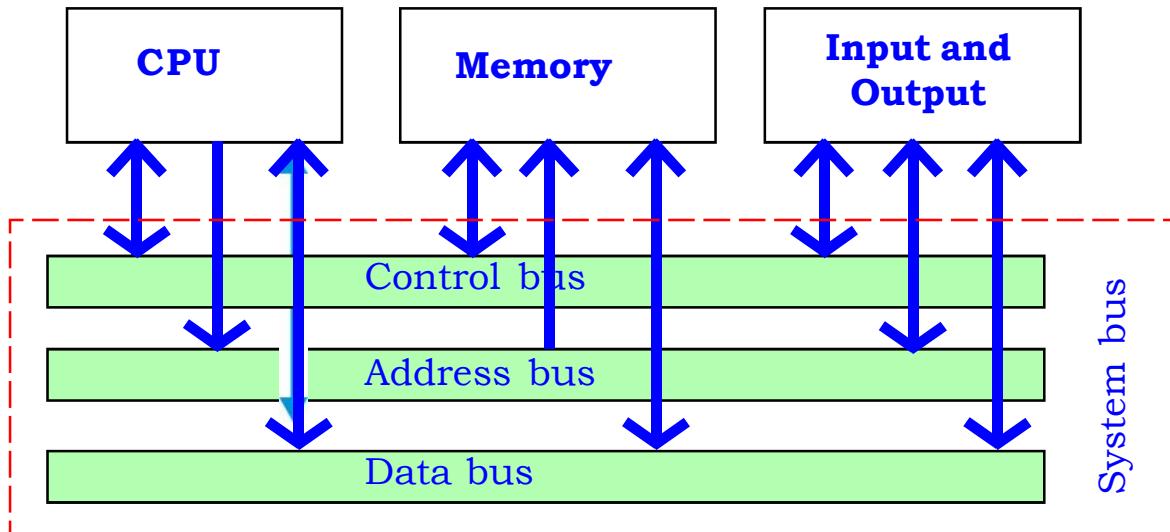


Figure 1.11 Bus structure

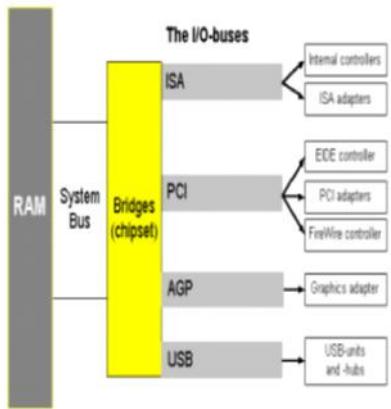
The functional features of a bus are:

- A bus is a set of wires and each wire can carry one bit of data.
- A bus width is defined by the number of wires in the bus.

A computer bus can be divided into two types: **Internal bus** and **External bus**.

- The **Internal bus** connects major computer components like, processor, memory and I/O. It is also called as **System bus**.
- The **External bus** connects the different external devices, peripherals, expansion slots, I/O ports and drive connections to the rest of computer. The external bus allows various devices to be attached to the computer, thus expanding the computer's capabilities. It is also called as **Expansion bus** and is slower than the system bus.

A system bus or expansion bus comprise of three kinds of buses: **data bus**, **address bus** and **control bus** shown in figure 1.11.



- **Data bus** provides a path to transfer data between CPU and memory. The data bus may consist of 32, 64, 128 lines of wire. The number of lines is referred to as the width of the data bus. The data bus width affects the speed of computer. In a 16-bit line bus can transfer 16 bits of data. The bus speed is measured in MHz and higher the bus speed, faster the processor speed.
- **Address bus** connects CPU and RAM with a set of lines of wire similar to data bus. The address bus width determines the maximum number of memory locations the computer can address.

Figure 1.12 I/O Buses shows the computer can address. Pentium Pro, II, III, IV have 36-bit address bus that can address 2^{36} bytes or 64 GB of memory. PCs presently have a bus speed varying from 100 MHz to 400 MHz.

- **Control bus** is used to control the access to and the use of the data and address lines.

1.3 Memory

A computer memory refers to the electronic storing space for instructions and data where the computer's processor can reach quickly. The computer storage refers to permanent computer memory that stores all the data files and instructions even after the computer system is turned off.

A computer processor has very limited memory. Thus it has to rely on other kinds of memories to store data, instructions and results.

The memory in a computer can be of two basic types: **Internal memory** and **Secondary memory** shown in figure 1.13

- **Internal memory**

Internal memory includes **registers**, **cache memory** and **primary memory** which can be directly accessed by the processor. It is used for temporary storage

of data and instructions on which the processor is currently working. This memory is the fastest among all other memories and is expensive. Therefore a very small part of internal memory is used in the computer system. The features of internal memory are:

- Temporary storage
- Limited storage capacity
- Fast access
- High cost

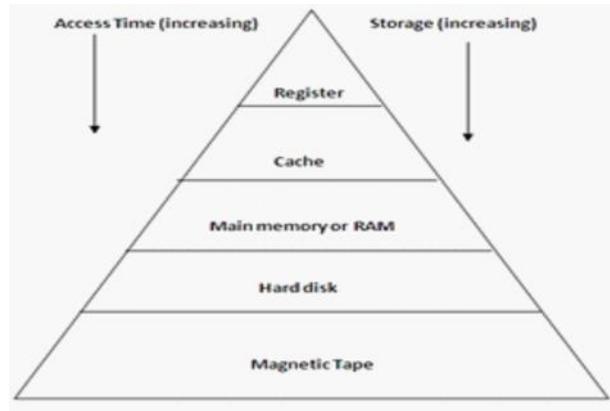


Figure 1.13 Memory accessing levels of the processor

Registers

The registers are high speed temporary storage areas located inside the CPU. After the CPU gets the data and instructions from the cache or RAM, the data and instructions are moved to registers for processing. These registers work under the direction of the control unit (CU) to accept, store and transfer instructions or data, and perform arithmetic or logical comparisons at high speed. Since CPU uses registers for the processing of data, the number of registers in a CPU and the size of each register affect the power and speed of a CPU.

Cache memory

The cache memory is a very high speed memory placed in between RAM and CPU. Cache memory stores data that is used more often, temporarily and makes it available to CPU at a fast rate. Hence it is used to increase the speed of processing. During processing, the CPU first checks cache for the required data. If data is not found in cache, then it looks in the RAM for data.

The Cache memory is a high speed memory available inside CPU to speed up access of data and instructions stored in RAM memory.

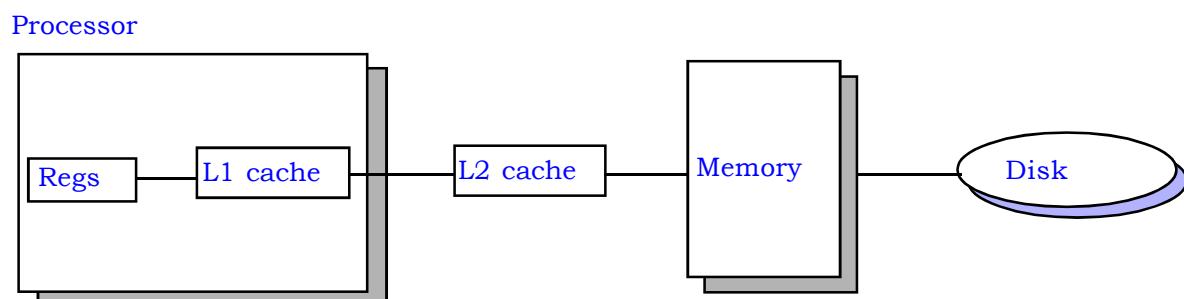


Figure 1.14 Illustration of cache memory

Cache memory is very expensive, so it is smaller in size. Generally, computers have cache memory of sizes 256 KB to 2MB.

Cache memory is built into the processor, and may also be located next to it on a separate chip between CPU and RAM. Cache built into the CPU is faster than separate cache, almost at the speed of the microprocessor itself. However, separate cache is roughly twice as fast as RAM.

The CPU has a built-in Level1 (L1) cache and Level2 (L2) cache, as shown in figure 1.14 below. In addition to the built-in L1 and L2 cache, some CPUs have a separate cache chip on the motherboard called Level3 (L3) cache. These days, high-end processor comes with built-in L3 cache, like in Intel core i7. The L1, L2 and L3 cache store the most recently executable instructions, the next ones and the possible ones, respectively. Typically, CPUs have cache size varying from 256KB (L1), 6MB (L2), to 12MB (L3) cache.

Primary memory

Primary memory is also known as main memory. This memory is of two types: Random Access Memory (RAM) and Read Only Memory (ROM)

- **RAM** temporarily stores the computer's operating system, application programs and current data so that the processor can reach them quickly. RAM is a faster memory and volatile in nature. i.e. when the power is switched off, the data in this memory is lost.
- **ROM** is a small memory, which stores the boot firmware (called BIOS). BIOS hold enough information to enable the computer to check its hardware and load its operating system into its RAM at the time of system booting. ROM is non-volatile in nature. i.e. even when the computer is switched off, the contents of ROM remains available.

Types of RAM

There are different types of RAM, depending on the technology used to construct a RAM. Some of the common types are:

DRAM or Dynamic RAM is the most common type of memory chip. DRAM is mostly used as main memory, since it is small and cheap. It uses transistors and capacitors. The transistors are arranged in a matrix of rows and columns. The capacitor holds the bits of information 0 and 1. The transistor and capacitor are paired to make a memory cell. The transistor acts as a switch that lets the control circuitry on the memory chip read the capacitor or change its state.

DRAM must be refreshed continually to store information; otherwise it will lose what it is holding. The refresh operation occurs automatically thousands of times per second. DRAM is slow because the refreshing takes time. Access speed of DRAM ranges from 50 to 150 ns.

SRAM or Static Random Access memory chip is usually used in cache memory due to its high speed. SRAM uses multiple transistors (4 to 6), for each memory cell. It does not have a capacitor in each cell. A SRAM memory cell has more parts, so it takes more space on a chip than DRAM cell. It does not need constant refreshing and therefore is faster than DRAM. SRAM is more expensive than DRAM, and it takes up more space. It stores information as long as it is supplied with power. SRAM is very fast and easier to use. The access speed of SRAM ranges from 2 to 10ns.

SDRAM or Synchronous Dynamic Random Access Memory is a special type of DRAM that is synchronized to the system clock. Since it is synchronized to the CPU, it knows when the next cycle is coming, and has the data ready when the CPU requests it. This increases efficiency by reducing CPU waiting time.

DDR-SDRAM or Double-Data Rate SDRAM works the same way as does ordinary SDRAM. Data transfer rate is double when compared to SDRAM.

1.4 Power Supply to a Computer System

Electric power is the main source of supply for the operation of electronic components of a computer. Therefore continuous power supply is essential for the computer to prevent them from failures, breakdown or shutdown. All computers come with a power supply.

There are two types of power supply connected to a computer system. They are, Switch Mode Power Supply (SMPS) and Uninterruptable Power Supply (UPS).

- **SMPS**

An SMPS converts AC power from an electrical outlet to the DC power needed by system components. An SMPS is a metal box in the rear of the system that is attached to the computer chassis and to the system board. The power supply contains the power card plug and a fan for cooling, because it generates a lot of heat. An SMPS with a rating of more than 300 watts is needed; any less will not reliably power modern components. In a PC the SMPS converts 230 volts of AC to 5 to 12 DC volts and the wattage is around 180 to 300 watts, 450 watts and 500 watts.

- **UPS**

An UPS is a power supply that includes a battery to maintain power in the event of a power failure. Typically, an UPS keeps a computer running for several minutes to few hours after a power failure, enabling us to save data that is in RAM and then shut down the computer gracefully.

Many UPS now offer a software component that enables us to automatically backup and shut down procedures in case there is a power failure while we are away from the computer.

- **Types of UPS**

There are two types of UPS: **Online UPS** and **Standby UPS**

Online UPS – An online UPS avoids those momentary power lapses by continuously providing power from its own inverter, even when the power line is functioning properly. Online UPS is more costly than Standby UPS. For a PC with color monitor 15", requires an UPS of 500VA and for a PC with color monitor 17", requires an UPS of 600VA.

Standby UPS – A Standby UPS (or off-line UPS) monitors the power line and switches to battery power as soon as it detects a problem. The switch over to battery, however, can require several milliseconds, during which time the computer is not receiving any power.

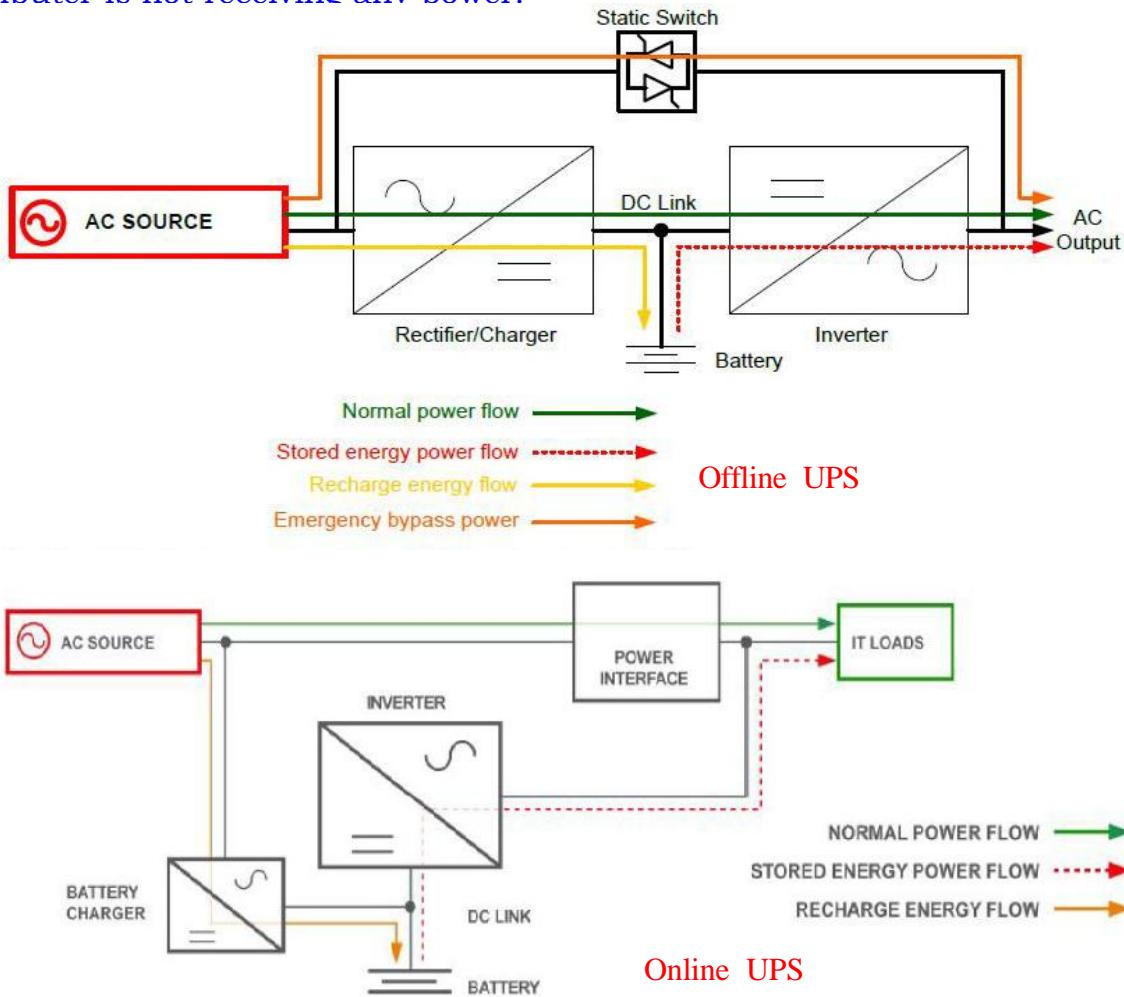


Figure 1.15 Types of UPS

1.5 Assembling the Computer System

Computer configuration is the process of setting up your hardware devices and assigning resources to them so that they work together without problems. A properly-configured system will allow you to avoid nasty resource conflict problems, and make it easier for you to upgrade your system with new equipment in the future. An improperly-configured system will lead to strange errors and problems, and make upgrading a nightmare.

Basic components for assembling a new computer system

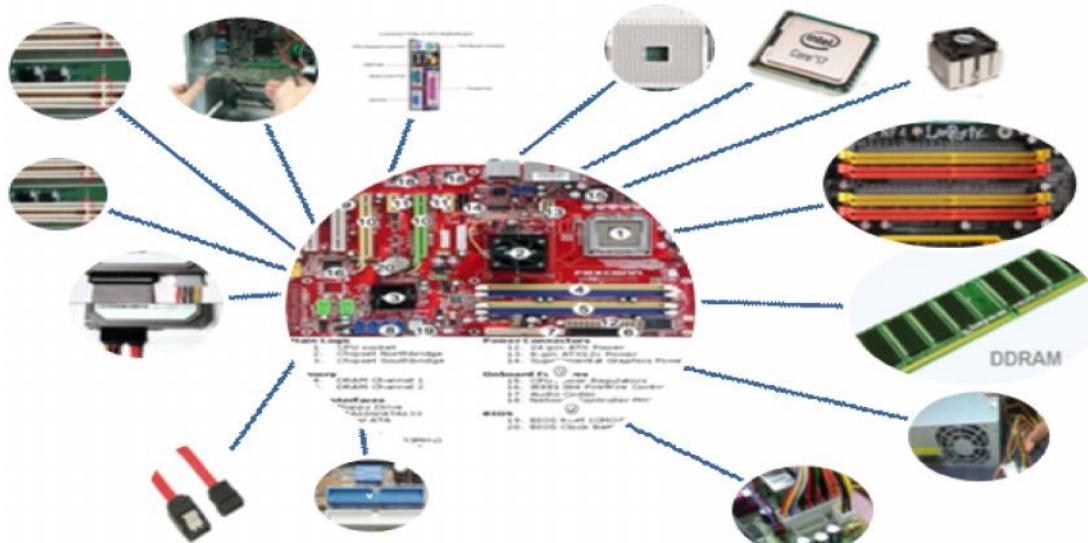


Figure 1.16 Components of Computer System

Sl.No	Component Name	Specifications	Brand	Amount
1.	Processor	Dual core,i3,i5,i7	Intel/ AMD/ Motorola	
2.	Mother Board	XT/AT/AT Baby/ATX		
3.	Primary Memory (RAM)	DDRAM1 DDRAM2 DDRAM3 1GB/2GB/4GB/8GB/16GB		
4.	Secondary Memory (HDD)	IDE/SATA/SCSI 500GB/1TB		
5.	Cabinet with SMPS	SMPS with 450W/500W/550W		
6.	Keyboard	PS2/USB/Wireless		
7.	Mouse	PS2/USB/Wireless		
8.	Monitor	CRT/LED/TFT 14"/18"		
9.	Printer(optional)	Dot-matrix/inkjet/Laser		
10.	Camera(optional)	High resolution		
11.	Mike(optional)	Analog		
12.	UPS(atleast 20minutes)	Online /Offline		
13.	Speaker	2.1 /5.1		

Points to remember

- ◆ The **motherboard** is the main circuit board inside a computer which provides a platform for all the components and peripherals to communicate with each other.
- ◆ The motherboard may be characterized by the **form factor**, **chipset** and **type of processor socket** used.
- ◆ The **motherboard types** are XT, AT, Baby AT and ATX motherboards.
- ◆ The **motherboard components** are Processor (CPU), BIOS, CMOS, Slots, Disk Controllers, I-O Ports/Interfaces and BUS
- ◆ The **processor** is the main component on the motherboard and is called the brain of the computer.
- ◆ The **clock speed** of a CPU is defined as the frequency with which a processor executes instructions or the data is processed.
- ◆ The **north-bridge** and **south bridge** are the two chips in the core logic chipset on a PC motherboard, used to manage data communications between a CPU and a motherboard.
- ◆ **BIOS** is a small chip on the motherboard that holds a set of instructions to load the hardware settings required to activate various devices like keyboards, monitors or disk drives.
- ◆ **CMOS** is a type of memory chip to store the date, time and system setup parameters.
- ◆ A **slot** also called as expansion slots, allows expanding the capabilities of a computer to give the computer new features or increased performance.
- ◆ The **disk controller** is the circuit which enables the CPU to communicate with a hard disk, floppy disk or other kind of disk drive.
- ◆ The **ports** and **interfaces** are used to connect external devices like printers, keyboards or scanners to the computer, which gets connected to the computer's motherboard.
- ◆ A **bus** is a collection of parallel wires that form a pathway to carry address, data, and control signals.
- ◆ A **system bus** or expansion bus comprises of data bus, address bus and control bus
- ◆ A **computer memory** refers to the electronic storing space for instructions and data where the computer's processor can reach quickly.
- ◆ The **parts of internal memory** are registers, cache and primary memory-RAM and ROM.
- ◆ **Cache memory** is a high speed memory available inside CPU in order to speed up access to data and instructions stored in RAM memory.
- ◆ The **types of RAM** are DRAM, SRAM, SDRAM and DDR-SDRAM.
- ◆ **Power supply** is essential for the computer to prevent computers from failures, breakdown or shutdown.
- ◆ **Types of power supply** connected to a computer system are SMPS or UPS

Review questions

One marks questions:

1. What is a motherboard?
2. What is microprocessor?
3. What is the purpose of registers in the CPU?
4. How does the computer communicate with other devices?
5. What is system bus?
6. What is the function of control bus?
7. What is a data bus?
8. What is a port?
9. What is an interface?
10. Expand PCI.
11. How many bits of data are sent in a serial port?
12. Expand USB.
13. Give one feature of USB port.
14. What is meant by plug and play device?
15. Name any one USB device.
16. Is device controller a hardware or software?
17. What is cache memory?
18. Where is L1 cache located?
19. Where is L2 cache located?
20. Expand SDRAM.
21. Give the expansion of DDRRAM.
22. Expand SMPS.
23. What is the use of SMPS?
24. What is the approximate power consumed by a PC?
25. Expand UPS.
26. What is the use of UPS?
27. List the types of UPS.

Two marks questions:

28. Name any two types of motherboard.
29. Mention any two characteristics of motherboard.
30. Mention the components of motherboard.
31. Explain system bus.
32. What is data bus and address bus?
33. What is the purpose of expansion slot?
34. What is the purpose of AGP slot?
35. Name the different types of I/O ports.
36. Explain serial port.

37. Explain parallel port.
38. Explain USB port.
39. What is meant by plug and play card?
40. What is the purpose of ports and buses?

Three marks question questions:

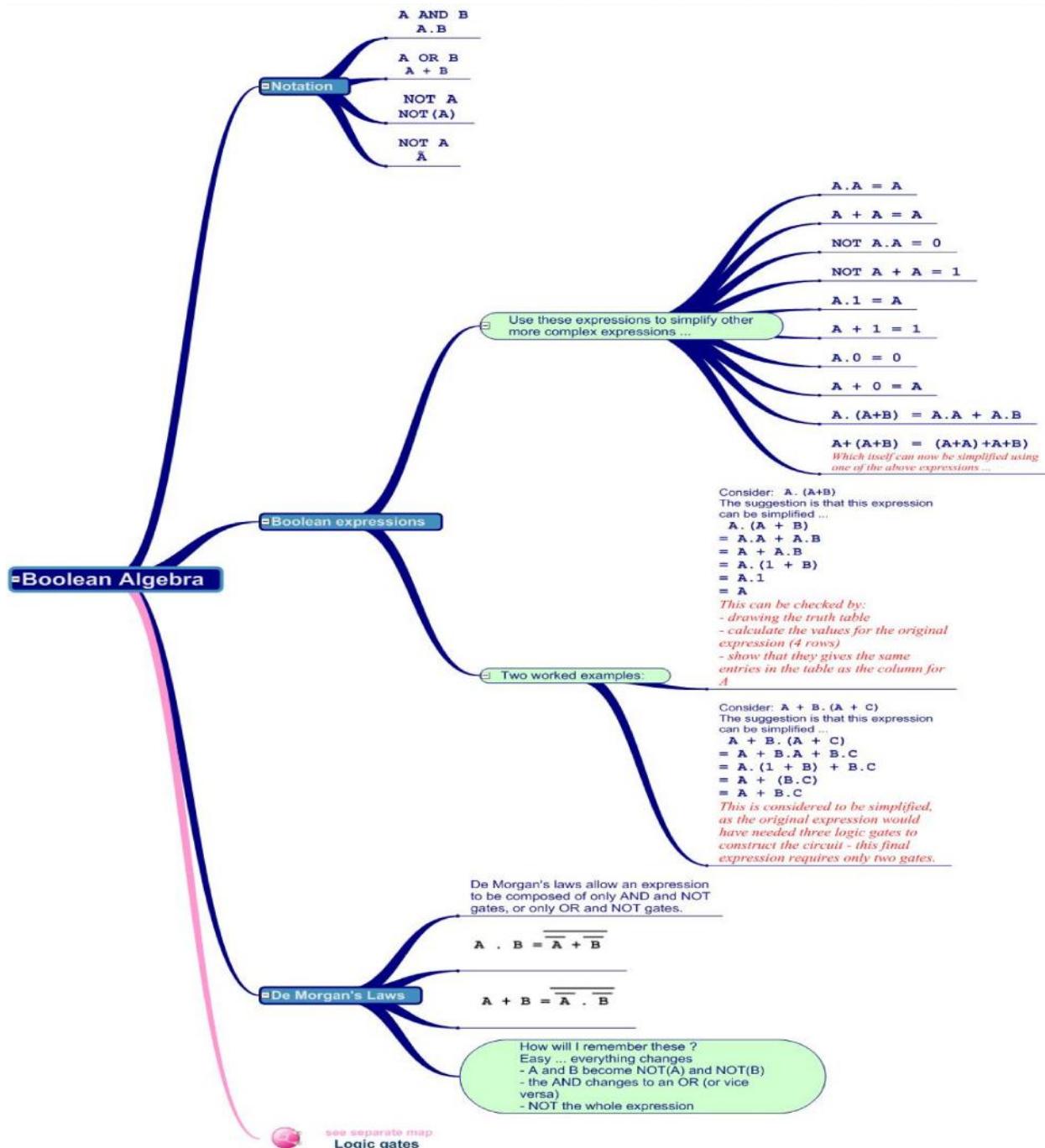
41. Explain the different components of motherboard.
42. Explain the characteristics of motherboard.
43. Explain the Schematic diagram of Motherboard.
44. Explain different types of I/O ports.
45. Give the features of USB port.
46. Explain cache memory.
47. Explain the types of power supply.
48. What is the purpose of ports, buses and controllers in the I/O system?
49. What is a Slot? mention any two types.
50. Name the different components of North bridge.

Chapter 2

Boolean Algebra

Objectives:

- To understand the concept of boolean algebra
- To understand the concept of simplifications of boolean expressions



2.1 Introducton to Boolean Algebra

In the previous course, we have seen that computers normally use binary numbers. In this chapter, you will learn about an algebra that deals with the binary number system. This algebra, known as Boolean algebra, is very useful in designing logic circuits used by the processors of computer systems. In addition to this, you will also learn about the elementary logic gates that are used to build up circuits of different types to perform the necessary arithmetic operations. These logic gates are the building blocks of all the circuits in a computer. Finally, in this chapter, we will also learn how to use Boolean Algebra to design simple logic circuits frequently used by the arithmetic logic unit of almost all computers.

Long ago Aristotle constructed a complete system of formal logic and wrote six famous works on the subject, contributing greatly to the organization of man's reasoning. For centuries afterward, mathematicians kept on trying to solve these logic problems using conventional algebra but only George Boole could manipulate these symbols successfully to arrive at a solution with his own mathematical system of logic. Boole's revolutionary paper 'An Investigation of the laws of the thought' was published in 1854 which led to the development of new system, the algebra of logic, 'BOOLEAN ALGEBRA'.

Boole's work remained confined to papers only until 1938 when Claude E. Shannon wrote a paper titled A Symbolic Analysis of Relay Switching Circuits. In this paper he applied Boolean Algebra to solve relay problems. As logic problems are binary decisions and Boolean Algebra effectively deals with these binary values. Thus it is also called 'Switching Algebra'.

2.2 Binary Valued Quantities - Variable and Constants

Everyday we have to make logic decisions. For example, consider the following questions:

"Should I carry the book or not?"

"Should I use calculator or not?"

"should I miss TV programme or not?"

Each of these questions requires the answer YES or NO. These are the only two possible answers.

Therefore, each of the above mentioned is a binary decision. Binary decision making also applies to formal logic.

A variable used in an algebraic formula is generally assumed that the variable may take any numerical value through the entire field of real numbers. However a variable used in Boolean Algebra or Boolean equation can have only one of two possible values. The two values are FALSE (or 0) and TRUE (or 1). Thus, sentences which can be determined to be TRUE or FALSE are called logical statements or truth functions and the results TRUE or FALSE are called truth

values. The truth values are depicted by logical constants TRUE and FALSE or 1 and 0 respectively. 1 means TRUE and 0 means FALSE. The variables which can store these truth values are called logical variables or binary valued variables as these can store one of the two values 1 or 0 (TRUE or FALSE).

The decision which results into either YES (TRUE or 1) or NO (FALSE or 0) is called a Binary Decision.

Also, if an equation describing logical circuitry has several variables, it is still understood that each of the variables can assume only the values 0 and 1. For instance, in the equation $A + B = C$, each of the variables A, B and C may have only the values 0 or 1.

2.3.0 LOGICAL OPERATIONS

There are some specific operations that can be applied on truth functions. Before learning about these operations, you must know about compound logical functions and logical operations.

2.3.1 Logical Function or Compound Statement

Algebraic variables like a, b, c or x, y, z etc. are combined with the help of mathematical operators like +, -, x, / to form algebraic expressions.

For example, $2 \times A + 3 \times B - 6 = (10 \times Z) / 2 \times Y$ i.e., $2A + 3B - 6C = 10Z/2Y$

Similarly, logic statements or truth functions are combined with the help of Logical Operators like AND, OR and NOT to form a compound statement or logical function.

These logical operators are also used to combine logical variables and logical constants to form logical expressions.

For example, assuming that x, y and z are logical variables, the logical expressions are

X NOT Y OR Z

Y AND X OR Z

2.3.2 Logical Operators

Truth Table is a table which represents all the possible values of logical variables/statements along with all the possible results for the given combinations of values.

Before we start discussion about logical operators, let us first understand what a Truth Table is ?. Logical statements can have only one of the two values TRUE (YES or 1) or FALSE (NO or 0).

For example, if X and Y are the logical statements and R is the result, then the truth table can be written as follows:

X	Y	R
0	0	0
0	1	0
1	0	0
1	1	1

Table 1.1

If result of any logical statement or expression is always TRUE or 1, it is called **Tautology** and if the result is always FALSE or 0 it is called **Fallacy**.

1 represents TRUE value and 0 represents FALSE value.

This is a truth table i.e., table of truth values of truth functions.

Now let us proceed with our discussion about logical operators. There are three logical operators: NOT, OR and AND Operators.

NOT Operator

This operator operates on single variable and operation performed by NOT operator is called **complementation** and the symbol we use for it is $\bar{}$ (bar). Thus \bar{X} means complement of X and \bar{YZ} means complement of YZ. As we know, the variables used in Boolean equations have a unique characteristic that they may assume only one of two possible values 0 and 1, where 0 denotes FALSE and 1 denotes TRUE value. Thus the complement operation can be defined quite simply.

$$\bar{0} = 1 \quad \text{or} \quad \text{NOT (FALSE)} = \text{TRUE and}$$

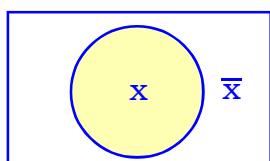
$$\bar{1} = 0 \quad \text{or} \quad \text{NOT (TRUE)} = \text{FALSE and}$$

The truth table for the NOT operator is

X	\bar{X}
0	1
1	0

Several other symbols like ' \sim ' are also used for the complementation symbol. If \sim is used then $\sim X$ is read as 'negation of X' and if symbol ' $'$ is used then X' is read as complement of X.

Table 1.2 Truth Table for NOT operator



NOT operation is singular or unary operation as it operates on single variable.

Venn diagram for \bar{x} is given above where shaded area depicts \bar{x} .

Figure 1.3. Venn diagram for \bar{x}

OR operator

A second important operator in Boolean algebra is OR operator which denotes operation called **logical addition** and the symbol we use for it is +. The + symbol, therefore, does not mean arithmetic addition, but is a **logical addition** or **logical OR symbol**. Thus, $X + Y$ can be read as X OR Y. For OR operation, the possible input and output combinations are as follows :

$0 + 0 = 1$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 1$

The truth table of OR operator is given below:

X	Y	$X+Y$
0	0	0
0	1	1
1	0	1
1	1	1

Note that when any one or both X and Y is 1, $X + Y$ is 1.

and both X and Y is 0, $X+Y$ is 0

Table 1.4 : Truth Table for OR operator

To avoid ambiguity, there are other symbols e.g., \cup and \vee have been recommended as replacements for the + sign. Computer people still use the + sign, however, which was the symbol originally proposed by Boole.

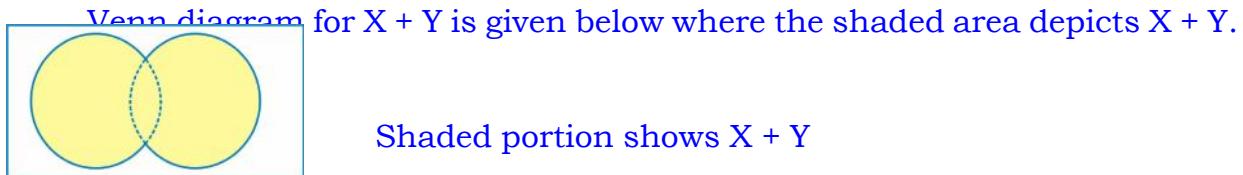


Figure 1.2 : Venn diagram for $X+Y$

AND Operator

AND operator performs another important operation of Boolean Algebra called **logical multiplication** and the symbol for AND operation is ‘.’ (dot). Thus $X.Y$ will be read as X AND Y. The rules for AND operation are :

$0.0 = 0$
$0.1 = 0$
$1.0 = 0$
$1.1 = 1$

And the truth table for AND is as follows :

X	Y	X.Y
0	0	0
0	1	0
1	0	0
1	1	1

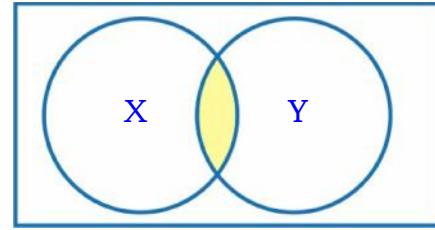


Table 1.10 : Truth Table for AND operator

FIGURE 1.3 : Venn Diagram for (X.Y)

Note that only when both X and Y are 1's, X.Y has the result 1. If any one of X and Y is 0, XY result 0. Venn diagram for X.Y is given in the figure above where the shaded area depicts X.Y

2.4.0 Evaluation of Boolean Expressions Using Truth Table

Logical variables are combined by means of logical operators AND, OR and NOT to form a Boolean expression. For example, $X+Y.Z+\bar{Z}$ is a Boolean expression.

It is often convenient to shorten $X.Y.Z$ to XYZ and using this convention, above expression can be written as $X+\bar{Y}Z+\bar{Z}$

To study a Boolean expression, it is very useful to construct a table of values for the variables and then to evaluate the expression for each of the possible combinations of variables in turn. Consider the expression $X+\bar{Y}Z$. Here three variables X, Y, Z are forming the expression. Each variable can assume the value 0 or 1. The possible combinations of values may be arranged in ascending order as in Table 1.11

X	Y	Z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Since X, Y, and Z are the three (3) variables in total. A truth table involving 3 input variables will have $2^3 = 8$ rows or combinations in total. The left most column will have half of total entries (4 entries) as zeroes and half as 1's (in total 8). The next column will have number of 0's and 1's halved than first column completing 8 rows and so on. That is why, first column has four 0's and four 1's, next column has two 0's followed by two 1's completing 8 rows in total and the last column has one 0 followed by one 1 completing 8 rows in total.

Table 1.11 Possible Combinations of X, Y and Z

So a column is added to list Y.Z (Table 1.12)

X	Y	Z	Y.Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

AND operation is applied only on columns Y and Z.

Table 1.12 Truth Table for (Y.Z)

One more column is now added to list the values of \overline{YZ} (Table 1.13)

X	Y	Z	Y.Z	\overline{YZ}
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

Note that \overline{YZ} contains complemented values of YZ.

Table 1.13 truth table for YZ and \overline{YZ}

Now values of X are ORed (logical addition) to the values of \overline{YZ} and the resultant values are contained in the last column (Table 1.14).

X	Y	Z	Y.Z	\overline{YZ}	$X + \overline{YZ}$
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	0	1

Now observe the expression $X + \overline{YZ}$, after ANDing Y and Z, the result has been complemented and then ORed with X. Here the result is 0 only when both the columns X and \overline{YZ} have 0, otherwise if there is 1 in any of the two columns X and \overline{YZ} , the result is 1.

Table 1.14 Truth Table for $X + \overline{YZ}$.

A Boolean expression will be evaluated using precedence rules. The order of evaluation of an expression is called as precedence. The precedence is, firstly NOT, then AND and then OR. If there is parenthesis, then the expression in parenthesis is evaluated first.

Example 1.13: In the Boolean algebra, verify using truth table that $X+XY = X$ for each X, Y in 0 and 1.

As the expression $X+XY=X$ is a two variable expression, so we require four possible combinations of values of X, Y . Truth Table will be as follows:

X	Y	XY	X+XY
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Comparing the columns $X+XY$ and X , we find, contents of both the columns are identical, hence verified.

Example 1.14: In the Boolean Algebra, verify using truth table that

$$\overline{X+Y} = \overline{X} \cdot \overline{Y} \text{ in } 0 \text{ and } 1.$$

Solution: As it is a 2-variable expression, truth table will be as follows:

X	Y	X+Y	$\overline{X+Y}$	\overline{X}	\overline{Y}	$\overline{X} \cdot \overline{Y}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Comparing the columns $\overline{X+Y}$ and $\overline{X} \cdot \overline{Y}$ both the columns are identical, hence verified.

Example 1.15: Prepare a table of combinations for the following Boolean algebra expressions:

$$(a) \overline{X} \overline{Y} + \overline{X} Y \quad (b) XY\overline{Z} + \overline{X}\overline{Y}Z \quad (c) \overline{X} Y \overline{Z} + X \overline{Y}$$

Solution: (a) As $\bar{X}\bar{Y} + \bar{X}Y$ is a 2-variable expression, its truth table is as follows:

X	Y	\bar{X}	\bar{Y}	$\bar{X}\bar{Y}$	$\bar{X}Y$	$\bar{X}\bar{Y} + \bar{X}Y$
0	0	1	1	1	0	1
0	1	1	0	0	1	1
1	0	0	1	0	0	0
1	1	0	0	0	0	0

(b) Truth table for this 3 variable expression is as follows :

X	Y	Z	\bar{X}	\bar{Y}	\bar{Z}	XYZ	$\bar{X}\bar{Y}Z$	$XYZ + \bar{X}\bar{Y}Z$
0	0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	1	1
0	1	0	1	0	1	0	0	0
0	1	1	1	0	0	0	0	0
1	0	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0	0
1	1	0	0	0	1	1	0	1
1	1	1	0	0	0	0	0	0

(a) Truth table for $\bar{X}YZ + XY$ is as follows:

X	Y	Z	\bar{X}	\bar{Y}	\bar{Z}	$\bar{X}YZ$	XY	$\bar{X}YZ + XY$
0	0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0	0
0	1	0	1	0	1	1	0	1
0	1	1	1	0	0	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	0	1	0	0	1	1
1	1	0	0	0	1	0	0	0
1	1	1	0	0	0	0	0	0

Example 1.16 Prepare truth table for the following Boolean algebra expressions:

(a) $X(\bar{Y}+\bar{Z}) + X\bar{Y}$ (b) $X\bar{Y}(\bar{Z}+Y\bar{Z}) + \bar{Z}$ (c) $A[(\bar{B}+C)+\bar{C}]$

Solution (a) Truth table for $X(+)$ $+X$ is as follows :

X	Y	Z	\bar{Y}	\bar{Z}	$(\bar{Y}+\bar{Z})$	$X(\bar{Y}+\bar{Z})$	$X\bar{Y}$	$X(\bar{Y}+\bar{Z})+X\bar{Y}$
0	0	0	1	1	1	0	0	0
0	0	1	1	0	1	0	0	0
0	1	0	0	1	1	0	0	0
0	1	1	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	1	1
1	1	0	0	1	1	1	0	1
1	1	1	0	0	0	0	0	0

(b) Truth table for $X\bar{Y}$ $(\bar{Z}+Y)+$ is as follows:

X	Y	Z	\bar{Y}	\bar{Z}	YZ	$Z+YZ$	$X\bar{Y}$	$X\bar{Y}(Z+YZ)$	$X\bar{Y}(Z+YZ)+\bar{Z}$
0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	0	1	0	0	0
0	1	0	0	1	1	1	0	0	1
0	1	1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	1	0	1
1	0	1	1	0	0	1	1	1	1
1	1	0	0	1	1	1	0	0	1
1	1	1	0	0	0	1	0	0	0

(c) Truth table for $A[(\bar{B}+C)+\bar{C}]$ is as follows :

A	B	C	\bar{B}	\bar{C}	$(\bar{B}+C)$	$(\bar{B}+C)+\bar{C}$	$A[(\bar{B}+C)+\bar{C}]$
0	0	0	1	1	1	1	0
0	0	1	1	0	1	1	0
0	1	0	0	1	0	1	0
0	1	1	0	0	1	1	0
1	0	0	1	1	1	1	1
1	0	1	1	0	1	1	1
1	1	0	0	1	0	1	1
1	1	1	0	0	1	1	1

2.4.1 BASIC LOGIC GATES

After Shannon applied Boolean algebra in telephone switching circuits, engineers realized that Boolean algebra could be applied to computer electronics as well.

In the computers, these Boolean operations are performed by logic gates.

What is a Logic Gate?

Gates are digital (two-state) circuits because the input and output signals are either low voltage (denotes 0) or high voltage (denotes 1). Gates are often called logic circuits because they can be analyzed with Boolean algebra.

A Gate is simply an electronic circuit which operates on one or more input signals to produce an output signal.

There are three types of logic gates:

- NOT gate or Inverter
- OR gate
- AND gate

Inverter (NOT Gate)

An inverter (NOT Gate) is a gate with only one input signal and one output signal. The output state is always the opposite of the input state.

An inverter is also called a NOT gate because the output is not the same as the input. The output is complement (opposite) of the input. Following tables summarizes the operation:

X	\bar{X}
Low	High
High	Low

X	\bar{X}
0	1
1	0

Table 1.15 Truth Table for NOT gate

Table 1.16 Alternative truth table for NOT gate

A low input or 0 produces high output or 1 and vice versa. The symbol for inverter is given in adjacent Fig. 1.4.

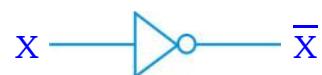


Fig. 1.4. Not gate symbol

NOT Gate is a gate or an electronic circuit that accepts only one input and produces one output signal. The output state is always the complement of the input state.

OR Gate

The OR Gate has two or more input signals, but only one output signal. This gate gives the logical addition of the inputs. If any of the input signals or both is 1 (high), the output signal is 1 (high). The output will be low if all the inputs are low.

An OR gate can have as many inputs as desired. No matter how many inputs are there, the action of OR gate is the same.

The OR gate has two or more input signals, but only one output signal. The output will be the logical addition of the inputs.

Following tables show OR action

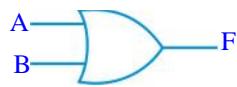
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

Table : $F=X+Y$

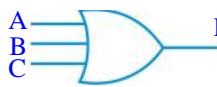
X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Table : $F=X+Y+Z$

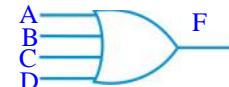
The symbol for OR gate is given below:



a) 2 input OR gate



b) 3 input OR gate



c) 4 input OR gate

Figure 1.5

AND gate

The AND Gate can have two or more than two input signals and produce an output signal. When all the inputs are 1 or high only then the output is 1, otherwise output is 0 only.

If any one or all the inputs is 0, the output is 0. To obtain output as 1, all inputs must be 1.

An AND gate can have as many inputs as desired.

The AND Gate has two or more input signals, but only one output signal. The output will be the logical multiplication of the inputs.

Following tables illustrate AND action.

X	Y	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Table 1.19 Two input AND gate

X	Y	Z	X.Y.Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Table 1.20 Three input AND gate

The symbol for AND is

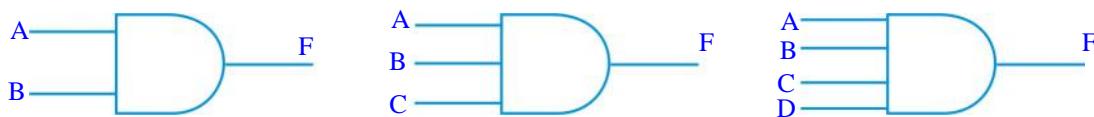


Figure 1.6 (a) 2-input AND gate (b) 3-input AND gate (c) 4 input AND gate

2.5 BASIC POSTULATES OF BOOLEAN ALGEBRA

Boolean algebra is a system of mathematics and consists of fundamental laws. These fundamental laws are used to build a workable, cohesive framework upon which are based the theorems of Boolean algebra. These fundamental laws are known as Basic Postulates of Boolean algebra. These postulates state the basic relations in Boolean algebra:

The fundamental laws of the Boolean algebra are called as the postulates of Boolean algebra

The Boolean postulates are:

- I. If $X \neq 0$ then $X = 1$; and If $X \neq 1$ then $X = 0$
- II. OR Relations (Logical Addition)

$$0+0=0$$



$$0+1=1$$



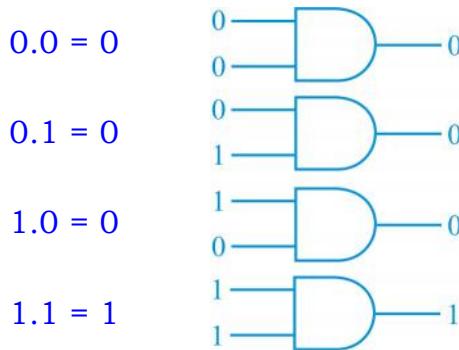
$$1+0=1$$



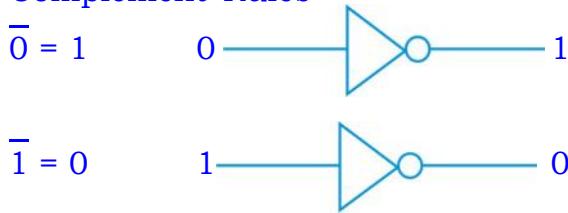
$$1+1=1$$



III AND Relations (Logical Multiplication)



IV Complement Rules



PRINCIPLE OF DUALITY

This is a very important principle used in Boolean algebra. This states that starting with a Boolean relation another Boolean relation can be derived by

- i. Changing each OR sign (+) to an AND sign (.)
- ii. Changing each AND sign (.) to an OR sign (+)
- iii. Changing each 0 by 1 and each 1 by 0.

The derived relation using duality principle is called dual of original expression.

For instance, we take postulates of OR relation, which states that

$$(a) 0 + 0 = 0 \quad (b) 0 + 1 = 1 \quad (c) 1 + 0 = 1 \quad (d) 1 + 1 = 1$$

Now working according to above guidelines, '+' is changed to '.' 0's are replaced by 1's and 1's are replaced by 0's, these equations become

$$(i) 1 \cdot 1 = 1 \quad (ii) 1 \cdot 0 = 0 \quad (iii) 0 \cdot 1 = 0 \quad (iv) 0 \cdot 0 = 0$$

These are nothing but postulate III related to AND relations. We'll be applying this duality principle in the theorems of Boolean algebra.

Basic theorems of Boolean algebra

Basic postulates of Boolean algebra are used to define basic theorems of Boolean algebra that provide all the tools necessary for manipulating Boolean expressions. Although simple in appearance, these theorems may be used to construct the Boolean algebra expressions.

Boolean theorems can be proved by substituting all possible values of the variables that are 0 and 1. This technique of proving theorems is called as **proof by perfect induction**. Boolean theorems can also be proved using truth table also.

Proof by perfect induction is a method of proving Boolean theorems by substituting all possible values of the variables.

2.5.1 Properties of 0 and 1

- a) $0+X=X$  (gate representation of (a))
- b) $1+X=1$  (gate representation of (b))
- c) $0.X=0$  (gate representation of (c))
- d) $1.X=X$  (gate representation of (d))

Proof a) $0+x = x$

$$\begin{aligned}
 \text{If } x = 0, \text{ then } LHS &= 0 + x \\
 &= 0 + 0 \\
 &= 0 \quad \{ \text{By OR relation} \} \\
 &= x \\
 &= RHS
 \end{aligned}$$

$$\begin{aligned}
 \text{If } x = 1, \text{ then } LHS &= 0 + x \\
 &= 0 + 1 \\
 &= 1 \quad \{ \text{By OR relation} \} \\
 &= x \\
 &= RHS
 \end{aligned}$$

Thus, for every value of x , $0 + x = x$ always.

O	x	$R=0+x$
0	0	0
0	1	1

Truth table for above expression is given in table 1.21, where R signifies the output.

Table 1.21 Truth Table for $0 + x = x$

As X can have values either 0 or 1 (postulate 1) both the values ORed with 0 produce the same output as that of X. hence proved.

(a) $1 + x = 1$

Proof: If $x = 0$,

$$\begin{aligned} \text{LHS} &= 1 + x \\ &= 1 + 0 \\ &= 1 \quad \{ \text{By OR relation} \} \end{aligned}$$

If $x = 1$,

$$\begin{aligned} \text{LHS} &= 1 + x \\ &= 1 + 1 \\ &= 1 \quad \{ \text{By OR relation} \} \end{aligned}$$

Thus, for every value of x , $1 + x = 1$ always.

Truth table for above expression is given below in Table 1.21, where R signifies the output or result.

1	x	$1 + x$
1	0	1
1	1	1

Table 1.22 Truth Table for $1 + x = 1$

Again x can have values 0 or 1. Both the values (0 and 1) ORed with 1 produce the output as 1. Therefore $1+X=1$ is a tautology.

(a) $0 \cdot x = 0$

Proof: If $x = 0$,

$$\begin{aligned} \text{LHS} &= 0 \cdot x \\ &= 0 \cdot 0 \\ &= 0 \quad \{ \text{By AND relation} \} \\ &= \text{RHS} \end{aligned}$$

$$\begin{aligned}
 \text{If } x = 1, \quad \text{LHS} &= 0.x \\
 &= 0.1 \\
 &= 0 \quad \{ \text{By AND relation} \} \\
 &= \text{RHS}
 \end{aligned}$$

Thus, for every value of x , $0.x = 0$ always.

As both the possible values of X (0 and 1) are to be ANDed with 0, produce the output as 0. The truth table for this expression is as follows:

0	X	R=0.X
0	0	0
0	1	0

Table 1.23 Truth Table for $0.X = 0$

Both the values of X (0 and 1), when ANDed with, produce the output as 0. Hence proved. Therefore, $0.X=0$ is a fallacy.

(d) $1.X = X$

$$\begin{aligned}
 \text{Proof: If } x = 0, \quad \text{LHS} &= 1.x \\
 &= 1.0 \\
 &= 0 \quad \{ \text{By AND relation} \} \\
 &= x \\
 &= \text{RHS}
 \end{aligned}$$

$$\begin{aligned}
 \text{If } x = 1, \quad \text{LHS} &= 1.x \\
 &= 1.1 \\
 &= 1 \quad \{ \text{By AND relation} \} \\
 &= y \\
 &= \text{RHS}
 \end{aligned}$$

Thus, for every value of x , $1.x = x$ always.

Now both the possible values of X (0 and 1) are to be ANDed with 1 to produce the output R. Thus the truth table for it will be as follows :

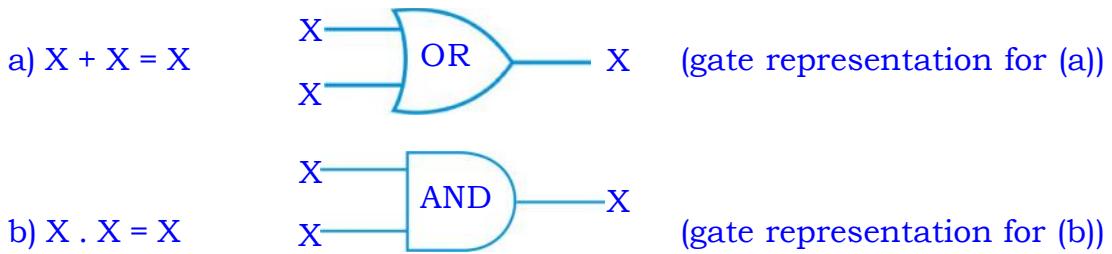
1	X	1.X
1	0	0
1	1	1

Table 1.24 : Truth Table for $1.X=X$

Now observe both the values (0 and 1) when ANDed with 1 produce the same output as that of X. Hence proved.

2.5.2 Indempotence Law

This law states that when a variable is combines with itself using OR or AND operator, the output is the same variable.



Proof :

$$(a) \quad X + X = X$$

$$\begin{aligned} \text{If } x = 0, \text{ consider LHS} &= x + x \\ &= 0 + 0 \\ &= 0 \quad \{ \text{By OR relation} \} \\ &= x \\ &= \text{RHS} \end{aligned}$$

$$\begin{aligned} \text{If } x = 1, \text{ consider LHS} &= x + x \\ &= 1 + 1 \\ &= 1 \quad \{ \text{By OR relation} \} \\ &= x \\ &= \text{RHS} \end{aligned}$$

Thus, for every value of x, $x + x = x$ always.

To prove this law, we will make truth table for above expression. As X is to be ORed with itself only, we will prepare truth table with the two possible values of X (0 and 1).

X	X	X+X
0	0	0
1	1	1

Table 1.25 Truth Table for $x + x = x$

(b) $X \cdot X = X$

Here X if ANDed with itself.

Proof: If $x = 0$, consider

LHS	= $x \cdot x$
	= 0.0
	= 0
	{ By AND relation }
	= x
	= RHS

If $x = 1$,

LHS	= $x \cdot x$
	= 1.1
	= 1
	{ By AND relation }
	= x
	= RHS

Thus, for every value of x, $x + x = x$ always.

x	x	$x \cdot x$
0	0	0
1	1	1

Table 1.26 Truth Table for $X \cdot X = X$

2.5.3 Involution

This law states that the complement of a variable is complemented again, we get the same variable.

$$\overline{\overline{X}} = X \quad \text{ie., } X \rightarrow \overline{X} \rightarrow \overline{\overline{X}} = X$$

Proof: If $x = 0$, then $\bar{x} = 1$ and $\overline{\overline{x}} = \bar{\bar{1}} = 0 = x$ If $x = 1$, then $\bar{x} = 0$ and $\overline{\overline{x}} = \bar{\bar{0}} = 1 = x$

Thus, if a variable is complemented twice, we get the same variable.

We'll prepare truth table which is given below:

X	\bar{X}	$\overline{\overline{X}}$
0	1	0
1	0	1

Table 1.27 Truth Table for $\overline{\overline{X}}=X$

First column represents possible values of X, second column represents complement of X (i.e., \bar{X}) and the third column represents complement of \bar{X} (i.e., $\overline{\overline{X}}$) which is same as that of X. Hence proved.

This law is also called double-inversion rule.

2.5.4 Complementarity Laws

Here, we will combine a variable with its complement.

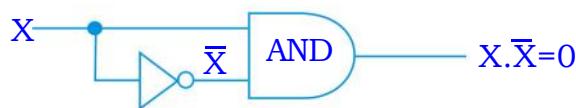
- i. These laws states that

a) $X + \bar{X} = 1$



(gate representation of (a))

b) $X \cdot \bar{X} = 0$



Proof: If $x = 0$, LHS $= x + \bar{x}$

$$= 0 + 1 \quad (\bar{x} = 1)$$

$$= 1 \quad \{ \text{By OR relation} \}$$

$$= \text{RHS}$$

If $x = 1$, LHS $= x + \bar{x}$

$$= 1 + 0$$

$$= 1 \quad \{ \text{By OR relation} \}$$

$$= \text{RHS}$$

Thus, for every value of x, $x + \bar{x} = 1$ always.

We will prove $x + \bar{x}=1$ with the help of truth table which is given below :

X	\bar{X}	$X + \bar{X}$
0	1	1
1	0	1

Table 1.28 Truth Table for $X + \bar{X} = 1$

Here, in the first column possible values of X have been taken, second column consists of \bar{X} values (complement values of X), X and \bar{X} values are ORed and the output is shown in third column. As the equation holds true for both possible values of X, it is a **tautology**.

$$(b) \quad X \cdot \bar{X} = 0$$

Proof: If $x = 0$, LHS $= x \cdot \bar{x}$

$$= 0 \cdot 1 \quad (\bar{x} = 1)$$

$$= 0 \quad \{ \text{By AND relation} \}$$

If $x = 1$, LHS $= x \cdot \bar{x}$

$$= 1 \cdot 0 \quad (\bar{x} = 0)$$

$$= 0 \quad \{ \text{By AND relation} \}$$

Thus, for every value of x, $x \cdot \bar{x} = 0$ always.

Truth table for the expression is as follows:

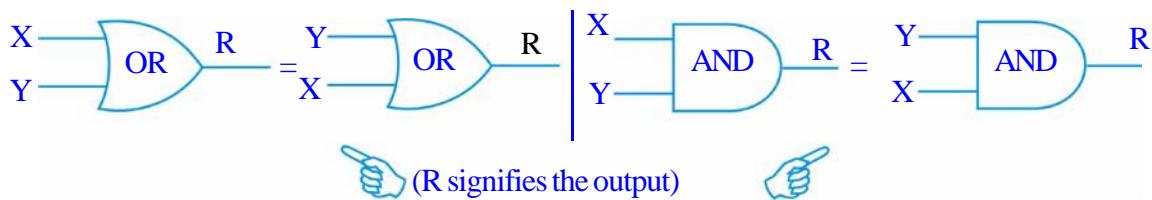
X	\bar{X}	$X \cdot \bar{X}$
0	1	0
1	0	0

Table 1.29 Truth table for $X \cdot \bar{X} = 0$

The equations $X \cdot X = 0$ as it holds true for both the values of X. Hence proved. Observe that $X \cdot X = 0$. It is a fallacy. It is the dual of $X + \bar{X} = 1$.

2.5.5 Commutative Law

These laws state that a) $x + y = y + x$ and b) $x \cdot y = y \cdot x$



$$\begin{aligned} \text{If } x = 0 \quad \text{then LHS} &= x + y \\ &= 0 + y \\ &= y \end{aligned}$$

$$\begin{aligned} \text{RHS} &= y + x \\ &= y + 0 \\ &= y \end{aligned}$$

Therefore, for $x = 0$, $x + y = y + x$

$$\begin{aligned} \text{If } x = 1 \quad \text{then LHS} &= x + y \\ &= 1 + y \\ &= 1 \end{aligned}$$

$$\begin{aligned} \text{RHS} &= y + x \\ &= y + 1 \\ &= 1 \end{aligned}$$

Therefore, for $x = 1$, $x + y = y + x$. Hence the proof.

X	Y	X+Y	Y+X
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Table 1.30 Truth Table for $X + Y = Y + X$

Compare the columns $X + Y$ and $Y + X$, both of these are identical. Hence also proved by truth table.

(b) Truth Table for $X \cdot Y = Y \cdot X$ is given below:

Proof: If $x = 0$ then LHS = $x \cdot y$

$$\begin{aligned} &= 0 \cdot y \\ &= 0 \end{aligned}$$

$$\begin{aligned} \text{RHS} &= y \cdot x \\ &= y \cdot 0 \\ &= 0 \end{aligned}$$

Therefore, for $x = 0$, $x \cdot y = y \cdot x$

If $x = 1$ then LHS = $x \cdot y$

$$= 1 \cdot y$$

$$= y$$

Therefore, for $x = 1$, $x + y = y + x$. Hence the proof.

X	Y	X.Y	Y.X
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

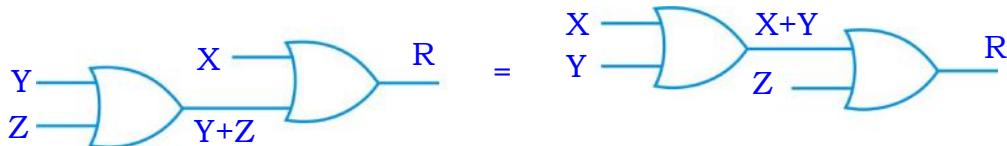
Table 1.31 Truth Table for $X \cdot Y = Y \cdot X$

Both of the columns $X \cdot Y$ and $Y \cdot X$ are identical, hence proved.

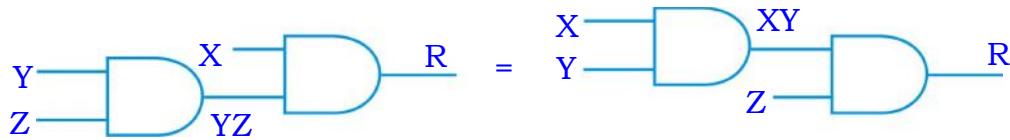
2.5.6 Associative Law

These laws state that

(a) $X + (Y + Z) = (X + Y) + Z$ (associative Law of addition)



(b) $X(YZ) = (XY)Z$ (associative Law of multiplication)



a) $X + (Y + Z) = (X + Y) + Z$

Proof: If $X = 0$ then LHS = $X + (Y + Z)$

$$= 0 + (Y + Z)$$

$$= Y + Z$$

$$\text{RHS} = (X + Y) + Z$$

$$= (0 + Y) + Z$$

$$= Y + Z$$

Therefore for $X=0$, $X + (Y + Z) = (X + Y) + Z$

$$\text{If } X=1, \text{ then LHS} = X + (Y + Z)$$

$$= 1 + (Y + Z)$$

$$= 1$$

$$\text{RHS} = (X + Y) + Z$$

$$= 1 + (Y + Z)$$

$$= 1 + Z$$

$$\text{Therefore } X=1, X + (Y + Z) = (X + Y) + Z$$

$$= 1$$

Proof. (a) Truth table for $X + (Y + Z) = (X + Y) + Z$ is given below :

X	Y	Z	$Y+Z$	$X+Y$	$X+(Y+Z)$	$(X+Y)+Z$
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

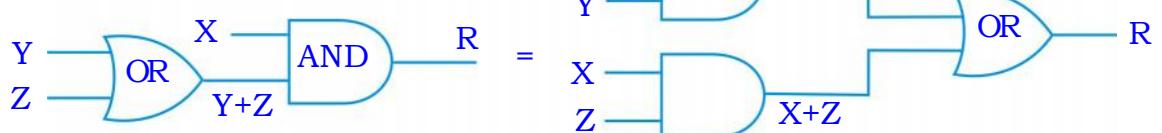
(a) Table 1.32 Truth Table for $X + (Y + Z) = (X + Y) + Z$

Compare the columns $X+(Y+Z)$ and $(X+Y)+Z$, both of these are identical. Hence proved. Note : Give proof with table for rule (b). Since rule (b) is a dual of rule (a), hence it is also proved.

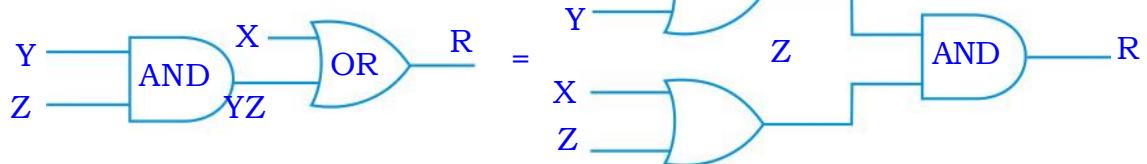
2.5.7 Distributive Law

This law states that

(a) $X(Y + Z) = XY + XZ$



(a) $X + YZ = (X+Y)(X+Z)$



Proof: a) $X(Y+Z) = XY + XZ$

$$\begin{aligned} \text{If } X=0, \quad \text{LHS} &= X(Y+Z) \\ &= 0(Y+Z) \\ &= 0 \end{aligned}$$

$$\begin{aligned} \text{RHS} &= XY + XZ \\ &= 0.Y + 0.Z \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

$$\begin{aligned} \text{If } X=1, \quad \text{LHS} &= X(Y+Z) \\ &= 1(Y+Z) \\ &= Y+Z \end{aligned}$$

$$\begin{aligned} \text{RHS} &= XY + XZ \\ &= 1.Y + 1.Z \\ &= Y + Z \end{aligned}$$

Therefore, for every value of x, LHS = RHS. i.e., $x(y+z) = xz + yz$

Truth Table for $X(Y+Z) = XY+XZ$ is given below:

Table 1.33 Truth table for $X(Y+Z) = XY+XZ$

X	Y	Z	Y+Z	XY	XZ	X(Y+Z)	XY+XZ
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	0	1	1	1
1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1

Both the columns $X(Y+Z)$ and $XY+XZ$ are identical, hence proved.

Note : Since rule (b) is dual of rule (a), hence it is also proved

$$(b) X + YZ = (X+Y)(X+Z)$$

Proof: RHS = $(X+Y)(X+Z)$

$$\begin{aligned} &= XX + XZ + XY + YZ \\ &= X+XZ+XY+YZ \quad (\because XX=X) \end{aligned}$$

$$= X(1 + Z + Y) + YZ$$

$$= X + YZ \quad (1 + z + y = 1)$$

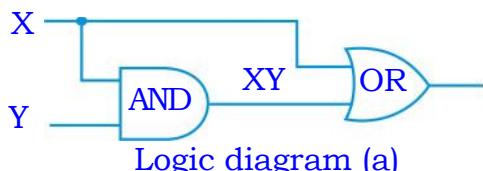
= LHS Hence the proof

Truth table for $X+YZ = (X+Y)(X+Z)$ is given below

2.5.8 Absorption Law

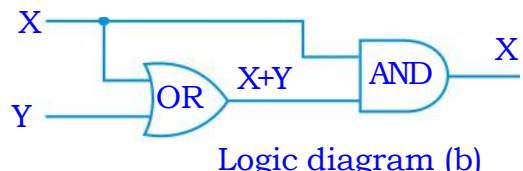
According to this law

$$a) X+XY=X$$



Logic diagram (a)

$$b) X(X+Y)=X$$



Logic diagram (b)

Proof: a) $X+XY = X$

$$\begin{aligned} \text{LHS} &= x + xy \\ &= x(1 + y) \\ &= x \cdot 1 \quad (\because 1+Y=1) \\ &= x \quad (\because X \cdot 1=X) \\ &= \text{RHS} \end{aligned}$$

Truth Table for $X+XY = X$ is given below:

X	Y	XY	$X+XY$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Table 1.34 : Truth Table for $X+XY = X$

Column X and $X+XY$ are identical. Hence proved

(b) Since rule (b) is dual of rule (a), it is also proved. However, we are giving the algebraic proof of this law.

$$\begin{aligned}
 \text{L.H.S.} &= X(X+Y) = X.X + XY \\
 &= X.X + XY \\
 &= X + XY && (\text{X.X} = \text{X Indempotence Law}) \\
 &= X(1+Y) \\
 &= X.1 && (\text{using } 1 + Y = 1 \text{ properties of 0, 1}) \\
 &= X && (X \cdot 1 = X \text{ using property of 0, 1}) \\
 &= \text{RHS}
 \end{aligned}$$

Truth table for $X(X+Y)=X$

X	Y	$X+Y$	$X(X+Y)$
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

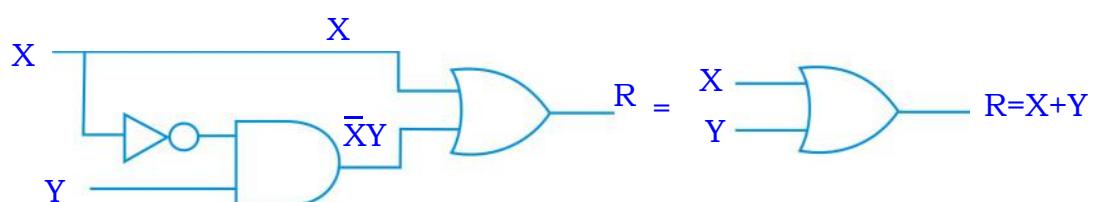
Some Other Rules of Boolean Algebra

There are some more rules of Boolean algebra which are given below:

$$X+\bar{X}Y=X+Y \text{ (This is the third distributive law)}$$

This rule can easily be proved by truth tables. As you are quite familiar with truth tables now, truth table proof is left for you as exercise, the other proofs of these rules are being given here:

$$X+\bar{X}Y=X+Y$$



Proof : $LHS = X + XY$

$$\begin{aligned}
 &= (x+x)(x+y) \quad \{ x+x= 1 \} \\
 &= 1.(x+y) \\
 &= x+y \\
 &= RHS
 \end{aligned}$$

All the theorems of Boolean algebra, which we have been covered so far, are summarized in the following table:

Table 1.35 Boolean algebra rules

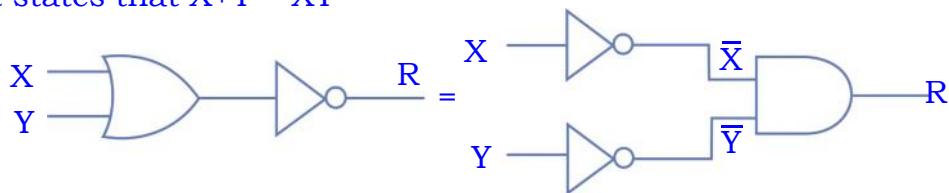
1	$0+X=X$	Properties of 0
2	$0.X = 0$	
3	$1+X=1$	Properties of 1
4	$1.X = X$	
5	$X + X = X$	Indempotence law
6	$X . X = X$	
7	$\overline{\overline{X}} = X$	Involution
8	$X + \overline{X} = 1$	Complementarity law
9	$X . \overline{X} = 0$	
10	$X + Y = Y + X$	Commutative law
11	$X . Y = Y . X$	
12	$X + (Y + Z) = (X+Y)+Z$	Associative law
13	$X(YZ) = (XY) Z$	
14	$X (Y+Z) = XY+XZ$	Distributive law
15	$X+YZ=(X+Y) (X+Z)$	
16	$X+XY=X$	Absorption law
17	$X . (X+Y) = X$	
18	$X+\overline{X}Y=X+Y$	

2.6 De Morgan's theorems

One of the most powerful identities used in Boolean algebra is De Morgan's theorem. Augustus De Morgan had paved the way to Boolean algebra by discovering these two important theorems. This section introduces these two theorems of De Morgan.

De Morgan's First Theorem

It states that $\overline{X+Y} = \overline{X}\overline{Y}$



Proof: To prove this theorem, we need to recall complementarity laws, which state that $X + \bar{X} = 1$ and $X \cdot \bar{X} = 0$ i.e., a logical variable/expression when added with its complement produces the output 1 and when multiplied with its complement produces the output 0.

Now to prove De Morgan's first theorem, we will use complementarity laws.

Let us assume that $P=X+Y$ where, P, X, Y are logical variables. Then, according to complementation law $P + \bar{P} = 1$ and $P \cdot \bar{P} = 0$.

That means, if P, X, Y are Boolean variables then this complementarity law must hold for variable P. i.e., $\bar{P} = \bar{X+Y} = \bar{X}\bar{Y}$

$$\text{Therefore } P + \bar{P} = (X+Y) + \bar{X}\bar{Y}$$

$$(X+Y) + \bar{X}\bar{Y} \text{ must be equal to 1. (As } X + \bar{X} = 1)$$

$$\text{And, } (X+Y) \cdot \bar{X}\bar{Y} \text{ must be equal to 0 (As } X \cdot \bar{X} = 0)$$

$$\text{Let us first prove the first part, i.e., } (X + Y) + (\bar{X}\bar{Y}) = 1$$

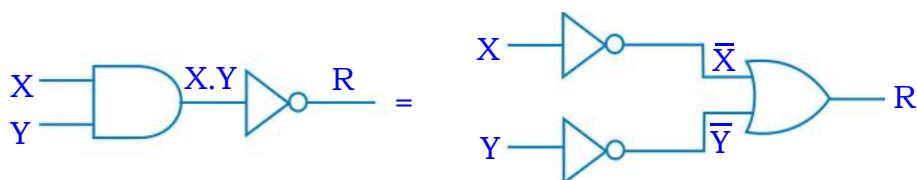
$$\begin{aligned} (X + Y) + (\bar{X}\bar{Y}) &= ((X + Y) + \bar{X}) \cdot ((X + Y) + \bar{Y}) && \text{(ref. } X+YZ=(X+Y)(X+Z)) \\ &= (X + \bar{X} + Y) \cdot (X + Y + \bar{Y}) \\ &= (1 + Y) \cdot (X + 1) \\ &= 1 \cdot 1 && \text{(ref. } X + \bar{X} = 1) \\ &= 1 \end{aligned}$$

$$\text{Now, let us prove the second part. i.e., } (X + Y) \cdot (\bar{X}\bar{Y}) = 0$$

$$\begin{aligned} (X + Y) \cdot (\bar{X}\bar{Y}) &= \bar{X}\bar{Y} \cdot (X + Y) && \text{(ref. } X(YZ)=((XY)Z)) \\ &= X\bar{X}\bar{Y} + \bar{X}Y\bar{Y} \\ &= 0 \cdot Y + X \cdot 0 \\ &= 0 + 0 = 0 && \text{(ref. } X \cdot \bar{X} = 0) \end{aligned}$$

2.6.2 De Morgan's Second Theorem

This theorem states that: $(\bar{X} \cdot \bar{Y}) = \bar{X} + \bar{Y}$



Proof: Again to prove this theorem, we will make use of complementarity law i.e.,

$$X + \bar{X} = 1 \text{ and } X \cdot \bar{X} = 0$$

If $\bar{X} \cdot \bar{Y}$'s complement is $\bar{X} + \bar{Y}$ then it must be true that

$$(a) \quad XY + (\bar{X} + \bar{Y}) = 1 \text{ and } (b) \quad XY(\bar{X} + \bar{Y}) = 0$$

To prove the first part

$$\begin{aligned}
 \text{L.H.S.} &= XY + (\bar{X} + \bar{Y}) \\
 &= (\bar{X} + \bar{Y}) + XY \\
 &= (\bar{X} + \bar{Y} + X) \cdot (\bar{X} + \bar{Y} + Y) \\
 &= (X + \bar{X} + \bar{Y}) \cdot (\bar{X} + Y + \bar{Y}) \\
 &= (1 + \bar{Y}) \cdot (\bar{X} + 1) \quad (\text{ref. } X + \bar{X} = 1) \\
 &= 1 \cdot 1 \quad (\text{ref. } 1 + X = 1) \\
 &= 1 = \text{R.H.S.}
 \end{aligned}$$

Now, the second part. i.e., $XY \cdot (\bar{X} + \bar{Y}) = 0$

$$\begin{aligned}
 \text{L.H.S.} &= XY \cdot (\bar{X} + \bar{Y}) \quad (\text{ref. } X(Y+Z)=XY+XZ) \\
 &= XY\bar{X} + XY\bar{Y} \\
 &= X\bar{X}Y + XY\bar{Y} \\
 &= 0 \cdot Y + X \cdot 0 \quad (\text{ref. } X \cdot \bar{X} = 0) \\
 &= 0 + 0 \\
 &= 0 \\
 &= \text{RHS}
 \end{aligned}$$

$$XY \cdot (\bar{X} + \bar{Y}) = 0 \text{ and } XY(\bar{X} + \bar{Y}) = 1$$

Thus, $\overline{XY} = \bar{X} + \bar{Y}$ Hence the theorem.

Although the identities above represent De Morgan's theorem, the transformation is more easily performed by following these steps:

- (i) Complement the entire function
- (ii) Change all the ANDs (.) to ORs (+) and all the ORs (+) to ANDs (.)
- (iii) Complement each of the individual variables.

This process is called De Morganization.

'Break the line, change the sign' to De Morganize a Boolean expression.

a) Solve using De Morgan's Theorem

$$\begin{aligned}
 \overline{\overline{AB} + \overline{A} + AB} &= \overline{\overline{AB}} + \overline{\overline{A}} + \overline{AB} \\
 &= \overline{\overline{AB}} \cdot \overline{\overline{A}} \cdot \overline{AB} \\
 &= ABA(\overline{A} + \overline{B}) \\
 &= ABAA\overline{A} + ABAB\overline{B} \\
 &= AB(A\overline{A} + A\overline{B}) \\
 &= AB(0 + A\overline{B}) \\
 &= AB \cdot 0 + ABA\overline{B} \\
 &= 0 + 0 \\
 &= 0
 \end{aligned}
 \quad (\because \overline{AB} = \overline{A} + \overline{B}; \text{ Demorgan's 2nd theorem}) \\
 \quad (\because (\overline{X+Y} = \overline{X} \cdot \overline{Y}) \text{ Demorgan's law})$$

2.6.3 Applications of De Morgan's Theorem

1. De Morgan's theorem useful in the implementation of the basic gate operations with alternative gates, particularly with NAND and NOR gates which are readily available in IC form.
 2. De Morgan's theorem is used in the simplification of Boolean expressions.
 3. De Morgan's laws commonly apply to text searching using Boolean operators AND, OR and NOT. Consider a set of documents containing the words "cars" or "trucks". De Morgan's laws hold that these two searches will return the same set of documents.
 4. De Morgan's laws are an example of a more general concept of mathematical duality.

2.6.4 Basic Duality of Boolean algebra

We already have talked about duality principle. If you observe all the theorems and rules covered so far, you'll find a basic duality which underlines all Boolean algebra. The postulates and theorems which have been presented can all be divided into pairs.

For example, $X+X.Y = X$

Its dual will be $X.(X+Y) = X$

(Remember, change . to + and vice versa; complement 0 and 1.)

Similarly, $(X+Y)+Z = X+(Y+Z)$ is the dual of $(X.Y).Z=X.(Y.Z)$

and $X+0 = X$ is dual of $X \cdot 1 = X$

In proving the theorems or rules of Boolean algebra, it is then necessary to prove only one theorem, and the dual of the theorem follows necessarily.

In effect, all Boolean algebra is predicated on this two-for-one basis.

Example 1.17: Give the dual of following result in Boolean algebra:

$$XX = 0 \text{ for each } X.$$

Solution: Using duality principle, dual of $X\bar{X}=0$ is $X+\bar{X}=1$ (By changing (.) to (+) and vice versa and by replacing 1's by 0's and vice versa).

Example 1.18: Give the dual of $X+0=X$ for each X .

Solution: Using duality principle, dual of $X+0=X$ is $X \cdot 1=X$

Example 1.19: State the principle of duality in Boolean algebra and give the dual of the Boolean expression: $(X+Y) \cdot (\bar{X}+\bar{Z}) \cdot (Y+Z)$

Solution: Principle of duality states that from every Boolean relation, another Boolean relation can be derived by

- (i) Changing each OR sign (+) to an AND (.) sign
- (ii) Changing each AND (.) sign to an OR (+) sign
- (iii) Replacing each 1 by 0 each 0 by 1

The new derived relation is known as the dual of the original relation.

Dual of $(X+Y) \cdot (\bar{X}+\bar{Z}) \cdot (Y+Z)$ will be

$$(X \cdot Y) + (\bar{X} \cdot \bar{Z}) + (Y \cdot Z) = XY + \bar{X}\bar{Z} + YZ$$

2.7 DERIVATION OF BOOLEAN EXPRESSION

Boolean expressions which consist of a single variable or its complement e.g., X or \bar{Y} or Z are known as literals.

Now before starting derivation of Boolean expression, first we will talk about two very important terms. These are (i) Minterms (ii)Maxterms

2.7.1 Minterms

Minterm is a **product** of all the literals (with or without the bar) within the logic system.

One of the most powerful theorems within Boolean algebra states that any Boolean function can be expressed as the sum of products of all the variables within the system. For example, $X+Y$ can be expressed as the sum of several products, each of the product containing letters X and Y . These products are called Minterms and each product contains all the literals with or without the bar.

Also when values are given for different variables, minterm can easily be formed. E.g., if $X=0$, $Y=1$, $Z=0$ then minterm will be $\bar{X}YZ$ i.e., for variable with a value 0, take its complement and the one with value 1, multiply it as it is. Similarly for $X=1$, $Y=0$, $Z=0$, minterm will be $X\bar{Y}\bar{Z}$.

Steps involved in minterm expansion of expression

1. First convert the given expression in sum of products form.
2. In each term, if any variable is missing (e.g., in the following example Y is missing in first term and X is missing in second term), multiply that term with (missing term+missing term) factor, (e.g., if Y is missing multiply with $Y+\bar{Y}$).
3. Expand the expression.
4. Remove all duplicate terms and we will have minterm form of an expression.

Example 1.20: Convert $X+Y$ to minterms.

Solution: $X+Y=X.1+Y.1$

$$\begin{aligned}
 &=X.(Y+\bar{Y})+Y(X+\bar{X}) && (X+\bar{X}=1 \text{ complementary law}) \\
 &=XY+X\bar{Y}+XY+\bar{X}Y \\
 &=XY+XY+X\bar{Y}+\bar{X}Y \\
 &=XY + X\bar{Y} + \bar{X}Y && (XY + XY = XY \text{ Indempotent law})
 \end{aligned}$$

Note that each term in the above example contains all the letters used: X and Y. The terms XY, X and Y are therefore minterms. This process is called expansion of expression.

Other procedure for expansion could be

1. Write down all the terms
2. Put X's where letters much be inserted to convert the term to a product term.
3. Use all combinations of X's in each term to generate minterms.
4. Drop out duplicate terms.

Example 1.21: Find the minterms for $AB+C$.

Solution: It is a 3 variable expression, so a product term must have all three letters, A, B and C.

1. Write down all terms $AB+C$
 2. Insert X's where letters are missing $ABX+XXC$
 3. Write all the combinations of X's in first term $\bar{A}\bar{B}\bar{C}, ABC$
- Write all the combinations of X's in second term $\bar{A}\bar{B}C, A\bar{B}C, ABC, \bar{A}BC$
4. Add all of them. Therefore, $AB+C = AB\bar{C}+ABC+\bar{A}\bar{B}C+A\bar{B}C+ABC+\bar{A}BC$
 5. Now remove all duplicate terms. $AB\bar{C}+ABC+\bar{A}\bar{B}C+A\bar{B}C+\bar{A}BC$

Now to verify, we will prove vice versa

$$\begin{aligned}
 &AB\bar{C}+ABC+\bar{A}\bar{B}C+ABC = AB + C \\
 LHS &= AB\bar{C}+ABC+\bar{A}\bar{B}C+A\bar{B}C+\bar{A}BC \\
 &= \bar{A}\bar{B}C+\bar{A}BC+A\bar{B}C+AB\bar{C}+ABC \\
 &= \bar{A}C(\bar{B} + B) + A\bar{B}C + AB(\bar{C} + C) \\
 &= \bar{A}C.1 + A\bar{B}C + AB.1
 \end{aligned}$$

$$\begin{aligned}
 &= \bar{A}C \cdot 1 + A\bar{B}C + AB \cdot 1 \\
 &= \bar{A}C + AB + \bar{A}\bar{B}C \\
 &= \bar{A}C + A(B + \bar{B}C) \\
 &= \bar{A}C + A(B + C) && (B + \bar{B}C = B + C \text{ Absorption law}) \\
 &= \bar{A}C + AB + AC \\
 &= \bar{A}C + AC + AB \\
 &= C(\bar{A} + A) + AB \\
 &= C \cdot 1 + AB \\
 &= C + AB \\
 &= AB + C \\
 &= RHS
 \end{aligned}$$

Shorthand minterm Notation

Since all the letters (2 in case of 2 variable expression, 3 in case of 3 variable expressions) must appear in every product, a shorthand notation has been developed that saves actually writing down the letters themselves. To form this notation, following steps are to be followed:

1. First of all, copy original terms.
2. Substitute 0's for barred letters and 1's for non-barred letters
3. Express the decimal equivalent of binary word as a subscript of m.

Example 1.22: To find the minterm designation of $\bar{X}\bar{Y}\bar{Z}$

Solution: 1. Copy original form = $\bar{X}\bar{Y}\bar{Z}$

2. Substitute 1's for non-barred and 0's for barred letters.

Binary equivalent = 100

3. Decimal equivalent of 100 = $1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 4 + 0 + 0 = 4$
4. Express as decimal subscript of

Thus $\bar{X}\bar{Y}\bar{Z} = m_4$

Similarly, minterm designation of $A\bar{B}\bar{C}\bar{D}$ would be

Copy Original Term $A\bar{B}\bar{C}\bar{D}$

Binary equivalent = 1010

Decimal equivalent = $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 8 + 0 + 2 + 0 = 10$

Express as subscript of m = m_{10}

2.7.2 Maxterms

A maxterm is a **sum** of all the literals (with or without the bar) within the logic system.

Trying to be logical about logic, if there is something called minterm, there surely must be one called maxterm and there is.

If the value of a variable is 1, then its complement is added otherwise the variable is added as it is.

Example: If the values of variables are X=0, Y=1 and Z=1 then its Maxterm will be $\bar{X} + \bar{Y} + Z$ (Y and Z are 1's, so their complements are taken; X= 0, so it is taken as it is).

Similarly if the given values are X=1, Y=0, Z =0 and W=1 then its Maxterm is $\bar{X} + Y + Z + \bar{W}$.

Maxterms can also be written as M (Capital M) with a subscript which is decimal equivalent of given input combination e.g., above mentioned Maxterm $\bar{X}+Y+Z+\bar{W}$ whose input combination is 1001 can be written as M_9 as decimal equivalent of 1001 is 9.

2.7.3 Canonical Expression

Canonical expression can be represented in following two forms:

- (i) Sum-of-Products (SOP)
- (ii) Product-of-sums (POS)

Boolean Expression composed entirely either of minterms or maxterms is referred to as **Canonical Expression**.

Sum-of-Products (SOP)

A logical expression is derived from two sets of known values:

- Various possible input values
- The desired output values for each of the input combinations.

Let us consider a specific problem.

A logical network has two inputs X and Y and an output Z. The relationship between inputs and outputs is to be as follows:

- (i) When X=0 and Y=0 then Z=1
- (ii) When X =0 and Y=1 then Z=0
- (iii) When X =1 and Y=0, then Z=1

- (iv) When $X=1$ and $Y=1$, then $Z=1$

We can prepare a truth table from the above relations as follows:

X	Y	Z	Product Terms
0	0	1	$\bar{X}\bar{Y}$
0	1	0	$\bar{X}Y$
1	0	1	$X\bar{Y}$
1	1	1	XY

Table 1.36 truth table for product terms (2-input)

Here, we have added one more column to the table consisting list of product terms or minterms. Adding all the terms for which the output is 1. i.e., $Z=1$ we get following expression:

$$\bar{X}\bar{Y} + \bar{X}Y + XY = Z$$

Now see, it is an expression containing only minterms. This type of expression is called **minterm canonical form of Boolean expression** or **canonical sum-of-products form of expression**.

When a Boolean expression is represented purely as sum of minterms, it said to be in canonical SOP form.

Example 1.23: A Boolean function F defined on three input variables X, Y and Z is 1 if and only if number of 1(one) inputs is odd (e.g., F is 1 if $X=1, Y=0, Z=0$), Draw the truth table for the above function and express it in canonical sum of products from.

Solution: The output is 1, only if one of the inputs is odd. All the possible combinations when one of inputs is odd are

$$X=1, Y=0, Z=0$$

$$X=0, Y=1, Z=0$$

$$X=0, Y=0, Z=1$$

$$X=1, Y=1, Z=1$$

For these combinations output is 1, otherwise output is 0. Preparing the truth table for it we get the following truth table.

X	Y	Z	F	Product Terms/ Minterms
0	0	0	0	\overline{XYZ}
0	0	1	1	\overline{XYZ}
0	1	0	1	\overline{XYZ}
0	1	1	0	\overline{XYZ}
1	0	0	1	\overline{XYZ}
1	0	1	0	\overline{XYZ}
1	1	0	0	\overline{XYZ}
1	1	1	1	\overline{XYZ}

Table 1.37 truth table for product terms (3-input)

Adding all the minterms (product terms) for which output is 1, get

$$\overline{XYZ} + \overline{XYZ} + \overline{XYZ} + \overline{XYZ}$$

This is the desired Canonical SOP from

So, deriving SOP expression from truth table can be summarized as follows:

1. For a given expression, prepare a truth table for all possible combinations of inputs.
2. Add a new column for minterms and list the minterms for all the combinations.
3. Add all the minterms for which there is output as 1. This gives you the desired canonical S-O-P expression.

Another method of deriving canonical SOP expression is algebraic method. This is just the same as above. We will take another example here.

Example 1.24: Convert $\overline{XY} + \overline{XZ}$ into canonical SOP form.

Solution: Rule 1: Simplify the given expression using appropriate theorems/rules.

$$\begin{aligned} \overline{(\overline{XY}) + (\overline{XZ})} &= (X + \overline{Y})(X + Z) \quad \text{using demorgan's law} \\ &= X + \overline{YZ} \quad \text{(Using Distributive law)} \end{aligned}$$

Since it is a 3 variable expression, a product term must have all 3 variables.

Rule 2: Wherever a literal is missing, multiply that term with

$$\text{missing variable} + \overline{\text{missing variable}}$$

$$X + \overline{YZ} = X(Y + \overline{Y})(Z + \overline{Z}) + (X + \overline{X})\overline{YZ}$$

(Y, Z are missing in first term, x is missing in second term)

$$\begin{aligned}
 &= (XY + X\bar{Y})(Z + \bar{Z}) + X\bar{Y}Z + \bar{X}\bar{Y}Z \\
 &= Z(XY + X\bar{Y}) + \bar{Z}(XY + X\bar{Y}) + X\bar{Y}Z + \bar{X}\bar{Y}Z \\
 &= XYZ + X\bar{Y}Z + XY\bar{Z} + X\bar{Y}\bar{Z} + X\bar{Y}Z + \bar{X}\bar{Y}Z
 \end{aligned}$$

Rule 3: By removing the duplicate terms, we get $XYZ + X\bar{Y}Z + XY\bar{Z} + X\bar{Y}\bar{Z} + \bar{X}\bar{Y}Z$. This is the desired Canonical SOP from.

Above Canonical SOP expression can also be represented by following shorthand notation. Here F is a variable function and m is a notation for minterm. This specifies that output F is sum of 1st, 4th, 5th, 6th and 7th minterms.

$$\text{i.e., } F = m_1 + m_4 + m_5 + m_6 + m_7 \text{ or } F = \Sigma(1, 4, 5, 6, 7)$$

Converting Shorthand notation to minterms

We already have learnt how to represent minterm into shorthand notation. Now we will learn how to convert vice versa.

Rule1: Find binary equivalent of decimal subscript e.g., for m_6 subscript is 6, binary equivalent of 6 is 110.

Rule2: For every 1's write the variable as it is and for 0's write variable's complemented form i.e., for 110 it is $XY\bar{Z}$. $XY\bar{Z}$ is the required minterm for m_6 .

Example 1.25: Convert the following three input function F denoted by the expression into its canonical SOP form.

Solution: If three inputs are X, Y and Z then

$$\begin{aligned}
 F &= m_0 + m_1 + m_2 + m_5 \\
 m_0 &= 000 \Rightarrow \bar{X}\bar{Y}\bar{Z} \\
 m_1 &= 001 \Rightarrow \bar{X}\bar{Y}Z \\
 m_2 &= 010 \Rightarrow \bar{X}YZ \\
 m_5 &= 101 \Rightarrow X\bar{Y}Z
 \end{aligned}$$

Canonical SOP form of the expression is

$$\bar{X}\bar{Y}\bar{Z} + \bar{X}\bar{Y}Z + \bar{X}YZ + X\bar{Y}Z$$

Product-of-sum form (POS)

When a Boolean expression is represented purely as product of Maxterms, it is said to be in canonical Product-of-Sum form.

This form of expression is also referred to as Maxterm canonical form of Boolean expression.

Just as any Boolean expression can be transformed into a sum of minterms, it can also be represented as a product of Maxterms.

(a) Truth table method

The truth Table method for arriving at the desired expression is as follows:

1. Prepare a table of inputs and outputs
2. Add one additional column of sum terms. For each row of the table, a sum term is formed by adding all the variables in complemented or uncomplemented form. i.e., if input value for a given variable is 1, variable is complemented and if 0, not complemented.

Example: If $X=0, Y=1, Z=1$ then Sum term will be $X + \bar{Y} + \bar{Z}$

Now the desired expression is product of the sums from the rows in which the output is 0.

Example 1.26: Express in the product of sums from the Boolean function $F(X, Y, Z)$ and the truth table for which is given below:

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Solution: Add a new column containing Maxterms. Now the table is as follow:

X	Y	Z	F	Maxterms
0	0	0	1	$X + Y + Z$
0	0	1	0	$X + Y + \bar{Z}$
0	1	0	1	$X + \bar{Y} + Z$
0	1	1	0	$X + \bar{Y} + \bar{Z}$
1	0	0	1	$\bar{X} + Y + Z$
1	0	1	0	$\bar{X} + Y + \bar{Z}$
1	1	0	1	$\bar{X} + \bar{Y} + Z$
1	1	1	1	$\bar{X} + \bar{Y} + \bar{Z}$

Now by multiplying maxterms for the output 0's, we get the desired product of sums expression which is $(X + Y + \bar{Z})(X + \bar{Y} + \bar{Z})(\bar{X} + Y + \bar{Z})$

(b) Algebraic Method

We will explain this method with the help of an example.

Example 1.27 Express $\bar{X}Y + Y(\bar{Z}(\bar{Z} + Y))$ into canonical product-of-sums form.

Solution: Rule 1: Simplify the given expression using appropriate theorems/rules:

$$\begin{aligned}\bar{X}Y + Y(\bar{Z}(\bar{Z} + Y)) &= \bar{X}Y + Y(\bar{Z}\bar{Z} + Y\bar{Z}) && \{ X(Y+Z) = XY+XZ \} \\ &= \bar{X}Y + Y(\bar{Z} + Y\bar{Z}) && (Z \cdot Z = Z \text{ as } X \cdot X = X) \\ &= \bar{X}Y + Y\bar{Z}(1 + Y) \\ &= \bar{X}Y + Y\bar{Z} \cdot 1 && \{ 1 + Y = 1 \} \\ &= \bar{X}Y + Y\bar{Z}\end{aligned}$$

Rule 2: To convert into product of sums form, apply the Boolean algebra rule which states that $X + YZ = (X + Y)(X + Z)$

$$\begin{aligned}\bar{X}Y + Y\bar{Z} &= (\bar{X}Y + Y)(\bar{X}Y + \bar{Z}) && (X + Y = Y + X) \\ &= (Y + \bar{X}Y)(\bar{Z} + \bar{X}Y) \\ &= (Y + \bar{X})(Y + Y)(\bar{Z} + \bar{X})(\bar{Z} + Y) \\ &= (\bar{X} + Y)Y(\bar{X} + \bar{Z})(Y + \bar{Z}) && (X + Y = Y)\end{aligned}$$

Now, this is in product of sums form but not in canonical product of sums form (In Canonical expression all the sum terms are Maxterms.)

Rule 3: After converting into product of sum terms, in a sum term for a missing variable add (Missing variable . missing variable.) e.g., if variable Y is missing add $Y\bar{Y}$.

$$(\bar{X} + Y)(Y)(\bar{X} + \bar{Z})(Y + \bar{Z}) = (\bar{X} + Y + Z\bar{Z})(X\bar{X} + Y + Z\bar{Z})(\bar{X} + Y\bar{Y} + \bar{Z})(X\bar{X} + Y + \bar{Z})$$

Rule 4: Keep on simplifying the expression (using the rule, $X+YZ=(X+Y)(X+Z)$) until you get product of sum terms which are maxterms.

$$(\bar{X} + Y + Z\bar{Z}) = (\bar{X} + Y + Z)(\bar{X} + Y + \bar{Z}) = M_4 \cdot M_5$$

$$\begin{aligned}(X\bar{X} + Y + Z\bar{Z}) &= (X\bar{X} + Y + Z)(X\bar{X} + Y + \bar{Z}) \\ &= (X + Y + Z)(\bar{X} + Y + Z)(X + Y + \bar{Z})(\bar{X} + Y + \bar{Z}) = M_0 \cdot M_4 \cdot M_1 \cdot M_5\end{aligned}$$

$$(\bar{X} + Y\bar{Y} + \bar{Z}) = (\bar{X} + Y + \bar{Z})(\bar{X} + \bar{Y} + \bar{Z}) = M_5 \cdot M_7$$

$$(X\bar{X} + Y + \bar{Z}) = (X + Y + \bar{Z})(\bar{X} + Y + \bar{Z}) = M_1 \cdot M_5$$

$$(\bar{X} + Y)(Y)(\bar{X} + Z)(Y + \bar{Z}) = (\bar{X} + Y + Z)(\bar{X} + Y + \bar{Z})(X + Y + Z)(\bar{X} + Y + \bar{Z})(X + Y + \bar{Z})(\bar{X} + Y + \bar{Z})$$

Short hand = $M_4, M_5, M_0, M_4, M_1, M_5, M_5, M_7, M_1, M_5 = M(0, 4, 5, 7)$

Rule 5: Removing all the duplicate terms, we get

$$(\bar{X} + Y + Z) (\bar{X} + Y + \bar{Z}) (X + Y + Z) (X + Y + \bar{Z}) (\bar{X} + \bar{Y} + \bar{Z})$$

This is the desired canonical product of sums form of expression.

Shorthand maxterm notation

Shorthand notation of the above given canonical product of sums expression is

$$F = \prod (0, 1, 4, 5, 7) \text{ or } F = \prod M(0, 1, 4, 5, 7)$$

This specifies that output F is product 0th, 1st, 4th and 7th Maxterms

$$\text{i.e., } F = M_0 \cdot M_1 \cdot M_5 \cdot M_7$$

Here, M_0 means Maxterm for Binary equivalent of 0 i.e., 000 ie., X=0, Y=0, Z=0

And, Maxterm will be $(X+Y+Z)$ (Complemented variable is 1 and uncomplemented variable is 0)

Similarly, M_1 Means 0 0 1 $X+Y+\bar{Z}$

$$\text{AS } F = M_0 \cdot M_1 \cdot M_4 \cdot M_5 \cdot M_7$$

$$\text{and } M_0 = 000 \quad X + Y + Z$$

$$M_1 = 001 \quad X + Y + \bar{Z}$$

$$M_4 = 100 \quad \bar{X} + Y + Z$$

$$M_5 = 101 \quad \bar{X} + Y + \bar{Z}$$

$$M_7 = 111 \quad \bar{X} + \bar{Y} + \bar{Z}$$

$$F = (X + Y + Z) (X + Y + \bar{Z}) (\bar{X} + Y + Z) (\bar{X} + Y + \bar{Z}) (\bar{X} + \bar{Y} + \bar{Z})$$

Example 1.28: Convert the following function into canonical product of sums form: $F(X, Y, Z) = \prod M(0, 2, 4, 5)$

Note: To convert an expression from shorthand SOP form to shorthand POS form, just create truth table from given expression. From the created truth table, derive other form of expression. For example, from truth table, you can convert an expression $F(X, Y, Z) = \sum(0, 1, 3, 5)$ to $\prod M(2, 4, 6, 7)$

Solution: $F \prod(X, Y, Z) = \prod M(0, 2, 4, 5) = M_0 \cdot M_2 \cdot M_4 \cdot M_5$

$$M_0 = 000 \quad X + \underline{Y} + Z$$

$$M_2 = 010 \quad \underline{X} + \bar{Y} + Z$$

$$M_4 = 100 \quad \bar{X} + \underline{Y} + \underline{Z}$$

$$M_5 = 101 \quad \bar{X} + Y + \bar{Z}$$

$$\Rightarrow F = (X + Y + Z) (X + \bar{Y} + Z) (\bar{X} + \underline{Y} + \underline{Z}) (\bar{X} + Y + \bar{Z})$$

Sum term V/s Maxterm and product term V/s minterm

Sum term means sum of the variables. It does not necessarily mean that all the variables must be included whereas Maxterm means a sum-term having the entire variables.

For Example, for 3 Variables $F(X, Y, Z)$ functions $X + Y$, $X + Z$, $\bar{Y} + Z$ etc. are sum terms whereas $X + Y + Z$, $\bar{X} + Y + \bar{Z}$, $X + \bar{Y} + Z$ etc. are Maxterms.

Similarly, product term means product of the variables, not necessarily all the variables, whereas minterm means product of all the variables.

For A 3 variable (a, b, c) function $\bar{a}bc$, $a\bar{b}c$, $ab\bar{c}$ etc. are minterms whereas ab , bc , bc , ac etc. are product terms only.

Same is the difference between Canonical SOP or POS expression. A Canonical SOP or POS expression must have all the Minterms or Maxterms respectively, whereas a simple SOP or POS expression can just have product terms or sum terms respectively.

2.7.4 Minimization of Boolean expression

After obtaining an SOP or POS expression, the next thing to do is to simplify the Boolean expression, because Boolean operations are practically implemented in the form of gates. A minimized Boolean expression means less number of gates which means simplified circuitry. This section deals with two methods simplification of Boolean expression.

Algebraic Method

This method makes use of Boolean postulates, rules and theorems to simplify the expressions.

Example 1.29 simplify $A\bar{B}C\bar{D} + A\bar{B}CD + ABC\bar{D} + ABCD$

Solution: $A\bar{B}C\bar{D} + A\bar{B}CD + ABC\bar{D} + ABCD$

$$\begin{aligned} & A\bar{B}C(\bar{D} + D) + ABC(\bar{D} + D) = A\bar{B}C \cdot 1 + ABC \cdot 1 \quad (\bar{D} + D = 1) \\ & = AC(\bar{B} + B) = AC \end{aligned}$$

Example 1.30: Reduce the expression $\bar{X}\bar{Y} + \bar{X} + XY$

Solution: $\bar{X}\bar{Y} + \bar{X} + XY$

$$\begin{aligned} & = (\bar{X} + \bar{Y}) + \bar{X} + XY \quad (\text{using Demorgan's 2nd theorem i.e., }) \\ & = \bar{X} + \bar{X} + XY + \bar{Y} \\ & = \bar{X} + XY + \bar{Y} \quad \{\bar{X} + \bar{X} = \bar{X} \text{ as } X + X = X\} \\ & = (\bar{X} + X)(\bar{X} + Y) + \bar{Y} \quad (\text{Putting } X + \bar{X} = 1) \\ & = \bar{X} + Y + \bar{Y} \quad \{Y + \bar{Y} = 1\} \\ & = \bar{X} + 1 \quad (\text{Putting } Y + \bar{Y} = 1) \\ & = 1 \quad \{\text{putting } \bar{X} + 1 = 1 \text{ as } 1 + X = 1\} \end{aligned}$$

Example 1.31: Minimize $AB + \overline{AC} + A\overline{B}C (AB + C)$

$$\begin{aligned}
 \textbf{Solution} \quad AB + \overline{AC} + A\overline{B}C (AB + C) &= AB + \overline{AC} + A\overline{B}CAB + A\overline{B}CC \\
 &= AB + \overline{AC} + AAB\overline{B}C + A\overline{B}CC \\
 &= AB + \overline{AC} + A\overline{B}C && \{BB = 0 \text{ and } CC = C\} \\
 &= AB + \overline{A} + \overline{C} + A\overline{B}C && \{\overline{AC} = \overline{A} + \overline{C}\} \\
 &= \overline{A} + AB + \overline{C} + A\overline{B}C && \{\text{rearranging the terms}\} \\
 &= \overline{A} + B + \overline{C} + A\overline{B}C && \{\overline{A} + AB = A + B \text{ because } X + \overline{XY} = X + Y\} \\
 &= \overline{A} + \overline{C} + B + AC\overline{B} && (B + \overline{B}AC = B + AC \text{ because } X + \overline{XY} = X + Y) \\
 &= \overline{A} + \overline{C} + B + AC && (\overline{C} + CA = \overline{C} + A) \\
 &= \overline{A} + B + \overline{C} + AC \\
 &= \overline{A} + B + \overline{C} + A \\
 &= A + \overline{A} + B + \overline{C} \\
 &= 1 + B + \overline{C} && \{A + \overline{A} = 1\} \\
 &= 1 && (1 + X = 1)
 \end{aligned}$$

Example 1.32: Reduce $\overline{X}\overline{Y}\overline{Z} + \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + X\overline{Y}Z$

$$\begin{aligned}
 \textbf{Solution.} \quad \overline{X}\overline{Y}\overline{Z} + \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + X\overline{Y}Z &= \overline{X}(\overline{Y}\overline{Z} + Y\overline{Z}) + X(\overline{Y}\overline{Z} + Y\overline{Z}) \\
 &= (X + \overline{X})(\overline{Y}\overline{Z} + Y\overline{Z}) \\
 &= \overline{Z}(\overline{Y} + Y) \\
 &= \overline{Z}
 \end{aligned}$$

2.8 Simplification using Karnaugh Maps

Truth tables provide a nice, natural way to list all values of a function. There are several other ways to represent function values. One of the way is Karnaugh Map (in short K-Map) named after its originator Maurice Karnaugh. These maps are sometimes also called Veitch diagrams.

Karnaugh Map or K-Map is a graphical display of the fundamental product in a truth table.

Karnaugh map is nothing but a rectangle made up of certain number of squares, each square representing a Maxterm or Minterm.

2.8.1 Sum of products Reduction using Karnaugh Map

In S-O-P reduction each square of K-Map represents a minterm of the given function. Thus, for a function of n variables, there would be a map of 2^n squares, each representing a minterm (refer to Fig. 1.7). Given a K-map, for SOP reduction the map is filled in by placing in squares whose minterms lead to a 1 output.

Following are 2,3,4 variable K-maps for SOP reduction. (see fig. 1.7)

Note in every square a number is written. These subscripted numbers denote that this square corresponds to that number's minterm. For example, in 3 variable map X Y Z box has been given number 2 which means this square corresponds to M_2 . Similarly, box number 7 means it corresponds to m_7 and so on.

Please notice the numbering scheme here, it is 0, 1, 3, 2 then 4, 5, 7, 6 and so on, always squares are marked using this scheme while making a K-map.

	$(0)\bar{Y}$	$(1)Y$
$(0)\bar{X}$	$\bar{X}\bar{Y}$	$\bar{X}Y$
0		1
$(1)X$	$X\bar{Y}$	XY
2		3

(a)

	$(0)\bar{Y}$	$(1)Y$
$(0)\bar{X}$		
0		1
$(1)X$		
2		3

(a)

	$(00)\bar{Y}\bar{Z}$	$(01)\bar{Y}Z$	$(11)YZ$	$(10)Y\bar{Z}$
$(0)\bar{X}$	$\bar{X}\bar{Y}\bar{Z}$ 0	$\bar{X}\bar{Y}Z$ 1	$\bar{X}YZ$ 3	$\bar{X}Y\bar{Z}$ 2
$(1)X$	$X\bar{Y}\bar{Z}$ 4	$X\bar{Y}Z$ 5	XYZ 7	$XY\bar{Z}$ 6

(c)

	$(00)\bar{Y}\bar{Z}$	$(01)\bar{Y}Z$	$(11)YZ$	$(10)Y\bar{Z}$
$(0)\bar{X}$	0	1	3	2
$(1)X$	4	5	7	6

(d)

\bar{X}	$(00)\bar{Y}\bar{Z}$	$(01)\bar{Y}Z$	$(11)YZ$	$(10)Y\bar{Z}$
$(00)W\bar{X}$	$\bar{W}\bar{X}\bar{Y}\bar{Z}$ 0	$\bar{W}\bar{X}\bar{Y}Z$ 1	$\bar{W}\bar{X}YZ$ 3	$\bar{W}\bar{X}Y\bar{Z}$ 2
$(01)\bar{W}X$	$\bar{W}X\bar{Y}\bar{Z}$ 4	$\bar{W}X\bar{Y}Z$ 5	$\bar{W}XY\bar{Z}$ 7	$\bar{W}XY\bar{Z}$ 6
$(11)WX$	$W\bar{X}\bar{Y}\bar{Z}$ 12	$W\bar{X}\bar{Y}Z$ 13	$WXY\bar{Z}$ 15	$WXY\bar{Z}$ 14
$(10)W\bar{X}$	$W\bar{X}\bar{Y}\bar{Z}$ 8	$W\bar{X}\bar{Y}Z$ 9	$W\bar{X}YZ$ 11	$W\bar{X}Y\bar{Z}$ 10

\bar{X}	$(00)\bar{Y}\bar{Z}$	$(01)\bar{Y}Z$	$(11)YZ$	$(10)Y\bar{Z}$
$(00)W\bar{X}$	0	1	3	2
$(01)\bar{W}X$	4	5	7	6
$(11)WX$	12	13	15	14
$(10)W\bar{X}$	8	9	11	10

4-variable K Map representing minterms.

Observe carefully above given K-map. See the binary numbers at the top of K-map. These do not follow binary progression, instead they differ by only one place when moving from left to right : 00, 01, 11, 10. It is done so that only one variable changes from complemented to un complemented form or vice versa. The terms are $\bar{A}\bar{B}$, $\bar{A}B$, AB , $A\bar{B}$

This binary code 00, 01, 11, 10 is called Gray code. Gray Code is the binary code in which each successive number differs only in one place. That is why box numbering scheme follows above order only.

How to Map in K-Map?

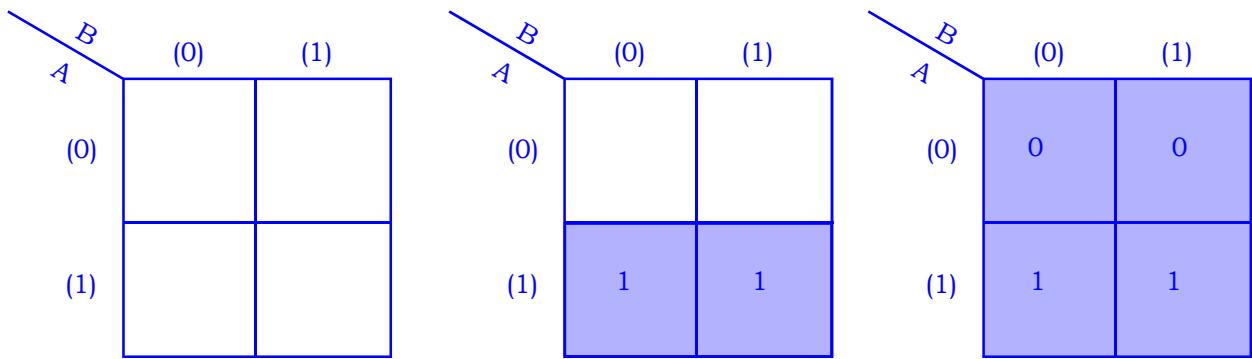
We'll take an example of 2 variable map to be illustrated, with the following truth table for mapping (Table 1.38)

Table 1.38

A	B	F
0	0	0
0	1	0
1	0	1
1	1	1

Canonical S-O-P expression for this table is $F=AB + AB$ or $F = \Sigma(2,3)$.

To map this function first we'll draw an empty 2-variable K-map as shown in Fig. 1.8(a)



Now look for output 1 in the given truth table (1.38) for a given truth table.,.

For minterms M_2 and M_3 the output is 1. Thus mark 1 in the squares for m_2 and m_3 i.e., square numbered as 2 and the one numbered as 3. Now our K-map will look like fig 1.8 (b)

After entering 1's for all 1 outputs, enter 0's in all blank squares. K-map will now look like Fig 1.8 same is the method for mapping 3-variable and 4-variable maps i.e., enter 1's for all 1 outputs in the corresponding squares and then enter 0's in the rest of the squares.

How to reduce Boolean expression in S-O-P form using K-map?

For reducing the expression, first we have to mark pairs, quads and octets.

To reduce an expression, adjacent 1's are encircled. If 2 adjacent 1's are encircled, it makes a pair; if 4 adjacent 1's are encircled, it makes a quad; and if 8 adjacent 1's are encircled, it makes an octet.

While encircling groups of 1's, firstly search for octets and mark them, then for quads and lastly go for pairs. This is because a bigger group removes more variables thereby making the resultant expression simpler.

Reduction of a pair : In the K-map in fig. 1.9, after mapping a given function $F(W, X, Y, Z)$ two pairs have been marked. Pair-1 is $m_0 + m_4$ (group of 0th minterm and 4th minterm as these numbers tell us minterm's subscript). Pair-2 is $m_{14} + m_{15}$.

Observe that Pair-1 is a vertical pair. Moving vertically in pair-1, see one variable X is changing its state from \bar{X} to X as m_0 is $\bar{W} \bar{X} \bar{Y} \bar{Z}$ and m_4 is $\bar{W} X \bar{Y} \bar{Z}$. Compare the two and we see $\bar{W} \bar{X} \bar{Y} \bar{Z}$ changes to $\bar{W} X \bar{Y} \bar{Z}$. So, the variable X can be removed.

	$(00)\bar{Y}\bar{Z}$	$(01)\bar{Y}Z$	$(11)YZ$	$(10)Y\bar{Z}$
$(00)W\bar{X}$	1 0	0 1	0 3	0 2
$(01)\bar{W}X$	1 4	0 5	0 7	0 6
$(11)WX$	0 12	0 13	1 15	1 14
$(10)W\bar{X}$	0 8	0 9	0 11	0 10

	$(00)\bar{Y}\bar{Z}$	$(01)\bar{Y}Z$	$(11)YZ$	$(10)Y\bar{Z}$
$(00)W\bar{X}$	1 0	0 1	0 3	0 2
$(01)\bar{W}X$	1 4	0 5	1 7	1 6
$(11)WX$	1 12	0 13	0 15	1 14
$(10)W\bar{X}$	1 8	0 9	0 11	0 10

Pair Reduction Rule

Remove the variable which changes its state from complemented to uncomplemented or vice versa. Pair removes one variable only.

Thus reduced expression for Pair-1 is $\bar{W}\bar{Y}\bar{Z}$ as $\bar{W}\bar{X}\bar{Y}\bar{Z}$ (m_0) changes to $\bar{W}X\bar{Y}\bar{Z}$ (m_4)

We can prove the same algebraically also as follows :

$$\begin{aligned}
 \text{Pair-1} = m_0 + m_4 &= \bar{W}\bar{X}\bar{Y}\bar{Z} + \bar{W}X\bar{Y}\bar{Z} \\
 &= \bar{W}\bar{Y}\bar{Z}(\bar{X} + X) \\
 &= \bar{W}\bar{Y}\bar{Z} \cdot 1 \quad (\bar{X} + X = 1) \\
 &= \bar{W}\bar{Y}\bar{Z}
 \end{aligned}$$

Similarly, reduced expression for Pair-2 ($m_{14} + m_{15}$) will be WXY as $WXYZ$ (m_{14}) changes to $WXY\bar{Z}$ (m_{15}). Z will be removed as it is changing its state from \bar{Z} to Z .

Reduction of a quad

If we are given with the K-map shown in fig. 1.10 in which two quads have been marked.

Quad-1 is $m_0 + m_4 + m_{12} + m_8$ and Quad-2 is $m_7 + m_6 + m_{15} + m_{14}$. When we move across quad-1, two variables change their states i.e., W and X are changing their states, so these two variables will be removed.

Quad, Reduction Rule

Remove the two variables which change their states. A Quad removes two variables. Thus reduced expression for quad-1 is $\bar{Y}\bar{Z}$ as W and X (both) are removed.

Similarly, in Quad -2 ($m_7 + m_6 + m_{15} + m_{14}$), horizontally moving, variable Z is removed as $\bar{W} X Y Z$ (m_7) changes to $\bar{W} X Y \bar{Z}$ (m_6) and vertically moving, variable W is removed as (m_7) changes to $WXYZ$. Thus reduced expression for quad-2 is (by removing W and Z) $XY + \bar{Y}\bar{Z}$

Reduction of an octet

Suppose, we have K-map with an octet marked as shown in Fig. 1.11.

$W\bar{X}$	(00) $\bar{Y}\bar{Z}$	(01) $\bar{Y}\bar{Z}$	(11)YZ	(10)Y \bar{Z}
(00)WX	0 0	0 1	0 3	0 2
(01)W \bar{X}	0 4	0 5	0 7	0 6
(11)WX	1 12	1 13	1 15	1 14
(10)W \bar{X}	1 8	1 9	1 11	1 10

While moving horizontally in the octet two variables Y and Z are removed and moving vertically one variable x is removed. Thus eliminating X, Y and Z, the reduced expression for the octet is W only.

Octet Reduction Rule

Remove the three variables which change their states. An octet removes 3-variables. But after marking pairs, quads and octets, there are certain other things to be taken care of before arriving at the final expression. These are map rolling, overlapping groups and redundant groups.

Map Rolling

Map Rolling means roll the map i.e., consider the map as if its left edges are touching the right edges and top edges are touching the bottom edges. This is a special property of Karnaugh maps that its opposite edges squares and corner squares are considered contiguous (Just as the world map is treated contiguous at its opposite ends). As in opposite edges squares and in corner squares only one variable changes its state from complemented to uncomplemented state or vice versa. Therefore, while making the pairs, quads and octets, map must be rolled. Following pairs, quads and octets are marking after rolling the map.

	CD	$(00)\bar{C}\bar{D}$	$(01)\bar{C}D$	$(11)CD$	$(10)C\bar{D}$
AB	$\bar{A}\bar{B}$	0	1	3	2
$(00)\bar{A}\bar{B}$	1	4	5	7	6
$(11)AB$	12	13	15	14	
$(10)A\bar{B}$	8	9	11		10

$$\bar{A}\bar{B}\bar{D} + \bar{B}\bar{C}D$$

	CD	$(00)\bar{C}\bar{D}$	$(01)\bar{C}D$	$(11)CD$	$(10)C\bar{D}$
AB	$\bar{A}\bar{B}$	0	1	1	2
$(01)\bar{A}\bar{B}$	1	4	5	7	6
$(11)AB$	1	12	13	15	14
$(10)A\bar{B}$	8	1	9	1	10

$$\bar{B}\bar{D} + \bar{B}D$$

Overlapping Groups

Overlapping means same 1 can be encircled more than once. For example, if the following K-map is given:

	CD	$(00)\bar{C}\bar{D}$	$(01)\bar{C}D$	$(11)CD$	$(10)C\bar{D}$
AB	$\bar{A}\bar{B}$	0	1	3	2
$(01)\bar{A}\bar{B}$		1	1	1	1
$(11)AB$		4	5	7	6
$(10)A\bar{B}$			1	1	1
		12	13	15	14
		8	9	11	10

Observe that 1 for m_7 has been encircled twice. Once for Pair-1($m_5 + m_7$) and again for Quad ($m_7 + m_6 + m_{15} + m_{14}$). Also 1 for m_{14} has been encircled twice. For the Quad and for Pair-2 ($m_{14} + m_{10}$).

Here, reduced expression for Pair-1 is ABD

Reduced expression for Quad is BC

Reduced expression for Pair-2 is ACD

Thus final reduced expression for this map is $ABD + BC + ACD$

Thus reduced expression for entire K-map is sum of all reduced expressions in the very K-map.

But before writing the final expression we must take care of redundant Groups.

Redundant Groups

Redundant group is a group whose all 1's are overlapped by other groups (i.e., pairs, quads, octets). Here is an example, given below.

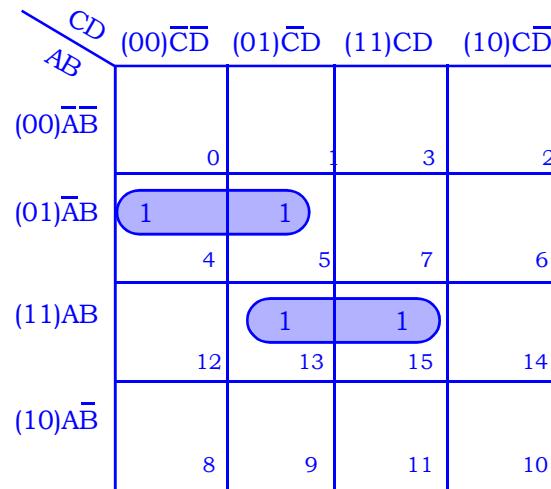
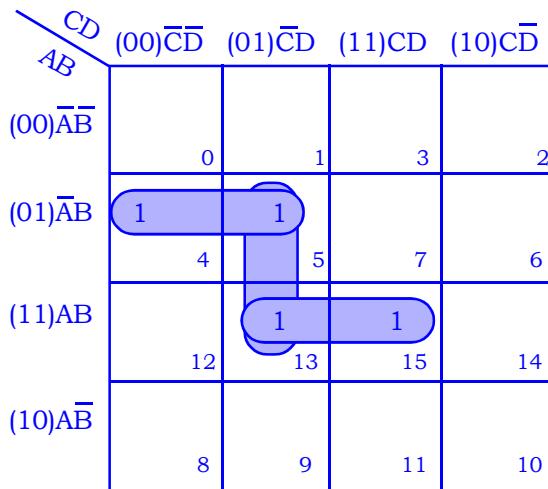


Fig. 1.14(a) has a redundant group. There are three pairs : Pair-1 ($m_4 + m_5$), Pair-2 ($m_5 + m_{13}$), Pair-3 ($m_{13} + m_{15}$). But Pair-2 is a redundant group as its all 1's are marked by other groups.

With this redundant group, the reduced expression will be $\bar{A}\bar{B}\bar{C} + BD + ABD$. For a simpler expression, Redundant Groups must be removed. After removing the redundant group, we get the K-map shown in fig. 1.14 (b).

The reduced expression, for K-map in fig. 1.14 (b), will be

$$\bar{A}\bar{B}\bar{C} + ABD$$

Which is much simpler expression Σ .

Thus removal of redundant group leads to much simpler expression.

Summary of all the rules for S-O-P reduction using K-map

1. Prepare the truth table for given function.
2. Draw an empty K-map for the given function (i.e., 2 variable K-map for 2 variable function; 3 variable K-map for 3 variable function, and so on).

3. Map the given function by entering 1's for the outputs as 1 in the corresponding squares.
4. Enter 0's in all left out empty squares.
5. Encircle adjacent 1's in form of octets, quads and pairs. Do not forget to roll the map and overlap.
6. Remove redundant groups, if any.
7. Write the reduced expressions for all the groups and OR (+) them.

Example 1.33 Reduce $F(a, b, c, d) = \Sigma m(0, 2, 7, 8, 10, 15)$ using Karnaugh map.

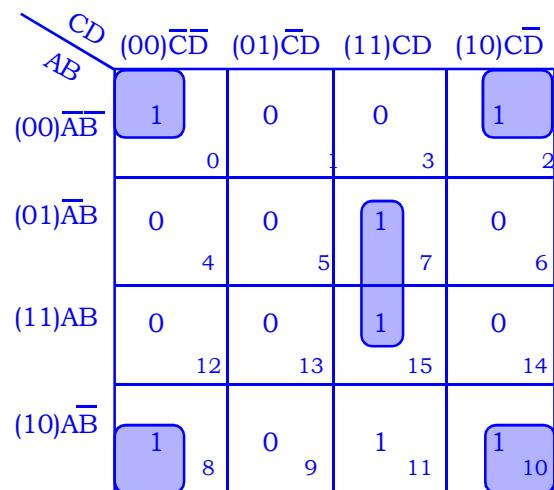
Solution: Given $F(a, b, c, d) = \Sigma m(0, 2, 7, 8, 10, 15)$

$$\begin{aligned}
 &= m_0 + m_2 + m_7 + m_8 + m_{10} + m_{15} \\
 m_0 &= 0000 = \bar{A} \bar{B} \bar{C} \bar{D} & m_2 &= 0010 = \bar{A} \bar{B} C \bar{D} \\
 m_7 &= 0111 = \bar{A} B C D & m_8 &= 1000 = A \bar{B} \bar{C} \bar{D} \\
 m_{10} &= 1010 = A \bar{B} C \bar{D} & m_{15} &= 1111 = A B C D
 \end{aligned}$$

Truth table for the given function is as follows :

A	B	C	D	F
0	0	0	0	1
0	0	0	1	
0	0	1	0	1
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	1
1	0	0	0	1
1	0	0	1	
1	0	1	0	1
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	1
1	1	1	1	

Mapping the given function in a K-map we get



In the above K-map two groups have been marked, one Pair and One Quad.

Pair is $m_7 + m_{15}$

And

$$\text{Quad is } m_0 + m_2 + m_8 + m_{10}$$

Reduced expression for pair $(m_7 + m_{15})$ is BCD as A is removed. Reduced expression for quad $(m_0 + m_2 + m_8 + m_{10})$ is \overline{BD} as for horizontal corners C is removed and for vertical corners A is removed.

Thus final reduced expression is BCD + \overline{BD}

Example 1.34: What is the simplified Boolean equation for the function?

$$F(A,B,C,D) = \Sigma(7,9,10,11,12,13,14,15)$$

Solution: Completing the given Karnaugh map by entering 0's in the empty squares, by numbering the squares with their minterm's subscripts and then by encircling all possible groups, we get the following K-map.

There is one pair, three quads

$$\text{Pair-1} = m_7 + m_{15}$$

$$\text{Quad-1} = m_{12} + m_{13} + m_{14} + m_{15}$$

$$\text{Quad-2} = m_{13} + m_{15} + m_9 + m_{11}$$

$$\text{Quad-3} = m_{15} + m_{11} + m_{14} + m_{10}$$

		CD	(00) \overline{CD}	(01) \overline{CD}	(11)CD	(10)CD
		AB	0	0	0	0
(00)	\overline{AB}	0	0	1	3	2
	(01)	4	5	1	7	6
(11)	AB	1	1	1	1	1
	(10)	12	13	15	14	10
(10)	\overline{AB}	0	1	1	1	1
	(01)	8	9	11	10	

Reduced expression for pair-1 $(m_7 + m_{15})$ is BCD, as \overline{ABCD} (m_7) changes to $ABCD$ (m_{15}) eliminating A.

Reduced expression for Quad-1 $(m_{12} + m_{13} + m_{14})$ is AB, as while moving across the Quad, C and D both are removed because both are changing their states from complemented to uncomplemented or vice-versa.

Reduced expression for Quad 2 $(m_{13} + m_{15} + m_9 + m_{11})$ is AD, as moving horizontally, C is removed and moving vertically, B is removed.

Reduced expression of Quad-3 $(m_{15} + m_{11} + m_{14} + m_{10})$ is AC as horizontal movement removes D and vertical movement removes B.

Thus, Pair-1 = BCD, Quad-1 = AB, Quad-2 = AD, Quad-3 = AC

Hence final reduced expression will be BCD+AB+AD+AC.

Example 1.35: Obtain a simplified expression for a Boolean function F (X, Y, A) the Karnaugh map for which is given below:

$\bar{Y}\bar{Z}$	[00]	[01]	[11]	[10]
[0]	0	1	3	2
[1]	4	5	7	6

Solution: Completing the given K-map.

We have 1 group which is a Quad i.e.,

$$m_1 + m_3 + m_5 + m_7$$

Reduced expression for this Quad is Z , as moving horizontally from $\bar{X} \bar{Y} Z (m1)$ to $\bar{X} \bar{Y} Z (m3)$, Y is removed (Y changing from \bar{Y} to Y) and moving vertically from $m1$ to $m5$ or $m3$ to $m7$, X changes to X, thus \bar{X} is removed.

$\bar{Y}\bar{Z}$	(00) $\bar{Y}\bar{Z}$	(01) $\bar{Y}Z$	(11) YZ	(10) $Y\bar{Z}$
(0) \bar{X}	0 0	1 1	1 1	0 2
(1)X	0 4	1 5	1 7	0 6

(c)

Example 1.36: Minimize the following function using a Karnaugh map:

$$F(W, X, Y, Z) = \Sigma (0, 4, 8, 12)$$

Solution: Given function $F(W, X, Y, Z) = \Sigma (0, 4, 8, 12)$

$$\begin{aligned} F &= m_0 + m_4 + m_8 + m_{12} \\ m_0 &= 0000 = \bar{W} \bar{X} \bar{Y} \bar{Z} \\ m_4 &= 0100 = \bar{W} X \bar{Y} \bar{Z} \\ m_8 &= 1000 = W \bar{X} Y \bar{Z} \\ m_{12} &= 1100 = W X \bar{Y} Z \end{aligned}$$

$\bar{Y}\bar{Z}$	(00) $\bar{Y}\bar{Z}$	(01) $\bar{Y}Z$	(11) YZ	(10) $Y\bar{Z}$
(00) $\bar{W}\bar{X}$	1 0	0 1	0 3	0 2
(01) $\bar{W}X$	1 4	0 5	0 7	0
(11) WX	1 6	0	0	0
(10) $W\bar{X}$	1 12	14 0	15 0	0

Maping the given function on a K-Map, we get $(m_0 + m_4 + m_8)$

Reduced expression for this quad is $\bar{Y}\bar{Z}$ as while moving across the Quad W and X are removed. Because these are changing their states from complemented to uncomplemented or vice versa.

Thus, final reduced expression is $\bar{Y}\bar{Z}$.

Example 1.37: Using the Karnaugh technique obtain the simplified expression as sum products for the following map:

		(00)	(01)	(11)	(10)
		(0)		1	1
		(1)		1	1
				(d)	

Solution: Completing the given K-map, we get one group which is a Quad has been marked.

Quad reduces two variables. Moving horizontally, Z is removed as it changes from Z to \bar{Z} and moving vertically, X is removed as it changes from X to \bar{X} . Thus only one variable Y is left. Hence Reduced S-O-P expression is Y. Thus F=Y assuming F is the given function.

		(00) $\bar{Y}\bar{Z}$	(01) $\bar{Y}Z$	(11) YZ	(10) $Y\bar{Z}$
		(0) \bar{X}	0 0	0 1	1 3 1 2
		(1) X	0 4	0 5	1 7 1 6

2.8.2 Product-of-Sum Reduction using Karnaugh Map

In POS reduction each square of K-map represents a Maxterm. Karnaugh map is just the same as that of the used in S-O-P reduction. For a function of n variables, map would represent 2^n squares, each representing a maxterm.

For POS reduction map is filled by placing 0's in squares whose Maxterms lead to output 0. Following are 2, 3 & 4 variable K-Maps for POS reduction.

	(0) Y	(1) \bar{Y}
(0) X	0	1
(1) \bar{X}	2	3

(a)

	(0) Y	(1) \bar{Y}
(0) X	$(X + Y)$	$(X + \bar{Y})$
(1) \bar{X}	$(\bar{X} + Y)$	$(\bar{X} + \bar{Y})$

(b)

2-variable K-Map representing Maxterms.

	(00) $Y+Z$	(01) $Y+\bar{Z}$	(11) $\bar{Y}+Z$	(10) $Y+\bar{Z}$
(0) X	0	1	3	2
(1) \bar{X}	4	5	7	6

(c)

	(00) $Y+Z$	(01) $Y+\bar{Z}$	(11) $\bar{Y}+Z$	(10) $\bar{Y}+Z$
(0) X	$X+Y+Z$	$X+Y+\bar{Z}$	$X+\bar{Y}+Z$	$X+\bar{Y}+Z$
(1) \bar{X}	$\bar{X}+Y+Z$	$\bar{X}+Y+\bar{Z}$	$\bar{X}+\bar{Y}+Z$	$\bar{X}+\bar{Y}+Z$

(d)

3-variable K-Map representing Maxterms

	(00)	(01)	(11)	(10)
[00]	0	1	3	2
[00]	4	5	7	6
[00]	12	13	15	14
[00]	8	9	11	10

(e)

	(00) $Y+Z$	(01) $Y+\bar{Z}$	(11) $\bar{Y}+\bar{Z}$	(10) $\bar{Y}+Z$
[00] $W+X$	$W + X + Y + Z$	$W + X + Y + \bar{Z}$	$W + X + \bar{Y} + \bar{Z}$	$W + X + \bar{Y} + Z$
[01] $W+\bar{X}$	$W + \bar{X} + Y + Z$	$W + \bar{X} + Y + \bar{Z}$	$W + \bar{X} + \bar{Y} + \bar{Z}$	$W + \bar{X} + \bar{Y} + Z$
[11] $\bar{W}+\bar{X}$	$\bar{W} + \bar{X} + Y + Z$	$\bar{W} + X + Y + \bar{Z}$	$\bar{W} + \bar{X} + \bar{Y} + \bar{Z}$	$\bar{W} + \bar{X} + \bar{Y} + Z$
[10] $\bar{W}+X$	$\bar{W} + X + Y + Z$	$\bar{W} + X + Y + \bar{Z}$	$\bar{W} + X + \bar{Y} + \bar{Z}$	$\bar{W} + X + \bar{Y} + Z$

(f)

4 variable K-Map representing Minterms

Figure 1.115: 2,3,4 variable K-Maps of POS expression.

Again the numbers in the squares represent Maxterm subscripts. Box with number 1 represent M₁, Number 6 represent, M₆, and so on. Also notice box numbering scheme is the same i.e., 0, 1, 3, 2 ; 4, 5, 7, 6 ; 12, 13, 15, 14 ; 8, 9, 11, 10.

One more similarity in SOP K-map and POS K-map is that they are binary progression in gray code only. So, here also some Gray Code appears at the top.

But one major difference is that in POS K-Map, complemented letters represent 1's uncomplemented letters represent 0's, whereas it is just the opposite in SOP K-Map. Thus in the fig 1.15 (b), (d), (f) for 0's uncomplemented letters appear and for 1's complemented letters appear.

How to derive POS Boolean expression using K-Map?

Rules for deriving expression are the same except for the thing i.e., POS expression adjacent 0's are encircled in the form of pairs, quads and octets. Therefore, rules for deriving POS Boolean expression can be summarized as follows:

1. Prepare the truth table for a given function.
2. Draw an empty K-map for given function (i.e., 2-variable K-map for 2 variable function; 3 variable K-map for 3 variable function and so on).
3. Map the given function by entering 0's then squares numbered 5 and 13 will be having 0's)
4. Enter 1's in all left out empty squares.
5. Encircle adjacent 0's in the form of octets, quads, and pair. Do not forget to role the map and overlap.
6. Remove redundant groups, if any.
7. Write the reduced expressions for all the groups and AND (.) them.

Example 1.38: Reduce the following Karnaugh map in Product of sums form:

		(00)	(01)	(11)	(10)
		0	0	0	1
		(0)	0	1	1
A	$B\bar{C}$	0	1	1	1
	$B{C}$	1	0	0	0

Solution: To reach at POS expression, we'll have to encircle all possible groups of adjacent 0's encircling we get the following K-map.

There are 3 pairs which are;

		$[00]B+C$	$[01]B+\bar{C}$	$[11]\bar{B}+\bar{C}$	$\bar{B}+C$
		0	0	0	
		0	1	3	2
$[0]A$	0	0			
	1		0	0	7
$[1]\bar{A}$	0	4	5		6
	1				

Pair1: $M_0 \cdot M_1$;

Pair 2: $M_0 \cdot M_4$;

Pair 3: $M_1 \cdot M_3$;

But there is one redundant group also i.e., Pair-1 (it's all 0's are encircled by other groups). Thus removing this redundant pair-1, we have only two groups now.

Reduced POS expression for Pair-2 is $(B+C)$, as while moving across pair-2, A changes its state from A to A, thus A is removed.

Reduced POS expression for Pair 3 is $(A+\bar{C})$, as while moving across Pair 3 B changes to \bar{B} , hence eliminated.

Final POS expression will be $(B+C).(A+\bar{C})$

Example 1.39: Find the minimum POS expression of

$$Y(A, B, C, D) = \Pi(0, 1, 3, 5, 6, 7, 10, 14, 15).$$

Solution: As the given function is 4 variable function, we'll draw 4 variable K-Map and then put 0's for the given Maxterms. i.e., in the squares whose numbers are 0, 1, 3, 5, 6, 7, 10, 14, 15 as each square number represents its Maxterm. So, K-map will be

		$(00)C+D$	$(01)C+\bar{D}$	$(11)\bar{C}+\bar{D}$	$(10)\bar{C}+D$
		0	0	0	1
		0	1	3	2
$(00)A+B$	0				
	1				
$(01)A+\bar{B}$	0				
	1				
$(11)\bar{A}+\bar{B}$	0				
	1				
$(10)\bar{A}+B$	0				
	1				
		8	9	11	10

Encircling adjacent 0's we have following groups:

$$\text{Pair-1} = M_0, M_1;$$

$$\text{Pair -2}= M_{14}, M_{10};$$

$$\text{Quad} = M_1, M_3, M_5, M_7;$$

$$\text{Quad-2} = M_7, M_6, M_{15}, M_{14};$$

Reduced expressions are the following:

$$\text{For Pair -1, } (A+B+C)$$

(as D is eliminated: D changes to \bar{D})

$$\text{For pair-2, } (\bar{A}+\bar{C}+D)$$

(\bar{B} Changes to B; hence eliminated)

$$\text{For Quad-1, } (A+\bar{D})$$

(Horizontally C and vertically B is eliminated

as C, B are changing their states)

$$\text{For Quad-2, } (\bar{B}+\bar{C})$$

(horizontally D and vertically A is

eliminated)

Hence final POS expression will be

$$Y(A, B, C, D) = (A+B+C) (\bar{A}+\bar{C}+D+) (A+\bar{D}) (\bar{B}+\bar{C})$$

Review questions:

One mark questions:

1. What is another name of Boolean algebra?
2. What do you understand by logic function?
3. Give examples for logic function.
4. What is meant by tautology and fallacy?
5. Prove the $1+Y$ is a tautology and $0.Y$ is a fallacy.
6. State indempotence law.
7. Prove indempotence law using truth table.
8. Draw logic diagram to represent indempotence law.
9. State Involution law.
10. Prove Involution law using truth table.
11. Draw logic diagram to represent Involution law.
12. State Complementarity law.
13. Prove Complementarity law using truth table.
14. Draw logic diagram to represent Complementarity law.
15. State Commutative law.
16. Prove Commutative law using truth table.
17. Draw logic diagram to represent Commutative law.
18. State Associative law.
19. Prove Associative law using truth table.
20. Draw logic diagram to represent Associative law.
21. State Distributive law.
22. Prove Distributive law using truth table.
23. Draw logic diagram to represent Distributive law.
24. Prove that $X+XY = X$ (Absorption law)
25. Prove that $X(X+Y) = X$ (Absorption law)
26. Draw logic diagram to represent Absorption law.
27. Prove that $XY + X\bar{Y} = X$
28. Prove that $(X+Y)(X+\bar{Y}) = X$
29. Prove that $X + \bar{X} Y = X + Y$

-
30. What is a minterm?
 31. Find the minterm for $XY + Z$.
 32. What is a maxterm?
 33. Find the maxterm for $X + \bar{Y} + Z$.
 34. What is the canonical form of Boolean expression?

Two marks questions:

1. Prove algebraically that $(X + Y)(X + Z) = X + YZ$
2. Prove algebraically that $X + \bar{X}Y = X + Y$
3. Use duality theorem to derive another Boolean relation from : $A + \bar{A}B = A + B$
4. What would be complement of the following :
 - (a) $\bar{A}(\bar{B}\bar{C} + \bar{B}\bar{C})$
 - (b) $AB + \bar{C}D$
 - (c) $XY + \bar{Y}Z + ZZ$
 - (d) $X + \bar{X}Y + \bar{X}Z$
5. What are the fundamental products for each of the input words; $ABCD = 0010$, $ABCD = 110$, $ABCD = 1110$. Write SOP expression.
6. A truth table has output 1 for each of these inputs. $ABCD = 0011$, $ABCD = 0101$, $ABCD = 1000$, what are the fundamental products and write minterm expression.
7. Construct a Boolean function of three variables X, Y and Z that has an output 1 when exactly two of X, Y and Z are having values 0, and an output 0 in all other cases.
8. Construct a truth table for three variables A, B and C that will have an output 1 when $XYZ = 100$, $XYZ = 101$, $XYZ = 110$ and $XYZ = 111$. Write the Boolean expression for logic network in SOP form.
9. Convert the following expressions to canonical Product-of-Sum form:
 - (a) $(A+C)(C+D)$
 - (b) $A(B+C)(\bar{C} + \bar{D})$
 - (c) $(X+Y)(Y+Z)(X+Z)$
10. Convert the following expressions to canonical Sum-of-Product form:
 - (a) $(X+\bar{X}Y+\bar{X}Z)$
 - (b) $YZ + \bar{X}Y$
 - (c) $AB (\bar{B} + \bar{C})$
11. Draw Karnaugh maps for the following expressions:
 - (a) $XY + \bar{X}Y$
 - (b) $XY\bar{Z} + \bar{X}YZ$
 - (c) $\bar{X}\bar{Y}\bar{Z} + X\bar{Y}\bar{Z} + \bar{X}Y\bar{Z}$
12. Draw a general K-map for four variables A, B, C and D.
13. Given the expression in four variables , draw the K – map for the function:
 - (a) $m_2 + m_3 + m_5 + m_7 + m_9 + m_{11} + m_{13}$
 - (b) $m_0 + m_2 + m_4 + m_8 + m_9 + m_{10} + m_{11} + m_{12} + m_{13}$

14. Draw the K – map for the function in three variables given below.
- (a) $m_0 + m_2 + m_4 + m_6 + m_7$
 (b) $m_1 + m_2 + m_3 + m_5 + m_7$
15. Write S-O-P expression corresponding to the function F in the following truth table and draw the logic gate diagram (use OR and AND gates)

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Three marks questions:

- State and prove any three theorems of Boolean algebra.
- State and prove associative law of addition and multiplication.
- State and prove De Morgan's theorems by the method of perfect induction.
- Obtain the minterm expression for the Boolean function $F = A+BC$.
- Explain with an example how to express a Boolean function in its sum-of-products form.
- Explain with an example how to express a Boolean function in its product-of-sum form.
- Construct a truth table for minterms and maxterms for three variables and designate the terms.
- Using basic gates, construct a logic circuit for the Boolean expression $(X+Y).(X+Z).(Y+Z)$
- Simplify the following Boolean expressions and draw logic circuit diagrams of the simplified expressions using only NAND gates.
 - $\overline{ABC} + \overline{ABC} + \overline{ABC} + ABC$
 - $AC + AB + \overline{ABC} + BC$
 - $(ABC).(ABC) + ABC + \overline{ABC}$
 - $\overline{ABC} + \overline{ABC} + \overline{ABC} + ABC$
 - $(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})$
- For a four variable map in w,x,y and z draw the subcubes for
 - WXY
 - WX
 - XYZ
 - Y
- Convert the following product-of-sums form into its corresponding sum-of-products form using write the truth table.
 $F(x,y,z) = \Pi(2,4,6,7)$
- (a) Reduce the following Boolean expression to the simplest form:
 $A.[B+C.(AB + AC)]$
- (b) Given : $F(x,y,z) = \Sigma(1,3,7)$ then prove that $F'(x,y,z) = \Pi(0,2,4,5,6)$

Five marks questions:

1. Using maps, simplify the following expressions in four variables W, X, Y and Z.
 - (a) $m_1 + m_3 + m_5 + m_6 + m_7 + m_9 + m_{11} + m_{13}$
 - (b) $m_0 + m_2 + m_4 + m_8 + m_9 + m_{10} + m_{11} + m_{12} + m_{13}$
2. For the Boolean function F and F' in the truth table , find the following:
 - (a) List the minterms of the functions F and F'
 - (b) Express F and F' in sum of minterms in algebraic form.
 - (c) Simplify the functions to an expression with a minimum number of literals.

A	B	C	F	F'
0	0	0	0	1
0	0	1	0	1
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0

3. State and prove De Morgan's theorems algebraically.
4. Find the complement of $F = X + YZ$, then show that $F \cdot F' = 0$ and $F + F' = 1$.
5. (a) State the two Absorption laws of Boolean algebra. Verify using truth table.
 (b) Simplify using laws of Boolean algebra. At each step state clearly the law used for simplification. $F = x.y + x.z + x.y.z$
6. Given the Boolean function $F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$. Reduce it using Karnuagh map method.
7. (a) State the two complement properties of Boolean algebra. Verify using the truth tables.(b) $x(\bar{y}\bar{z} + yz)$
8. Given the Boolean function $F(A,B,C,D) = \Sigma(5,6,7,8,9,10,14)$. Use Karnaugh's map to reduce the function F using SOP form. Write a logic gate diagram for the reduced SOP expression.
9. Given ; $F(A,B,C,D) = (0,2,4,6,8,10,14)$. Use Karnaugh map to reduce the function F using POS form. Write a logic gate diagram for the reduced POS expression.
10. Use Karnaugh map to reduce the given functions using SOP form. Draw the logic gate diagrams for the reduced SOP expression. You may use gates with more than two inputs. Assume that the variables and their complements are available as inputs.

11. Given the Boolean function $F(A,B,C,D)=\Sigma(0,4,8,9,10,11,12,13,15)$.

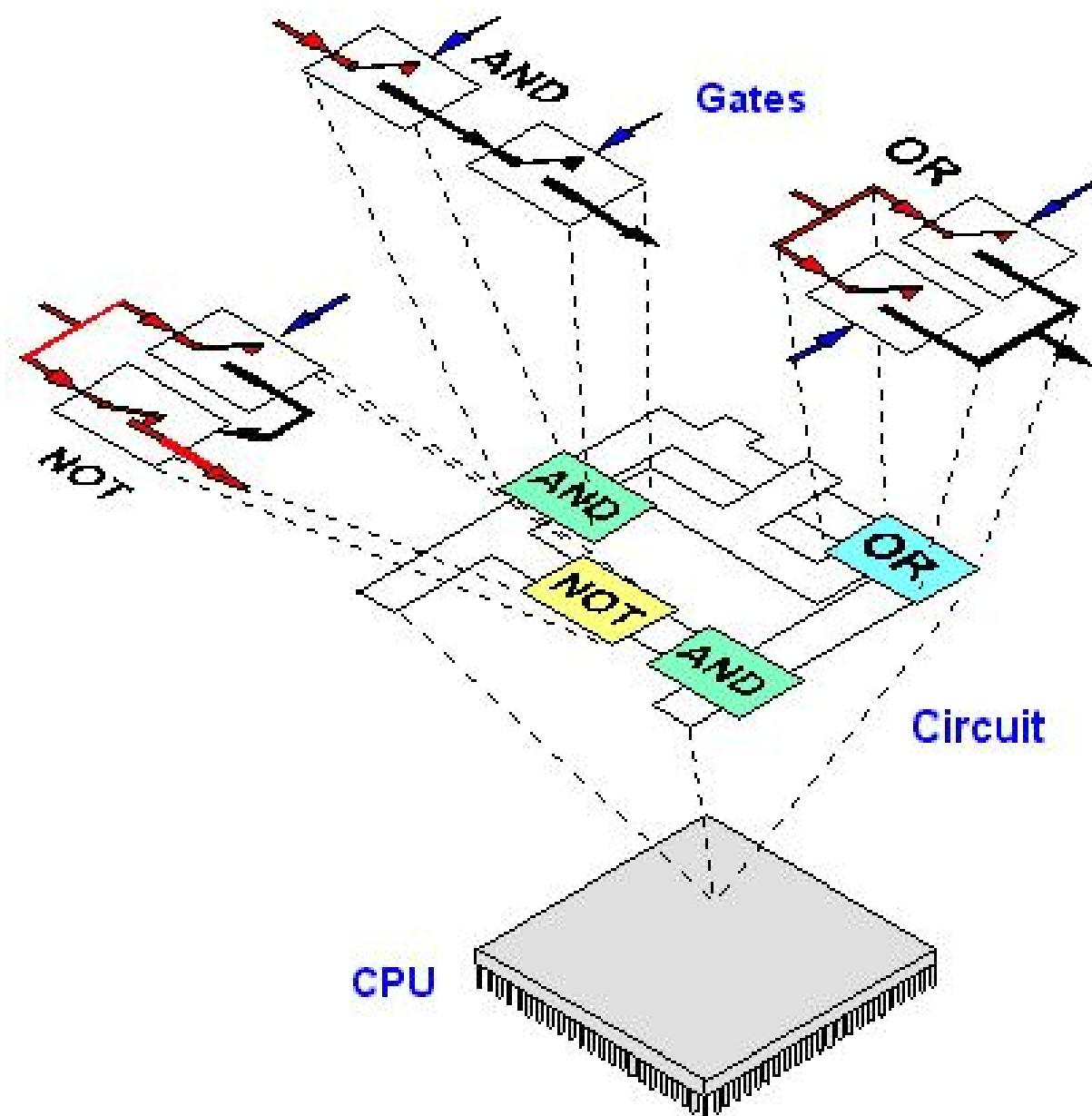
Reduce it by using Karnaugh map.

Working Sheet:

Chapter 3 Logic Gates

Objectives:

- Learning different types of gates
- Designing the logical circuits



LOGIC GATES

3.1 Introduction

After Shannon applied Boolean algebra in telephone switching circuits, engineers realized that Boolean algebra could be applied to computer electronics as well.

In the computers, these Boolean operations are performed by logic gates.

Elementary logic gates

Gates are digital (two-state) circuits because the input and output signals are either low voltage (denotes 0) or high voltage (denotes 1). Gates are often called logic circuits, because they can be analyzed with Boolean algebra.

A gate is simply an electronic circuit which operates on one or more signals and always produces an output signal.

Gates are classified into two types.

- (a) Basic gates
- (b) Derived gates

(a) Basic gates

There are three basic logic gates:

1. NOT gate (inverter)
2. OR gate
3. AND gate

(b) Derived gates

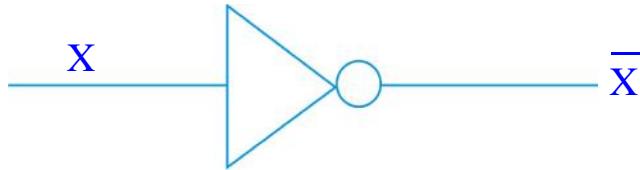
There are four derived gates:

1. NOR gate
2. NAND gate
3. XOR gate (Exclusive OR gate)
4. XNOR gate (Exclusive NOR gate)

3.1.1 Inverter (NOT gate)

An inverter is also called a NOT gate, because the output is not the same as the input. The output is sometimes called the complement (opposite) of the input.

An Inverter is a gate with only one input signal and one output signal; the output state is always the opposite of the input state.



X	X̄
Low	High
High	Low

X	X̄
0	1
1	0

A low input i.e., 0 produces high output i.e., 1 and vice versa. NOT operation is symbolized as $\overline{\quad}$ or \neg i.e., NOT X is written as X^1 or X .

3.1.2 OR Gate

The OR gate has two or more input signals but only one output signal. If one or more input signals are 1 (high), the output signal is 1 (high).

An OR gate can have as many inputs as desired. No matter how many inputs are there, the action of OR gate is the same.

The OR gate has two or more input signals but only one output signal. If any of the input signals is 1 (high), the output signal is 1 (high).

Following tables show OR action:

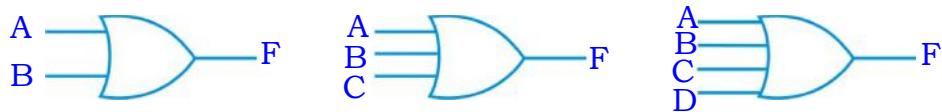
X	Y	F=X+Y
0	0	0
0	1	1
1	0	1
1	1	1

Table 3.3 Truth table for
2-input OR gatea

X	Y	Z	F=X+Y+Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Table 3.4 Truth Table for three input OR gate

The symbol for OR gate is given below:



OR operation is symbolized as + i.e., X or Y is written as $X+Y$.

3.1.3 AND gate

This gate also can have two or more inputs and always gives single output. If any input is 0, the output is 0. To obtain output as 1, all inputs must be 1. Thus, the AND represents the logical multiplication.

The AND Gate can have two or more input signals and produce one output signal. When all the inputs are high then the output is high. Otherwise, the output is low.

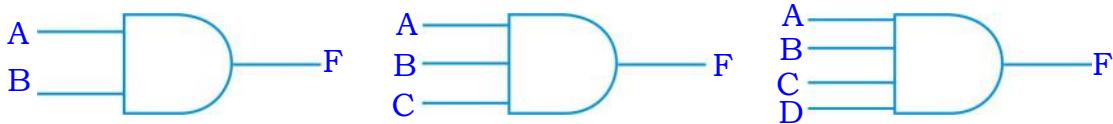
X	Y	F=X.Y
0	0	0
0	1	0
1	0	0
1	1	1

Table 3.5: 2- input AND gate

X	Y	Z	F=X.Y.Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Table 3.6: 3-input AND gate

The symbol for AND is



AND operation is symbolized as X AND Y is written as X.Y

3.2 Derived gates

3.2.1 NOR Gate

NOR gate is nothing but NOT OR gate or inverted OR gate. This means, an OR gate is always followed by a NOT gate to give NOR gate.

This gate also accepts two or more than two inputs and always produces single output. If either of the two inputs is 1 (high), the output will be 0 (low). Also, if all the inputs are low, then the output is high.

The NOR gate has two or more input signals but only one output signal. If all the inputs are 0 (low), then the output signal is 1 (high).

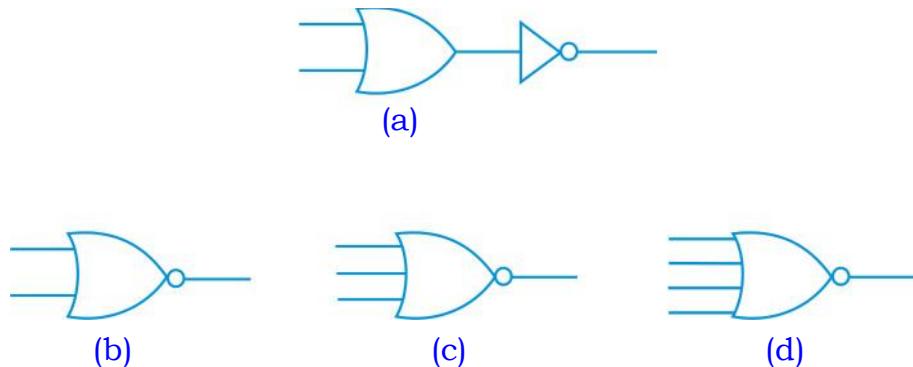
X	Y	F = $\overline{X+Y}$
0	0	1
0	1	0
1	0	0
1	1	0

Table 3.7: 2-input NOR gate

X	Y	Z	F = $\overline{X+Y+Z}$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Table 3.8: 3-input NOR gate

NOR operation is symbolized as i.e., X NOR Y is written as $\overline{X + Y}$.



3.2.2 NAND Gate

NAND gate is NOT AND gate or inverted AND gate. This means, an AND gate is always followed by a NOT gate to give NAND gate.

NAND gate can also have two or more inputs. This gate produces 0 (low) for all 1 (high) inputs and produces 1 (high) for other input combinations.

The NAND Gate has two or more input signals but only one output signal. If all of the inputs are 1 (high), then the output produced is 0 (low).

NAND action is illustrated in following Truth Tables (2.9 and 2.10)

X	Y	F = XY
0	0	1
0	1	1
1	0	1
1	1	0

Table 3.9: Truth table of 2-input NAND gate

X	Y	Z	F = XYZ
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Table 3.10 Truth table of 3-input NAND gate

The logical meaning of NAND gate can be shown as follows:

NAND operation is symbolized as i.e., X NAND Y is written as $\overline{X \cdot Y}$.

3.2.3 XOR Gate or Exclusive-OR Gate

The XOR Gate can also have two or more inputs, but produces one output signal. Exclusive-OR gate is different from OR gate. OR gate produces output 1 for any input combination having one or more 1's, but XOR gate produces output 1 for only those input combinations that have odd number of 1's.

- Accepts two or more inputs and produces single output.
- The output is 0 if there are even number of 1's in the inputs.
- The output is 1 if there are odd number of 1's in the inputs.

In Boolean algebra, \oplus sign stands for XOR operation. Thus, A XOR B can be written as $A \oplus B$.

Following Truth Tables (2.11 and 2.12) illustrates XOR operation.

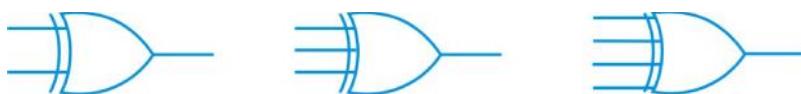
No. of 1's Even/odd	X	Y	$Z = A \oplus B$
Even	0	0	0
Odd	0	1	1
Odd	1	0	1
Even	1	1	0

Table 3.11 Truth table of 2-input XOR gate

No. of 1's	X	Y	Z	F
Even	0	0	0	0
Odd	0	0	1	1
Odd	0	1	0	1
Even	0	1	1	0
Odd	1	0	0	1
Even	1	0	1	0
Even	1	1	0	0
Odd	1	1	1	1

Table 3.12 3 input XOR gate

The symbols of XOR gates are given below:



XOR addition can be summarized as following:

$$0 \oplus 0 = 0; \quad 0 \oplus 1 = 1; \quad 1 \oplus 0 = 1; \quad 1 \oplus 1 = 0$$

The operation representing XOR may be written as $F = x \oplus y = \bar{x}y + x\bar{y}$

3.2.4 XNOR Gate or Exclusive NOR Gate

An XOR gate is followed by a NOT gate (inventor) becomes XNOR gate. Thus, The XNOR Gate is logically equivalent to an inverted XOR gate. Thus XNOR produces 1 (high) as output when the input combination has even number of 1's or when all the inputs are 0's.

Following truth tables 2.13 and 2.14 illustrate XNOR action.

No. of 1's Even/odd	X	Y	F
Even	0	0	1
Odd	0	1	0
Odd	1	0	0
Even	1	1	1

Table 3.13 2 input XNOR gate

No. of 1's	X	Y	Z	F
Even	0	0	0	1
Odd	0	0	1	0
Odd	0	1	0	0
Even	0	1	1	1
Odd	1	0	0	0
Even	1	0	1	1
Even	1	1	0	1
Odd	1	1	1	0

Table 3.14 3 input XNOR gate

Following are the XNOR gate symbols:

The bubble (small circle), on the output of NAND, NOR, XNOR gates represents complementation.

The expression representing XNOR may be written as

$$F = (\bar{X}+Y) \cdot (X+\bar{Y})$$

The XNOR operation is symbolized as \odot i.e. X NOR Y is written as $X\odot Y$.

Now that you are familiar with logic gates, you can use them in designing logic circuits.

3.2.5 Circuit diagrams

Boolean algebra is useless unless it can be translated into hardware, in the form of gates. This translation of Boolean algebra in the gates' form is known as logic circuits. A logic circuit can be represented diagrammatically using the traditional symbols of gates. Let us see how this is done, in the following examples.

Example 2.1: Design a circuit to realize the following:

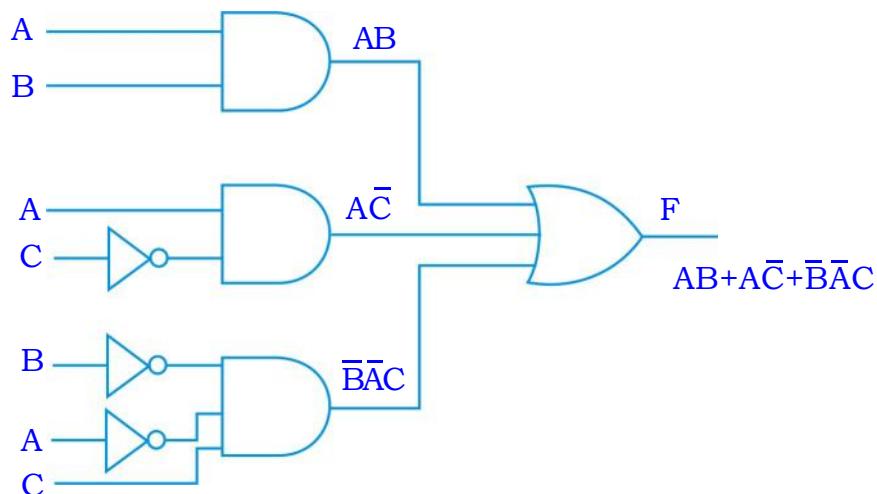
$$F(A, B, C) = AB + A\bar{C} + \bar{B}\bar{A}C$$

Solution: The given Boolean expression can also be written as follows

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

or $F(A, B, C) = (A \text{ AND } B) \text{ OR } (A \text{ AND } (\text{NOT } C)) \text{ OR } ((\text{NOT } B) \text{ AND } (\text{NOT } A) \text{ AND } C)$

Now these logical operators can easily be implemented in form of logic gates. Thus circuit diagram for above expression will be as follows:



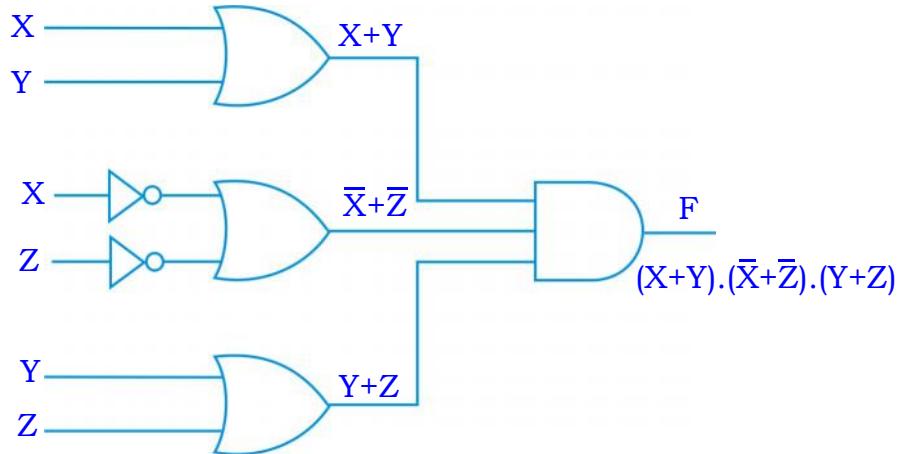
Example 2.2 Draw the diagram of digital circuit for the function:

$$F(X, Y, Z) = (X + Y) \cdot (\bar{X} + \bar{Z}) \cdot (Y + Z)$$

Solution: Above expression can also be written as

$$F(X, Y, Z) = (X \text{ OR } Y) \text{ AND } ((\text{NOT } X) \text{ OR } (\text{NOT } Z)) \text{ AND } (Y \text{ OR } Z)$$

Thus circuit diagram will be



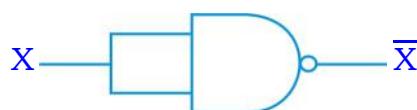
3.2.6 NAND, NOR as Universal Gates

We can design circuits using AND, OR and NOT gates as we have done so far. But NAND and NOR gates are more popular as these are less expensive and easier to design. Also, The basic gates AND, OR and NOT can easily be implemented using NAND/NOR gates. Thus NAND, NOR gates are also referred to as Universal Gates.

Universal gate is a gate using which all the basic gates can be designed. NAND and NOR gates are called as the universal gates.

NAND-to-NOT logic

Not Operation



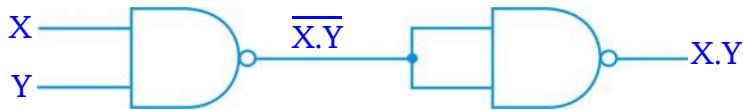
$$\text{NOT } X = X \text{ NAND } X$$

$$\begin{aligned}
 &= \overline{X \cdot X} && \{\text{De Morgan's Second Theorem}\} \\
 &= \overline{X} + \overline{X} && \{\text{De Morgan's Second Theorem}\} \\
 &= \overline{X}
 \end{aligned}$$

NAND-to-AND logic

AND and OR operations from NAND gates are shown below:

AND operation



AND operation using NAND is

$$X \cdot Y = (X \text{ NAND } Y) \text{ NAND } (X \text{ NAND } Y)$$

$$\text{Proof: } X \text{ NAND } Y = \overline{X \cdot Y}$$

$$= \overline{X} + \overline{Y}$$

(De Morgan's Second Theorem)

$$(X \text{ NAND } Y) \text{ NAND } (X \text{ NAND } Y)$$

$$= (\overline{X} + \overline{Y}) \text{ NAND } (\overline{X} + \overline{Y})$$

$$= \overline{\overline{X} + \overline{Y}} \cdot \overline{\overline{X} + \overline{Y}}$$

(De Morgan's Second Theorem)

$$= \overline{\overline{X} \cdot \overline{Y}} + \overline{\overline{X} \cdot \overline{Y}}$$

(De Morgan's First Theorem)

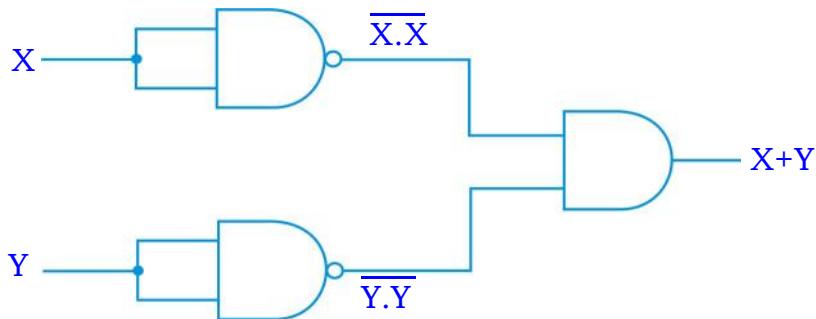
$$= X \cdot Y + X \cdot Y$$

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

$$= XY$$

$$(X + X = X)$$

OR Operation



$$X + Y = (X \text{ NAND } X) \text{ NAND } (Y \text{ NAND } Y)$$

$$\text{Proof: } X \text{ NAND } X = \overline{X \cdot X}$$

{De Morgan's Second Theorem}

$$= \overline{X} + \overline{X}$$

{De Morgan's Second Theorem}

$$= \overline{X}$$

$$\{ X + X = X \}$$

Similarly, $Y \text{ NAND } Y = \overline{Y}$

$$\text{Therefore, } (X \text{ NAND } X) \text{ NAND } (Y \text{ NAND } Y) = \overline{\overline{X} \text{ NAND } \overline{Y}}$$

$$= \overline{(X \cdot Y)}$$

$$= \overline{\overline{X} + \overline{Y}} \quad \{ \overline{\overline{X}} = X, \overline{\overline{Y}} = Y \}$$

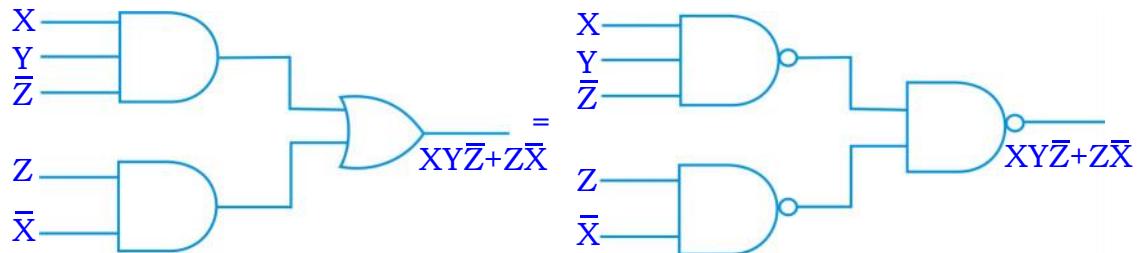
$$= X + Y$$

NAND-to-NAND logic is best suited for Boolean expression in Sum-of-Products form.

Design rule for NAND-TO-NAND logic Network (only for two-level-circuits)

- 1 Derive simplified sum-of products expression.
- 2 Draw a circuit diagram using AND, OR and NOT gates.
- 3 Just replace AND, OR and NOT gates with NAND gates

For example, $XY\bar{Z} + \bar{X}Z$ can be drawn as follows, assuming that inputs and their complements are available:



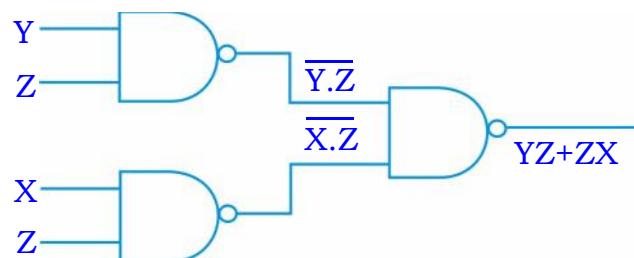
Example 2.3: Draw the diagram of a digital circuit for the function

$$F(X, Y, Z) = YZ + XZ \text{ using NAND gates only.}$$

Solution : $F(X, Y, Z) = YZ + XZ$ can be written as

$$= (Y \text{ NAND } Z) \text{ NAND } (X \text{ NAND } Z)$$

Thus logic circuit diagram is



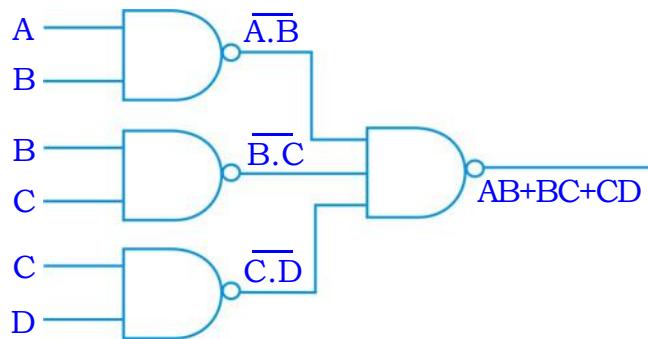
Example 2.4: Draw the diagram of digital circuit:

$$F(A, B, C) = AB + BC + CD \text{ using NAND-to-NAND logic.}$$

Solution: $F(A, B, C) = AB + BC + CD$

$$= (A \text{ NAND } B) \text{ NAND } (B \text{ NAND } C) \text{ NAND } (C \text{ NAND } D)$$

Thus logic circuit is



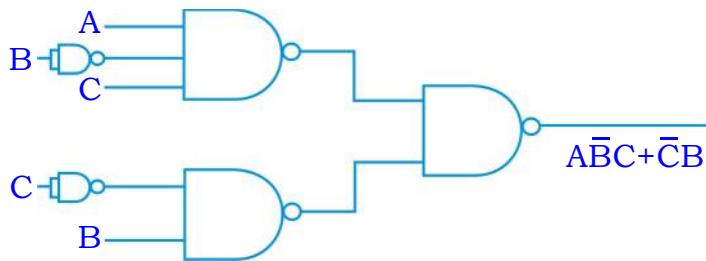
Example 2.5: Draw the circuit diagram for

$$F = A\bar{B}C + \bar{C}B \text{ using NAND-to-NAND LOGIC ONLY.}$$

Solution: $F = A\bar{B}C + \bar{C}B$

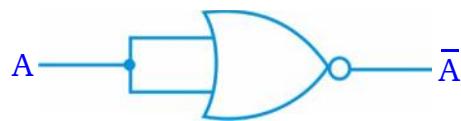
$$= ((A \text{ NAND } (\text{NOT } B) \text{ NAND } C) \text{ NAND } ((\text{NOT } C) \text{ NAND } B))$$

Circuit Diagram is



NOR-to-NOT logic

NOT, AND and OR operations can be implemented in NOR-to-NOR form as shown on below.



$$\text{NOT } X = X \text{ NOR } X$$

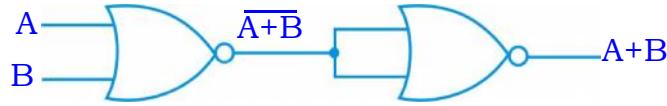
$$= \overline{X+X}$$

$$= \overline{X}\overline{X}$$

$$= \overline{X}$$

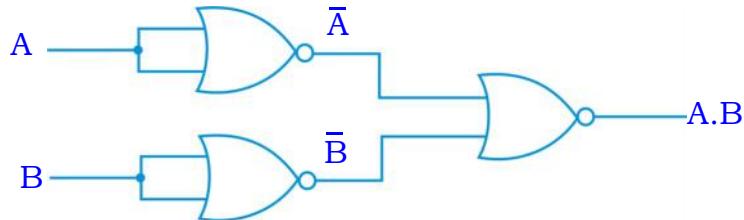
NOR to OR Operation

$$A + B = (A \text{ NOR } B) \text{ NOR } (A \text{ NOR } B)$$



NOR to AND Operation

$$A \cdot B = (A \text{ NOR } A) \text{ NOR } (B \text{ NOR } B)$$



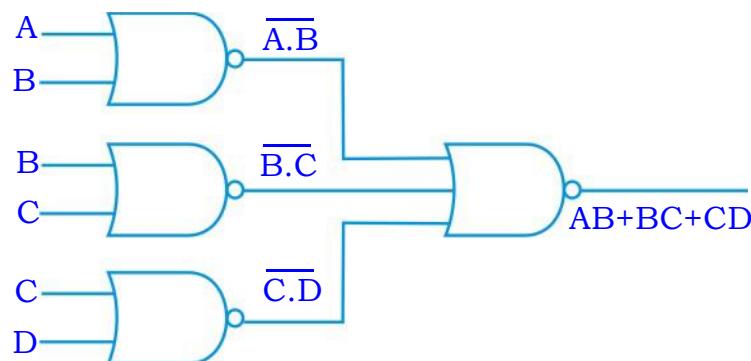
NOR-to-NOR logic is best suited for Boolean expression in Product-of-sum form.

Design rule for NOR-to-NOR logic network (only for two-level-circuits):

- 1 Derive a simplified product-of-sums form of the expression.
- 2 Draw a circuit diagram using NOT, OR and AND gates.
- 3 Finally substitute NOR gates for NOT, OR and AND gates.

Example 2.6: Represent $(X + Y)(Y + Z)(Z + X)$ in NOR-to-NOR form.

Solution: $(X + Y)(Y + Z)(Z + X) = (X \text{ NOR } Y) \text{ NOR } (Y \text{ NOR } Z) \text{ NOR } (Z \text{ NOR } X)$



One mark questions:

1. What is a logic gate?
2. Mention the three basic logic gates?
3. Which basic gate is named as Inverter?
4. Which are the three logic operations?
5. Write the standard symbol for AND gate.
6. Write the truth table for AND gate.
7. Write the logic circuit for AND gate.
8. Write the standard symbol for OR gate.
9. Write the truth table for OR gate.
10. Write the logic circuit for OR gate.
11. Write the standard symbol for NOT gate.
12. Write the truth table for NOT gate.
13. Write the logic circuit for NOT gate.
14. What is a truth table?
15. What is meant by universal gates?
16. Mention different universal gates.
17. What is the output of the two input NAND gate for the inputs:
A=0, B=1?
18. What are the values of the inputs to a three input NAND gate, if its output is 1?
19. What are the values of the inputs to a three input NAND gate, if its output is 0?
20. What is the output of the two input OR gate for the inputs: A=0, B=0?
21. What are the values of the inputs to a three input OR gate, if its output is 0?
22. What are the values of the inputs to a three input OR gate, if its output is 1?
23. For the truth table given below, what type of logic gate does the output X represent?

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

24. For the truth table given below, what type of logic gate does the output X represent?

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

25. State any one of the principle from duality of theorems.

Three Marks Questions

1. What is meant by proof by perfect induction? Give an example.
2. Write the truth table and standard symbol of NAND gate.
3. Explain the working of NAND gate .(write the output conditions)
4. Write the truth table and standard symbol of NOR gate.
5. Explain the working of NOR gate .(write the output conditions)
6. Draw the logic gate diagram to implement AND and OR gates using NAND gates only. (any two gates)
7. Draw the logic gate diagram to implement AND and OR gates using NOR gates only. (any two gates)
8. Draw the logic gate diagram to implement NOT using
(a) only NOR gates (b) only NAND gates.
9. State De Morgan's theorems.
10. What is principle of duality? Give an example.
11. Give the dual form of (any two) :
 - (a) $0 \cdot X + X \cdot Y + 1 \cdot X$
 - (b) $X \cdot (Y+Z) = X \cdot Y + X \cdot Z$
 - (c) $X + \bar{X} \cdot Y = X + Y$
 - (d) $1 + X = 1$
12. Simplify the following logical expression using De Morgan's theorems.
 - (a) $(A+B) \cdot C$
 - (b) $(A+BC) \cdot (D+EF)$

13. Prove the following rules using the proof by perfect induction.

(a) $X\bar{Y} + XY = X$

(b) $X + Y = X + Y$

14. Draw logic circuit diagram for the following expressions.

(a) $Y = AB + \bar{B}C + \bar{C}\bar{A}$

(b) $Y = \bar{X}\bar{Y} + Z\bar{X} + \bar{Y}Z$

15. Simplify the following Boolean expressions.

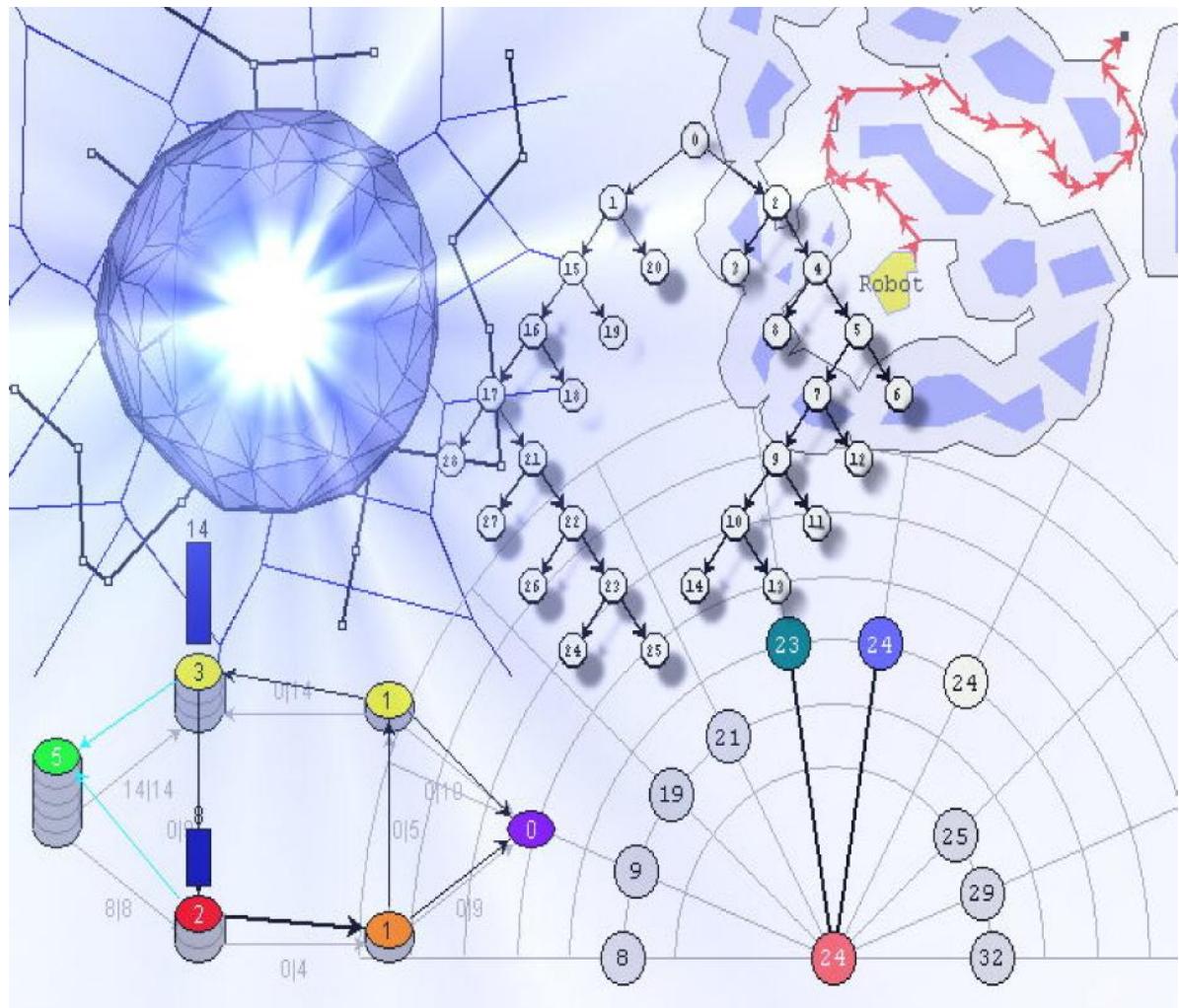
(a) $A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$

(b) $AB + A\bar{B} + \bar{A}C + \bar{A}\bar{C}$

Chapter 4 Data Structures

Objectives:

- To understand the concept of data structures
- To understand the types of data structures
- To understand the operations on different data structures
- The implementation of different data structures



4.1 Introduction

Data is a collection of raw facts that are processed by computers. Data may contain single value or set of values. For example students name may contain three sub items like first name, middle name, and last name, whereas students regno can be treated as a single item. These data have to be processed to form information.

In order to process the data the data should be organized in a particular way. This leads to structuring of data. Utilization of space by data in memory and optimization of time taken for processing, plays an important role in data structures. Data structures are efficient way to organize the data.

Data structures provide a means to manage large amount of data efficiently. Data structures are also key factor in designing efficient algorithms. In this chapter we will discuss about various types of data structures and operations performed on them.

Data structure means organization or structuring a collection of data items in appropriate form so as to improve the efficiency of storage and processing.



A data structure is a specialized format for organizing and storing data.

4.2 Data representation

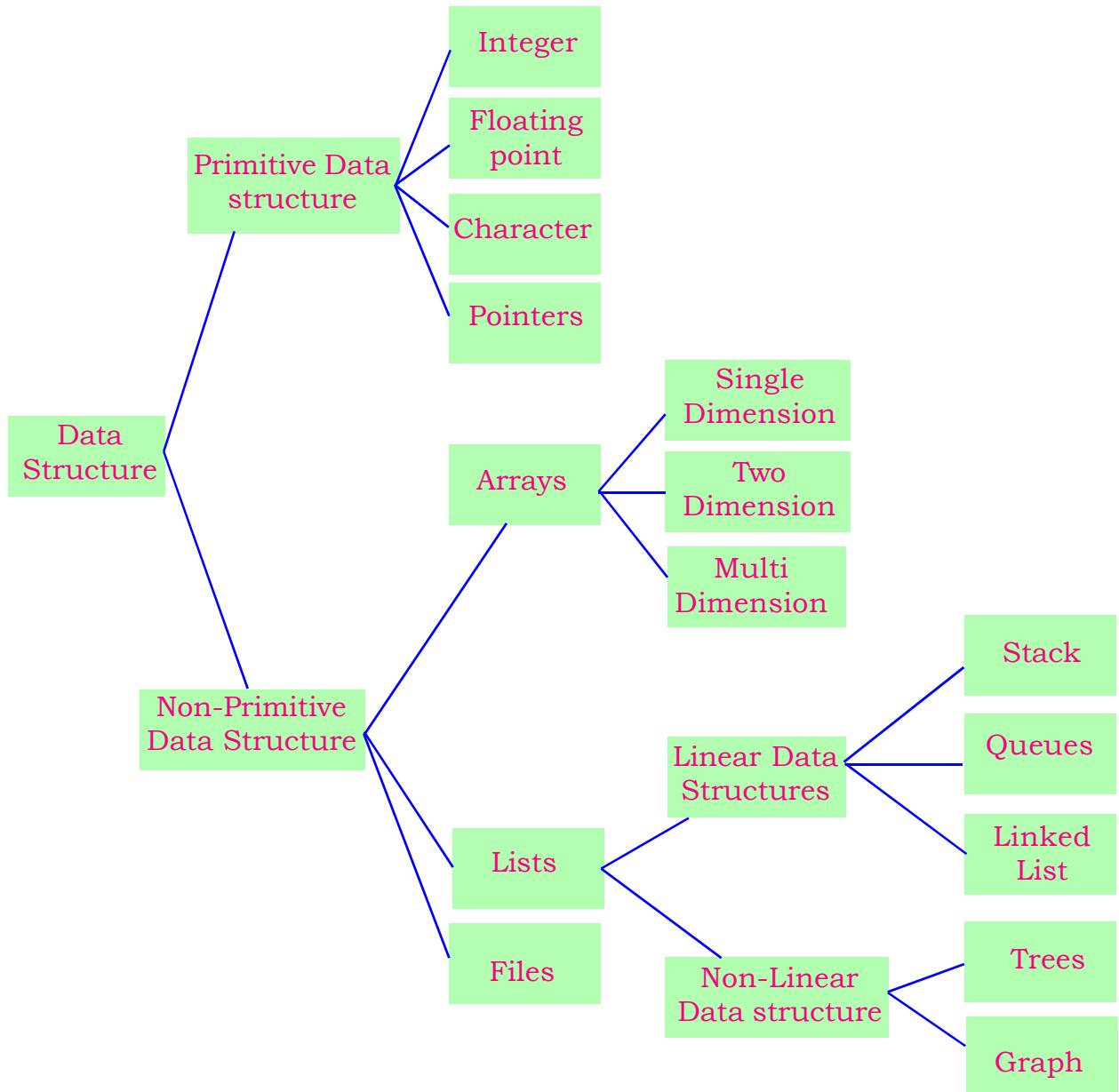
Computers memory is used to store the data which is required for processing. This process is known as data representation. Data may be of same type or different types. A need may arise to group these items as single unit. Data structures provide efficient way of combining these data types and process data.

Data structure is a method of storing data in memory so that it can be used efficiently. The study of data structure mainly deals with:

- Logical or mathematical description of structure.
- Implementation of structure on a computer.
- Analysis of structure which includes determining amount of space in memory and optimization of time taken for processing.

We consider only the first and second cases are considered in this chapter.

4.3 classification of data structures



4.3.1 Primitive data structures:

Data structures that are directly operated upon by machine-level instructions are known as primitive data structures.

The integer, real (float), logical data, character data, pointer and reference are primitive data structures.

4.3.2 Operations on primitive data structures

The various operations that can be performed on primitive data structures are:

- **Create:** Create operation is used to create a new data structure. . This operation reserves memory space for the program elements. It can be carried out at compile time and run-time.

For example, int x;

- **Destroy:** Destroy operation is used to destroy or remove the data structures from the memory space.

When the program execution ends, the data structure is automatically destroyed and the memory allocated is eventually de-allocated. C++ allows the destructor member function destroy the object.

- **Select:** Select operation is used by programmers to access the data within data structure. This operation updates or alters data.

- **Update:** Update operation is used to change data of data structures. An assignment operation is a good example of update operation.

For example, int x = 2; Here, 2 is assigned to x.

Again, x = 4; 4 is reassigned to x. The value of x now is 4 because 2 is automatically replaced by 4, i.e. updated.

4.3.3 Non primitive data structures:

Non-primitive data structures are more complex data structures. These data structures are derived from the primitive data structures. They stress on formation of groups of homogeneous and heterogeneous data elements.

Non-primitive data structures are classified as arrays, lists and files.

Array is the collection of homogenous elements under the same name. The different types of arrays are one-dimensional, two-dimensional and multi-dimensional.

Data structures under lists are classified as linear and non-linear data structures.

4.3.4 Linear data structure:

Linear data structures are a kind of data structure that has homogenous elements. Each element is referred to by an index. The linear data structures are Stack, Queues and Linked Lists.

Data elements of linear data structures will have linear relationship between data elements. One of the methods is to have linear relationship between elements by means of sequential memory locations. The other method is to have linear relationship between data elements by means of pointers or links.

4.3.5 Non-linear data structure:

A non-linear data structure is a data structure in which a data item is connected to several other data items. The data item has the possibility to reach one or more data items. Every data item is attached to several other data items in a way that is specific for reflecting relationships. The data items are not arranged in a sequential structure.

Trees and Graphs are the examples of non-linear data structures.

4.4 Operations on linear data structure

The basic operations on non-linear data structures are as follows:

- **Traversal:** The process of accessing each data item exactly once to perform some operation is called traversing.
- **Insertion:** The process of adding a new data item into the given collection of data items is called insertion.
- **Deletion:** The process of removing an existing data item from the given collection of data items is called deletion.
- **Searching:** The process of finding the location of a data item in the given collection of data items is called as searching.
- **Sorting:** The process of arrangement of data items in ascending or descending order is called sorting.
- **Merging:** The process of combining the data items of two structures to form a single structure is called merging.

4.5 Arrays

An array is a collection of homogeneous elements with unique name and the elements are arranged one after another in adjacent memory location.

The data items in an array are called as elements. These elements are accessed by numbers called as subscripts or indices. Since the elements are accessed using subscripts, arrays are also called as subscripted variables.

4.5.1 Types of Arrays:

There are three types of array:

- One-dimensional Array
 - Two-dimensional Array
 - Multi-dimensional array

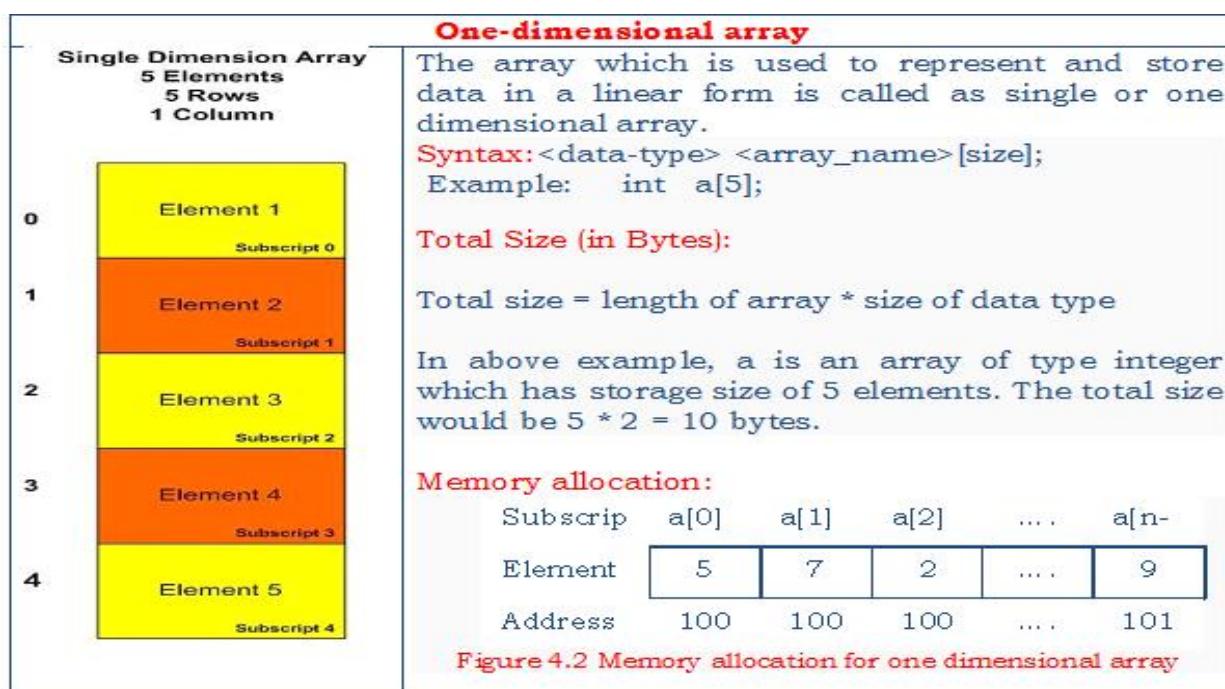
4.5.2 One-dimension Array

An array with only one row or column is called one-dimensional array. It is finite collection of n number of elements of same type such that

- Elements are stored in contiguous locations
 - Elements can be referred by indexing

Array can be denoted as: data type Arrayname[size];

Here, size specifies the number of elements in the array and the index (subscript) values ranges from 0 to n-1.



Features:

- Array size should be positive number only.
- String array always terminates with null character ('\0').
- Array elements are counted from 0 to n-1.
- Useful for multiple reading of elements.

Calculating the length of Array

Arrays can store a list of finite number (n) of data items of same data type in consecutive locations. The number n is called size or length of the array.

The length of an array can be calculated by $L = UB - LB + 1$

Here, UB is the largest index and LB is the smallest index of an array.

Example: If an array A has values 10, 20, 30, 40, 50, 60 stored in locations 0, 1, 2, 3, 4, 5 then UB = 5 and LB= 0

Size of the array $L = 5 - 0 + 1 = 6$

4.5.3 Representation of one-dimensional arrays in memory

Elements of linear array are stored in consecutive memory locations. Let P be the location of the element. Address of first element of linear array A is given by Base(A) called the **Base address** of A. Using this we can calculate the address of any element of A by the formula

$$LOC(A[P]) = \text{Base}(A) + W(P - LB)$$

Here W is the number of words per memory cell.

Example: Suppose if a string S is used to store a string ABCDE in it with starting address at 1000, one can find the address of fourth element as follows:

Address content location

1000	A	S[0]
1001	B	S[1]
1002	C	S[2]
1003	D	S[3]
1004	E	S[4]

Now the address of element S[3] can be calculated as follows:

$$\begin{aligned} \text{Address}(S[3]) &= \text{Base}(S) + W(P - LB) \text{ Here } W = \\ &1 \text{ for characters} \end{aligned}$$

$$= 1000 + 1(3 - 0)$$

$$= 1003$$

4.5.4 Basic operations on one-dimensional arrays

The following operations are performed on arrays:

1. **Traversing:** Accessing each element of the array exactly once to do some operation.
2. **Searching:** Finding the location of an element in the array.
3. **Sorting:** Arranging the elements of the array in some order.
4. **Insertion:** Inserting an element into the array.
5. **Deletion:** Removing an element from the array.
6. **Merging:** Combining one or more arrays to form a single array.

4.5.5 Traversing a Linear Array

Traversing is the process of visiting each subscript at least once from the beginning to last element.

For example, to find the maximum element of the array we need to access each element of the array.

Algorithm: Let A be a linear array with LB and UB as lower bound and upper bound. This algorithm traverses the array A by applying the operation PROCESS to each element of A.

1. for LOC = LB to UB
 PROCESS A[LOC]
 End of for
2. Exit

Program: To input and output the elements of the array.

```
#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
void main()
{
    int    a[10], i, n;
    clrscr();
    cout<<"How many elements? ";
    cin>>n;
    cout<<"Enter the elements: ";
    for(i=0; i<n; i++)
        cin>>a[i];
    cout<<"The elements are ";
    for(i=0; i<n; i++)
        cout<<setw(4)<<a[i];
    getch();
}
```

```
How many elements? 5
Enter the elements  5 10 20 15 10
The elements are  5 10 20 15 10
```

4.5.6 Searching an element

It refers to finding the location of the element in a linear array. There are many different algorithms. But the most common methods are linear search and binary search.

Linear Search

This is the simplest method in which the element to be searched is compared with each element of the array one by one from the beginning till end of the array. Since searching is one after the other it is also called as sequential search or linear search.

Algorithm: A is the name of the array with N elements. ELE is the element to be searched. This algorithm finds the location loc where the search ELE element is stored.

```
Step 1:    LOC = -1
Step 2:    for P = 0 to N-1
              if( A[P] = ELE)
                  LOC = P
                  GOTO 3
              End of if
              End of for
Step 3:    if(LOC >= 0)
                  PRINT LOC
              else
                  PRINT "Search is unsuccessful"
Step 4:    Exit
```

Program: To find the location of an element

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main( )
{
    int a[50], i, pos, ele, n;
    clrscr();
    cout<<"Enter the number of elements: ";
```

```

cin>>n;
cout<<"Enter the elements: ";
for(i=0; i<n; i++)
    cin>>a[i];
cout<<"Enter the search element: ";
cin>>ele;
pos=-1;
for(i=0; i<n ;i++)
if(ele == a[i])
{
    pos = i;
    break;
}
if(pos>= 0)
    cout<<ele<<" is present at position "<<pos<<endl;
else
    cout<<ele<<" is not present"<<endl;
getch( );
}

```

Enter the number of elements: 5
 Enter the elements: 10 20 50 40 30
 Enter the search element: 40
 40 is present at position 3

Binary Search:

When the elements of the array are in sorted order, the best method of searching is binary search. This method compares the element to be searched with the middle element of the array. If the comparison does not match the element is searched either at the right-half of the array or at the left-half of the array.

Let B and E denote the beginning and end locations of the array. The middle element A[M] can be obtained by first finding the middle location M by $M = \text{int}(B+E)/2$, where int is the integer value of the expression.

If $A[M] = ELE$, then search is successful. Otherwise a new segment is found as follows:

1. If $ELE < A[M]$, searching is continued at the left-half of the segment. Reset $E = M - 1$.
2. If $ELE > A[M]$, searching is continued at the right-half of the segment. Reset $B = M + 1$.

3. If ELE not found then we get a condition B>E. This results in unsuccessful search.

Algorithm: A is the sorted array with LB as lower bound and UB as the upper bound respectively. Let B, E, M denote beginning, end and middle locations of the segments of A.

```

Step 1:    set B = 0, E = n-1  LOC=-1
Step 2:    while (B <= E)
            M= int(B+E)/2
            if(ELE = A[M])
                loc = M
                GOTO 4
            else
                if(ELE <A[M])
                    E = M-1
                else
                    B = M+1
            End of while
Step 3:    if(LOC >= 0)
                PRINT LOC
            else
                PRINT "Search is unsuccessful"
Step 4:    Exit

```

Program: To find the location of an element.

```

#include<iostream.h>
void main()
{
    int a[10], i, n, m, loc, b, e, pos, ele;
    cout<<"How many elements? ";
    cin>>n;
    cout<<"Enter the elements: ";
    for(i=0; i<n; i++)
        cin>>a[i];
    cout<<"Enter the search element ";
    cin>>ele;
    pos=-1;
    b=0;
    e=n-1;
    while(b<=e)
    {

```

```

m=(b+e)/2;
if(ele==a[m])
{
    pos=m;
    break;
}
else
{
    if(ele<a[m])
        e=m-1;
    else
        b=m+1;
}
if(pos>=0)
    cout<<"Position= "<<pos;
else
    cout<<"Search is unsuccessful";
}

```

How many elements? 7

Enter the elements: 10 20 30 40 50 60 70

Enter the search element 60

Position= 5

4.5.7 Insertion an element

Insertion refers to inserting an element into the array. A new element can be done provided the array should be large enough to accommodate the new element.

When an element is to be inserted into a particular position, all the elements from the asked position to the last element should be shifted into the higher order positions.

Example: Let A be an array with items 10, 20, 40, 50 and 60 stored at consecutive locations. Suppose item=30 has to be inserted at position 2. The following procedure is applied.

- Move number 60 to position 5.
- Move number 50 to position 4.
- Move number 40 to position 3
- Position 2 is blank. Insert 30 into the position 2. i.e., A[2] = 30.



Algorithm: A is the array with N elements. ITEM is the element to be inserted in the position P.

```

Step 1:   for I = N-1 downto P
          A[I+1] = A[I]
          End of for
Step 2:   A[P] = ITEM
Step 3:   N = N+1
Step 4:   Exit
  
```

Program: To insert an element into the array.

```

#include<iostream.h>
#include<iomanip.h>
void main()
{
    int a[10], i, n, ele, p;

    cout<<"How many elements? ";
    cin>>n;

    cout<<"Enter the elements: ";
    for(i=0; i<n; i++)
        cin>>a[i];

    cout<<"Enter the element to be inserted: ";
    cin>>ele;

    cout<<"Enter the position ( 0 to " <<n<<" ): ";
    cin>>p;

    if(p > n)
  
```

```

        cout<<"Invalid position";
else
{
    for(i=n-1; i>=p; i--)
        a[i+1] = a[i];

    a[p] = ele;
    n = n+1;

    cout<<"The elements after the insertion are ";
    for(i=0; i<n; i++)
        cout<<setw(4)<<a[i];
}
}

```

```

How many elements? 5
Enter the elements: 1 2 4 5 6
Enter the element to be inserted: 3
Enter the position ( 0 to 5): 2
The elements after the insertion are 1 2 3 4 5 6

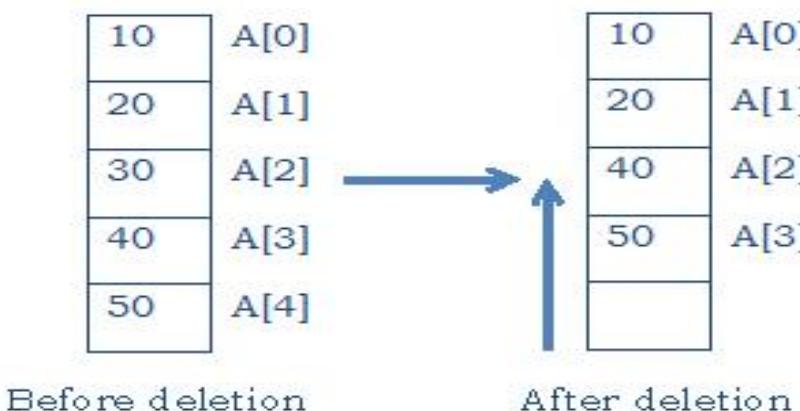
```

4.5.8 Deleting an element from the array

Deletion refers to removing an element from the array. When an element is to be deleted from a particular position, all the subsequent shifted into the lower order positions.

Example: Let A be an array with items 10, 20, 30, 40 and 50 stored at consecutive locations. Suppose item=30 has to be deleted at position 2. The following procedure is applied.

- Copy 30 to Item. i.e., Item = 30
- Move number 40 to position 2.
- Move number 50 to position 3.



Algorithm: A is the array with N elements. ITEM is the element to be deleted in the position P and it is stored into the variable Item.

Step 1: Item = A[P]
Step 2: for I = P to N-1
 A[I] = A[I+1]
 End of for
Step 2: N = N-1
Step 4: Exit

Program: To delete an element from the array.

```
#include<iostream.h>
#include<iomanip.h>
void main()
{
    int a[10], i, n, ele, p;

    cout<<"How many elements? ";
    cin>>n;

    cout<<"Enter the elements: ";
    for(i=0; i<n; i++)
        cin>>a[i];

    cout<<"Enter the position (0 to "<<n-1<<"): ";
    cin>>p;

    if(p > n-1)
        cout<<"Invalid position";
    else
    {
        ele = a[p];
        for(i=p; i<n; i++)
            a[i] = a[i+1];
        n = n-1;
        cout<<"The elements after the deletion is ";
        for(i=0; i<n; i++)
            cout<<setw(4)<<a[i];
    }
}
```

```
How many elements? 5
Enter the elements: 10 20 30 40 50
Enter the position (0 to 4): 2
The elements after the deletion is 10 20 40 50
```

4.5.9 Sorting the elements

Sorting is the arrangement of elements of the array in some order. There are various methods like bubble sort, shell sort, selection sort, quick sort, heap sort, insertion sort etc. But only insertion sort is discussed in this chapter.

Insertion Sort:

The first element of the array is assumed to be in the correct position. The next element is considered as the key element and compared with the elements before the key element and is inserted in its correct position.

Example: Consider the following list of numbers 70 30 40 10 80 stored in the consecutive locations.

Step 1: Assuming 30 in correct position 70 is compared with 30 . Since 30 is less than 70 it is inserted before 70. The list now is 30 70 40 10 80

Step 2: Now 40 is the key element. First it is compared with 70. Since 40 is less than 70 it is inserted before 70. The list now is 30 40 70 10 80

Step 3: Now 10 is the key element. First it is compared with 70. Since it is less it is exchanged. Next it is compared with 40 and it is exchanged. Finally it is compared with 30 and placed in the first position. The list now is

10 30 40 70 80

Step 4: Now 80 is the key element and compared with the sorted elements and placed in the position. Since 80 is greater than 70 it retains its position.

Algorithm: Let A be an array with N unsorted elements. The following algorithm sorts the elements in order.

```

Step 1:    for I = 1 to N-1
Step 2:          J = I
                  While ( J >= 1 )
                      If( A[J] < A[J-1])
                          temp = A[J]
                          A[J] = A[J-1]
                          A[J-1] = temp
                      If end
                      J = J-1
                  While end
              for end
Step 3:    Exit

```

4.5.10 Two-dimensional arrays

A two dimensional array is a collection of elements and each element is identified by a pair of indices called as subscripts. The elements are stored in contiguous memory locations.

We logically represent the elements of two-dimensional array as rows and columns. If a two-dimensional array has m rows and n columns then the elements are accessed using two indices I and J, where $0 \leq I \leq m-1$ and $0 \leq J \leq n-1$. Thus the expression $A[I][J]$ represent an element present at I^{th} -row and J^{th} -column of the array A. The number of rows and columns in a matrix is called as the order of the matrix and denoted as $m \times n$.

The number of elements can be obtained by multiplying number of rows and number of columns.

	[0]	[1]	[2]
A[0]	2	-2	3
A[1]	1	0	-3
A[2]	2	2	5

In the above figure, the total number of elements in the array would be, rows \times columns = $3 \times 3 = 9$ -elements.

Representation of 2-dimensional array in memory

Suppose A is the array of order $m \times n$. To store $m \times n$ number of elements, we need $m \times n$ memory locations. The elements should be in contiguous memory locations.

There are two methods:

- Row-major order
- Column-major order

Row-major order

Let A be the array of order $m \times n$. In row-major order, all the first-row elements are stored in sequential memory locations and then all the second-row elements are stored and so on.

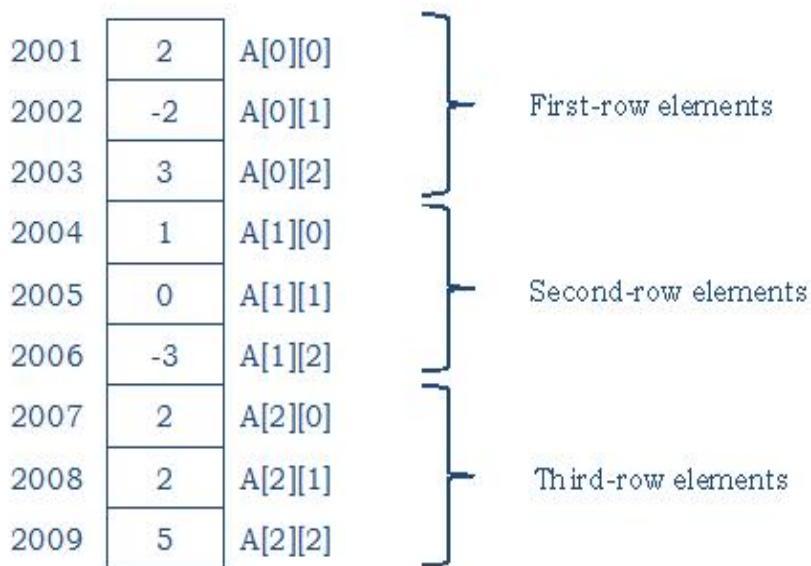
Base(A) is the address of the first element. The memory address of any element $A[I][J]$ can be obtained by the formula

$$\text{LOC}(A[I][J]) = \text{Base}(A) + W[n(I-LB) + (J-LB)]$$

where W is the number of words per memory location.

Example: Consider the array of order 3×3 .

	[0]	[1]	[2]
A[0]	2	-2	3
A[1]	1	0	-3
A[2]	2	2	5



Column-major order

Let A be the array of order $m \times n$. In column-major order, all the first-column elements are stored in sequential memory locations and then all the second-column elements are stored and so on.

Base(A) is the address of the first element. The memory address of any element A[I][J] can be obtained by the formula

$$LOC(A[I][J]) = Base(A) + W[(I-LB) + m(J-LB)]$$

where W is the number of words per memory location.

Example: Consider the array of order 3×3 .

	[0]	[1]	[2]
A[0]	2	-2	3
A[1]	1	0	-3
A[2]	2	2	5

2001	2	A[0][0]		FIRST COLUMN ELEMENTS
2002	1	A[1][0]		
2003	2	A[2][0]		
2004	-2	A[0][0]		SECOND COLUMN ELEMENTS
2005	0	A[1][0]		
2006	2	A[2][0]		
2007	3	A[0][0]		THIRD COLUMN ELEMENTS
2008	-3	A[1][0]		
2009	5	A[2][0]		

Program: To read and print the elements in column-major order.

```
#include<iostream.h>
#include<iomanip.h>
void main()
{
    int a[5][5], i, j, r, c;

    cout<<"Enter the order: ";
    cin>>r>>c;

    cout<<"Enter the elements: "<<endl;
    for(i=0; i<c; i++)
        for(j=0; j<r; j++)
            cin>>a[j][i];

    cout<<"The matrix in column-major order is: "<<endl;
    for(i=0; i<r; i++)
    {
        for(j=0; j<c; j++)
            cout<<setw(4)<<a[i][j];
        cout<<endl;
    }
}
```

```

Enter the order: 2 3
Enter the elements:
1 2 3
4 5 6
The matrix in column-major order is:
1   2   3
4   5   6

```

Example: Consider the array A of order 25×4 with base value 2000 and one word per memory location. Find the address of $A[12][3]$ in row-major order and column-major order.

Solution:

$$\begin{aligned} \text{Given } & \quad \text{Base}(A) = 2000, \quad m = 25, \quad n = 4 \quad LB = 0 \\ & \quad W = 1, \quad I = 12, \quad J = 3 \end{aligned}$$

Row-major order:

$$\begin{aligned}
\text{LOC}(A[I][J]) &= \text{Base}(A) + W[n(I-LB) + (J-LB)] \\
\text{LOC}(A[12][3]) &= 2000 + 1[4(12-0)+(3-0)] \\
&= 2000 + 4(12) + 3 \\
&= 2000 + 48 + 3 \\
&= 2051
\end{aligned}$$

Column-major order:

$$\begin{aligned}
\text{LOC}(A[I][J]) &= \text{Base}(A) + W[(I-LB) + m(J-LB)] \\
\text{LOC}(A[12][3]) &= 2000 + 1[(12-0)+25(3-0)] \\
&= 2000 + 1(12 + 75) \\
&= 2000 + 87 \\
&= 2087
\end{aligned}$$

Applications of arrays

1. Arrays are used to implement other data structures such as heaps, hash tables, queues, stacks and strings etc.
2. Arrays are used to implement mathematical vectors and matrices.
3. Many databases include one-dimensional arrays whose elements are records.

Advantages of arrays

1. It is used to represent multiple data items of same type by using only single name.

2. It can be used to implement other data structures like linked lists, stacks, queues, trees, graphs etc.
3. Two-dimensional arrays are used to represent matrices.

Disadvantages of arrays

1. We must know in advance that how many elements are to be stored in array.
2. Array is static structure. It means that array is of fixed size. The memory which is allocated to array cannot be increased or reduced.
3. Since array is of fixed size, if we allocate more memory than requirement then the memory space will be wasted. If we allocate less memory than requirement, then it will create problem.
4. The elements of array are stored in consecutive memory locations. So insertions and deletions are very difficult and time consuming.

STACKS

4.6.1 Introduction

A **stack** is an ordered collection of items where the addition of new items and the removal of existing items always take place at the same end. This end is commonly referred to as the “top”. The end opposite to top is known as the **base**.

The base of the stack is significant since items stored in the stack that are closer to the base represent those that have been in the stack the longest. The most recently added item is the one that is in position to be removed first. This ordering principle is sometimes called **LIFO, last-in first-out**. Newer items are near the top, while older items are near the base.

Many examples of stacks occur in everyday situations. Almost any cafeteria has a stack of trays or plates where you take the one at the top, uncovering a new tray or plate for the next customer in line. Imagine a stack of books on a desk. The only book whose cover is visible is the one on top. To access others in the stack, we need to remove the ones that are placed on top of them. Another Figure shows another stack. This one contains a number of primitive data objects.

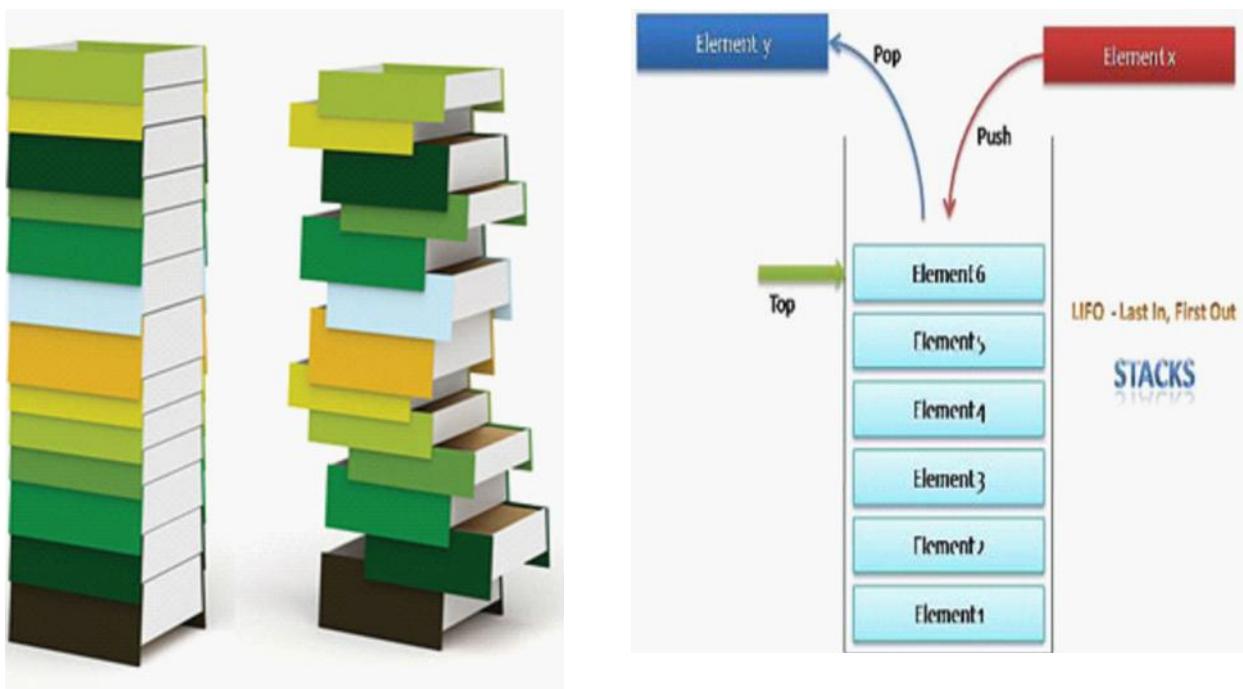


Figure A Stack of Books

One of the most useful ideas related to stacks comes from the simple observation of items as they are added and then removed. Assume you start out with a clean desktop. Now, place books one at a time on top of each other. You are constructing a stack. Consider what happens when you begin removing books. The order that they are removed is exactly the reverse of the order that they were placed. Stacks are fundamentally important, as they can be used to reverse the order of items. The order of insertion is the reverse of the order of removal.

Considering this reversal property, you can perhaps think of examples of stacks that occur as you use your computer. For example, every web browser has a Back button. As you navigate from web page to web page, those pages are placed on a stack (actually it is the URLs that are going on the stack). The current page that you are viewing is on the top and the first page you looked at is at the base. If you click on the Back button, you begin to move in reverse order through the pages.

4.6.2 Representation of stacks in memory

The representation of a stack in the memory can be done in two ways.

- Static representation using arrays
- Dynamic representation using linked lists

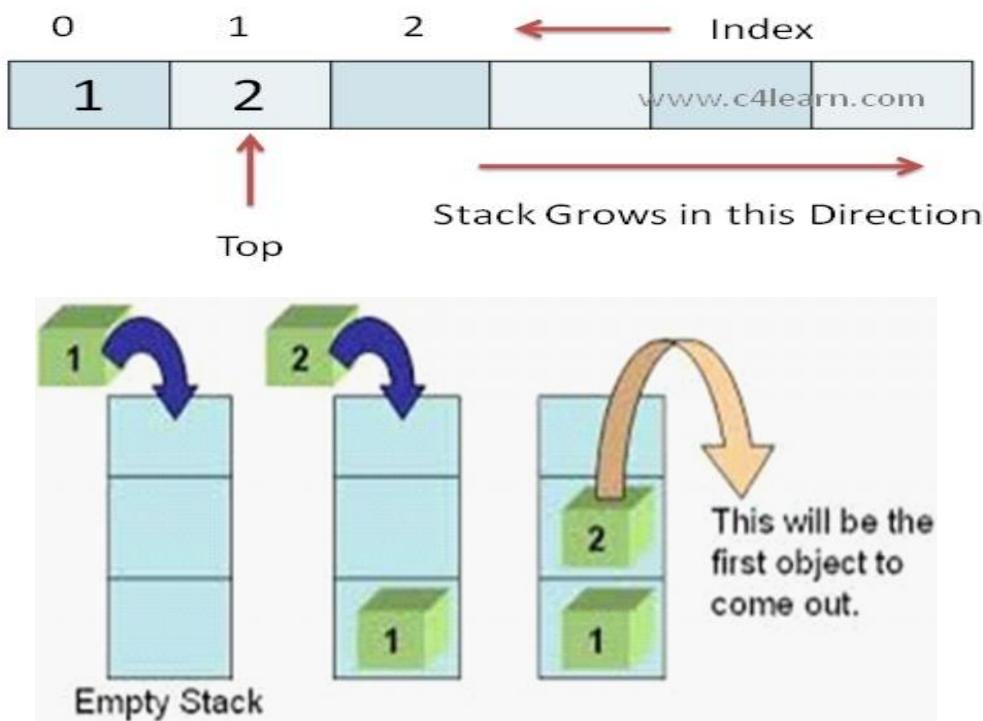
Array Representation of a Stack

Stack can be represented using a one-dimensional array. A block of memory is allocated which is required to accommodate the items to the full capacity of the stack. The items into the stack are stored in a sequential order from the first location of the memory block.

A pointer TOP contains the location of the top element of the stack. A variable MAXSTK contains the maximum number of elements that can be stored in the stack.

The condition $\text{TOP} = \text{MAXSTK}$ indicates that the stack is full and $\text{TOP} = \text{NULL}$ indicates that the stack empty.

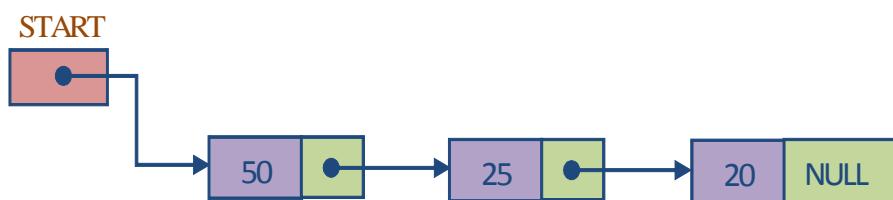
Representing a stack using arrays is easy and convenient. However, it is useful for fixed sized stacks. Sometimes in a program, the size of a stack may be required to increase during execution, i.e. dynamic creation of a stack. Dynamic creation of a stack is not possible using arrays. This requires linked lists.



Linked list representation of a Stack

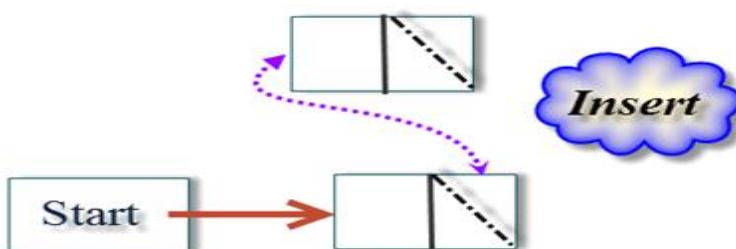
The size of the array needs to be fixed to store the items into the stack. If the stack is full we cannot insert an additional item into the array. It gives an overflow exception. But in linked list we can increase the size at runtime by creating a new node. So it is better to implement stack data structure using

Linked list data structure. The linked list structure will be studied in detail later.



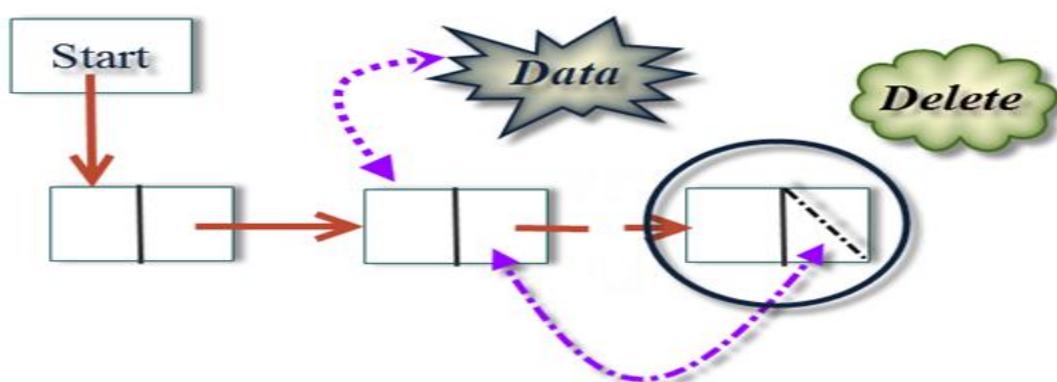
Insertion

Insertion operation refers to Inserting an element into stack. We create a new node and insert an element into the stack. To follow the stack principle "**Last-in-first-out**", a node need to be created from the end of the Linked list and element need to be inserted into the node from the end of the linked list.



Deletion

Deletion operation is to delete an element or node from the Linked list. Deletion can be done by deleting the top-most item from the stack as the last item inserted is the first item that needs to be deleted as per stack principle. So the recently inserted item i.e, top item must be deleted from the linked list to perform as stack deletion.



4.6.3 Operation Stack

The stack abstract data type is defined by the following structure and operations. A stack is structured, as described above, as an ordered collection of items where items are added to and removed from the end called the “top.” Stacks are ordered LIFO. The **stack operations** are given below.

- **stack()** creates a new stack that is empty. It needs no parameters and returns an empty stack.
- **push(item)** adds a new item to the top of the stack. It needs the item and returns nothing.
- **pop()** removes the top item from the stack. It needs no parameters and returns the item. The stack is modified.
- **peek()** returns the top item from the stack but does not remove it. It needs no parameters. The stack is not modified.
- **isEmpty()** tests whether the stack is empty. It needs no parameters and returns a Boolean value.
- **size()** returns the number of items on the stack. It needs no parameters and returns an integer.

Algorithm for PUSH Operation: PUSH(STACK , TOP, SIZE, ITEM)

STACK is the array that contains N elements and TOP is the pointer to the top element of the array. ITEM the element to be inserted. This procedure inserts ITEM into the STACK.

Step 1:	If TOP = N -1 then	[check overflow]
	PRINT “Stack is full”	
	Exit	
	End of If	
Step 2:	TOP = TOP + 1	[Increment the TOP]
Step 3:	STACK[TOP] = ITEM	[Insert the ITEM]
Step 4:	Return	

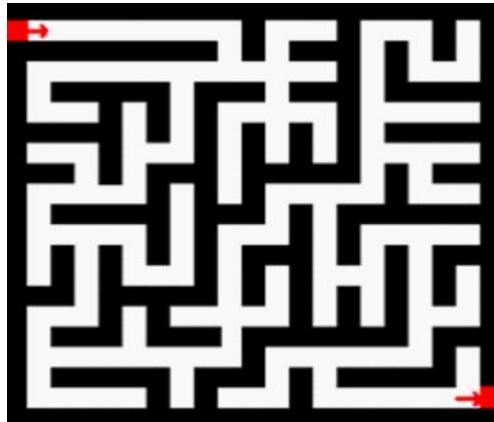
Algorithm for POP Operation: POP(STACK , TOP, ITEM)

STACK is the array that store N items. TOP is the pointer to the top element of the array. This procedure deleted top element from STACK.

Step 1:	If TOP = NULL then PRINT "Stack is empty" Exit End of If	[check underflow]
Step 2:	ITEM = STACK[TOP]	[Copy the top element]
Step 3:	TOP = TOP - 1	[Decrement the top]
Step 4:	Return	

4.6.4 Application of Stacks

- The simplest application of a stack is to reverse a word. You push a given word to stack - letter by letter - and then pop letters from the stack.
- Another application is an “undo” mechanism in text editors; this operation is accomplished by keeping all text changes in a stack.
- Backtracking



Backtracking: This is a process when you need to access the most recent data element in a series of elements. Think of a labyrinth or maze - how do you find a way from an entrance to an exit?

Once you reach a dead end, you must backtrack. But backtrack to where? to the previous choice point. Therefore, at each choice point you store on a stack all possible choices. Then backtracking simply means popping a next choice from the stack.

- Language processing:
 - Space for parameters and local variables is created internally using a stack.
 - Compiler's syntax check for matching braces is implemented by using stack.
 - support for recursion
- Conversion of decimal number into binary
- To solve tower of Hanoi
- Expression evaluation and syntax parsing
- Conversion of infix expression into prefix and postfix.
- Rearranging railroad cars

- Quick sort
- Stock span problem
- Runtime memory management

Arithmetic expression: An expression is a combination of operands and operators that after evaluation results in a single value. Operands consist of constants and variables. Operators consists of {, +, - , * , / etc.,) [Expressions can be

- Infix expression
- Post fix expression
- Prefix expression

Infix expression: If an operator is in between two operands it is called infix expression.

Example: $a + b$, where a and b are the operands and + is an operator.

Postfix expression: If an operator follows the two operands it is called post fix expression.

Example: ab +

Prefix expression: If an operator precedes two operands, it is called prefix expression.

Example: +ab

Algorithm for Infix to Postfix

1. Examine the next element in the input.
2. If it is operand, output it.
3. If it is opening parenthesis, push it on stack.
4. If it is an operator, then
 - If stack is empty, push operator on stack.
 - If the top of stack is opening parenthesis, push operator on stack
 - If it has higher priority than the top of stack, push operator on stack.
 - Else pop the operator from the stack and output it, repeat step 4.
5. If it is a closing parenthesis, pop operators from stack and output them until an opening parenthesis is encountered. Pop and discard the opening parenthesis.
6. If there is more input go to step 1.

7. If there is no more input, pop the remaining operators to output.

Example: Suppose we want to convert $2*3/(2-1)+5*3$ into Postfix form.

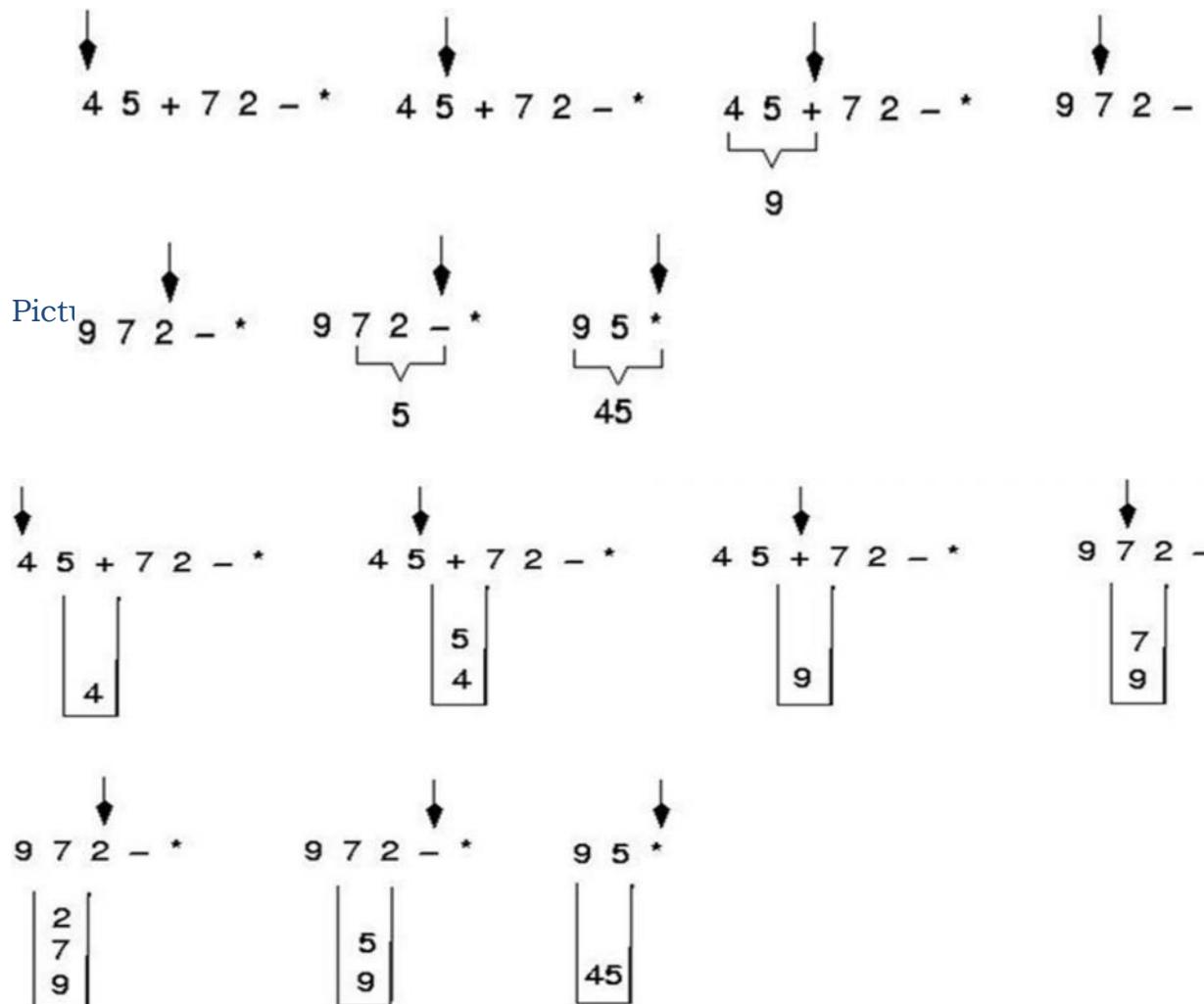
Expression	Stack	Output
2	Empty	2
*	*	2
3	*	23
/	/	23*
(/(23*
2	/(23*2
-	/(-	23*2
1	/(-	23*21
)	/	23*21-
+	+	23*21-/
5	+	23*21-/5
*	+*	23*21-/53
3	+*	23*21-/53
	Empty	23*21-/53*+

So, the Postfix Expression is $23*21-/53*+$

Evaluating a postfix expression using stack

- The postfix expression to be evaluated is scanned from left to right.
- Each operator in a postfix string refers to the previous two operands in the string.
- Each time we read an operand we push it into a stack. When we reach an operator, its operands will then be top two elements on the stack.
- We can then pop these two elements, perform the indicated operation on them, and push the result on the stack.
- So that it will be available for use as an operand of the next operator.
- Initialize an empty stack.
- While character remain in the input stream
 - Read next character.
 - If character is an operand, push it into the stack.

- Else, if character is an operator, pop top two characters off the stack, apply the operator, and push the answer back into the stack.
- Pop the answer off the stack.



Algorithm for evaluating a postfix expression

WHILE more input items exist

{

```

        If symb is an operand then
            push (operand_stk, symb)
        else                                //symbol is an operator
        {
            Opnd1 = pop(operand_stk);
            Opnd2 = pop(operand_stk);
            Value = opnd2 symb opnd1
            Push(operand_stk, value);
        }
    }      Result = pop (operand_stk);
  
```

Consider an expression S having operators, operands, left parenthesis and right parenthesis. Operators includes +, - , /, *. Evaluations are performed from left to right otherwise indicated by parenthesis. Stack is used to hold operators and left parenthesis. The postfix expression can be obtained from left to right using operands from } the operators which are removed from STACK. Left parentheses are pushed onto stack and right parenthesis at the end of S. Algorithm is completed when STACK is empty.

Example 1: Convert $A + (B * C - (D/E^ F))$ into postfix notation.

Symbol scanned	stack	expression
1. A	(A
2. +	(+	A
3. ((+(A
4. B	(+()	A B
5. *	(+(*	A B
6. C	(+(*	A B C
7. -	(+(-	A B C *
8. ((+(-()	A B C *
9. D	(+(-()	A B C * D
10. /	(+(-() /	A B C * D
11. E	(+(-() /	A B C * D E
12. ^	(+(-() / ^	A B C * D E
13. F	(+(-() / ^	A B C * D E F
14.)	(+(-	A B C * D E F ^ /

Examples 2: Convert the following infix expressions postfix notation.

+	((+	A
B	((+	AB
)	(AB+
*	(*	AB+
((*()	AB+
C	(*()	AB+C
-	(*(-	AB+C
D	(*(-)	AB+CD
)	(*	AB+CD-
/	(/	AB+CD-*
E	(/	AB+CD-*E
)		AB+CD-*E/

Answer: Postfix expression of $(A+B)*(C-D)/E$ is $AB+CD-*E/$

Infix Expression	Prefix Expression
A +B -C	-+ABC
(A+B) *(C+D)	*+AB+CD
A/B*C-D+E/F/(G+H)	+*/ABCD//EF+GH
((A + B) * C - (D-E)) * (F+G)	*-+*+ABC-DE +FG
A-B/(C *D/E)	-A/B/*CDE

Queues

4.7.1 Introduction

We now turn our attention to another linear data structure. This one is called **queue**. Like stacks, queues are relatively simple and yet can be used to solve a wide range of important problems.

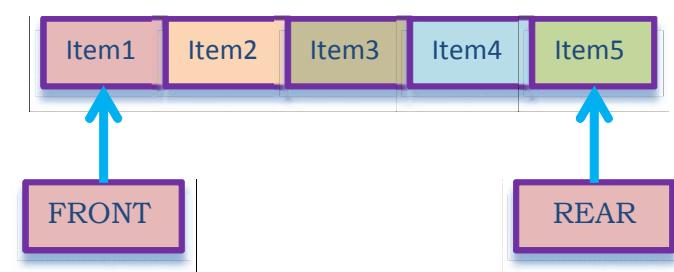
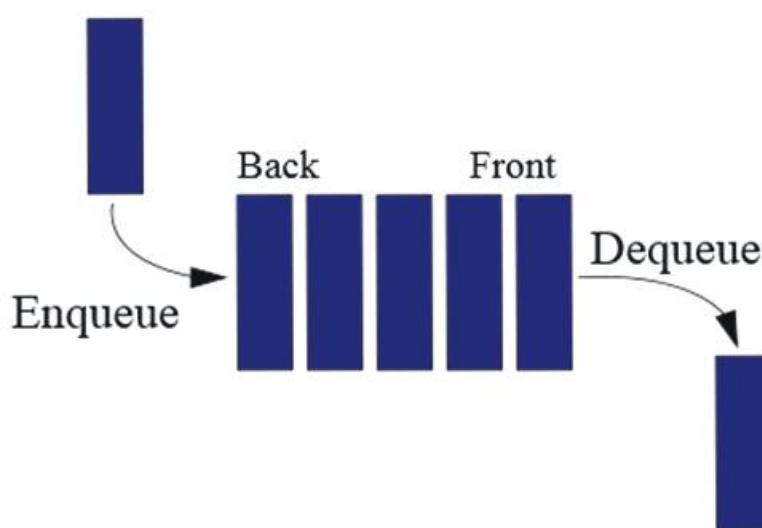
A **queue** is an ordered collection of items where the addition of new items and the removal of existing items always take place at different ends.

A queue is an ordered collection of items where an item is inserted at one end called the “rear,” and an existing item is removed at the other end, called the “front.” Queues maintain a FIFO ordering property.

Insertion and deletion is performed according to the first-in first-out (FIFO) principle. An excellent example of a queue is a line of students in the food court of a canteen. New additions to the line are made to the back of the queue, while removal (or serving) happens in the front. In the queue only two operations are allowed **enqueue** and **dequeue**. Enqueue means to insert an item into the back of the queue, dequeue means removing the front item. The picture demonstrates the FIFO access.

The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

Queue is also called as FIFO list, i.e. **First-In-First-Out**. Here insertions are limited to one end of the list called the **rear**, whereas deletions are limited to other end called as **front**.

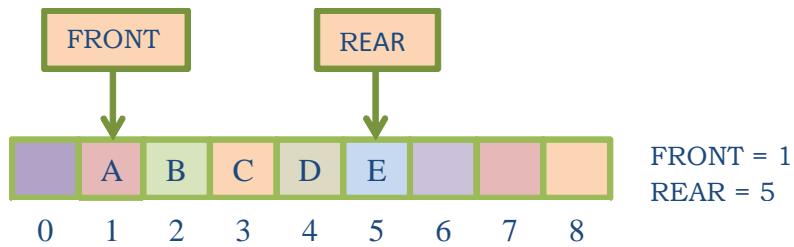


4.7.2 Types of queues

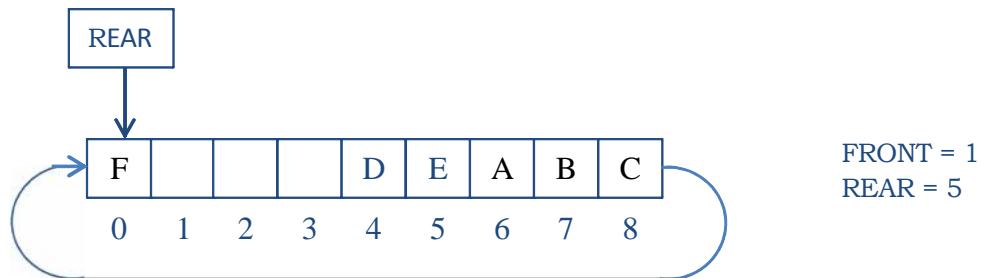
Queue can be of four types:

1. Simple Queue
2. Circular Queue
3. Priority Queue
4. Dequeue (Double Ended queue)

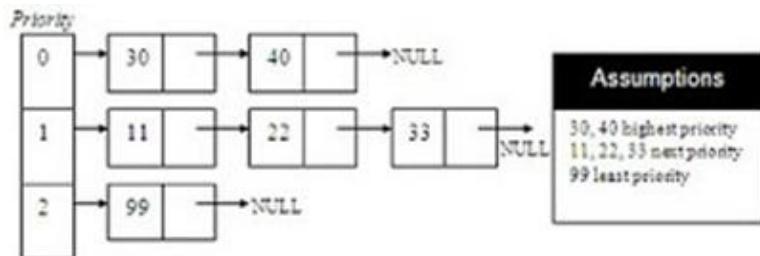
- 1. Simple Queue:** In Simple queue insertion occurs at the rear end of the list, and deletion occurs at the front end of the list.



- 2. Circular Queue:** A circular queue is a queue in which all nodes are treated as circular such that the last node follows the first node.



- 3. Priority Queue:** A priority queue is a queue that contains items that have some preset priority. An element can be inserted or removed from any position depending on some priority.



- 4. Dequeue (Double Ended queue):**

It is a queue in which insertion and deletion takes place at both the ends.



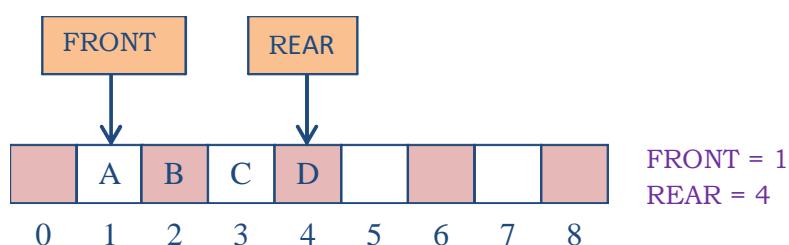
4.7.3 Operations on queue

The queue abstract data type is defined by the following structure and operations. The queue operations are given below.

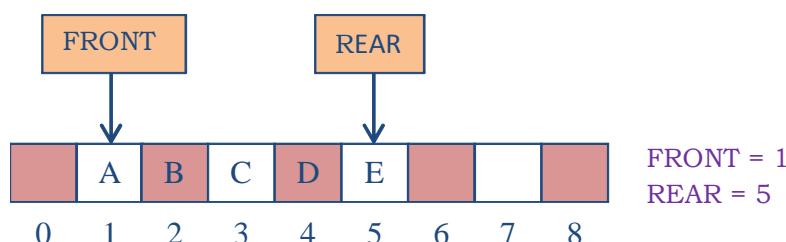
- **Queue()** creates a new queue that is empty. It needs no parameters and returns an empty queue.
- **enqueue(item)** adds a new item to the rear of the queue. It needs the item and returns nothing. This operation is generally called as **push**.
- **dequeue()** removes the front item from the queue. It needs no parameters and returns the item. The queue is modified. This operation is generally called as **pop**.
- **isEmpty()** tests to see whether the queue is empty. It needs no parameters and returns a Boolean value.
- **size()** returns the number of items in the queue. It needs no parameters and returns an integer.

4.7.4 Memory representation of a queue using arrays

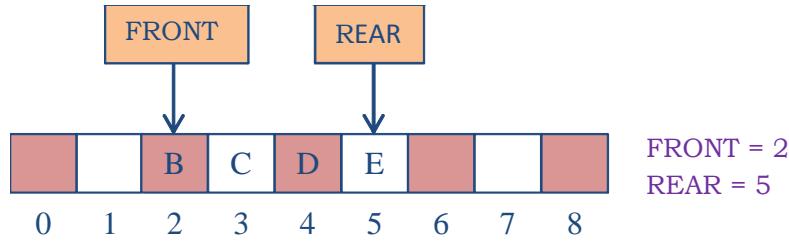
Queue is represented in memory linear array. Let QUEUE be a linear queue. Two pointer variables called FRONT and REAR are maintained. The pointer variable FRONT contains the location of the element to be removed and the pointer variable REAR contains location of the last element inserted. The condition FRONT = NULL indicates that the queue is empty and the condition REAR = N indicates that the queue is full.



$\text{REAR} = \text{REAR} + 1$, $\text{QUEUE}[\text{REAR}] = 'E'$

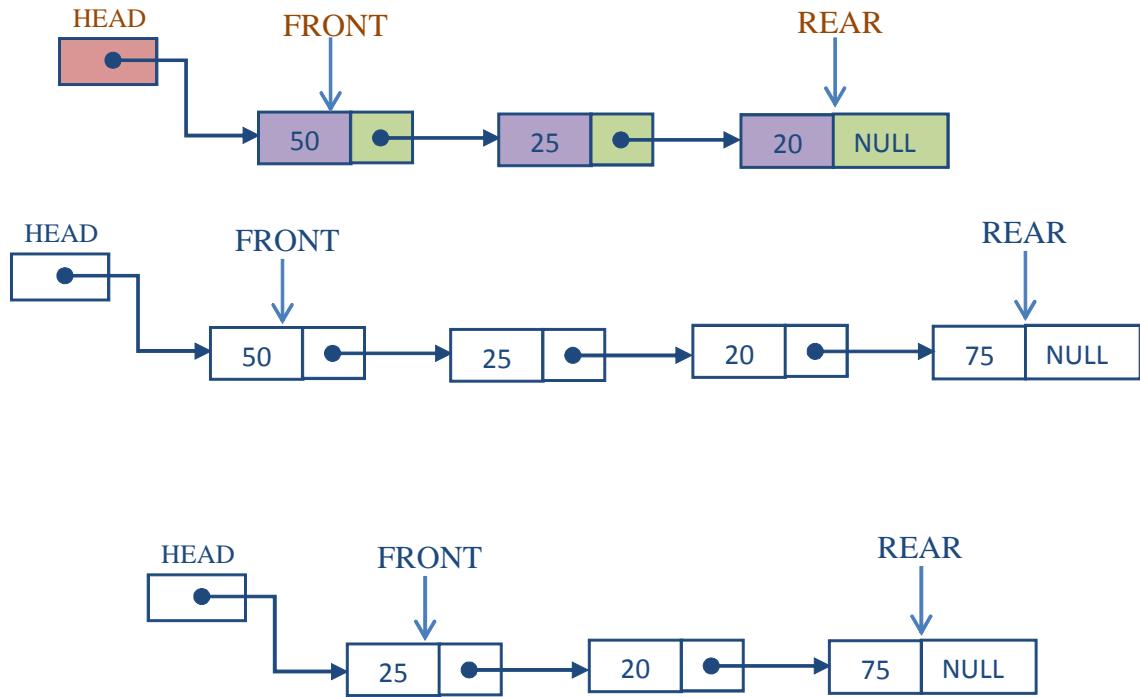


ITEM = QUEUE[FRONT], FRONT = FRONT + 1



Memory representation of a queue using linked list

Queues are also represented using linked lists. In queues an item should be inserted from rear end and an item should be removed from the front end. The pointer front contains location of the first node of the linked list and another pointer rear contains location of the last node.



Algorithm for insertion

Algorithm: Let QUEUE be the linear array consisting of N elements. FRONT is the pointer that contains the location of the element to be deleted and REAR contains the location of the inserted element. ITEM is the element to be inserted.

```

Step 1: If REAR = N-1 Then      [Check for overflow]
        PRINT "Overflow"
        Exit
Step 2: If FRONT = NULL Then   [Check whether QUEUE is empty]
        FRONT = 0
        REAR = 0
        Else
            REAR = REAR + 1      [Increment REAR Pointer]
Step 3: QUEUE[REAR] = ITEM    [Copy ITEM to REAR position]
Step 4: Return

```

Algorithm for deletion

Algorithm: Let QUEUE is the linear array consisting of N elements. FRONT is the pointer that contains the location of the element to be deleted and REAR contains the location of the inserted element. This algorithm deletes the element at FRONT position.

```

Step 1: If FRONT = NULL Then  [Check whether QUEUE is empty]
        PRINT "Underflow"
        Exit
Step 2 : ITEM = QUEUE[FRONT]
Step 3: If FRONT = REAR Then  [If QUEUE has only one element]
        FRONT = NULL
        REAR = NULL
        Else
            FRONT = FRONT + 1    [Increment FRONT pointer]
Step 3: Return

```

4.7.5 Applications of queues

- Simulation
- Various features of operating system.
[Operating systems often maintain a queue of processes that are ready to execute or that are waiting for a particular event to occur.]
- Multi-programming platform systems
- Different type of scheduling algorithm
- Round robin technique or Algorithm
- Printer server routines
- Various applications software is also based on queue data structure
- Operating systems often maintain a queue of processes that are ready to execute or that are waiting for a particular event to occur.

LINKED LISTS

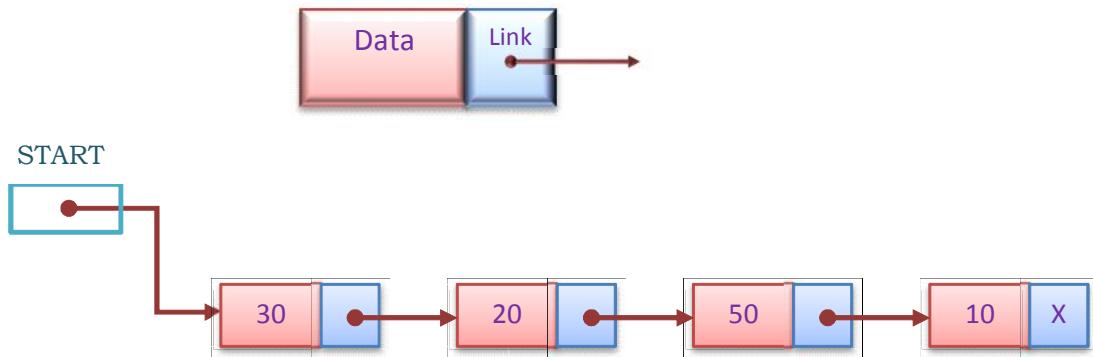
4.8.1 Introduction

One disadvantage of using arrays is that arrays are **static** structures and therefore cannot be easily extended or reduced to fit the data set. During insertion, the array elements are shifted into higher order memory locations and during deletion, elements are shifted into lower order memory locations. In this chapter we study another data structure called Linked Lists that addresses these limitations of arrays. The linked list uses **dynamic memory allocations**.

Linked list

A linked list is a linear collection of data elements called nodes and the linear order is given by means of pointers.

Each node contains two parts fields: the data and a reference to the next node. The first part contains the information and the second part contains the address of the next node in the list. This is also called the link field.



The above picture is linked list with 4 nodes, each node is contains two parts. The left part of the node represents **the information part** of the node and the right part represents the **next pointer field** that contains the address of the next node. A pointer START gives the location of the first node. This pointer is also represented as HEAD. Note that the link field of the last node contains NULL.

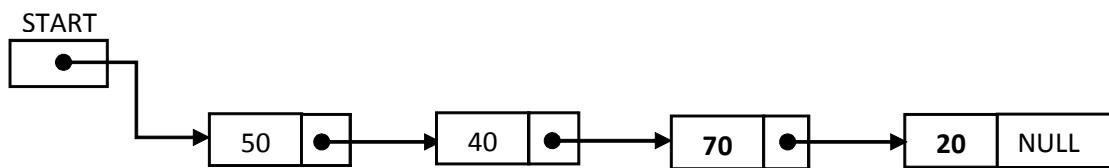
4.8.2 Types of linked lists

There are three types of linked lists.

1. Singly linked list (SLL)
2. Doubly linked list (DLL)
3. Circular linked list (CLL)

Single linked list

A singly linked list contains two fields in each node – the data field and link field. The data field contains the data of that node while the link field contains address of the next node. Since there is only one link field in each node, the linked list is called as singly linked list.



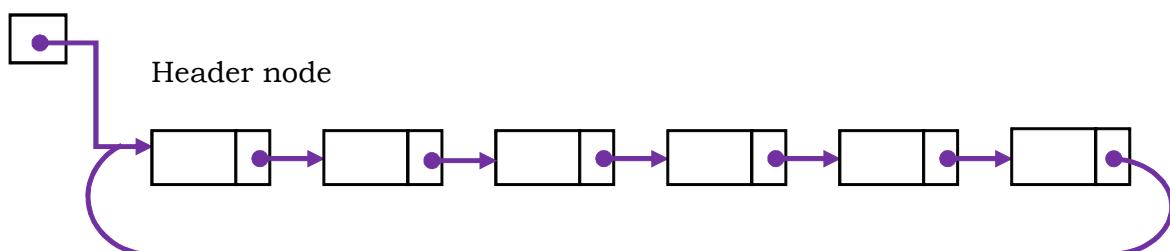
In any linked list the nodes need not necessarily represent a set of consecutive memory locations (or contiguous memory locations).

Circular linked lists:

In a singly linked list, a pointer is available to access all the succeeding nodes, but not preceding nodes. In the singly linked lists, the link field of the last node contains NULL.

In circular lists, if the link field of the last node contains the address of the first node first node, such a linked list is called as circular linked list.

In a circular linked list, it is possible to reach any node from any other node.



Doubly linked lists:

It is a linked list in which each node is points both to the next node and also to the previous node.

In doubly linked list each node contains three parts – FORW, BACK and INFO.

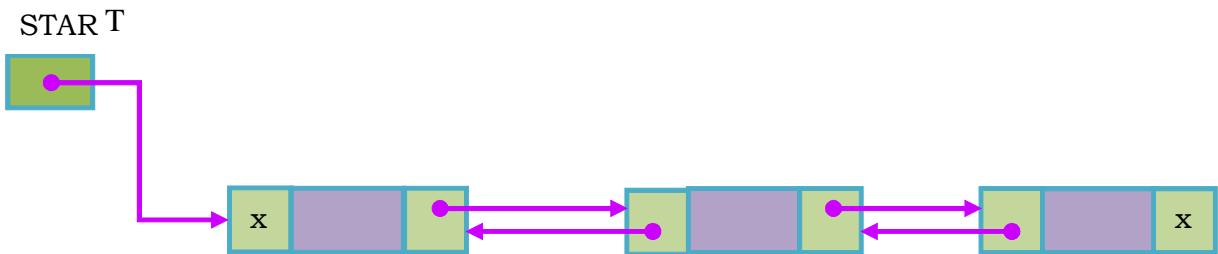


BACK: It is a pointer field containing the address of the previous node.

FORW: It is a pointer field that contains the address of the next node.

INFO: It contains the actual data.

In the first node, if BACK contains NULL, it indicates that it is the first node in the list. The node in which FORW contains NULL indicates that the node is the last node in the linked list.



In our discussion, we will only study the singly linked list.

4.8.3 Operations on linked lists:

The operations that are performed on linked lists are

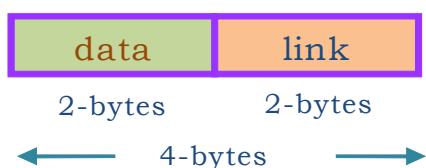
1. Creating a linked list
 2. Traversing a linked list
 3. Inserting an item into a linked list
 4. Deleting an item from the linked list
 5. Searching an item in the linked list
 6. Merging two or more linked lists

Creating a linked list

Linked list is linear data structure which contains a group of nodes and the nodes are sequentially arranged. Nodes are composed of data and address of the next node or reference of the next node. These nodes are sequentially or linearly arrayed that is why the Linked list is a linear data structure. In linked list we start with a node and create nodes and link to the starting node in order and sequentially. The pointer START contains the location of the first node. Also the next pointer field of the last node must be assigned to NULL. In this topic we will be discussing about Single Linked List. Elements can be inserted anywhere in the linked list and any node can be deleted.

The nodes of a linked list can be created by the following structure declaration.

```
struct Node  
{  
    int data;  
    Node* link;  
}  
Node *node1, *node2;
```



Here data is the information field and link is the link field. The link field contains a pointer variable that refers to the same node structure. Such a reference is called as self-addressing pointer.

The above declaration creates two variable structures. Each structure will be taken as node.

Thus it is linked list that contains two nodes. Another node cannot be inserted if it is not already declared. Also, there may be cases where the memory needs of a program can only be determined during runtime. On these cases, programs need to dynamically allocate memory, for which the C++ language uses the operators **new** and **delete**.

- Operator new allocates space.
- Operator new[] allocates memory space for array.
- Operator delete deallocate storage space.
- Operator delete[] deallocate memory space for array.

Operator new and new[]

Dynamic memory is allocated using operator new. new is followed by a data type specifier and, if a sequence of more than one element is required, the number of these within brackets []. It returns a pointer to the beginning of the new block of memory allocated.

Syntax: pointer = new type
 pointer = new type [number_of_elements]

The first expression is used to allocate memory to contain one single element of the required data type. The second one is used to allocate a block (an array) of elements of data type type, where number_of_elements is an integer value representing the amount of these. For example:

For example, int *tmp;
 tmp = new int [5]

Operator delete and delete[]

In most cases, memory allocated dynamically is only needed during specific periods of time within a program; once it is no longer needed, it can be freed so that the memory becomes available again for other requests of dynamic memory. For this purpose the operator delete is used.

Syntax: delete pointer;
 delete [] pointer;

The first statement releases the memory of a single element allocated using new, and the second one releases the memory allocated for arrays of elements using new and a size in brackets [].

The value passed as argument to delete shall be either a pointer to a memory block previously allocated with new, or a *null pointer* (in the case of a *null pointer*, delete produces no effect).

Dynamically the memory space is allocated for the linked list using new operator as follows:

```
Node *newNode;  
p = new Node;  
data(p) = num;  
link(p) = NULL;
```

Program: Creating a linked list and appending nodes into the linked list.

```
#include <iostream.h>  
class LinkList  
{  
    private:  
        struct Node  
        {  
            int data;  
            Node* link;  
        } *START;  
    public:  
        LinkList();  
        void Print(); //Prints the contents of linked list  
        void Append(int num); //Adds a new node at the end  
        void Count(); //Counts number of nodes in the linked  
list  
};  
LinkList::LinkList() //Constructor  
{  
    START = NULL;  
}  
  
// Prints the contents of linked list  
void LinkList::Print()  
{  
    if (START == NULL)  
    {  
        cout<< "Linked list is empty"<<endl;
```

```
        return;
    }
    //Traverse
    cout<<"Linked list contains";
    Node* p = START;
    while(p != NULL)
    {
        cout<<"    "<<p->data;;
        p = p->link ;
    }
}
// Adds a new node at the end of the linked list
void LinkList::Append(int num)
{
    Node *newNode;
    newNode = new Node;
    newNode->data = num;
    newNode->link = NULL;

    if(START == NULL)
    {
        //create first node
        START = newNode;
        cout<<endl<<num<<" is inserted as the first node"<<endl;
    }
    else
    {
        //Traverse
        Node *p = START;
        while(p->link != NULL)
        p = p->link;
        //add newnode to the end of the linked list
        p->link = newNode;
        cout<<endl<<num<<" is inserted"<<endl;
    }
}
// Counts number of nodes present in the linked list
void LinkList::Count()
{
    Node *p;
    int c = 0;
    //Traverse the entire Linked List
    for (p = START; p != NULL; p = p->link)
        c++;
}
```

```
        cout<<endl<<"No. of elements in the linked list= "<<c<<endl;
}
void main()
{
    LinkList* obj = new LinkList();

    obj->Print();
    obj->Append(100);
    obj->Print();
    obj->Count();

    obj->Append(200);
    obj->Print();
    obj->Count();

    obj->Append(300);
    obj->Print();
    obj->Count();
}
```

```
Linked list is empty
100 is inserted as the first node
Linked list contains 100
No. of elements in the linked list= 1
200 is inserted
Linked list contains 100 200
No. of elements in the linked list= 2
300 is inserted
Linked list contains 100 200 300
No. of elements in the linked list= 3
```

Traversing a linked list

Traversing is the process of accessing each node of the linked list exactly once to perform some operation.

To traverse a linked list, steps to be followed are given here.

1. To begin with, move to the first node.
2. Fetch the data from the node and perform the required operation depending on the data type.
3. Advance the pointer to the next node.
4. Step 2 and step 3 is repeated until all the nodes are visited.

Algorithm: This algorithm traverses the linked list. Here START contains the address of the first node. Another pointer p is temporarily used to visit all the nodes from the beginning to the end of the linked list.

Step 1:	p = START	[Initialize p to first node]
Step 2:	while p != NULL	
Step 3:	PROCESS data(p)	[Fetch the data]
Step 4:	p = link(p)	[Advance p to next node]
Step 5:	End of while	
Step 6:	Return	

Memory allocation to a linked list

Computer memory is a limited resource. Therefore some mechanism is required where the memory space of the deleted node becomes available for future use. Also, to insert a node to the linked list, the deleted node should be inserted into the linked list. The operating system of the computer maintains a special list called AVAIL list that contains only the unused deleted nodes.

AVAIL list is linked list that contains only the unused nodes. Whenever a node is deleted from the linked list, its memory is added to the AVAIL list and whenever a node is to be inserted into the linked list, a node is deleted from the AVAIL list and is inserted into the linked list. AVAIL list is also called as the free-storage list or free-pool.

Inserting a node into the linked list

Sometimes we need insert a node into the linked list. A node can be inserted into the linked list only if there are free nodes available in the AVAIL list. Otherwise, a node cannot be inserted. Accordingly there are three types of insertions.

1. Inserting a node at the beginning of the linked list
2. Inserting a node at the given position.
3. Inserting a node at the end of the linked list

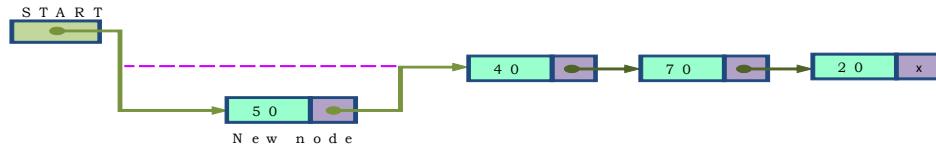
Inserting a node at beginning of the linked list

START is pointer that contains the memory address of the first node. To insert an ITEM into the linked list, the following procedure is used.

1. Create a new node.
2. Fill data into the data field of the new node.
3. Mark its next pointer field as NULL.
4. Attach this newly created node to START.
5. Make the new node as the STARTing node.

Algorithm (inst-beg): This algorithm inserts a node at the beginning of the linked list.

1. $p \leftarrow \text{new Node};$
2. $\text{data}(p) \leftarrow \text{num};$
3. $\text{link}(p) \leftarrow \text{START}$
4. $\text{START} \leftarrow p$
5. Return



Program: Inserting node at the beginning of the linked list.

```

#include <iostream.h>
#include<ctype.h>
class LinkList
{
private:
    struct Node
    {
        int data;
        Node* link;
    }*START;
public:
    LinkList();
    void Print();
    void Count();
    void insert(int num);
};
LinkList::LinkList()
{
    START = NULL;
}
void LinkList::Print()
{
    if (START == NULL)
    {
        cout<< "Linked list is empty"<<endl;
        return;
    }
    cout<<endl<<"Linked list contains";
  
```

```
Node* p = START;
while(p != NULL)
{
    cout<<“ <<p->data;;
    p = p->link ;
}
void LinkList::insert(int num)
{
    Node *newNode;
    newNode = new Node;
    newNode->data = num;
    newNode->link = START;
    START = newNode;
    cout<<num<<“ is inserted at the beginning”<<endl;
}
void LinkList::Count()
{
    Node *p;
    int c = 0;
        //Traverse the entire Linked List
    for (p = START; p != NULL; p = p->link)
        c++;
    cout<<endl<<“No. of elements in the linked list= “<<c<<endl;
}
void main()
{
    LinkList* obj = new LinkList;
    int item;
    char ch;

    cout<<“Do you want to insert a node at the beginning (Y/N): “;
    cin>>ch;
    while(toupper(ch) == ‘Y’)
    {
        cout<<“Enter the item to be inserted at the beginning: “;
        cin>>item;
        obj->insert(item);
        cout<<“Do you want to insert a node at the beginning (Y/N): “;
        cin>>ch;
    }
    obj->Print();
    obj->Count();
    cout<<“T H A N K Y O U”;
}
```

```

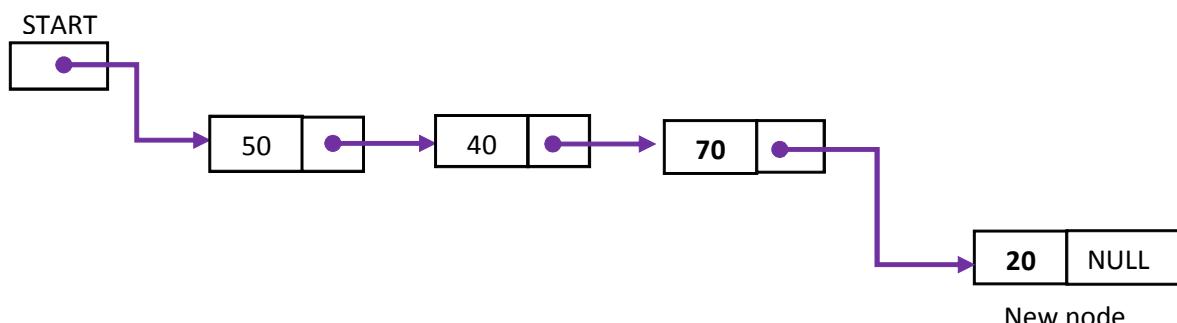
Do you want to insert a node at the beginning (Y/N): y
Enter the item to be inserted at the beginning: 111
111 is inserted at the beginning
Do you want to insert a node at the beginning (Y/N): y
Enter the item to be inserted at the beginning: 222
222 is inserted at the beginning
Do you want to insert a node at the beginning (Y/N): y
Enter the item to be inserted at the beginning: 333
333 is inserted at the beginning
Do you want to insert a node at the beginning (Y/N): n

Linked list contains 333 222 111
No. of elements in the linked list= 3
T H A N K Y O U

```

Inserting a node at the end of the linked list

We can insert a node at the end of the linked list. To insert at the end, we have to traverse the liked list to know the address of the last node.



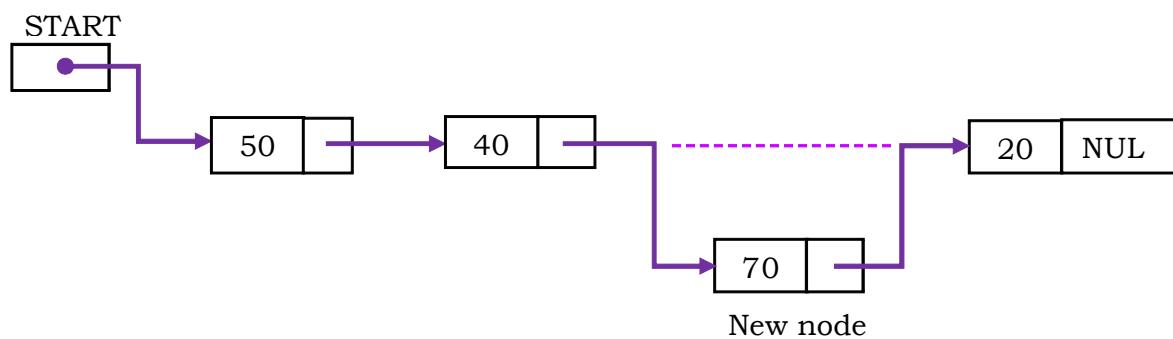
Algorithm (inst-end): Inserting a node at the end of the linked list.

1. START
2. [Identify the last node in the list]
 - a. $p \leftarrow \text{START}$
 - b. while $p \neq \text{NULL}$

$$p \leftarrow \text{next}(p)$$
 - while end
3. $N \leftarrow \text{new Node}$ [Create new node copy the address to the pointer N]
4. $\text{data}(N) \leftarrow \text{item}$
5. $\text{link}(N) \leftarrow \text{NULL}$
6. $\text{link}(p) \leftarrow N$
7. RETURN

Inserting a node at a given position

We can insert a node at a given position. The following procedure inserts a node at the given position.



Algorithm (INST-POS): This algorithm inserts item into the linked list at the given position pos.

```

1. START
2. [[Initialize] count ← 0
   p1 ← START
3. while (p1 != NULL)
   count ← count +1
   p1 ← link(p1)
   while end
4. a. if (pos = 1)
   call function INST-BEG()
   else
   b.   if (pos = count +1)
   call function INST-END()
   else
   c.   if( pos <= count)
         p1 ← START
         for (i = 1; i <= pos; i++ )
             p1 ← next(p1)
         for end
   d.   [create] p2 ← new node
       data(p2) ← item
       link(p2) ← link(p1)
       link(p1) ← p2
   else
   e.   PRINT “ Invalid position “
5. RETURN

```

Garbage collection:

If a node is deleted from the linked list or if a linked list is deleted, we require the space to be available for future use. The memory space of the deleted nodes is immediately reinserted into the free-storage list. The operating system of the computer periodically collects all the deleted space into the free-storage

list. This technique of collecting deleted space into free-storage list is called as garbage collection.

Deleting an item from the linked list:

The deletion operation is classified into three types:

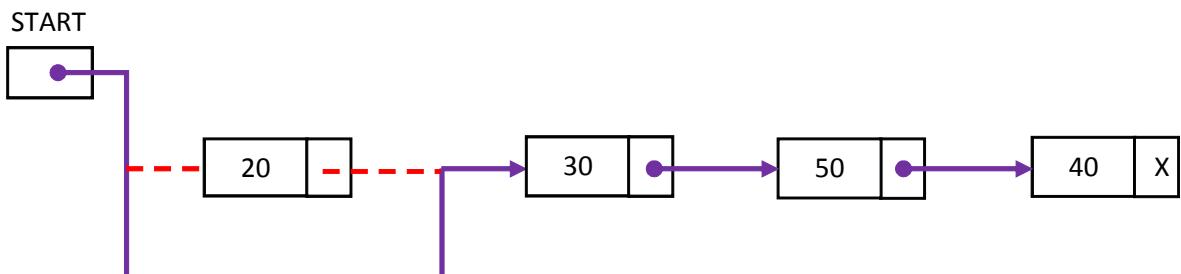
1. Deletion of the first node
2. Deletion of the last node
3. Deletion of the node at the given position

Underflow

If we try to delete a node from the linked list which is empty, the condition is called as underflow.

Deletion of the first node:

We can easily delete the first node of the linked list by changing the START to point to the second node of the linked list. If there is only one node in the liked list, the START can be stored with NULL.



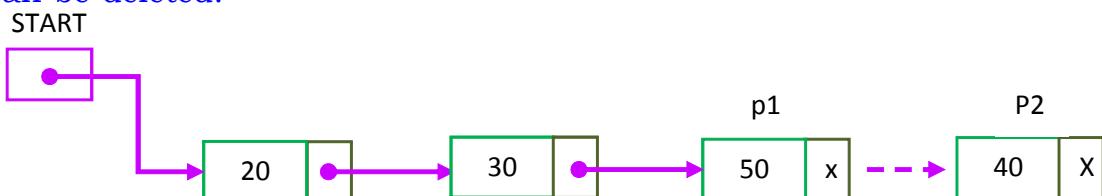
Algorithm (DELE-BEG): This algorithm first copy data in the first node to a variable and deletes the first node of the linked list.

```

Step1.      START
Step2.      p ← START
Step3.      PRINT data(p)
Step4.      START← link(p)
Step5.      free(p)
Step6.      RETURN
  
```

Deleting a node at the end (DELE-END)

To delete the last node, we should find location of the second last node. By assigning NULL to link field of the second last node, last node of the linked list can be deleted.



Algorithm (DELETE-END): This used two pointers p1 and p2. Pointer p1 is used to traverse the linked list and pointer p2 keeps the location of the previous node of p1.

1. START
2. p2 \leftarrow START
3. while (link(p2) != NULL)
 - p1 \leftarrow p2
 - p2 \leftarrow link(p2)
 while end
4. PRINT data(p2)
5. link(p1) \leftarrow NULL
6. free(p1)
7. STOP

Deleting a node at the given position

To delete a node at the given position, the following procedure is used.

1. Find location of the node to be deleted and the previous node.
2. Delete the node to be deleted by changing the location of the previous node of the node to be deleted.

Algorithm (DELETE-POS): Here p1 and p2 are the pointers. Pointer p2 is used to traverse the linked list. If pointer p2 has the location of the node to be deleted, p1 keeps the location of previous node of p2.

1. START
2. Count \leftarrow 0
3. p1 \leftarrow START
4. while(p1 != NULL)
 - count = count + 1
 - p1 \leftarrow link(p1)
 while end
4. if(pos = 1)

CALL DELE-BEG()
- else
 - if(pos = count)

CALL DELE-END()
 - else
 - if(pos < count)

p2 \leftarrow START

for i = 1 to pos

p1 \leftarrow p2

p2 \leftarrow link(p2)
- for end

```

PRINT data(p2)
link(p1) ← link(p2)
free(p2)
else    PRINT "Invalid position"
5. RETURN

```

Non-linear data structure

4.9.1 Introduction

A non-linear data structure is a data structure in which a data item is connected to several other data items, so that a given data item has the possibility to reach one-or-more data items.

The data items in a non-linear data structure represent hierarchical relationship. Each data item is called a node. Examples of non-linear data-structures are Graphs and Trees.

Pros

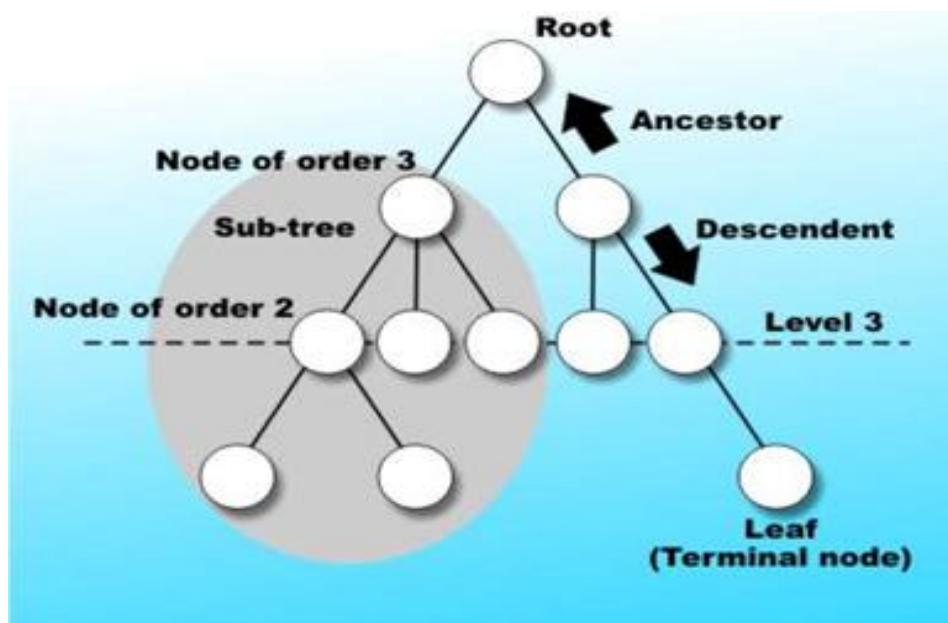
- Uses memory efficiently as contiguous memory is not required for allocating data items.
- The length of the data items is not necessary to be known prior to allocation.

Cons

- Overhead of the link to the next data item.

4.9.2 TREES

A tree is a data structure consisting of nodes organized as a hierarchy - see figure.



Terminology

A node is a structure which may contain a value, a condition, or represent a separate data structure (which could be a tree of its own). Each node in a tree has zero or more **child nodes**, which are below it in the tree (by convention, trees are drawn growing downwards). A node that has a child is called the **parent node** (or *ancestor node*, or superior). A node has at most one parent.

Nodes that do not have any children are called **leaf nodes**. They are also referred to as terminal nodes.

The **height** of a node is the length of the longest downward path to a leaf from that node. The height of the root is the height of the tree. The **depth** of a node is the length of the path to its root (i.e., its root path).

The topmost node in a tree is called the **root node**. Being the topmost node, the root node will not have parents. It is the node at which operations on the tree commonly begin. All other nodes can be reached from it by following **edges** or **links**.

An **internal node** or **inner node** is any node of a tree that has child nodes and is thus not a leaf node.

A **subtree** of a tree T is a tree consisting of a node in T and all of its descendants in T .

Binary trees

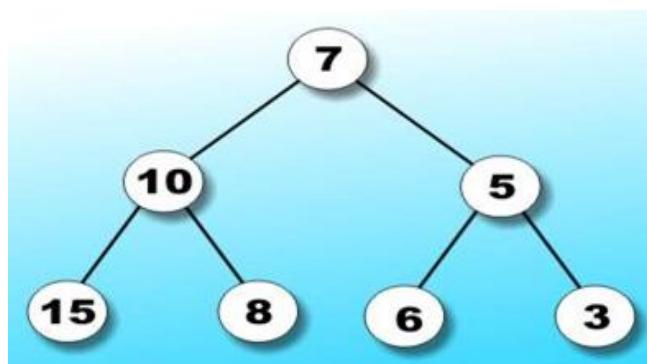
The simplest form of tree is a binary tree. A binary tree is a tree in which each node has at most two descendants - a node can have just one but it can't have more than two.

A binary tree consists of

- a *node* (called the **root node**) and
- left and right sub-trees.

Both the sub-trees are themselves binary trees.

The importance of a binary tree is that it can create a data structure that mimics a "yes/no" decision making process.



Root Node: Node at the “top” of a tree - the one from which all operations on the tree commence. The root node may not exist (a NULL tree with no nodes in it) or have 0, 1 or 2 children in a binary tree.

Leaf Node: Node at the “bottom” of a tree - farthest from the root. Leaf nodes have no children.

Complete Tree: Tree in which each leaf is at the same distance from the root. i.e. all the nodes have maximum two subtrees.

Height: Number of nodes which must be traversed from the root to reach a leaf of a tree.

4.9.3 GRAPHS

A graph is a set of *vertices* and *edges* which connect them. A *graph* is a collection of nodes called *vertices*, and the connections between them, called *edges*.

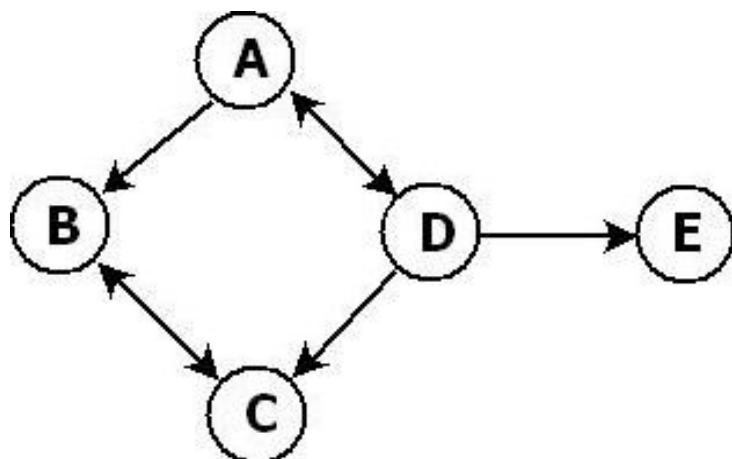
Undirected and directed graphs

When the edges in a graph have a direction, the graph is called a *directed graph* or *digraph*, and the edges are called *directed edges* or *arcs*.

Neighbors and adjacency

A vertex that is the end point of an edge is called a *neighbor* of the vertex, that is its starting-point. The first vertex is said to be *adjacent* to the second.

The following diagram shows a graph with 5 vertices and 7 edges. The edges between A and D and B and C are pairs that make a bidirectional connection, represented here by a double-headed arrow.



One mark questions

1. What are data structures?
2. Differentiate between one-dimensional and two-dimensional array.
3. Give any two examples for primitive data structures.
4. Mention any two examples for non-primitive data structures.
5. What are primitive data structures ?
6. What are non-primitive data structures ?
7. Name the data structure which is called LIFO list.
8. What is the other name of queue.
9. Define an array.
10. What are lists ?
11. What is meant by linear data structures ?
12. What are non-linear data structures ?
13. What is a stack ?
14. What is a queue ?
15. Name the data structure whose relationship between data elements is by means of links.
16. What is a linked list ?
17. Mention any one application of stack .
18. What do you mean by traversal in data structure.
19. Define searching in one-dimensional array.
20. What is meant by sorting in an array.
21. Mention the types of searching in the array.
22. What is a binary tree.
23. What do you mean by depth of a tree.
24. What are the operations that can be performed on stacks
25. What are the operations that can be performed on queues ?
26. Define the term PUSH and POP operation in stack.
27. What is FIFO list ?
28. What is LIFO list ?
29. Mention the different types of queues.

Two marks questions:

1. How are data structure classified ?
2. Justify the need for using arrays.
3. How are arrays classified ?
4. Mention the various operations performed on arrays .
5. How do you find the length of the array ?
6. Mention the types of linked lists.
7. What is a stack ? Mention the types of operations performed on the stacks.
8. What is a queue ? Mention the various operations performed on the queue.

Three marks questions:

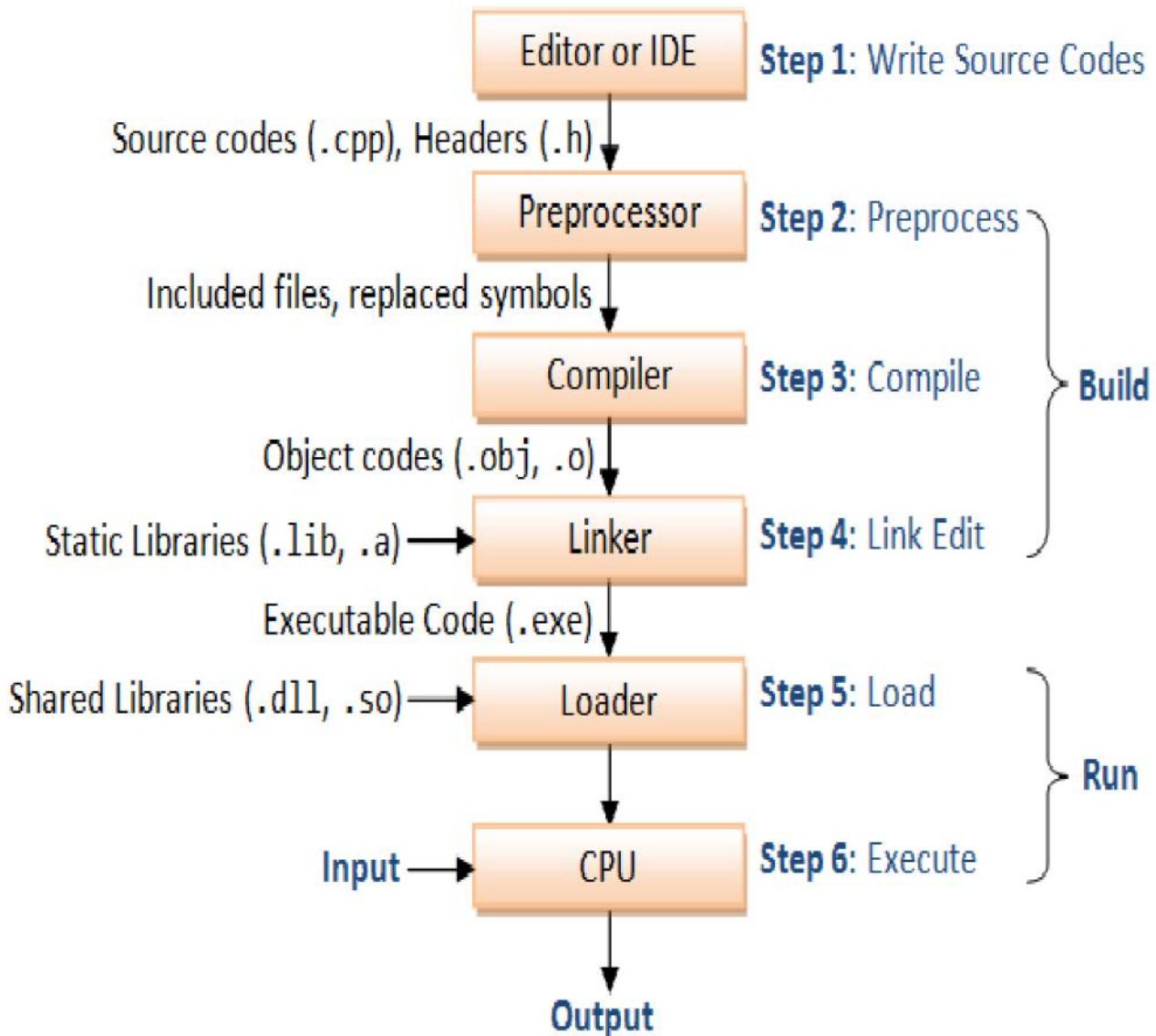
1. Mention the various operations performed on data structures.
2. Explain the memory representation of a one-dimensional array.
3. Explain the memory representation of a stack using one-dimensional array.
4. Explain the memory representation of queue using one-dimensional array.
5. Explain the memory representation of single linked list .
6. Define the following with respect to binary tree
a. root b. subtree c. depth
7. Write an algorithm for traversal in a linear array.
8. Give the memory representation of two-dimensional array.

Five marks questions:

1. Write an algorithm to insert an element in an array.
2. Write an algorithm to delete an element in an array.
3. Write an algorithm to search an element in an array using binary search.
4. Write an algorithm to sort an array using insertion sort.
5. Write an algorithm for push and pop operation in stack using array.
6. Write an algorithm to insert a data element at the rear end of the queue.
7. Write an algorithm to delete a data element from the front end of the queue.
8. Write an algorithm to insert a data element at the beginning of a linked list.
9. Write an algorithm to delete a data element at the end of a linked list.
10. Apply binary search for the following sequence of numbers.
10, 20, 30, 35, 40, 45, 50, 55, 60 search for item = 35

Chapter 5**Review of C++****Objectives:**

- To understand the basics of C++
- To understand different control structures.
- To understand structured data types arrays, string, functions and structures.



5.1 Review of C++ language

OPP OOP emphasizes on data. The ideology here is to unite both the data and the functions that operate on that data into a single unit called as “**object**”.

Therefore, an object is an identifiable entity with some characteristics and behavior.

OOP view any problem as object rather than as a procedure. For example, we can say ‘mobile’ is an object and its characteristics are color, weight, display, size etc. Its features include price, voice call, video call, memory size etc. Here **OOP considers the characteristics as data and features as functions.**

Another important concept with respect to OOP is the ‘**Class**’. A class serves as a plan or blueprint that specifies what data and what functions should be included in the objects of that class.

OOP characteristics

The characteristics of OOP are:

·Abstraction	Data encapsulation	Inheritance
Polymorphism	Polymorphism	Dynamic binding
Message Passing		.

Modularity

Modularity is a concept where given problem is divided into sub-problems and the sub-problems are further divided. Each sub-problem is solved by writing a subprogram. Each subprogram is called a module.

Abstraction

Abstraction is an act which represents the essential features of an entity without including explanations or any background details about it.

OOP implements abstraction using classes as a list of abstract attributes.

Data Encapsulation

The binding of data and functions into a single unit called as the class is known as **encapsulation**.

The concept of insulating the data from direct access by the program is called **data hiding**.

Inheritance

Inheritance is the process by which objects of one class acquires the properties of the objects of another class.

Polymorphism

Polymorphism means taking more than one form. **Polymorphism** is the ability for a message to be processed in more than one form.

The process of making an operator to exhibit different behaviors in different instances is known as **operator overloading**.

Using a single function name to perform different types of tasks is known as **function overloading**.

Polymorphism allows objects to have different internal structures to share the same external interface. It supports to implement inheritance to a great extent.

Dynamic Binding

Binding means linking of a function call to the code to be executed when the function is called.

Dynamic Binding means that the code associated with a function call is not known until the time of the call at run-time. It is associated with polymorphism and inheritance.

Message Passing

The objects of a class can communicate with each other. The objects communicate with each other by sending and receiving messages. A message means a request for the execution of a function for an object. Therefore, a message invokes a function in the form of an object to generate the desired output.

For example, consider the message: stack1.push();

Here, we are invoking a function push of an object stack1. stack1 is an object of the class stack.

5.2 Fundamentals of C++

Bjarne Stroustrup developed C++ at AT & T Bell Laboratories in Murray Hill, New Jersey. He developed this language in early 1980's and was very much concerned in making C++ more efficient.

C++ character set

The following are the character set in C++.

Alphabets	A, B, , Za, b,, z
Numerals	0, 1, 2,, 9
Special characters:	+ - * % / \ . < > , = _ @ ! ^ & ~ { } [] () etc

Tokens

A token is a smallest individual unit in a program or a lexical unit.

The most fundamental elements of a C++ program are "tokens". These elements help us to construct statements, definitions, declarations, and so on, which in turn helps us to construct complete programs. C++ has following tokens:

Identifiers	Keywords	Variables	Constants
Punctuators	Operators		

Identifiers

An identifier is a name given to the programming elements such as variables, arrays, functions etc. It can contain letter, digits and underscore. C++ is case sensitive and henceforth it treats uppercase and lowercase characters differently.

Exampe:	Student STUDENT	_amount AMOUNT	marks1 RANK5	total_score _Ad12
---------	--------------------	-------------------	-----------------	----------------------

Keywords

Keyword is a predefined word that gives special meaning to the compiler. They are reserved words which can be used for their intended purpose and should not be used as normal identifier. Some of the keywords are given below:

and	asm	auto	bool	break	case
catch	char				

Constants or Literals

Constant is an identifier whose value is fixed and does not change during the execution of the program.

Integer constants

Integer constants are numbers that has no fractional pars or exponent. It refers to the whole numbers. Integers always begin with a digit or + or -.

We can specify integer constants in decimal, octal, or hexadecimal form.

Unsigned constants:

To specify an unsigned type, use either the **u** or **U** suffix. To specify a long type, use either the **l** or **L** suffix.

Example 7.4: 328u 0x7FFFFFFL 0776745ul;

Floating point constants

Floating-point constants are also called as **real constants**. These values contain decimal points (.) and can contain exponents. They are used to represent values that will have a fractional part and can be represented in two forms - fractional form and exponent form.

In the fractional form, the fractional number contains integer part and fractional part. A dot (.) is used to separate the integer part and fractional part.

Example: 65.05 125.5 -125.75

In the exponential form, the fractional number contains constants a mantissa and exponent. Mantissa contains the value of the number and the exponent contains the magnitude of the number. The **exponent**, if present, specifies the magnitude of the number as a **power of 10**.

Example: 7.6: 23.46e0 // $23.46 \times 10^0 = 23.46 \times 1 = 23.46$
 23.46e1 // $23.46 \times 10^1 = 23.46 \times 10 = 234.6$

Character constants

Character constants are specified as single character enclosed in pair of single quotation marks. Single character constants are internally represented as ASCII codes.

For example: ‘A’ ‘p’ ‘5’

There is another class of characters which cannot be displayed on the screen (non-graphic characters) but they have special meaning and are used for special purpose. Such character constants are called as **escape sequences**.

For example: \ ' " \\ \0 \a

String constants

A string zero or more characters enclosed by double quotation marks (“”) is called a string or string constant.

String constants are treated as an array of char. By default, the compiler adds a special character called the ‘**null character**’ ('\0') at the end of the string to mark the end of the string.

For example: “Empress College” “Tumkur” “C++ Programming\0”

Punctuators

Punctuators are symbols in C++ and have syntactic and semantic meaning to the compiler. But do not by themselves specify an operation that yields a value. Some punctuators can be either alone or in combination. The following characters are considered as punctuators:

For example: ! % ^ & * () -

C++ Operators

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations.

The operators can be either ‘**unary**’ or ‘**binary**’ or even ‘**ternary**’ operators.

Unary operators

Unary operations have only one operand.

For example: ! & ~ * ++ — + -

Binary operators

The binary operators are those operators that operate on two operands. These operators can be arithmetic operators, relational, logical operators, bitwise, assignment operators.

Arithmetic Operators

The arithmetic operators are: + - * / %

These operators are used in arithmetic expressions

Relational Operators:

The relational operators are: == != > < >= <=

These operators are used in relational operations. The relational operations always give 0 (or False) or 1 (or True).

Logical Operators

The logical operators supported by C++ are: `&&` `||` and `!`

Logical operators are used in logical expressions. Such expressions always give 0 (or False) or 1 (or True).

Bitwise Operators

Bitwise operator works on bits. The bitwise operators are: `&` `|` `^`
`~` `<<` `>>`

The Bitwise operators supported by C++ are listed in the following table 7.9.

Shorthand operators

We combine the assignment operators with arithmetic operators and bitwise operators. Such operators are called as shorthand operators

The shorthand operators are: `+=` `-=` `*=` `/=` `%=`
`&=` `|=` `^=`

Assignment Operator

It is an operator used to assign a value to a variable.

The syntax is variable = value or expression;

Example: `n = 10;` `sum = n + 35;`

Special Operators

Some special operators are there in C++. They are

`sizeof()` `comma(,)` `dot(.)` `pointer(*)`

Ternary operators

The ternary operators are those operators that operate on three or more operands. Ternary operator is also called as conditional operator.

? is the ternary operator. This operator can be used as an alternative to if-else statement.

Precedence of operators or Hierarchy of operators

If an expression contains multiple operators, the order in which operations are to be carried out is called hierarchy.

Example : `x = 7 + 3 * 2;`

Here x is assigned 13, not 20 because operator * has higher precedence than +. First multiply 3 and 2 and then adds into 7 and assign it to the variable x.

Type Conversion

Converting an expression of a given type into another type is known as type conversion.

Type conversions are of two types, they are

- Ø Implicit conversion
- Ø Explicit conversion

Implicit conversion

Implicit conversions do not require any type conversion operator. They are automatically performed when a value is copied to a compatible type. The C++ compiler will implicitly does conversion.

Explicit conversion will be performed by the user. The user can convert an expression from one type to another type. Such conversions are called as explicit conversion or type casting.

5.3 Structure of C++ Program

Include files
Class declarations
Member function declaration
main() function

Include files: This section is used to include all the preprocessor directives that are necessary for the program being written.

Class declaration: A class is a blueprint for the objects. It describes the data and functions that the object is going to use.

Member function declarations: This section defines or declares the user-defined functions that other functions or the main() function may use.

main() function: This is also a function that integrates all the other functions of the program. It contains the body of the function. The body should be enclosed within curled braces {and}.

The body contains two parts: Local declarations and executable statements. The local declaration refer to the declaration of all those variables that are used within the main() function. The executable statements are the statements that perform the required operations to solve the problem.

5.4 Library functions

C++ provides many built-in functions that save the programming time. They include mathematical functions, character functions, string functions, console input-output functions and some general standard library functions. Some of them are given below and also discussed in detail in Functions chapter.

Character Functions

All the character functions require **ctype.h** header file. The following table lists the function.

Some functions of character manipulation are given below:

isalpha()	isdigit()	isupper()	islower()
tolower()	toupper()	toascii()	isspace()
			ispunct()

String Functions

The string functions are present in the **string.h** header file. Some string functions are given below:

Some of the functions are: **strlen()** **strcat()** **strcpy()** **strrev()**
strupr() **strlwr()** **strcmp()** **strcmpi()**

Console I/O functions

The following are the list of functions in **stdio.h** are:

getchar() **putchar()** **gets()** **puts()**

5.5 Data types

A variable is an identifier whose value is allowed to change during the execution of the program. A variable actually represent the name of the memory location.

Declaration of a variable

The syntax for declaring a variable is datatype variable_name;

Example: int n = 50;

Initializing a variable

The syntax to initialize a variable is: datatype variable_name = value;

Example : Let b be a variable declared of the type int. Then, int b = 100;

C++ compiler allows us to declare a variable at run time. **This is dynamic initialization.** They can be initialized anywhere in the program before they are used.

Example int a, b;

....

```
int temp = a;
a = b;
b = a;
```

5.5 Data types: C++ support the following data types.

Data types classification

There are two types of data types.

- Ø Simple or fundamental data types
- Ø Complex or Derived data types

The simple or fundamental data types are the primary data types which are not composed of any other data types.

The simple data types/fundamental data types include int, char, float double and void.

Modifiers

The data type modifiers are listed here: signed, unsigned, long and short.

Example: unsigned int b;

Derived data types

These data types are constructed using simple or fundamental data types. They include arrays, functions, pointers and references.

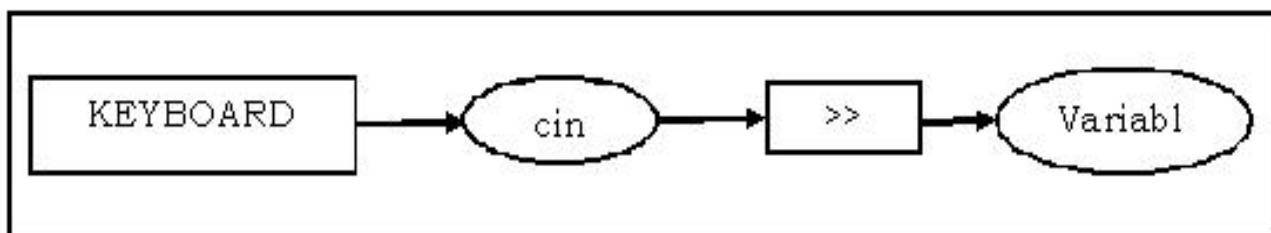
User defined data types

These data types are also constructed using simple or fundamental data types. Some user defined data types include structure, union, class and enumerated.

5.6 Input and output Operators

Input and output operators are used to perform input and output operations.

Input Operator “>>”

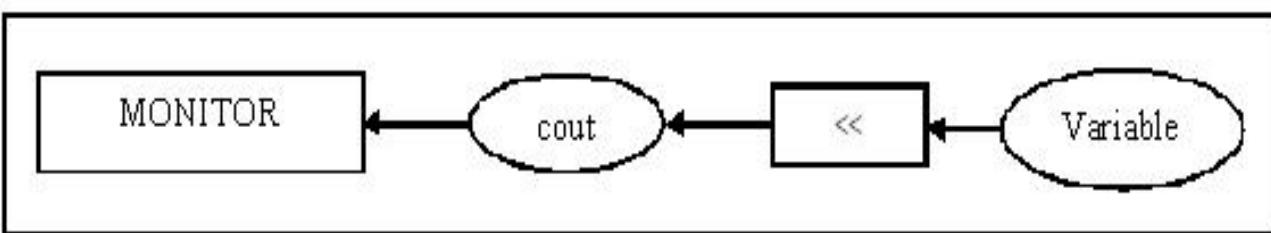


The standard input device is usually the keyboard. Input in C++ is done by using **stream extraction operator (>>)** on the cin stream. The operator must be followed by the variable that will store the data that is going to be extracted from the stream.

Example : int age;
 cin>>age;

Output Operator “<<”

The standard output device is the screen (monitor) and outputting in C++ is done by using the object followed by the **stream insertion operator** which is written as << . cout stands for Console output.



Example: cout<< " Let us learn C++"; // prints Let us learn C++
//on the screen.

Cascading of I/O operators:

C++ supports the use of stream extraction (<<) and stream insertion (>>) operators many times in a single input (cin) and output (cout) statements. If a program requires more than one input variable then it is possible to input these variables in a single cin statement using multiple stream extraction operators. Similarly, when we want to output more than one result then this can be done using a single cout statement with multiple stream insertion operators. This is called as **cascading of input output operators**.

Example : cout<<" enter the value for x";
 cin>> x;
 cout<<" enter the value for y";
 cin>>y;

Instead of using cin statement twice, we can use a single cin statement and input the values for the two variables x and y using multiple stream extraction operator as shown below.

```
cout<<" enter the value for x and y";  
      cin>>x>>y;
```

Similarly, we can even output multiple results in a single cout statement using cascading of stream insertion operator as shown below.

```
cout<<" the sum of" <<x<<" and "<<y<<"="<<x+y;
```

5.7 Control Statements

C++ provides us **control structures**, statements that can alter the flow of a sequence of instructions. .

Compound statements

Compound statements or block is a group of statements which are separated by semicolons (;) and grouped together in a block enclosed in braces { and } is called a compound statement.

Example: {
 temp = a;
 a = b;
 b = temp;
 }

Types of control statements

- C++ supports two basic control statements.

- Selection statements
- Iteration statements

Selection statements This statement allows us to select a statement or set of statements for execution based on some condition.

The different selection statements are:

- i. if statement
- ii. if-else statement
- iii. nested statement
- iv. switch statement

if statement

This is the simplest form of **if** statement. This statement is also called as **one-way branching**. This statement is used to decide whether a statement or a set of statements should be executed or not. The decision is based on a condition which can be evaluated to TRUE or FALSE.

Example if (n == 100) cout<< " n is 100 ";

The if-else statement

This statement is also called as **two-way branching**. It is used when there are alternative statements need to be executed based on the condition. It executes some set of statements when the given condition is TRUE and if the condition is FALSE then other set of statements to be executed.

Nested-if statement

An if statement may contain another if statement within it. Such an if statement is called as nested if statement.

There are two forms of nested if statements:

Format I: if-else-if statement

This structure is also called as else-if ladder. This structure will be used to verify a range of values. This statement allows a choice to be made between different possible alternatives. A choice must be made between more than two possibilities.

Format II:

This structure contains an if-else statement within another if-else statement.

Switch statement

C++ has a built-in **multiple-branch selection** statement, switch. This successively tests the value of an expression against a list of integer or character constants. When a match is found, the statements associated with that code is executed.

Iteration statements or loops

Iteration statements are also called as loops. Loop is a statement that allows repeated execution of a set of instructions until certain condition is satisfied. This condition may be predefined or post-defined. Loops have a purpose to repeat a

statement or set of statements a certain number of times or some condition is fulfilled. We use three types of looping structures in C++.

- Ø while loop
- Ø do- while loop
- Ø for loop

while loop

This looping structure is also called as **pre-tested looping structure**. This statement repeats the execution of a set of statements while the condition is TRUE.

Example: c = 1;
 while(c <= 10)
 cout<<setw(4)<<c;

do-while loop

This looping structure is also called as **post-tested looping structure**. Unlike while loop that test the loop condition at the beginning, the do-while loop checks the condition after the execution of the statements in it. This means that a do-while loop always executes at least once. Its functionality is exactly the same as the while loop, except that the condition in the do while loop is evaluated after the execution of statement, instead of before.

Example: c = 1;
 do
 {
 cout<<setw(4)<<c;
 } while(c <= 10);

The for loop

This statement is called as the **fixed execution looping statement**. It is normally used when we know in advance exactly how many times a set of statements should be repeatedly executed again and again. It provides initialization, loop-end-condition and increment/decrement process statements in a single line. When one is aware of fixed number of iterations, then this looping structure is best suited.

Jump statements or Transfer of control from within loop

Transfer of control within looping are used to

- Ø Terminate the execution of loop
- Ø Exiting a loop
- Ø Half way through to skip the loop.

All these can be done by using **break**, **exit**, and **continue** statements.

5.8 ARRAYS

Array fundamentals

An array is collection of objects and all the objects have the same name. Each object is called an element. The elements are numbered as 0, 1, 2,..., n-1.

These numbers are called as indices or subscripts. These numbers are used to locate the positions of elements within the array.

The method of numbering the i^{th} element with index $i-1$ is called as **zero-based indexing**. That is, the element is always same as the number of “steps” from the initial element $a[0]$ to that element. For example, the element $a[3]$ is 3 steps from the element $a[0]$.

Types of arrays

There are three types of arrays.

- i. One-dimensional array
- ii. Two-dimensional array
- iii. Multi-dimensional array

One-dimensional array

It is an array in which each element is accessed by using **only one subscript**. The only one subscript represents the position of the element in the array.

Two-dimensional array

It is an array in which each element is accessed using 2-subscripts. The subscripts represent the position of element in the array.

Multi-dimensional array

A multidimensional array is an array of n-dimensions. In other words, an array of arrays is called a multidimensional array. A one-dimensional array of one-dimensional arrays is called a two-dimensional array; a one-dimensional array to two-dimensional arrays is called a three-dimensional array and so on.

One-dimensional array:

It is an array in which each element is accessed by using **only one subscript**. The only one subscript represents the position of the element in the array.

Declaration of one-dimensional array

Syntax datatype array-name[size];

Example : int marks[50];

Initialization of one-dimensional arrays

You can give values to each array element when the array is first defined.

Example : int a[5] = {9, -5, 6, 2, 8};

In the above example, value 9 is stored in $a[0]$, value -5 is stored in $a[1]$, value 6 is store in $a[2]$, value 2 is stored in $a[3]$ and value 8 is store in $a[4]$.

Memory representation of one-dimensional arrays:

The elements of one-dimensional arrays are stored in contiguous memory locations.

Example : Consider the declaration, `char a[5];`

The element `a[0]` is allocated at a particular memory location, the element `a[1]` is allocated at next memory location and so forth. Since the array is of the type `char`, each element requires 1-byte.

Two-dimensional arrays

It is an array in which each element is accessed using 2-subscripts. The subscripts represent the position of the elements in the array.

The elements of two dimensional arrays are represented as rows and columns. To identify any element, we should know its row-number and column-number.

Declaration of two-dimensional array:

Syntax `datatype array-name[row-size][column-size];`

Example : `int a[2][3];`

Initialization of two-dimensional arrays

Example : `int a[2][3] = {1, 2, 3, 4, 5, 6};`

`a` is a two dimensional array which contains 2 rows and 3 columns and these assignments would be

$$\begin{array}{lll} a[0][0] = 1 & a[0][1] = 2 & a[0][2] = 3 \\ a[1][0] = 4 & a[1][1] = 5 & a[1][2] = 6 \end{array}$$

If the values are missing in an initialize, they are automatically set to 0.

Example : `int b[2][3] = {`

`{1, 2},`

`{3}`

`};`

will initialize the first two elements to the first row, the next element to the second row. The remaining elements are automatically set 0.

$$\begin{array}{lll} b[0][0] = 1 & b[0][1] = 2 & b[0][2] = 0 \\ b[1][0] = 3 & b[1][1] = 0 & b[1][2] = 0 \end{array}$$

Multi-dimensional array

A multidimensional array is an array of n-dimensions. In other words, an array of arrays is called a multidimensional array. A one-dimensional array of one-dimensional arrays is called a two-dimensional array; a one-dimensional array to two-dimensional arrays is called a three-dimensional array and so on.

5.9 FUNCTIONS

If the programs are complex and lengthy, they can be modularized into subprograms. The subprograms are called as functions. The subprograms can be developed independently, compiled and tested. They can be reused in other programs also.

A function is a named group of statements developed to solve a sub-problem and returns always a value to other functions when it is called.

Types of functions

There are two types of functions:

- i. Library functions
- ii. User-defined functions.

i. Library functions

A standard library is a collection of pre-defined functions and other programming elements which are accessed through header files.

Header files are the files containing standard functions that our programs may use. This chapter contains the information about some header files of C++ standard library and some functions of it. The header files should be written within angled brackets and its functions are included into our programs by #include directive.

User-defined functions

We can create our own functions or sub-programs to solve our problem. Such functions are normally referred to as **user defined functions**.

A user-defined function is a complete and independent program, which can be used (or invoked) by the main program or by the other sub-programs. The user-defined functions are written to perform definite calculations, after performing their task they send back the result to the calling program or sub-program.

Different header files

As said earlier, header files are the files containing standard functions that our programs may use. C++ contains many header files and is listed below.

stdio.h

This header file contains functions and macros to perform standard I/O operations. When we include the header file iostream.h, the header file stdio.h is automatically included into our program.

string.h

This header file declares functions to manipulate strings.

stdlib.h

This header file is used to declare conversion routines, search/sort routines and other miscellaneous things.

iostream.h

This header file contains C++ streams and i/o routines.

iomanip.h

This header file contains functions and macros for I/O manipulators for creating parameterized manipulations.

math.h

This header file declares prototypes for the mathematical functions and error handlers. The functions are used to perform mathematical calculations.

Mathematical library functions

C++ provides many mathematical functions. These functions can be used in mathematical expressions and statements. The various functions are ceil(), exp(), fabs(), floor(), log(), pow() etc.

Character and string functions

A character is any single character enclosed within single quotes. Some functions accept a character as argument. The argument is processed as an int by using its ASCII code. To use these functions, the header file **ctype.h** should be included.

Inputting single character

We can input a character using the function `get()`.

```
char ch; char ch;  
cin = get(ch); OR ch = cin.get();
```

Outputting single character

```
cout.put(ch);
```

`put()` function is used to display single character.

The general form is

Example : `char ch; ch = cin.get(); cout.put(ch);`

String functions

A string is sequence of characters enclosed within double quotes. Strings are manipulated as one-dimensional array of characters and terminated by null ('\0') character. C++ provides many functions to manipulate strings. To use these functions, the header file **string.h** should be included.

```
char string-name[size];
```

Declaring a string variable

The general form to declare a string is:

Ø `string-name` is the name of the string variable

- Ø Size is the number of characters in the string. The size helps the compiler to allocate required number of memory locations to store the string.

Example : char st[50];

Initializing a string

Like other variables, strings can also be initialized when they are declared.

Example : char s[10] = "Karnataka";

There are only 9 characters in the string. The null character ('\0') is automatically appended to the end of the string.

Inputting a string

C++ provides the function getline() to read a string.

```
cin.getline(string, size);
```

Example cin.getline(st, 25);

Outputting a string

C++ provides the function write() to output a string.

```
cout.write(string, size);
```

Example : cout.write(st, 25);

Some string manipulation functions are given below:

strlen() function

This function returns the length of the string. i.e., the number of characters present in the string, excluding the null character.

The general form is variable = strlen(string);

Example 12.9: l = strlen("Empress"); Returns 7.

strcat() function

This function is used to concatenate two strings. The process of combining two strings to form a string is called as concatenation.

strcpy() function

A string cannot be copied to another string by using assignment statement. The function strcpy() is used to copy a string into another string.

strcmp() function

This function is used to alphabetically compare a string with another string. This function is **case-sensitive**. i.e., it treats the uppercase letters and lowercase letters as different.

strcmpi() function

This function is used to alphabetically compare a string with another string. This function is **not case-sensitive**. i.e., it treats the uppercase letters and lowercase letters as same.

strrev() function

This function is used to reverse the characters in a string.

5.10 USER DEFINED FUNCTIONS

Definition

User-defined function is a function defined by the user to solve his/her problem. Such a function can be called (or invoked) from anywhere and any number of times in the program.

Function definition or structure of user-defined function

Function-header

Any user-defined function has the following structure.

```
return-type-specifier function-name(argument-list with declaration)
{
    Local-variable-declarations;
    Executable-statement-1;
    Body of the function
    Executable-statement-2;
    .....
    Executable-statement-n;
    return(expression);
}
```

Ø **Return-type-specifier** is the data type of the value return by the function to another function when it is called. The return-type-specifier can be char, int, float or void. The data type void is used when the function return no value to the calling function.

Ø **Function-name** is the name of the function. It is an identifier to identify the particular function in a program.

Ø **Argument-list with declaration** is the list of arguments or parameters or variables with their declaration. Each argument should be declared separately. Each declaration should be separated by comma. The list should be enclosed within parenthesis.

Ø The complete line is called the function header. Note that there is no semicolon at the end.

Ø **Local-variable declaration** is the declaration of the variables that are used within the function. Since these variables are local to the function, these variables are called as local variables.

Ø **Executable-statements** are the statements that perform the necessary operations to solve the problem. If the function is returning a value, a return

statement should be included. Otherwise, return statement is not necessary.

Ø Local declaration and executable statements are together called as **body of the function**. The body of the function should be enclosed within the curled braces.

Calling a function

`variable = function-name(argument-list);`

OR

`variable = function-name();`

A function can be called by specifying its name, followed by list of arguments enclosed within the parenthesis. The arguments should be separated by commas. If the function does not pass any arguments, then an empty pair of parenthesis should follow the name of the function.

The function call should be a simple expression such as an assignment statement or it may be part of a complex expression.

Example : `big = biggest(a, b, c);`

main() function

In C++, the `main()` function returns a value of type **int** to the operating system. If the program runs successfully, 0 is returned. Otherwise, a non-zero value is returned to the operating system, indicating that the program contains errors. If the `main()` function is not returning a value, the datatype **void** can be used as return-type-specifier.

The general form of `main()` function is:

<pre>int main() { Executable-statements; return 0; }</pre>	OR	<pre>void main() { Executable-statements; }</pre>
--	-----------	---

Returning a value

When a function is called, the statements in the called function are executed. After executing the statements, the function returns a value to the calling function. The return statement is used to return a value. A function can return only one value or none to the calling function, for every function call.

The general form of return statement is:

`return(expression);` OR `return 0;`

Function prototypes

Like all variables are declared before they are used in the statements, the function should also be declared. The function prototype is a declaration of the

function in the calling function.

The general form of function prototype is

return-type-specifier function-name(type arg1, type arg2,);

OR

return-type-specifier function-name(type , type ,);

Example : float volume(int x, float y, float z);
 Or float volume(int, float, float);

Types of arguments

A variable in a function header is called an argument. The arguments are used to pass information from the calling function to the called function.

Actual arguments

The function call statement contains name of the function and list of arguments to be passed. These arguments or parameters are called as actual arguments. The arguments can be constants, variables or expressions. Actual arguments have values stored in them before the function call hence the name actual.

Example : In the function call g = gcd(a, b);

Formal arguments

The function header contains return-type-specifier, function name and list of arguments with their declaration. These arguments are called as formal arguments or dummy arguments. Formal arguments get their values from the actual arguments.

Example: In the function header int gcd(int x, int y)
 x and y are the formal arguments.

Local variables

The variables declared inside function or block is said belong to that block. These variables are called as Local variables. Values of local variables are accessible only in that block. The function's formal arguments are also considered as local variables.

Global variables

The variables declared outside the function are called as global variables. These variables are referred by the same data type and same name throughout the program in both the calling function and called function. Whenever if some variables are to be treated as same value in both main() and in other functions, it is advisable to use global variables.

The availability of the values of global variables starts from the point of definition to the rest of the program.

Types of Functions :

There are 5 types of functions.

- i. Functions with no arguments and no return values
- ii. Functions with arguments and with no return values
- iii. Functions with no arguments and with return values
- iv. Functions with arguments and with return values
- v. Recursive functions

Functions with no arguments and with no return values

In this method, the function simply performs an independent task. The function does not receive or send any arguments.

Example :

```
void natural()
{
    for(int i=1; i <= 10; i++)
        cout<<setw(4)<<i;
}
```

Functions with arguments and with no return values

In this method, the function receives some arguments and does not return any value.

Example :

```
void average(int x, int y, int z)
{
    float sum, avg;
    sum = a + b + c;
    avg = sum/3.0;
    cout<<"Average = "avg<<endl;
}
```

Functions with no arguments and with return values

In this method, the function receives no arguments but return a value.

Example :

```
int greatest()
{
    if(a>b)
        return(a);
    return(b);
}
```

Functions with arguments and with return values

In this method, the function receives some arguments and returns a value.

Example :

```
float interest(float p, float t, float r)
{
    si = (p * t * r)/100;
    return(si);
}
```

Recursive functions

Recursive function is a function that calls itself. The process of calling a function by itself is called as recursion.

Recursive functions must have one or more terminating conditions to terminate recursion. Otherwise, recursion will become infinite.

Passing default arguments to functions

In C++, to call a function we need not pass all the values for the arguments to a function from the calling function. It allows us to assign default values to the formal arguments.

Example : Consider the prototype

```
float interest (float amt, int time, float rate = 0.15);
```

0.15 is the default value provided to the argument rate.

The function call statement can be as follows:

```
si = interest( 5000, 5 ); // third argument is missing
```

Default values should be assigned only in the function prototype. It should not be repeated in the function definition.

Passing constant arguments

In C++, we can declare some arguments as constants. The compiler cannot modify the arguments marked as constants.

Example :

```
int strlen(const char *p);
int total( constint x, const int y);
```

Pass by value or Call by value

A function can be called by passing arguments from the calling function into the called function. Thus the data is transferred through argument list.

Pass by reference or call by reference

We can pass parameters to the function by using reference variables. When we pass arguments by reference, the formal arguments in the called function become the **aliases** to the actual arguments of the calling function. i.e., the called function is actually uses the original data with a different name.

Passing arrays to functions

To pass an array to a function, we just pass the name of the array to the function. i.e., we are referring the address of the first element of the array. Using this address, the function can access all the elements of the array.

Passing structures to functions

We can pass structures to functions as we pass other arguments. Structures are passed to functions in pass-by-value method. i.e., the function works with copy of the structures. The function can also return a structure after processing it.

Whenever we pass the address of the structure to the function, we should include the address-of (&) operator.

5.11 Structures

A structure is a collection of simple variables. The variables in a structure can be of same or different types: Some can be int, some can be float and so on. The data items in a structure are called the **members** of the structure.

Defining a structure

The process of defining a structure is equivalent to defining your own data type.

```
struct structure-name
{
    datatype member-name-1;
    datatype member-name-2;
    .....
    datatype member-name-n;
};
```

Example: A structure definition to hold employee information.

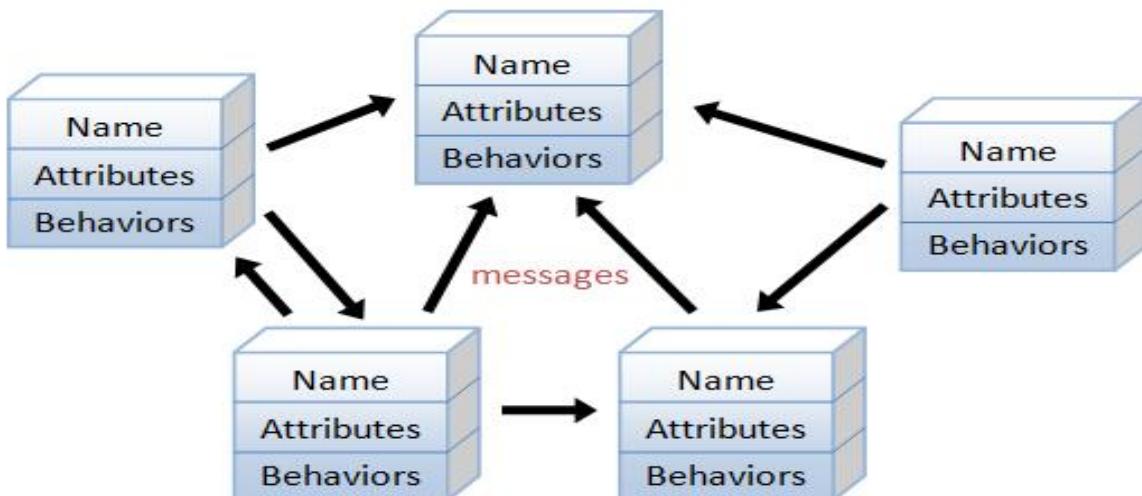
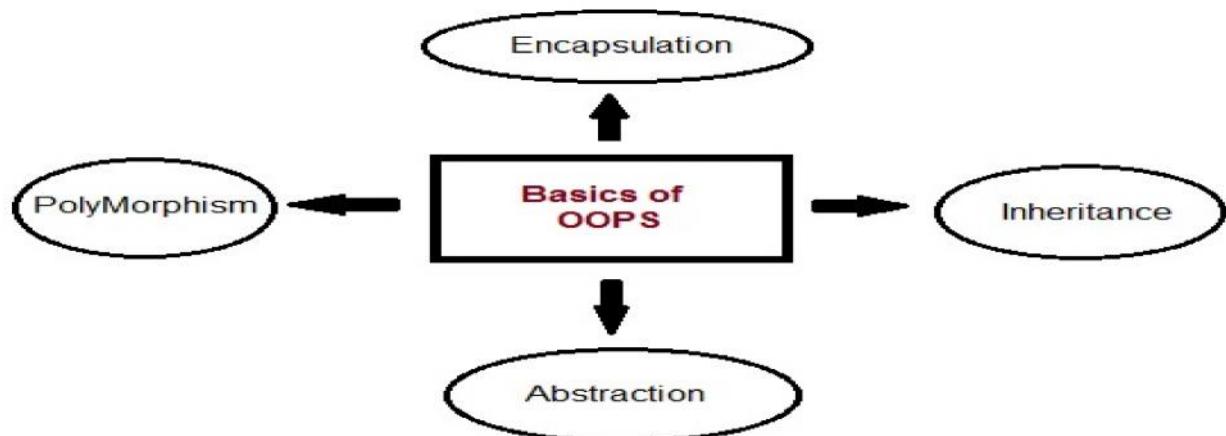
```
struct employee
{
    int idno;
    char name[15];
    char designation[10];
    float salary;
};
```

Chapter 6

BASIC CONCEPTS OF OOP

Objectives:

- Provides an overview of object oriented programming
- To understand concept of objects, classes and other related terminologies
- To highlight advantages of OOP
- To highlight limitations of OOP
- To identify the application areas



An object-oriented program consists of many well-encapsulated objects and interacting with each other by sending messages

6.1 Introduction

Object oriented programming is the principle of design and development of programs using modular approach. Object oriented programming approach provides advantages in creation and development of software for real life applications. The advantage is that small modules of program can be developed in shorter span of time and these modules can be shared by a number of applications. The basic element of object oriented programming is the data. The programs are built by combining data and functions that operate on the data. In this chapter we learn about the basic concepts, advantages and terminologies used in Object oriented programming. Some of the object oriented programming languages are C ++, Java, C # and so on.

The object oriented programming methods use data as the main element in the program. The data is tied to the function that operates on the data and the other functions cannot modify the data tied to a given function. Thus in object oriented programming, a problem is decomposed into a number of components called objects. An object is a collection of set of data known as member data and the functions that operate on these data a known as member functions or Methods. The member data are encapsulated in an object and then can be accessed or modified only by the member functions. An object can be accessed only if permitted by other member functions. Various objects of a program can interact with each other by sending messages.

Object oriented programming methods modularize a program by creating memory area for data and member functions (methods) together as a single entity. All objects are created according to the specifications of the entity defined.

Object is the basic unit of OOP. To design OO Model, first a set of classes are defined. A class is a Template from which objects are created. The Template of a class specifies the data , member functions and their attributes.

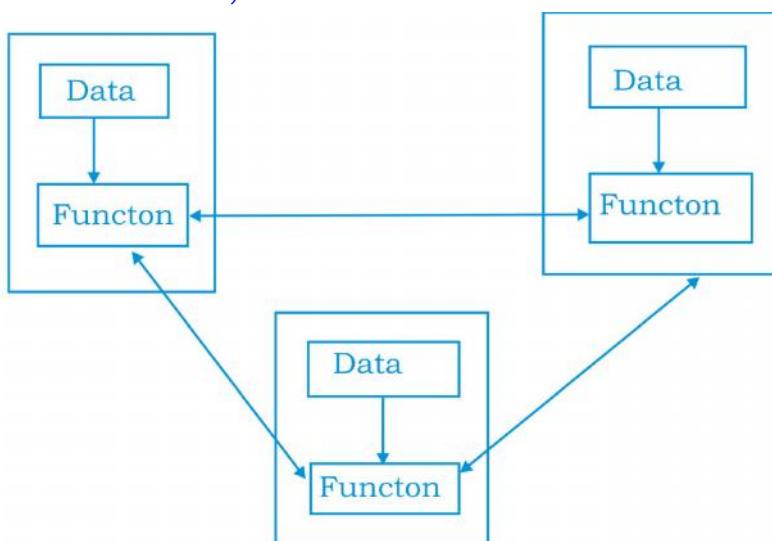


Figure 6.1 Organization of Data and functions in object oriented programming

6.2 Basic concepts of object oriented programming

The following are the major characteristics of any object oriented programming language. They are

- Objects
- Classes
- Data abstraction
- Data encapsulation
- Inheritance
- Overloading
- Polymorphism
- Dynamic binding
- Message passing

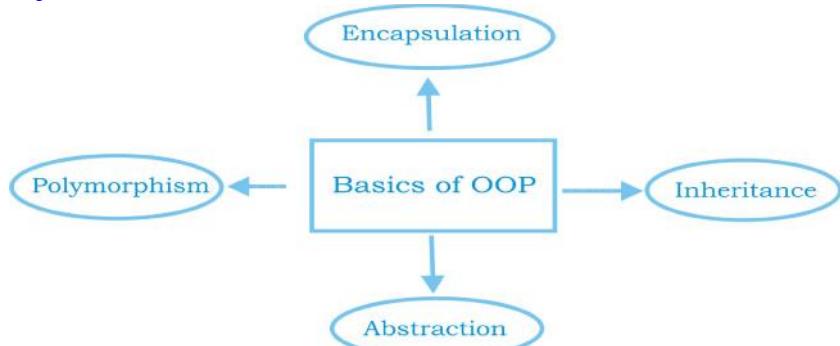


Fig 6.2 Components of OOP

6.2.1 Objects

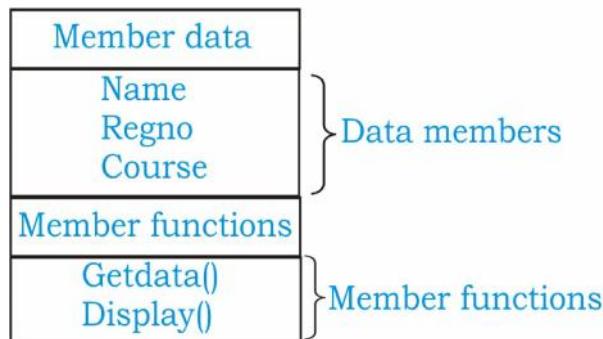
Objects are basic building blocks for designing programs. An object may represent a person, place or a table of data. An object is a collection of data members and associated member functions. Each object is identified by a unique name. Every object must be a member of a particular class.

Ex: Apple, orange, mango are the objects of class fruit.

Objects take up space in memory and have address associated with them. For example, structure variable in a C program.

At the time of execution of a program, the objects interact by sending messages to one another. The objects can interact with one another without having to know the details of data or functions within an object.

Object-1(Student)



Object-2(Teacher)

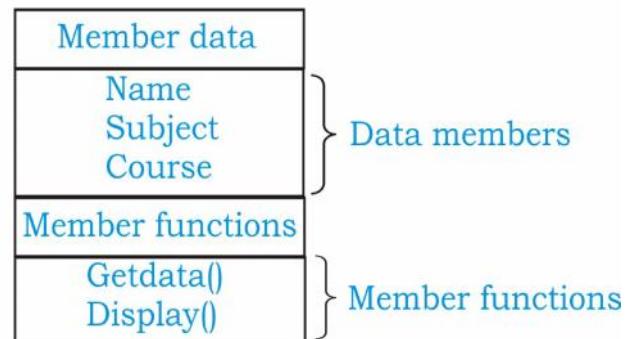


Fig 1.2 Objects containing member data and member funtions

6.2.2 Classes

The objects can contain data and code to manipulate the data. The objects can be made user defined data types with the help of a class. Therefore objects

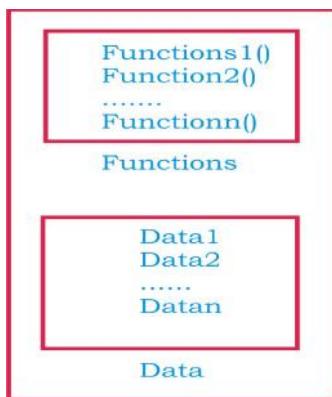


Fig 1.3 Classes

are variables of the type class. A class is a way of grouping objects having similar characteristics. Once a class is defined, any number of objects of that class are created.

For example, planets, sun, moon are members of class solar system.

Classes are user defined data types. A class can hold both data and functions.

6.2.3 Data abstraction

Data abstraction permits the user to use an object without knowing its internal working. Abstraction refers to the process of representing essential features without including background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost, and functions to operate on these attributes.

6.2.4 Data encapsulation

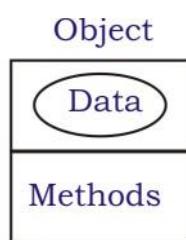


Fig 6.5 Encapsulation of data

Data encapsulation combines data and functions into a single unit called class. Data encapsulation will prevent direct access to data. The data can be accessed only through methods (function) present inside the class. The data cannot be modified by an external non-member function of a class. Data encapsulation enables data hiding or information hiding.

6.2.5 Inheritance

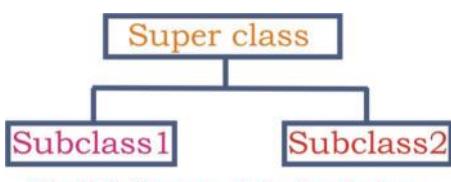


Fig 6.6 Base and derived class

In OOP, the concept of inheritance provides the idea of reusability. This means that we can add additional features to an existing class without modifying it. Thus the process of forming a new class from an existing class is known as Inheritance. The objects of one class acquire the properties of another class through inheritance.

The existing class is known as base class. The new class is known as derived class.

The derived class shares some of the properties of the base class. Therefore a code from a base class can be reused by a derived class. In addition to this the new class may combine features from two different base classes too.

In single inheritance, each subclass has only one superclass. In Multiple inheritance, each subclass has more than one super class.

6.2.6 Overloading

Overloading allows objects to have different meaning depending upon context. There are 2 types of overloading namely

1. Operator overloading
2. Function overloading

When an existing operator operates on new data type, it is called operator overloading.

Function overloading means two or more functions have same name ,but differ in the number of arguments or data type of arguments. Therefore it is said that (function name) is overloaded. Function overloading therefore is the process of defining same function name to carry out similar types of activities with various data items.

6.2.7 Polymorphism

Polymorphism is a feature of object oriented programming where a function can take multiple forms based on the type of arguments, number of arguments and data type of return value.

The ability of an operator and function to take multiple forms is known as polymorphism.

Example 1.2: Consider the addition operation. In addition of 2 numbers the result is the sum of 2 numbers.

In addition of 2 strings the operation is string concatenation. When an operator behaves differently based on operands, then it is said that operator is overloaded. Similarly when same function is used for multiple tasks in the same program by changing argument type and number, it is known as function overloading.

6.2.8 Dynamic Binding

Binding is the process of connecting one program to another. Dynamic binding means code associated with a procedure call is known only at the time of program execution routine.

6.2.9 Message Passing

In OOP, processing is done by sending messages to objects. A message for an object is request for execution of procedure. The request will involve a procedure (function) in the receiving object that generates desired results. Message passing involves specifying the name of object, the name of the function (message) and the information to be sent.

6.3 Advantages of OOP over earlier programming methods

- The programs are modularized based on the principle of classes and objects.
- Linking code & object allows related objects to share common code. This reduces code duplication and code reusability.
- Data is encapsulated along with functions. Therefore external non- member function cannot access or modify data, thus providing data security.
- Easier to develop complex software, because complexity can be minimized through inheritance.
- The concept of data abstraction separates object specification and object implementation.
- Creation and implementation of OOP code is easy and reduces software development time.
- OOP can communicate through message passing which makes interface description with outside system very simple.

6.4 Limitations of OOP

The main disadvantages of using Object oriented programming are:

- OOP software is not having set standards.
- The adaptability of flow diagrams and object oriented programming using classes and objects is a complex process.
- To convert a real world problem into an object oriented model is difficult.
- The classes are overly generalized.

6.5 Applications of object oriented programming

Object oriented programming approach is an easier method to design and implement programs. The programs are easier to upgrade and modify. The standard class libraries can be used by the programmers so that development time is minimized. The graphical user interface design for windows operating system using object oriented programming is the most interesting feature of programming. The common application areas of Object oriented programming are:

- Computer graphic applications
- CAD/CAM software
- Object -oriented Database
- User Interface design such as windows
- Real-time systems
- Simulation and Modeling
- Artificial intelligence and expert systems

Points to remember

- Object oriented programming: Object oriented programming is a programming paradigm that uses “objects” to design applications and computer programs. The OOP uses several techniques such as inheritance, abstraction, modularity, polymorphism and encapsulation.
- Object: Object represents data and associated functions as a single unit.
- Class: A class is a way of grouping objects having similar characteristics.
- Abstraction: Abstraction refers to the representation of essential features of an object as a program object. An abstract class defines an interface, but does not provide implementation details.
- Encapsulation: It is a way of combining data and associated functions into a single unit. Encapsulation implements abstraction.
- Inheritance: It is the capability of a class to inherit the properties of another class. The class that inherits the properties from another class is known as derived or subclass. The class that provides its properties to subclass is known as base class.
- Polymorphism: It is ability of a function to have same name and multiple forms. The appropriate function is called automatically by the compiler depending on the number and type of arguments.
- Message passing: The processing of data in object oriented programming is carried out by sending messages to objects.

One mark questions:

1. What is the fundamental idea of object oriented programming?
2. What is an object?
3. Define the term class.
4. Define the term data abstraction
5. What is encapsulation?
6. What is meant by function overloading?
7. Define polymorphism
8. What is inheritance?
9. What is a base class?
10. What is a derived class?
11. How are base class and derived class related?
12. Define the term data hiding

Two marks questions:

1. What is the significance of classes in OOP?
2. What is the difference between program module and an object?
3. Mention different types of inheritance.

4. Mention any two advantages of object oriented programming over earlier programming methods.

Three mark questions

1. Briefly discuss the classes and objects.
2. Explain inheritance
3. Write short notes on polymorphism.
4. Mention any 4 high level languages that follow object oriented programming approach.

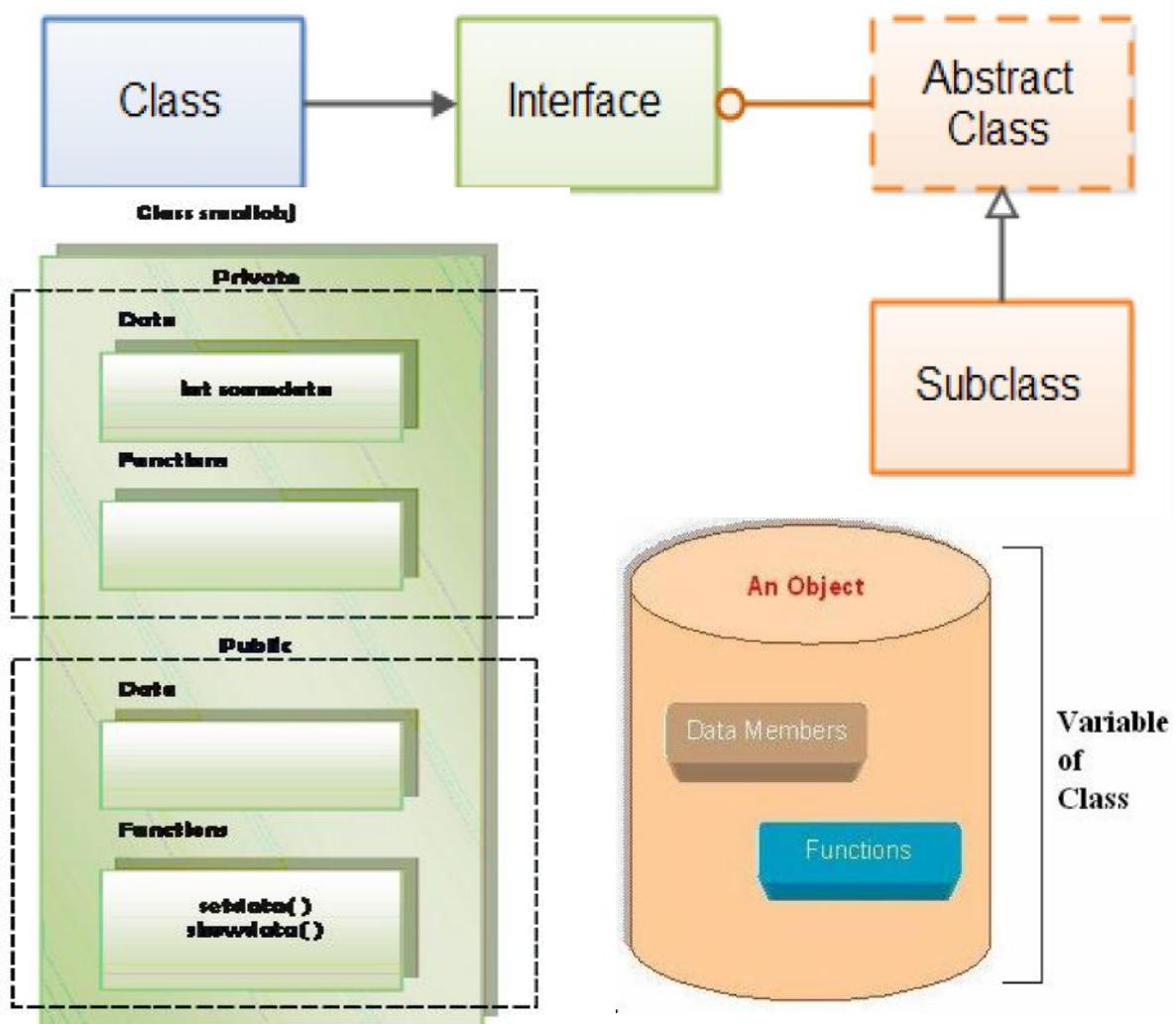
Five marks answer questions

1. Write the differences between procedural programming and object oriented programming
2. Explain advantages OOPs
3. Write the disadvantages of object oriented programming
4. Write the real life applications of object oriented programming.

Chapter 7 Classes and objects

Objectives:

- To understand the need of classes.
- To understand the definition and declaration of classes
- To understand the use of members functions inside and outside classes
- To understand the process of accesing and manipulating the members of class
- To illustrate use of classes through application of simple programs.



7.1 Introduction

In the previous chapter we have learnt the fundamentals units of an object oriented programming. In this chapter let us understand the basic units of data representation for development of any object oriented programming namely classes and objects.

Real world objects can be represented as data variables in computer domain. Data variables are identified as account numbers, balance_payment, Employee_code and so on. When these real world objects are represented as data in computers, they can also be manipulated or processed by using functions in C++. For instance account information of a customer in a bank can be updated, balance_payment can be computed and so on. To combine the data variables (objects) along with appropriate functions for manipulation of these objects, the concept of classes were introduced in object oriented programming.

The elementary concept of object oriented programming begins with the use of classes and objects. A class is a very powerful keyword in C++. A class declaration defines new user-defined data types that link data and the code that manipulates the data. In other words, a class combines data elements and functions (operations) into a single entity.

We have already discussed the concept of combining of data elements into a group called structures in the previous year. Application of user-defined functions for performing different operations has also been discussed. Classes combine both data elements and operations associated with the data.

Let us look at the terminologies in procedural programming language and object oriented programming.

Procedural programming	Object oriented programming
Variables User-defined data types Structure members Functions Function call	Objects Classes Instance variables Methods Message passing

The data elements in a class are called member data and the functions in a class are called member functions. This chapter explains classes, objects, member functions, data hiding, abstraction, encapsulation and how these features are implemented using classes (further chapters provide details of single inheritance, multiple inheritances, insight to polymorphism and other functions).

7.2 Definition and declaration of classes and objects:

A class definition is a process of naming a class and data variables, and interface operations of the class.

A class declaration specifies the representation of objects of the class and set of operations that can be applied to such objects.

The definition and declaration of a class indicates the following:

- The data variables known as member data of a class, describe the characteristics of a class.
- Member functions are the set of operations that are performed on the objects of the class. There may be zero or more member functions in a class. This is also known as interface.
- The access control specifiers to the class members within a program are specified.
- Class tagname for external operations for accessing and manipulating the instance of a class.

The General syntax for defining a class is as follows:

```
class user_defined_name
{
    private:
        Member data
        Member functions
    protected:
        Member data
        Member functions
    public:
        Member data
        Member functions
};
```

Keyword **class** is used to declare a class. User_defined_name is the name of the class.

Class body is enclosed in a pair of flower brackets. Class body contains the declaration of its members (data and functions). There are generally three types of members namely private, public and protected. These are discussed in the following section.

The class function definition describes how the class functions are implemented.

Example 7.1: Let us declare a class for representation of bank account.

```
class account
{
    private:                                //implicit by default
        int accno;
        char name[20];
        char acctype[4];
        int bal_amt;
    public:                                 //member functions
        void get_data();
        void displaydata();
}
```

In the above example, class name account is a new type identifier that is used to declare instance of that class type. The class account contains four member data and two methods or member functions. Both of these member data are private by default, while both member functions are public by default.

The functions get_data() and put_data() are used to write instructions to perform operations on member data. These two functions only can provide access to get member data from outside the class. Therefore, the data cannot be accessed by any other function that is not a member of the class account.

Program 7.1 Program to show the use of class and object

```
#include<iostream.h>
#include<conio.h>
class Test
{
    private:
        int a,b;
    public:
        void getnum()
        {
            cout<<"Enter Numbers for Addition: ";
            cin>>a>>b;
        }
        void show_data()
        {
            cout<<"Addition of two numbers: "<<(a+b);
        }
};
```

```

int main()
{
    clrscr();
    Test a1;
    a1.getnum();
    a1.show_data();
    getch();
    return 0;
}

```

Enter Numbers for Addition: 50 60

Addition of two numbers: 110

In the above program a, b are data members of the class Test. get_num() and show_add() are member functions. get_num() is used to input data values. show_add() function is used to add two numbers and display the sum of two numbers.

7.3 Access specifiers

An object is defined to be an instance of a class. Every data member of a class is specified by three levels of access protection for hiding data and function members internal to the class. The access specifiers define the scope of data. The access specifiers are indicated using the following keywords.

7.3.1 private

private access means a member data can only be accessed by the member function. Members declared under private are accessible only within the class. If no access specifier is mentioned, then by default, members are private.

Example 7.2: **private:**

```

int    x;
float  y;

```

7.3.2 public

public access means that members can be accessed by any function outside the class also.

Example 7.3: **class box**

```

{
    int    length;
    public : int width;
    private : int height;
    void set_height ( int i )
    {
        height = i ;
    }
}

```

```

        int get_height ( )
        {
            return height ;
        }
    };
int main()
{
    box object;
    object.length = 10;
    object.width = 20;
    object.set_height ( 30); // private variable can be accessed
                           return 0;           only through its public method
}

```

Thus keyword public identifies both class data and functions that constitute the public interface for the class. In the above program data member width is a public variable. set_height() and get_height() are member functions. The private variable height can be accessed only through the member functions of the class.

Thus keyword public identifies both class data and functions that constitute the public interface for the class.

7.3.3 Protected

The members which are declared using protected can be accessed only by the member functions, friends of the class and also by the member functions derived from this class. The members cannot be accessed from outside. The protected access specifier is therefore similar to private specifier. The difference is discussed in detail in further chapters.

7.4. Members of the class

The members of a class can be data or functions. Private and protected members of a class can be accessed only through the member functions of that class. No function outside a class can include statements to access data directly. The public data members of objects of a class can be accessed using direct member access operator (.)

The syntax for accessing class member is

```

class-object. member-data
class-object.member-function(arguments);

```

Example 7.4: class rectangle

```

{
    int l;
    int b;
public:

```

```

        void get_data();
        void compute_area();
        void display();
    };
int main()
{
    rectangle r1;
    r1.get_data();
    r1.compute_data();
    r1.display();
    return 0;
}

```

In the above example `a` and `b` are data members of class `rectangle` and `get_data()`, `compute_data()` and `display()` are member functions. `r1` is a class variable that invokes the member functions.

Program 7.2. Program to show the use of access specifiers with classes and objects

```

#include<iostream.h>
#include<conio.h>
class data
{
private:
    int day;
    int month;
    int year;
public:
    void date(int dd,int mm,int yy)
    {
        day = dd;
        month = mm;
        year = yy;
        cout<<"\t"<<day<<"\t"<<month<<"\t"<<year<<endl;
    }
};
int main()
{
    clrscr();
    data date1,date2;
    date1.date(7,12,2014);
    date2.date(8,12,2014);
    date1.date(12,12,2014);
    getch();
}

```

```

        return 0;
    }
}

```

In the above program day, month and year are private members and date is a public function. The output of the above program is as follows.

7	12	2014
8	12	2014
12	12	2014

7.5 Member functions

Member functions can be defined in two places:

- Inside class definition
- Outside class definition

7.5.1 Inside class definition

To define member function inside a class the function declaration within the class is replaced by actual function definition inside the class. A function defined in a class is treated as inline function. Only small functions are defined inside class definition.

Syntax: return_type classname(member function)

Example 7.5:

```

class rectangle
{
    int length;
    int breadth;
public:
    void get_data ()
    {
        cin>>length;
        cin>>breadth;
    }
    void put_data (void)
    {
        cout <<length ;
        cout <<breadth;
    }
};

```

7.5.2 Outside class definition

A function declared as a member of a class is known as member function. Member functions declared within a class must be defined separately outside the class. The definition of member function is similar to normal function. But a

member function has an ‘identity label’ in the header. This label tells the compiler which class the function belongs to. The scope of the member function is limited to the class mentioned in the header. Scope resolution operator :: is used to define the member function.

Syntax: For member function outside a class

```
return_type classname :: memberfunction(arg1, arg2, ..., argn)
{
    function body;
}
```

The member functions have following characteristics:

- Type and number of arguments in member function must be same as types and number of data declared in class definition.
- The symbol :: is known as scope resolution operator. Usage of :: along with class name is the header of function definition. The scope resolution operator identifies the function as a member of particular class.
- Several classes can use same function name. Membership label will resolve their scope.
- Member function can access private data of a class. A non-member function cannot.

Example 7.6: The following program segment shows how a member function is defined outside class.

```
class operation
{
    private :
        int a;
        int b;
    public :
        int sum();
        int product();
};

int operation::sum()
{
    return (a+b) ;
}

int operation (product)
{
    return (a*b) ;
}
```

In the above example sum() and product() are member functions defined outside class operation. **The use of scope resolution operator implies that these member functions are defined outside the class.**

Program 7.3 To use classes using member functions inside and outside class definition.

```
# include<iostream.h>
class item // class declaration
{
    int number;           // private by default
    float cost;
public :
    void getdata(int a, float b);
    void putdata(void)    // function defined here
    {
        cout <<"Number: "<< number << endl;
        cout << "Cost: "<< cost << endl;
    }
};
// member function outside class definition
void item :: getdata(int a, float b)
{
    number = a;
    cost = b;
}
// main program
int main()
{
    item x;                // create object x
    x.getdata(250, 10.5);
    x.putdata();
    return 0;
}
```

In the above program one member functions putdata() is defined inside class definition and the other member function getdata() is defined outside class definition. Here is the output of program.

Number : 250

Cost : 10.5

7.6. Defining objects of a class

When a class is defined, it specifies the type information the objects of the class store. Once the class is defined an object is created from that class. The

objects of a class are declared in the same manner like any other variable declaration.

Syntax :

```
class user_defined_name
{
    private:           //Members
    public:            // Methods
};

User_defined_name object1, object2, ... ;
```

Example 7.6:

```
class num
{
    private:
        int x;
        int y;
    public:
        int sum(int p, int q);
        int diff(int p, int q);
};

void main()
{
    num s1,s2;
    s1.sum(200,300);
    s2.diff(600,500);
}
```

Note that an object is an instance of a class template.

Example 7.6 A class to create studentname, rollno, sex, age.

```
class student
{
    int rollno;
    char name[20];
    char gender;
    int age;
public:
    void get_data();
    void display_data();
};

student obj1, obj2;      //Obj1, obj2 are objects of class student
```

7.7 Arrays as members of classes

So far we have considered primary data values as member of class. Just like arrays within structures, it is possible to use arrays as member data of class type.

Example 7.7: class marks

```

{
    int m[5];
    int i ;
public:
    void setval(void);
    void display(void);
};
```

The array variable `m` is a private member of class `marks`. This can be used by member function `setval()` and `display()` as follows.

```

void marks :: setval(void)
{
    cout<< "Enter marks: "<<endl;
    for (i=0;i<5;i++)
        cin>>m[i] ;
}
void marks::display(void)
{
    cout<< "The marks are : ";
    for(i =0;i<5;i++)
        cout<<setw(4)<<m[i]<<endl;
}
```

7.8 Array of objects

An array having class type elements is known as array of objects. An array of objects is declared after class definition and is defined in the same way as any other array.

Example 7.8: class employee

```

{
    char name[10] ;
    int age;
public:
    void getdata();
    void dispdata(void);
};
employee supervisor[3] ;
employee sales_executive[5] ;
employee team_leader[10] ;
```

In the above example, the array `supervisor` contains 3 objects namely `supervisor[0]`, `supervisor[1]`, `supervisor[2]`.

	Name	Age
supervisor[0]		
supervisor[1]		
supervisor[2]		

Storage of data items in an object array

Program 7.4 Program to show the use of array of objects

```
#include<iostream.h>
#include<conio.h>
class data
{
    int rollno, maths, science;
public:
    int avg();
    void getdata();
    void putdata();
};
void data::getdata()
{
    cout<<"Enter rollno: ";
    cin>>rollno;
    cout<<"Enter maths marks: ";
    cin>>maths;
    cout<<"Enter science marks: ";
    cin>>science;
    putdata();
}
int data::avg()
{
    int a;
    a=(maths+science)/2;
    return a;
}
void data::putdata()
{
    cout<<"Average = "<<avg()<<endl;
}

int main()
{
```

```

clrscr();
data stud[3];
for(int i=0;i<3;i++)
    stud[i].getdata();
return 0;
}

```

```

Enter rollno: 23
Enter maths marks: 56
Enter science marks: 78
Average = 67
Enter rollno: 24
Enter maths marks: 56
Enter science marks: 77
Average = 66
Enter rollno: 12
Enter maths marks: 44
Enter science marks: 89
Average = 66

```

In the above program stud is an array of objects, data is a class with member functions getdata() and putdata() defined outside the class definition.

7.9. Objects as function arguments

A function can receive an object as a function argument. This is similar to any other data type being sent as function argument. An object can be passed to a function in two ways:

- Copy of entire object is passed to function (pass-by-value).
- Only address of the object is transferred to the function (pass-by-reference).

In pass by value, a copy of the object is passed to the function. The function creates its own copy of the object and uses it. Therefore changes made to the object inside the function do not affect the original object.

In pass by reference, when an address of an object is passed to the function, the function directly works on the original object used in function call. This means changes made to the object inside the function will reflect in the original object, because the function is making changes in the original object itself.

Pass-by-reference is more efficient, since it require only to pass the address of the object and not the entire object.

Program 7.4: Program to show objects as arguments to function.

```
#include<iostream.h>
#include<conio.h>
class currency
{
    int rupee, paise;
    int total;
public:
    void get_value(int r, int p);
    void display(void);
};
void currency::get_value(int r, int p)
{
    rupee = r;
    paise = p;
    total = r*100 + p;
}
void currency::display(void)
{
    cout<<rupee<<" Rupees "<<" and "<<paise<<" paise"<<endl;
    cout<<"Converted value: "<<total<<" paise";
}
int main()
{
    currency c;
    c.get_value(5, 75);
    c.display();
}
```

5 Rupees and 75 paise
Converted value: 575 paise

Program 7.5: Program to show objects as arguments to function.

```
#include <iostream.h>
#include<conio.h>
class rup
{
private:
    int n1,n2;
public:
    rup():n1(1),n2(1)
```

```

    {
}
void get()
{
    cout<<“enter first number”;
    cin>>n1;
    cout<<“enter second number”;
    cin>>n2;
}
void print()
{
    cout<<“product=”<<n1<<endl;
    cout<<“product2=”<<n2<<endl;
}
void multi(rup r1,rup r2)
{
    n1=r1.n1*r1.n2;
    n2=r2.n1*r2.n2;
}
int main ()
{
    rup r1,r2,r3;
    clrscr();
    r1.get();
    r2.get();
    r3.multi(r1,r2);
    r3.print();
    getch();
    return 0;
}

```

7.10. Differences between structure and class:

In C++, a structure is a class defined with the keyword struct. Its members and base classes are public by default. A class is defined with the class keyword

Points to remember:

- A class is a user defined data that contains data members and objects
- An object is an instance of a class.
- Members: There are two types of members in a class. They are data member and member functions.
- Data members are different data variables similar to structure members.
- Data members may be declared in a class as private, public or protected.

- Private members are default by declaration. They are accessible only to member functions within a class only.
- Public members can be accessed inside as well as outside class definition.
- Member function also known as method, acts on data members in the class.
- Member functions may be defined inside or outside a class.
- Scope resolution operator :: is used when a global variable exists with the same name as local variable. This is also used in C++ when member functions are declared outside the class definition. The function name is preceded by the class name and scope resolution operator.
- Class members are accessed using dot(.) operator.
- An array having class type elements is known as an array of objects.
- Functions can receive objects as function arguments.
- It is possible to pass objects to a function using pass by value or pass by reference.

One mark questions:

1. What is a class?
2. What is an object?
3. What are the two types of members referenced in a class?
4. What are data members?
5. What is a member function?
6. Mention the access specifiers used with a class
7. Is it possible to access data outside a class?
8. Which type of data members are accessible outside a class?
9. Which access specifier is implicitly used in a class?
10. Define the term public access
11. Mention the operator used to access members of a class
12. What is the significance of scope resolution operator (::)?
13. How are objects of a class declared? Give an example
14. What is meant by an array of objects?
15. Write an example to show how objects can be used as function arguments?

Two/Three mark questions

1. Write the differences between class definition and class declaration.
2. Write the syntax and example for class definition.
3. Write the syntax and example for class declaration.
4. What is the significance of using access specifiers? Mention different access specifiers.
5. Discuss private access specifier with an example.

6. Write short notes on public access specifier.
7. Explain protected access specifier.
8. How are class members referenced? Discuss with suitable example.
9. What is meant by referencing member functions inside class definition and outside class definition?
10. Discuss how objects of a class are referenced with an example.
11. How can arrays be used as class members. Write an example.
12. How are objects passed as arguments to a function? Give an example.

Five mark questions

1. Explain class definition and class declaration with syntax and example.
2. Describe access specifiers in a class.
3. Explain member functions
 - a. Inside class definition
 - b. Outside class definition
4. What are the characteristics of member functions outside a class?
5. Explain how objects of a class can be defined?
6. Illustrate how an array of objects can be defined.
7. Describe how objects can be used as function arguments.
8. Let product list be a linear array of size N where each element of the array contains following fields : Itemcode, price and quantity. Declare a class product list with the three data members and member functions to perform the following :
 - a. Add values to the product list.
 - b. Printing the total stock value.
9. A class clock has following members : a. hour b. minute
Create member functions
 - a. To initialize the data members
 - b. Display the time
 - c. To convert hours and minutes to minutes
10. Write a program that receives arrival time, departure time and speed of an automobile in kilometers/hour as input to a class. Compute the distance travelled in meters/second and display the result using member functions.

Chapter 8

FUNCTION OVERLOADING AND MEMBER FUNCTIONS

Objects:

- Ø Need for function overloading
- Ø Concept of polymorphism
- Ø Application of function overloading through simple examples
- Ø Concept of inline functions
- Ø Use friend functions

```
→ void function_name()
{
    .....
}

→ void function_name(int a)
{
    .....
}

→ void function_name(char ch)
{
    .....
}

int main()
{
    int i;
    char c;

    function_name(c);
    function_name(i);
    function_name();

    .....
}
```

no argument call

Passing integer as an argument

Passing character as an argument

8.1 Introduction

In the previous chapter, we have learnt about classes and objects. The application of classes with data members and member functions is the principle feature of object oriented programming to develop and simplify programming methods. Another feature to help for easier programming is polymorphism. The definition of polymorphism has appeared in chapter 6. Let us recall the definition once again. Polymorphism refers to “one name having many forms, different behavior of an instance of an object depending on situations”.

C++ implements polymorphism through function overloading and operator overloading. The function overloading allows the user to create new abstract data types. In this chapter we learn about the need for function overloading, definition and declaration of function overloading and some examples. The discussion is limited to design of a set of functions that perform essentially the same thing, but with a different argument list. The selection of overloaded function depends on matching arguments at the time of compilation. The study in this chapter is also extended to inline and friend functions.

8.2 Need for function overloading

Function overloading means two or more functions have same name, but differ in the number of arguments or data type of arguments. Therefore it is said that (function name) is overloaded. Function overloading therefore is the process of defining same function name to carry out similar types of activities with various data items. The advantages of function overloading are:

- When different functions are created for different operations, then user has to call respective function depending on the situation. Instead, for different situations if the same function is called with different arguments using function overloading, then the compiler automatically decides about the appropriate function by comparing the argument types used in the call to the function and calls the required function. Thus the code is executed faster.
- It is easier to understand the flow of information and debug.
- Code maintenance is easy.
- Easier interface between programs and real world objects.

8.3 Definition and declaration of overloaded functions

The main factor in function overloading is a function's argument list. C++ can distinguish overloaded functions by the number and type of arguments. If there are two functions having same name and different types of arguments or different number of arguments, then function overloading is invoked automatically by the compiler. Function Overloading is also known as Compile time polymorphism.

Example 8.1:

```
int sum(int a, int b);
float sum(float p, float q);
```

The function `sum()` that takes two integer arguments is different from the function `sum()` that takes two float arguments. This is function overloading.

To overload a function, each overloaded function must be declared and defined separately.

Example 8.2:

```
int product(int p, int q, int r);
float product(float x, float y, float z);
int product(int p, int q, int r)
{
    cout << "product=" << p * q * r << endl;
}
float product(float x, float y, float z)
{
    cout << "product=" << x * x * y * y * z * z << endl;
}
```

In this example the function `product()` is overloaded two times. First time with three integer values and integer as return value, second time with three float values and return type being a float. The compiler automatically chooses the right type of function depending on the number of arguments.

8.4 Restrictions on overloaded functions

- Ø Each function in a set of overloaded functions must have different argument list.
- Ø If `typedef` is used for naming functions, then the function is not considered as different type.

8.4.1 Calling overloaded functions

The following programming example shows how overloaded functions can be called.

Program 8.1: To compute volume of cone, cube and cylinder using overloaded functions.

```
#include<iostream.h>
#include<conio.h>
class funoverload
{
public:
    int volume(int a)           // Volume of Cube
    {
        return a*a*a;
    }
}
```

```

        double volume(double r, double h)           // Volume of Cone
    {
        return (0.33*3.14*r*r*h);
    }
    double volume(double r, int h)             // Volume of Cylinder
    {
        return (3.14*r*r*h);
    }
    double volume(double l, double b, double h) //Volume of Cuboid
    {
        return (l*b*h);
    }
};

int main()
{
    clrscr();
    funoverload f1;
    cout<<"Volume of the Cube: "<<f1.volume(10)<<endl;
    cout<<"Volume of the Cone: "<<f1.volume(2.0,3.0)<<endl;
    cout<<"Volume of the Cylinder: "<<f1.volume(2.0,3)<<endl;
    cout<<"Volume of the Cuboid: "<<f1.volume(5.0,6.0,7.0)<<endl;
    return 0;
    getch();
}

```

Volume of the Cube: 1000
 Volume of the Cone: 12.4344
 Volume of the Cylinder: 37.68
 Volume of the Cuboid: 210

The above program finds volume of cube, cuboid, cone or cylinder depending on the number and type of arguments provided as input to the function called volume. Therefore though all functions have the same name volume, appropriate function is selected based on data type of argument list. The compiler selects the required function through function overloading.

8.5 Other functions in a class

The member functions of a class may be defined inside or outside a class. One such function defined inside a class is inline function.

8.5.1 inline functions

A function call generally involves the complex process of invoking a function, passing parameters to the function, allocating storage for local variables, thereby using extra time and memory space. It is possible to avoid these overheads of a function by using inline function. The inline function is a short function. Compiler replaces a function call with the body of the function.

The keyword `inline` is used to define inline functions. The inline function consists of function call along with function code and the process is known as expansion.

- Inline functions definition starts with keyword `inline`.
- The inline functions should be defined before all functions that call it.
- The compiler replaces the function call statement with the function code itself (expansion) and then compiles the entire code.
- They run little faster than normal functions as function calling overheads are saved.

Advantages of inline functions

- The inline member functions are compact function calls.
- Therefore the size of the object code is considerably reduced.
- The speed of execution of a program increases.
- Very efficient code can be generated.
- The readability of the program increases.

Disadvantage

As the body of inline function is substituted in place of a function call, the size of the executable file increases and more memory is needed.

Program 8.2: Finding the square of a number using inline functions.

```
#include <iostream.h>
inline int square (int a)
{
    return(a*a);
}
int main( )
{
    int x, y;
    x=square(5);
    cout<<"Square of 5 = "<<x<<endl;
    y=square(10);
    cout<<"Square of 10 = "<<y<<endl;
    return 0;
}
```

```
Square of 5 = 25
Square of 10 = 100
```

In the above example square() is an inline function that finds the square of a number.

Note: The inline function may not work some times for one of the following reasons:

- The inline function definition is too long or too complicated.
- The inline function is recursive.
- The inline function has looping constructs
- The inline function has a switch or goto.

8.5.2 friend functions

We have seen that private and protected members of a class cannot be accessed from outside the class in which they are declared. In other words non-member function does not have access to the private data members of a class. But there could be a situation where two classes must share a common function. C++ allows the common function to be shared between the two classes by making the common function as a friend to both the classes, thereby allowing the function to have access to the private data of both of these classes.

A friend function is a non-member function that is a friend of a class. The friend function is declared within a class with the prefix **friend**. But it should be defined outside the class like a normal function without the prefix friend. It can access public data members like non-member functions.

Syntax:

```
class class_name
{
    public:
        friend void function1(void);
        friend returntype_specifier function_name(arguments);
};
```

Example 8.3:

```
class base
{
    int val1, val2;
public:
    void getdata()
    {
        cout<<“Enter two values:”;
        cin>>val1>>val2;
    }
};
```

```
friend float mean(base ob);
float mean(base ob)
{
    return float(ob.val1+ob.val2)/2;
}
```

In the above example mean() is declared as friend function that computes mean value of two numbers that are input using getdata() function.

- A friend function although not a member function, has full access right to the private and protected members of the class.
 - A friend function cannot be called using the object of that class. It can be invoked like any normal function.
 - They are normal external functions that are given special access privileges.
 - It cannot access the member variables directly and has to use an object name.membername (Here, is a membership operator).
 - The function is declared with keyword friend. But while defining friend function it does not use either keyword friend or :: operator.
-

Program 8.3 To indicate the use of friend function.

```
#include <iostream.h>
class myclass
{
    private:
        int a,b;
    public:
        void set_val(int i, int j);
        friend int add(myclass obj);
};

void myclass::set_val(int i,int j)
{
    a = i;
    b = j;
}

int add(myclass obj)
{
    return (obj.a+obj.b);
}
```

```
int main()
{
    myclass object;
    object.set_val(34, 56);
    cout << "Sum of 34 and 56 is "<<add(object)<<endl;
    return 0;
}
```

Sum of 34 and 56 is 90

In the above program segment the function add() is a friend of class myclass. Since it is not a member of any class, it cannot be called like an object. Since it is a non-member function, add() should be declared inside a class under public or private or protected access specifier.

Points to remember

- A function name that has several definitions with respect to the number of arguments and type of arguments is known as function overloading.
- Function overloading implements polymorphism
- Inline functions are member functions defined inside a class.
- The function code is written along with function definition inside a class
- Friend function is a non member function of a class that has access to both private and protected access members

Review questions**One mark questions**

1. What is meant by function overloading?
2. When is function overloading needed?
3. Write an advantage of function overloading.
4. Name one condition for overloading of functions.
5. What is an inline function?
6. Write one advantage of inline function.
7. The inline function is always converted to a code block. True/false
8. What is a friend function?
9. Write an example for friend function declaration.
10. Write one condition for using a friend function.
11. Can the keyword friend be used while declaring a friend function?
12. Overloading a function means using different names to perform the same operations. True/false

Two marks questions:

1. What is meant by overloading implements polymorphism?
2. Write example for definition and declaration of function overloading
3. What are the restrictions on overloaded functions?
4. What are inline functions? Give an example.
5. Write the advantages of inline functions.
6. Create an inline function to check if a number is prime or not.
7. When are friend functions used? Write syntax for declaration of friend function.
8. Write any two characteristics of friend function.

Three mark questions:

1. Write any three reasons for function overloading.
2. What are the advantages of inline functions?
3. Write the advantages and disadvantages of inline functions.
4. When is it possible that an inline function may not work?
5. Write any three characteristics of friend function.

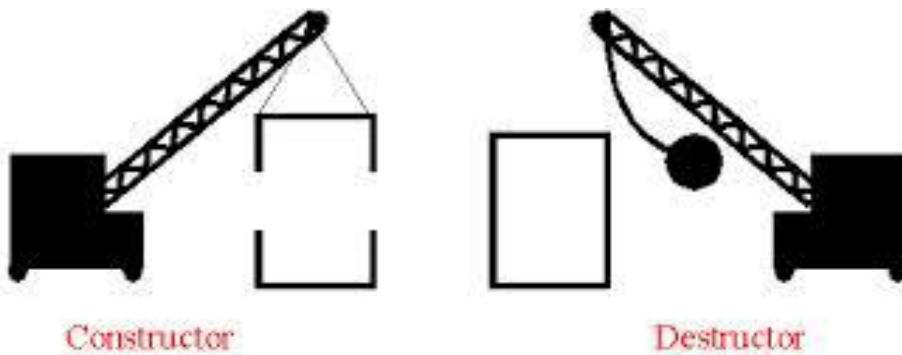
Five mark questions:

1. Explain the need for function overloading.
2. Discuss overloaded functions with syntax and example.
3. Explain inline functions with syntax and example.
4. Explain friend functions and their characteristics.

Chapter 9
CONSTRUCTORS AND DESTRUCTORS

Objectives:

- Need for use of constructors
- To identify different types of constructors
- To highlight implicit and explicit declaration of constructors
- To understand the need for constructor overloading
- To understand the need for destructors



MyClass *MyObjPtr = new MyClass();

delete MyObjPtr;

9.1 Introduction:

In the previous chapters, we have learnt to use classes, member data and member functions of a class. The use of polymorphism through function overloading has simplified object oriented programming further. In real life applications, sometimes the objects may have to be initialized before using them. The initialization is normally done by a member function which initializes data members to pre-defined values. In this chapter we discuss about special member functions that are used to initialize and destroy the objects of a class automatically so that memory utilization can be optimized.

It is sometimes convenient if an object can initialize itself when it is first created, without the need to make a separate call to a member function. This is possible with special member functions. Automatic initialization is carried out using such a special member function called **constructor**. Generally, we can say, a constructor constructs the data members of an object. i.e., initializes a value to the data members.

“A constructor is a special member function that is used in classes to initialize the objects of a class automatically”.

A constructor is a member function of a class with the same name as that of the class. A constructor is defined like other member functions of a class. It can be defined either inside the class definition or outside the class definition.

It is called constructor because it constructs the values of data members of the class. The following rules are used for writing a constructor function:

1. A Constructor always has name that is same as the class name of which they are the members. This will help the compiler to identify that they are the constructors.
2. There is no return type for constructors (not even void). Since, the constructor is called automatically by the system, there is no program for it to return anything to; a return value would not make sense.
3. A constructor should be declared in public section.
4. A constructor is invoked automatically when objects are created. Constructors can have default arguments.
5. It is not possible to refer to the address of the constructors.
6. The constructors make implicit calls to the operators **new** and **delete** when memory allocation is required.

9.2 Declaration of constructor inside the class definition

Example 9.1: class test

```

{
    int m, x, y;
public:

    test()      //constructor inside the class definition
    {
        x = 0; y = 0;
    }
};
```

Program 9.1: Use of constructor.

```

#include <iostream.h>
#include <iomanip.h>
class counter
{
private:
    int a;
public:
    counter()
    {
        a = 0;
    }
    void inc_count()
    {
        a++;
    }
    int get_count()
    {
        return a;
    }
};
int main()
{
    counter c1, c2;
    cout<<"c1 = "<<c1.get_count();
    cout<<"c2 = "<<c2.get_count();
    c1.inc_count();
    c2.inc_count();
    cout<<"c1 = "<<c1.get_count()<<endl;
    cout<<"c2 = "<<c2.get_count()<<endl;
}
```

The above code segment contains a constructor with the name counter(), which is same as the class name. This constructor initializes the class variable a to 0. When, in main(), an object for the counter class is defined will invoke or call the constructor counter() and initialize the variable a to 0. i.e., in main() we are defining two instances for the class counter(), c1 and c2. Now, the constructor counter will be called twice automatically, once for object c1 and once for object c2.

Note that when a constructor is declared for a class, initialization of class objects is done automatically.

9.3 Types of constructors:

There are three types of constructors, namely:

1. Default constructor
2. Parameterized constructor
3. Copy constructor

9.3.1 Default Constructor

A constructor which does not accept any arguments is called a zero argument constructor. It is also known as default constructor. The default constructors are useful when the objects are need to be created without having to type initial values. Default constructor simply allocated memory to data members of objects. Features of default constructor are

- For every object created, this constructor is automatically called.
- All objects of a class are initialized to same set of values by the default constructor.
- If different objects are to be initialized with different values, it cannot be done using default constructor.

Note:

- When a user-defined class does not contain an explicit constructor, compiler automatically invokes default constructor.
- Declaring a constructor with arguments, hides default constructor.

Syntax: classname::classname //constructor without arguments
 { }

Example 9.2: student() { }

Program 9.2: To show initialization of an object using default constructor.

```
#include<iostream.h>
#include<iomanip.h>
class x
{
    private:
        int a, b;
    public:
        x() { a=10, b=20; }
        void display()
        {
            cout<<a<<setw(5)<<b<<endl;
        }
};
void main()
{
    x obj1,obj2;
    obj1.display();
    obj2.display();
}
```

```
10 20
10 20
```

Program 9.3: To show the default initialization of constructor member function using scope resolution operator.

```
#include<iostream.h>
#include<string.h>
#include <ctype.h>
class student
{
    private:
        int regno;
        char name[20];
    public:
        student(); //constructor
        void display();
};
student : :student()
{
    regno = 0;
    strcpy(name, "Book");
}
```

```

void student::display()
{
    cout<<"Reg. No: "<<regno<<endl;
    cout<<"Name: "<<name<<endl;
}
void main()
{
    student s1;
    cout<<"Use of default constructor:"<<endl;
    s1.display();
}

```

Use of default constructor:

Reg. No: 0
Name: Book

The above program shows the use of default constructor student that initializes the members automatically as soon as they are created.

Disadvantages of default constructor:

- When many objects of the same class are created, all objects are initialized to same set of values by default constructor
- It is not possible to initialize different objects with different initial values using default constructor.

9.3.2 Parameterized Constructors:

A constructor that takes one or more arguments is called parameterized constructor. Using this constructor, it is possible to initialize different objects with different values.

Parameterized constructors are also invoked automatically, whenever objects with arguments are created. The parameters are used to initialize the object.

Features of parameterized constructors:

- The parameterized constructors can be overloaded.
- For an object created with one argument, constructor with only one argument is invoked and executed.
- The parameterized constructor can have default arguments and default values.

Invoking constructors

A constructor is automatically invoked by C++ compiler with an object declaration. The constructors can be invoked through the following methods:

1. Explicit call
2. Implicit call
3. Initialization at the time of declaration with = operator

Explicit call

In explicit call, declaration of an object is followed by assignment operator, constructor name and argument list enclosed in parenthesis.

Program 9.4: To use parameterized constructor through explicit call

```
#include<iostream.h>
class num
{
    private:
        int a,b;
    public:
        num(int p, int q) {a = p,b = q;}
        void display()
        {
            cout<<"a = "<<a<<" and b = "<<b<<endl;
        }
};
void main()
{
    num obj1=num(10,20);
    num obj2=num(40,50);
    cout<<"First construction: ";obj1.display();
    cout<<"Second construction: ";obj2.display();
}
```

First construction: a = 10 and b = 20
 Second construction: a = 40 and b = 50

In the above program code the objects obj1 and obj2 are called explicitly by using parameterized constructor.

Implicit call

An Implicit call means the declaration of the object is followed by argument list enclosed in parentheses.

Program 9.5: Program to initialise the data members using implicit declaration

```
#include<iostream.h>
class num
{
    private:
```

```

        int a,b;
    public:
        num(int m, int n)
        { a = m, b = n; }
    void display()
    {
        cout<<"a= "<<a<<" b= "<<b<<endl;
    }
};

void main()
{
    num obj1(10,20);
    num obj2(40,50);
    obj1.display();
    obj2.display();
}

```

```

a=10  b= 20
a=40  b=50

```

Note that:

1. One parameterized constructor num(int m, int n) is defined inside the class.
2. For each object, different values are passed from function main(). Therefore different objects can be initialized to different values.

Initialization of objects during declaration with assignment operator

This method is used for the constructor with exactly one argument. In this method declaration is followed by assignment operator and value to be initialized.

Program 9.6: To initialize objects using assignment operator

```

#include<iostream.h>
class num
{
private:
    int a;
public:
    num(int m) { a = m; }
    void display()
    {
        cout<<a<<endl;
    }
};

void main()
{

```

```
    num obj1=100;  
    num obj2=200;  
    cout<<"Object1 = ";obj1.display();  
    cout<<"Object2 = ";obj2.display();  
}
```

Object1 = 100
Object2 = 200

In the above example, the constructors obj1 and obj2 are initialized using = operator.

Note that

1. This method is applicable only to the constructors that have exactly one parameter.
 2. If a class contains at least one parameterized constructor, then necessary arguments have to be passed to the constructor in the declaration itself.
 3. If user does not want to pass arguments, then default constructor must be created in the class.

Example 9.1: num obj1(10,20);

```
num obj1(10,-1);  
num obj2;           //error  
num() {}    //This default constructor must be included in  
            //class num to avoid errors
```

The above example shows that it is necessary to use a default constructor num() to avoid error during the declaration of num obj2;

9.3.3 Copy constructor

Copy constructor is a parameterized constructor using which one object can be copied to another object. Copy constructors are used in the following situations

- To initialize an object with the values of already existing objects.
 - When objects must be returned as function values
 - To state objects as by value parameters of a function.

Copy constructor can accept a single argument of reference to same class type. The argument must be passed as a constant reference type.

Syntax: classname :: classname(classname &ptr)

Example 9.2: `x::x(x &ptr)`

Here, x is a class name and ptr is a pointer to a class object x.

If there is more than one argument present in the copy constructor, it must contain default arguments.

Note that:

1. Copy constructor is not invoked explicitly.
2. Copy constructor is invoked automatically when a new object is created and equated to an already existing object in the declaration statement itself.

Example 9.3:

```
x      a1;           //default constructor
x      a2 = a1;       //copy constructor
a1.display();
```

The above example shows the use of copy constructor to create a new object a2 using existing object a1.

3. When a new object is declared and existing object is passed as a parameter to it in the declaration, then also copy constructor is invoked.

Example 9.4:

```
x      a1(100,200); //parameterized constructor
x      a2(a1);      //copy constructor is invoked for
                   //object a2 with a1 as parameter
```

4. When an object is passed to a function using pass-by-value, copy constructor is automatically called.

Example 9.5:

```
void test( x )
{
    _____
}

main()
{
    x      a;
    test(a); //copy constructor is invoked
}
```

5. Copy constructor is invoked when an object returns a value.

Example 9.6:

```
class measure
{
    int feet;
    float inches;
    measure sum(measure&);
};
```

```

measure measure::sum(measure &m)
{
    feet=feet+m.feet;
    inches=inches+m.inches;
    if(inches>=12)
    {
        feet++;
        inches=inches-12;
    }
    return measure(feet,inches);
}

```

Program 9.7: To find factorial of a number using copy constructor

```

#include<iostream.h>
class copy
{
    int var;
public:
    copy(int temp)
    {
        var = temp;
    }
    int calculate()
    {
        int fact, i;
        fact = 1;
        for(i = 1; i <= var; i++)
            fact = fact * i;
        return fact;
    }
};
void main()
{
    int n;
    cout<<"Enter the number: ";
    cin>>n;
    copy obj(n);
    copy cpy = obj;
    cout<<"Before copying: "<<n<<"! = "<<obj.calculate()<<endl;
    cout<<"After copying: "<<n<<"! = "<<cpy.calculate()<<endl;
}

```

```
Enter the number: 5
Before copying: 5! = 120
After copying: 5! = 120
```

9.4 Constructor Overloading

Constructors are used to initialize the data members of a class. We can use constructors also to specify the input values for the data members. But the default constructor cannot be used for this purpose. So, we have to overload the default constructor. Overloading a constructor means specifying additional operation by passing the required arguments. Depending on the requirement one can define any number of overloaded constructors in a single class. Depending on the type and number of arguments passed, the compiler decides which version of the constructor to invoke during object creation. The following example has overloaded constructor that is used to specify the input values for the data members of the class.

Program 9.8: To find the simple interest

```
#include<iostream.h>
class simpleinterest
{
private:
    float p, r, t, si;
public:
    simpleinterest() //default constructor
    {
    }
    simpleinterest(float x, float y, float z)//parameterized
    {
        //constructor
        p = x;
        r = y;
        t = z;
    }
    void computesi()
    {
        cout<<"Simple interest is :"<< (p * r * t)/100.0;
    }
};
void main()
{
    simpleinterest si1, si2(10000.0, 12.0, 2.0);
    si2.computesi();
}
```

Simple interest is 2400.0

In the above code segment, there are two constructors, one is simpleinterest() and another one is simpleinterest(float x, float y, float z). The first one is the default constructor and the second one is the three-argument constructor. When we overload a constructor in a class, then it is the job of the programmer to define too the default constructor.

9.5 Destructors:

As we have seen, a constructor, the special member function, will be automatically called when an object is defined for a class. In the same way, a destructor will be called automatically when an object is destroyed. Destroying an object means, de-allocating all the resources such as memory that was allocated for the object by the constructor. A destructor is a special member function that will be executed automatically when an object is destroyed. It will have, like constructor, the name same as that of the class but preceded by a tilde (~).

Syntax:

```
class classname
{
    private:
        //data variables
        //method
    public:
        classname();           //constructor
        ~classname();          //destructor
}
```

Example 9.7: When we want to define a destructor for a class, then we can do this as shown below.

```
class counter
{
    private:
        int    counter;
    public:
        counter( )           //Constructor
        {
            counter = 0;
        }
        ~counter()          //Destructor
        { }
};
```

In the above example the object counter is automatically destroyed destructors

- The destructor name is same as that of class. The first character must be tilde (~).

- Destructors do not have a return value. Destructor can never return a value.
- They take no arguments. Therefore destructors cannot be overloaded.
- The most common use of destructors is to de-allocate memory that was allocated for the object by the constructor. Also, they are declared public.

Example 9.8: class account

```

{
    private:
        float balance;
        float rate;
    public:
        account();           //constructor
        ~account();          //destructor
};
```

Program 9.9: To show the use of destructor

```

#include<iostream.h>
#include<conio.h>
#include <iomanip.h>
class num
{
    private:
        int x;
    public:
        num();
        void display();
        ~num();
};
num::num()
{
    cout<<" In constructor: \n";
    x=100;
}
num::~num()
{
    cout<<"in destructor"<<endl;
}
void num::display()
{
    cout<<"Value of x = "<<x<<endl;
}
int main()
{
    clrscr();
```

```

    num a;
    a.display();
    getch();
    return 0;
}

```

In constructor:
 Value of x = 100
 In destructor

In the above program after executing the program, before the control is transferred out of the function main(), the destructor ~string() is invoked for each object a, so that memory allotted for data member is de allotted. The object y and p are deleted using the destructor.

Points to remember:

- A Constructor is a special member function that is executed automatically whenever an object of a class is created.
- Constructors should have either public or protected access.
- The three types of constructors are default constructor, parameterized constructor and copy constructor.
- A constructor that does not take any arguments is a default constructor.
- A constructor that takes one or more arguments is called parameterized constructor.
- Parameterized constructor can be invoked by explicit or implicit call.
- Copy constructor is a parameterized constructor using which one object can be copied to another object.
- Constructor overloading means passing arguments to the constructor.
- Destructors are member functions that destroy an object automatically.

One mark questions:

1. What is a constructor?
2. Write one reason which defines the need to use a constructor.
 Simplify
3. What should be the access parameters for constructor declaration?
4. Can a constructor return a value to a calling function?
5. How many types of constructors are there?
6. What is a default constructor?
7. What is the drawback of default constructor?
8. Is it possible to overload a default constructor?

9. What is a parameterized constructor?
10. Write any one feature of parameterized constructor.
11. Name two methods through which constructors can be invoked.
12. What is an explicit call?
13. What is an implicit call with reference to constructors?
14. When is =used with constructors?
15. What is a copy constructor?
16. Write the syntax for declaration of copy constructor.
17. Can a copy constructor be invoked explicitly?
18. What is meant by constructor overloading?
19. What is a destructor?
20. Which operator is used with destructor?

Two marks questions:

1. What is a constructor? Give an example
2. Why are constructors needed in a program? Justify
3. Write the syntax and example for default constructor.
4. Mention the features of parameterized constructors.
5. Which are the different methods through which constructors are invoked?
6. Write an example to show the use of parameterized constructor through explicit call.
7. When is copy constructor used in a program?
8. Write syntax and example for copy constructor.

Three marks questions:

1. Mention three types of constructors
2. What are the features of default constructors?
3. What are the disadvantages of default constructor?
4. Write short notes for constructor overloading.

Five marks questions:

1. Write the rules for writing a constructor function.
2. Explain default constructor with syntax and example.
3. Explain parameterized constructor with syntax and example.
4. Explain with an example to show how a constructors is invoked explicitly.
5. Explain the features of copy constructor.
6. Explain destructors with syntax and example.

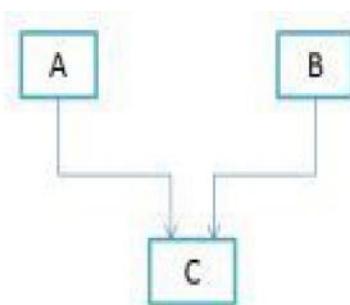
CHAPTER - 10 INHERITANCE

OBJECTIVES

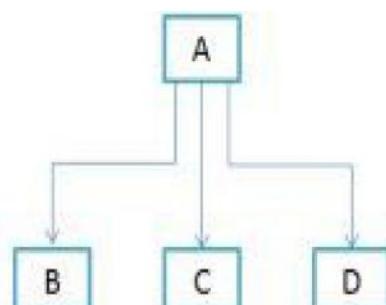
- To understand the concepts of inheritance.
- Usage of inheritance.
- The role of visibility mode.
- Levels of inheritance.
- Concepts of virtual base class.
- Concepts of abstract class.
- Constructors and destructors in derived class.



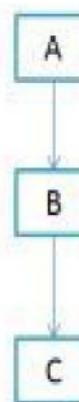
(a) Single Inheritance



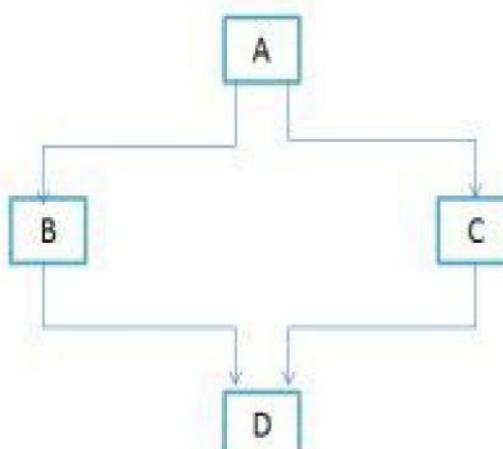
(b) Multiple Inheritance



(c) Hierarchical Inheritance



(d) Multilevel Inheritance



(e) Hybrid Inheritance

10.1 Introduction:

Inheritance is another important aspect of object oriented programming. C++ supports this concept. C++ classes can be used in several ways. This is basically done by creating new classes, reusing the properties of existing one.

C++ allows the user to create a new class (derived class) from an existing class (base class). The derived class inherits all features from a base class and it can have additional features of its own.

Inheritance is the capability of one class to inherit properties from another class.

10.2 Base Class:

It is the class whose properties are inherited by another class. It is also called Super Class.

10.3 Derived Class:

It is the class that inherits properties from base class (es). It is also called Sub Class.

Inheritance has following advantages:

1. Reusing existing code
2. Faster development time
3. Easy to maintain
4. Easy to extend
5. Memory utilization

The following example illustrates the needs of inheritance.

Suppose X is a class already defined and we need to redefine another class Y having same properties of X and in addition to its own. Suppose if we use direct option without using inheritance, it has following problems.

1. Code written in X is repeated again in Y which leads to unnecessary wastage of memory space.
2. Testing to be done separately for both class X and class Y leads to wastage of time.

The above problems can be solved by using the concept of inheritance.

If we reuse the code of X even in Y without rewriting it. The class Y inherits all the properties of X. The class X is called Base class and the class Y is called derived class.

10.3.1 Defining derived classes

When you declare a class, you can indicate what class it derives from by writing a colon after the class name, the type of derivation (public or private or protected) and the class from which it derives.

```
class derived_class_name : visibility_mode base_class_name
{
    // .....
    //members of the derived class
};
```

Where,

class	→	keyword
derived_class_name	→	Name of the derived class
:	→	shows the derivation from the base class
visibility mode	→	Specifies the type of derivation
base_class	→	Name of the base class

The body of the derived class contains its own data members and member function. Like base class definition, the derived class definition must be terminated by a semicolon.

If no visibility mode is specified, then by default the visibility mode is considered private.

Note that all members of the class except private are inherited. Following are some examples of derived class definition.

10.3.2 public derived class

```
class father      // Base class
{
    private:
        char   name[50];
        int    age;
    public:
        char   caste[50];
        int    boys;
        int    girls;
        void   readdata();
        void   printdata();
};

class son: public father           //public derived class
{
    private:
        char company[50];
        float salary;
    public:
        void getdata();
        void display();
};
```

Similarly we can write private derived class.

10.3.3 private derived class

```
class son: private father      // private derived class
{
    private:
        char   company[50];
        float  salary;
    public :
        void   getdata();
        void   display();
};
```

Similarly we can write protected derived class as:

10.3.4 protected derived class

```
class son: protected father     // protected derived class
{
    private:
        char   company[50];
        float  salary;
    public :
        void   getdata();
        void   display();
};
```

10.4 Visibility mode

The visibility mode (private, public and protected) in the definition of the derived class specifies whether features of the base class are privately derived or publicly derived or protectedly derived. The visibility mode basically controls the access specifier to be for inheritable member of base class in the derived class.

The role of visibility modes:

Base class	Derived class		
	Public mode	Private mode	Protected mode
Private	Not inherited	Not inherited	Not inherited
Public	Public	Private	Private
Protected	Protected	Private	Private

10.4.1 Public Inheritance

This is the most used inheritance mode. In this

- The public members of a base class become public members of the derived class.

- The private members of a base class can not be inherited to the derived class.
- The protected members of a base class stay protected in a derived class.

```
class subclass: public superclass
class student                                //base class
{
    private:
        int rollno;
        char name[50];
        float per;
    public:
        void input();
        void output();
};

class comp: public student                     //publicly derived class
{
    private:
        int marks;
    public:
        void read();
        void write();
};
```

10.4.2. Private Inheritance

- The public members of a base class become the private members of the derived class.
- The private members of a base class cannot be inherited to the derived class.
- The protected members of a base class stay protected in a derived class.

10.4.3. Protected Inheritance

- The public members of a base class become protected in a derived class.
 - The private members of a base class cannot be inherited to the derived class.
 - The protected members of a base class stay protected in a derived class.
- class subclass : **protected** superclass

10.5 Levels of Inheritance

A derived class extends its features by inheriting some or all the properties from its base class and adding new features of its own. While inheriting, the derived class can share properties from:

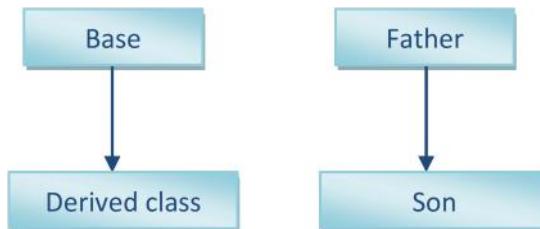
- Only one class
- More than one class
- More than one level

Based on this relationship, inheritance can be classified into five forms.

1. Single inheritance
2. Multilevel inheritance
3. Multiple inheritance
4. Hierarchical inheritance.
5. Hybrid inheritance

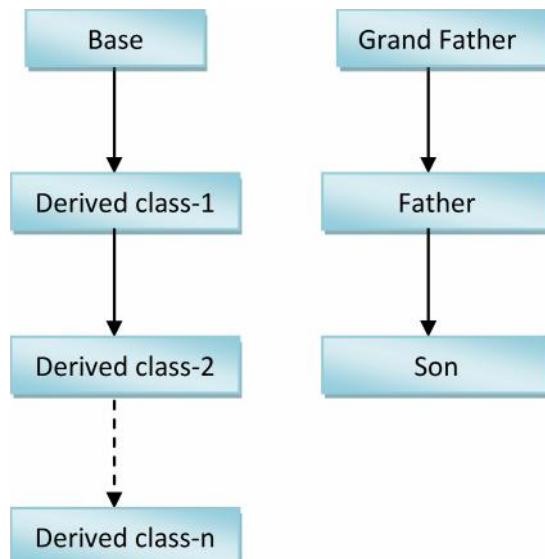
10.5.1 Single Inheritance

If a class is derived from a single base class, it is called as single inheritance.



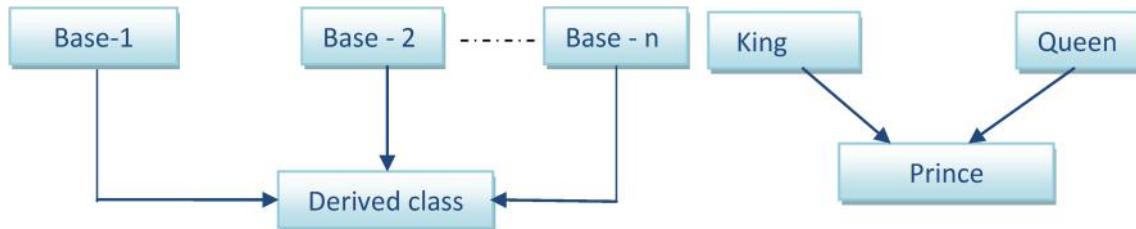
10.5.2 Multilevel Inheritance

The classes can also be derived from the classes that are already derived. This type of inheritance is called multilevel inheritance.



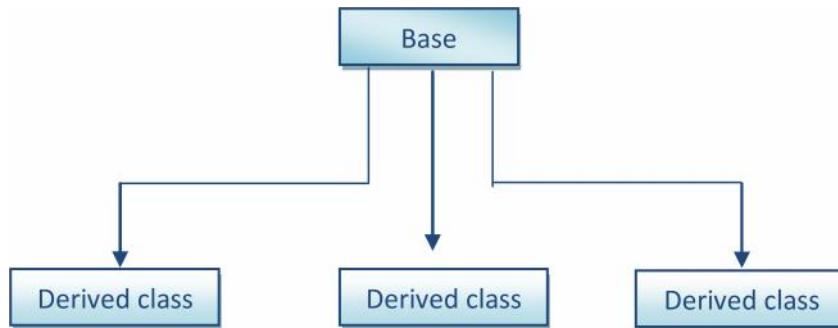
10.5.3 Multiple Inheritance

If a class is derived from more than one base class, it is known as multiple inheritance.

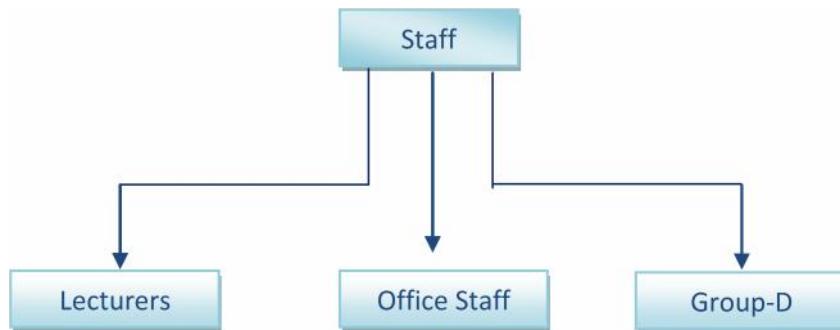


10.5.4 Hierarchical Inheritance

If a number of classes are derived from a single base class, it is called as hierarchical inheritance.

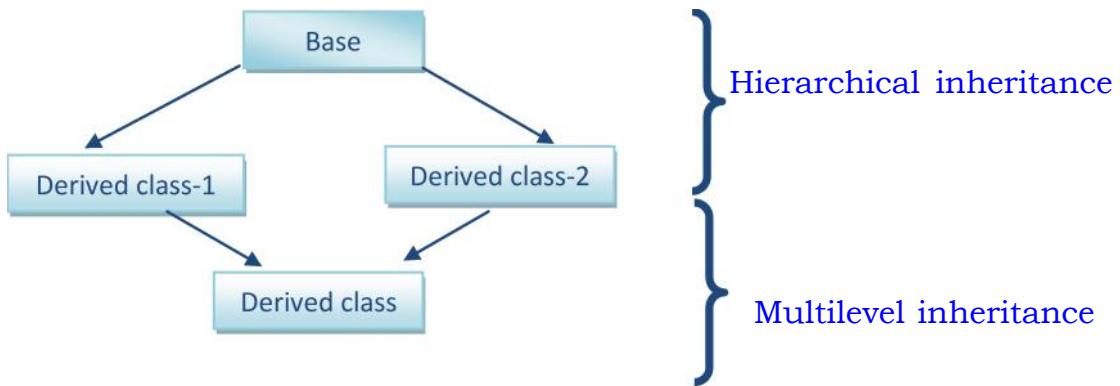


Example for Hierarchical Inheritance



10.5.5 Hybrid Inheritance

Hybrid Inheritance is combination of Hierarchical and Multilevel Inheritance.



Program to illustrate single level inheritance

```

#include<iostream.h>
#include<conio.h>
class base
{
private:
    int rollno;
    char name[20];
public:
    void read()
    {
        cout<<"Enter Roll No and name "<<endl;
        cin>> rollno >>name;
    }

    void display()
    {
        cout<<"roll no: "<< rollno <<endl;
        cout<<"name : "<<name <<endl;
    }
};

class derive: public base
{
private:
    int m1;
    int m2;
    int t;
public:
    void read1()
    {
        cout<<"enter first marks and second marks: "<<endl;
    }
};
  
```

```

        cin>>m1>>m2;
        t = m1+m2;
    }
    void display1()
    {
        cout<<"First marks = "<<m1<<endl;
        cout<<"Second marks = "<<m2 <<endl;
        cout<<"Total marks = "<<t<<endl;
    }

};

void main()
{
    derive ob;
    clrscr();
    ob.read();
    ob.read1();
    ob.display();
    ob.display1();
    getch();
}

```

Enter Roll No and name
1234 Ravindra
Enter first marks and second marks: 100 100
Roll no : 1234
Name : Ravindra
First marks = 100
Second marks = 100
Total marks = 200

10.6 Relationship between classes

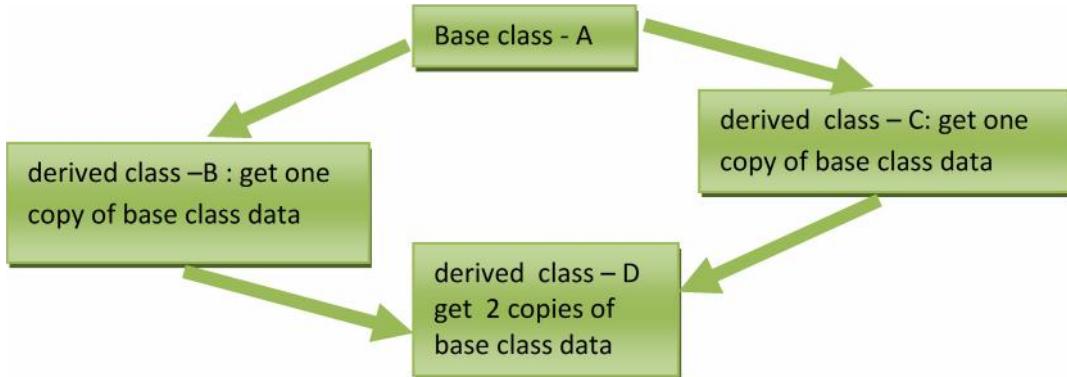
10.6.1. Virtual base classes:

Consider a situation where the program design would require one base class (call it A) and two derived classes namely B and C, which are inherited from the base class A. Further, derived class D is created from B and C. See the figure.

In the public inheritance, B and C inherit one copy of the base class data, whereas derived class D inherited from B and C get two copies of the base class data.

Now suppose a member function of D now wants to access the data members of the base class an ambiguity problem of which of the two copies is to access will

be encountered. The compiler will generate an error message for this ambiguity. To overcome this problem, the derived class B and C should be declared as virtual.



When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class. This can be achieved by preceding the base class name with the word **virtual**.

```

Example: class A
{
    _____
};

class B: virtual public A
{
    _____
};

class C: virtual public A
{
    _____
};

class D: public B, public C
{
    _____
};
  
```

10.6.2 Abstract classes

An abstract class is one that is not used to create objects. An abstract class is designed only to act as a base class (to be inherited by other classes). It is a design concept in program development and provides a base upon which other classes may be built. In the previous example, the class A is an abstract class since it was not used to create any objects.

10.6.3 Constructors in derived classes

As we know, the constructors play an important role in initializing objects. We did not use them earlier in the derived classes for the sake of simplicity. One important thing to note here is that, as long as no base class constructor takes any arguments, the derived class need not have a constructor function. However, if any base class contains a constructor with one or more arguments, then it is mandatory for the derived class to have a constructor and pass the arguments to the base class constructors. Remember, while applying inheritance we usually create objects using the derived class. Thus, it makes sense for the derived class to pass arguments to the base class constructor. When both the derived and base classes contain constructors, the base constructor is executed first and then the constructor in the derived class is executed.

In case of multiple inheritances, the base classes are constructed in the order in which they appear in the declaration of the derived class. Similarly, in a multilevel inheritance, the constructors will be executed in the order of inheritance.

Since the derived class takes the responsibility of supplying initial values to its base classes, we supply the initial values that are required by all the classes together, when a derived class object is declared. How are they passed to the base class constructors so that they can do their job? C++ supports a special argument passing mechanism for such situations.

The constructor of the derived class receives the entire list of values as its arguments and passes them on to the base constructors in the order in which they are declared in the derived class. The base constructors are called and executed before executing the statements in the body of the derived constructor.

Example: Use of constructor in single inheritance

```
class base_class
{
protected:
public:
    base_class()      // base class constructor
    {
    }
};
```

```
class derived_class: public base_class
{
protected:
public :
    derived _class() // derived class constructor
    {
    }
};

};
```

10.6.4 Constructors in derived classes

If the constructors are called down the line from the base to the derived class, the destructors are called just in the reverse order. That is from the derived class up to the base class.

Points to remember:

- Inheritance: It is the capability of one class to inherit properties from another class.
- Base Class: It is the class whose properties are inherited by another class. It is also called Super Class.
- Derived Class: It is the class that inherits properties from base class(es). It is also called Sub Class.
- Advantages of inheritance
 - ❖ Reusing existing code.
 - ❖ Faster development time.
 - ❖ Easy to maintain.
 - ❖ Easy to extend.
 - ❖ Memory utilization
- All members of the class except private are inherited.
- The visibility mode (private, public protected) in the definition of the derived class specifies where features of the base class are privately derived or publicly derived or protected derived. The visibility mode basically controls the access specifier to be for inheritable member of base class in the derived class.
- Inheritance can be classified into five forms.
 - ❖ Single inheritance
 - ❖ Multilevel inheritance
 - ❖ Multiple inheritance
 - ❖ Hierarchical inheritance
 - ❖ Hybrid inheritance

- Single Inheritance: If a class is derived from a single base class, it is called as Single Inheritance.
- Multilevel Inheritance: The classes can also be derived from the classes that are already derived. This type of inheritance is called multilevel inheritance.
- Multiple Inheritance: If a class is derived from more than one base class, it is known as multiple inheritance.
- Hierarchical Inheritance: If a number of classes are derived from a single base class, it is called as hierarchical inheritance.
- Hybrid inheritance: Hybrid Inheritance is combination of Hierarchical and Multilevel Inheritance.
- In the publicly derived class, the public and protected members remain public and protected.
- In the privately derived class, the public and protected members of the base class become private members.
- In the protected derived class, the public and protected members of the base class become protected members.
- Virtual base class: When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class. This can be achieved by preceding the base class' name with the word **virtual**.
- An abstract class is one that is not used to create objects. An abstract class is designed only to act as a base class (to be inherited by other classes).

Review questions**One marks questions:**

1. What is inheritance?
2. How to implement inheritance?
3. What is base class?
4. What is derived class?
5. What is public access specifier?
6. What is private access specifier
7. Mention any one advantage of inheritance?
8. Is inheritance possible in c?
9. What is the use of protected access specifier?
10. What is the use of public access specifier?
11. What is the use of private access specifier?
12. What is singe inheritance?
13. What is multilevel inheritance?
14. What is hierarchical inheritance?
15. What is hybrid inheritance?
16. What is multiple inheritance?
17. What is virtual base class?
18. What is an abstract class?
19. When is it necessary to use inheritance?
20. What is visibility mode?

Two marks questions:

1. How to implement inheritance?
2. What is the difference between public and private access specifier?
3. Mention any 2 advantages of inheritance?
4. Mention any 2 types of inheritance?
5. What is the difference between inheritance and polymorphism?
6. What is singe inheritance? Give an example.
7. What is multilevel inheritance? Give an example.
8. What is hierarchical inheritance? Give an example.
9. What is hybrid inheritance? Give an example.
10. What is multiple inheritance? Give an example.
11. What is virtual base class? Give example.
12. What is an abstract class?
13. Which are the components which cannot be inherited?
14. Explain Single Inheritance with a suitable C++ program.
15. Explain the requirements of a virtual base class.
16. When is it necessary to use inheritance?
17. What is visibility mode? What is its role?

18. How does inheritance influence the working of constructors?
19. How does inheritance influence the working of destructors?

Three marks questions:

1. What is the difference between public and private access specifier with respect to inheritance?
2. What are the advantages of inheritance?
3. What are the types of inheritance?

Five marks questions:

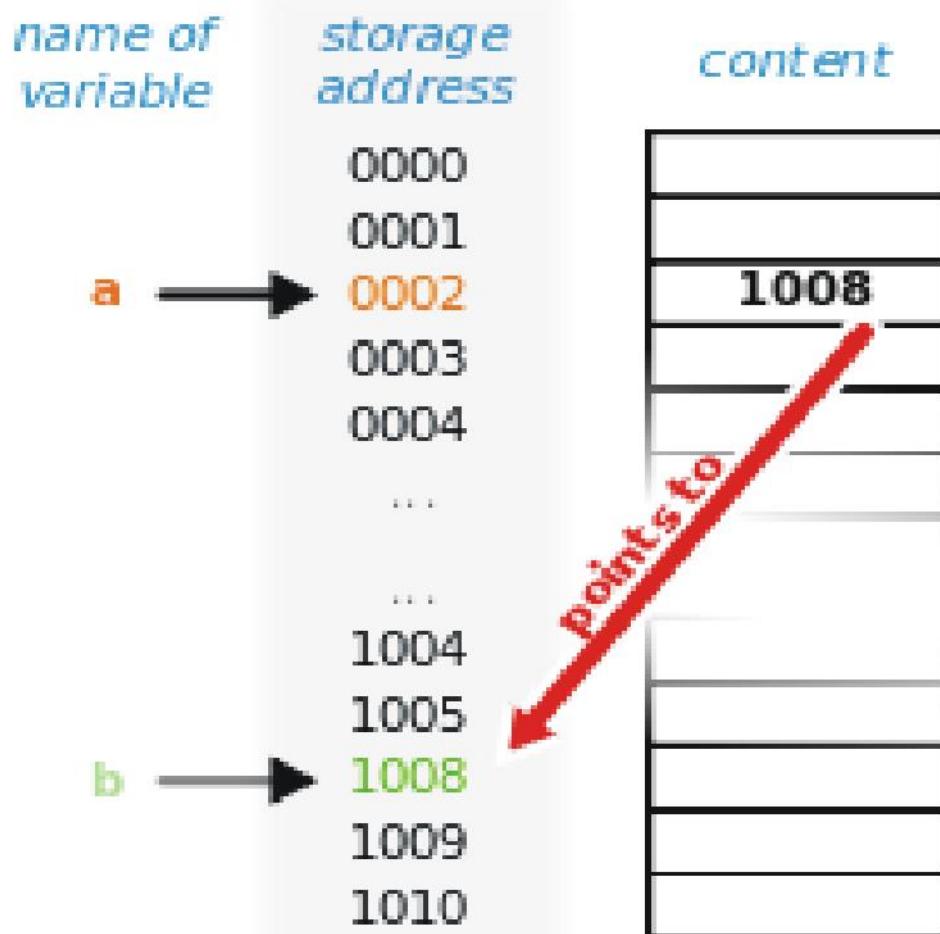
1. What is the difference between public and private and protected access specifier?
2. What are the advantages of inheritance?
3. What are the types of inheritance? Explain any 2.
4. What is virtual base class? Give example.
5. Which are the components which can not be inherited?
6. Explain single inheritance with a suitable C++ program.
7. Explain the requirements of a virtual base class.
8. What is visibility mode? What is its role with respect to inheritance?
9. How does inheritance influence the working of constructors and destructors?

CHAPTER-11

POINTERS

Objectives:

- To understand the concepts of pointers.
- Usage of pointers.
- The role of pointers in array, strings, structures.
- Concepts of dynamic and static allocation of memory.
- Relationship between pointers and functions.
- Relationship between pointers and objects.



I11.1introduction:

When writing a program, you declare the necessary variables that you will need in order to accomplish your work. When declaring variables, you are simply asking the computer to reserve a set amount of space in its memory for a particular object you want to use. When you declare a variable, the computer reserves an amount of space for that variable, and uses the variable's name to refer to that memory space. This will allow you to store the value of that variable, in that space. Indeed, the computer refers to that space using an address. Therefore, everything you declare has an address, just like the address of your house. You can find out what address a particular variable is using.

Pointers are a powerful concept in C++ and have the following advantages.

- It is possible to write efficient programs.
- Memory is utilized properly.
- Dynamically allocate & deallocate memory.
- Easy to deal with hardware components.
- Establishes communication between program and data.

I12.Memory representation of pointers.

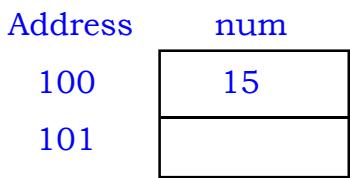
Before understanding the concept of pointers it is necessary to know the memory organization. Memory is organized as an array of bytes. A byte is basic storage and accessible unit in memory. Each byte is identifiable by a unique number called address. Suppose we have 1KB of memory, since $1\text{KB}=1024$ bytes, the memory can be viewed as an array of locations of size 1024 with the subscript range (0 to 1023). 0 represents the address of first location; 1 represents the address of second location; and so on 1023 represents the address of last location.

We know that variables are declared before they are used in a program. Declaration of a variable tells the compiler to perform the following.

- Allocate a location in memory. The number of location depends on data type.
- Establish relation between address of the location and the name of the variable.

Consider the declaration, int num;

This declaration tells the compiler to reserve a location in memory. We know that the size of int type is two byte. So the location would be two bytes wide.



In the above figure, num is the variable that stores the value 15 and address of num is 100. The address of a variable is also an unsigned integer number. It can also be retrieved and stored in another variable.

Pointer:

A pointer is a variable that holds a memory address, usually the location of another variable in memory.

11.3 Declaration and Initialization of pointer:

The general form is, data-type *variable_name;

data-type is any valid data type supported by C++ or any user defined type and variable_name is the name of pointer variable. The presence of * indicates that it is a pointer variable.

Defining a Pointer Variable:

int *iptr;	iptr is declared to be pointer variable of int type.
float *fptr;	fptr is declared to be pointer variable of float type.
char *cptr;	cptr is declared to be pointer variable of character type.

Pointer Variables Assignment:

We can assign the address of a variable to a pointer variable as follows:

```
int num = 25;
int *iptr;
iptr = &num;
```

In the above example, the variable num (=25) is assigned to pointer variable iptr.

11.4 The address-of operator (&):

& is a unary operator that returns the memory address of its operand. For example, if var is an integer variable, then &var is its address. This operator has the same precedence and right-to-left associativity as the other unary operators.

You should read & operator as “the address-of” which means &var will be read as “the address of var”.

Example: int num = 25;

```
int *iptr;
iptr = &num;           // The Address of Operator &
```

11.5 Pointer operator or Indirection Operator (*):

The second operator is Indirection Operator *, and it is the complement of &. It is a unary operator that returns the value of the variable located at the address specified by its operand.

Example:

```
int num = 25;
int *iptr;           //Pointer operator (Indirection Operator *):
iptr = &num;
```

The following program executes the above two operations

```
#include <iostream>
#include <iomanip.h>
void main( )
{
    int var;
    int *ptr;
    int val;

    var = 3000;
    ptr = &var;
    val = *ptr;

    cout << "Value of var: " << var << endl;
    cout << "Value of ptr: " << ptr << endl;
    cout << "Value of val: " << val << endl;
}
```

Value of var: 3000
 Value of ptr: 0xbff64494
 Value of val: 3000

11.6 Pointer Arithmetic:

As you understood, pointer is an address which is a numeric value. Therefore, you can perform arithmetic operations on a pointer just as you can on a numeric value.

There are four arithmetic operators that can be used on pointers: ++, —, +, and . (dot operator).

Following operations can be performed on pointers.

- We can add an integer value to a pointer.
- We can subtract an integer value from a pointer,

- We can compare two pointers, if they point the elements of the same array
- We can subtract one pointer from another pointer if both point to the same array.
- We can assign one pointer to another pointer provided both are of same type.

Following operations cannot be performed on pointers.

- Addition of two pointers.
- Subtraction of one pointer from another pointer when they do not point to the same array.
- Multiplication of two pointers.
- Division of two pointers.

Example:

- a. Suppose if p is an integer pointer then $p++$ will increment p by 2 bytes. Each time a pointer is incremented by 1, it points to the memory location of the next element of its base type.
- b. Suppose if p is a char pointer then $p++$ will incremented p by 1-byte.
- c. $p--$ each time a pointer is decremented by 1, it points to the memory location of the previous element of its base type.
- d. $p=p + \text{integer value.}$
 $p=p - \text{integer value.}$

11.7 Pointers and Arrays:

There is a close relationship between arrays and pointers in C++.

Consider the declaration.

int a[6];

The elements of the array can be referred to in the program as $a[0]$, $a[1]$, ..., $a[9]$. When the program is compiled, the compiler does not save the addresses

of all the elements, but only the address of the first element, $a[0]$. When the program needs to access any element, $a[i]$, it calculates its address by adding i units to the address of $a[0]$. The number of bytes in each “unit” is, in our example, equal to the `sizeof(int)`. i.e., 2. In general, it is equal to the number of bytes required to store an element of the array.

The address of $a[0]$ can be explicitly obtained using the & (address-of) operator. i.e., `&a[0]`. Since the data type of $a[0]$ is int, the data type of `&a[0]` is, as usual, `int*` (pointer to int).



C++ allows us to use the name of the array a, without any subscript, as another name for &a[0].

The following example shows the relationship between pointer and one-dimensional array.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main( )
{
    int    a[10], i, n;
    cout<<"How many elements? ";
    cin>>n;
    cout<<"Enter array elements: ";
    for(i=0; i<n; i++)
        cin>>*(a+i);
    cout<<"The given array elements are ";
    for(i=0; i<n; i++)
        cout<<setw(4)<<*(a+i);
    getch();
}
```

```
How many elements? 5
Enter array elements: 1 2 3 4 5
The given array elements are 1 2 3 4 5
```

11.8 Array of pointers:

As we know that there is an array of integers, array of float, similarly, there can be an array of pointers. Since we know that pointer is a variable which stores address of another variable, an array of pointers means that it is a collection of addresses.

The example below shows the array of pointers.

```
int    *iptr[5];
int    i=10, j=20, k=30, l=40, m=50;

iptr[0] = &i;      *iptr[0] = 10;
iptr[1] = &j;      *iptr[1] = 20;
iptr[2] = &k;      *iptr[2] = 30;
iptr[3] = &l;      *iptr[3] = 40;
iptr[4] = &m;      *iptr[4] = 50;
```

11.9 Pointers and Strings:

We have already discussed that there is a close relationship between array and pointers. Similarly there is also a close relationship between strings and pointers in C++. String is sequence of characters ends with null ('\0') character. Suppose we have declared an array of 5 elements of the data type character.

```
char s[5];
char *cptr;
cptr = s;
```

Here, s is array of characters (strings). cptr is character pointer to string. s also represents character pointer to string.

The elements of the array can be referred to in the program as s[0], s[1], ..., s[5]. When the program is compiled, the compiler does not save the addresses of all the elements, but only the name of the array. Here, s gives the base address of the array. i.e., the address of the first character in the string variable and hence can be regarded as pointer to character. Since we know that string always end with null character, it is enough for us to know the starting address of a string to be able to access entire string. The number of bytes allocated for a string is determined by the number of characters within string.

Let us now consider a string constant "HELLO". s is pointer to the memory location where 'H' is stored. Here, s can be viewed as a character array of size 6, the only difference being that a can be reassigned another memory location.

```
char s[5] = "Hello";
```

Here, s gives address of 'H'.

- *a gives 'H'
- a[0] gives 'H'
- a++ gives address of 'E'
- *a++ gives 'E'

H	E	L	L	O	\0
s[0]	s[1]	s[2]	s[3]	s[4]	s[5]

11.10 Pointers as Function Parameters.

A pointer can be a parameter. It works like a reference parameter to allow change to argument from within the function.

```
void swap(int *x, int *y)
{
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
}
swap(&num1, &num2);
```

11.11 Pointers and Structures

We can create pointers to structure variables.

```
struct student
{
    int    rollno;
    float  fees;
};

student    s;
student    *sp = &s;
(*sp).rollno = 104;
```

The above statements can be written using the operator -> as

```
ptr -> member;
sp -> rollno = 104;
```

11.12. Memory allocation of pointers (Dynamic and Static)

The compiler allocates the required memory space for a declared variable. For example, integer variable it reserves 2- bytes, float variable it reserves 4-bytes, character variable it reserves 1-byte and so on. Therefore every data and instruction that is being executed must be allocated some space in the main or internal memory. Memory allocation is done in two ways:

- Static allocation of memory
- Dynamic allocation of memory

11.12.1 Static allocation of memory

In the static memory allocation, the amount of memory to be allocated is predicted and pre known. This memory is allocated during the compilation itself. All the variables declared normally, are allocated memory statically.

Example: int a; //Allocates 2 bytes of memory space during the
 //compilation time.

11.12.2 Dynamic allocation of memory (new and delete)

In the dynamic memory allocation, the amount of memory to be allocated is not known. This memory is allocated during run-time as and when required.

C++ supports dynamic allocation and deallocation of objects using the **new** and **delete** operators. These operators allocate memory for objects from a pool called the **free store**. The new operator calls the special function operator **new** and the delete operator calls the special function operator **delete**.

We can allocate storage for a variable while program is running by using new operator. Dynamic allocation is perhaps the key to pointers. It is used to allocate memory without having to define variables and then make pointers

point to them. Although the concept may appear confusing, it is really simple. The following codes demonstrate how to allocate memory for different variables.

To allocate memory of type integer, int *iptr = new int;

```
int *pNumber;  
pNumber = new int;
```

The first line declares the pointer, pNumber. The second line then allocates memory for an integer and then makes pNumber point to this new memory. Here is another example, this time using a double:

```
double *pDouble;  
pDouble = new double;
```

To allocate memory for array, double *dptr = new double[25];

To allocate dynamic structure variables or objects,

```
student sp = new student; //student is tag name of structure
```

The formula is the same every time, so you can't really fail with this bit. What is different about dynamic allocation, however, is that the memory you allocate is not deleted when the function returns, or when execution leaves the current block. So, if we rewrite the above example using dynamic allocation, we can see that it works fine now:

```
#include<iostream.h>  
#include<conio.h>  
#include<iomanip.h>  
int *p;  
void SomeFunction()  
{  
    // make p pointer point to a new integer  
    p = new int;  
    *p = 25;  
}  
  
void main()  
{  
    SomeFunction(); // make pPointer point to something  
    cout<<"Value of *p: "<<*p;  
}
```

Output Value of *p: 25

When SomeFunction is called, it allocates some memory and makes p point to it. This time, when the function returns, the new memory is left intact, so p still points to something useful.

delete pointer:

Memory that is dynamically allocated using the NEW operator can be freed using delete operator. The delete operator calls the operator delete function, which frees memory back to the available pool.

Releasing Dynamic Memory

Use delete function to free dynamic memory as: delete iptr;

To free dynamic array memory, delete [] dptr;

To free dynamic structure, delete student;

Static allocation of memory

Memory is allocated before the execution of the program begins.
(During Compilation)

No memory allocation or deallocation actions are performed during Execution.

Variables remain permanently allocated.

Implemented using stacks and heaps.

11.13 Free store(heap memory)

Free store is a pool of unallocated memory heap given to a program that is used by the program for dynamic allocation during execution.

11.14 Memory Leak

If the objects, that are allocated memory dynamically, are not deleted using delete, the memory block remains occupied even at the end of the program. Such memory blocks are known as orphaned memory blocks. These orphaned memory blocks when increase in number, bring adverse effect on the system. This situation is called memory leak.

11.15 Self Referential Structure

The self referential structures are structures that include an element that is a pointer to another structure of the same type.

Dynamic allocation of memory

Memory is allocated during the execution of the program.

Memory Bindings are established and destroyed during the Execution.

Allocated only when program unit is active.

Implemented using data segments.

```
struct node
{
    int data;
    struct node *next;
};
```

11.16 Pointers and functions

A function is named unit of a group of program statements designed to perform a specific task and returns single value. There is a close relationship between pointers and functions. We know that a function uses arguments in order to carry its assignment. The arguments are usually provided to the function. When necessary, a function also declares its own variable to get the desired return value. Like other variables, pointers can be provided to a function, with just a few rules. When declaring a function that takes a pointer as an argument, make sure you use the asterisk for each argument. When calling the function, use the references to the variables. The function will perform its assignment on the referenced variable(s). After the function has performed its assignment, the changed value(s) of the argument(s) will be preserved and given to the calling function. To pass pointer arguments, use the asterisks when declaring the function and use the ampersand (&) when calling the function.

Invoking of function can be done by following two methods:

- By passing the references.
- By passing the pointers.

11.16.1. Invoking functions by passing the references

When parameters are passed to the functions by reference, the formal parameters become reference (or aliases) to the actual parameters in the calling function. This means that invoking the called function does not create its own copy of original values, rather than, it refers to the original values by different names i.e., their references. Thus the called function works with the original data and any change in the values gets reflected to the data.

The call by reference method is useful in situation where the values of the original variable are to be changed using a function. Say, for instance a function is to be invoked that swap two variables that are passed by references. The following example program explains it.

Program to swap the values of two variables using pass-by-reference method:

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

```

void swap(int &, int &);
int a=10, b=30;

cout<<"Original values: ";
cout<<"a = "<<a<<" and b = "<<b<<endl;
swap(a,b);
cout<<"values after swapping: ";
cout<<"a = "<<a<<" and b = "<<b<<endl;
getch();
}

void swap( int &x , int &y)
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}
void swap( int &x , int &y)
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}

```

Original values: a = 10 and b = 30

Values after swapping: a = 30 and b = 10

In the above program the function swap() creates reference x for first incoming integer and reference y for second incoming integer. Thus the original values are worked with, but by using the names x and y. Notice that the, function call statement is simple one. i.e., swap(a, b);

But the function declaration (prototype) and definition include the reference symbol &. The function declaration and definition, both start as: void swap(int &x , int &y)

Therefore, by passing the references the function works with the original values (i.e., the same memory area in which original values are stored) but in case alias names to refer to them. Thus the values are not duplicated. The same happens when pointers are passed but in different manner.

11.16.2. Invoking functions by passing the pointers

When the pointers are passed to the function, the addresses of actual arguments in the calling function are copied into formal arguments of the called function. This means that using formal arguments (the addresses of original

values) in the called function, we can make changes into the actual arguments of the calling function, therefore here also, the called function does not create own copy of original values rather, it refers to the original values by the addresses (passed through pointers) it receives.

To swap two values, we have seen how the passing references method works. The same can be achieved by passing addresses through pointers. The following example program explains it.

Program to swap values of two variables using pass by references method:

```
#include <iostream.h>
#include <conio.h>
void main()
{
    void swap(int *x int *y);
    int a=10,b=30;
    cout<<"Original values: ";
    cout<<"a = "<<a<<" and b="<<b<<endl;
    swap(&a, &b);
    cout<<"values after swapping: ";
    cout<<"a = "<<a<<" and b = "<<b<<endl;
    getch();
}
void swap( int *x , int *y)
{
    int temp;
    temp= *x;
    *x = *y;
    *y = *temp;
}
```

Original values: a = 10 and b = 30

Values after swapping: a = 30 and b = 10

The above program invokes swap() by passing addresses of a and b. i.e., swap(&a , &b); Here, &a and &b pass the addresses of a and b respectively.

The function definition receives the incoming addresses in corresponding pointers x and y. Notice the function declaration.

```
void swap( int *x, int *y);
```

Thus, we have seen that using call-by-reference, in both ways, we are able to return more than one value at a time (we sent back changed values of two

variables a and b), which are not possible using ordinary return statement. A return statement can return only one value from a function at a time.

11.17 Memory Comes, Memory Goes

There's always a complication and this one could become quite serious, although it's very easy to remedy. The problem is that although the memory that you allocate using dynamic allocation is conveniently left intact, it actually never gets deleted automatically. That is, the memory will stay allocated until you tell the computer that you've finished with it. The upshot of this is that if you don't tell the computer that you've finished with the memory, it will be wasting space that other applications or other parts of your application could be using. This eventually will lead to a system crash through all the memory being used up, so it's pretty important, freeing the memory when you've finished with it is very simple:

11.18 Pointers and objects

As we know that there is pointer to variables, pointer to strings, pointer to structures, similarly there is pointer to objects. The pointers pointing to objects are referred to as object pointers.

Declaration of pointers to objects

```
class_name *object-pointer;
```

Here, class_name is the name of an already defined class and object-pointer is the pointer to an object of this classtype.

Example: employee *eptr;

Here, employee is an already defined class. When accessing members of a class using an object pointer, the arrow operator (->) is used instead of dot (.) operator.

The following program illustrates how to access an object given a pointer to it.

```
#include<iostream.h>
#include <iomanip.h>
#include<conio.h>
class emp
{
    private:
        int    empno;
        char   name[20];
        float  salary;
    public:
        void   get();
        void   display();
```

```
};

void emp::get()
{
    cout<<“Enter employee number: “;
    cin>>empno;
    cout<<“Enter employee name: “;
    cin>>name;
    cout<<“Enter employee salary: “;
    cin>>salary;
}
void emp::display()
{
    cout<<“Employee number: “<<empno<<endl;
    cout<<“Employee name: “<<name<<endl;
    cout<<“Employee salary: “<<salary;
}
void main()
{
    emp e, *ep;

    ep = &e;
    clrscr( );
    ep->get( );
    ep->display( );
    getch( );
}
```

```
Enter employee number: 2505
Enter employee name: Harshini
Enter employee salary: 6500.00
Employee number: 2505
Employee name: Harshini
Employee salary: 6500.00
```

The given program is self referential. Here, *ep is pointer to an object.

11.19 this pointer

Every object in C++ has access to its own address through an important pointer called this pointer. The this pointer is an implicit parameter to all member functions. Therefore, inside a member function, this may be used to refer to the invoking object.

Friend functions do not have a this pointer, because friends are not members of a class. Only member functions have this pointer.

Points to Remember:

- Pointers are a powerful concept in C++ and have following advantages.
 - ❖ It is possible to write efficient programs
 - ❖ Memory is utilized properly
 - ❖ Dynamically allocate & de allocate-memory
 - ❖ Easy to deal with hardware components
 - ❖ Establishes communication between program and data
- A pointer is a variable that holds a memory address, usually the location of another variable in memory.
- Following operations that can be performed over pointers.
 - ❖ We can add an integer value to a pointer.
 - ❖ We can subtract an integer value from a pointer,
 - ❖ We can compare two pointers. if they point the elements of the same array
 - ❖ We can subtract one pointer from another pointer if both point to the same array.
- Following operations that cannot be performed over pointers.
 - ❖ Addition of two pointers
 - ❖ Subtraction of one pointer from another pointer when they do not point to the same array
 - ❖ Multiplication of two pointers
 - ❖ Division of two pointers
- There is a close relationship between arrays and pointers in C++.
- C++ allows us to use the name of the array a , without any subscript, as another name for &a[0].
- Array of pointers means that it is a collection of address.
- There is also a close relationship between strings and pointers in C++.
- A pointer can be a parameter. It works like a reference parameter to allow change to argument from within function
- We can create pointers to structure variables
- In the static memory allocation, the amount of memory to be allocated is predicted and pre known.

- In the dynamic memory allocation, the amount of memory to be allocated is not known. This memory is allocated during run-time as and when required.
- We can allocate storage for a variable while program is running by using new operator.
- Use delete to free dynamic memory.
- Free store is a pool of unallocated heap memory given to a program that is used by the program for dynamic allocation during execution.
- Memory leak: If the objects, that are allocated memory dynamically, are not deleted using delete, the memory block remains occupied even at the end of the program. Such memory blocks are known as orphaned memory blocks. These orphaned memory blocks when increase in number, bring adverse effect on the system. This situation is called memory leak
- The self referential structures are structures that include an element that is a pointer to another structure of the same type.
- There is a close relationship between pointers and functions. We know that a function uses arguments in order to carry its assignment.
- Invoking of function can be done by following two methods.
 - By passing the references
 - By passing the pointers
- The pointers pointing to objects are referred to as object pointers.
- Every object in C++ has access to its own address through an important pointer called this pointer.

Review Questions

One marks questions.

1. What do you mean by pointer?.
2. Mention any one advantage of pointer?
3. What is address operator?
4. What is pointer operator?
5. How to declare pointer?
6. How to initialize pointer?
7. What is static memory?
8. What is dynamic memory?
9. What is free store?
10. Write a definition for a variable of type pointer to float.
11. What is new operator in C++?
12. What is delete operator in C++?

Two marks questions.

1. What do you mean by pointer? Explain with example.
2. Mention any 2 advantages of pointer?
3. What is address operator? Give example.
4. What is pointer operator? Give example.
5. How to declare pointer? Give example.
6. How to initialize pointer? Give example.
7. What is static memory?
8. What is dynamic memory?
9. What is free store?
10. Illustrate the use of “self referential structures” with the help of example.
11. What is new operator in C++?
12. What is delete operator in C++?
13. What is array of pointers? Give example.

Three marks questions:

1. What are the advantages of pointer?
2. How dynamic memory allocation is different from static memory allocation.
3. What is new operator in C++? Give example.
4. What is delete operator in C++? Give example.
5. Show the general form new and delete operator in C++?
6. What is array of pointers? Give example.
7. What is the relationship between array and pointers? Give example.
8. What is the relationship between string and pointers? Give example.
9. What is the relationship between structures and pointers? Give example.
10. What is the relationship between object and pointers? Give example.

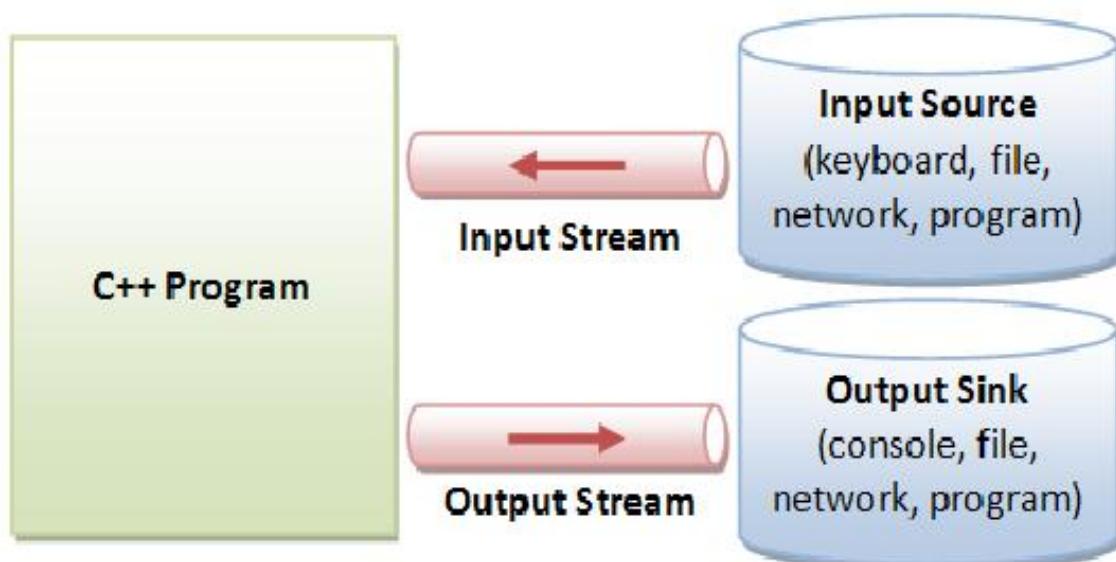
Five marks questions:

1. Show the general form new and delete operator in C++?
2. What is the relationship between object and pointers? Give example.
3. Explain with example by passing the reference.
4. Explain with example by passing the pointers.

CHAPTER 12

DATA FILE HANDLING**OBJECTIVES**

- To understand the concepts of files.
- Usage of types of files.
- The role of text and binary files.
- Concept of opening and closing of files
- Concept of input and output operations in text files.
- Concept of input and output operations in binary files.
- Concept of file pointers and their manipulations.

**Internal Data Formats:**

- Text: `char`, `wchar_t`
- `int`, `float`, `double`, etc.

External Data Formats:

- Text in various encodings (US-ASCII, ISO-8859-1, UCS-2, UTF-8, UTF-16, UTF-16BE, UTF16-LE, etc.)
- Binary (raw bytes)

12.1 Introduction:

We have already used the `cin` and `cout` for handling the input and output operations. They are used to accept inputs through keyboard and display outputs on the screen. C++ provides a rich set of operations for both unformatted and formatted I/O operations. In C++, these IO operations are implemented through `iostream` library.

The C++ standard libraries provide an extensive set of input/output capabilities which we will see in subsequent topics. This chapter will discuss very basic and most common I/O operations required for C++ programming.

C++ I/O occurs in streams, which are sequences of bytes. If bytes flow from device like a keyboard, a disk drive, or a network connection etc. to main memory, this is called **input operation** and if bytes flow from main memory to a device like a display screen, a printer, a disk drive, or a network connection etc., this is called **output operation**.

Most computer programs work with files. This is because files help in storing information permanently. Word processors create document files. Database programs create files of information. Compilers read source files and generate executable files. So, we see, it is the files that are mostly worked with, inside programs. A file itself is a bunch of bytes stored on some storage device like tape or magnetic disk etc.

In C++, file input/output facilities are implemented through a header file of C++ standard library. This header file is `fstream.h`.

In C++, a file, at its lowest level, is interpreted simply as a sequence, or stream of bytes. In C++, file I/O library manages the transfer of these bytes. At this level, the notion of a data type is absent.

On the other hand, file, at user level, consists of a sequence of possibly intermixed data types-characters, arithmetic values and class objects.

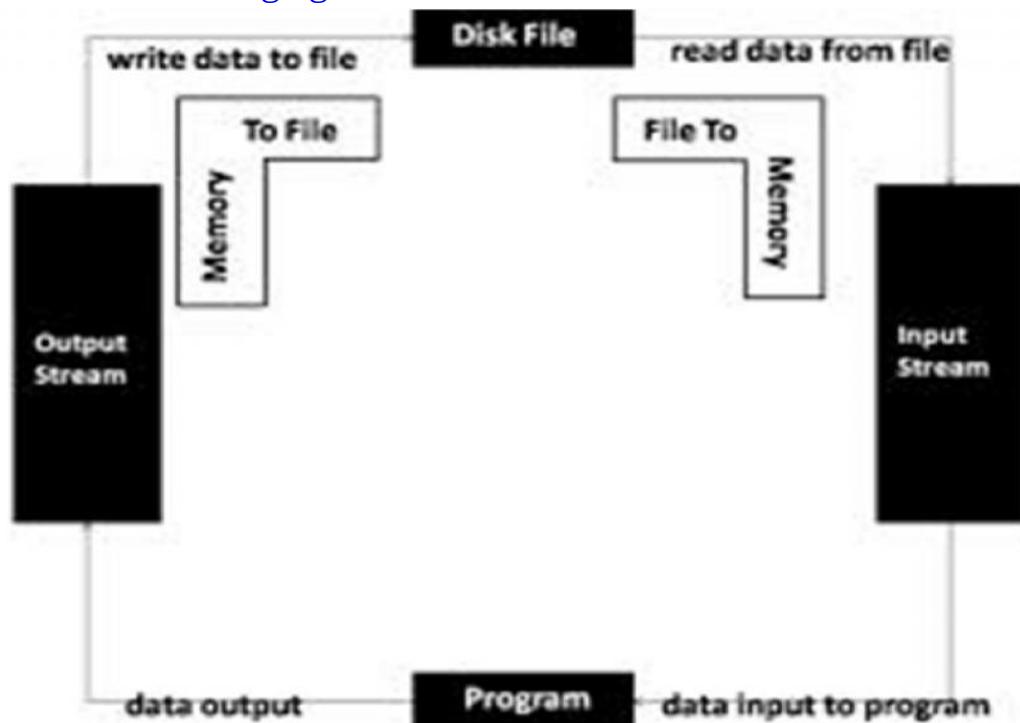
The `fstream` library predefines a set of operations for handling files related input and output. It defines certain classes that help to perform file input and output. For example, `ifstream` class ties a file to the program for input. `ofstream` class ties a file to the program for output and `fstream` classifies a file to the program for both input and output. File manipulation and related operations using streams are the topics we are going to discuss in this chapter.

12.2 fstream.h header file

The C++ input/output operations are very much similar to the console input and output operations. The file operations also make use of streams as an interface between the programs and the files.

A stream is a general name given to a flow of data at the lowest level; data is just the binary data without any notion of data type. Different streams are used to represent different kinds of data flow such as whether data is flowing into the memory or out of the memory. Each stream is associated with a particular class, which contains member functions and definitions for dealing with that particular kind of data flow. For example, the ifstream class represents input disk files.

The stream that supplies data to the program is known as **input stream**. It reads the data from the file and hand it over to the program. The stream that receives data from the program is known as **output stream**. It writes the received data to the file. Following figure shows it.



The information / data stored under a specific name on a storage device, is called a file.

Stream refers to a sequence of bytes

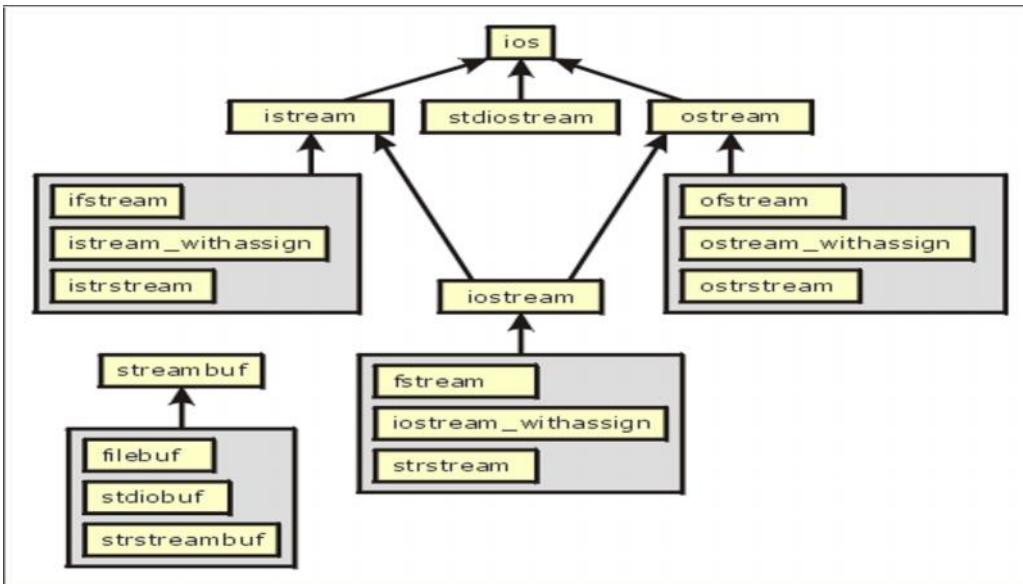


fig: Stream class hierarchy for I/O operations:

The `fstream` base is derived from the basic class `ios`. The class `ifstream` is derived from both `istream` and `fstream` base and similarly, the class `ofstream` is derived from both the `ostream` and `fstream` base. Both `ifstream` and `ofstream` act as base classes for `fstream` class. The class `filebuf` is derived from class `streambuf`. The complete class hierarchy is shown above. Central to file handling are three classes. They are `ifstream`, `ofstream` and `fstream`.

That is why, if you include `fstream.h` file in your file handling program, you need not to include `iostream.h` file as classes of `fstream.h` inherit from `iostream.h` only. The functions of these classes have summarized in the below table.

12.2.1 Classes for file stream operation

Classes	Meanings
<code>filebuf</code>	It sets the file buffers to read and write.
<code>fstreambase</code>	It provides the facilities for file operations and consists of <code>open()</code> and <code>close()</code> member functions. This is base class for <code>fstream</code> , <code>ifstream</code> , <code>ofstream</code> .
<code>ifstream</code>	Stream class to read from files. It provides input operations for file. It inherits the function <code>get()</code> , <code>getline()</code> , <code>read()</code> and functions supporting random access (<code>seekg()</code> and <code>tellg()</code>) from <code>istream</code> class defined inside <code>iostream.h</code> file.
<code>ofstream</code>	Stream class to write on files. It provides output operations for file. It inherits the function <code>put()</code> , <code>write()</code> and functions

supporting random access (`seekp()` and `tellp ()`) from `ostream` class defined inside `iostream.h` file.

`fstream` Stream class to both read and write from/to files. It provides support for simultaneous input and output operations. It inherits all the functions from `istream` and `ostream` classes through `iostream` class defined inside `iostream.h` file.

12.3 Types of data Files:

Files are used to store data or information permanently for future use. Depending on how data are stored and retrieved, the files are classified into two types. They are **Text file** and **Binary file**.

12.3.1 Text file:

It is a file that stores information in ASCII characters. In text files, each line of text is terminated with a special character known as EOL (End-of-line) character or delimiter character. When this EOL character is read or written, certain internal translations take place.

12.3.2 Binary file:

It is a file that contains information in the same format as it is held in memory. In binary files, no delimiters are used for a line and no translations occur here.

12.4 Opening and Closing files:

In C++, while opening a file, we need the stream like input, output and `input_output`. To create an input stream you must declare the stream to be of class `ifstream`. To create an output stream you must declare the stream to be of class `ofstream`. Streams that will be performing both input and output operations must be declared as class `fstream`.

Opening a file can be accomplished in two ways:

- Opening file using constructor
- Opening file using `open()` member function

The first method is preferred when a single file is used with a stream. However for managing multiple files with the same stream, the second method is preferred.

12.4.1 Opening file using constructor:

The syntax of opening a file for output purpose only using an object of `ofstream` class and the constructor is as follows:

```
ofstream ofstream _object("file_name");
```

ofstream _object is an object of type ofstream and “file name” is any valid identifier of a file to be opened for output purpose only.

Example: ofstreamfout(“results.dat”); //output only
 ofstreamfout(“text.dat”); //output only

fout is declared to be an object of ofstream type and it is made to represent the file results.dat and text.dat opened for output purpose only.

The syntax of opening a file for input purpose only using an object of ifstream class and the constructor is as follows.

ifstream ifstream _object(“file_name”);

ifstream _object is an object of type ifstream and “file name” is any valid identifier name of a file to be opened for input purpose only.

Example: ifstream fin(“results.dat”); //input only
 ifstream fin(“text.dat”); //input only

fin is declared to be an object of ifstream type and it is made to represent the file results.dat and text.dat opened for input purpose only.

12.4.2 Opening file using open():

The syntax for opening a file for output purpose only using an object of ofstream class and open() member function is as follows:

ofstream-object.open(“filename”)

ofstream-object is an object of type ofstream and “filename” is any valid name of a file to be opened for output purpose only.

Example: ofstream ofile;
 ofile.open(“data1”);
 ofile.open(“text.dat”);

ofile is declared to be an object of ofstream type and is made to represent the file data1 or text.dat opened for output purpose only.

The syntax for opening a file for input purpose only using an object of ifstream class and open() member function is as follows:

ifstream-object.open(“filename”)

ifstream-object is an object of type ifstream and “file name” is any valid name of a file to be opened for input purpose only.

Example: ifstream ifile;
 ifile.open(“data1”);
 ifile.open(“text.dat”);

ifile is declared to be an object of ifstream type and is made to represent the file data1 or text.dat opened for output purpose only.

If we have to open a file for both input and output operations, we use objects of fstream class. We know that the class fstream is derived from both ifstream and ofstream. As a result objects of the class can invoke all the members of the function of its base class.

The syntax for opening a file, an object of type fstream class and the constructor is as follows:

```
fstream fstream-object("filename", mode)
```

The syntax for opening a file an object of type fstream class and the open () member function is as follows:

```
fstream-object.open("filename",mode)
```

The need of mode is provided in the following table:

12.4.3. Concepts of file modes (in, out, app modes)

File mode parameter	Meaning	Stream type
ios::app	Append to end of file	ofstream
ios::in	open file for reading only	ifstream
ios::out	open file for writing only	ofstream
ios::ate	Open file for updation and move the file pointer to the end of file	ifstream , ofstream
ios::binary	Opening a binary file	ifstream , ofstream
ios::noreplace	Turn down opening if the file already exists	ofstream
ios::nocreate	Turn down opening if the file does not exists	ofstream
ios::trunc	On opening, delete the contents of file	ofstream

All these flags can be combined using the bitwise operator OR (|).

Example: fstream fout("text.dat",ios::out); open text.dat in output mode
 fstream fin("text.dat",ios::in); open text.dat in input mode
 fstream file;
 file.open ("example.bin", ios::out | ios::app | ios::binary);

If we want to open the file "example.bin" in binary mode to add data we could do it by the above call to member function.

12.4.4. Closing File:

We learn that opening a file establishes the linkage between a stream object and an operating system file. After the intended operations are done with the file, the file should be safely saved on the secondary storage for later retrieval. This is done by the member function `close()`. The function on its execution removes the linkage between the file and the stream object.

Syntax: `stream_object.close();`
 `fout.close();`
 `fin.close();`

12.5. Input and output operation on text files:

We know that a text file consists of a group of ASCII values. The data in text files are organized into lines with new line character as terminator. Text files need following types of character input and output operations:

- `put()` function
- `get()` function

put(): The `put()` member function belongs to the class `ofstream` and writes a single character to the associated stream.

Syntax: `ofstream_object.put(ch);`

`ch` is character constant or `char` variable. The function writes `ch` onto the file represented by `ofstream_object`.

```
char ch = 'a';
ofstream fout("text.txt");
fout.put(ch);
```

Write the character stored in `ch` onto the file represented by the object `fout`. i.e., `text.txt`.

get(): The `get()` member function belongs to the class `ifstream` and the function `get()` reads a single character from the associated stream.

Syntax: `ifstream_object.get(ch);`

`ch` is character constant or `char` variable. The function reads `ch` from the file represented by the `ifstream_object` into the variable `ch`.

```
char ch = 'a';
ifstream fin("text.txt");
fin.get(ch);
```

Reads a character into the variable `ch` the current byte position from the file represented by the object `fin`, i.e., `text.txt`.

String I/O:**The getline() function:**

It is used to read a whole line of text. It belongs to the class ifstream.

Syntax: fin.getline(buffer, SIZE);

Reads SIZE characters from the file represented by the object fin or till the new line character is encountered, whichever comes first into the buffer.

Example: char book[SIZE];
 fstream fin;
 fin.getline(book, SIZE);

Input and output operation on binary files:

The binary files are of very much use when we have to deal with database consisting of records. Since the records usually comprise heterogeneous data types, the binary files help optimize storage space and file I/O would be faster when compared to text files. Binary files needs following types of input and output operations.

- write() member function
- read() member function

write(): The write() member function belongs to the class ofstream and which is used to write binary data to a file.

Syntax: ofstream_object.write((char *) &variable, sizeof(variable));

fout is an object of type ofstream. The function requires two arguments. The first argument ofstream_object is the address of the variable, the contents of which are written to the file and the second argument is the size of the variable. The address of the variable is type casted to pointer to char type. It is because the write function does not bother to know the type of variable. It requires data in terms of only bytes. The function writes the contents of variable to the file represented by the object fout.

Example: student s;
 ofstream fout("std.dat",ios::binary);
 fout.write((char*) &s, sizeof(s));

Write the contents of the object s to the file std.dat.

read(): The read() member function belongs to the class ifstream and which is used to read binary data from a file.

Syntax: ifstream_object.read((char *) &variable, sizeof(variable));

ifstream_object is an object of type ifstream. The function requires two arguments. The first argument is the address of the variable, the contents of which are read from the file and the second argument is the size of the variable.

The address of the variable is type casted to pointer to char type. It is because the read function does not bother to know the type of variable. It requires data in terms of only bytes. The function reads a record from the file represented by the object fin to the object std.

Example:

```
student s;
ifstream fin("std.dat",ios::binary);
fin.read((char*) &s, sizeof(s));
```

Read a student record from the file std.dat into the object s.

12.6. Detecting end of file:

While reading the contents of a file, care has to be taken to see to it that the operation does not cross the end of file. The ios class provides a member function by name eof(), which helps in detecting the end of file. Once the end of file is detected with the use of eof() member function, we can stop reading further.

eоф() returns true (non zero) if end of file is encountered while reading; otherwise return false(zero).

```
if(fin.eof())
{
    statements;
}
```

This is used to execute set statements on reaching the end of the file represented by the object fin.

```
while (!fin.eof())
{
    statements;
}
```

This is used to execute set statements as long as the end of the file fin is not reached.

12.7. File pointers and their manipulation:

In C++, the file I/O operations are associated with the two file pointers, known as get pointer and the put pointer. These are synonymous for input pointer and output pointer respectively. They are useful in traversing the opened file while reading or writing.

- ifstream, like istream, has a pointer known as the get pointer that points to the element to be read in the next input operation.
- ofstream, like ostream, has a pointer known as the put pointer that points to the location where the next element has to be written.

When an input or output operation is performed, the appropriate pointer is automatically advanced. So the programmer need not bother about incrementing the file pointer to the next location inside the file for the next action.

There are three modes under which we can open a file:

- Read only mode
- Write only mode
- Append mode

When a file is opened in a read only mode, the get pointer is automatically set to the very first byte (0th byte) of the file. This helps to read the file contents from the beginning (The bytes in a file is numbered starting from zero).

Similarly, when a file is opened in a write only mode, the contents of file are erased (if it exists) and the put pointer is set to the first byte of the file, so that we can write data from the beginning. In some situations, it would be necessary for us to add new data (or text) to the existing file. In this case we have to open the file in append mode. When a file is opened in a append mode, the put pointer moves to the end-of-file, so that we write new data from that location.

These internal stream pointers that point to the reading or writing locations within a stream can be manipulated using the following member functions:

- seekg()
- seekp()
- tellg()
- tellp()

seekg():

Move the get pointer to a specified location from the beginning of a file. There are two types:

- seekg(long);
- seekg(offset, seekdir);

The seekg(long) moves the get pointer to a specified location from the beginning of a file.

Example: inf.seekg(20) ;

The above example tells that the get pointer points to 20th byte in a file from 0th byte.

The seekg(offset, seekdir) has two arguments: offset and seekdir. The offset indicates the number of bytes the get pointer is to be moved from seekdir position.

The offset takes long data type and seekdir (direction for seeking the offset position inside a file) takes one of the following three seek direction constants. These constants are defined in ios class.

Constant	Meaning
ios::beg	Offset specified from the beginning of the file
ios::cur	Offset specified from the current position of the get pointer
ios::end	Offset specified from the end of the file

Examples:

```
inf.seekg(0, ios::beg);
Move the get pointer to the 0th byte (i.e., beginning of the file).
inf.seekg(20, ios::beg);
```

Moves the get pointer to the 20th byte (i.e., from current position of the file in forward direction).

```
inf.seekg(-20, ios::beg);
```

The above example tells that the get pointer points to 20th byte in a file from end of file in backward direction.

seekp():

Move the put pointer to a specified location from the beginning of a file. There are two types:

- seekp(long);
- seekp(offset, seekdir);

The seekp(long) moves the put pointer to a specified location from the beginning of a file.

Example: inf.seekg(20) ;

The above example tells that the put pointer points to 20th byte in a file from 0th byte.

The seekp(offset, seekdir) has two arguments offset and seekdir. The offset indicates the number of bytes the put pointer is to be moved from seekdir position.

The offset takes long data type and seekdir (direction for seeking the offset position inside a file) takes one of the following three seek direction constants. These constants are defined in ios class (see above table).

Examples: inf.seekp(0, ios::beg);

Move the put pointer to the 0th byte (i.e, beginning of the file) for writing.

```
inf.seekg(20, ios::beg);
```

Move the put pointer to the 20th byte (i.e, from current position of the file in forward direction) for writing.

```
inf.seekg(-20, ios::beg);
```

The above example tells that the put pointer points to 20th byte in a file from end of file in backward direction.

tellg() member function:

The ifstream class provides the member function name tellg(). The purpose of the function is to return current position of the get pointer.

Syntax: int position;
 position = fin.tellg();

tellp() member function:

The ifstream class provides the member function name tellp(); The purpose of the function is to return current position of the put pointer.

Syntax: int position;
 position = fin.tellp();

Basic operation on binary file in C++

The basic Operation on Binary File in C++ are

1. Searching.
2. Appending data.
3. Inserting data in sorted files.
4. Deleting a record
5. Modifying data

Points to remember:

- File: The information / data stored under a specific name on a storage device, is called a file.
- Stream: It refers to a sequence of bytes.
- ifstream: Stream class to read from files. It provides input operations for file.
- ofstream: Stream class to write on files. It provides output operations for file.
- fstream: Stream class to both read and write from/to files.
- Text file: It is a file that stores information in ASCII characters.

- Binary file: It is a file that contains information in the same format as it is held in memory.
- File can be opened in two ways :
 - a. Opening file using constructor: Useful when a single file used with stream.
 - b. Opening file using open(): Useful for managing multiple files with the same stream.
- The classes defined with inside fstream.h derive from classes under iostream.h.
- File can be closed using function close ().
- File modes: Describes how a file is to be used.
- ios::in: Open file for reading only.
- ios::out: Open file for writing only.
- ios::app: Append to end of file.
- The put() member function belongs to the class ofstream and writes a single character to the associated stream.
- The get() member function belongs to the class ifstream and the function get() reads a single character from the associated stream.
- getline(): It is used to read a whole line of text. It belongs to the class ifstream.
- The write() member function belongs to the class ofstream and which is used to write binary data to a file.
- The read() member function belongs to the class ifstream and which is used to read binary data from a file.
- eof(): Helps in detecting the end of file.
- eof(): returns true (non-zero) if end-of-file is encountered while reading, otherwise return false(zero).
- seekg(): Moves the get pointer to a specified location from the beginning of a file.
- The seekg(long) moves the get pointer to a specified location from the beginning of a file.
- seekp(): Moves the put pointer to a specified location from the beginning of a file.

- tellg(): The ifstream class provides the member function name tellg(); The purpose of the function is to return current position of the get pointer.
- tellp(): The ifstream class provides the member function name tellp(); The purpose of the function is to return current position of the put pointer.

One marks questions:

1. Which header file is required for file handling functions in C++.
2. What is stream?
3. Name the streams generally used for file I/O.
4. What are output streams?
5. What are input streams?
6. Mention the methods of opening file within C++ program.
7. Write the member functions belonging to fstream class.
8. What is ifstream class
9. What is ofstream class.
10. Write the member functions belonging to ofstream class.
11. Write the member functions belonging to ifstream class.
12. Name the stream classes supported by C++ for file input.
13. Name the stream classes supported by C++ for output.
14. Mention the file modes.
15. What is ios :: in?
16. What is ios::out?
17. Mention the types of file.
18. What is text file.
19. What is binary file.
20. What is the use of write () function.
21. What is the use of writeln () function.
22. What is the use of get () function.
23. What is the use of put () function.
24. What is the use of getline () function.
25. What is the use of read () function.
26. What is the use of seekp () function.
27. What is the use of seekg () function.
28. What is the use of eof () function.

Two marks questions:

1. What is stream? Name the streams generally used for file I/O.
2. What are input and output streams?
3. Mention the methods of opening file within C++ . Discuss any one.
4. Write the member functions belonging to fstream class.
5. Differentiate between ifstream class and ofstream class.
6. Differentiate between read () and write ().
7. Differentiate between get () and getline ().

8. Write the member functions belonging to ofstream class.
9. Write the member functions belonging to ifstream class.
10. Name the stream classes supported by C++ for file input and output.
11. What are the advantages of saving data in Binary form

Three marks questions:

1. Mention the methods of opening file within C++ program. Discuss.
2. Differentiate between ifstream class and ofstream class.
3. Differentiate between read () and write () .
4. Differentiate between get () and getline () .
5. Name the stream classes supported by C++ for file input and output.
6. Mention the types of file. Explain any one.
7. What are the advantages of saving data in 1.Binary form. 2. Text form.

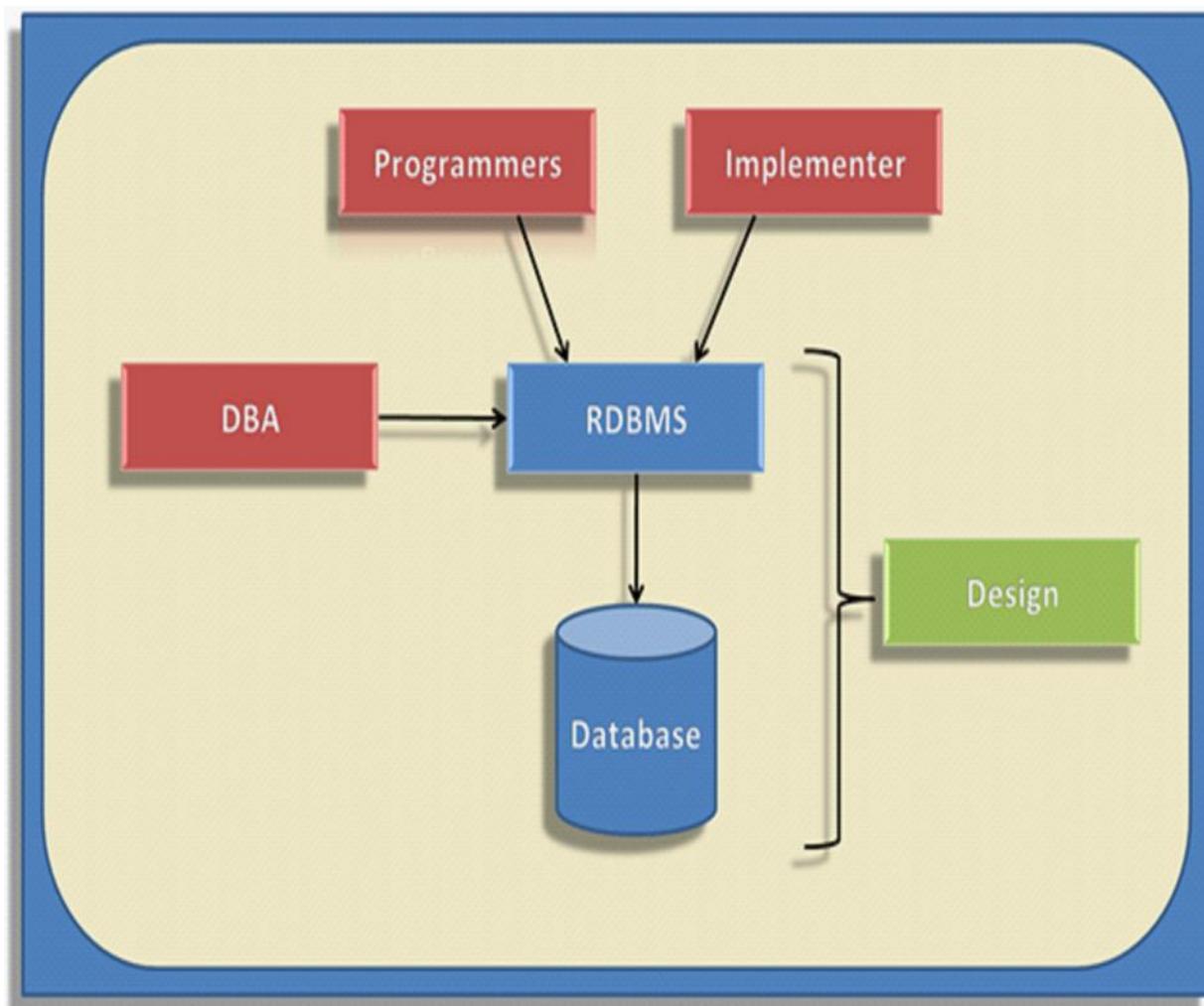
Five marks questions:

1. What are input and output streams?
2. What is significance of fstream.h header file.
3. Mention the methods of opening file within C++, Discuss.
4. Differentiate between ifstream class and ofstream class.
5. Differentiate between read () and write () with example.
6. Differentiate between get () and getline () with example.
7. Explain any three file modes.
8. Differentiate between ios::in and ios::out.

CHAPTER 13

DATABASE CONCEPTS**OBJECTIVES**

- To understand the concept of database.
- Various terms used in database.
- To understand various data model.
- To understand how the keys are used in database
- Data warehouse and datamining.

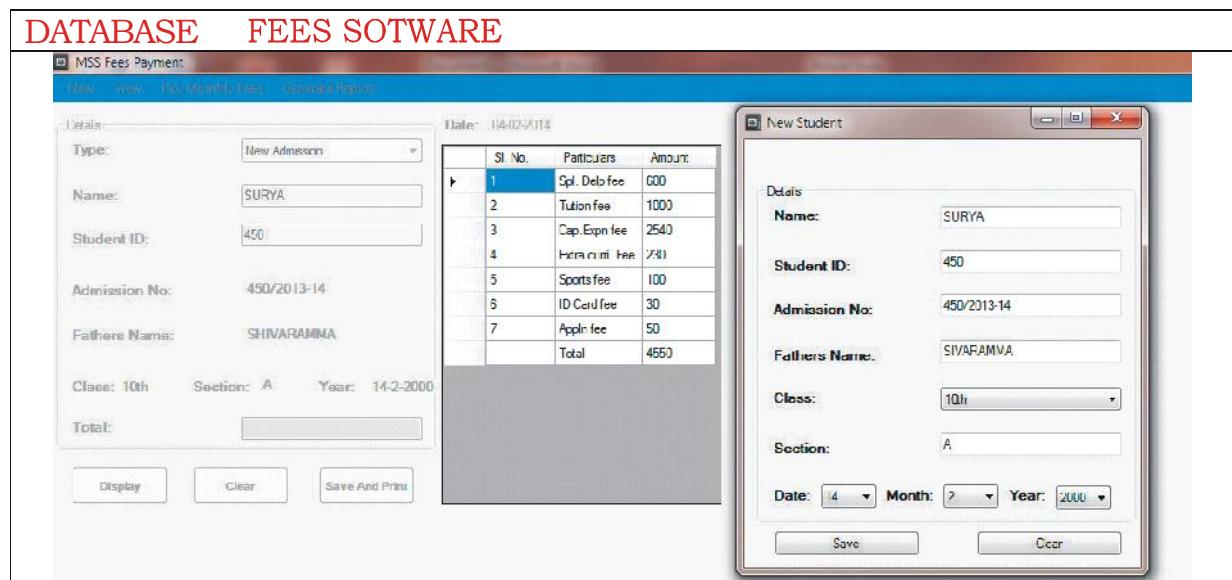


13.1 Introduction :

The large and complex data which are collected, entered, stored and accessed based on the users needs are in the form of queries. Unique softwares are developed and used, which are highly secured and complex. This chapter will illustrate the database concepts.

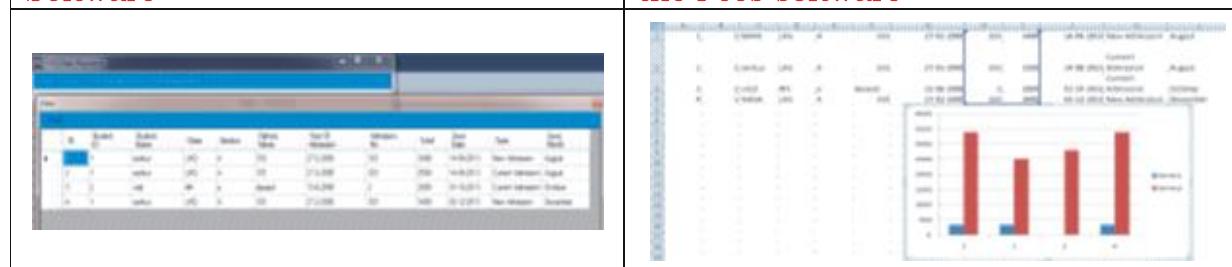
Today the database is used everywhere and in all walks of life. The database is becoming the backbone of all the softwares from standalone, client-server, online, mainframe, supercomputers etc. For example use in business, government agency, service organization etc. Wherever the data and information is required, there the database software is used and the results are presented in the form of reports or graphical representation, which are easier for understanding, so that future activities can be carried out based on these reports.

One of the area where database is used can be schools; for example. The Fees payment software. The basic requirements for new student admission are Name, student id, admission no, father name, class, section, amount, date of entry into the school are entered and saved. Some of the activities can be admission fees/ annual fees, monthly fees, late payment etc., created and the respective reports are generated.



Reports Generated for the fees Software

Graphs and charts are generated for the Fees software



13.2 APPLICATION OF DATABASE:

Databases are widely used. Here are some representative applications:

1. **Banking:** For customer information, accounts, and loans, and banking transactions.
2. **Water meter billing :** The RR number and all the details are stored in the database and connected to the server based works.
3. **Rail and Airlines:** For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner terminals situated around the world accessed the central database system through phone lines and other data networks.
4. **Colleges :** For student information, course registrations, and grades.
5. **Credit card transactions:** For purchases on credit cards and generation of monthly statements.
6. **Telecommunication:** For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
7. **Finance:** For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.
8. **Sales:** For customer, product, and purchase information.
9. **Manufacturing:** For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores, and orders for items.
10. **Human resources:** For information about employees recruitment, salaries, payroll taxes and benefits, and for generation of paychecks.

13.3 Origin of data

The fact is something that is really occurred or is actually the case. The things that are happening and happened in real form or virtual form are considered to be fact. The fact is pursued by sense organs.

Data: Data is a collection of facts, figures, statistics, which can be processed to produce meaningful information.

The process of converting fact to data will be the first task, for any person in database concepts. The human intervention is mandatory for converting from fact to data form. The data may be in the form of letters, numbers, symbols, images, sound, video etc. The origin of fact can be from the organization/within or outside organization or any part of universe. For example the marks obtained by the student in the exam is 80, 80 is the fact, on the marks card the 80 entered will be in numeric symbols which is data.

The different forms of data and its representation is illustrated. One such form is sound represented using musical notes (software generated), these notes is stored in the form of bytes in the sound file (software digitized). In the hardware data is stored in the form of bits.

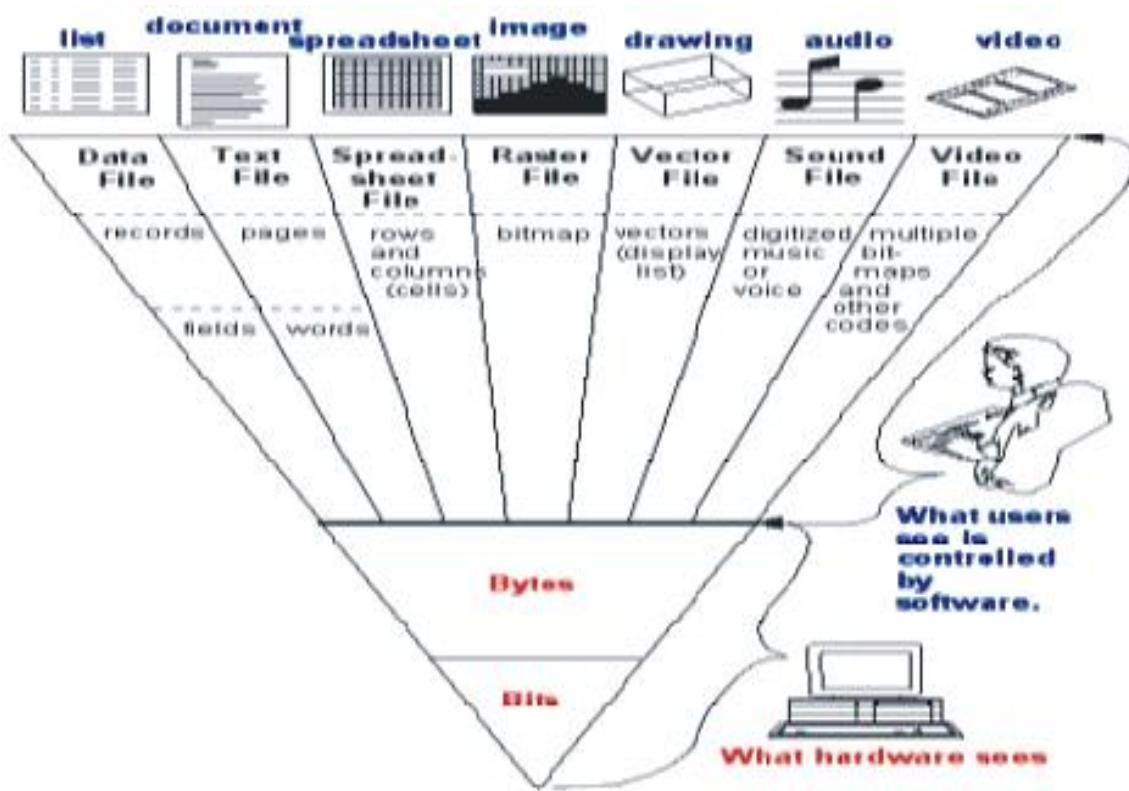


Fig. 13.1 Different forms of data

Information: Information is processed data with some definite meaning. Information represents facts, figures, or statistics, which have proper meaning.

For example, in the marks card of the student total marks, percentage and the result are processed data known as the information.

13.4 Evolution of Database - Manual File systems

Historically, When the file management came into existence, such systems were often manual, paper-and-pencil systems. The papers within these systems were organized to facilitate the expected use of the data. Typically, this was accomplished through a system of file folders and filing cabinets. As long as a collection of data was relatively small and an organization's users had few reporting requirements, the manual system served its role well as a data repository. However, as organizations grew and as reporting requirements became more complex, keeping track of data in a manual file system became more difficult. Therefore, companies looked to computer technology for help.

Computerized File systems

Initially, the computer files within the file system were similar to the manual files. The description of computer files requires a specialized vocabulary. Every discipline develops its own terminology to enable its practitioners to communicate clearly.

Differences between Manual and Computerized data processing

Manual Data processing	Computerized Electronic Data processing
The Volume of the data, which can be processed, is limited in a desirable time.	The volume of data which can be processed can be very large.
Manual data processing requires large quantity of paper	Reasonable less amount of paper is used.
The speed and accuracy at which the job is executed is limited.	The job executed is faster and Accurate.
Labour cost is high.	Labour cost is economical.
Storage medium is paper	Storage medium is Secondary storage medium.

13.5 DATA PROCESSING CYCLE

The way information is processed in a computer information management system. The information processing cycle consists of five specific steps:

1. **Data input**– This is any kind of data- letters, numbers, symbols, shapes, images or whatever raw material put into the computer system that needs processing. Input data is put into the computer using a keyboard, mouse or other devices such as the scanner, microphone and the digital camera. In general data must be converted to computer understandable form (English to machine code by the input devices).
2. **Data processing** – The processing is a series of actions or operations from the input data to generate outputs. some of the operations are classification based on some condition, calculation, sorting, indexing, accessing data, extracting part of filed/attribute, substring etc., conversion of data into information by the central processing unit. For example; when the computer adds $4+4=8$ that is an act of processing.
3. **Storage** - Data and information not currently being used must be stored so it can be accessed later. There are two types of storage; primary and secondary storage. Primary storage is the computer circuitry that temporarily holds data

waiting to be processed (RAM) and it is inside the computer. Secondary storage is where data is held permanently. A floppy disk, hard disk or CD- ROM is examples of this kind of storage.

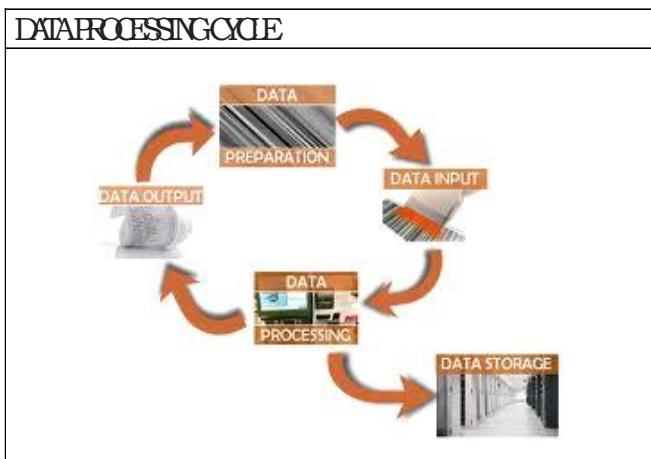


Fig. 13.2 Data Processing cycle

4 . O u t p u t - T h e result(information) obtained after processing the data must be presented to the user in user understandable form. The result may be in form of reports(hardcopy/ softcopy). Some of the output can be animated with sound and video/ picture.

5. Communication – Computers nowadays have communication ability which increases their power. With wired or wireless communication connections, data may be input from afar, processed in a remote area and stored in several different places and then be transmitted by modem as an e-mail or posted to the website where the online services are rendered.

13.6 Database terms :

File : File is basic unit of storage in computer system. The file is the large collection of related data.

Database: A Database is a collection of logically related data organized in a way that data can be easily accessed, managed and updated.

Tables : In Relational database, a table is a collection of data elements organized in terms of rows and columns. A table is also considered as convenient representation of relations. Table is the most simplest form of data storage. Below is an example :

Table : Employee,
Columns :Emp_Id,NAME,AGE,SALARY
Rows :There are four rows.

Employee table			
Emp_ID	NAME	AGE	SALARY
1	RAJAPPA	43	45000.00
2	ALEX	37	56444.00
3	RAMESH	55	56000.00
4	IMRAN	28	60000.00

Fig 13.3 Table with rows and columns

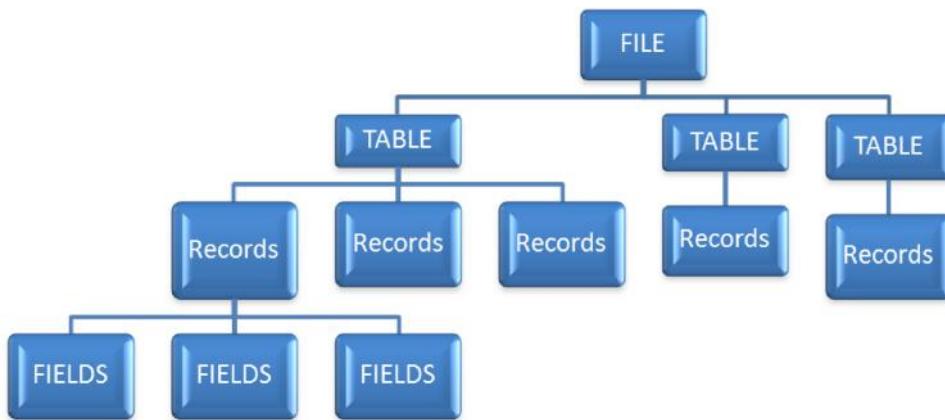


Fig 13.4 Different database terms

3	JAIPRAKASH	50	56000.00
---	------------	----	----------

Fig 13.5 Single persons details

Records: A single entry in a table is called a Record or Row. A Record in a table represents set of related data. Following is an example of single record.

Tuple :Records are also called the tuple.

Fields : Each Columns is identified by a distinct header called attribute or field

Domain :Set of values for an attribute in that column.

An Entity is an object such as a table or Form An Entity Relationship is how each table link to each other

13.7 Data types of DBMS(Data types in DBMS)

1. Integer – Hold whole number without fractions.
2. Single and double precision – Seven significant value for a number.
3. Logical data type-Store data that has only two values true or false.
4. Characters – Include letter, number, spaces, symbols and punctuation. Characters fields or variables store text information like name, address, but size will be one byte.
5. Strings – Sequence of character more than one. Fixed length is 0 to 63Kb and dynamic strings length range from 0 to 2 billion characters.
6. Memo data type – Store more than 255 characters. A memo fields can store up to 65536 characters. Long documents can store OLE objects.
7. Index fields –Used to store relevant information along with the documents. The document input to an index field is used to find those documents when needed. The programs provides up to 25 user definable index fields in an index set. Name drop-down look-up list, Standard, auto-complete History list.
8. Currency fields – The currency field accepts data in dollar form by default.
9. Date fields -The date fields accepts data entered in date format.
10. Text fields –Accepts data as an alpha-numeric text string.

13.8 DBMS-DATABASE MANAGEMENT SYSTEM

Database	Vendor
SQL Server	Microsoft
Oracle	Oracle
DB2	IBM
MySQL	Oracle
Sybase	SAP

A DBMS is a software that allows creation, definition and manipulation of database. DBMS is actually a tool used to perform any kind of operation on data in database. DBMS also provides protection and security to database. It maintains data consistency in case of multiple users. Here are some examples of popular DBMS, MySQL, Oracle, Sybase, Microsoft Access and IBM DB2 etc.

The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

Features of Database System

Database approach came into existence due to the overcome of the drawbacks of file processing system. In the database approach, the data is stored at a central location and is shared among multiple users. Thus, the main advantage of DBMS

is **centralized data management**. The centralized nature of database system provides several advantages, which overcome the limitations of the conventional file processing system. These advantages are listed here.

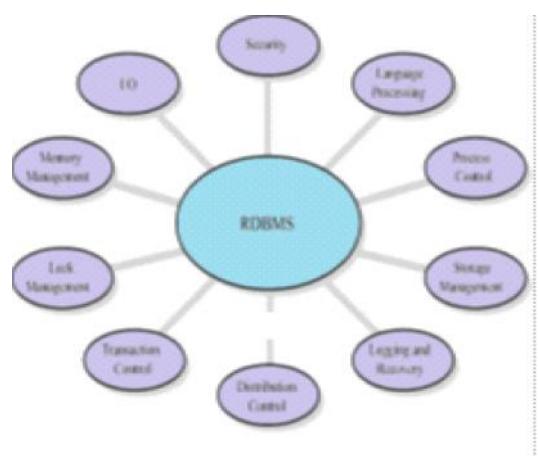


Fig 13.6 Features of DBMS

saves the storage space. Note that the redundancy in the database systems cannot be eliminated completely as there could be some performance and technical reasons for having some amount of redundancy. However, the DBMS should be capable of controlling this redundancy in order to avoid data inconsistencies.

- **Enforcing data integrity:** In database approach, enforcing data integrity is much easier. Data integrity refers to the validity of data and it can be compromised

in a number of ways. Data integrity constraints can be enforced automatically by the DBMS, and others may have to be checked by the application programs.

- **Data sharing:** The data stored in the database can be shared among multiple users or application programs. Moreover, new applications can be developed to use the same stored data. Due to shared data, it is possible to satisfy the data requirements of the new applications without having to create any additional data or with minimal modification.
- **Ease of application development:** The application programmer needs to develop the application programs according to the users' needs. The other issues like concurrent access, security, data integrity, etc., are handled by the RDBMS itself. This makes the application development an easier task.
- **Data security:** Since the data is stored centrally, enforcing security constraints is much easier. The RDBMS ensures that the only means of access to the database is through an authorized channel only. Hence, data security checks can be carried out whenever access is attempted to sensitive data. To ensure security, a RDBMS provides security tools such as user codes and passwords. Different checks can be established for each type of access (addition, modification, deletion, etc.) to each piece of information in the database.
- **Multiple user interfaces:** In order to meet the needs of various users having different technical knowledge, DBMS provides different types of interfaces such as query languages, application program interfaces, and graphical user interfaces (GUI) that include forms-style and menu-driven interfaces. A form-style interface displays a form to each user and user interacts using these forms. In menu-driven interface, the user interaction is through lists of options known as menus.

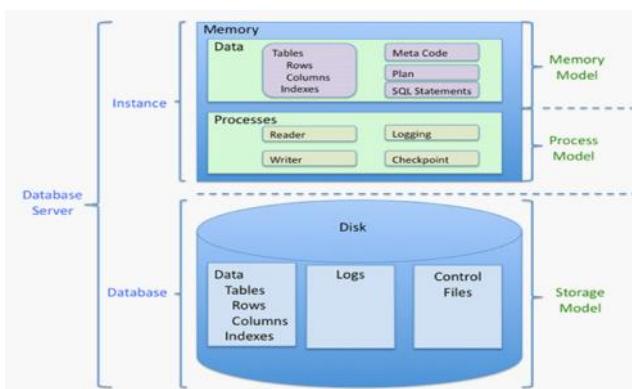


Fig 13.7 Different Layers Database transaction, the RDBMS recovery subsystem either reverts back the database to the state which existed prior to the start of the transaction or resumes the transaction from the point it was interrupted so that its complete effect can be recorded in the database.

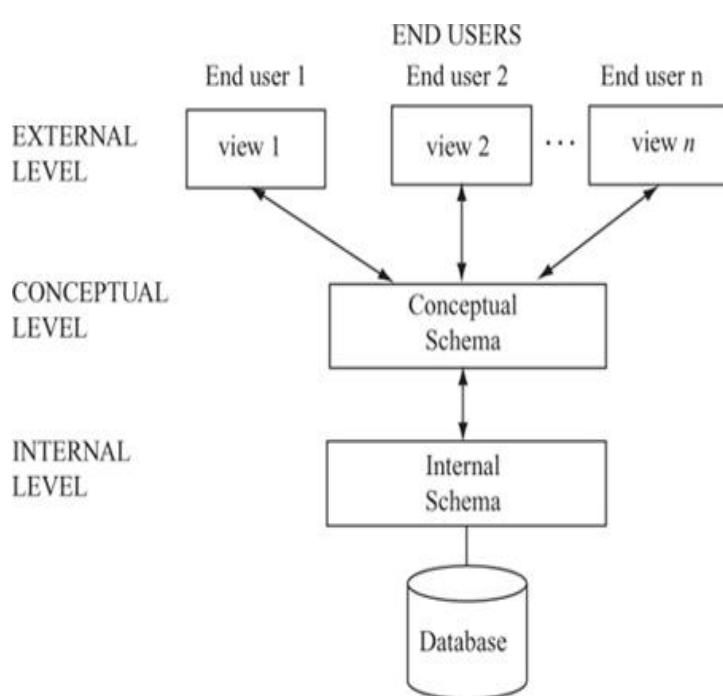
Note : Why a Spreadsheet Is Not a Database

While a spreadsheet allows for the creation of multiple tables, it does not support even the most basic data-base functionality such as support for self-documentation through metadata, enforcement of data types or domains to ensure consistency of data within a column, defined relationships among tables, or constraints to ensure consistency of data across related tables. Most users lack the necessary training to recognize the limitations of spreadsheets for these types of tasks.

13.9 Data Abstraction

The DBMS architecture describes how data in the database is viewed by the users. It is not concerned with how the data is handled and processed by the RDBMS. The database users are provided with an abstract view of the data by hiding certain details of how data is physically stored. This enables the users to manipulate the data without worrying about where it is located or how it is actually stored.

In this architecture, the overall database description can be defined at three levels, namely, internal, conceptual, and external levels and thus, named **three-level RDBMS architecture**. This architecture is proposed by ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee) and hence, is also known as ANSI/SPARC architecture.



Internal level: Internal level is the lowest level of data abstraction that deals with the physical representation of the database on the computer and thus, is also known as **physical level**. It describes how the data is physically stored and organized on the storage medium. At this level, various aspects are considered to achieve optimal runtime performance and storage space utilization. These aspects include storage space allocation techniques for data and indexes, access paths such as indexes,

Fig 13.8 Different Levels of Database

data compression and encryption techniques, and record placement.

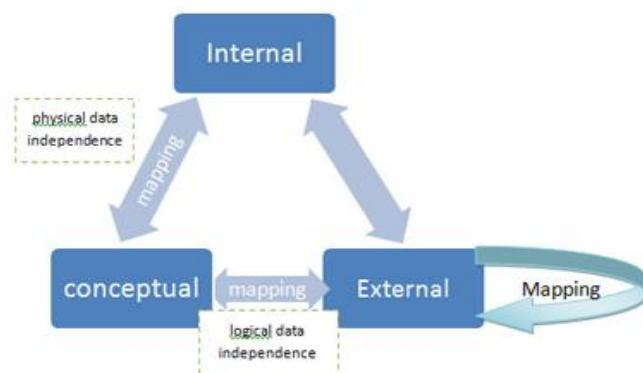
Conceptual level: This level of abstraction deals with the logical structure of the entire database and thus, is also known as **logical level**. Conceptual level describes what data is stored in the database, the relationships among the data and complete view of the user's requirements without any concern for the physical implementation. That is, it hides the complexity of physical storage structures.

The conceptual view is the overall view of the database and it includes all the information that is going to be represented in the database.

External level: External level is the highest level of abstraction that deals with the user's view of the database and thus, is also known as **view level**. In general, most of the users and application programs do not require the entire data stored in the database. The external level describes a part of the database for a particular group of users. It permits users to access data in a way that is customized according to their needs, so that the same data can be seen by different users in different ways, at the same time. In this way, it provides a powerful and flexible security mechanism by hiding the parts of the database from certain users, as the user is not aware of existence of any attributes that are missing from the view.

DBMS users : The broad classification of dbms users are

1. **Application programmers and system analysts:** System analysts determine the requirement of end users; especially naive, parametric end users, and develop specifications for transactions that meet these requirements. Application programmers implement these parameters in programs.
2. **End users :** People who require access to the database for querying updating and generating reports. The database exists primarily for/their use.
3. **Database Administrator (DBA):** DBA is responsible for authorization access to the database for coordinating and monitoring its use, and for acquiring the needed software and hardware resources.



4. **Database designers:** Database designers are responsible for identifying the data to be stored in the database for choosing appropriate structures to represent and store the data.

Schema objects are database

objects that contain data or govern or perform operations on data.

13.10 Data Independence:

Data Independence is an ability of a database to modify a schema definition at one level without affecting a schema in the next higher level (the ability to change the conceptual schema without affecting the external schemas or application programs). Data independence occurs because when the schema is changed at one level, the schema at next level remains unchanged and only the mapping between the two levels is changed. Two types of data independence are:

1. Physical Data Independence.
2. Logical Data Independence

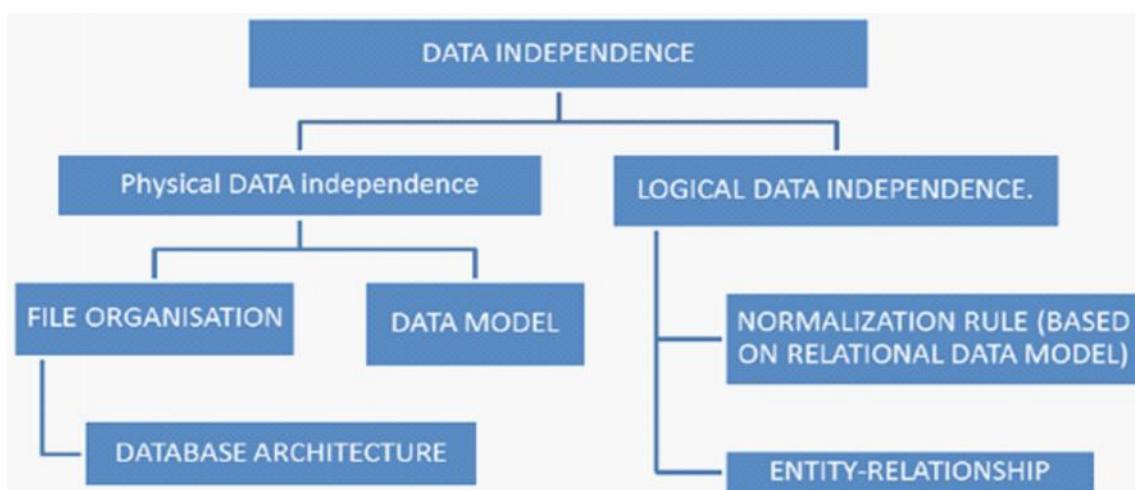


Fig 13.9 Data Independence

Physical Data independence

All schemas are logical and actual data is stored in bit format on the disk. Namely storage medium: Hard disk (all the files will be stored), floppies, drum, tapes, SD etc., System designs choose to organize, access and process records and files in different ways depending on the type of application and the needs of users. The three commonly used file organizations used in dbms/rdbms data processing applications are sequential, direct and indexed sequential access method(ISAM). The selection of a particular file organization depends upon the application used. To access a record some key field or unique identifying value that is found in every record in a file is used.

Serial File Organization : With serial file organization, records are arranged one after another, in no particular order-other than, the chronological order in which records are added to the file. Serial organization is commonly found in the transaction data. Where records are created in a file in the order in which transaction takes place. Serial file organization provides advantages like fast

access to next records in sequence, stored in economical storage media and easy to do the file backup facility, updating is slowly in this file organization.

Sequential File organization : Records are stored one after another in an ascending or descending order determined by the key field of the records. Example payroll file, records are stored in the form of employee id. Sequentially organized files that are processed by computer systems are normally stored on storage media such as magnetic tape, punched cards, or magnetic disks. To access these records the computer must read the file in sequence from the beginning. The first record is read and processed first, then the second record in the file sequence, and so on. To locate a particular record, the computer program must read in each record in sequence and compare its key field to the one that is needed. The retrieval search ends only when the desired key matches with the key field of the currently read record. On an average, about half of the file has to be searched to retrieve the desired record from a sequential file.

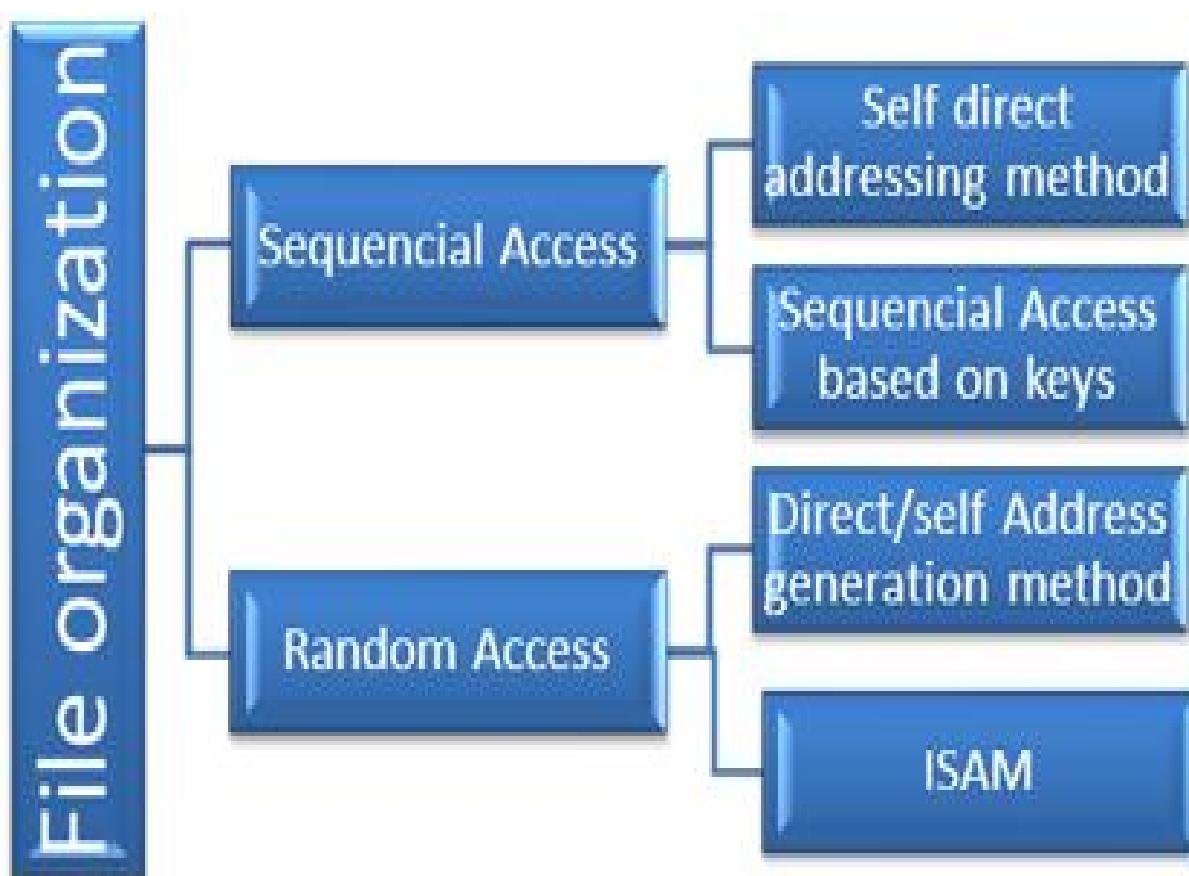


Fig 13.10 File organisation

Random/Direct Access File Organization: Direct access file organization allow immediate direct access to individual records on the file. The record are stored and retrieved using a relative record number, which gives the position of the record in the file. This type of organization also allows the file to accessed sequentially. The primary storage in a CPU truly provides for direct access. There are some devices outside the CPU which can provide the direct feature;the direct access storage devices have the capability of directly reaching any location. Although there are several types of storage devices including discs and other mass storage.

Self(direct) Addressing: Under self-direct addressing, a record key is used as its relative address. Therefore, anyone can compute the record's address from the record key and the physical address of the first record in the file. Advantage is self-addressing no need to store an index.

Disadvantages are, the records must be of fixed length, if some records are deleted the space remains empty.

Random access method : Records are stored on disk by using a hashing algorithm. The key field is fed through hashing algorithm and a relative address is created. This address gives the position on the disk where the record is to be stored. The desired records can be directly accessed using randomizing procedure or hashing without accessing all other records in the file. Randomizing procedure is characterized by the fact that records are stored in such a way that there is no relationship between the keys of the adjacent records. The technique provide for converting the records key number to a physical location represented by a disk address through a computational procedure.

Advantages : The access to, and retrieval of a records is quick and direct.

Transactions need not be stored and placed in sequence prior to processing
Best used for online transaction.

Disadvantages: Address generation overhead is involved for accessing each record due to hashing function.

May be less efficient in the use of storage space than sequentially organized files.

Indexed Sequential Access Method(ISAM): ISAM is the hybrid between sequential and direct access file organization. The records within the file are stored sequentially but direct access to individual records is possible through an index. Indexing permit access to selected records without searching the entire file.

Advantages: ISAM permits efficient and economical use of sequential processing techniques when the activity ratio is high.

Permits direct access processing of records in a relatively efficient way when the activity ratio is low.

Cylinder	Highest Record key in the cylinder
1	84
2	250
3	398
4	479
Cylinder 1 track index	
Track	Highest Record key in the cylinder
1	84
2	250
3	398
4	479
Cylinder 2 track index	
Track	Highest Record key in the cylinder
1	95
2	110
3	175
4	250

Fig 13.11 Table describing ISAM

Disadvantages: Files must be stored in a direct-access storage device. Hence relatively expensive hardware and software resources are required.

Access to records may be slower than direct file.

Less efficient in the use of storage space than some other alternatives.

Different types of Architecture

The design of a Database Management System highly depends on its architecture. It can be centralized or decentralized or hierarchical. DBMS architecture can be seen as single tier or multi-tier. N-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed or replaced.

Database architecture is logically divided into three types.

1. Logical one-tier **In 1-tier architecture**,
2. Logical two-tier Client / Server architecture
3. Logical three-tier Client/Server architecture

Logical one-tier **In 1-tier architecture**

In 1-tier architecture, DBMS is the only entity where user directly sits on DBMS and uses it. Any changes done here will directly be done on DBMS itself. It does not provide handy tools for end users and preferably database designer and programmers use single tier architecture.

Fig 13.12 Logical one tier architecture



Logical two-tier Client / Server architecture**Two-tier Client / Server Architecture**

Two-tier Client / Server architecture is used for User Interface program and Application Programs that runs on client side. An interface called ODBC (Open Database Connectivity) provides an API that allows client side program to call the DBMS. Most DBMS vendors provide ODBC drivers. A client program may connect to several DBMS's. In this architecture some variation of client is also possible for example in some DBMS's more functionality is transferred to the client including data dictionary, optimization etc. Such clients are called Data server.

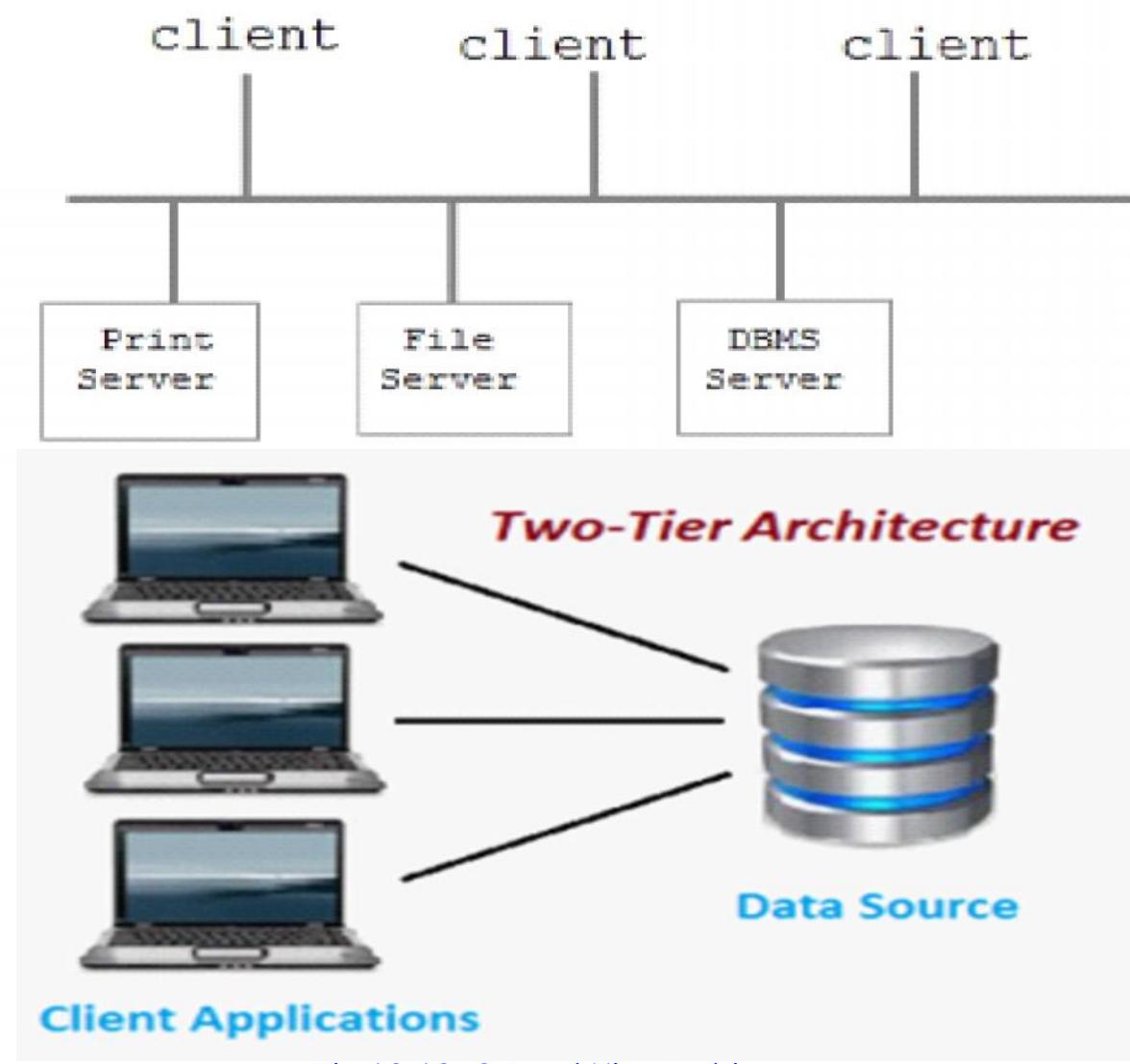


Fig 13.13 2-Level Tier Architecture

Three-tier Client / Server Architecture

Three-tier Client / Server database architecture is commonly used architecture for web applications. Intermediate layer called Application server or Web Server stores the web connectivity software and the business logic (constraints) part of application used to access the right amount of data from the database server. This layer acts like medium for sending partially processed data between the database server and the client.

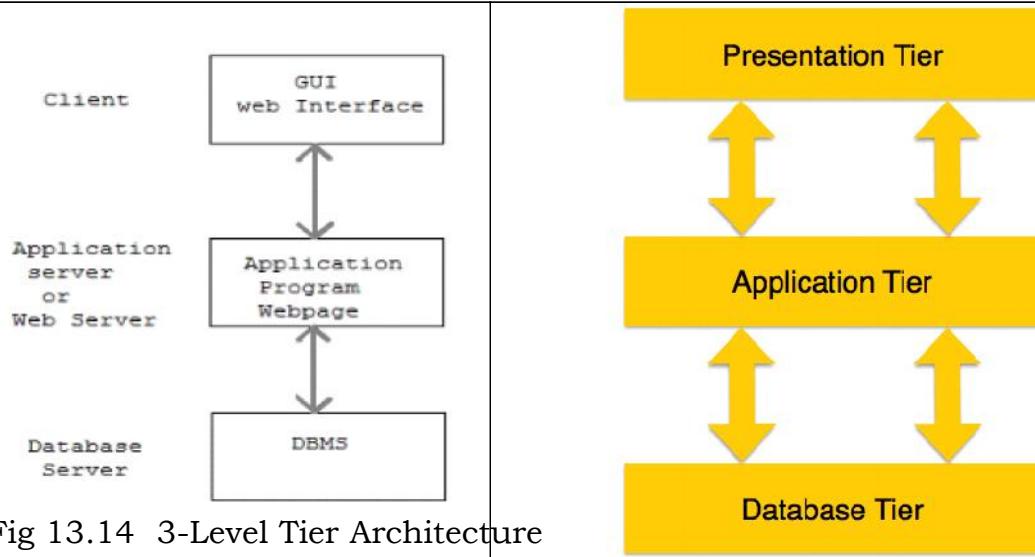


Fig 13.14 3-Level Tier Architecture

Database (Data) Tier: At this tier, only database resides. Database along with its query processing languages sits in layer-3 of 3-tier architecture. It also contains all relations and their constraints.

Application (Middle) Tier: At this tier the application server and program, which access database, resides. For a user this application tier works as abstracted view of database. Users are unaware of any existence of database beyond application. For database-tier, application tier is the user of it. Database tier is not aware of any other user beyond application tier. This tier works as mediator between the two.

User (Presentation) Tier: An end user sits on this tier. From a users aspect this tier is everything. He/she doesn't know about any existence or form of database beyond this layer. At this layer multiple views of database can be provided by the application. All views are generated by applications, which reside in application tier. Multiple tier database architecture is highly modifiable as almost all its components are independent and can be changed independently.

13.11 Database Model

A database model or simply a **data model** is an abstract model that describes how the data is represented and used. A data model consists of a set of data

structures and conceptual tools that is used to describe the structure (data types, relationships, and constraints) of a database.

A data model not only describes the structure of the data, it also defines a set of operations that can be performed on the data. A data model generally consists of **data model theory**, which is a formal description of how data may be structured and used, and **data model instance**, which is a practical data model designed for a particular application. The process of applying a data model theory to create a data model instance is known as **data modeling**.

The main objective of database system is to highlight only the essential features and to hide the storage and data organization details from the user. This is known as data abstraction. A database model provides the necessary means to achieve data abstraction.

A Database model defines the logical design of data. The model describes the relationships between different parts of the data.

In history of database design, three models have been in use.

- * Hierarchical Model
- * Network Model
- * Relational Model

13.11.1 Hierarchical Model

The hierarchical data model is the oldest type of data model, developed by IBM in 1968. This data model organizes the data in a tree-like structure, in which each **child node** (also known as **dependents**) can have only one **parent node**. The database based on the hierarchical data model comprises a set of records connected to one another through links. The link is an association between two or more records. The top of the tree structure consists of a single node that does not have any parent and is called the **root node**.

The root may have any number of dependents; each of these dependents may have any number of lower level dependents. Each child node can have only one parent node and a parent node can have any number of (many) child nodes. It, therefore, represents only one-to-one and one-to-many relationships. The collection of same type of records is known as a **record type**.

For simplicity, only few fields of each record type are shown. One complete record of each record type represents a **node**.

In this model each entity has only one parent but can have several children . At the top of hierarchy there is only one entity which is called Root.

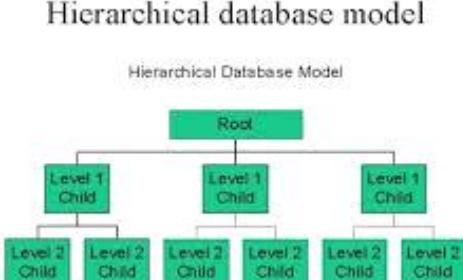
Hierarchical Model	Example										
	<p style="text-align: center;">Department</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>No.</td> <td>Name</td> </tr> </table> <p style="text-align: center;">course</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>no.</td> <td>name</td> <td>unit</td> </tr> </table> <p style="text-align: center;">student</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>id</td> <td>name</td> <td>course</td> </tr> </table> <p style="text-align: center;">Professor</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>id</td> <td>Name</td> </tr> </table>	No.	Name	no.	name	unit	id	name	course	id	Name
No.	Name										
no.	name	unit									
id	name	course									
id	Name										

Fig 13.15 Hierarchical Model

Advantage	Dis-advantage
The hierarchical data model is that the data access is quite predictable in the structure and, therefore, both the retrieval and updates can be highly optimized by the DBMS.	The main drawback of this model is that the links are ‘hard coded’ into the data structure, that is, the link is permanently established and cannot be modified. The hard coding makes the hierarchical model rigid. In addition, the physical links make it difficult to expand or modify the database and the changes require substantial redesigning efforts.

13.11.2 Network Model

The first specification of network data model was presented by Conference on Data Systems Languages (CODASYL) in 1969, followed by the second specification in 1971. It is powerful but complicated. In a network model the data is also represented by a collection of records, and relationships among data are represented by links. However, the link in a network data model represents an association between precisely two records. Like hierarchical data model, each record of a particular record type represents a node. However, unlike hierarchical data model, all the nodes are linked to each other without any hierarchy. The main difference between hierarchical and network data model is that in

hierarchical data model, the data is organized in the form of trees and in network data model, the data is organized in the form of graphs.

In the network model, entities are organized in a graph, in which some entities can be accessed through several path

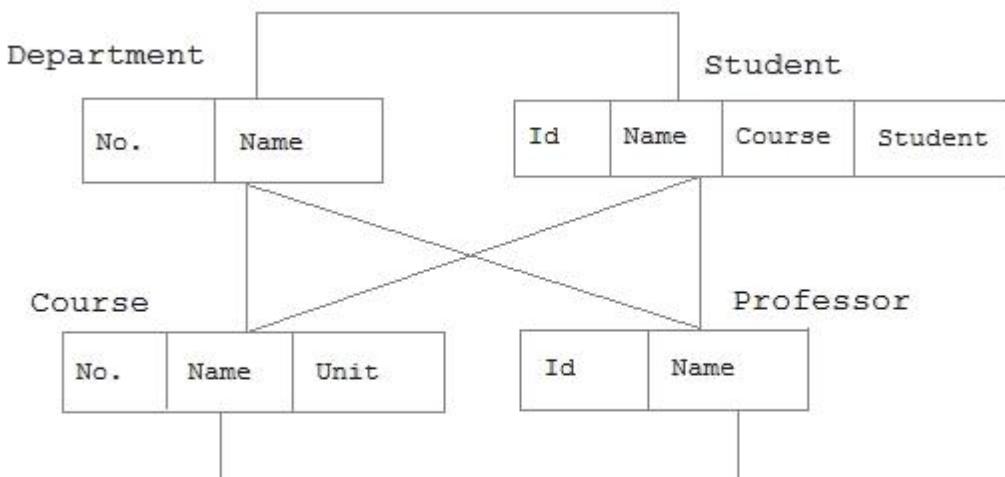


Fig 13.16 Network Model of database

Network Model of database

Advantage	Dis-advantage
The network data model is that a parent node can have many child nodes and a child can also have many parent nodes. Thus, the network model permits the modeling of many-to-many relationships in data.	The network data model is that it can be quite complicated to maintain all the links and a single broken link can lead to problems in the database. In addition, since there are no restrictions on the number of relationships, the database design can become complex.

13.11.3 Relational Model

The relational data model was developed by E. F. Codd in 1970. In the relational data model, unlike the hierarchical and network models, there are no physical links. All data is maintained in the form of tables (generally, known as **relations**) consisting of rows and columns. Each row (record) represents an entity and a column (field) represents an attribute of the entity. The relationship between the two tables is implemented through a common attribute in the tables and not by physical links or pointers. This makes the querying much easier in a relational database system than in the hierarchical or network database systems. Thus,

the relational model has become more programmer friendly and much more dominant and popular in both industrial and academic scenarios. Oracle, Sybase, DB2, Ingres, Informix, MS-SQL Server are few of the popular relational DBMSs.

In this model, data is organized in two-dimensional tables called relations. The tables or relation are related to each other.

Relational Model of database

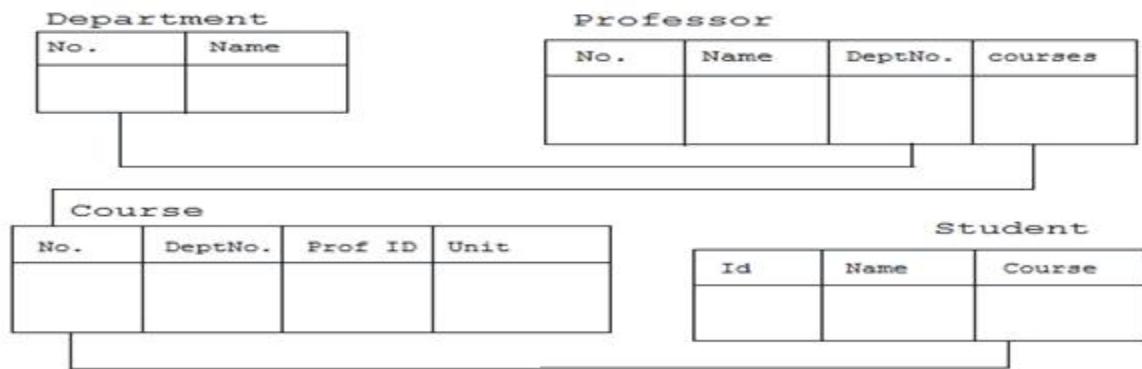


Fig 13.17 Relational Model of database

Basic Rules for the Relational Datamodel

13.12 Codd's Rule

E.F Codd was a Computer Scientist who invented Relational model for Database management. Based on relational model, Relation database was created. Codd proposed 13 rules popularly known as Codd's 12 rules to test DBMS's concept against his relational model. Codd's rule actually define what quality a DBMS requires in order to become a Relational Database Management System(RDBMS). Till now, there is hardly any commercial product that follows all the 13 Codd's rules. Even Oracle follows only eight and half out(8.5) of 13. The Codd's 12 rules are as follows.

Rule zero

This rule states that for a system to qualify as an RDBMS, it must be able to manage database entirely through the relational capabilities.

Rule 1 : Information rule

All information(including meta-data) is to be represented as stored data in cells of tables. The rows and columns have to be strictly unordered.

Rule 2 : Guaranteed Access

Each unique piece of data(atomic value) should be accessible by :

Table Name + primary key(Row) + Attribute(column).

NOTE : Ability to directly access via POINTER is a violation of this rule.

Rule 3 : Systemetic treatment of NULL

Rule 3 : Systemetic treatment of NULL

Null has several meanings, it can mean missing data, not applicable or no value. It should be handled consistently. Primary key must not be null. Expression on NULL must give null.

Rule 4 : Active Online Catalog

Database dictionary(catalog) must have description of Database.

Catalog to be governed by same rule as rest of the database. The same query language to be used on catalog as on application database.

Rule 5 : Powerful language

One well defined language must be there to provide all manners of access to data. Example: SQL. If a file supporting table can be accessed by any manner except SQL interface, then its a violation to this rule.

Rule 6 : View Updation rule

All view that are theoretically updatable should be updatable by the system.

Rule 7 : Relational Level Operation

There must be Insert, Delete, Update operations at each level of relations. Set operation like Union, Intersection and minus should also be supported.

Rule 8 : Physical Data Independence

The physical storage of data should not matter to the system. If say, some file supporting table were renamed or moved from one disk to another, it should not effect the application.

Rule 9 : Logical Data Independence

If there is change in the logical structure(table structures) of the database the user view of data should not change. Say, if a table is split into two tables, a new view should give result as the join of the two tables. This rule is most difficult to satisfy.

Rule 10 : Integrity Independence

The database should be able to con-force its own integrity rather than using other programs. Key and Check constraints, trigger etc should be stored in Data Dictionary. This also make *RDBMS* independent of front-end.

Rule 11 : Distribution Independence

A database should work properly regardless of its distribution across a network. This lays foundation of distributed database.

Rule 12 : Non-subversion rule

If low level access is allowed to a system it should not be able to subvert or bypass integrity rule to change data. This can be achieved by some sort of looking or encryption.

13.13 Logical database concepts : Normalization, Entities, attributes, relations

13.13.1 Normalization Rule: Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

There are a few rules for database normalization. Each rule is called a “normal form.” If the first rule is observed, the database is said to be in “first normal form.” If the first three rules are observed, the database is considered to be in “third normal form.” Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.

As with many formal rules and specifications, real world scenarios do not always allow for perfect compliance. In general, normalization requires additional tables and some customers find this cumbersome. If you decide to violate one of the first three rules of normalization, make sure that your application anticipates any problems that could occur, such as redundant data and inconsistent dependencies.

Normalization rule are divided into following normal form.

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. BCNF

First Normal Form (1NF)

A row of data cannot contain repeating group of data i.e each column must have a unique value. Each row of data must have a unique identifier i.e *Primary key*. For example consider a table which is not in First normal form

Student Table :

S_id	S_Name	S_Subject
401	Daryl	Maths
401	Daryl	Maths
402	Ramesh	Physics
403	Rakshana	Maths
404	Nakshatria	Computer Science

You can clearly see here that student name Daryl is used twice in the table and subject *maths* is also repeated. This violates the *First Normal form*. To reduce above table to *First Normal form* break the table into two different tables

New Student Table :

S_id	S_Name	S_Subject
401	Daryl	Maths
402	Ramesh	Physics
403	Rakshana	Maths
404	Nakshatria	Computer Science

Subject Table :

Fig 13.18 1NF Student table with S_id,S_name,S_subject

Suject_id	S_id	S_Subject
35	401	Maths
35	401	Maths
33	403	Physics
34	404	Chemistry
41	405	Computer Science

In Student table concatenation of subject_id and student_id is the Primary key. Now both the Student table and Subject table are normalized to first normal form

Second Normal Form (2NF)

A table to be normalized to Second Normal Form should meet all the needs of First Normal Form and there must not be any partial dependency of any column on primary key. It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails Second normal form. For example, consider a table which is not in Second normal form.

Library Table : Fig 13.19 Student table with 1NF rule rewritten

Library_id	S_Name	Issue_id	Issue_name	Book_detail
101	RAMU	10	Rakesh	C++
102	RAMU	11	Rakesh	Java
103	Zama	12	Gopal	MATHS
104	SATISH	13	Gopal	MATHS

In Library table concatenation of Library_id and issue_id is the primary key. This table is in First Normal form but not in Second Normal form because there are partial dependencies of columns on primary key. S_Name is only dependent on Library_id, Issue_name is dependent on Issue_id and there is no link between Book_detail and S_name.

To reduce Library table to Second Normal form break the table into following three different tables.

Library_id S_Name

Library_id	S_Name
101	RAMU
102	RAMU
103	Zama
104	SATISH

Issue_Detail Table :

Issue_id	Issue_name
10	Rakesh
11	Rakesh
12	Gopal
13	Gopal

Book_Detail Table :

Library_id	Issue_id	Book_detail
101	10	C++
102	11	Java
103	12	MATHS
104	13	MATHS

Now all these three table comply with Second Normal form.

Library_id S_Name		Issue_Detail Table :	
Library_id	S_Name	Issue_id	Issue_name
101	RAMU	10	Rakesh
102	RAMU	11	Rakesh
103	Zama	12	Gopal
104	SATISH	13	Gopal

Book_Detail Table :		
Library_id	Issue_id	Book_detail
101	10	C++
102	11	Java
103	12	MATHS
104	13	MATHS

Fig 13.20 2NF Library Tablewith primary key and Issue key

Now all these three table comply with Second Normal form.

Third Normal Form (3NF)

Third Normal form applies that every non-prime attribute of table must be dependent on primary key. The transitive functional dependency should be removed from the table. The table must be in Second Normal form.

For example, consider a table with following fields.

Student_Detail Table :

Student_id	Student_name	DOB	Street	city	State	Pin
------------	--------------	-----	--------	------	-------	-----

In this table Student_id is Primary key, but street, city and state depends upon pin. The dependency between pin and other fields is called transitive dependency. Hence to apply 3NF, we need to move the street, city and state to new table, with pin as primary key.

New Student_Detail Table :

Student_id	Student_name	DOB	pin
------------	--------------	-----	-----

Address Table :

pin	Street	city	state
-----	--------	------	-------

The advantage of removing transitive dependency is,

- * Amount of data duplication is reduced.
- * Data integrity achieved.

Boyce and Codd Normal Form (BCNF)

BCNF is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.

- When a relation has more than one candidate key, anomalies may result even though the relation is in 3NF.
- 3NF does not deal satisfactorily with the case of a relation with overlapping candidate keys
- i.e. composite candidate keys with at least one attribute in common.

- BCNF is based on the concept of a *determinant*.
- A determinant is any attribute (simple or composite) on which some other attribute is fully functionally dependent.
- A relation is in BCNF if, and only if, every determinant is a candidate key.

Consider the following relation and determinants.

$R(a,b,c,d)$

$a,c \rightarrow b,d$

$a,d \rightarrow b$

Here, the first determinant suggests that the primary key of R could be changed from a,b to a,c . If this change was done all of the non-key attributes present in R could still be determined, and therefore this change is legal. However, the second determinant indicates that a,d determines b , but a,d could not be the key of R as a,d does not determine all of the non key attributes of R (it does not determine c). We would say that the first determinate is a candidate key, but the second determinant is not a candidate key, and thus this relation is not in BCNF (but is in 3rd normal form).

„A relation $R(X)$ is in Boyce–Codd Normal Form if for every non-trivial functional dependency $Y \rightarrow Z$ defined on it, Y contains a key K of $R(X)$. That is, Y is a superkey for $R(X)$.

13.13.2 Entity-Relationship Diagram

ER-Diagram is a visual representation of data that describes how data is related to each other.

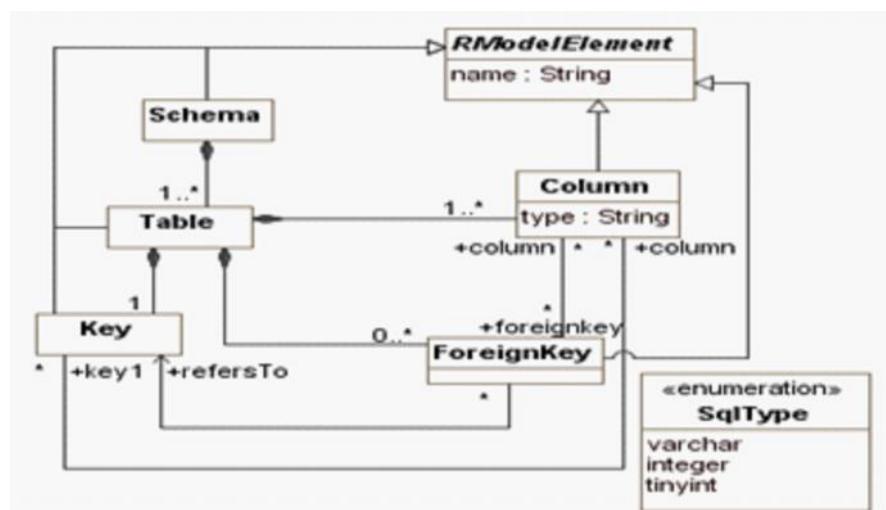


Fig 13.21 Entity-Relationship Symbols

Components of E-R Diagram

The E-R diagram has three main components.

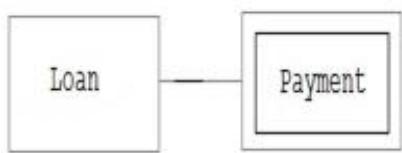
1) Entity

An Entity can be any object, place, person or class. In E-R Diagram, an entity is represented using rectangles. Consider an example of an Organization. Employee, Manager, Department, Product and many more can

be taken as entities from an Organization.

Entity	In E-R Diagram
	

Fig 13.22 Entity and example



Weak entity is an entity that depends on another entity. Weak entity doesn't have key attribute of their own. Double rectangle represents weak entity.

Fig 13.23 Entity with loan and payment

2) Attribute

An Attribute describes a property or characteristic of an entity. For example, Name, Age, Address etc. can be attributes of a Student. An attribute is represented using ellipse.

Attribute

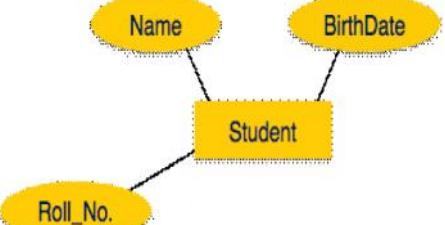
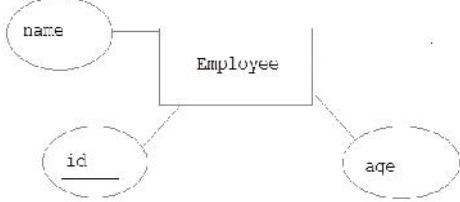
Attribute	Attribute example
	

Fig 13.24 Attribute with example

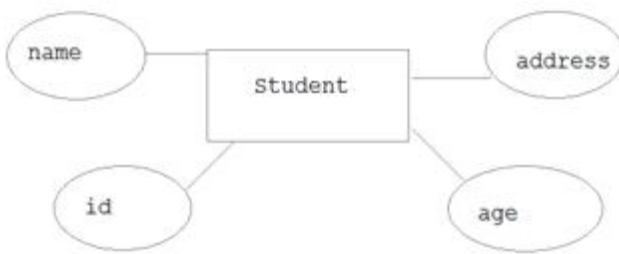
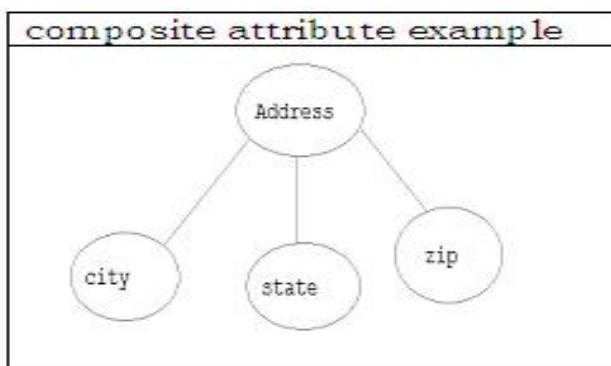


Fig 13.25 Attribute key with primary key

Key Attribute Key attribute represents the main characteristic of an Entity. It is used to represent Primary key. Ellipse with underlying lines represent Key Attribute.



Composite Attribute : An attribute can also have their own attributes. These attributes are known as Composite attribute.

composite attribute example

Fig 13.26 Composite attribute

3) Relationship

A Relationship describes relations between entities. Relationship is represented using diamonds.

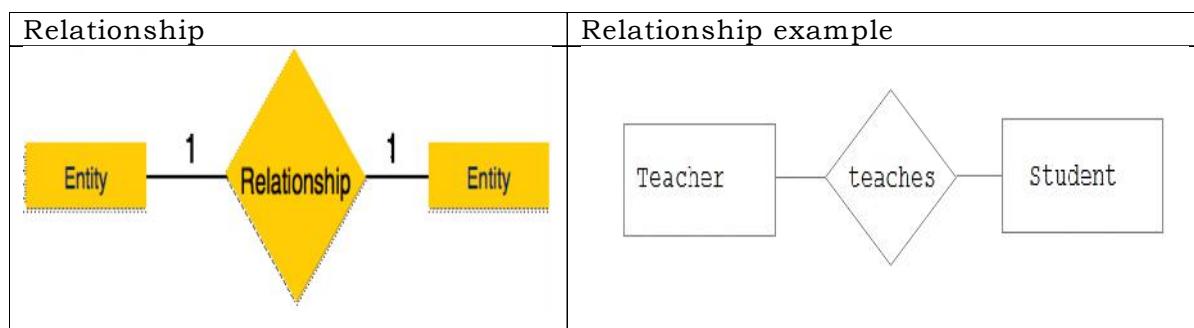


Fig 13.27 Relationship with example

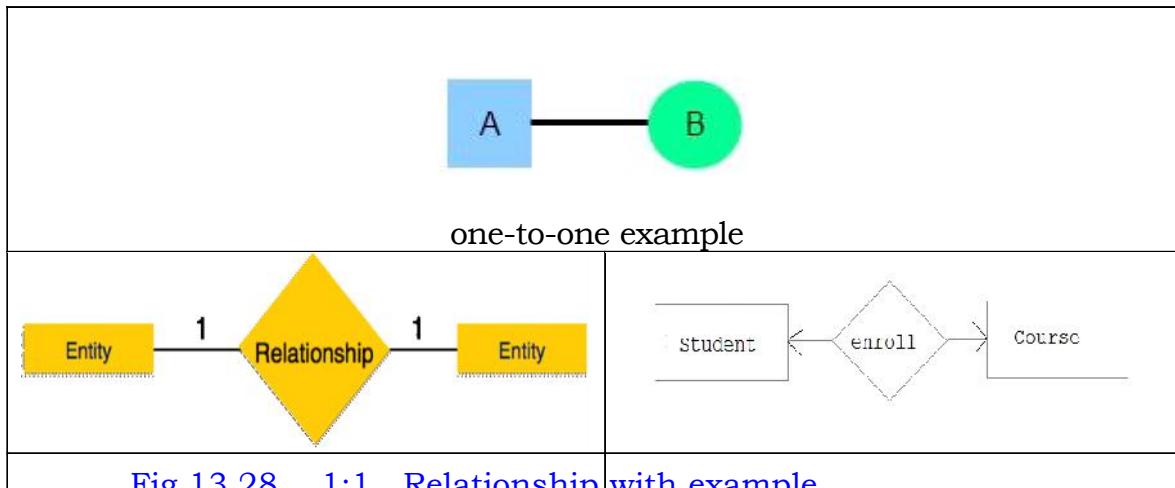
There are three types of relationship that exist between Entities.

- * Binary Relationship
- * Recursive Relationship
- * Ternary Relationship

Binary Relationship

Binary Relationship means relation between two Entities. This is further divided into three types.

1. One to One : This type of relationship is rarely seen in real world.



The above example describes that one student can enroll only for one course and a course will also have only one Student. This is not what you will usually see in relationship.

2. One to Many : It reflects business rule that one entity is associated with many number of same entity. For example, Student enrolls for only one Course but a Course can have many Students.

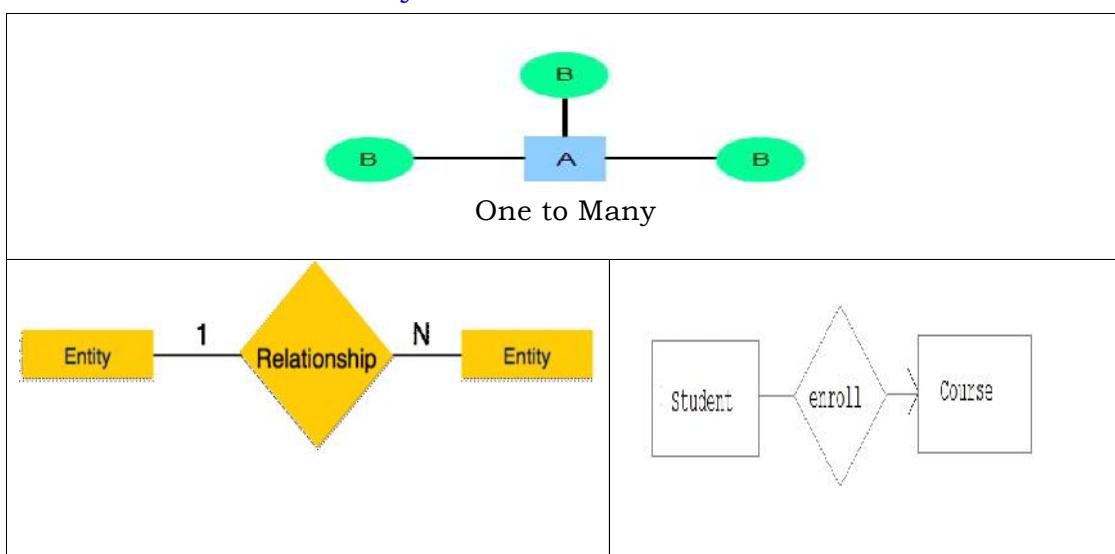


Fig 13.29 1:M Relationship with example

The arrows in the diagram describes that one student can enroll for only one course.

3. Many to Many : The above diagram represents that many students can enroll for more than one courses.

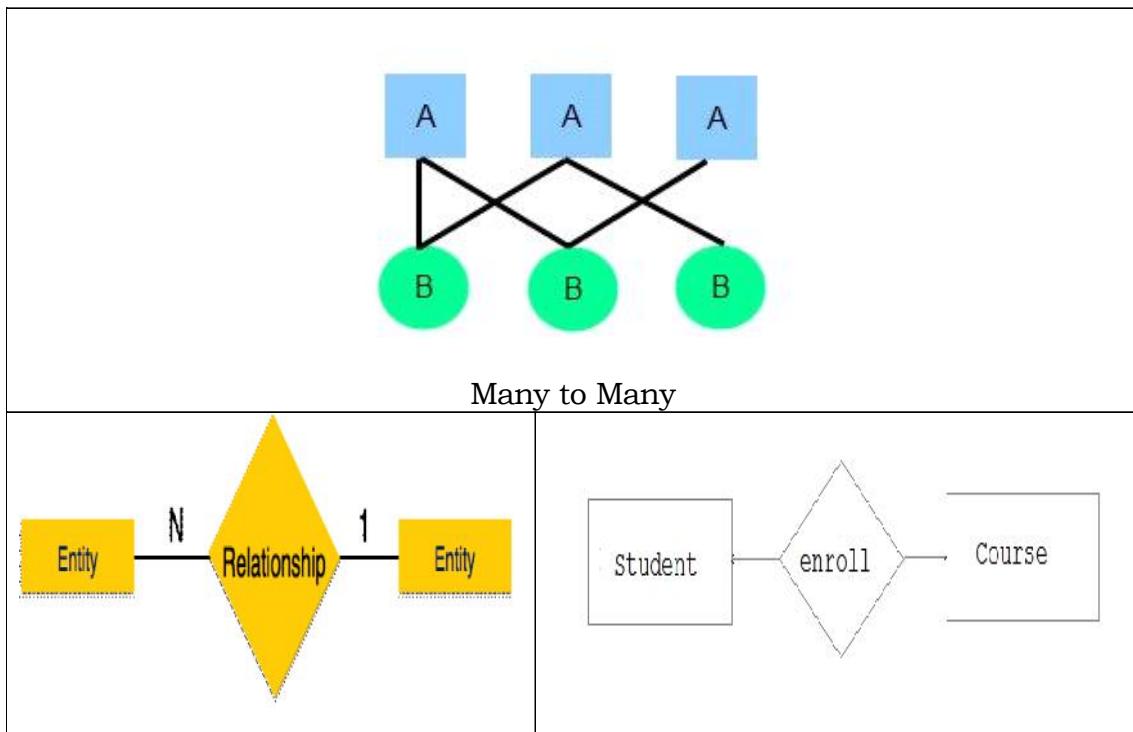


Fig 13.30 M:M Relationship with example

PARTICIPATION CONSTRAINTS

- **Total Participation:** Each entity in the entity is involved in the relationship. Total participation is represented by double lines.
- **Partial participation:** Not all entities are involved in the relationship. Partial participation is represented by single line.

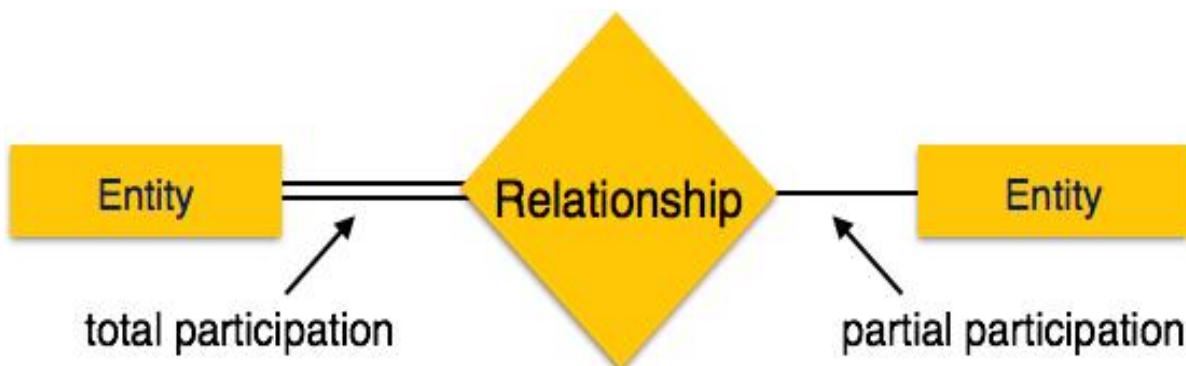
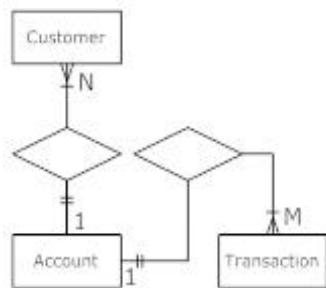


Fig 13.31 Total participation and partial participation

13.13.3 Cardinality



Cardinality specifies how many instances of an entity relate to one instance of another entity.

Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinality specifies the absolute minimum number of relationships.

Fig 13.32 Cardinality

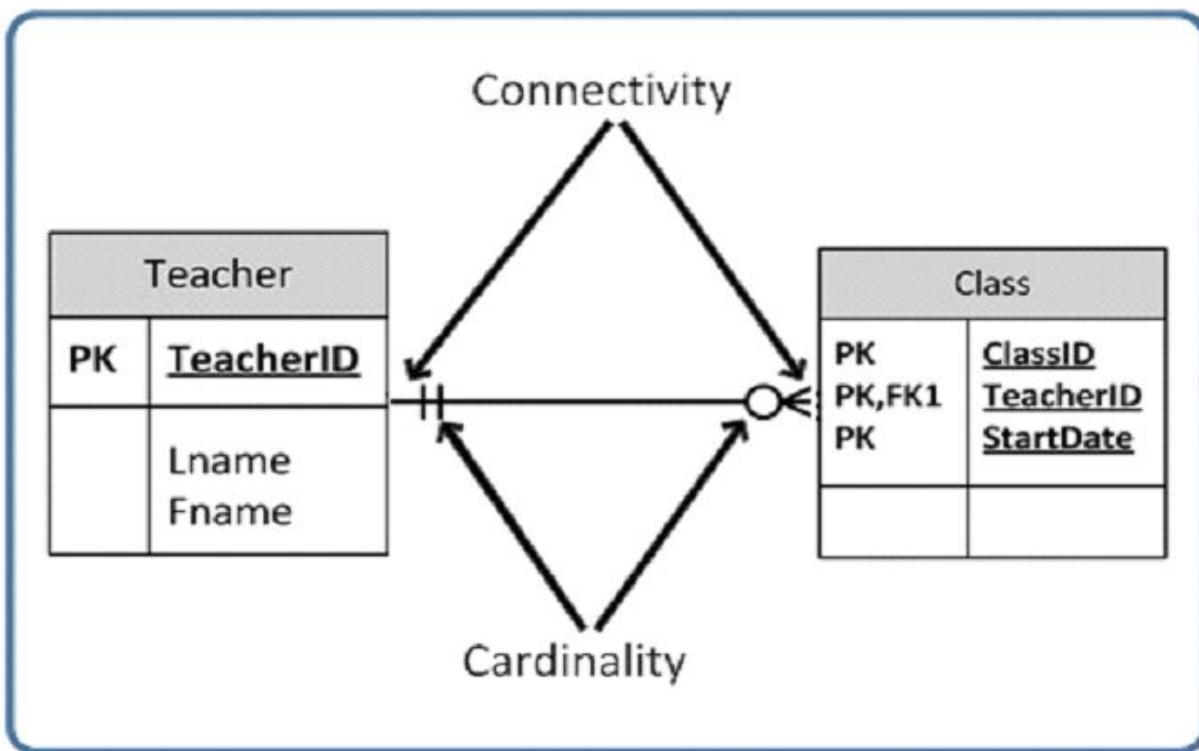


Fig 13.33 Cardinality Notation with examples

Cardinality Notation Cardinality specifies how many instances of an entity relate to one instance of another entity. Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinality specifies the absolute minimum number of relationships. When the minimum number is zero, the relationship is usually called optional and when the minimum number is one or more, the relationship is usually called mandatory.

<p>Information Engineering style</p> <pre> graph TD A[one to one] --- B[one to many (mandatory)] A --- C[many] A --- D[one or more (mandatory)] A --- E[one and only one (mandatory)] A --- F[zero or one (optional)] A --- G[zero or many (optional)] </pre> <p>Company</p> <p>Employee</p> <p>Projects</p>	<p>Chen style</p> <p>M:N</p> <p>Ordinality - describes the minimum (optional vs mandatory)</p> <p>Cardinality - describes the maximum</p> <p>1:N ($n=0,1,2,3\dots$) one to zero or more</p> <p>M:N ($m \text{ and } n=0,1,2,3\dots$) zero or more to zero or more (many to many)</p> <p>1:1 one to one</p>	<pre> graph TD Company[Company] -- 1 --> Employee[Employee] Employee -- N --> M_N{M:N} M_N --> Projects[Projects] </pre>
<p>Bachman style</p> <pre> graph TD A[one to one] --- B[zero or more to one or more] A --- C[one to one or more] </pre> <p>Company</p> <p>Employee</p> <p>Projects</p>	<p>Martin style</p> <p>1 - one, and only one (mandatory)</p> <p>* - many (zero or more - optional)</p> <p>1...* - one or more (mandatory)</p> <p>0...1 - zero or one (optional)</p> <p>(0,1) - zero or one (optional)</p> <p>(1,n) - one or more (mandatory)</p> <p>(0,n) - zero or more (optional)</p> <p>(1,1) - one and only one (mandatory)</p>	<pre> graph TD Company[Company] -- 1 --> Employee[Employee] Employee -- "*" --> M_N{M:N} M_N --> Projects[Projects] </pre>

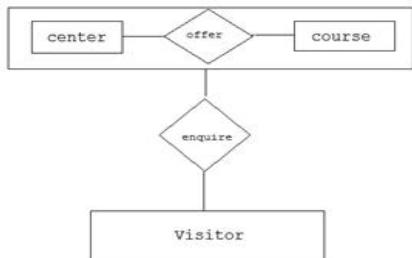
Fig 13.34 Cardinality Notation

Recursive Relationship

Fig 13.35 Recursive Relationship

Generalization	Specialization
Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity. In generalization, the higher level entity can also combine with other lower level entity to make further higher level entity.	*Specialization* is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, some higher level entities may not have lower-level entity sets at all.
<pre> graph TD Account[Account] -- "is A" --> Saving[Saving] Account -- "is A" --> Current[Current] style Account fill:#fff,stroke:#000 style Saving fill:#fff,stroke:#000 style Current fill:#fff,stroke:#000 </pre>	<pre> graph TD Student[Student] -- "is A" --> RxStudent[Rx-Student] Student -- "is A" --> CurrStudent[Curr Student] style Student fill:#fff,stroke:#000 style RxStudent fill:#fff,stroke:#000 style CurrStudent fill:#fff,stroke:#000 </pre>

Fig13.36 Generalization & Specialization



When an Entity is related with itself it is known as Recursive Relationship.

Fig13.37 Aggregation

Aggregation : Aggregation is a process when relation between two entity is treated as a single entity. Here the relation between Center and Course, is acting as an Entity in relation with Visitor.

13.14 Keys

The word “key” is used in the context of relational database design. They are used to establish and identify relation between tables. The key is a set of one or more columns whose combined values are unique among all occurrences in a given table.

Table	Employees			Relation	
Employee_id	Name	Age	city	Salary	Attribute
1	Rajappa	42	Tumkur	42000	
2	Naveen	35	Bidar	35000	Tuple
	2 Ravindra	42	Sagar	45000	
Relation/table	Employees			Domain	
Attribute	Employee_id,Name,age,city,salary				
tuple	2,naveen,35,Bidar,35000				
domain	42000,35000,45000				
candidate Keys	Employee_id,Name				
Primary Key	Employee_id				
Secondary/Alternate key	Name				

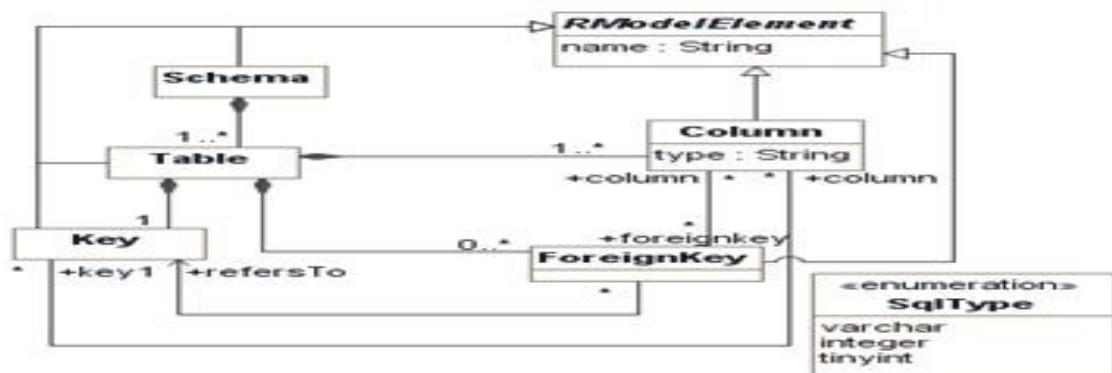


Fig 13.38 Database terms with example

There are various types of relational keys:

1. **Candidate key(ck):** A Candidate key is any set of one or more columns whose combined values are unique among all occurrences(ie tuples or rows). Since a null value is not guaranteed to be unique, no component of a candidate key is allowed to be null.
2. **Primary key(pk):** Primary key is a candidate key that is most appropriate to become the main key of the table. Primary key is a key that uniquely identify each record in a table. Example student_id, Employee_id, Bank_acoount_no, Transfer_certificate_id, Driving_licence_no etc., by which only one value can exist, no duplication can exist.

Primary Key

A table with five columns labeled `s_id`, `s_name`, `age`, `course`, and `address`. A red arrow points to the `s_id` column, indicating it is the primary key.

3. **Alternate key/Secondary key(sk)**: The alternate keys of any table are simply those candidate keys which are not currently selected as the primary key. An alternative key is a function of all candidate keys minus the primary key.
5. **Super Key** : A superkey is basically all sets of columns for which no two rows share the same values for those sets. An attribute or set of attributes that uniquely identifies a tuple within a relation/table. Super Key is a superset of Candidate key.
6. **Foreign key(fk)** :A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables.

Table	Employees				
Employee_id	Name	Age	city	Salary	Car_loan_id
1	Rajappa	42	Tumkur	42000	585

Table	BMWcars				
Car_loan_id	Model	Loanamount	EM	NoofEM	Balance
585	Basic	1800000	80000	225	1000000

Fig 13.39 Foreign key with example

1. **Composite Key** : Key that consists of two or more attributes that uniquely identify an entity occurrence is called Composite key. But any attribute that makes up the Composite key is not a simple key in its own. Example: Consider a Relation or Table R1. Let A,B,C,D,E are the attributes of this relation.



Fig.14.47 Composite Key

$R(A,B,C,D,E)$

A?BCDE This means the attribute 'A' uniquely determines the other attributes B,C,D,E.

BC?ADE This means the attributes 'BC' jointly determines all the other attributes A,D,E in the relation.

Primary Key :A

Candidate Keys :A, BC

Super Keys : A,BC,ABC,AD

Note : ABC,AD are not Candidate Keys since both are not minimal super keys.

13.15 Relational Algebra

In order to implement a DBMS, there must exist a set of rules which state how the database system will behave. For instance, somewhere in the DBMS must be a set of statements which indicate than when someone inserts data into a row of a relation, it has the effect which the user expects. One way to specify this is to use words to write an 'essay' as to how the DBMS will operate, but words tend to be imprecise and open to interpretation. Instead, relational databases are more usually defined using Relational Algebra.

Relational Algebra is :

- the formal description of how a relational database operates
- an interface to the data stored in the database itself
- the mathematics which underpin SQL operations

Operators in relational algebra are not necessarily the same as SQL operators, even if they have the same name. For example, the SELECT statement exists in SQL, and also exists in relational algebra. These two uses of SELECT are not the same. The DBMS must take whatever SQL statements the user types in and translate them into relational algebra operations before applying them to the database.

Terminology

- Relation – a set of tuples.
- Tuple – a collection of attributes which describe some real world entity.
- Attribute – a real world role played by a named domain.
- Domain – a set of atomic values.
- Set – a mathematical definition for a collection of objects which contains no duplicates.

Operators – Write

- INSERT – provides a list of attribute values for a new tuple in a relation. This operator is the same as SQL.
- DELETE – provides a condition on the attributes of a relation to determine which tuple(s) to remove from the relation. This operator is the same as SQL.
- MODIFY – changes the values of one or more attributes in one or more tuples of a relation, as identified by a condition operating on the attributes of the relation. This is equivalent to SQL UPDATE.

Operators – Retrieval

There are two groups of operations:

- Mathematical set theory based relations:
UNION, INTERSECTION, DIFFERENCE, and CARTESIAN PRODUCT.

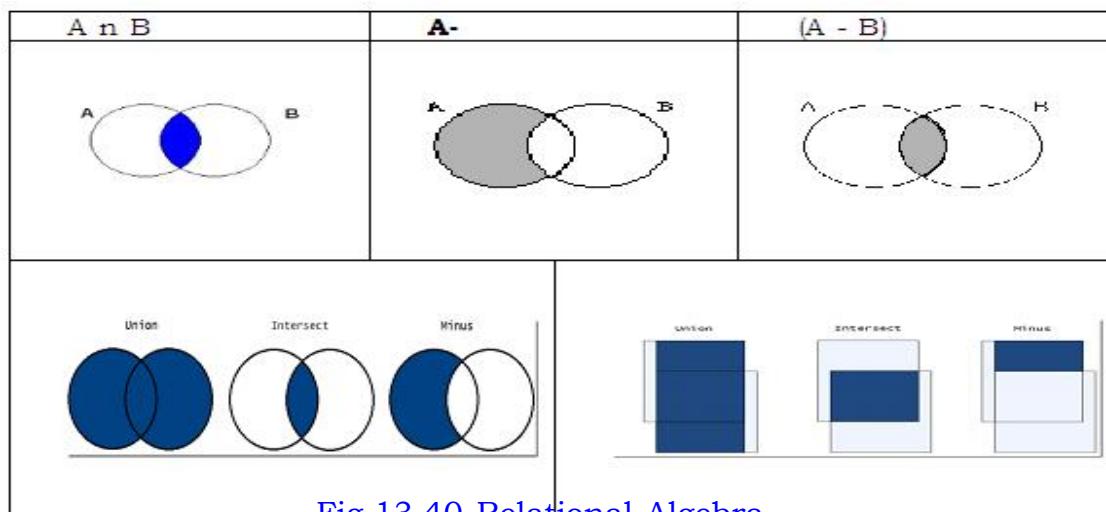


Fig.13.40 Relational Algebra

- Special database operations:
SELECT (not the same as SQL SELECT), PROJECT, and JOIN.

Relational SELECT

SELECT is used to obtain a subset of the tuples of a relation that satisfy a *select condition*.

For example, find all employees born after 14th Feb 2014:

SELECT_{dob > '14/feb/2014'}(employee)

Relational PROJECT

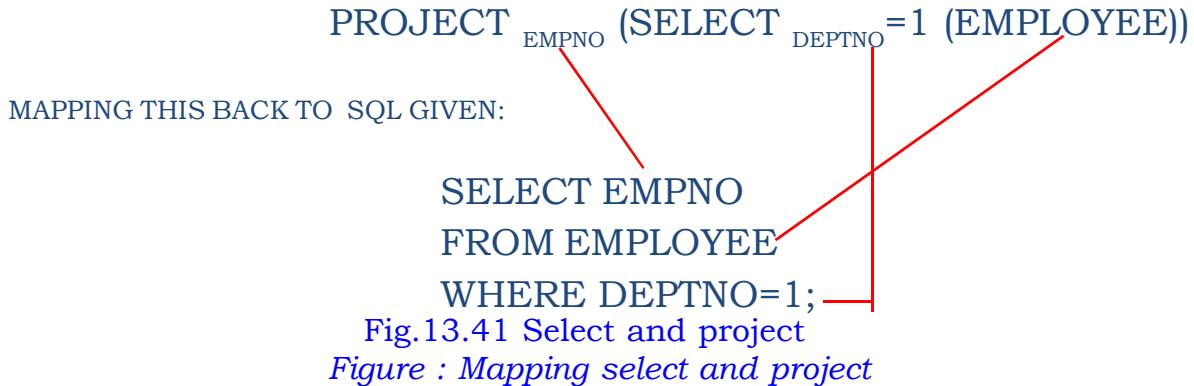
The PROJECT operation is used to select a subset of the attributes of a relation by specifying the names of the required attributes.

For example, to get a list of all employees surnames and employee numbers:

PROJECT_{surname, empno}(employee)

SELECT and PROJECT

SELECT and PROJECT can be combined together. For example, to get a list of employee numbers for employees in department number 1:



Set Operations – semantics

Consider two relations R and S.

- UNION of R and S

the union of two relations is a relation that includes all the tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.

- INTERSECTION of R and S

the intersection of R and S is a relation that includes all tuples that are both in R and S.

- DIFFERENCE of R and S

the difference of R and S is the relation that contains all the tuples that are in R but that are not in S.

SET Operations – requirements

For set operations to function correctly the relations R and S must be union compatible. Two relations are union compatible if

- they have the same number of attributes
- the domain of each attribute in column order is the same in both R and S.

UNION Example

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R UNION S

A	1
B	2
C	2
D	3
E	5
F	4
E	4

Fig.13.42 Union

INTERSECTION Example

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R INTERSECTION S

A	1
D	3

Fig.13.43 Intersection

DIFFERENCE Example

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R DIFFERENCE S

B	2
F	4
E	5

S DIFFERENCE R

C	2
E	4

Fig.13.44 Difference

CARTESIAN PRODUCT

The Cartesian Product is also an operator which works on two sets. It is sometimes called the CROSS PRODUCT or CROSS JOIN.

It combines the tuples of one relation with all the tuples of the other relation.

CARTESIAN PRODUCT example

R	R CROSS S						
A	1	A	1	F	4	A	1
B	2	C	2	F	4	C	2
D	3	D	3	F	4	D	3
F	4	E	4	F	4	E	4
E	5	A	1	E	5	A	1
		B	2	E	5	C	2
		B	2	D	5	D	3
		B	2	E	5	E	4
		D	3	A	5		
		D	3	C			
		D	3	D			
		D	3	E			

Fig.13.45 Cartesian product

Figure : CARTESIAN PRODUCT

JOIN Operator

JOIN is used to combine related tuples from two relations:

- In its simplest form the JOIN operator is just the cross product of the two relations.
- As the join becomes more complex, tuples are removed within the cross product to make the result of the join more meaningful.
- JOIN allows you to evaluate a join condition between the attributes of the relations on which the join is undertaken.

The notation used is

R JOIN_{join condition} S

R	ColA	ColB
A	1	
B	2	
D	3	
F	4	
E	5	

JOIN Example

R JOIN_{R.ColA = S.SColA} S

A	1	A	1
D	3	D	3
E	5	E	4

S	SColA	SColB
A	1	
C	2	
D	3	
E	4	

R JOIN_{R.ColB = S.SColB} S

A	1	A	1
B	2	C	2
D	3	D	3
F	4	E	4

Figure : JOIN

Fig.13.46 Join

Natural Join

Invariably the JOIN involves an equality test, and thus is often described as an equi-join. Such joins result in two attributes in the resulting relation having exactly the same value. A ‘natural join’ will remove the duplicate attribute(s).

- In most systems a natural join will require that the attributes have the same name to identify the attribute(s) to be used in the join. This may require a renaming mechanism.
- If you do use natural joins make sure that the relations do not have two attributes with the same name by accident.

OUTER JOINS

Notice that much of the data is lost when applying a join to two relations. In some cases this lost data might hold useful information. An outer join retains the information that would have been lost from the tables, replacing missing data with nulls.

There are three forms of the outer join, depending on which data is to be kept.

- LEFT OUTER JOIN – keep data from the left-hand table
- RIGHT OUTER JOIN – keep data from the right-hand table
- FULL OUTER JOIN – keep data from both tables

OUTER JOIN example 1

<i>ColB</i>	R LEFT OUTER JOIN R.ColA = S.SColA				<i>ColA</i>
1	A	1	A	1	
2	D	3	D	3	
3	E	5	E	4	
4	B	2	-	-	
5	F	4	-	-	

<i>SColB</i>	R RIGHT OUTER JOIN R.ColA = S.SColA				<i>SColA</i>
1	A	1	A	1	
2	D	3	D	3	
3	E	5	E	4	
4	-	-	C	2	

Fig.13.47 Outer Join left/right *OUTER JOIN (left/right)*

JOIN example 2			
CoB	R FULL OUTER JOIN		S
	R.Cola = S.SColA	S_R	CoLA
1	A	1	A
2	D	3	B
3	E	5	D
4	B	2	F
5	F	4	E
-	-	C	-
		2	-

SCoB	S	SCoLA
1		A
2		C
3		D
4		E

Fig.13.48 Outer Join Full

OUTER JOIN (full)

Consider the following SQL to find which departments have had employees on the 'Further Accounting' course.

```
SELECT DISTINCT dname
FROM department, course, empcourse, employee
WHERE cname = 'Further Accounting'
AND course.courseno = empcourse.courseno
AND empcourse.empno = employee.empno
AND employee.depno = department.depno;
```

The equivalent relational algebra is

```
PROJECTdname (department JOINdepno = depno (
PROJECTdepno (employee JOINempno = empno (
PROJECTempno (empcourse JOINcourseno = courseno (
PROJECTcourseno (SELECTcname = Further Accounting, course) ))) )) )
```

Symbolic Notation

From the example, one can see that for complicated cases a large amount of the answer is formed from operator names, such as PROJECT and JOIN. It is therefore commonplace to use symbolic notation to represent the operators.

- SELECT -> σ (sigma)
- PROJECT -> $\delta(\pi)$
- PRODUCT -> \times (times)
- JOIN -> $| \times |$ (bow-tie)

- UNION -> *" (cup)
- INTERSECTION ->)" (cap)
- DIFFERENCE -> - (minus)
- RENAME -> ñ (rho)

Usage

The symbolic operators are used as with the verbal ones. So, to find all employees in department 1:

$\text{SELECT}_{\text{depno} = 1}(\text{employee})$
 becomes $\delta_{\text{depno} = 1}(\text{employee})$

Conditions can be combined together using ^ (AND) and v (OR). For example, all employees in department 1 called 'URS':

$\text{SELECT}_{\text{depno} = 1 \wedge \text{surname} = \text{'URS'}}(\text{employee})$
 becomes $\delta_{\text{depno} = 1 \wedge \text{surname} = \text{'URS'}}(\text{employee})$

The use of the symbolic notation can lend itself to brevity. Even better, when the JOIN is a natural join, the JOIN condition may be omitted from |x|. The earlier example resulted in:

$\text{PROJECT}_{\text{dname}} (\text{department} \text{ JOIN}_{\text{depno} = \text{depno}} ($
 $\text{PROJECT}_{\text{depno}} (\text{employee} \text{ JOIN}_{\text{empno} = \text{empno}} ($
 $\text{PROJECT}_{\text{empno}} (\text{empcourse} \text{ JOIN}_{\text{courseno} = \text{courseno}} ($
 $\text{PROJECT}_{\text{courseno}} (\text{SELECT}_{\text{cname} = \text{'Further Accounting'}}$
 $\text{course)))))))$

becomes

$\delta_{\text{dname}} (\text{department} | \times | ($
 $\delta_{\text{depno}} (\text{employee} | \times | ($
 $\delta_{\text{empno}} (\text{empcourse} | \times | ($
 $\delta_{\text{courseno}} (\delta_{\text{cname} = \text{'Further Accounting'}} \text{ course})))))$

Rename Operator

The rename operator returns an existing relation under a new name. $\tilde{n}_A(B)$ is the relation B with its name changed to A. For example, find the employees in the same Department as employee 3.

$$\begin{aligned} \tilde{N}_{\text{emp2.surname}, \text{emp2.forenames}} & (\\ \delta_{\text{employee.empno} = 3 \wedge \text{employee.depno} = \text{emp2.depno}} & (\\ \text{employee} \times (\tilde{n}_{\text{emp2}} \text{employee}) &)) \end{aligned}$$

Derivable Operators

- Fundamental operators: δ , σ , \times , * , $-$, \tilde{n}
- Derivable operators: $| \times |,)"$

Equivalence

$$A | \times |_c B \hat{=} \delta_{a1,a2,...aN}(\sigma_c(A \times B))$$

- where c is the join condition (eg $A.a1 = B.a1$),
- and $a1,a2,...aN$ are all the attributes of A and B without repetition.

C is called the join-condition, and is usually the comparison of primary and foreign key. Where there are N tables, there are usually N-1 join-conditions. In the case of a natural join, the conditions can be missed out, but otherwise missing out conditions results in a ptimizat product (a common mistake to make).

Equivalences

The same relational algebraic expression can be written in many different ways. The order in which tuples appear in relations is never significant.

- $A \times B \hat{=} B \times A$
- $A)" B \hat{=} B)" A$
- $A {}^* B \hat{=} B {}^* A$
- $(A - B)$ is not the same as $(B - A)$

- $\delta_{c_1} (\delta_{c_2}(A)) \hat{=} \delta_{c_2} (\delta_{c_1}(A)) \hat{=} \delta_{c_1 \wedge c_2}(A)$
- $\delta_{a_1}(A) \hat{=} \delta_{a_1}(\delta_{a_1, \text{etc}}(A))$

where etc represents any other attributes of A.

- many other equivalences exist.

While equivalent expressions always give the same result, some may be much easier to evaluate than others.

When any query is submitted to the DBMS, its query optimizer tries to find the most efficient equivalent expression before evaluating it.

Comparing RA and SQL	
Relational algebra:	SQL:
<ul style="list-style-type: none"> • Is closed (the result of every expression is a relation) • Has a rigorous foundation • Has simple semantics • Is used for reasoning, query optimization, etc. 	<ul style="list-style-type: none"> • Is a superset of relational algebra • Has convenient formatting features, etc. • Provides aggregate functions • Has complicated semantics • Is an end-user language
<p>Any relational language as powerful as relational algebra is called relationally complete.</p> <p>A relationally complete language can perform all basic, meaningful operations on relations.</p>	SQL is a superset of relational algebra, it is also relationally complete.

13.16 Data warehouse

A data warehouse is a repository of an organization's electronically stored data. Data warehouses are designed to facilitate reporting and supporting data analysis. The concept of data warehouses was introduced in late 1980's. The concept was introduced to meet the growing demands for management information and analysis that could not be met by operational systems.

Separate computer databases began to be built that were specifically designed to support management information and analysis purposes. These data warehouse

were able to bring in data from a range of different data sources, such as, mainframe computer, minicomputer, as well as personal computer and office automation software such as spreadsheets and integrate this information in a single place. This capability, coupled with user-friendly reporting tools, and freedom from operational impacts has led to a growth of this type of computer system.

Data ware house have evolved though several fundamental stages like:

Offline operational databases – Data warehouse in this initial stage are developed by simply copying the database of an operational system to an off-line server where the processing load of reporting does not impact on the operational system's performance.

Offline data warehouse – Database warehouses in this stages of evolution are updated on regular time cycle(usually daily, weekly or monthly) form operational systems and the data is stored in a integrated reporting-oriented data structure.

Real Time data warehouse – Data warehouses are updated on transaction or event basis, event time an operational system performs a transaction.

Integrated data warehouses – Data warehouses used to generate activity or transactions that are passed back into the operational systems for use in the daily activity of the organization.

Components of data warehouses

Data Sources: Data sources refer to any electronic repository of information that contains data of interest for management use or analytics. From mainframe(IBM DB2,ISAM,Adabas, etc.), client-server databases (e.g Oracle database, Informix, Microsoft SQL Server etc.,), PC databases (e.g Microsoft Access), and ESS and other electronic store of data. Data needs to be passed from these to systems to the data warehouse either on the transaction-by-transaction basis for real-time data warehouses or on a regular cycle(e.g daily or weekly) for offline data warehouses.

Data transformation: The data transformation layer receives data from the data sources, cleaned and standardizes and loads it into the data repository. This is often called “staging” data as data often passes through a temporary database whilst it is being transformed. This activity of transformation data can be performed either by manually created code or a specific type of software could be used called an Extract, Transform and load(ETL) tool.

Reporting : The data in the data warehouses must be available to the organization's staff if the data warehouses is to be useful. There are a very large number of applications that perform this function or reporting can be custom-developed. Some are Business intelligence tools, Executive information systems, Online Analytical processing(OLAP) Tools, Data Mining etc.,

Metadata: Metadata or “Data about data” is used to inform operators and users of the data warehouses about its status and the information held within the data warehouses.

Operations : Data warehouses operations comprises of the processes of loading, manipulating and extracting data from the data warehouse. Operations also cover users management security, capacity management and related functions.

Optional components: In addition the following components also exist in some data warehouses: 1. Dependent data marts. 2. Logical data marts. 3. Operational data store.

Advantages of data warehouses:

1. Enhance end-user access to reports and analysis of information.
2. Increases data consistency.
3. Increases productivity and decreases computing costs.
4. Able to combine data from different sources, in one place.
5. Data warehouses provide an infrastructure that could support changes to data and replication of the changed data back into the operational systems.

Disadvantages

Extracting, cleaning and loading data could be time consuming.

Data warehouses can get outdated relatively quickly.

Problems with compatibility with systems already in place.

Providing training to end-users.

Security could develop into a serious issue, especially if the data warehouses is internet accessible.

A data warehouses is usually not static and maintenance costs are high.

13.17 Data Mining : Data mining is concerned with the analysis and picking out relevant information. It is the computer, which is responsible for finding the patterns by identifying the underlying rules of the features in the data.

Data mining analysis tends to work from the data up and the best techniques are those developed with an orientation towards large volumes of data, making use of as much of the collected data as possible to arrive at reliable conclusions and decisions.

The analysis process starts with a set of data, uses a methodology to develop an optimal representation to the structure of the data during which time knowledge is acquired. Once knowledge has been acquired this can be extended to larger sets of data working on the assumption that the larger data set has a structure similar to the sample data. Again this is analogous to a mining operation where large amounts of low grade materials are sifted through in order to find something of value.

Some of the data mining software's are SPSS, SAS, Think Analytics and G-Sat etc.

The phases start with the raw data and finish with the extracted knowledge which was acquired as a result of the following stages:

Selection- Selecting or segmenting the data according to some criteria e.g. all those people who won a car, in this way subsets of the data can be determined.

Preprocessing – This is the data cleaning stage where certain information is removed which deemed unnecessary and may slow down queries for e.g.: gender of the patient. The data is reconfigured to ensure a consistent format as there is a possibility of inconsistent formats because the data is drawn from several sources e.g. gender may be recorded as F or M also as 1 or 0.

Transformation – The data is not merely transferred, but transformed. E.g.: demographic overlays commonly used in market research. The data is made useable and navigable.

Data mining- This stage is concerned with the extraction of patterns from the data. A pattern can be defined as given a set of facts(data) F, a language L, and some measure of certainty C a pattern is a statement S in L that describes relationships among a subset Fs of F with a certainly c such that S is simpler in some sense than the enumeration of all the facts in Fs.

Interpretation and Evaluation – The patterns identified by the system are interpreted into knowledge which can be used to support human decision-making e.g. prediction and classification tasks, summarizing the content of a database or explaining observed phenomena.

Summary

- > The basic concepts of database that can be used by various users to store and retrieve thaе data in standardized format.
- > DBMS features, parts, problems and solutions.
- > Three database structures.
- > Entity relations.
- > Various relationships.
- > Keys
- > Database warehouse, Data mining

Review questions**One mark questions**

1. What is data?
2. What is information?
3. What is database?
4. What is a field?
5. What is a record?
6. What is an entity?
7. What is an instance?
8. What is an attribute?
9. What is domain?
10. What is a relation?
11. What is a table?
12. What is normalization?
13. What is a key?
14. Give the symbol notation for project.
15. What is data mining?

Two marks questions

1. How database helps us?
2. How do we get data?
3. Name the data types supported by DBMS.
4. What is generalization?
5. What is specification?
6. What is the difference between serial and direct access file organization?
7. Give the advantages and disadvantages of Index Sequential Access Method.
8. Classify various types of keys used in Database.
9. What is relation Algebra?
10. Give an example for relation **selection** with example.
11. What is Cartesian product?
12. What is Join operation?
13. What is data warehouse?
14. What is data mining?

Three marks questions

1. Mention the applications of database.
2. List different forms of data(any three)
3. Give the difference between Manual and Electronic file systems.
4. Explain Boyce and Codd form (BCNF)
5. Explain any three components of E-R diagram.
6. What is a relationship? Classify and give example.
7. Explain physical data independence.
8. Explain ISAM with example
- 9.Explain database users
- 10.Explain hierarchical data model.
- 11.Explain relational data model.
- 12.Explain outer Join with example.
- 13.List the components of data warehouse.

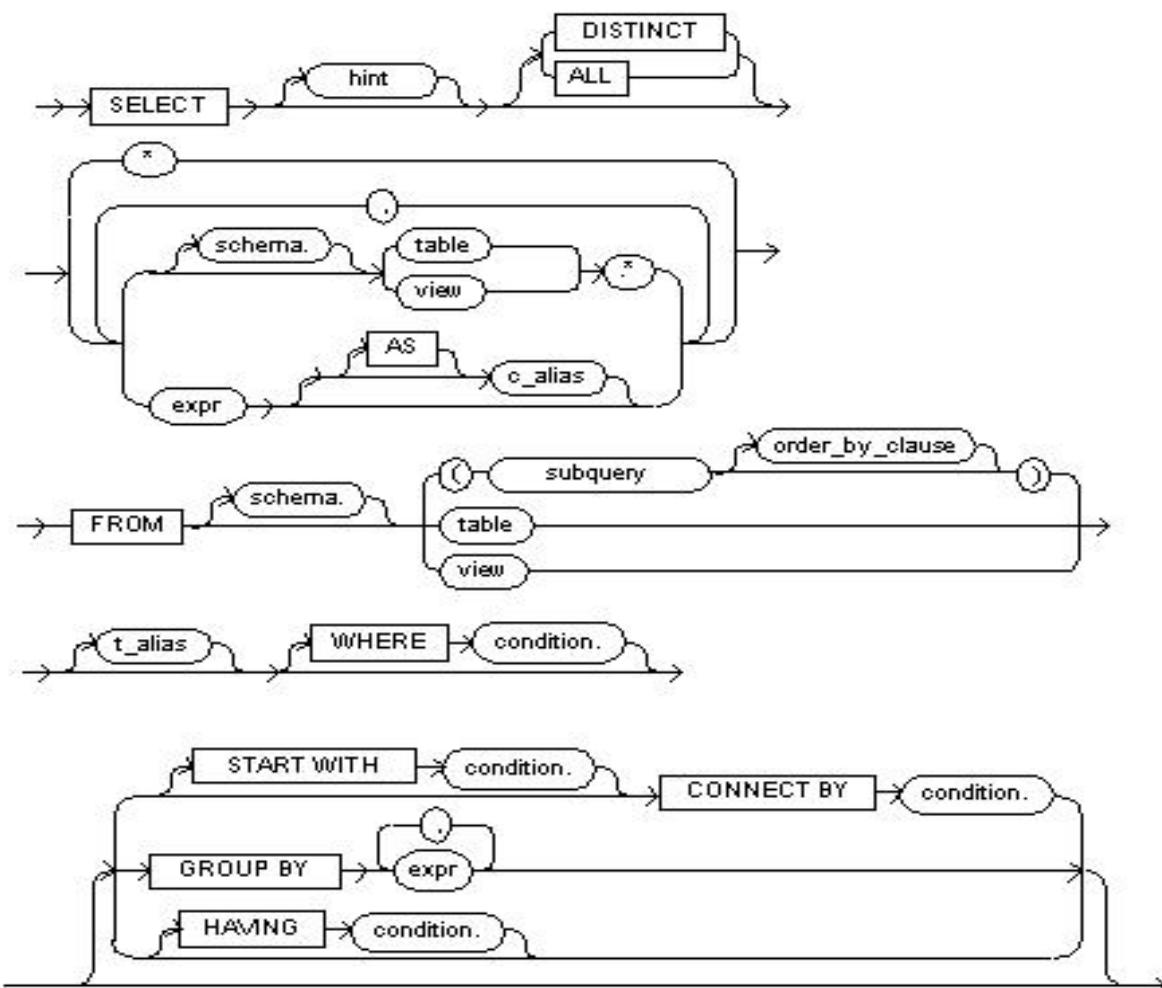
Five marks questions

1. Explain data processing cycle?
2. Explain varius datatypes used in DBMS?
3. Explain Normalization with classifications and example
4. Explain cardinality with example.
5. Explain data independence in detail.
6. Discuss file organization with respect to physical data independence.
7. Explain the features of database system.
8. Explain DBMS Architecture.
9. Explain database model.
- 10.Explain Codd's rules for database management.
- 11.Write comprising RA and SQL.
- 12.List any five types of relational keys.
- 13.Explain Entity-Relationship in detail.
14. Explain the concept of Data abstraction.
- 15.Define and explain the phases of data mining.

CHAPTER 14

SQL Commands**OBJECTIVES**

- To understand SQL commands usage.
- Learning the data types, expressions, operators.
- The use of syntax and constraints for SQL.
- Commands for DDL and DML.
- Various built in functions SQL.



14.1 Introduction:

Structured Query Language helps to make practice on SQL commands which provides immediate results. SQL is a language of database, it includes database

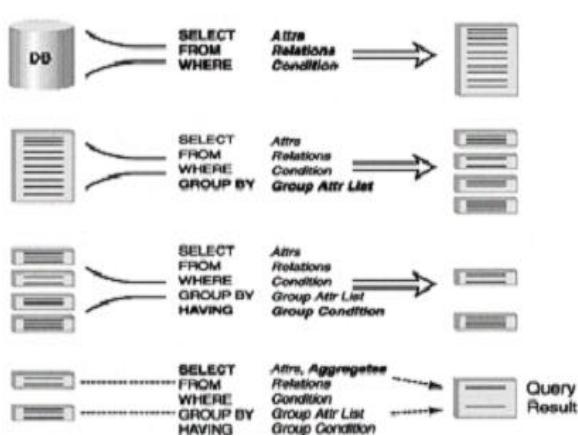


Fig 14.1 Database to Query creation, deletion, fetching rows and modifying rows etc.

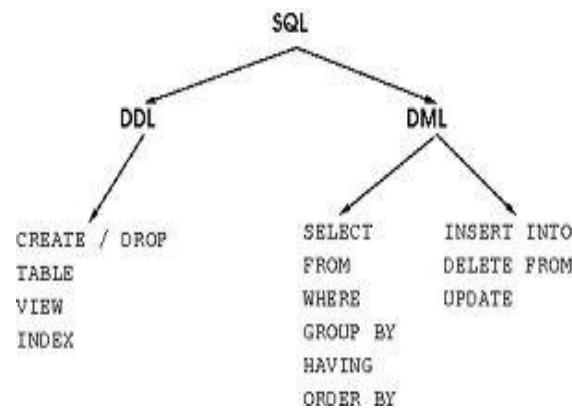


Fig 14.2 SQL languages classifications

SQL is an ANSI (American National Standards Institute) standard but there are many different versions of the SQL language.

SQL is Structured Query Language, which is a dbase language for storing, manipulating and retrieving data stored in relational database.

SQL is the standard language for Relation Database System. All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, postgresql and SQL Server use SQL as standard database language.

Also, they are using different dialects, such as:

- MS SQL Server using T-SQL,
- Oracle using PL/SQL,
- MS Access version of SQL is called JET SQL (native format) etc.
- Allows users to access data in relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in database and manipulate that data.
- Allows embedding within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures, and views

History:

- **1970:** Dr. E. F. "Codd" of IBM is known as the father of relational databases. He described a relational model for databases.
- **1974:** Structured Query Language appeared.
- **1978:** IBM worked to develop Codd's ideas and released a product named System/R.
- **1986:** IBM developed the first prototype of relational database and standardized by ANSI. The first relational database was released by Relational Software and its later becoming Oracle.

14.1.1 SQL ARCHITECTURE:

When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.

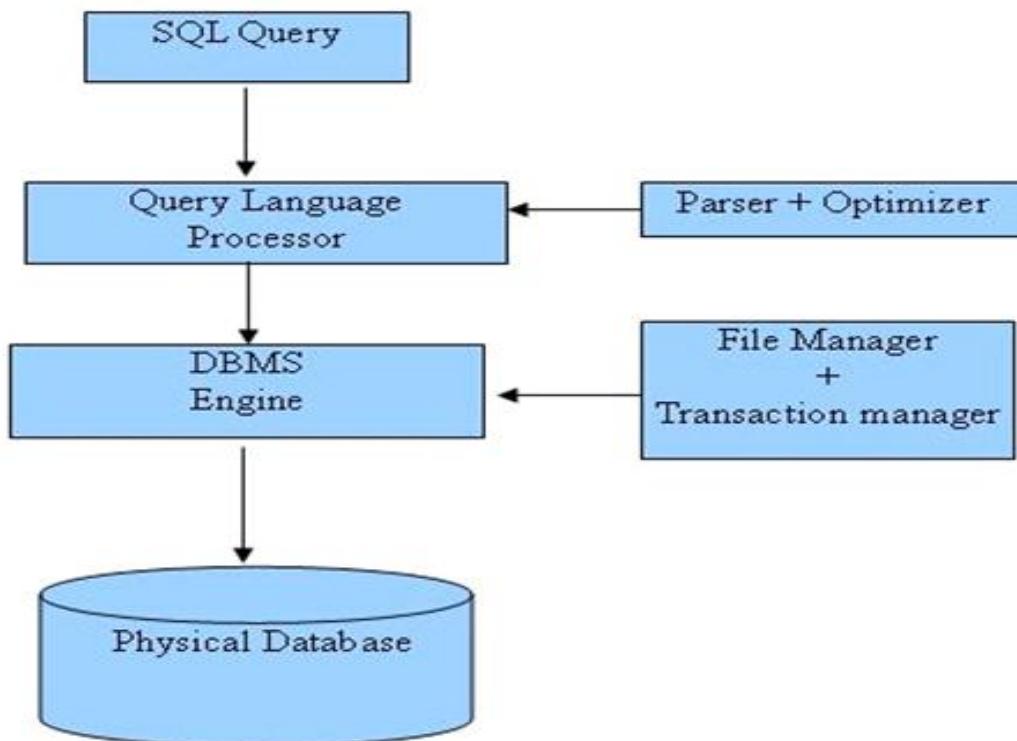


Fig 14.3 SQL Architecture with different layers

There are various components included in the process. These components are Query Dispatcher, Optimization Engines, Classic Query Engine and SQL Query Engine, etc. Classic query engine handles all non-SQL queries but SQL query engine won't handle logical files.

MySQL MySQL is an open source SQL database, which is developed by Swedish company MySQL AB. MySQL is pronounced "my ess-que-ell," in contrast with SQL, pronounced "sequel." MySQL is supporting many different platforms including Microsoft Windows, the major Linux distributions, UNIX, and Mac OS X. MySQL has free and paid versions, depending on its usage (non-commercial/commercial) and features. MySQL comes with a very fast, multi-threaded, multi-user, and robust SQL database server.	MS SQL Server MS SQL Server is a Relational Database Management System developed by Microsoft Inc. Its primary query languages are: T-SQL, ANSI SQL.	ORACLE It is a very large and multi-user database management system. Oracle is a relational database management system developed by 'Oracle Corporation'. Oracle works to efficiently manage its resource, a database of information, among the multiple clients requesting and sending data in the network. It is an excellent database server choice for client/server computing. Oracle supports all major operating systems for both clients and servers, including MSDOS, NetWare, UnixWare, OS/2 and most UNIX flavors.	MS ACCESS This is one of the most popular Microsoft products. Microsoft Access is an entry-level database management software. MS Access database is not only an inexpensive but also powerful database for small-scale projects. MS Access uses the Jet database engine, which utilizes a specific SQL language dialect (sometimes referred to as Jet SQL). MS Access comes with the professional edition of MS Office package. MS Access has easy-to-use intuitive graphical interface.
--	--	---	---

History: Development of MySQL by Michael Widenius & David Axmark beginning in 1994. First internal release on 23 May 1995. Windows version was released on 8 January 1998 for Windows 95 and NT. Version 3.23: beta from June 2000, production release January 2001. Version 4.0: beta from August 2002, production release March 2003 (unions). Version 4.01: beta from August 2003, Jyoti adopts MySQL for database tracking. Version 4.1: beta from June 2004, production release October 2004. Version 5.0: beta from March 2005, production release October 2005. Sun Microsystems acquired MySQL AB on 26 February 2008. Version 5.1: production release 27 November 2008.	History: 1987 - Sybase releases SQL Server for UNIX. 1988 - Microsoft, Sybase, and Aston-Tate port SQL Server to OS/2. 1989 - Microsoft, Sybase, and Aston-Tate release SQL Server 1.0 for OS/2. 1990 - SQL Server 1.1 is released with support for Windows 3.0 clients. Aston-Tate drops out of SQL Server development. 2000 - Microsoft releases SQL Server 2000. 2001 - Microsoft releases XML for SQL Server Web Release 1 (download). 2002 - Microsoft releases SQLXML 2.0 (renamed from XML for SQL Server). 2002 - Microsoft releases SQLXML 3.0. 2005 - Microsoft releases SQL Server 2005 on November 7th, 2005.	History: Oracle began in 1977 and celebrating its 32 wonderful years in the industry (from 1977 to 2009). 1977 - Larry Ellison, Bob Miner and Ed Oates founded Software Development Laboratories to undertake development work. 1979 - Version 2.0 of Oracle was released and it became first commercial relational database and first SQL database. The company changed its name to Relational Software Inc. (RSI). 1981 - RSI started developing tools for Oracle. 1982 - RSI was renamed to Oracle Corporation. 1983 - Oracle released version 3.0, rewritten in C language and ran on multiple platforms. 1984 - Oracle version 4.0 was released. It contained features like concurrency control - multi-version read consistency, etc. 1985 - Oracle version 4.0 was released. It contained features like concurrency control - multi-version read consistency, etc. 2007 - Oracle has released Oracle11g. The new version focused on better partitioning, easy migration etc.	History: 1992 - Access version 1.0 was released. 1993 - Access 1.1 released to improve compatibility with inclusion the Access Basic programming language. The most significant transition was from Access 97 to Access 2000. 2010 - Access 2010, a new database format was introduced ACCDB which supports complex data types such as multi-valued and attachment fields.
--	---	---	--

Features: High Performance. High Availability. Scalability and Flexibility Run anything. Robust Transactional Support. Web and Data Warehouse Strengths. Strong Data Protection. Comprehensive Application Development. Management Ease. Open Source Freedom and 24 x 7 Support. Lowest Total Cost of Ownership.	Features: High Performance. High Availability. Database mirroring. Database snapshots. CLR integration. Service Broker. DDL triggers. Ranking functions. Row version-based isolation levels. XML integration. TRY...CATCH. Database Mail.	Features: Concurrency Read Consistency Locking Mechanisms Quiescent Database Portability Self-managing database SQL*Plus ASM Scheduler Resource Manager Data Warehousing Materialized views Bitmap indexes Table compression Parallel Execution Analytic SQL Data mining Partitioning	Features: Users can create tables, queries, forms and reports and connect them together with macros. The import and export of data to many formats including Excel, Outlook, ASCII, dBase, Paradox, FoxPro, SQL Server, Oracle, ODBC, etc. There is also the Jet Database format (MDB or ACCDB in Access 2007), which can contain the application and data in one file. This makes it very convenient to distribute the entire application to another user, who can run it in disconnected environments. Microsoft Access offers parameterized queries. These queries and Access tables can be referenced from other programs like VB6 and .NET through DAO or ADO. The desktop editions of Microsoft SQL Server can be used with Access as an alternative to the Jet Database Engine. Microsoft Access is a file server- based database. Unlike client-server relational database management systems (RDBMS), Microsoft Access does not implement database triggers, stored procedures, or transaction logging.
---	---	--	--

14.2 SQL Commands:

The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into groups based on their nature:

14.2.1 DDL - Data Definition Language:

DDL defines the conceptual schema providing a link between the logical (the way the user views the data) and the physical (the way in which the data is stored physically) structures of the database. The logical structure of a database is a schema. A subschema is the way a specific application views the data form the database.

Following are the functions of the Data Definition Language (DDL):-

1. DDL define the physical characteristics of each record, filed in the record, field's data type, field's length, field's logical name and also specify relationships among those records.
2. DDL describes the schema and subschema.
3. DDL indicates the keys of the records.
4. DDL provides means for associating related records or fields.
5. DDL provides data security measures.
6. DDL provides for the logical and physical data independence.

Few of the basic commands for DDL are :-

Command	Description
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database.

14.2.2 DML - Data Manipulation Language:

1. DML provides the data manipulation techniques like selection, insertion, deletion, update, modification, replacement, retrieval, sorting and display of data or records.
2. DML facilitates use of relationship between the records.
3. DML enables the user and application program to be independent of the physical data structures and database structures maintenance by allowing to process data on a logical and symbolic basis rather than a physical on a physical location basis.
4. DML provide for independence of programming languages by supporting several high-level programming languages like COBOL,PL/1 and C++.

Few of the basic commands for DML are :-

Command	Description
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

DCL - Data Control Language:

Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

DQL - Data Query Language:

Command	Description
SELECT	Retrieves certain records from one or more tables

14.3 Data types in SQL

SQL data type is an attribute that specifies type of data of any object. Each column, variable and expression has related data type in SQL.

You would use these data types while creating your tables. You would choose a particular data type for a table column based on your requirement.

SQL Server offers six categories of data types for your use:

14.3.1 Exact Numeric Data Types:

DATA TYPE	FROM	TO
Int	-2,147,483,648	2,147,483,647
numeric	-10^38 +1	10^38 -1

14.3.2 Floating point numeric Data Types:

DATA TYPE	FROM	TO
Float	-1.79E + 308	1.79E + 308
Real	-3.40E + 38	3.40E + 38

14.3.3 Date and Time Data Types:

DATA TYPE	FROM	TO
datetime	Jan 1, 1753	Dec 31, 9999
Date	Stores a date like MARCH 26, 2014	
Time	Stores a time of day like 12:30 P.M.	

Note: Here, datetime has 3.33 milliseconds accuracy whereas small datetime has 1 minute accuracy.

14.3.4 Character, Strings Data Types:

DATA TYPE	FROM	TO
Char	char	Maximum length of 8,000 characters.(Fixed length non-Unicode characters)
varchar	varchar	Maximum of 8,000 characters.(Variable-length non-Unicode data).

14.4 Operator in SQL

An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations.

Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

- Arithmetic operators
- Comparison operators
- Logical operators
- Operators used to negate conditions

14.4.1 SQL Arithmetic Operators:

Assume variable a holds 10 and variable b holds 20, then:

Operator	Description	Example
+	Addition - Adds values on either side of the operator	a + b will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	a - b will give -10
*	Multiplication - Multiplies values on either side of the operator	a * b will give 200
/	Division - Divides left hand operand by right hand operand	b / a will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	b % a will give 0

14.4.2 Comparison Operators:

Assume variable a holds 10 and variable b holds 20, then:

Show Examples

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(a = b) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a != b) is true.
<>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a <> b) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) is true.
!<	Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true.	(a !< b) is false.
!>	Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true.	(a !> b) is true.

14.4.3 Logical Operators:

Here is a list of all the logical operators available in SQL.

Show Examples

Operator	Description
ALL	The ALL operator is used to compare a value to all values in another value set.
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
ANY	The ANY operator is used to compare a value to any applicable value in the list according to the condition.
BETWEEN	The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
EXISTS	The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria.

IN	The IN operator is used to compare a value to a list of literal values that have been specified.
LIKE	The LIKE operator is used to compare a value to similar values using wildcard operators.
NOT	The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. This is a negate operator.
OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
IS NULL	The NULL operator is used to compare a value with a NULL value.
UNIQUE	The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).

14.5 SQL expression

SQL EXPRESSIONS are like formulas and they are written in query language. You can also use them to query the database for specific set of data. An expression is a combination of one or more values, operators, and SQL functions that evaluate to a value.

Syntax:

Consider the basic syntax of the SELECT statement as follows:

```
SELECT column1, column2, columnN
FROM table_name
WHERE [CONDITION | EXPRESSION];
```

There are different types of SQL expressions, which are mentioned below:

14.5.1 SQL - Boolean Expressions:

SQL Boolean Expressions fetch the data on the basis of matching single value. Following is the syntax:

```
SELECT column1, column2, columnN
FROM table_name
WHERE SINGLE VALUE MATCHTING EXPRESSION;
```

Consider the EMPLOYEES table having the following records:

Here is simple example showing usage of SQL Boolean Expressions:

```
SQL> SELECT * FROM EMPLOYEES WHERE age =45;
+-----+-----+-----+-----+
| ID | NAME      | AGE | ADDRESS | SALARY |
+-----+-----+-----+-----+
| 3  | Srinivas   | 45  | Mangalore | 37000.00 |
+-----+-----+-----+-----+
1 row inset(0.00 sec)
```

14.5.2 Numeric Expression:

This expression is used to perform any mathematical operation in any query. Following is the syntax:

```
SELECT numerical_expression as OPERATION_NAME
[FROM table_name
WHERE CONDITION];
```

Here numerical_expression is used for mathematical expression or any formula. Following is a simple examples showing usage of SQL Numeric Expressions:

```
SQL> SELECT (15+6) AS ADDITION
+-----+
| ADDITION |
+-----+
| 21 |
+-----+
1 row inset(0.00 sec)
```

There are several built-in functions like avg(), sum(), count() etc., to perform what is known as aggregate data calculations against a table or a specific table column.

```
SQL> SELECT COUNT(*) AS "RECORDS" FROM EMPLOYEES;
+-----+
| RECORDS |
+-----+
| 7 |
+-----+
1 row inset(0.00 sec)
```

14.5.3 Date Expressions:

Date Expressions return current system date and time values:

```
SQL> SELECT CURRENT_TIMESTAMP;
+-----+
| Current_Timestamp |
+-----+
| 2014-03-21 06:40:23 |
+-----+
1 row inset(0.00 sec)
```

Another date expression is as follows:

```
SQL> SELECT GETDATE();  
+-----+  
| GETDATE |  
+-----+  
| 2014-01-14 12:07:18.140 |  
+-----+  
1 row inset(0.00 sec)
```

14.6 SQL Constraints:

Constraints are the rules enforced on data columns on table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Constraints could be column level or table level. Column level constraints are applied only to one column whereas table level constraints are applied to the whole table.

Following are commonly used constraints available in SQL:

The constraints available in SQL are Foreign Key, Not Null, Unique, Check.

Constraints can be defined in two ways

- 1) The constraints can be specified immediately after the column definition. This is called column-level definition.
- 2) The constraints can be specified after all the columns are defined. This is called table-level definition.

14.6.1 SQL Primary key:

This constraint defines a column or combination of columns which uniquely identifies each row in the table.

Syntax to define a Primary key at column level:

column name datatype [CONSTRAINT constraint_name] PRIMARY KEY

Syntax to define a Primary key at table level:

[CONSTRAINT constraint_name] PRIMARY KEY

(column_name1,column_name2,...)

column_name1, column_name2 are the names of the columns which define the primary Key.

The syntax within the bracket i.e. [CONSTRAINT constraint_name] is optional.

For Example: To create an employee table with Primary Key constraint, the query would be like.

Primary Key at column level:

```
CREATE TABLE employee  
( id number(5) PRIMARY KEY,  
name char(20),  
dept char(10),  
age number(2),  
salary number(10),  
city char(10)  
);
```

or

```
CREATE TABLE employee  
( id number(5) CONSTRAINT emp_id_pk PRIMARY KEY,  
name char(20),  
dept char(10),  
age number(2),  
salary number(10),  
city char(10)  
);
```

Primary Key at column level:

```
CREATE TABLE employee  
( id number(5),  
name char(20),  
dept char(10),  
age number(2),  
salary number(10),  
city char(10),  
CONSTRAINT emp_id_pk PRIMARY KEY (id)  
);
```

Primary Key at table level:

```
CREATE TABLE employee  
( id number(5), NOT NULL,  
name char(20),  
dept char(10),  
age number(2),  
salary number(10),  
city char(10),  
ALTER TABLE employee ADD CONSTRAINT PK_EMPLOYEE_ID PRIMARY KEY  
(id) );
```

14.6.2 Foreign key or Referential Integrity :

This constraint identifies any column referencing the PRIMARY KEY in another table. It establishes a relationship between two columns in the same table or between different tables. For a column to be defined as a Foreign Key, it should be defined as a Primary Key in the table which it is referring. One or more columns can be defined as Foreign key.

Syntax to define a Foreign key at column level:

[CONSTRAINT constraint_name] REFERENCES

Referenced_Table_name(column_name)

Syntax to define a Foreign key at table level:

[CONSTRAINT constraint_name] FOREIGN KEY(column_name) REFERENCES
referenced_table_name(column_name);

For Example:

1) Lets use the “sports” table and “order_items”.

Foreign Key at column level:

```
CREATE TABLE product
( product_id number(5) CONSTRAINT pd_id_pk PRIMARY KEY,
product_name char(20),
supplier_name char(20),
unit_price number(10)
);
```



```
SQL> select o.order_id,p.product_name,p.unit_price,p.supplier_name
  2  from sports p, order_items o
  3  where o.product_id=p.product_id;
```

ORDER_ID	PRODUCT_NAME	UNIT_PRICE	SUPPLIER_NAME
20141	BAT	15000	SUNNY

```
SQL>
SQL>
SQL>
SQL>
```

14.6.3 Not Null Constraint :

This constraint ensures all rows in the table contain a definite value for the column which is specified as not null. Which means a null value is not allowed.

Syntax to define a Not Null constraint:

[CONSTRAINT constraint name] NOT NULL

For Example: To create a employee table with Null value, the query would be like

```
CREATE TABLE employee
( id number(5),
name char(20) CONSTRAINT nm_nn NOT NULL,
dept char(10),
age number(2),
salary number(10),
CITY char(10)
);
```

14.6.4 Unique Key:

This constraint ensures that a column or a group of columns in each row have a distinct value. A column(s) can have a null value but the values cannot be duplicated.

Syntax to define a Unique key at column level:

[CONSTRAINT constraint_name] UNIQUE

Syntax to define a Unique key at table level:

[CONSTRAINT constraint_name] UNIQUE(column_name)

For Example: To create an employee table with Unique key, the query would be like,

14.6.5 Check Constraint :

This constraint defines a business rule on a column. All the rows must satisfy this rule. The constraint can be applied for a single column or a group of columns.

Syntax to define a Check constraint:

[CONSTRAINT constraint_name] CHECK (condition)

For Example: In the employee table to select the gender of a person, the query would be like

Check Constraint at column level:

14.7 Implementation of SQL commands:

In this chapter the SQL commands are explained along with the syntax and example, which are worked out in SQL 8.1i. The example is taken as employees table and all the syntax like create, alter, drop are illustrated with the example and DML commands like insert,select,where,order,group etc. are given.

14.7.1 CREATE TABLE statement is used to create a new table.

Basic syntax of CREATE TABLE statement is as follows:

```
CREATE TABLE table_name(
    column1 datatype,
    column2 datatype,
    column3 datatype,
    .....
    columnN datatype,
    PRIMARY KEY( one or more columns )
);
```

CREATE TABLE is the keyword telling the database system what you want to do. In this case, you want to create a new table. The unique name or identifier for the table follows the CREATE TABLE statement.

Then in brackets comes the list defining each column in the table and what sort of data type it is. The syntax becomes clearer with an example below.

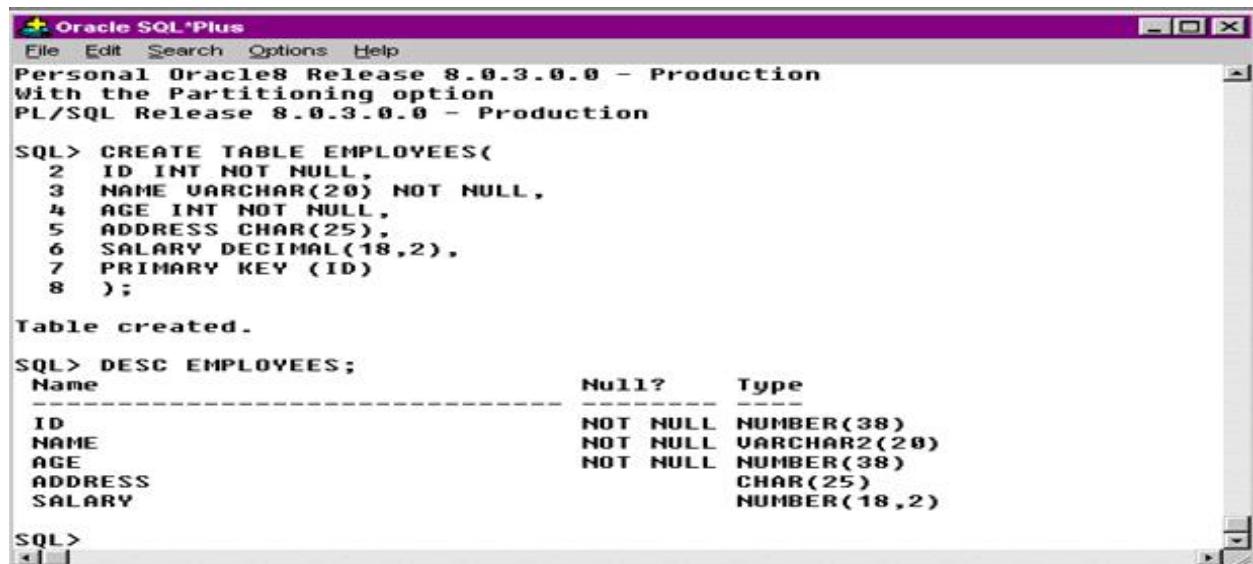
A copy of an existing table can be created using a combination of the CREATE TABLE statement and the SELECT statement.

Following is an example, which creates a EMPLOYEES table with ID as primary key and NOT NULL are the constraints showing that these fields cannot be NULL while creating records in this table:

You can check complete details at [Create Table Using another Table](#).

You can verify if your table has been created successfully by looking at the message displayed by the SQL server, otherwise you can use **DESC** command as follows:

Now, you have EMPLOYEES table available in your database which you can use to store required information related to EMPLOYEES.



The screenshot shows the Oracle SQL*Plus interface. The title bar reads "Oracle SQL*Plus" and "Personal Oracle8 Release 8.0.3.0.0 - Production". The menu bar includes "File", "Edit", "Search", "Options", and "Help". The main window displays the following SQL session:

```

Personal Oracle8 Release 8.0.3.0.0 - Production
With the Partitioning option
PL/SQL Release 8.0.3.0.0 - Production

SQL> CREATE TABLE EMPLOYEES(
  2  ID INT NOT NULL,
  3  NAME VARCHAR(20) NOT NULL,
  4  AGE INT NOT NULL,
  5  ADDRESS CHAR(25),
  6  SALARY DECIMAL(18,2),
  7  PRIMARY KEY (ID)
  8  );

Table created.

SQL> DESC EMPLOYEES;
  Name          Null?    Type
  ID            NOT NULL NUMBER(38)
  NAME          NOT NULL VARCHAR2(20)
  AGE           NOT NULL NUMBER(38)
  ADDRESS        CHAR(25)
  SALARY         NUMBER(18,2)

```

14.7.2 Alter statement The table can be modified or changed by using the Alter Command. The command is **ALTER table Table Tablename(columnname datatype(size);**

```
SQL> alter table employee modify salary number(15,2);
```

Table altered.

* **Drop statement:**

The SQL **DROP TABLE** statement is used to remove a table definition and all data, indexes, triggers, constraints, and permission specifications for that table.

NOTE: You have to be careful while using this command because once a table is deleted then the table along with information available in the table would also be lost forever.

Syntax: Basic syntax of DROP TABLE statement is as follows:

```
DROP TABLE table_name;
```

Example: Let us first verify EMPLOYEES table and then we would delete it from the database:

This means EMPLOYEES table is available in the database, so let us drop it as follows:



```
SQL> drop table employees;
Table dropped.

SQL>
SQL>
SQL>
SQL>
```

14.7.3 Insert: The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax:

There are two basic syntaxes of INSERT INTO statement as follows:

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)
VALUES (value1, value2, value3,...valueN);
```

Here, column1, column2,...columnN are the names of the columns in the table into which you want to insert data.

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table. The SQL INSERT INTO syntax would be as follows:

```
SQL> INSERT INTO EMPLOYEES(ID,NAME,AGE,ADDRESS,SALARY) VALUES(1,'RAJAPPA',42,'TUMKUR',40000.00);
1 row created.

SQL> INSERT INTO EMPLOYEES(ID,NAME,AGE,ADDRESS,SALARY) VALUES(2,'NAVEEN',39,'BIDAR',35000.00);
1 row created.

SQL> INSERT INTO EMPLOYEES(ID,NAME,AGE,ADDRESS,SALARY) VALUES(3,'SRINIVAS',45,'MANGALORE',45000.00
1 row created.

SQL> INSERT INTO EMPLOYEES(ID,NAME,AGE,ADDRESS,SALARY) VALUES(4,'NAGAMANI',52,'MYSORE',38000.00);
1 row created.

SQL> INSERT INTO EMPLOYEES(ID,NAME,AGE,ADDRESS,SALARY) VALUES(5,'SHARON',41,'GULBARGA',42000.00);
1 row created.
```

Populate one table using another table:

You can populate data into a table through select statement over another table provided another table has a set of fields, which are required to populate first table. Here is the syntax.

```
INSERT INTO first_table_name [(column1, column2,... columnN)]
  SELECT column1, column2,...columnN
    FROM second_table_name
   [WHERE condition];
```

14.7.4 SELECT : Select statement is used to fetch the data from a database table which returns data in the form of result table. These result tables are called result-sets.

Syntax:

The basic syntax of SELECT statement is as follows:

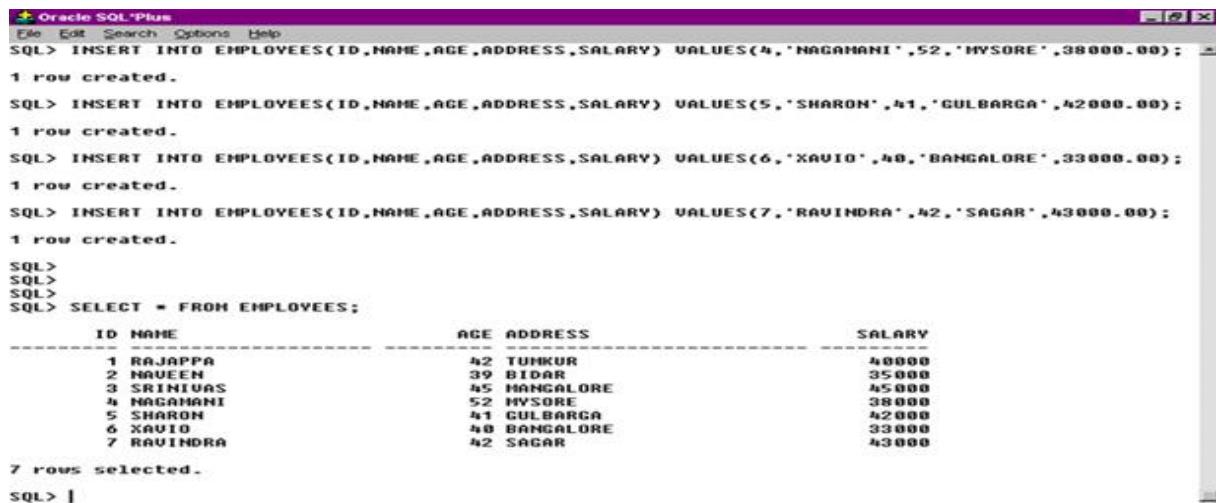
```
SELECT column1, column2, columnN FROM table_name;
```

Here, column1, column2, ... are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax:

```
SELECT * FROM table_name;
```

Example:

Consider the EMPLOYEES table having the following records:



The screenshot shows the Oracle SQL*Plus interface. The user has run several SQL commands to insert data into the EMPLOYEES table and then selected all columns for all rows.

```

SQL> INSERT INTO EMPLOYEES(ID,NAME,AGE,ADDRESS,SALARY) VALUES(4,'NAGAMANI',52,'MYSORE',38000.00);
1 row created.

SQL> INSERT INTO EMPLOYEES(ID,NAME,AGE,ADDRESS,SALARY) VALUES(5,'SHARON',41,'GULBARGA',42000.00);
1 row created.

SQL> INSERT INTO EMPLOYEES(ID,NAME,AGE,ADDRESS,SALARY) VALUES(6,'XAVIO',40,'BANGALORE',33000.00);
1 row created.

SQL> INSERT INTO EMPLOYEES(ID,NAME,AGE,ADDRESS,SALARY) VALUES(7,'RAVINDRA',42,'SAGAR',43000.00);
1 row created.

SQL>
SQL>
SQL> SELECT * FROM EMPLOYEES;
      ID NAME          AGE ADDRESS        SALARY
-----  --  ---  -----
       1 RAJAPPA        42  TUMKUR        40000
       2 NAVEEN         39  BIDAR         35000
       3 SRIWIAS        45  MANGALORE     45000
       4 NAGAMANI       52  MYSORE        38000
       5 SHARON         41  GULBARGA     42000
       6 XAVIO          40  BANGALORE     33000
       7 RAVINDRA       42  SAGAR         43000

7 rows selected.

SQL> |
```

The SQL **WHERE** clause is used to specify a condition while fetching the data

from single table or joining with multiple tables.

If the given condition is satisfied then only it returns specific value from the table. You would use WHERE clause to filter the records and fetching only necessary records.

The WHERE clause is not only used in SELECT statement, but it is also used in UPDATE, DELETE statement etc., which we would examine in subsequent chapters.

Syntax:

The basic syntax of SELECT statement with WHERE clause is as follows:

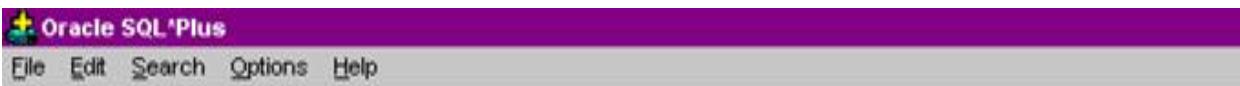
```
SELECT column1, column2, columnN  
FROM table_name  
WHERE [condition]
```

You can specify a condition using comparison or logical operators like $>$, $<$, $=$, LIKE, NOT, etc. Below examples would make this concept clear.

Example:

Consider the EMPLOYEES table having the following records:

Following is an example which would fetch ID, Name and Salary fields from the EMPLOYEES table where salary is greater than 35000:



```
Oracle SQL*Plus  
File Edit Search Options Help
```

SQL> SELECT ID,NAME,SALARY FROM EMPLOYEES WHERE SALARY >35000 OR AGE<45;

ID	NAME	SALARY
1	RAJAPPA	40000
2	NAVEEN	35000
3	SRINIVAS	45000
4	NAGAMANI	38000
5	SHARON	42000
6	XAVIO	33000
7	RAVINDRA	43000

7 rows selected.

Following is an example, which would fetch ID, Name and Salary fields for all employees whose name is Naveen. Here, it is important to note that all the strings should be given inside single quotes ('') whereas numeric values should be given without any quote as in above example:

```
SQL> SELECT ID, NAME, SALARY  
FROM EMPLOYEES  
WHERE NAME ='Naveen';
```

This would produce the following result:

ID	NAME	SALARY
2	Naveen	35000.00

The SQL **AND** and **OR** operators are used to combine multiple conditions to narrow data in an SQL statement. These two operators are called conjunctive operators.

These operators provide a means to make multiple comparisons with different operators in the same SQL statement.

14.7.5 AND Operator:

The **AND** operator allows the existence of multiple conditions in an SQL statement's WHERE clause.

Syntax:

The basic syntax of AND operator with WHERE clause is as follows:

```
SELECT column1, column2, columnN  
FROM table_name  
WHERE [condition1] AND [condition2]...AND [conditionN];
```

You can combine N number of conditions using AND operator. For an action to be taken by the SQL statement, whether it be a transaction or query, all conditions separated by the AND must be TRUE.

Example: Consider the EMPLOYEES table having the following records:

Following is an example, which would fetch ID, Name and Salary fields from the EMPLOYEES table where salary is greater than 2000 AND age is less than 25 years:

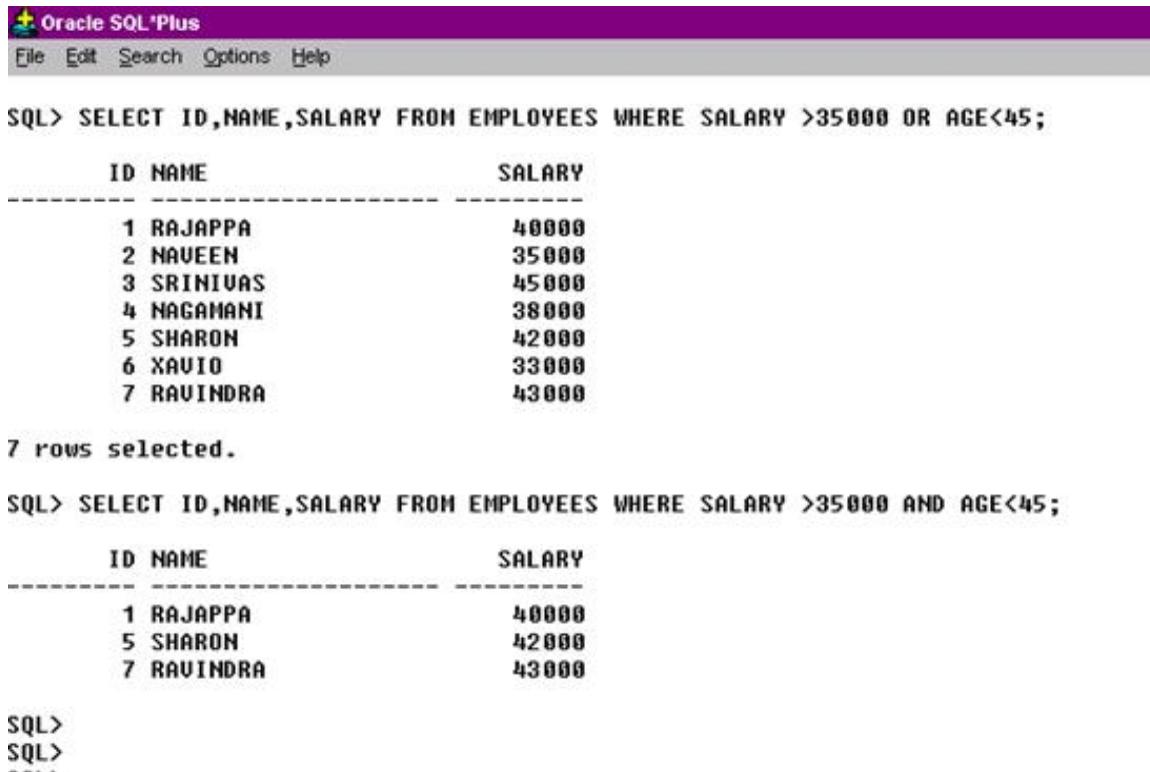
14.7.6 OR Operator:

The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.

Syntax: The basic syntax of OR operator with WHERE clause is as follows:

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition1] OR [condition2]...OR [conditionN]
```

You can combine N number of conditions using OR operator. For an action to be taken by the SQL statement, whether it be a transaction or query, only any ONE of the conditions separated by the OR must be TRUE.



```
SQL> SELECT ID,NAME,SALARY FROM EMPLOYEES WHERE SALARY >35000 OR AGE<45;
      ID NAME          SALARY
----- -----
      1 RAJAPPA        40000
      2 NAVEEN         35000
      3 SRINIUS        45000
      4 NAGAMANI       38000
      5 SHARON         42000
      6 XAVIO          33000
      7 RAVINDRA       43000
7 rows selected.

SQL> SELECT ID,NAME,SALARY FROM EMPLOYEES WHERE SALARY >35000 AND AGE<45;
      ID NAME          SALARY
----- -----
      1 RAJAPPA        40000
      5 SHARON         42000
      7 RAVINDRA       43000
5 rows selected.
```

14.7.7 Update: The SQL **UPDATE** Query is used to modify the existing records

in a table.

You can use WHERE clause with UPDATE query to update selected rows otherwise all the rows would be affected.

Syntax: The basic syntax of UPDATE query with WHERE clause is as follows:

```
UPDATE table_name
SET column1 = value1, column2 = value2...., columnN = valueN
WHERE [condition];
```

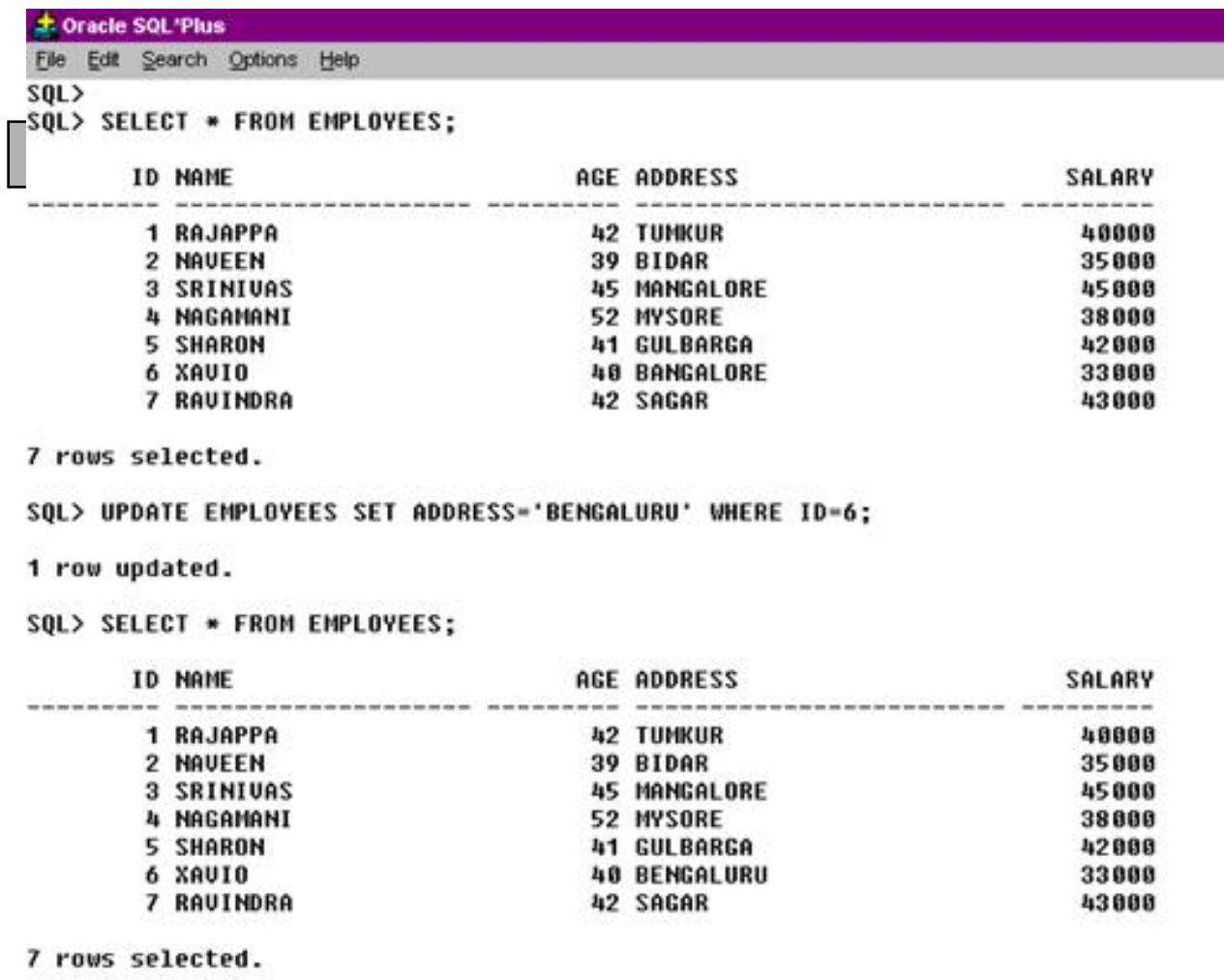
You can combine N number of conditions using AND or OR operators.

Example:

Consider the EMPLOYEES table having the following records:

Following is an example, which would update ADDRESS for a customer whose ID is 6:

```
SQL> UPDATE EMPLOYEES SET ADDRESS ='Bengaluru' WHERE ID =6;
```



The screenshot shows the Oracle SQL*Plus interface. The menu bar includes File, Edit, Search, Options, and Help. The SQL prompt is at the top. The command `SELECT * FROM EMPLOYEES;` is run, displaying a table of employee data. The data is as follows:

ID	NAME	AGE	ADDRESS	SALARY
1	RAJAPPA	42	TUMKUR	40000
2	NAVEEN	39	BIDAR	35000
3	SRINIVAS	45	MANGALORE	45000
4	NAGAMANI	52	mysore	38000
5	SHARON	41	GULBARGA	42000
6	XAVIO	40	BANGALORE	33000
7	RAVINDRA	42	SAGAR	43000

7 rows selected.

The command `UPDATE EMPLOYEES SET ADDRESS='BENGALURU' WHERE ID=6;` is run, updating the address for employee ID 6. The message "1 row updated." is displayed.

The final command `SELECT * FROM EMPLOYEES;` is run again, showing the updated data:

ID	NAME	AGE	ADDRESS	SALARY
1	RAJAPPA	42	TUMKUR	40000
2	NAVEEN	39	BIDAR	35000
3	SRINIVAS	45	MANGALORE	45000
4	NAGAMANI	52	mysore	38000
5	SHARON	41	GULBARGA	42000
6	XAVIO	40	BENGALURU	33000
7	RAVINDRA	42	SAGAR	43000

7 rows selected.

14.7.8 **DELETE** Query is used to delete the existing records from a table.

You can use WHERE clause with DELETE query to delete selected rows, otherwise all the records would be deleted.

Syntax: The basic syntax of DELETE query with WHERE clause is as follows:

```
DELETE FROM table_name
WHERE [condition];
```

You can combine N number of conditions using AND or OR operators.

Following is an example, which would DELETE a customer, whose ID is 3:

```
Oracle SQL*Plus
File Edit Search Options Help
```

```
SQL> select id,name,age,address,salary from employees;
```

ID	NAME	AGE	ADDRESS	SALARY
1	RAJAPPA	42	TUMKUR	40000
2	NAVEEN	39	BIDAR	35000
9	Padmashree	51	BANGALORE	39000
4	NAGAHANI	52	MYSORE	38000
5	SHARON	41	GULBARGA	42000
6	XAVIO	40	BANGALORE	33000
7	RAVINDRA	42	SAGR	43000
3	Srinivas	48	Mangalore	48000
8	HEMALATHA	35	BANGALORE	60000

```
9 rows selected.
```

```
SQL> delete from employees where id=3;
```

```
1 row deleted.
```

```
SQL> SELECT * FROM EMPLOYEES
WHERE ROWNUM <= 3;
```

This would produce the following result:

+-----+	+-----+	+-----+	+-----+
ID NAME	AGE	ADDRESS	SALARY
+-----+	+-----+	+-----+	+-----+
1 Rajappa	42 Tumkur		40000.00
2 Naveen	39 Bidar	35000.00	
3 Srinivas	45 Mangalore		32000.00

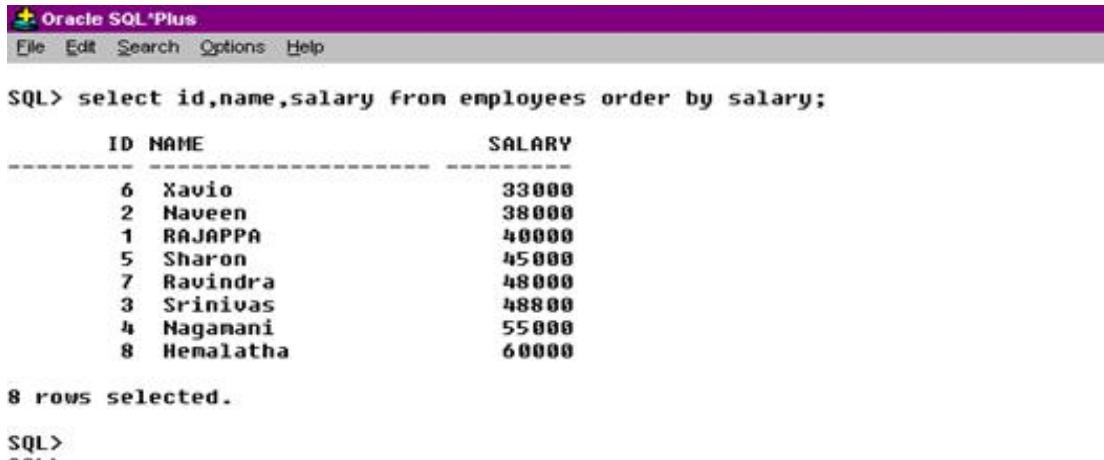
14.7.9 ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns. Some database sorts query results in ascending order by default.

Syntax: The basic syntax of ORDER BY clause is as follows:

```
SELECT column-list
FROM table_name
[WHERE condition]
[ORDER BY column1, column2,.. columnN][ASC | DESC];
```

You can use more than one column in the ORDER BY clause. Make sure whatever column you are using to sort, that column should be in column-list.

Example: Consider the EMPLOYEES table having the following records:



The screenshot shows the Oracle SQL*Plus interface. The menu bar includes File, Edit, Search, Options, Help, and a logo. The command line displays the SQL command: `SQL> select id,name,salary from employees order by salary;`. The output shows a table with three columns: ID, NAME, and SALARY. The data is sorted by salary in ascending order. The output ends with the message `8 rows selected.`

ID	NAME	SALARY
6	Xavio	33000
2	Naveen	38000
1	RAJAPPA	40000
5	Sharon	45000
7	Ravindra	48000
3	Srinivas	48800
4	Nagamani	55000
8	Hemalatha	60000

8 rows selected.

SQL>

14.7.10 GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups.

The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

Syntax: The basic syntax of GROUP BY clause is given below. The GROUP BY clause must follow the conditions in the WHERE clause and must precede the ORDER BY clause if one is used.

```
SELECT column1, column2
FROM table_name
WHERE [ conditions ]
GROUP BY column1, column2
ORDER BY column1, column2
```

If you want to know the total amount of salary on each customer, then GROUP BY query would be as follows:

Group functions are built-in SQL functions that operate on groups of rows and return one value for the entire group. These functions are: **COUNT, MAX, MIN, AVG, SUM, DISTINCT**

SQL COUNT (): This function returns the number of rows in the table that satisfies the condition specified in the WHERE condition. If the WHERE condition is not specified, then the query returns the total number of rows in the table.

For Example: If you want the number of employees in a particular department, the query would be:

```
SELECT COUNT (*) FROM employee
```

```
WHERE dept = 'Computer Science';
```

If you want the total number of employees in all the department, the query would take the form:

```
SELECT COUNT (*) FROM employee;
```

SQL DISTINCT(): This function is used to select the distinct rows.

For Example: If you want to select all distinct department names from employee table, the query would be:

```
Select Distinct dept FROM employee;
```

To get the count of employees with unique name, the query would be:

```
SELECT COUNT (DISTINCT name) FROM employee;
```

SQL MAX(): This function is used to get the maximum value from a column.

To get the maximum salary drawn by an employee, the query would be:

```
SELECT MAX (salary) FROM employee;
```

SQL MIN(): This function is used to get the minimum value from a column.

To get the minimum salary drawn by an employee, the query would be:

```
SELECT MIN (salary) FROM employee;
```

SQL AVG(): This function is used to get the average value of a numeric column.

To get the average salary, the query would be

```
SELECT AVG (salary) FROM employee;
```

SQL SUM(): This function is used to get the sum of a numeric column

To get the total salary given out to the employees,

Example: Consider the EMPLOYEES table is having the following records:

```

Oracle SQL*Plus
File Edit Search Options Help
3 Srinivas          45 Mangalore      48800
4 Nagamani         52 Mysore        55000
5 Sharon           41 Gulbarga       45000
6 Xavio            40 Bangalore     33000
7 Ravindra          42 Sagar          48000
8 Hemalatha        32 Bangalore     60000

8 rows selected.

SQL> select count(*) from employees;

COUNT(*)
-----
8

SQL> select max(salary) from employees;

MAX(SALARY)
-----
60000

SQL> select min(salary) from employees;

MIN(SALARY)
-----
33000

SQL> select avg(salary) from employees;

AVG(SALARY)
-----
45975

SQL>

```

SELECT SUM (salary) FROM employee;

Now again, if you want to know the total amount of salary on each customer, then GROUP BY query would be as follows:

```

SQL> SELECT NAME, SUM(SALARY) FROM EMPLOYEES
      GROUP BY NAME;

```

DISTINCT keyword is used in conjunction with SELECT statement to eliminate all the duplicate records and fetching only unique records.

There may be a situation when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only unique records instead of fetching duplicate records.

Syntax: The basic syntax of DISTINCT keyword to eliminate duplicate records is as follows:

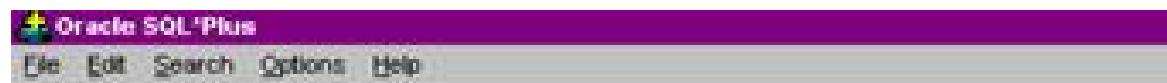
First, let us see how the following SELECT query returns duplicate salary records:

This would produce the following result where salary 45000 is coming twice which is a duplicate record from the original table.

```
SELECT DISTINCT column1, column2, .....columnN
FROM table_name
WHERE [condition]
```

ID	NAME	AGE	ADDRESS	SALARY
1	Rajappa	42	Tumkur	40000.00
2	Naveen	39	Bidar	35000.00
3	Srinivas	45	Mangalore	32000.00
4	Nagamani	52	Myosre	38000.00
5	Sheron	41	Gulbarga	42000.00
6	Xavio	40	Bangalore	33000.00
7	Ravindra	44	Sagar	43000.00

```
SQL> SELECT SALARY FROM EMPLOYEES
      ORDER BY SALARY;
```



```
SQL> select id,name,salary from employees order by salary;

  ID NAME          SALARY
-----  -----
    6 Xavio        33000
    2 Naveen       38000
    1 RAJAPPA     40000
    5 Sharon       45000
    7 Ravindra     48000
    3 Srinivas     48000
    4 Nagamani     55000
    8 Hemalatha   60000

8 rows selected.

SQL>
```

14.7.12 Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

Consider the following two tables, (a) sports table is as follows:

(b) Another table is ORDER_items as follows:

Now, let us join these two tables in our SELECT statement as follows:

This would produce the following result:

Here, it is noticeable that the join is performed in the WHERE clause. Several operators can be used to join tables, such as =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, and NOT; they can all be used to join tables. However, the most common operator is the equal symbol.

SQL Join Types:

There are different types of joins available in SQL:

- **INNER JOIN:** returns rows when there is a match in both tables.
- **LEFT JOIN:** returns all rows from the left table, even if there are no matches in the right table.
- **RIGHT JOIN:** returns all rows from the right table, even if there are no matches in the left table.
- **FULL JOIN:** returns rows when there is a match in one of the tables.
- **SELF JOIN:** is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- **CARTESIAN JOIN:** returns the Cartesian product of the sets of records from the two or more joined tables.

In order to experiment the join commands, we are creating two tables one called the sports and the order_items for the sports items. While preparing the join operations, we can use alias so that it becomes easy. This is illustrated in the given example. p is the sports table and o is the order items table.

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> create table sports(
  2 product_id number(5) constraint Pd_id_pk PRIMARY KEY,
  3 product_name char(20),
  4 supplier_name varchar(20),
  5 unit_price number(10)
  6 );
```

Table created.

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> create table order_items(
  2 order_id number(5),
  3 product_id number(5),
  4 product_name char(20),
  5 supplier_name varchar(20),
  6 unit_price number(10),
  7 constraint od_id_pk primary key(order_id),
  8 constraint pd_id_fk Foreign key (product_id) references sports(product_id)
  9 );
```

Table created.

SQL>
SQL>

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> insert into order_items values(20141,101,'BAT', 'GPUC',3);
```

1 row created.

SQL>
SQL>
SQL>
SQL>

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> select o.order_id,p.product_name,p.unit_price,p.supplier_name
  2 from sports p, order_items o
  3 where o.product_id=p.product_id;
```

ORDER_ID	PRODUCT_NAME	UNIT_PRICE	SUPPLIER_NAME
20141	BAT	15000	SUNNY

SQL>
SQL>
SQL>

This would produce the following result:

There are two other clauses (i.e., operators), which are very similar to UNION clause:

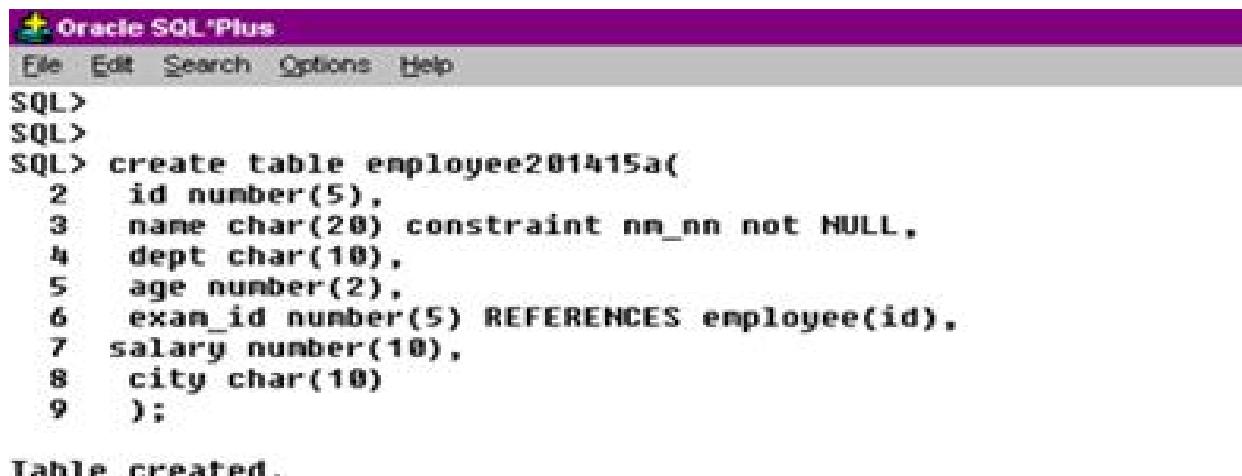
- SQL INTERSECT Clause: is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement.
- SQL EXCEPT Clause : combines two SELECT statements and returns rows from the first SELECT statement that are not returned by the second SELECT statement.

14.7.13 NULL: The SQL **NULL** is the term used to represent a missing value. A NULL value in a table is a value in a field that appears to be blank.

A field with a NULL value is a field with no value. It is very important to understand that a NULL value is different than a zero value or a field that contains spaces.

Syntax:

The basic syntax of **NULL** while creating a table:



```
+ Oracle SQL*Plus
File Edit Search Options Help
SQL>
SQL>
SQL> create table employee201415a(
  2   id number(5),
  3   name char(20) constraint nn_nn not NULL,
  4   dept char(10),
  5   age number(2),
  6   exam_id number(5) REFERENCES employee(id),
  7   salary number(10),
  8   city char(10)
  9 );
```

Table created.

Here, **NOT NULL** signifies that column should always accept an explicit value of the given data type. There are two columns where we did not use NOT NULL, which means these columns could be NULL.

A field with a NULL value is one that has been left blank during record creation.

Example:

The NULL value can cause problems when selecting data, however, because when comparing an unknown value to any other value, the result is always unknown and not included in the final results.

You must use the **IS NULL** or **IS NOT NULL** operators in order to check for a NULL value.

Now, following is the usage of **IS NOT NULL** operator:

```
SQL> SELECT ID, NAME, AGE, ADDRESS, SALARY
      FROM EMPLOYEES
     WHERE SALARY IS NOT NULL;
```

This would produce the following result:

Now, following is the usage of **IS NULL** operator:

```
SQL> SELECT ID, NAME, AGE, ADDRESS, SALARY
      FROM EMPLOYEES
     WHERE SALARY IS NULL;
```

This would produce the following result:

You can rename a table or a column temporarily by giving another name known as alias.

```
SQL>
SQL> rename employee to employee2014;
```

Table renamed.

```
SQL> desc employee2014;
   Name          Null?    Type
   ----          -----   -----
   ID           NUMBER(5)
   NAME         CHAR(20)
   DEPT        CHAR(10)
   AGE          NUMBER(2)
   SALARY       NUMBER(15,2)
   CITY         CHAR(10)
```

```
SQL>
SQL>
```

The use of table aliases means to rename a table in a particular SQL statement.

14.8 Creating Views:

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables, or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic CREATE VIEW syntax is as follows:

```
CREATE VIEW view_name AS  
SELECT column1, column2.....  
FROM table_name  
WHERE [condition];
```

You can include multiple tables in your SELECT statement in very similar way as you use them in normal SQL SELECT query.

Example:

Consider the EMPLOYEES table having the following records:

```
SQL > CREATE VIEW EMPLOYEES_VIEW AS  
SELECT name, age  
FROM EMPLOYEES;
```

```
SQL > SELECT * FROM EMPLOYEES_VIEW;
```

14.9 The COMMIT Command:

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database.

The COMMIT command saves all transactions to the database since the last COMMIT or ROLLBACK command.

The syntax for COMMIT command is as follows:

```
COMMIT;
```

14.10 DCL commands are used to enforce database security in a multiple user database environment. Two types of DCL commands are GRANT and REVOKE. Only Database Administrator's or owners of the database object can provide/remove privileges on a database object.

14.10.1 GRANT Command

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

The Syntax for the GRANT command is:

```
GRANT privilege_name  
ON object_name  
TO {user_name | PUBLIC | role_name}  
[WITH GRANT OPTION];
```

Ñ **privilege_name** is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.

Ñ **object_name** is the name of an database object like TABLE, VIEW, stored proc and SEQUENCE.

Ñ **user_name** is the name of the user to whom an access right is being granted.

Ñ **user_name** is the name of the user to whom an access right is being granted.

Ñ **PUBLIC** is used to grant access rights to all users.

Ñ **ROLES** are a set of privileges grouped together.

Ñ **WITH GRANT OPTION** - allows a user to grant access rights to other users.

For Example: GRANT SELECT ON employee TO user1;This command grants a SELECT permission on employee table to user1.You should use the WITH GRANT option carefully because for example if you GRANT SELECT privilege on employee table to user1 using the WITH GRANT option, then user1 can GRANT SELECT privilege on employee table to another user, such as user2 etc. Later, if you REVOKE the SELECT privilege on employee from user1, still user2 will have SELECT privilege on employee table.

14.10.2 REVOKE Command:

The REVOKE command removes user access rights or privileges to the database objects.

The Syntax for the REVOKE command is:

```
REVOKE privilege_name  
ON object_name  
FROM {user_name | PUBLIC | role_name};
```

For Example: REVOKE SELECT ON employee FROM user1;This command will REVOKE a SELECT privilege on employee table from user1.When you REVOKE SELECT privilege on a table from a user, the user will not be able to SELECT data from that table anymore. However, if the user has received SELECT privileges on that table from more than one users, he/she can SELECT from that table until everyone who granted the permission revokes it. You cannot REVOKE privileges if they were not initially granted by you.

Privileges and Roles:

Privileges: Privileges defines the access rights provided to a user on a database object. There are two types of privileges.

1) System privileges - This allows the user to CREATE, ALTER, or DROP Database objects.

2) Object privileges - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or Delete data from database objects to which the privileges apply.

Few CREATE system privileges are listed below:

System Privileges	Description
CREATE object	allows users to create the specified object in their own schema.
CREATE ANY object	allows users to create the specified object in any schema.

The above rules also apply for ALTER and DROP system privileges.

Few of the object privileges are listed below:

Object Privileges	Description
INSERT	allows users to insert rows into a table.
SELECT	allows users to select data from a database object.
UPDATE	allows user to update data in a table.
EXECUTE	allows user to execute a stored procedure or a function.

System Role	Privileges Granted to the Role
CONNECT	CREATE TABLE, CREATE VIEW, CREATE SYNONYM, CREATE SEQUENCE, CREATE SESSION etc.
RESOURCE	CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER etc. The primary usage of the RESOURCE role is to restrict access to database objects.
DBA	ALL SYSTEM PRIVILEGES

14.11 SQL built-in functions

There are two types of functions in Oracle sql version.

14.11.1 Single Row Functions: Single row or Scalar functions return a value for every row that is processed in a query.

14.11.2 Group Functions: These functions group the rows of data based on the values returned by the query. This is discussed in SQL GROUP Functions. The group functions are used to calculate aggregate values like total or average, which return just one total or one average value after processing a group of rows.

There are four types of single row functions. They are:

- 1) **Numeric Functions:** These are functions that accept numeric input and return numeric values.
- 2) **Character or Text Functions:** These are functions that accept character input and can return both character and number values.
- 3) **Date Functions:** These are functions that take values that are of datatype DATE as input and return values of datatype DATE, except for the MONTHS_BETWEEN function, which returns a number.
- 4) **Conversion Functions:** These are functions that help us to convert a value in one form to another form. For Example: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE etc.

You can combine more than one function together in an expression. This is known as nesting of functions.

DUAL Table in Oracle

This is a single row and single column dummy table provided by oracle. This is used to perform mathematical calculations without using a table.

Select * from DUAL

Output:

DUMMY

X

Select 777 * 888 from Dual

Output:

777 * 888

689976

Function Name	Return Value	
ABS (x)	Absolute value of the number 'x'	
CEIL (x)	Integer value that is Greater than or equal to the number 'x'	
FLOOR (x)	Integer value that is Less than or equal to the number 'x'	
TRUNC (x, y)	Truncates value of number 'x' up to 'y' decimal places	
ROUND (x, y)	Rounded off value of the number 'x' up to the number 'y' decimal places	
Function Name	Examples	Return Value
ABS (x)	ABS (1) ABS (-1)	1 -1
CEIL (x)	CEIL (2.83) CEIL (2.49) CEIL (-1.6)	3 3 -1
FLOOR (x)	FLOOR (2.83) FLOOR (2.49) FLOOR (-1.6)	2 2 -2
TRUNC (x, y)	ROUND (125.456, 1) ROUND (125.456, 0) ROUND (124.456, -1)	125.4 125 120
ROUND (x, y)	TRUNC (140.234, 2) TRUNC (-54, 1) TRUNC (5.7) TRUNC (142, -1)	140.23 54 5 140

These functions can be used on database columns.

For Example: Let's consider the product table used in sql joins. We can use ROUND to round off the unit_price to the nearest integer, if any product has prices in fraction.

```
SELECT ROUND (unit_price) FROM product;
```

2) Character or Text Functions:

Character or text functions are used to manipulate text strings. They accept strings or characters as input and can return both character and number values as output.

Few of the character or text functions are as given below:

Function Name	Return Value
LOWER (string_value)	All the letters in 'string_value' is converted to lowercase.
UPPER (string_value)	All the letters in 'string_value' is converted to uppercase.
INITCAP (string_value)	All the letters in 'string_value' is converted to mixed case.
LTRIM (string_value, trim_text)	All occurrences of 'trim_text' is removed from the left of 'string_value'.
RTRIM (string_value, trim_text)	All occurrences of 'trim_text' is removed from the right of 'string_value'.
TRIM (trim_text FROM string_value)	All occurrences of 'trim_text' from the left and right of 'string_value' , 'trim_text' can also be only one character long .
SUBSTR (string_value, m, n)	Returns 'n' number of characters from 'string_value' starting from the 'm' position.
LENGTH (string_value)	Number of characters in 'string_value' is returned.
LPAD (string_value, n, pad_value)	Returns 'string_value' left-padded with 'pad_value'. The length of the whole string will be of 'n' characters.
RPAD (string_value, n, pad_value)	Returns 'string_value' right-padded with 'pad_value'. The length of the whole string will be of 'n' characters.

For Example, we can use the above UPPER() text function with the column value as follows.

```
SELECT UPPER (product_name) FROM product;
```

The following examples explains the usage of the above character or text functions

Function Name	Examples	Return Value
LOWER(string_value)	LOWER('Good Morning')	good morning
UPPER(string_value)	UPPER('Good Morning')	GOOD MORNING
INITCAP(string_value)	INITCAP('GOOD MORNING')	Good Morning
LTRIM(string_value, trim_text)	LTRIM ('Good Morning', 'Good')	Morning
RTRIM (string_value, trim_text)	RTRIM ('Good Morning', 'Morning')	Good
TRIM (trim_text FROM string_value)	TRIM ('o' FROM 'Good Morning')	Gd Mrning
SUBSTR (string_value, m, n)	SUBSTR ('Good Morning', 6, 7)	Morning
LENGTH (string_value)	LENGTH ('Good Morning')	12
LPAD (string_value, n, pad_value)	LPAD ('Good', 6, '*')	**Good
RPAD (string_value, n, pad_value)	RPAD ('Good', 6, '*')	Good**

3) Date Functions:

These are functions that take values that are of datatype DATE as input and return values of datatypes DATE, except for the MONTHS_BETWEEN function, which returns a number as output.

Few date functions are as given below.

Function Name	Return Value
ADD_MONTHS (date, n)	Returns a date value after adding 'n'months to the date 'x'.
MONTHS_BETWEEN (x1, x2)	Returns the number of months between dates x1 and x2.
ROUND (x, date_format)	Returns the date 'x' rounded off to the nearest century, year, month, date, hour, minute, or second as specified by the 'date_format'.
TRUNC (x, date_format)	Returns the date 'x' lesser than or equal to the nearest century, year, month, date, hour, minute, or second as specified by the 'date_format'.
NEXT_DAY (x, week_day)	Returns the next date of the 'week_day'on or after the date 'x' occurs.
LAST_DAY (x)	It is used to determine the number of days remaining in a month from the date 'x' specified.
SYSDATE	Returns the systems current date and time.
NEW_TIME (x, zone1, zone2)	Returns the date and time in zone2 if date 'x' represents the time in zone1.

Function Name	Examples	Return Value
ADD_MONTHS ()	ADD_MONTHS ('14-Feb-14', 9)	14-Nov-14
MONTHS_BETWEEN()	MONTHS_BETWEEN ('16-Sep-14', '16-Dec-14')	3
NEXT_DAY()	NEXT_DAY ('20-Mar-2014', 'Thursday')	21-Mar-2014
LAST_DAY()	LAST_DAY ('01-Jun-14')	30-Jun-14
NEW_TIME()	NEW_TIME ('01-Jun-18', 'IST', 'EST')	31-May-14

4) Conversion Functions:

These are functions that help us to convert a value in one form to another form. For Ex: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE.

Few of the conversion functions available in oracle are:

Function Name	Return Value
---------------	--------------

Function Name	Return Value
TO_CHAR (x [,y])	Converts Numeric and Date values to a character string value. It cannot be used for calculations since it is a string value.
TO_DATE (x [, date_format])	Converts a valid Numeric and Character values to a Date value. Date is formatted to the format specified by 'date_format'.
NVL (x, y)	If 'x' is NULL, replace it with 'y'. 'x' and 'y' must be of the same datatype.
DECODE (a, b, c, d, e, default_value)	Checks the value of 'a', if $a = b$, then returns 'c'. If $a = d$, then returns 'e'. Else, returns <i>default_value</i> .

The below table provides the examples for the above functions

Function Name	Examples	Return Value
TO_CHAR ()	TO_CHAR (3000, '\$9999') TO_CHAR (SYSDATE, 'Day, Month YYYY')	\$3000 WEDNESDAY, MARCH 2014
TO_DATE ()	TO_DATE ('19-MAR-2014')	19-MAR-14
NVL ()	NVL (null, 1)	1

Summary

- >Sql -Structured Query Language(SQL)
- >SQL ARCHITECTURE:
- >SQL languages: DDL,DML,DCL,
- > Data access and retrieval
- >SQL built-in function.

Review questions

One mark questions

1. Expand SQL.
2. Give the syntax for create command in SQL.
3. What is drop command in SQL.
4. Give the command to display all the details in the table.
5. What is update command?
6. What is commit command?

Two marks questions

1. Classify Numeric and Character string data types in SQL.
2. Classify various SQL operators.
3. Which are the logical operators in SQL.
4. How do you modify the column name and width for existing table?
5. Write the syntax for distinct command in SQL.
6. What is the use of NULL value?
7. What is create view command?
8. What is dual table?

Three marks questions

1. Explain the features of SQL?
2. List the components of SQL architecture.
3. Explain DDL commands with example.
4. Explain DML commands with example.
5. Explain with an example Boolean expression in SQL.
6. Explain AND operator using where in SQL.
7. List the built-in functions associated with Group functions in SQL.
8. What is the use of join command?
9. What are privileges and rules?
10. Classify various built-in functions in SQL.

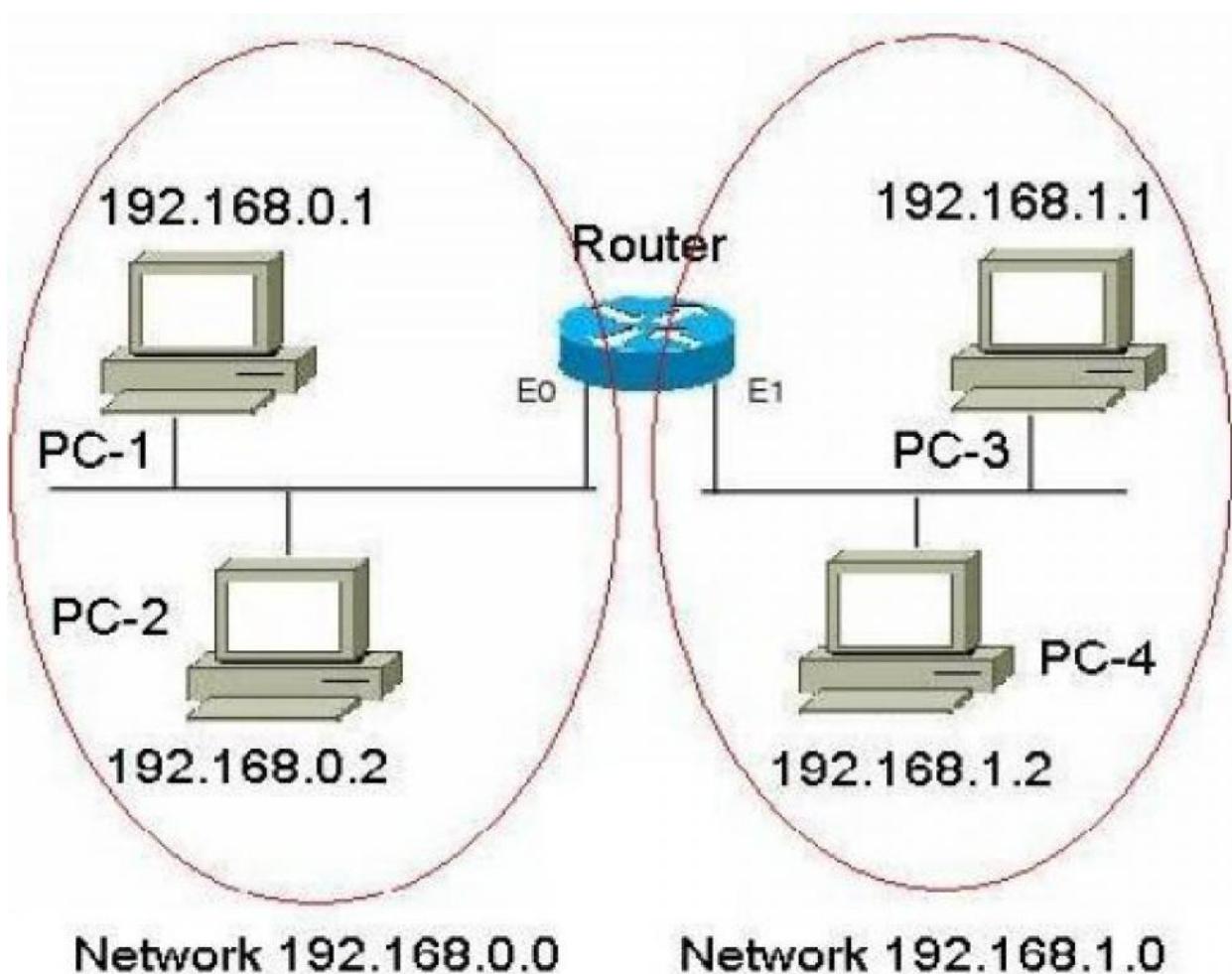
Five marks questions

1. Explain SQL constraints with example.
2. Explain with example to create details of employees and give the minimum and maximum in the salary domain.
3. Write the differences between order by and group by with example.

CHAPTER 15

Networking concepts**OBJECTIVES**

- To understand uses of networking.
- Various types of networking.
- To understand various devices used in networking.
- Applications used for networking
- How and methods of networking security.



15.1 Introduction

A Network is an inter-connection of autonomous computers. Two computers are said to be interconnected if they are capable of exchanging the information. Central to this definition is the fact that the computers are autonomous. This means that no computers on the network can start, stop or control another.

15.1.1 Network Goals:

The network goals are as listed below.

- (i) **Resource Sharing:** The aim is to make all the programs, data and peripherals available to anyone on the network irrespective of the physical location of the resources and the user.
- (ii) **Reliability:** A file can have copies on two or three different machines, so if one of them is unavailable, the other copies could be used. For military, banking and many other applications it is great of importance.
- (iii) **Cost Factor:** Personal computers have better price/performance ratio than micro computers. So it is better to have PC's, one per user with data stored on one shared file server machine.
- (iv) **Communication Medium:** Using a network, it is possible for managers, working far apart, to prepare financial report of the company. The changes at one end can be immediately noticed at another and hence it speeds up co-operation among them.

15.1.2 Need of Networking:

1. File sharing provides sharing and grouping of data files over the network.
2. Print sharing of computer resources such as hard disk and printers etc.
3. email tools for communication with the e-mail address.
4. Remote access able to access data and information, around the globe.
5. Sharing database to multiple users at the same time by ensuring the intergrity.

15.2.1 ARPANET

The Advanced Research Projects Agency Network (ARPANET) was one of the world's first operational packet switching networks, the first network to implement TCP/IP, and the progenitor of what was to become the global Internet. The network was initially funded by the Advanced Research Projects Agency (ARPA, later DARPA) within the U.S. Department of Defense for use by its projects at universities and research laboratories in the US. The packet switching of the

ARPANET, together with TCP/IP, would form the backbone of how the Internet works.

15.2.2 OSI Reference Model

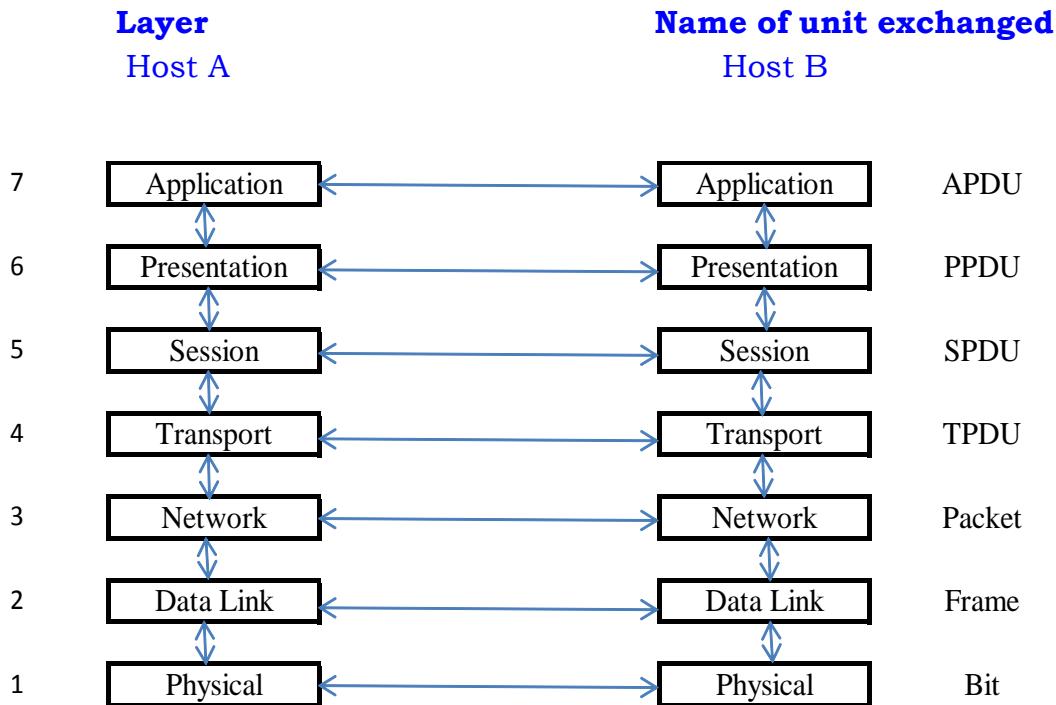


Figure 15.1 OSI Layers

The Physical Layer

The physical layer is concerned with transmitting raw bits over a communication channel. It also deals with mechanical, electrical and timing interfaces.

The Data Link Layer

The main function of the data link layer is to transform a raw transmission facility into a line that appears free of undetected transmission errors to the network layer.

The Network Layer

The network layer controls the operation of the subnet. The main function is to determine how packets are routed from source to destination.

The Transport Layer

The basic function of transport layer is to accept data from above layer and split it up into smaller units if needed, and pass these to the network layer and ensure that the pieces all arrive correctly at the other end. It also determines type of services to provide to the session layer.

The Session Layer

The session layer allows users on different machines to establish sessions between them. It includes dialog control, token management and synchronization.

The Presentation Layer

The presentation layer concerned with the syntax and semantics of the information transmitted concerned with moving bits around the layer.

The Application Layer

The application layer contains a variety of protocols that are commonly needed by the user. For example, HTTP (Hyper Text Transfer Protocol) which is the base for the World Wide Web (WWW) to access web pages.

15.2.3 TCP/IP (Transmission Control Protocol/Internet Protocol)

TCP/IP is a layered set of protocols. This protocol assumes that there is a way to communicate reliably between the two computers. Mail, like other application protocols, simply defines a set of commands and messages to be sent. TCP is responsible for making sure that the commands get through to the other end. It keeps track of what is sent, and retransmits anything that did not get through.

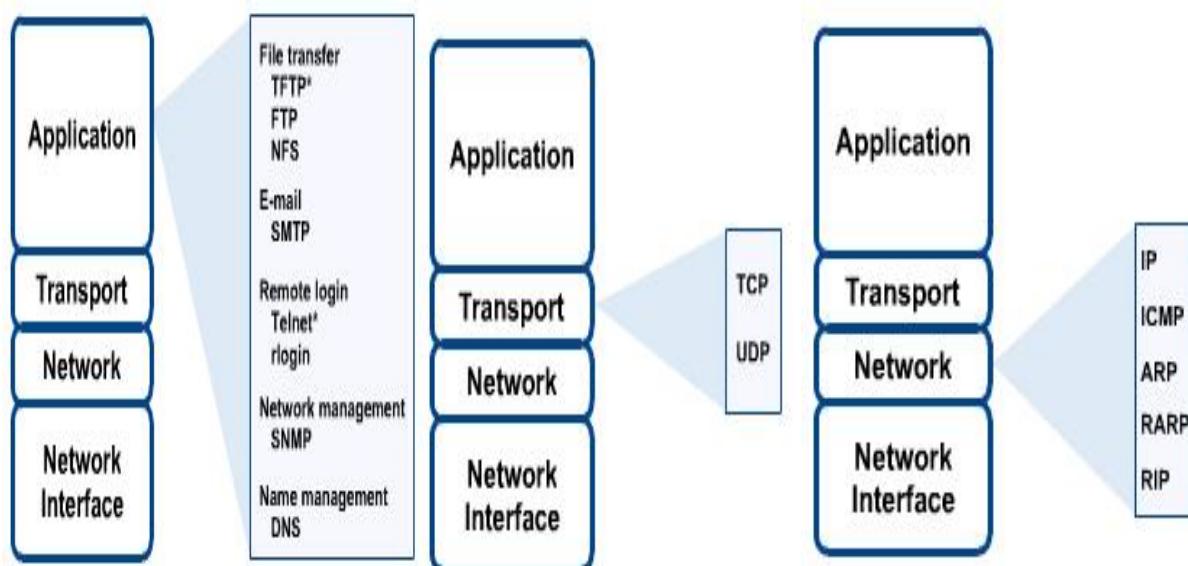


Figure 15.2 TCP/IP Layers

TCP/IP is the base communication protocol of the internet. The part of TCP/IP uses numeric IP addresses to join network segments and TCP part of TCP/IP provides reliable delivery messages between networked computers. It is based on the “catenet model”. This model assumes that there are a large number of independent networks connected together by gateways. The user should be able to access computers or other resources on any of these networks. Datagram will often pass through a dozen different networks before getting their final destination. The routing needed to accomplish this should be completely invisible to the user. As far as the user is concerned, all he need to know is "Internet address", in order to access another system. This is an address that looks like 128.64.194.1. It is actually a 32 bit number. However it is normally written as 4 decimal numbers, each representing 8 bits of the address. (The term “octet” is used by Internet documentation for such 8 bit chunks. The term “byte” is not used, because TCP/IP is supported by some computers that have byte sizes other than 8 bits).

Generally the structure of the address gives you some information about how to get to the system. We normally refer to systems by name, rather than by Internet address. When we specify a name, the network software looks it up in a database, and comes up with the corresponding Internet address. Most of the network software deals strictly in terms of the address. TCP/IP is built on “connection less” technology. Information is transferred as a sequence of “data grams”.

Each of these datagrams is sent through the network individually. There are provisions to open connections (i.e., to start a conversation that will continue for sometime). However at some level, information from those connections is broken up into datagrams, and those datagrams are treated by the network a completely separate. For example, suppose you want to transfer a 15000 octet file. Most networks can't handle a 15000 octet datagram. So the protocols will break this up into something like thirty 500 octet datagrams each. Each of these datagrams will be sent to the other end. At that point, they will be put back together into the 15000 octet file. However while those datagrams are in transmit, the network doesn't know that there is any connection between them. It is perfectly possible that datagram 14 will actually arrive before datagram 13. It is also possible that somewhere in the network, an error will occur, and some datagram won't get through at all. In that case, that datagram has to be sent again. Note by the way that the terms datagram and packet often seem to be nearly interchangeable. Technically, data gram is the right word to use when describing TCP/IP.

A data gram is a unit of data, which is what the protocols deal with.

A packet is a physical thing, appearing on an Ethernet or some wire.

In most cases a packet simply contains a data gram, so there is very little difference. However they can differ at times.

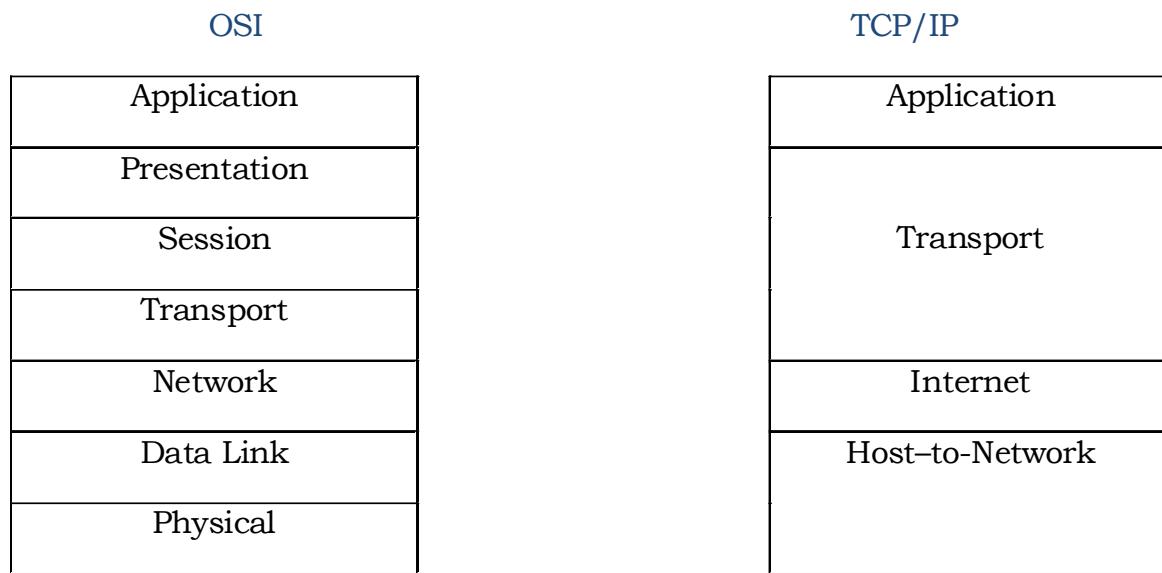


Figure 15.3 OSI and TCP comparative layers

15.3.1 HTTP (Hypertext Transfer Protocol)

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. HTTP allows an open-ended set of methods to be used to indicate the purpose of a request. It builds on the discipline of reference provided by the Uniform Resource Identifier (URI), as a location URL or name (URN) for indicating the resource on which a method is to be applied. Messages are passed to HTTP in a format similar to that used by internet mail and Multipurpose Internet Mail Extensions (MIME).

The HTTP has various built-in request methods which allow users to read a web page, or to read a web page's header, or to store a web page, or to append to a named resource or to remove the web page or to connect two existing resources or to break an existing connection between two resources.

15.3.2 FTP (File Transfer Protocol)

One of the original services on the internet was designed to allow for transferring files from one system to another. It goes by the name *ftp* which stands for file transfer protocol. Files of any type can be transferred, although you may have to specify whether the file is an ASCII or Binary file. They can be transferred to any system on the internet provided that the permissions are set accordingly.

Advantages of FTP

- (i) It is very useful to transfer the files from one network to another.
- (ii) It is an effective way to get a geographically dispersed group to co-operate on a project.
- (iii) It is popular way to share information over the internet. FTP works as a client/server process.

15.3.3 SLIP/PPP (Serial Line Internet Protocol)

Serial line IP (SLIP) was the first protocol for relaying the IP packets over dial-up lines. It defines an encapsulation mechanism, with little ease. There is no support for dynamic address assignment, link testing or multiplexing different protocols over a single link. SLIP has been largely supplanted by PPP.

PPP (Point to Point Protocols)

PPP is the internet standard for transmission of IP packets over serial lines. The PPP is currently the best solution for dial-up internet connections, including ISDN. PPP is a layered protocol, starting with a link control protocol (LCP) for link establishment, configuration and testing. Once the LCP is initialized, one or many of several network control protocols (NCPs) can be used to transport traffic for a particular protocol suite. The IP Control Protocol (IPCT), permits the transport of IP packets over a PPP link. PPP supports both synchronized and unsynchronized lines.

15.4.1 THE INTERNET

The Internet is a worldwide network of computer networks that evolved from the first network ARPAnet (Advanced Research Projects Agency network). The internet is made up of many networks each run by a different company and interconnected at peering points. It is an interconnection of large and small

networks around the globe. The common use of Internet standards allows users connected to one network to communicate with users on another network.

15.4.2 THE INTERSPACE

InterSpace is a client/server software program that allows multiple users to communicate online with real-time audio, video and text chat in dynamic 3D environments. InterSpace provides the most advanced form of communication available on the Internet today.

15.4.3 Elementary Terminology of Networks

Let us have a look at some typical hardware components of network.

(i) Nodes (Workstations)

The term nodes refer to the computer that are attached to a network and are seeking to share the resources of the network. Of course, if there were no nodes, there would be no network at all.

(ii) Server

Servers can be of two types: (1) non-dedicated servers and (2) dedicated servers

15.4.4 Types of Servers

Non-dedicated Servers

On small networks, a workstation that can double as a server is known as non-dedicated server since it is not completely dedicated to the cause of serving. Such servers can facilitate the resource-sharing among the work stations on a proportionately smaller scale. Since one computer works as a work station and as well as server, it is slower and requires more memory. The networks using such a server are known as **PEER-TO-PEER** networks.

Dedicated Servers

On bigger network installations, there is a computer reserved for server's job and its only job is to help workstations access data, software and hardware resources. It does not double-up as a workstations and such a server is known as dedicated server. The networks using such server are known as **MASTER-SLAVE** networks.

On a network, there may be several servers that allow the workstations to share specific resources. For example, there may be a server exclusively for serving files related request like storing files, deciding about their access privileges and regulating the amount of space allowed for each user. This server is known as **file server**. Similarly there may be **printer server** and **modem server**. The

printer server takes care of the printing requirements of a number of **workstations** and **the modem server** helps a group of network users use a modem to transmit long distance messages.

15.4.5 TYPES OF NETWORKS

A computer network means a group of networked components, i.e., computers that are linked by means of a communication system. A network can mean a small group of linked computers to a chain of a few hundred computers of different types (e.g., PCs, minis, mainframes, etc) spread around the world. Thus, networks vary in size, complexity and geographical spread. Mostly, computers are classified on the basis of geographical spread and on this basis, there can be 3 types of networks:

- Local Area Networks (LANs)
- Wide Area Networks (WANs)
- Metropolitan Area Networks (MANs)

Local Area Networks (LANs)

Small computer networks that are confined to a localized area (e.g., an office, a building or a factory) are known as Local Area Networks (LANs). The key purpose of a LAN is to serve its users in resource sharing. The hardware as well as software resources are shared through LANs. For instance, LAN users can share data, information, programs, printer, hard disks, modems, etc.

In a typical Lan configuration, one computer is designated as the file server. It stores all of the software that controls the network, as well as the software that can be shared by the computers attached to the network. Computers connected to the file server are called workstations. The workstations can be less powerful than the file server and they may have additional software on their hard drives. On most LANs, cables are used to connect the network interface cards in each computer.

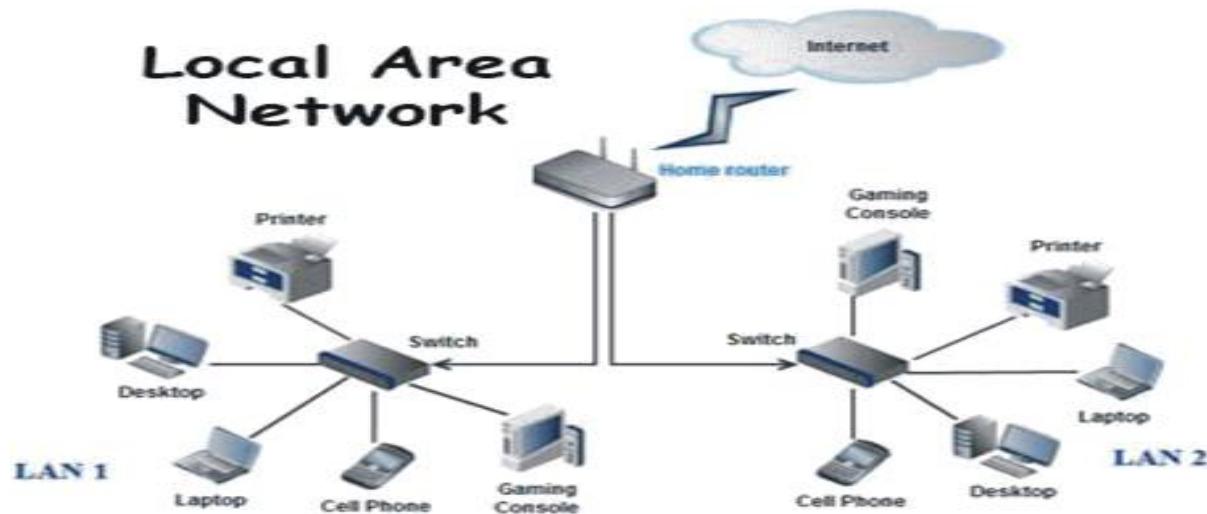
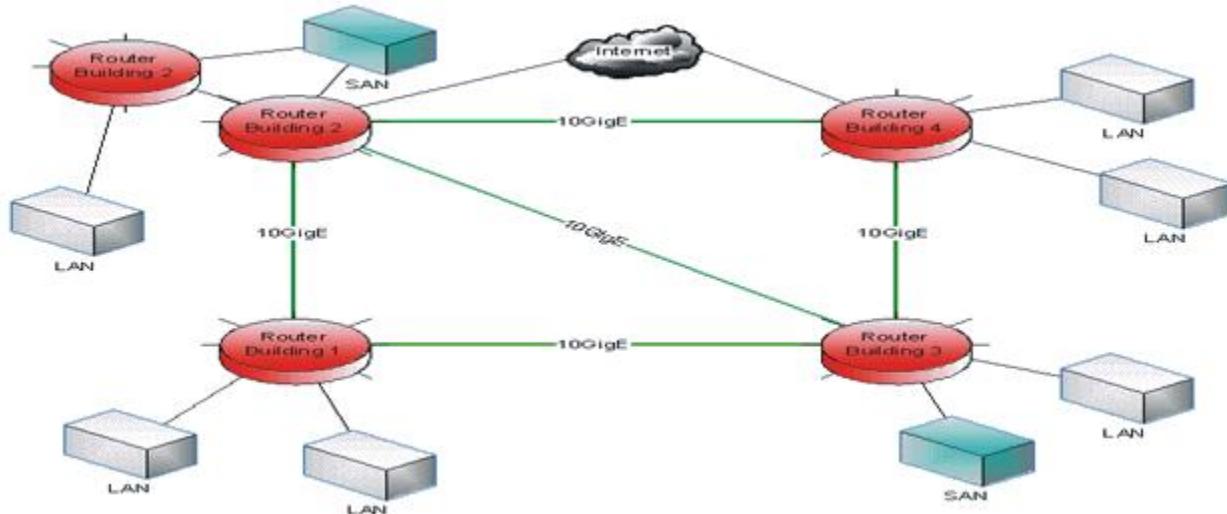


Figure 15.4 LAN topology

Metropolitan Area Networks (MANs)

Metropolitan Area Networks (MANs) are the networks spread over a city. For example, cable TV networks that are spread over a city can be termed as Metropolitan Area Networks (MANs). The purpose of a MAN is also the sharing of the hardware and the software resources among its users.



Wide Area Networks (WANs)

Figure 15.5 MAN topology

The networks spread across the countries are known as WANs. A Wide Area Networks (WANs) is a group of computers that are separated by large distances and tied together. It can even be a group of LANs that are spread across several locations and connected together to look like one big LAN. The

WANs link computers to facilitate fast and efficient exchange of information at lesser cost and higher speeds.

Computers connected to a Wide Area Networks (WANs) are often connected through public networks, such as the telephone system. Sometimes they can be connected through leased lines or satellites. The largest WAN in existence is the internet.

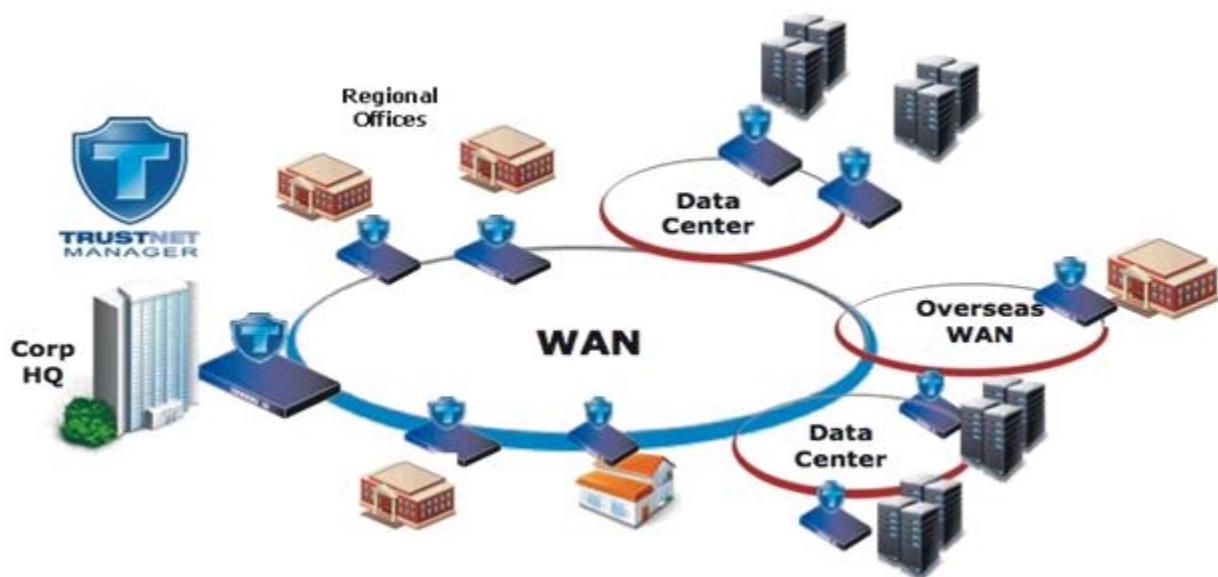


Figure 15.6 WAN topology

Difference between a LAN and a WAN

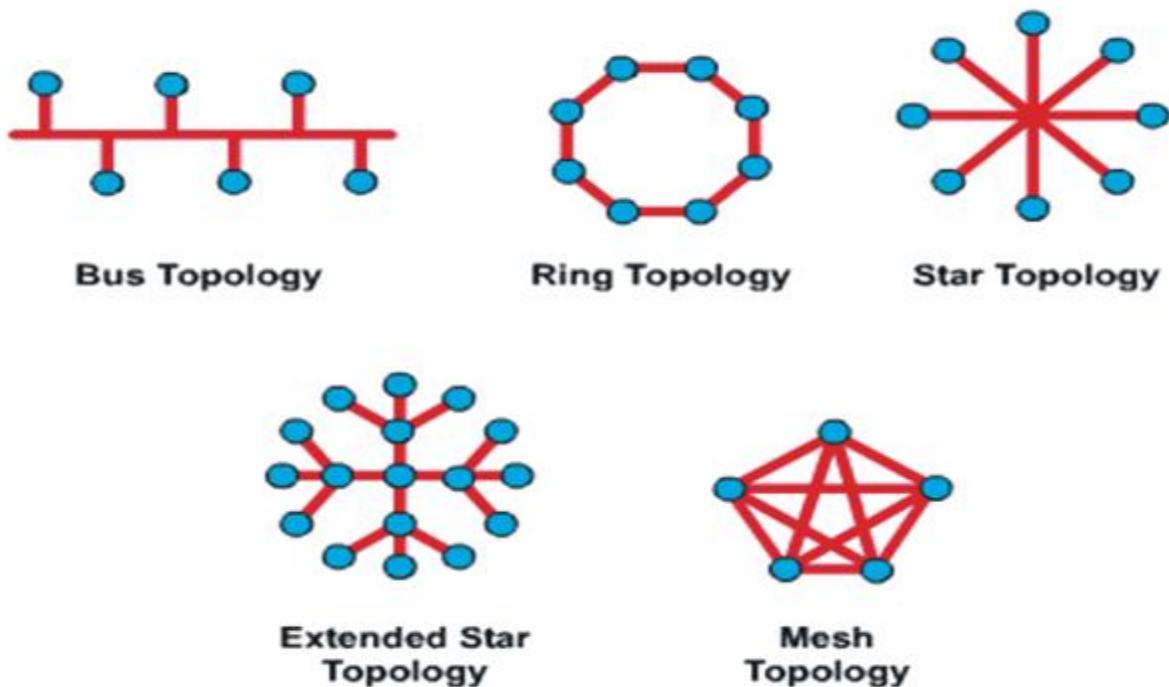
The next task is to distinguish between LANs and WANs. LANs are different in the following important respects.

1. The distance between the nodes is limited. There is an upper limit of approximately 10Kms and a lower limit of 1 meter.
2. While WANs usually operate at speeds of less than 1 mbps (one mega bits per second), LANs normally operate at between 1 and 10 mbps. Using optical fiber technology, it is possible to achieve speeds of the order of hundreds of mbps.

3. Because of the short distances involved, the error rates in LANs are much lower than in WANs. LANs error rate is thousand times lower than in WANs, so are normal.
4. The distance limitations involved in LANs normally mean that the entire network is under the ownership and control of a single organization. This is in sharp contrast to WANs, where the network is normally operated by the countries post and telecommunications authorities rather than by its users.

	LAN	WAN
1	Diameter of not more than a few kilometers.	Span entire countries.
2	A total data rate of at least several mbps.	Data rate less than one mbps.
3	Complete ownership by a single organization.	Owned by multiple organizations.
4	Very low error rates.	Comparatively higher error rates.

15.4.6 NETWORK TOPOLOGIES: The actual appearance or layout of networking



The Star Topology

This topology consists of a central node to which all other nodes are connected by a single path. It is the topology used in most existing information networks involving data processing or voice communications.

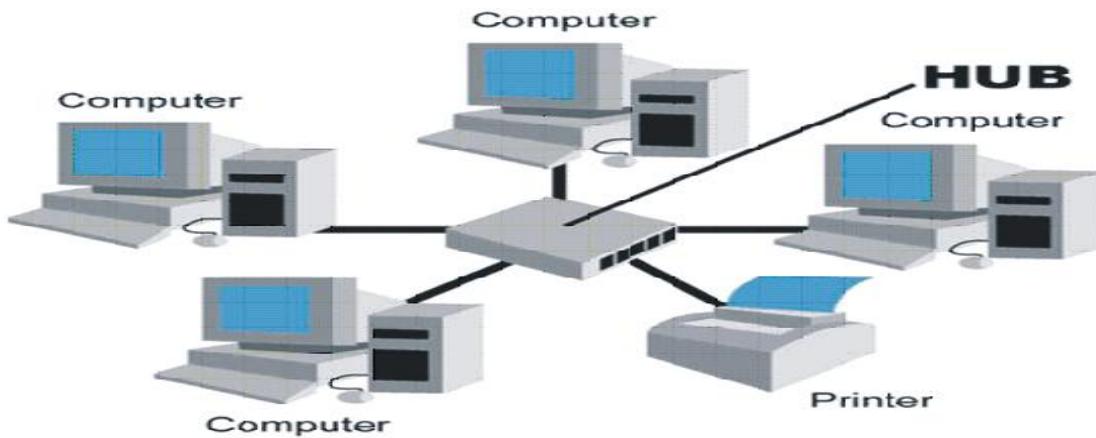


Figure 15.7 STAR topology with hub

Advantages of the Star topology

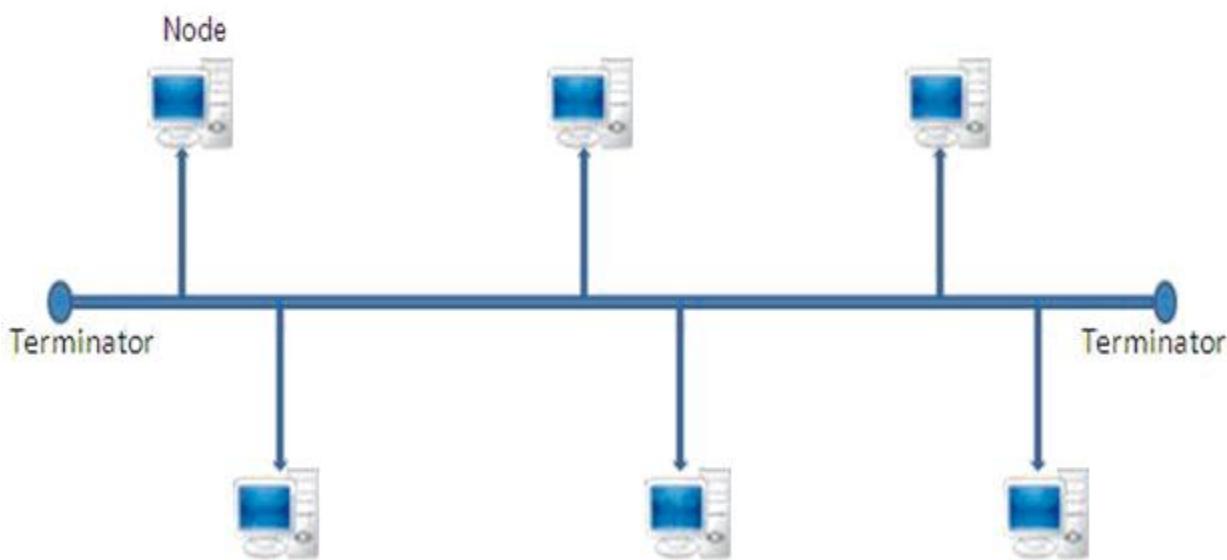
1. Ease of service. The star topology has a number of concentration points (where connections are joined). These provide easy access for service or reconfiguration of the network.
2. One device per connection. Connection points in any network are inherently prone to failure. In the star topology, failure of a single connection typically involves disconnecting one node from an otherwise fully functional network.

The Bus or Linear Topology

Another popular topology for data networks is the linear. This consists of a single length of the transmission medium (normally coaxial cable) onto which the various nodes are attached. The topology is used in traditional data communication network where the host at one end of the bus communicates with several terminals attached along its length.

The transmission from any station travels the length of the bus, in both directions, and can be received by all other stations. The bus has terminators at either end which absorb the signal, removing it from the bus.

Data is transmitted in small blocks, known as packets. Each packet has some data bits, plus a header containing its destination address. A station wanting to transmit some data sends it in packets along the bus. The destination device, on identifying the address on the packets, copies the data on to its disk.



Advantages of the linear topology

Figure 15.8 Linear topology

1. Short cable length and simple wiring layout. Because there is a single common data path connecting all nodes, the linear topology allows a very short cable length to be used. This decreases the installation cost, and also leads to a simple, easy to maintain wiring layout.
2. Resilient Architecture. The LINEAR architecture has an inherent simplicity that makes it very reliable from a hardware point of view. There is a single cable through which all the data propagates and to which all nodes are connected.
3. Easy to extend. Additional nodes can be connected to an existing bus network at any point along its length. More extensive additions can be achieved by adding extra segments connected by a type of signal amplifier known as repeater.

Disadvantages of the linear topology

1. Fault diagnosis is difficult. Although simplicity of the bus topology means that there is very little to go wrong, fault detection is not a simple matter. Control of the network is not centralized in any particular node. This means that detection of a fault may have to be performed from many points in the network.

2. Fault isolation is difficult. In the star topology, a defective node can easily be isolated from the network by removing its connection at the center. If a node is faulty on the bus, it must be rectified at the point where the node is connected to the network.
3. Repeater configuration. When BUS type network has its backbone extended using repeaters, configuration may be necessary.
4. Nodes must be intelligent. Each node on the network is directly connected to the central bus. This means that some way of deciding who can use the network at any given time must be performed in each node.

The Ring or Circular topology

The third topology that we will consider is the ring or the circular. In this case, each node is connected to two and only two neighboring nodes and is transmitted onwards to another. Thus data travels in one direction only, from node to node around the ring. After passing through each node, it returns to the sending node, which removes it.

It is important to note that data gets through rather than travels past each node. This means that the signal may be amplified before being repeated on the outward channel. node to node around the ring. After passing through each node, it returns to the sending node, which removes it.

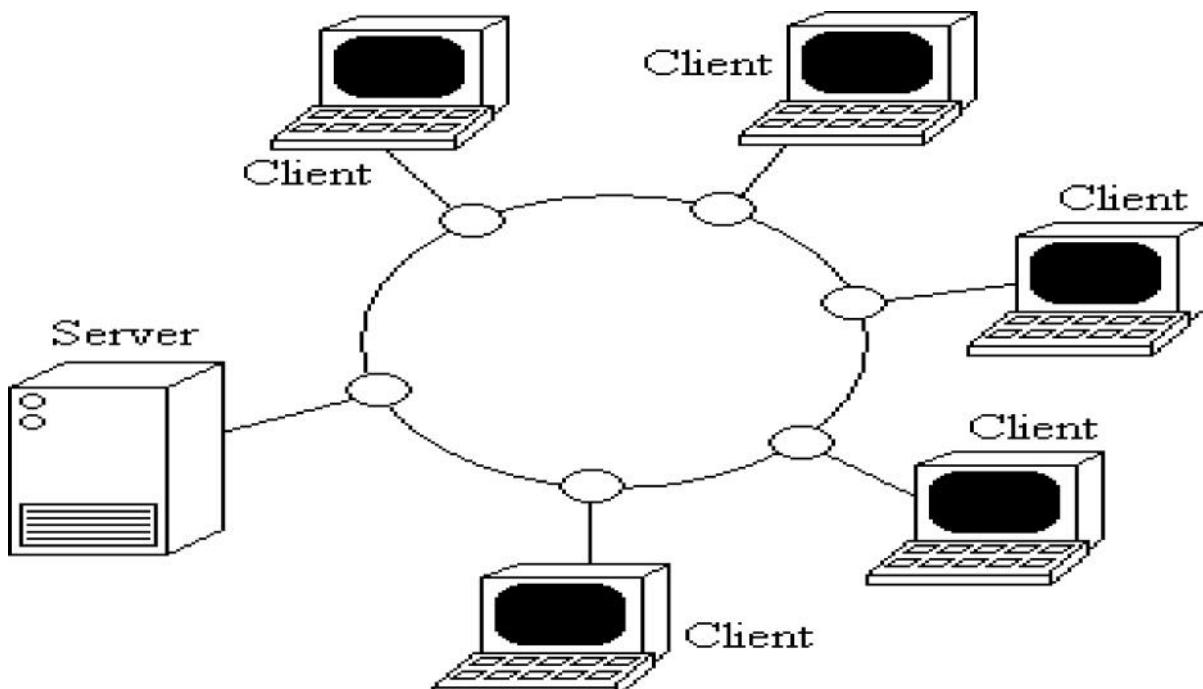


Figure 15.9 Ring topology



Figure 15.10 Ring topology

Advantages of the Ring topology

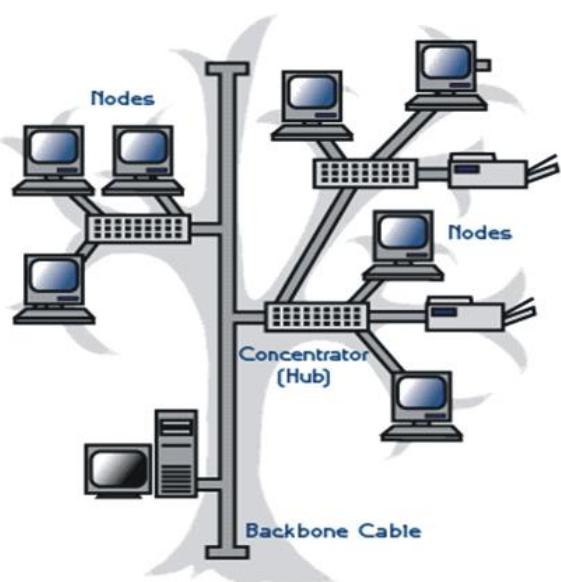
1. Short cable length. The amount of cabling involved in a ring topology is comparable to that of a bus and is small relative to that of a star. This means that less connections will be needed, which will in turn increase network reliability.
2. No wiring closet space required. Since there is only one cable connecting each node to its immediate neighbors, it is not necessary to allocate space in the building for wiring closet.
3. Suitable for optical fibers. Using optical fibers offers the possibility of very high speed transmission. Because traffic on a ring travels in one direction, it is easy to use optical fibers as a medium of transmission.

Disadvantages of the Ring topology

1. Node failure causes network failure. The transmission of data on a ring goes through every connected node on the ring before returning to the sender. If one node fails to pass data though itself, the entire network has failed and no traffic can flow until the defective node has been removed from the ring.
2. Difficult to diagnose faults. The fact that failure of one node will affect all others has serious imprecations for fault diagnosis. It may be necessary to examine a series of adjacent nodes to determine the faulty one. This operation may also require diagnostic facilities to be built into each node.
3. Network reconfiguration is difficult. It is not possible to shut a small section of the ring while keeping the majority of it working normally.

The Tree Topology

A variation of bus topology is the tree topology. The shape of the network is that of an inverted tree with the central root branching and sub branching to the extremities of the network.



Transmission in this topology takes place in the same way as in the bus topology. In both cases, there is no need to remove packets from the medium because when a signal reaches the end of the medium, it is absorbed by the terminators. Tree topology is best suited for the applications which have a hierarchical flow of date and control. Since the tree topology is a modification of a pure network topology, bus topology, it is a hybrid topology.

Figure 15.11 Tree topology

Graph Topology

In this topology, nodes are connected together in an arbitrary fashion. A link may or may not connect two or more nodes. There may be multiple links also. It is not necessary that all the nodes are connected. But if a path can be established in two nodes via one or more links is called a connected graph.

Mesh Topology

In this topology, each node is connected to more than one node to provide an alternative root in the case the host is either down or too busy. It is an extension to P-P network.

The mesh topology is excellent for long distance networking because it provides extensive backup, rerouting and pass through capabilities. Communication is possible between any two nodes on the network either directly or by passing through. This function is needed in the event of a line or node failure anywhere in the network. The mesh topology is commonly used in large internetworking environments with stars, rings and buses attached to each node. This is also idle for distributed networks.

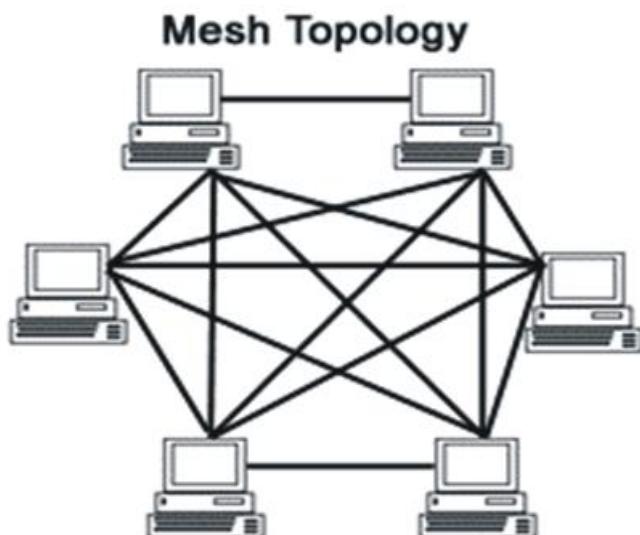


Figure 15.12 Mesh topology

When in a network each host is connected to other directly i.e., there is a direct link between each host, then the network is said to be fully connected. This characteristic is termed as full connectivity.

15.4.8 TRANSMISSION MEDIUM

By transmission media or communication channels of network, it is meant that the connecting cables or connecting media are being talked about. The cables that connect two or more workstations are the communication channels.

TWISTED PAIR CABLE

The most common form of wiring in data communication application is the twisted pair cable. As a **Voice Grade Medium** (VGM), it is the basis for most internal office telephone wiring. It consists of two identical wires wrapped together in a double helix.

Problems can occur due to differences in the electrical characteristics between the pair (e.g., length, resistance, and capacitance). For this reason, LAN applications will tend to use a higher quality cable known as **Data Grade Medium** (DGM).

Different types and categories of twisted-pair cable exist, but they all have two things in common:

- a. The wires come in pairs
- b. The pairs of wires are twisted around each other

ADVANTAGES:

1. It is simple and physically flexible.
2. It can be easily connected.
3. It is easy to install and maintain.
4. It has a low weight.
5. It is inexpensive.

DISADVANTAGES:

1. Its low bandwidth capabilities make it unsuitable for broadband applications.
2. Because of high attenuation, it is incapable of carrying a signal over long distances without the use of repeaters.
3. It supports maximum data rates 1 Mbps without conditioning and 10 Mbps with conditioning.

Types of Twisted Pair Cables

There are two types of twisted pair cables available. These are

- (i) **Unshielded Twisted Pair (UTP) Cable:** UTP cabling is used for variety of electronic communications. It is available in the following five categories:

Type	Description
CAT1	Voice-grade communications only; No data transmission
CAT2	Data-grade transmission up to 4 Mbps
CAT3	Data-grade transmission up to 10 Mbps
CAT4	Data-grade transmission up to 16 Mbps
CAT5	Data-grade transmission up to 1000 Mbps

The UTP cables can have maximum segment length of 100 meters.

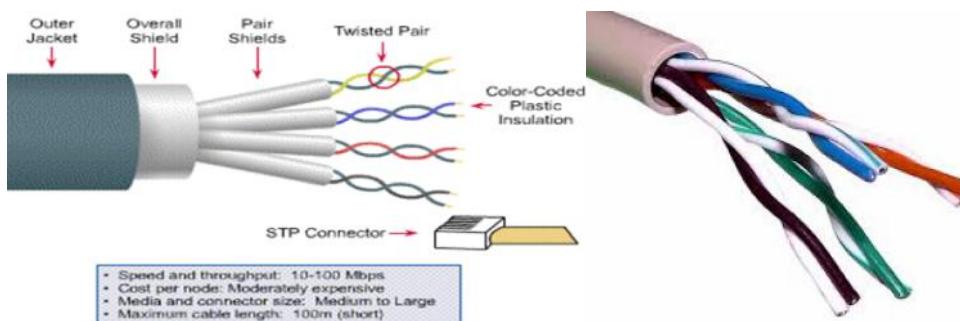


Figure 15.13 UTP cables

Figure 17.13 UTP cables

- (ii) **Shielded Twisted Pair (STP) Cable:** This type of cables comes with shielding of the individual pairs of wires, which further protects it from external interference. But these also, like UTP, can have maximum segment length of 100 meters. The advantage of STP over UTP is that it offers greater protection from interference and crosstalk

due to shielding. But it is definitely heavier and costlier than UTP and requires proper grounding at both the ends.



Figure 15.14 STP cable

Types of Coaxial Cables

The two most commonly used types of coaxial cables are **Thicknet** and **Thinnet**.

- (i) **Thicknet:** This form of coaxial cable is thicker than thinet. The thicknet coaxial cable segments can be upto 500 meters long.
- (ii) **Thinnet:** This form of coaxial cable is thinner and it can have maximum segment length of 185 meters i.e, using this cable, nodes having maximum distance of 185 meters can be joined.

Optical Fibers

Optical Fibers consist of thin strands of glass or glass like material which are so constructed that they carry light from source at one end of the fiber to a detector at the other end. The light sources used are either light emitting diodes (LEDs) or laser diodes (LDs). The data to be transmitted is modulated onto the light beam using frequency modulation techniques. The signals can then be picked up at the receiving end and demodulated. The bandwidth of the medium is potentially very high. For LEDs, this range between 20 and 150 mbps and higher rates are possible using LDs.

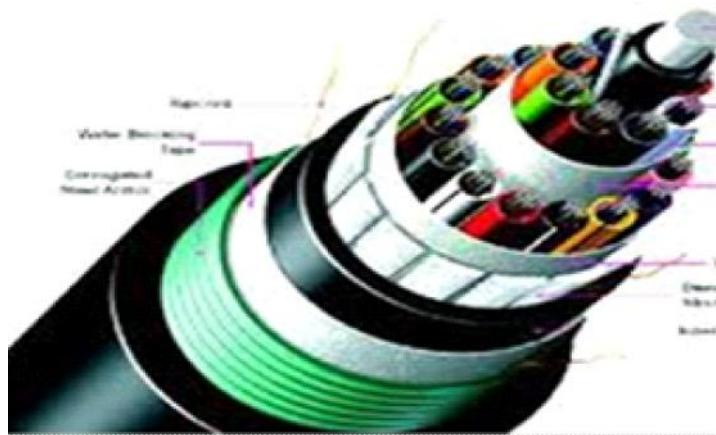


Figure 15.15 OPTICAL Fibers

Advantages:

1. It is immune to electrical and magnetic interference i.e., noise in any form because the information is travelling on a modulated light beam.
2. It is highly suitable for harsh industrial environments.
3. It guarantees source transmission and has a very high transmission capacity.
4. Fiber optic cables can be used for broadband transmission where several channels (i.e., bands of frequency) are handled in parallel and where it is

2. Signals from one signal antenna may split up and propagate by slightly different paths to the receiving antenna. When these out of phase signals recombine, they interfere, reducing the signal strength.
3. Microwave propagation is susceptible to weather effects like rains, thunder storms, etc.
4. Bandwidth allocation is extremely limited.
5. The cost of design, implementation and maintenance of microwave links is high.

Radio Wave

The transmission making use of radio frequencies is termed as radio-wave transmission.

Any radio setup has two parts:

- The **transmitter**
- The **receiver**

The transmitter takes some sort of message (it could be the sound of someone's voice, pictures for a TV set, data for a radio modem or whatever), encodes it onto a sine wave and transmits it with radio waves. The receiver receives the radio waves and decodes the message from the sine wave it receives. Both the transmitter and receiver use antennas to radiate and capture the radio signal.

ADVANTAGES:

1. Radio-wave transmission offers mobility.
2. It proves cheaper than digging trenches for laying cables and maintaining repeaters and cables if cables get broken by a variety of causes.
3. It offers freedom from land acquisition rights that are required for laying, repairing the cables.
4. It offers ease of communication over difficult terrain.

DISADVANTAGES:

1. Radio-wave communication is an insecure communication.
2. Radio-wave propagation is susceptible to weather effects like rains, thunder storms, etc.

Security of such communication links is almost nonexistent. Even so, the equipment has many advantages and is widely used by taxi repair, courier and delivery services.

Satellite (Satellite Microwave)

Radio wave can be classified by frequency and wave length. When the frequency is higher than 3 GHz, it is named microwave. Satellite communication is special case of microwave relay system. Satellite communication use the synchronous satellite to relay the radio signal transmitted from ground station. In recently, the use of wireless communication has gained more popularity. Compared to the traditional fixed wire terrestrial networks, satellite and microwave communications network features the time saving, fast implementation and broad coverage characteristics. It provides voice, fax, data and video services as well as email, file transfer, WWW internet applications. When fixed wire terrestrial communication networks are crushed by a disaster, the satellite and microwave system as a emergency backup facility will be stressed.

In satellite communication the earth station consists of a satellite dish that functions as an antenna and communication equipment to transmit and receive data from satellites passing overhead.

A number of communication satellites, owned by both government and private organizations, have been placed in stationary orbits about 22,300 miles above the earth's surface. These satellites act as relay stations for communication signals. The satellites accept data/ signals transmitted from an earth station, amplify them, and retransmit them to the other side of the earth in only one step.

Most communication satellites have multiple, independent reception and transmission devices known as transponders. In a commercial communication satellite, a single transponder is usually capable of handling a full-colour, commercial television transmission, complete with audio. Transponders for data transmission may be even larger. Some firms that market satellite communication service own a satellite. Others lease a portion of a satellite and provide transmission facilities in smaller units to ultimate users. The security in satellite transmission is usually provided by the coding and decoding equipment. Satellite communication has a number of advantages.

ADVANTAGES:

1. The area coverage through satellite transmission is quite large.
2. The laying and maintenance of intercontinental cable is difficult and expensive and this is where the satellite proves to be the best alternative.
3. The heavy usage of intercontinental traffic makes the satellite commercial attractive.

4. Satellites can cover large areas of the earth. This is particularly useful for sparsely populated areas.

DISADVANTAGES:

1. Technological limitations preventing the deployment of large, high gain antennas on the satellite platform.
2. Over-crowding of available bandwidths due to low antenna gains.
3. The high investment cost and insurance cost associated with significant probability of failure.
4. High atmospheric losses above 30 GHz limit carries frequencies.

Other Unguided Media

Apart from microwaves, radio waves and satellites, two other unguided media are also very popular. These are **infrared** and **laser** waves.

1. Infrared

This type of transmission uses infrared light to send the data. The infrared light transmits data through the air and can propagate throughout a room (bouncing off surfaces), but will not penetrate walls. The infrared transmission has become common in PDAs (Personal Digital Assistants) e.g., hand held devices like palm pilots etc. The infrared transmission is considered to be secure one.

2. Laser

The Laser transmission requires direct line-of-sight. It is unidirectional like microwave, but has much higher speed than microwaves. The laser transmission requires the use of a laser transmitter and a photo-sensitive receiver at each end. The laser transmission is point-to-point transmission, typically between buildings. But lasers have a certain disadvantage, which is: it can be adversely affected by weather.

15.4.9 SWITICHING TECHNIQUES

Different types of switching techniques are employed to provide communication between two computers. These are **circuit switching**, **message switching** and **packet switching**.

Circuit Switching

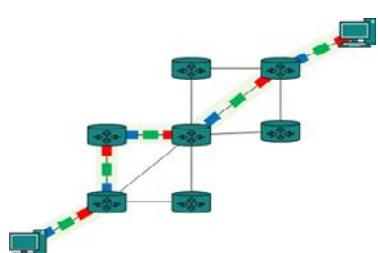


Figure 15.16 Circuit

In this technique, first the complete physical connection between two computers is established and then data are transmitted from the source computer to the destination computer. That is, when a computer places a telephone call, the switching equipment within the telephone system seeks out a physical copper path all the way from sender telephone to the receiver's telephone. The important property of this

switching technique is to setup an end to end path (connection) between computers before any data can be sent.

Message Switching

In this technique, the source computer sends the data or the message to the switching office first, which stores the data in the buffer. It then looks for a free link to another switching office and then sends the data to this office. This process is continued until the data is delivered to the destination computers. Owing to its working principle, it is also known as **store and forward**.

That is, store first (in switching office), forward later, one jump at a time.

Packet Switching

With message switching, there is no limit on block size, in contrast, packet switching places a tight upper limit on block size. A fixed size of packet which can be transmitted across the network is specified.

15.4.10 Communication Modes

The communication mode defines in which data can flow depending upon the type media used. They are Simplex, Half Duplex and Full Duplex.

Simplex

On this panel there is only one interface that is a transmitter and all other interfaces are receivers. The full bandwidth is completely for signals travelling across transmitter to receiver or receivers. On this channel transmitting interface cannot receive and receiving interface cannot transmit. For example Radio, TV, etc uses Simplex channels.

Half Duplex

On this channel each interface works as transmitter and receiver, but only one interface can transmit at a time. The full bandwidth of a channel is available to the transmitting interface which will not receive while transmitting. Generally it is used in Walkie-Talkies, Marine/Aviation, etc use Half Duplex channel.

Full Duplex

This channel has two ends, each serving as transmitter and receiver. Each interface can transmit and receive at the same time. The modern telephone system use Full Duplex channels. It is more expensive due to hardware for increased number of channels and bandwidth.

Figure 15.17 Message

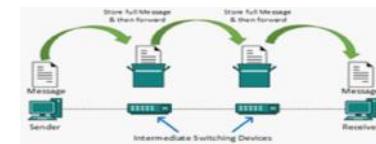
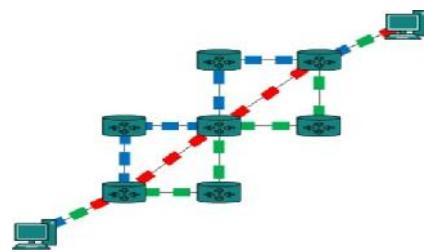


Figure 15.18 Packet

15.4.11 NETWORK DEVICES

In functioning of networks, many devices play important roles. Here, in this section we are going to discuss a few of them.

Modem (Modulator and Demodulator)

Modems allow you to combine the power of your computer with the global reach of the telephone system.

Because ordinary telephone lines cannot carry digital information, a modem changes the digital data from your computer into analog data, a format that can be carried by telephone lines. In a similar manner, the modem receiving the call then changes the analog signal back into digital data to the computer. This shift of digital data into analog data back again, allows two computers to communicate with one another, called modulation or demodulation.

With a modem you can send faxes to the office or important customers without leaving your computer. And with an online or internet connection, you can share recipes with fellow gourmets catch up on the latest news, view a weather map from Singapore, keep in touch with distant friends by electronic mail, the World Wide Web and much more.

Working on Modem

Modem converts digital signals to A/F (audio frequency) tones which are in the frequency range that the telephone lines can transmit and also it can convert transmitted tones back to digital information.

After the power is turned On in DTE (Data Terminal Equipment) and DCE (Data Communication Equipment), the terminal runs for self check, it asserts the Data Terminal Ready (DTR) signal to tell the modem that it is ready.

When modem is powered up and ready to transmit data, the modem will assert the Data Set Ready (DSR) signal to the terminal. Under the manual or terminal control the modem dials up the computer on the other end. If the computer is available it will send back a specified tone.

Now when the terminal has a character ready to send, it will assert the Request-To-Send (RTS) signal to the modem. The modem assert its Carrier Detect (CD) signal to the terminal to indicate that it has established contact with the computer. When the modem is fully ready to transmit the data it asserts Clear-To-Send (CTS) signal back to the terminal. The terminal then sends serial data characters to the modem. When the terminal has sent all the characters, it needs to make its RTS signal high. This causes the MODEM to unasserted its CTS signal and stop transmitting. Similar handshakes occur between modem and computers on other side also.

Modems are of two types :

1.Internal modems: The modems that are fixed within the computer.

2.External modems: The modems that are connected externally to a computer as other peripherals are connected.

Ethernet Card

As mentioned earlier, Ethernet is a LAN architecture developed by Xerox Corp in association with DEC and Intel. Ethernet uses bus or star topologies and can support data transfer rates of up to 10 Mbps.

The computers that are part of Ethernet have to install a special card called Ethernet card.

An Ethernet card contains connections for either Coaxial or Twisted pair cables (or both). If it is designed for coaxial cable, the connection will be BNC. If it is designed for twisted pair, it will have a RJ-45 connection. Some Ethernet cards also contain an AUI connector. This can be used to attach coaxial, twisted pair or fiber optic cables to an Ethernet card. When this connection is used, there is always an external transceiver attached to the workstation. These days many computers include an option for a pre-installed Ethernet Card.

Hub

A hub is a hardware device used to connect several computers together. A hub that contains multiple independent but connected modules of network and internetworked equipment. A similar term is concentrator. A concentrator is a device that provides a central connection point for cables from workstations, servers and peripherals. In a star topology, twisted pair wire is run from each workstation to a central concentrator.

Basically, hubs are multi slot concentrators into which a number multi port cards can be plugged to provide additional access as the network grows in size.

Hubs can be either passive or active.

1.Active Hubs: Electrically amplify the signal as it moves from one connected device to another. Active concentrators are used like repeaters to extent the length of a network.

2. Passive Hubs: Allow the signals to pass from one computer to another without any change.

Hubs usually can support 8, 12 or 24 RJ-45 ports. These are often used in a star or star wired ring topology and requires specialized software for port management.

Switch

A switch is a device that is used to segment networks into different sub networks called subnets or LAN segments. Segmenting the network into smaller subnets prevents traffic overloading in a network.

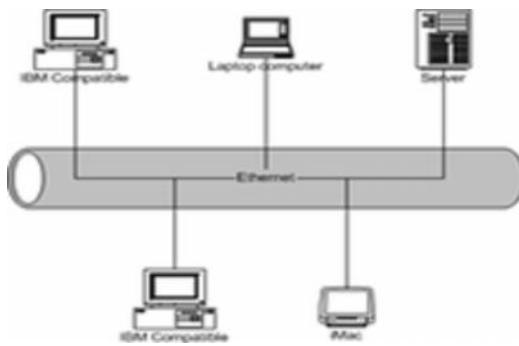


Figure 15.19 Modem with systems

How a switch functions

To insulate the transmission from the other ports, the switch establishes a temporary connection between the source and destination and then terminates the connection once the conversation is done.

Repeater

A repeater is a device that amplifies a signal being transmitted on the network. It is used in long network lines, which exceed the maximum rated distance for a single run.

Over distance, the cables connecting a network lose the signal transmitted. If the signal degrades too much, it fails to reach the destination. Or if it does arrive, the degradation of the message makes it useless. Repeaters can be installed along the way to ensure that data packets reach their destination. Repeaters are of two kinds: **amplifier and signal repeater**. The first merely amplifies all incoming signals over the network. However, it amplifies both the signal and any concurrent noise. The second type collects the inbound packet and then retransmits the packet as if it were starting from the source station.

Bridge

A bridge is a device that lets you link two networks together. Bridges are smart enough to know which computers are on which side of the bridge, so they allow only those messages that need to get to the other side of the bridge. As a packet arrives at the bridge, the bridge examines the physical destination address of the packet. The bridge then decides whether or not to let the packet cross.

Router

A device that works like a bridge but can handle different protocols is known as a router. For example, a router can link Ethernet to a mainframe.

If the destination is unknown to a router it sends the traffic (bound to unknown destination) to another router (using logical addresses) which knows the destination. A router differs from a bridge in a way that former uses logical addresses and the latter uses physical addresses.

A switch is responsible for filtering i.e., transforming data in a specific way and for forwarding packets (a piece of message being transmitted) between LAN segments. Switch support any packets protocol.

LANs that are segmented through switches are called switched LANs. In the case of Ethernet LANs, they are called switched Ethernet LANs.

How a Router functions

Compared to the hubs and switches, routers are smarter. Routers use a more complete packet address to determine which router or workstation should receive each packet next. Based on a network road map called a routing table, routers can help ensure that packets are travelling the most efficient paths to their destinations. If a link between two routers fails, the sending router can determine an alternate route to keep traffic moving.

15.5.1 Gateway

A Gateway is a device that connects dissimilar networks. A gateway operates at the highest layer of network abstraction. It expands the functionality of routers by performing data translation and protocol conversion. It is needed to convert Ethernet traffic from the LAN, to SNA (Systems Network Architecture) traffic on a legacy system. It then routes the SNA traffic to the mainframe. When the mainframe answers, the reverse process occurs.

A gateway is actually a node on a network that serves as an entrance to another network. In enterprises, the gateway is the computer that routes the traffic from a workstation to the outside network that is serving the web pages. In homes, the gateway is the ISP that connects the user to the internet.

In enterprises, the gateway node often acts as a proxy server (a machine that is not actually a server but appears as a server) and a firewall (a system designed to prevent unauthorized access to or from a private network). The gateway is also associated with both a router, which use headers and forwarding tables to determine where packets are sent, and a switch, which provides the actual path for the packet in and out of the gateway.

Wireless Vs Mobile Computing

Wireless refers to the method of transferring information between a computing device, such as Personal Data Assistant (PDA) and a data source, such as an agency data base server, without a physical connection. Wireless communication is simply data communication without the use of the physical connectivity. Not all wireless communications technologies are mobile.

Mobile simply describing a computing device that is not restricted to a desktop. A mobile device may be a PDA, a small cell phone or web phone, a laptop computer or any other of numerous other devices that allow the user to complete the computing task without being tethered, or connected to a network.

Mobile computing does not necessarily require wireless communication. Infact, it may not require communication between devices at all.

Wireless communication is simply data communication without the use of landlines. This may involve cellular telephone, two way radio, fixed wireless, LASER or satellite communications. Here the computing device is continuously connected to the base network.

Mobile or untethered, computing means that the computing device is not continuously connected to the base or central network. Mobile devices include PDAs, Laptop computers and many of today's cell phones. These products may communicate with a base location with or without a wireless connection.

GSM

GSM is short for Global System for Mobile communications, which is one of the leading digital cellular systems. The GSM standard for digital cell phones was established in Europe in the mid 1980s.

In covered areas, cell phone users can buy one phone that will work anywhere where the standard is supported. To connect to the specific service providers in these different countries, GSM uses simply switch Subscriber Identification Module (SIM) cards. SIM cards are small removable disks that slip in and out of GSM cell phones. They store all the connection data and identification numbers you need to access a particular wireless service provider.

GSM uses narrow band (TDMA), which allows eight simultaneous calls on the same radio frequency. TDMA is short for (Time Division Multiple Access), a technology for delivering digital wireless service using Time Division Multiplexing. TDMA works by dividing a radio frequency into time slots and then allocating slots to multiple calls. In this way, a single frequency can support multiple, simultaneous data channels. GSM operates in the 900 MHz and 1800 MHz bands.

15.6.1 What is a SIM card?

The SIM – Subscriber Identity Module – is a chip card, the size of a postage stamp. A SIM is a tiny computer chip that gives a cellular device its unique phone number. It has memory, a processor and the ability to interact with the

user. Current SIMs typically have 16 to 64 Kb of memory, which provides plenty of room for storing hundreds of personal phone numbers, text messages and other data.

CDMA

CDMA is short Code Division Multiple Access, a digital cellular technology that uses spread spectrum techniques. Unlike competing systems, such as GSM, that use TDMA, CDMA does not assign a specific frequency to each user. Instead, every channel uses the full available spectrum. Individual conversations are encoded with a pseudo random digital sequence. CDMA is a form of spread spectrum, which simply means that data is sent in small pieces over a number of the discrete frequencies available for use at any time in the specified range. All of the users transmit in the same wide band chunk of spectrum. Each user's signal is spread over the entire bandwidth by a unique spreading code. At the receiver end, that same unique code is used to recover the signal.

WLL (Wireless in local loop)

Wireless in local loop (WLL or WILL), is meant to serve subscribers at homes or offices. Wireless in local loop is analogous with local telephone service, but much more capable. A WLL system serves a local area by deploying multiplicity of multi channel transmit/receive bases stations (transceivers) that are within line of site of the intended customers. Each customer is equipped with a mini station of low power, into which the telephone is connected. The WLL unit consists of a radio transceiver and the WLL interface assembled in box. Two cables and a telephone connector are the only outlets from the box; one cable connects to a directional antenna and a phone receptacle to connect to a common telephone set. Example, a fax or modem could also be connected for fax or computer communication.

When calls are made from the telephone, it signals the base station for a connection, which is subsequently established through a switch center, exactly as in conventional telephony. An incoming call is identified at the switch center and rooted to the base station assigned to serve the telephone being called. The wireless connection is then made, and the call is completed in a conventional manner.

The WLL system can operate with GSM handsets/mobile units, as well as with GSM compatible subscriber units. The system is transparent to the central office and subscribers, and interfaces with the most standard central office switches and subscriber telephone equipment.

Advantages of WLL

- (i) WLL facilities do not significantly suffer from weather damage, vandalism and accidents.
- (ii) WLL system offers better bandwidth than traditional telephone systems.
- (iii) WLL system has much better bandwidth, superior customer service features and quality can be provided.

15.7.1 GPRS

GPRS is the abbreviation for General Packet Radio Service. GPRS is used for wireless communication using a mobile device. With this service you can access the internet, send emails and large data, real time news, download games and watch movies.

How does GPRS work?

You must be aware of how files are transferred from one location to another on your computer. They are broken down into packets and sent across. Similarly GPRS also uses the same function to transfer data through a network. The information is split into the smaller units or packets and sent through the network and is reassembled at the receiving end. GPRS provides a high speed data transfer, typically between 56 kilo bits per second to 114 kilo bits per second. A user of the GPRS network is charged only on the amount data is sent or received as opposed to the duration of the connection.

1G, 2G, 3G, 4G and 5G Networks

The “G” in wireless networks refers to the “generation” of the underlying wireless network technology. Technically generations are defined as follows.

1G Networks:

(NMT,C-Nets, AMPS, TACS) are considered to be the first analog cellular systems, which started early 1980s. There were radio telephone systems even

before that. 1G networks were conceived and designed purely for voice calls with almost no consideration of data services.

2G Networks:

(GSM, CDMAOne, D-AMPS) are the first digital cellular systems launched early 1990s, offering improved sound quality, better security and higher total capacity. GSM supports circuit switched data (CSD), allowing users to place dial-up data calls digitally, so that the networks switching station receives actual ones and zeros rather than the screech of an analog modem.

2.5G Networks: (GPRS, CDMA2000 1x) are the enhanced versions of 2G networks with theoretical data rates upto about 144k bit/s. GPRS offered the first always on data service.

3G Networks:

(UMTS FDD and TDD, CDMA 2000 1x EVDO, CDMA 2000 3x, TD-SCDMA, EDGE) are newer cellular networks that have data rates of 384k bit/sec and more. The UN's IMT – 2000 standard requires stationary speeds of 2Mbps and mobile speeds of 384kbps for a “true” 3G.

3G is a specification for the third generation (analog cellular was the first generation, digital PCS the second) of mobile communications technology. 3G promises increased bandwidth, up to 384 Kbps when a device is stationary or moving at pedestrian speed, 128 Kbps in a car and 2Mbps in fixed applications. UMTS (Universal Mobile Telecommunication System) is a broadband, packet-based transmission of text, digitized voice, video, and multimedia at data rates up to and possibly higher than 2 megabits per second (Mbps).

4G Network:

Based on the requirements for seamless interaction between networks, 4G is characterized by the following key attributes:

- (i) Support for Multiple Applications and Services** — Efficient support for unicast, multicast and broadcast services and the applications that rely on them. Prompt enforcement of Service Level Agreements (SLA) along

with privacy and other security features. Minimally, service classes include delay sensitive, loss sensitive, delay and loss sensitive and best effort.

(ii) **Quality of Service** — Consistent application of admission control and scheduling algorithms regardless of underlying infrastructure and operator diversity.

(iii) **Network Detection and Network Selection** — A mobile terminal that features multiple radio technologies or possibly uses software-defined radios if economical, allows participation in multiple networks simultaneously, thereby connecting to the best network with the most appropriate service parameters (cost, QoS and capacity among others) for the application. This requires establishing a uniform process for defining eligibility of a terminal to attach to a network and to determine the validity of link layer configuration.

5G Network:

The cellular concept was introduced in 5G Technology stands for 5th Generation Mobile technology. 5G technology has changed the means to use cell phones within very high bandwidth. User never experienced ever before such a high value technology. Nowadays mobile users have much awareness of the cell phone (mobile) technology. The 5G technologies include all type of advanced features which makes 5G technology most powerful and in huge demand in near future.

A new mobile generation has appeared every 10th year since the first 1G system (NMT) was introduced in 1981, including the 2G (GSM) system that started to roll out in 1992, 3G (W-CDMA/FOMA), which appeared in 2001, and “real” 4G standards fulfilling the IMT-Advanced requirements, that were ratified in 2011 and products expected in 2012-2013. Predecessor technologies have occurred on the market a few years before the new mobile generation.

KEY CONCEPTS OF 5G:

- Real wireless world with no more limitation with access and zone issues.
- Wearable devices with AI capabilities.
- Internet protocol version 6 (IPv6), where a visiting care-of mobile IP address is assigned according to location and connected network.
- One unified global standard.
- Pervasive networks providing ubiquitous computing: The user can simultaneously be connected to several wireless access technologies and

seamlessly move between them. These access technologies can be a 2.5G, 3G, 4G or 5G mobile networks, Wi-Fi, WPAN or any other future access technology. In 5G, the concept may be further developed into multiple concurrent data transfer paths.

- Cognitive radio technology, also known as smart-radio: allowing different radio technologies to share the same spectrum efficiently by adaptively finding unused spectrum and adapting the transmission scheme to the requirements of the technologies currently sharing the spectrum. This dynamic radio resource management is achieved in a distributed fashion, and relies on software defined radio.
- High Altitude stratospheric Platform Station (HAPS) systems.

Features of 5G Technology:

- 5G technology offer high resolution for crazy cell phone user and bi-directional large bandwidth shaping. The advanced billing interfaces of 5G technology makes it more attractive and effective.
- 5G technology also providing subscriber supervision tools for fast action.
- The high quality services of 5G technology based on Policy to avoid error.
- 5G technology is providing large broadcasting of data in Gigabit which supporting almost 65,000 connections.
- 5G technology offer transporter class gateway with unparalleled consistency.
- The traffic statistics by 5G technology makes it more accurate.
- Through remote management offered by 5G technology a user can get better and fast solution.
- The remote diagnostics also a great feature of 5G technology.
- The 5G technology is providing up to 25 Mbps connectivity speed.
- The 5G technology also support virtual private network.
- The new 5G technology will take all delivery service out of business prospect
- The uploading and downloading speed of 5G technology touching the peak.
- The 5G technology network offering enhanced and available connectivity just about the world.

EDGE

The new EDGE air interface has been developed specifically to meet the bandwidth needs of 3G. Enhanced Data rates for Global Evolution (EDGE) are a ratio based mobile high speed data standard. It allows data transmission speeds of 384 kbps to be achieved when all eight time slots are used. In fact, EDGE was formerly called GSM384. This means a maximum bit rate of 48 kbps per time slot. Even higher speed may be available in good ratio conditions. EDGE is considered an intermediate step in the evolution to 3G WCDMA (Wideband CDMA), although some carriers are expected to stop short of that final step.

15.8.1 Applications in networking

SMS

Short Message Service (SMS) is the transmission of short text messages to and from a mobile phone, fax machine and/or IP address. Messages must be no longer than some fixed number of alpha-numeric characters and contain no images or graphics. Once a message is sent, it is received by a Short Message Service Center (SMSC), which must then get it to the appropriate mobile device.

To do this, the SMSC sends a SMS request to the home location register (HLR) to find the roaming customer. Once the HLR receives the request, it will respond to the SMC with the subscriber's status: (1) inactive or active (2) where subscriber is roaming.

Chat

Chatting : Realtime communication between two users via computer. In telephone conservations, you say something, people hear it and respond, and one can hear their responses on the spot and can reply instantly. In the same manner, in chatting, you type a message on your screen, which is immediately received by the recipient; then the recipient can type a message in response to your message, which is received by you instantly.

Video Conferencing

A video conference is a live, visual connectio between two or more perople residing in seprate locations for the purpose of communication. People who have a multimedia PC with camera and video compression hardware, access to internet over an ordinary telephone line, and videophone software can see each other while talking, which is what is called Video conferencing.

15.8.2 Wi-Fi

Wi-Fi refers to **Wireless Fidelity**, which lets you connect to the internet without a direct line from your PC to the ISP. For Wi-Fi to work, you need:

- A broadband internet connection.
 - A wireless router, which relays your Internet connection from the “wall” to the PC.
 - A laptop or desktop with a wireless internet card or external wireless adapter.

Transmitting computer data without wires makes your data especially susceptible to hackers, computer users who can intercept your connection and steal your data. If you decide to use Wi-Fi at home, be sure that the network you set up is security enabled.

Wi-Fi Hotspots

A hotspot is a venue that offers Wi-Fi access. The public can use a laptop, Wi-Fi phone or other suitable portable devices to access the internet through a WiFi Hotspot. Hotspots are public locations (such as libraries, hotels, airports, etc) with free or fee-based wireless internet access. There are Wi-Fi hotspots in thousands of locations around the world.

WiMax

WiMax is a wireless digital communications system. WiMax can provide Broadband Wireless Access (BWA) up to 30 miles (50 km) for fixed stations and 3-10 miles (5-15 km) for mobile stations. WiMax requires a tower called WiMax Base Station, similar to a cell phone tower, which is connected to the Internet using a standard wired high-speed connection. But as opposed to a traditional Internet Service Provider (ISP), which divides that bandwidth among customers via wire, it uses a microwave link to establish a connection. In other words, WiMax does not depend on cables to connect each end-point, the internet connectivity to an end-user is provided through microwave link between the tower and the user-endpoint, known as WiMax Subscriber unit.

15.9.1 Network Security

The networking offers endless possibilities and opportunities to every user of it, alone with convince. But this convinces and endless benefits are not free from risks as there are many a risks to network security.

While ensuring network security, the concerns are to make sure that only legal or authorized user and programs gain access to information resources like databases. Also, certain control mechanisms are setup to ensure that properly authenticated users get access only to those resources that they are entitled to

use. Under this type of security, mechanisms like authorization, authentication, encrypted smart cards, biometrics and firewalls, etc are implemented.

The problems encountered under network security can be summarized as follows:

1. Physical security holes. When individuals gain unauthorized physical access to a computer and temper with files. Hackers do it by guessing passwords of various users and then gaining access to the network systems.
2. Software security holes. When badly written programs or 'privileged' software are compromised into doing things that they should not be doing.
3. Inconsistent usage holes. When a system administrator assembles a combination of hardware and software such that the system is seriously flawed from a security point of view.

PROTECTION METHODS

1. Authorization: It determines whether the service provider has granted access to the web service to the requestor. Basically, authorization confirms the service requestors credentials. It determines if the service requestor is entitled to perform the operation, which can range from invoking the web service to executing a certain part of its functionality. Authorization is performed by asking the user a legal login ID. If the user is able to provide a legal login ID, he/she is considered an authorized user.

2. Authentication: It ensures that each entity involved in using a web service—the requestor, the provider and the broker (if there is one) – is what it actually claims to be. Authentication involves accepting credentials from the entity and validating them against an authority.

Authentication also termed as password protection as the authorized user is asked to provide a valid password and if he or she is able to do this, he or she considered to be an authentic user.

3. Encrypted Smart Cards: Passwords in a remote login session generally pass over the network in unencrypted form; any hacker can simply record it and can use it later maliciously to corrupt data/files or to harm anyone etc. To counter such threats newer approaches are suggested such as encrypted smart cards.

An encrypted smart card is a hand held smart card that can generate a token that a computer system can recognize. Every time a new and different token is generated, which even though cracked or hacked, cannot be used later.

4. Bio Metric Systems: They form the most secure level of authorization. The Biometric systems involve some unique aspects of a person's body such as finger prints, retinal patterns, etc to establish his/her identity.

5. Firewall: A system designed to prevent unauthorized access to or from a private network is called firewall. They can be implemented in both hardware and software or a combination of both. Firewalls are frequently used to prevent

unauthorized internet users from accessing private networks connected to the internet, especially intranets. All messages entering or leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria.

There are several types of firewall techniques.

- (i) **Packet Filter:** Looks at each packet entering or leaving the network and accepts or rejects it based on user defined rules. It is fairly effective and transparent to users, but it is difficult to configure. In addition, it is susceptible to IP spoofing.
- (ii) **Application gateway:** It applies security mechanisms to specific applications, such as FTP and Telnet Servers. This is very effective, but can impose performance degradation.
- (iii) **Circuit Level Gateway:** It applies security mechanisms when a connection is established. Once the connection has been made, packets can flow between the hosts without further checking.
- (iv) **Proxy Server:** It intercepts all messages entering and leaving the network. The proxy server effectively hides the true network addresses.

15.10.1 Cookies

Cookies are messages that a web server transmits to a web browser so that a web server can keep track of the user's activity on a specific web site.

Hackers and Crackers

The **Crackers** malicious programmers who break into secure systems whereas **Hackers** are more interested in gaining knowledge about computer systems and possibly using this knowledge for playful pranks.

Cyber Law

Cyber Law is a generic term, which refers to all the legal and regulatory aspects of internet and the WWW.

India's IT Act

In India the cyber laws are contained in the information technology act, 2000 which was notified on 17 October 2000. It is based on the United Nations Commission for International Trade Related Laws (UNCITRAL) model law.

The IT act aims to provide the legal infrastructure for ecommerce in India by governing the transactions through the internet and other electronic medium.

15.11.1 Viruses

Computer Virus is a malicious program that requires a host and is designed to make a system sick, just like a real virus. Viruses can spread from computer to computer and they can replicate themselves. Some viruses are categorized as harmless pranks, while others are far more malicious. Broadly three types of viruses are:

- 1. File Infectors** – These types of viruses either infect executable files or attach themselves to a program file and create duplicate files.
- 2. Boot Sector Viruses** – Install themselves on the beginning tracks of a hard drive or the Master Boot Record or simply they change the pointer to an active boot sector.
- 3. Macro Viruses** – Infect data files like electronic spreadsheets or databases of several software packages.
- 4. Network Viruses** – These virus use protocols and commands of computer network to spread themselves on the network. Generally they use email or any data transfer files to spread themselves on the network.

Most viruses are spread by email attachment and warn them to be suspicious of any files attached to unsolicited messages.

The following are characteristics of a computer virus.

1. It is able to replicate
2. It requires a host program as a carrier
3. It is activated by external action
4. Its replication ability is limited to the system.

Virus Prevention

Virus Prevention is not a difficult task. All you need to be is extra careful and ensure to follow the following guidelines to lead virus free computing life.

1. Never use a “Foreign” disk or CD without scanning it for viruses.
2. Always scan files downloaded from the internet or other sources.
3. Never boot your PC from a floppy unless you are certain that it is virus free.
4. Write protect your disks.
5. Use licensed software.
6. Password protect your PC to prevent unattended modification.
7. Install and use antivirus software.
8. Keep antivirus software up to date.

Some of the antivirus are Kaspersky , Quick heal, K7, Norton 360, Micro trend titanium, AVG, Panda, ESET Nod32, Avast.McAfee etc.,

Cloud tecnology: Cloud technology or cloud computing as it is more commonly known today is a computing platform widely used by Information Technology (IT) Service Companies.

Review questions

One mark questions:

1. What is networking.
2. What is server?
3. What is client ?
4. What is topology?
5. Expand 2G.
6. What is a virus?
7. What is chatting?
8. What is cyber law?
9. What are cookies?
- 10.What are Hackers?

Two marks questions:

1. List the Goals for networking.
2. What do you mean by transmission modes?
3. Which are the switching technology used ?
4. What is SIM card ?
5. What is network security?

Three marks questions:

1. Explain the HTTP ?
2. Classify and explain servers.
3. Explain the types of networking.
4. Explain the cables and different types of cables used in transmission?
5. List the differences between simplex, half duplex and full duplex.
6. Explain the applications of networking?

Five marks questions:

1. Explain the working of OSI and TCP/IP?
2. Explain various networking devices used?
3. What is topology explain in detail.
4. What is gateway? Explain.
5. Explain the network security in detail?
6. Give the measures for preventing virus?

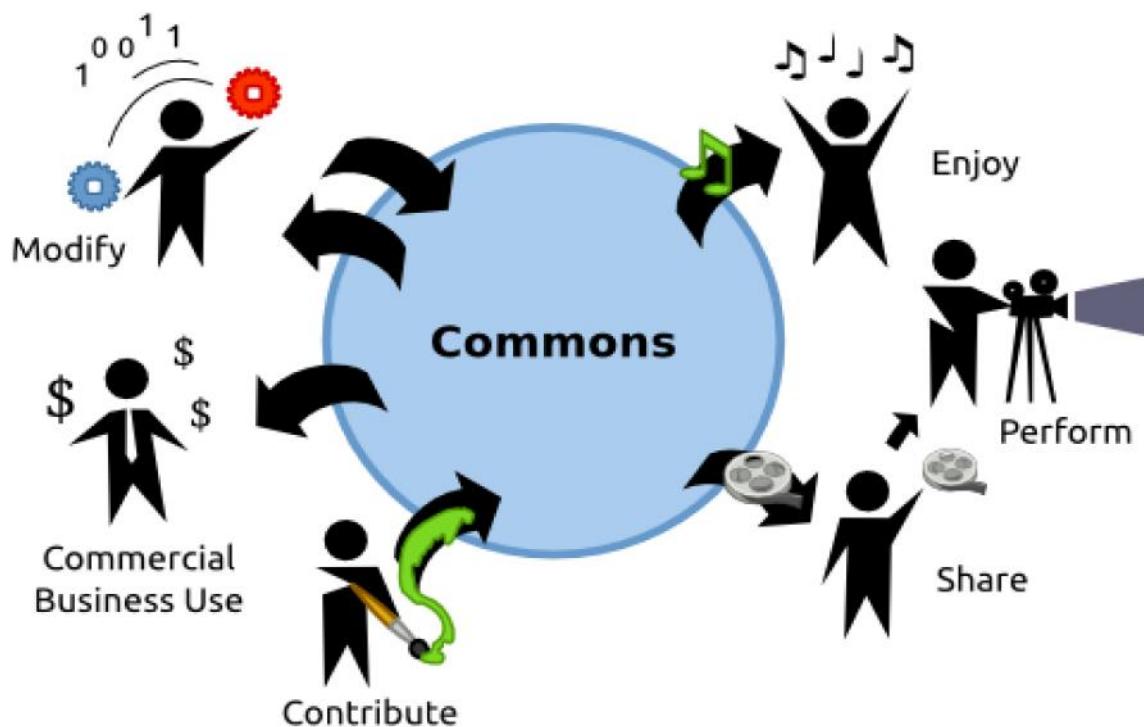
CHAPTER 16

Internet and Open source concepts**OBJECTIVES**

- Definitions and terms used in internet and web sites
- Various web browsers.
- To understand the free ware software.
- how and methods of Computer viruses attacks

Making Use of Creativity

Not everything on the Internet is in the commons. But the **following** permissions should all be available for real common resources.



16.1 INTRODUCTION

Broadly the term “Open source” software is used to refer to those categories of software/programs whose licenses do not impose much condition. Such software, generally, give users freedom to run/use the software for any purpose to study and modify the program, and to redistribute copies of either the original or modified program.

There are many categories of software that may be referred to as open source software. Following sub section is going to talk about the same.

16.1.2 Free software

Free software means the software is freely accessible and can be freely used, changed, improved, copied and distributed by all who wish to do so. And no payments are needed to be made for free software.

Free software is a matter of liberty, not price. To understand the concept, you should think of “free” as in “free speech”, not as in “free beer”. Free software is a matter of the users freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the user of the software.

1. The freedom to run the program, for any purpose.
2. The freedom to study how the program works and adapt it to your needs. Access to the source code is a precondition for this.
3. The freedom to redistribute copies so you can help your neighbor.
4. The freedom to improve the program and release your improvements to the public, so that the whole community benefits. Access to the source code is a precondition for this.

16.1.3 Open Source Software

Open Source Software, on the other hand, can be freely used but it does not have to be free of charge. Here the company constructing the business models around open source software may receive the payments concerning support, further development. What is important to know here is that in open source software, the source code is freely available to the customer.

Open source doesn't just mean access to the source code. The distribution terms of open source software must comply with the following criteria.

- 1. Free redistribution:** The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.
- 2. Source Code:** The program must include source code and must allow distribution in source code as well as complied form. Where some form of a product is not distributed with source code, there must be a well publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably,

downloading via the internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a pre processor or translator are not allowed.

3. **Derived works:** The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
4. **No discrimination against persons or groups:** The license must not discriminate against any person or group of persons.
5. **No discrimination against fields of Endeavor:** The license must not restrict anyone from making use of the program in a specific field of endeavor. For example it may not restrict the program from being used in a business or being used for genetic research.
6. **Distribution of license:** The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
7. **License must not be specific to a product:** The rights attached to the program must not depend on the programs being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the programs license, all parties to whom the program is redistributed should have the same rights as those that are guaranteed in conjunction with the original software distribution.
8. **The license must not restrict other software:** The license must not place restrictions on other software that is, distributed alone with the licensed software.
9. **License must be technology neutral:** No provision of the license may be predicated on any individual technology or style of interface.

Software which is free as well as open belongs to category FOSS (Free and Open Source Software). The terms free and open represent a differing emphasis on importance of freedom (free software) or technical progress (Open Source Software).

16.1.4 OSS and FLOSS

OSS refers to Open Source Software, which refers to software whose source code is available to customers and it can be modified and redistributed without any limitations. An OSS may come free of cost or with a payment of nominal charges that its developers may charge in the name of development, support of software.

FLOSS refers to Free Libre and Open Source Software or to Free Livre and Open Source Software. The term FLOSS is used to refer to software which is both free software as well as open source software. Here the words Libre (a Spanish word) and Livre (a Portuguese word) mean freedom.

16.1.5 GNU

GNU refers to GNU's not Unix. GNU project emphasizes on freedom and thus its logo type shows a gnu, an animal living in freedom.

16.1.6 FSF

FSF is Free Software Foundation. FSF is a non-profit organization created for the purpose of supporting free software movement. *Richard Stallman* founded FSF in 1985 to support GNU project and GNU licenses. FSF has founded many software developers to write free software. Now a day, it also works on legal and structural issues for the free software community.

16.2.1 OSI

OSI is Open Source Initiative. It is an organization dedicated to cause of promoting open source software. Bruce Perens and Eric Raymond were the founders of OSI that was founded in February 1998. OSI specifies the criteria for Open Source Software and properly defines the terms and specifications of Open Source Software.

Open Source does not just mean access to the Source code. The distribution terms of Open source software must comply with the open source definition by OSI.

16.2.2 W3C

W3C is acronym for World Wide Web Consortium. W3C is responsible for producing the software standards for World Wide Web. The W3C was created in October 1994, to lead the WWW to its full potential by developing common protocols that promote its evolution and ensure its interoperability.

16.2.3 Proprietary Software

Proprietary Software is the software that is neither open nor freely available. Its use is regulated and further distribution and modification is either forbidden or requires special permission by the supplier or vendor. Source code of Proprietary Software is normally not available.

Freeware

The term **Freeware** has no clear definition, but is generally used for software, which is available free of cost and which allows copying and further distribution, but not modification and whose source code is not available. Freeware should not be mistaken for open software or for free software.

Shareware

Shareware is software, which is made available with the right to redistribute copies, but it is stipulated that if one intends to use the software, often after a certain period of time, then a license fee should be paid.

- (i). in shareware the source code is not available
- (ii). Modifications to the software are not allowed.

16.2.4 WWW (World Wide Web)

The **WWW (World Wide Web)** is a set protocols that allows you to access any document on the net through a naming system based on URLs. WWW also specifies a way— the Hypertext Transfer Protocol (HTTP) to request and send a document over the internet. Before WWW, Internet was mainly used for obtaining textual information. But post-WWW, the Internet popularity grew tremendously because of graphic intensive nature of WWW. Therefore, we may attribute the explosion in use and popularity of Internet to WWW only.

16.2.5 Telnet

Telnet is an older Internet Utility that lets you log on to remote computer systems. Basically, a Telnet program gives you a character-based terminal window on another system. You get a login prompt on that system. If you've permitted access, you can work on that system, just as you would if you were sitting next to it. Telnet has been used by people who have logins on remote systems and want to do serious work there. Most notably, you can use Telnet to connect to thousands of catalogs at libraries around the world.

16.2.6 Web Browser

A **Web Browser** is a WWW client that navigates through the World Wide Web and displays web pages. Internet Explorer and Netscape Navigator are two most popular web browsers.

16.2.7 Web Server

Web Server is a WWW server that responds to the requests made by web browsers. Each website has a unique address called **URL (Uniform Resource Locator)**.

16.2.8 Web Page

The documents residing on web sites are called Web Pages. The web pages use HTTP.

1. Home Page: It is the top-level web page of a web site. When a web-site is opened, its home page is displayed.
2. Web Portal: It is a web site, which hosts other web sites. In other words, a web portal has hyperlink to many other web sites. By clicking upon these links, the corresponding websites can be opened.

16.3 URL and Domain Names

The Internet structure of the World Wide Web is built on a set of rules called Hypertext Transfer Protocol (HTTP) and a page description language called Hypertext Markup Language (HTML). HTTP uses internet addresses in a special format called a Uniform Resource Locator or URL, URLs look like this:

type://address/path

Where type: specifies the type of the server in which the file is located, address is the address of server, and path tells the location of file on the server. For example, in the following URL

<http://encycle.msn.com/getinfo/styles.asp>

http: specifies the type of server, encycle.msn.com is the address of server and getinfo/styles.asp is the path of the file style.asp.

Syntax Elements of URLs

URL is an address of a file on Internet. The components or syntax elements of URLs and a file's Internet address, or URL, is determined by the following:

1. The type of server or protocol
2. The name or address of the server on the Internet
3. The location of the file on the server

The intelligent browsers like Netscape Navigator or Microsoft Internet Explorer can display files in just about any format available on any of the common types of servers.

In any typical URL, the "http" identifies both the protocol and server. According to standard URL syntax, a colon (:) and two forward slashes (//) follow the protocol/server.

The next component of the address is the name of the server; in this case, server names have multiple components. Commonly a Web server's name will begin "www" for World Wide Web.

Internet Servers and What They Provide

Server Protocol Information It Provides : ftp gopher, httpmail news, File Transfer Protocol/Transfer Control Protocol/Internet Protocol (TCP/IP) Hypertext Transfer Protocol/Post Office Protocol (POP) Version 3 and Simple Mail Transfer Protocol (SMTP)/Network News Transfer Protocol (NNTP)
 Text and binary files that are organized in a hierarchical structure, much like a family tree. Text and binary files that are organized in a menu structure. Hypertext/hypermedia files Messages sent via electronic mail. Newsgroups that are organized in a hierarchical structure.

Domain Name : Domain name are used to identify one or more ip addressess. Domain names are used in URLs to identify particular web page/web pages.

Some Most Common Domains

In addition to it, a two letter abbreviation indicating the country name may be used e.g., <http://www.pue.kar.nic.in>. Here the last in suggests that it is based in India (.in) and pue.kar.nic.in is the domain name. Similarly, the URL <http://www.clearnet.nz> suggests that it is based in New Zealand (.nz).

Sl.No.	Domain ID	Affiliation	Remarks
1	com	Commercial	For commercial firms
2	edu	Education	For educational firms
3	gov	Government	For Government Organizations/bodies
4	mil	Military	For Military
5	net	Network resources	For ISPs/networks
6	org	Usually non-profit organizations	For NGOs and other no-profit
7	co	Company	For listed companies
8	biz	Business	For business
9	tv	Television	For television companies and channels

Some country abbreviations are listed below:

au	Australia
dk	Denmark
in	India
nz	New Zealand
uk	United Kingdom

16.4 Electronic Commerce

Electronic commerce is sophisticated combination of tecnologies and consumer-based services integrated to form a new paradigm in business transaction processing..

Business activities such as marketing,sales, sales promotion; sub contracts, supply; financing and insurance, commerical contracts, supply; financing and insurance commecial transactions:Ording, delivery, payment; product service and maintence;use of private and public services business-to-administrations(permissions, tax, customers, etc); banking, transnsport and logistics; public procurement(results can be seen of internet); automatic trading of digital goods and accounting.

Definition : E-commerce is the trade of goods and services with the help of telecommunication and computers.

E-commerce involves the automation of a variety of business to consumer(B2B,B2C,C2B,C2C) transaction through reliable and secure connections.

Some of the technologies and services used in e-commerce are

1. Electronic Data interchange(EDI) is the electronic interchange of business information using a standardized format. In other words, EDI is a process which allows one company to send information to another company electronically rather than with paper. Business entities which conduct business electronically are called trading partners.
2. e-mail
3. Electronic Funds transfer(EFT)
4. Electronic Benefits transfer(EBT),
5. Electronic forms(online admissions forms for college and other registrations)
6. Digital cash(DC)
7. Interoperable database access
8. Bulletin Boards(BBs)
9. Electronic Banking(EB)
10. Bar-coding-2D, Imaging, voice recognition,
- 11 security services such as firewalls, encryption, gateways etc.

How does eCommerce work?

Step 1: The merchant submits a credit card transaction to the NMAPAY Payment gateway on behalf of a customer via secure Web site connection, retail store, MOTO center or wireless device.

Step 2: NMAPAY receives the secure transaction information and passes it via a secure connection to the Merchant Bank's Processor.

Step 3: National Merchants Association submits the transaction to the Credit Card Network (a system of financial entities that communicate to manage the processing, clearing, and settlement of credit card transactions).

Step 4: The Credit Card Network routes the transaction to the Customer's Credit Card Issuing Bank.

Step 5: The Customer's Credit Card Issuing Bank approves or declines the transaction based on the customer's available funds and passes the transaction results back to the Credit Card Network.

Step 6: The Credit Card Network relays the transaction results to National Merchants Association

Step 7: National Merchants Association relays the transaction results through NMAPAY(website)

Step 8: NMAPAY stores the transaction results and sends them to the customer and/or the merchant. This step completes the authorization process - all in about three seconds or less!

Step 9: National Merchants Association sends the appropriate funds for the transaction to the Credit Card Network, which passes the funds to the Merchant's Bank. The bank then deposits the funds into the merchant's bank account. This step is known as the settlement process and typically the transaction funds are deposited into your primary bank account within 24 to 48 hours.



16.4.1 Types of e-commerce applications:

1. Business-to-Business(B2B)

2. Business-to Consumer(B2C)

3. Consumer-to-Business(C2B)

4. Consumer-to-Consumer(C2C)

1. Business-to-Business(B2B): The exchange of services, information and/or products from one business to another Business partners.Ex Ebay.com

2. Business-to-Consumer(B2C): The exchange of services, information and/or product from a business to a consumer.

3. Consumer-toBusiness(C2B): Customer directly contact with business vendors by posting their project work with set budge online so that the needy companies review it and contact the customer directly with bid. The consumer reviews all the bids and selects the company for further processing. Ex. guru.com, freelancer.com.

4. Consumer-to-consumer: Electronic commerce is an internet facilitated form of business(commerce).

16.4.2 Advantages of electronic commerce application and implementation:

- 1.Easier entry into new markets, especially geographically remote markets, for companies of all sizes and locations.
2. Creates a new markets through the ability to easily and economical rate potential for customers.
3. Global participation
4. Optimization of resource.
5. Reduce time to complete business transaction, particularly from delivery to payment.
6. Buyer makes a buying decision, creates the purchase order but does not print it.
- 7.Improved market intelligence and strategic planning.
8. EDI software creates an electronic version of the purchase order and transmits it automatically to the supplier.
9. Supplier's order entry system receives the purchase order and updates the system immediately on receipt.
- 10.Supplier's order entry system creates an acknowledgment an transmits it back to confirm receipt.

16.5 IPR -issues Intellectual Property Rights(IPR) in India

Does the nature of the technology require us to change the legal understanding or status of copyright as it stands now? What rights should be associated with Web content? How should the rights be expressed, and should the expression of the rights be used for notification, enforcement, or payment negotiation? We expect the answer to these questions does not lie solely in technology nor policy, but the rational combination of both.

Copyright has been the focus of protecting intellectual property on the Internet. As such, there have been both technological (IPR/encryption wrappers) and legislative efforts to continue incentives for authors to create useful works. Recent initiatives have been at the international level include at the OECD, and a conference (Dec. 96) hosted by the World Intellectual Property Organization_(WIPO).

IPR-related issues in India like patents, trademarks, copyrights, designs and geographical indications are governed by the Patents Act 1970 and Patent Rules 2003, Trademarks Act 1999 and the Trademarks Rules 2002, Indian Copyrights Act, 1957, Design Act 2000 and Rules 2001, and The Geographical

Indications of Goods (Registration & Protection) Act, 1999 and The Geographical Indications of Goods (Registration & Protection) Rules 2002, respectively.

IPR plays a key role in almost every sector and has become a crucial factor for investment decisions by many companies. All the above Acts and regulations are at par with international standards. India is now TRIPS-compliant. This is an international agreement administered by the World Trade Organization (WTO), which sets down minimum standards for many forms of intellectual property (IP) regulations as applied to the nationals of other WTO Members. The very well-balanced IPR regime in India acts as an incentive for foreign players to protect their Intellectual Property in India.

This can be established by the very fact that approximately 80% of patent filings in India are from the MNCs.

While the IPR regime in India consists of robust IP laws, it lacks effective enforcement, for which “least priority given to adjudication of IP matters” is often quoted as a reason. The key challenge is to sensitize the enforcement officials and the Judiciary to take up IP matters, at par with other economic offences, by bringing them under their policy radar. Further, it is imperative that there be established a ‘Think Tank’ or a group, which can bring the varied sets of stakeholders on to a common platform, leading to extensive/exhaustive and an all inclusive debate/discussion, facilitating well-informed policy decisions in accordance with India’s socio-economic-political needs. The challenges also lie in having an IP fund, which can be utilized for further developing the IP culture in the country. There is also the need to have a National IP Policy for India, which will help in working towards realizing the vision of India in the realm of IP. This will facilitate the creation of a strong socio-economic foundation and deep international trust.

FICCI's efforts emphasize the enhancing of the working of the Indian Patent Office, thereby, bringing greater transparency in its working, and facilitating the Government in developing a policy for India.

The IPR division tries to provide proactive business solutions through research, interactions at the highest political level while facilitating global networking. Further, since the IPR provides exclusive rights over assets, it is a major challenge for the country to balance the interests of the innovators and the interests of the society at large.

In today's highly competitive global economy, IPRs are giving companies the cutting edge and increasing their competitiveness. With recent changes in IP laws, various IP related issues have sprung up, which are highly complex in nature. FICCI envisions itself as the ‘thought’ leader in the field of IPR. FICCI

also views itself as being capable enough to assist the government and the industry captains in all IP related matters.

Showcasing its unparalleled capabilities in this sphere, FICCI's IPR division organizes the World IP Day on April 26th every year. In fact, on World IP Day 2010, FICCI prepared and submitted a discussion paper on the National IP Policy to the Government of India. In 2011 as well, FICCI submitted a brief report to the Government of India, with a view to safeguarding India's interests in the fields of traditional knowledge and traditional cultural expressions, at the International Governmental Committee meeting at WIPO.

Review questions

One mark questions:

1. What is open source software?
2. What are free software?
3. What is OSS and FLOSS
4. What is Proprietary software?
5. What is Freeware?
6. What are Browsers
7. What are URL?
8. What are Telnet?
9. What is domain name?
10. Define e-commerce?
11. Expand IPR.

Two marks questions:

1. List the OSS and Floss.
2. What is FSF.
3. What are OSI and W3C?
4. What is URL and HTTP?
5. Name the different protocols used?
6. List the services of e-commerce?
7. Write a note on WIPO.

Three marks questions:

1. What is Open source?
2. Write the advantages of www?
3. What is Telnet?
4. Write the web servers?
5. Write a note on open source?
6. Explain free software?
7. Explain URLs?
8. How ecommerce works?
9. Explain types of e-commerce?
10. Explain the IPR in india.

CHAPTER 17

Web designing

OBJECTIVES

- Knowing a web page and web site
- Html structures.
- How html programs are used in creating web sites
- To understand the concept of hosting and maintaining websites



17.1 Introduction

HTML is the “mother tongue” of your browser.

To make a long story short, HTML was invented in 1990 by a scientist called Tim Berners-Lee. The purpose was to make it easier for scientists at different universities to gain access to each other’s research documents. The project became a bigger success than Tim Berners-Lee had ever imagined. By inventing HTML he laid the foundation for the web as we know it today.

HTML is a language, which makes it possible to present information (e.g. scientific research) on the Internet. What you see when you view a page on the Internet is your browser’s interpretation of HTML. To see the HTML code of a page on the Internet, simply click “View” in the top menu of your browser and choose “Source”.

For the untrained eye, HTML code looks complicated but this tutorial will help you make sense of it all.

What can I use HTML for?

If you want to make websites, there is no way around HTML. Even if you’re using a program to create websites, such as Dreamweaver, a basic knowledge of HTML can make life a lot simpler and your website a lot better. The good news is that HTML is easy to learn and use. In just two lessons from now you will have learned how to make your first website.

HTML is used to make websites. It is as simple as that!

Okay, but what does H-T-M-L stand for?

HTML is an abbreviation of “HyperText Mark-up Language” - which is already more than you need to know at this stage. However, for the sake of good order, let us explain in greater detail.

- Hyper is the opposite of linear. In the good old days - when a mouse was something the cat chased - computer programs ran linearly: when the program had executed one action it went to the next line and after that, the next line and so on. But HTML is different - you can go wherever you want and whenever you want. For example, it is not necessary to visit MSN.com before you visit HTML.net.
- Text is self-explanatory.
- Mark-up is what you do with the text. You are marking up the text the same way you do in a text editing program with headings, bullets and bold text and so on.

- Language is what HTML is. It uses many English words.

17.1.1 HTML Structure : An HTML document has a definite structure that must be specified to the browser. The HTML's beginning and end must be defined, as well as the document's HEAD (which contains information for the browser that does not appear in the browser's main window) and its BODY (which contains the text that will appear in the browser's main window). The use and order of tags that define the HTML structure are described below.

The body of the document contains all that can be seen when the user loads the page.

<html>	Marks the beginning of your HTML
<head>	Begins the heading section of an HTML document
<title> ... </title>	Gives an HTML document a title that appears on the browser menu bar, also will appear on search engines or bookmarks referencing your site (must appear between the <HEAD> ... </HEAD> tags; should be straight text, no tags)
</head>	Defines the end of the heading
<body>	Defines the body of an HTML document (text contained within the <BODY> ... </BODY> tags appears in the main browser window). Can be used with "BGCOLOR", "TEXT", "LINK", and "VLINK" attributes
</body>	
</html>	Defines the end of your HTML document

The html can be created using any text editor like notepad.

The file msut be saved using the extension.html

In order to execute the html program use any web browser.

WEb page with basic features



figure 17.1 www.pue.kar.nic.in

The official website of the pue consists of the basic layout of web page like image, web page links, web pages with headings, Bulletin Board, URL, with forms and tables. This is one of the example for web page designing.

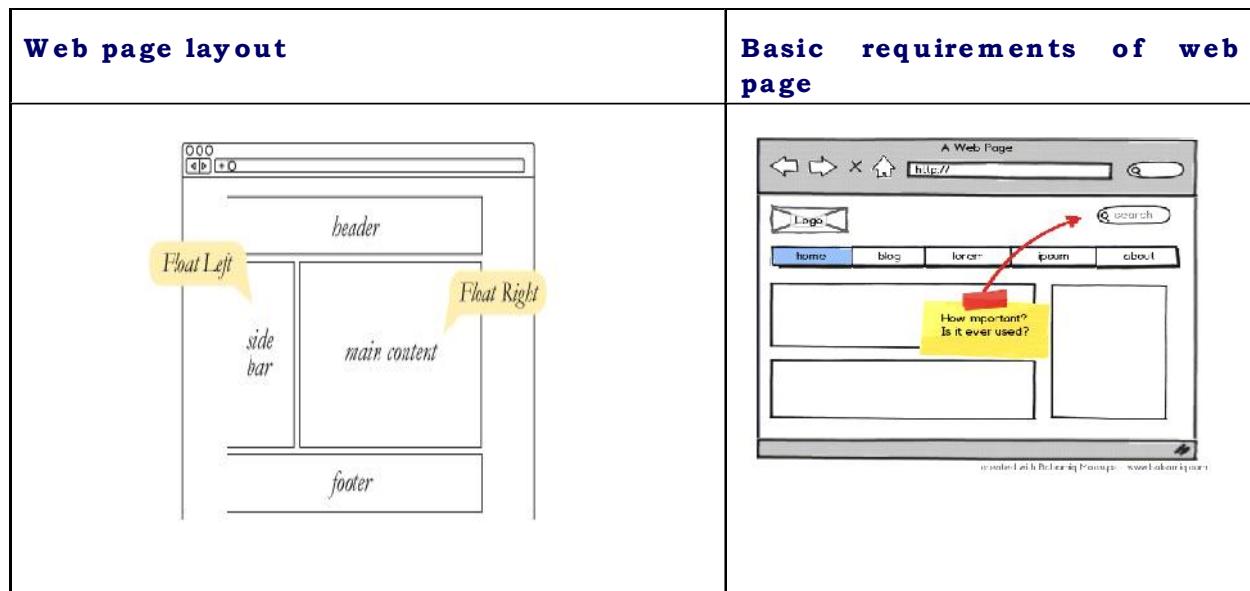


figure 17.2 Layout of HTML

Most of web pages contain the basic layouts such as address box to enter the domain name with forward and back navigating pages. Header of the web page, with or without sub heading, footer with the licence value, images etc. This chapter will highlight the commands used in advanced HTML with examples and samples are given.

17.2.1 Advanced HTML tags/commands

Text	Links	Images	Backgrounds	Others
Formatting	To local pages	Inserting images (GIF and jpg)	Colors	Tables
Resizing	To pages at other sites	Adding a link to an image	Images	Frames
Layout			Fixed Image	Forms
Listing	To bookmarks			

17.2.2 TEXT Formatting

<p>This text is bold</p> <p><i>This text is italic</i></p> <p>This is computer output</p> <p>This is _{subscript} and ^{superscript}</p>	<p>HTML uses tags like and <i><i></i> for formatting output, like bold or <i>italic</i> text.</p> <p>These HTML tags are called formatting tags there is a difference in the meaning of these tags:</p> <p> or <i> defines bold or italic text only.</p>
<p> or means that you want the text to be rendered in a way that the user understands as "important". Today, all major browsers render strong as bold and em as italics. However, if a browser one day wants to make a text highlighted with the strong feature, it might be cursive for example and not bold!</p>	

17.2.3 Resizing TEXT

These are the tags for changing the font size.

<big>text</big>	increase the size by one
<small>text</small>	decrease the size by one
<h1>text</h1>	writes text in biggest heading
<h6>text</h6>	writes text in smallest heading
text	writes text in smallest fontsize. (8 pt)
text	writes text in biggest fontsize (36 pt)

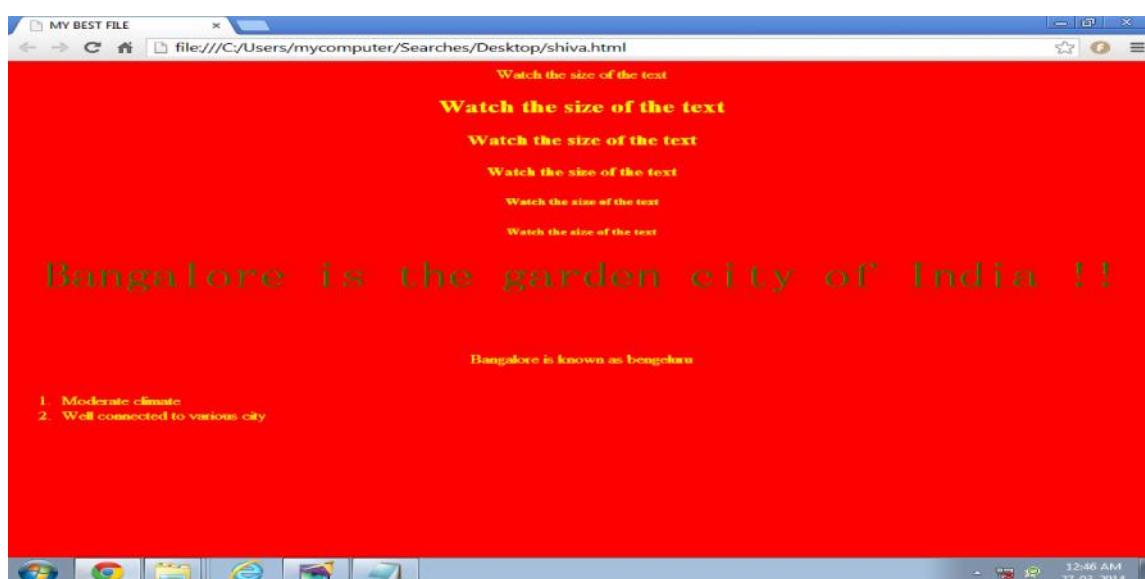
The **<small>** and **<big>** tags are special in that they can be repeated. If you want to increase the font size with a factor two, then you could do it like this:

<big><big> whatever</big></big>

17.2.4 Example for resizing text

```
<HTML>
    <HEAD>
        <TITLE>      MY BEST FILE      </TITLE>
    </HEAD>
<BODY BGCOLOR="RED" TEXT="YELLOW"
    <H1><CENTER> Watch the size of the text </CENTER></h1>
    <H2><CENTER> Watch the size of the text </CENTER></h2>
    <H3><CENTER> Watch the size of the text </CENTER></h3>
    <H4><CENTER> Watch the size of the text </CENTER></h4>
    <H5><CENTER> Watch the size of the text </CENTER></h5>
    <H6><CENTER> Watch the size of the text </CENTER></h6>
<P>
    <FONT FACE="SIMSUN" SIZE="12" COLOR="GREEN">
        <CENTER> Bangalore is the garden city of India !!  </CENTER>
    </FONT>  <BR>
</P>

<P><FONTFACE="SIMSUN" SIZE="12">
    <CENTER>  Bangalore is known as bengaluru
</FONT>
</CENTER>  <BR>
</P>
    <OL>
        <LI> Moderate climate
        <LI> Well connected to various city
    </OL>
</BODY>
</HTML>
```



17.2.5 TEXT layout

These tags will let you control the layout.

HTML	EXPLANATION	RESULT	HTML
<code><p>text</p></code>	Adds a paragraph break after the text. (2 linebreaks).	Hello world- a linebreak does not insert a linebreak in HTML	Hello world - a linebreak does not insert a linebreak in HTML
<code><p align="left">text</p></code>	Left justify text in paragraph.	you will need to insert	<code><p>you will need</p></code>
<code><p align="center">text</p></code>	Center text in paragraph.	special tags	<code><p align="left">special tags</p></code>
<code><p align="right">text</p></code>	Right justify text in paragraph.	that will insert linebreaks where you want it!	that will insert <code>
</code> linebreaks where <code>
</code> you want it! <code>
</code>
<code>text
</code>	Adds a single linebreak where the tag is.	Another method is to write a	Another method is of course to write a sentence, that is long enough to force a linebreak. <code>
</code>
<code><nobr>text</nobr></code>	Turns off automatic linebreaks - even if text is wider than the window.	This option can however be turned offwith the nobr-	<code><nobr>This option can however be turned off<wbr>with the nobr tag,<wbr>unless a wbr is used to force it!</nobr></code>
<code>text<wbr></code>	Allows the browser to insert a linebreak at exactly this point - even if the text is within <code><nobr></code> tags.	tag.unless a wbr is used to force it!	

HTML	EXPLANATION	RESULT	HTML
<center>text</center>	Center text.	force it!	<center>You can center</center>
<div align="center">text</div>	Center text.	You can also center	And turn the center off
<div align="left">text</div>	Left justify text.	And turn the center off	<div align="center">And on!</div>
<div align="right">text</div>	Right justify text.	And on!	<div align="left">Go left!</div>
		Go left!	<div align="right">Go Right!</div>
		Go right!	

Note in particular the difference between the `<p>` and the `<div>` tags.

The `<div>` tag allows you to justify content without being forced to add a double linebreak.

Also, note that these alignment tags are not limited to text. They work on text, images, applets or whatever it is that you insert on the page.

17.2.6 Number Listing

This page shows how to make different kinds of numbered lists.

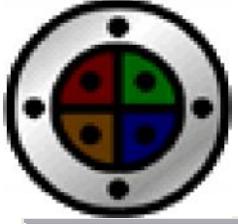
You have the following number options:

- Plain numbers
- Capital Letters
- Small Letters
- Capital Roman Numbers
- Small Roman Numbers

HTML-CODE	EXPLANATION / EXAMPLE
 text text text 	Makes a numbered list using the default number type: 1. text 2. text 3. text
<ol start="5">	Starts a numbered list, first # being 5. 5. This is one line 6. This is another line 7. And this is the final line
<ol type="A">	Starts a numbered list, using capital letters. A. This is one line B. This is another line C. And this is the final line
<ol type="a">	Starts a numbered list, using small letters. a. This is one line b. This is another line c. And this is the final line
<ol type="I">	Starts a numbered list, using capital roman numbers. I. This is one line II. This is another line III. And this is the final line
<ol type="i">	Starts a numbered list, using small roman numbers. i. This is one line ii. This is another line iii. And this is the final line
<ol type="1"> <ol type="I" start="7">	Starts a numbered list, using normal numbers. 1. This is one line 2. This is another line 3. And this is the final line
<ol type="I" start="7">	An example of how type and start can be combined. VII. This is one line VIII. This is another line IX. And this is the final line

17.2.7 Links		
<p>Links The tags used to produce links are the <a> and .</p> <p>The <a> tells where the link should start and the indicates where the link ends.</p> <p>Everything between these two will work as a link.</p> <p>The target of the link is added to the <a> tag using the href="http://www.whateverpage.com" setting.</p> <p>The example below shows how to make the word here work as a link to yahoo.</p> <div style="border: 1px solid black; padding: 10px;"> Click < a href="http://www.yahoo.co m ">here to go to yahoo. </div> <p>You simply:</p> <ul style="list-style-type: none"> Specify the target in the . Then add the text that should work as a link. Finally add an tag to indicate where the link ends. <p>You can name bookmarks anything you like. Bookmarks are very useful on pages which are very long as they can be used to quickly go to another part of the page.</p>	<p>Page link using html tags Linking to anchors is very similar to normal links. Normal links always point to the top of a page. Anchors point to a place within a page.</p> <p>A # in front of a link location specifies that the link is pointing to an anchor on a page. (Anchor meaning a specific place in the middle of your page).</p> <p>To link to an anchor you need to:</p> <p>Create a link pointing to the anchor Create the anchor itself. An anchor is created using the <a> tag. If you want to create an anchor called chapter4, you simply add this line where you want the anchor to be:</p> <div style="border: 1px solid black; padding: 10px;"> < a name="chapter4"> </div> <p>After doing this, you can make a link using the normal here syntax:</p> <p>Note: When linking to an anchor on a # in front of the anchor. When you link to an anchor on enter blabla</p> <p>When you link to anchors on ex syntax:</p> <div style="border: 1px solid black; padding: 10px;"> < a href="http://www.yahoo.com#Y a> </div>	<p>BOOKMARKS Bookmarks</p> <p>Bookmarks on a page are very easy to make as they also use the <a> tag. Instead of changing the href variable you use the name variable. For example:</p> <pre>The First Text In The Page</pre> <p>Will create a bookmark called top in the text which the tag surrounds. An image can also be contained in this tag. You can then link to this using a standard hyperlink:</p> <pre>Back To Top</pre>

17.2.8 Inserting Images

Inserting Images	Resizing the images	Adding border to the images	Linking Images in html
<p>The tag used to insert an image is called img.</p> <p>Below you see an image called "rainbow.gif".</p>  <p>Here is the HTML code used to insert the image on this webpage:</p> <pre></pre> <p>If the image is stored in the same folder as the HTML page, you can leave out the domain reference (http://www.echoecho.com/) and simply insert the image with this code:</p> <pre></pre> <p>On the following pages we will discuss different ways to control.</p>	<p>You can change the size of an image using the width and height attributes.</p> <p>In general, it is not advisable to reduce image size using these settings, since the image will be transferred over the internet in its original size no matter what reduction is set for it. This will slow the loading of your webpage. This means, that if you have an image that is bigger in size than you want it to be on your page, you should reduce the size in a graphics program, rather than reducing the size on the webpage using the width and height attributes. On the contrary, sometimes, it can be wise to enlarge images using this technique.</p> <p>Below are two presentations of the exact same image - with different settings for width and height.</p>  <pre></pre>  <pre></pre>	<p>You can add a border to the image using the border setting shown in the example below:</p> <p>Note: Netscape browsers will only show the border if the image is a link.</p>  <pre></pre> <p>Adding a border to your image might help the visitor recognize that the image is a link. However, the net is filled with images that work as links and have no borders indicating it - so the average visitor is used to letting the mouse run over images to see if they are links.</p> <p>Still - if you have an image that is often mistaken you might consider adding a border to it - although you should probably consider changing the image entirely - since if it does not indicate by itself that it is a link then it isn't serving its purpose.</p>	<p>If you want to make an image work as a link, the method is exactly the same as with texts.</p> <p>You simply place the <code><a href></code> and the <code></code> tags on each side of the image.</p> <p>Below is the HTML code used to make the image work as a link to a page called myfile.htm:</p>  <pre></pre> <p>If you haven't entered a border setting you will see a small border around the image after turning it into a link. To turn off this border, simply add <code>border="0"</code> to the <code></code> tag:</p> <pre></pre> <p>Images that work as links can show a popup text when you place the mouse over it. This is done with the alt property in the <code></code> tag.</p> <p>For example:</p> <pre></pre> 

17.2.9 Background

If you want to add a background image instead of a plain color there are some considerations you should make before doing so:

- Is the background image discrete enough to not take away the focus from what's written on it?
- Will the background image work with the text colors and link colors I set up for the page?
- Will the background image work with the other images I want to put on the page?
- How long will the page take to load my background image? Is it simply too big?
- Will the background image work when it is copied to fill the entire page? In all screen resolutions?

After answering these questions, if you still want to add the background image you will need to specify in the `<body>` tag which image should be used for the background.

```
<body background="drkrainbow.gif">
```

Note:

If the image you're using is smaller than the screen, the image will be replicated until it fills the entire screen.

If, say you wanted a striped background for your page, you wouldn't have to make a huge image for it. Basically you could just make an image that is two pixels high and one pixel wide. When inserted on the page the two dots will be copied to fill the page - thus making what looks like a full screen striped image.

When you choose to use a background image for the page it is always a good idea to specify a background color as well.

```
<body background="drkrainbow.gif" bgcolour="#333333">
```

The reason is that until the background image is loaded, the background color will be shown.

If there is too much difference between the background color and the background image, it will look disturbing once the browser shifts from the background color to the image.

Therefore it is a good idea to specify a background color that matches the colors of the image as close as possible.

You may have noticed that background images scroll with the page when you use the scroll bar.

17.2.10 Background color and fixed images

The background image will scroll when the user scrolls down the page, unless you have set it to be fixed:

```
<body  
background="drkrainbow.gif" bgproperties="fixed">
```

By adding the `bgproperties="fixed"` you force the browser to let the background be fixed even if the user is scrolling down the page.

Note: Fixed backgrounds are only supported by MSIE and do not work in Netscape browsers - instead they simply act as normal backgrounds.

As mentioned earlier in this section a limited use of colors can add more power to the few colors that are used.

The most important tool for adding colors to certain areas of the page rather than the entire background is tables.

17.2.11 Tables

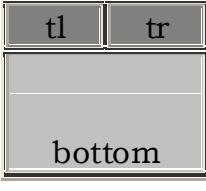
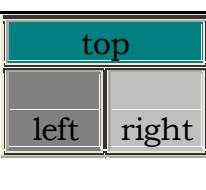
. The following properties can be added to the `<table>` tag:

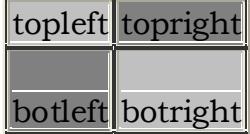
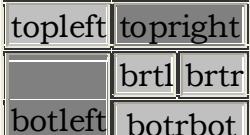
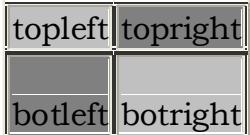
The following properties can be added to the <table> tag:		Rows/cell These settings can be added to both <tr> and <td> tags.
Property	Description	PROPERTY
align=left center right	left align table center table right align table	align=left right center
background=file name	image inserted behind the table	background=filename
bgcolor=#rrggbb	background color	bgcolor=#rrggbb
border=n	border thickness	bordercolor=#rrggbb
bordercolor=#rrggb	border color	bordercolordark=#rrggbb
bordercolordark=#rrggbb	border shadow	bordercolordark=#rrggbb
cellpadding=n	distance between cell and content	valign=top
cellspacing=n	space between cells	valign=middle
nowrap	protects against linebreaks, even though the content might be wider than the browser window.	valign=bottom
frame=void, above, below, lhs, rhs, hsides, vsides, box	removes all outer borders shows border on top of table shows border on bottom of table shows border on left side of table shows border on right side of table shows border on both horizontal sides shows border on both vertical sides shows border on all sides of table	width=n n%
valign=top	aligns content to	

PROPERTY	DESCRIPTION
colspan=n	number of columns a cell should span protects against linebreaks, even though the content of a cell might be wider than the browser window
nowrap	
rowspan=n	number of rows a cell should span

These settings are only valid for <td> tags. Note: Table properties are set for the entire table. If certain properties are set for single cells, they will have higher priority than the settings for the table as a whole. Note: Settings for columns(<td> tag) have higher priority than settings for rows(<tr> tag). Settings for cells (<tr> or <td> tags) have higher priority than settings for the table as a whole(<table> tag).

17.2.12 Frames

Frames	On this page you can see examples of different framesets.
	<pre><frameset rows="16%,84%"> <frame src="top.htm" name="top"> <frame src="bottom.htm" name="bottom"> </frameset></pre>
	<pre><frameset rows="16%,84%"> <frameset cols="50%,50%"> <frame src="t1.htm" name="t1"> <frame src="tr.htm" name="tr"> </frameset> <frame src="bottom.htm" name="bottom"> </frameset></pre>
	<pre><frameset rows="16%,84%"> <frame src="top.htm" name="top"> <frameset cols="50%,50%"> <frame src="left.htm" name="left"> <frame src="right.htm" name="right"> </frameset> </frameset></pre>

	<pre><frameset rows="50%,50%" cols="50%,50%"> <frame src="topleft.htm" name="topleft"> <frame src="topright.htm" name="topright"> <frame src="botleft.htm" name="botleft"> <frame src="botright.htm" name="botright"> </frameset></pre>
	<pre><frameset rows="50%,50%" cols="50%,50%"> <frame src="topleft.htm" name="topleft"> <frame src="topright.htm" name="topright"> <frame src="botleft.htm" name="botleft"> <frameset rows="50%,50%"> <frameset cols="50%,50%"> <frame src="brtl.htm" name="brtl"> <frame src="brtr.htm" name="brtr"> </frameset> <frame src="botright.htm" name="botright"> </frameset> </frameset></pre>
	<pre><frameset rows="50%,*" cols="320,*"> <frame src="topleft.htm" name="topleft"> <frame src="topright.htm" name="topright"> <frame src="botleft.htm" name="botleft"> <frame src="botright.htm" name="botright"> </frameset></pre>

17.2.13 Forms

These fields can be added to your forms:

- Ø Text field
- Ø Password field
- Ø Hidden field
- Ø Text area
- Ø Check box
- Ø Radio button
- Ø Drop-down menu
- Ø Submit button
- Ø Reset button
- Ø Image button

You can click on the field type to get a detailed explanation.

Finally, if you want to learn how to validate inputs to form fields (valid email address etc.)

Text fields are one line areas that allow the user to input text.

17.2.14 SETTINGS:

Below is a listing of valid settings for text fields:

HTML	EXPLANATION	EXAMPLE
text	One line text field	
size=	Characters shown.	
maxlength=	Max characters allowed.	
name=	Name of the field.	
value=	Initial value in the field.	
align=	Alignment of the field.	
tabindex=	Tab order of the field.	

The size option defines the width of the field. That is how many visible characters it can contain.

The maxlength option defines the maximum length of the field. That is how many characters can be entered in the field.

If you do not specify a maxlength, the visitor can easily enter more characters than are visible in the field at one time.

The name setting adds an internal name to the field so the program that handles the form can identify the fields.

The value setting defines what will appear in the box as the default value.

The align setting defines how the field is aligned.

Valid entries are: TOP, MIDDLE, BOTTOM, RIGHT, LEFT, TEXTTOP, BASELINE, ABSMIDDLE, ABSBOTTOM. The alignments are explained in the image section. You can learn about the different alignments .

The tabindex setting defines in which order the different fields should be activated when the visitor clicks the tab key.

AN EXAMPLE:

HTML	EXPLANATION	EXAMPLE
<pre><html> <head> <title>My Page</ title> </head> <body> <form name="myform" action="http://www.mydom ain.com/myformhandler.cgi " method="POST"> <div align="center">

 <input type="text" size="25" value="Enter your name here!">

 </div> </form> </body> </html></pre>	<p>textarea rows= cols= name=</p> <p>wrap= off virtual physical</p> <p>text size= maxlength= name= value=</p> <p>password size= maxlength= name= value=</p> <p>checkbox name= value=</p> <p>radio name= value=</p> <p>select name= size= multiple=</p> <p>option selected value=</p> <p>hidden name= value=</p> <p>reset name= value=</p> <p>submit name= value=</p> <p>image name=</p>	<p>Text area - several lines Rows in the field. Columns in the field. Name of the field.</p> <p>Control linebreaks. Turns off linebreaks. Shows linebreaks, but sends text as entered. Inserts linebreaks when needed and even sends it.</p> <p>One line text field Characters shown. Max characters allowed. Name of the field. Initial value in the field.</p> <p>Password field. Characters shown. Characters allowed to enter. Name of the field. Initial value in the field.</p> <p>Choose one or more options Name of the field. Initial value in the field.</p> <p>Choose only one option Name of the field. Initial value in the field.</p> <p>Drop-down menu Name of the field. Number of items in list. Allow multiple choice if yes.</p> <p>Individual items in the menu. Make an item default. Value to send if selected.</p> <p>Does not show on the form. Name of the field. Value to send.</p> <p>Button to reset all fields Name of the button. Text shown on the button.</p> <p>Button to submit the form Name of the button. Text shown on the button.</p> <p>Image behaving as button Name of the image.</p>
output	Enter your name here!	

create an Html program using table

```

<!DOCTYPE html>
<html>
    <head>
        <style>
            td, th, table
            {
                border:1px solid black;
            }
        </style>
    </head>
    <body>
        <table style="width:300px">
            <tr>
                <th>First Name</th>
                <th>Last Name</th>
                <th>Marks in HTML</th>
            </tr>
            <tr>
                <td>Santhosh</td>
                <td>Rajendran</td>
                <td>50</td>
            </tr>
            <tr>
                <td>Dheeraj</td>
                <td>Naik</td>
                <td>94</td>
            </tr>
            <tr>
                <td>John</td>
                <td>Matthew</td>
                <td>80</td>
            </tr>
        </table>
    </body>
</html>

```



The screenshot shows a Google Chrome window displaying a table. The table has three columns: "First Name", "Last Name", and "Marks in HTML". The data is as follows:

First Name	Last Name	Marks in HTML
Santhosh	Rajendran	50
Dheeraj	Naik	94
John	Matthew	80

17.3.1 Web Hosting

Web Hosting is a means of hosting web-server application on a computer system through which electronic content on the Internet is readily available to any web-browser client.

Various types of web hosting services are available.

1. Free Hosting
2. Virtual or Shared Hosting
3. Dedicated Hosting
4. Collocation Hosting

1. Free Hosting: This type of hosting is available with many prominent sites that offer to host some web pages for no cost. Free is for fun. If you want to experiment with a site or put up a small, personal site for fun of it, a free package will suffice.

2. Virtual or Shared Hosting: This type of hosting is provided under one's own domain name, www.yuorname.com. With a hosting plan with a web hosting company, one can present oneself as a fully independent identity to his/her web audience.

Virtual Hosting is where one's web site domain is hosted on the web server of hosting company along with other web sites. One can access and update to the site and its files are carefully secured. Through a log on ID and password, one has 24 hour access to maintain one's site.

3. Dedicated Hosting: In this type of hosting, the company wishing to go online, rents an entire web server from the hosting company. This is suitable for companies hosting larger websites, maintaining others' sites or managing a big online mall etc. Dedicated is for large, high-traffic sites, or for those with special needs such as e-commerce or security. They are also good for those folks for whom money is no object.

4. Co-location Hosting: For those who do not fit the dedicated- server mold, hosting companies offer a similar, but less restrictive hosting, known as co-location hosting. In this type of hosting, the company actually owns the server on which its site is hosted. That is, the company owning the site rather than the web hosting company is responsible for all server administration. The web hosting company is only responsible for providing rack-space and the physical needs. This generally includes a high speed connection to the internet, a regulated power supply and a limited amount of hands on technical support, such as data backup or hardware upgrades.

Web 2.0

The arrival of Web 2.0 has added many new features to the web applications; it has revolutionized the information sharing, user-oriented design, interoperability on the internet. This has provided information sharing in a way that was never dreamed about few years ago.

The Internet based tools like RSS, social book marking, press release. Online marketing, blog's, forums etc made an everlasting impression on people's lives as it has crossed the hurdle of socio-economic barriers.

17.3.2 Domain Registration

Domain hosting services ensure optimal performance of your website irrespective of what platform it is built on. We support various programming languages such as PHP v5, Perl, Python and CGI, and we offer affordable web hosting services for personal websites, small business websites, as well as large enterprise portals.

Web hosting allows for users to have another company store and maintain your web site for you or your company. A web hosting company may or may not be needed depending upon what is available through your Internet Service Provider. Check with your Internet Service provider to see if they offer a comparable solution to other Web Hosting companies.

When setting up with a web hosting company, we recommend that you verify the below information with them before setting the page up.

- Domain Registration
- E-Mail forwarding
- Site Statistics
- Business Account
- Bandwidth Limitations
- Front Page Extensions
- CGI, Perl, and PHP Scripts

Figure 17.3 Domain rates in rupees

Setting up a domain
Users who want their own unique domain or URL are required to know these details

1. Determining name
2. Think about the name
3. Getting an ISP and Web host
4. Domain Name Server (DNS)
5. Register
6. Why should I setup a domain name
7. What is a domain name alias or domain alias?

17.4.1 Uploading html files: Some of the steps to be followed while uploading a HTML web page using uploading software

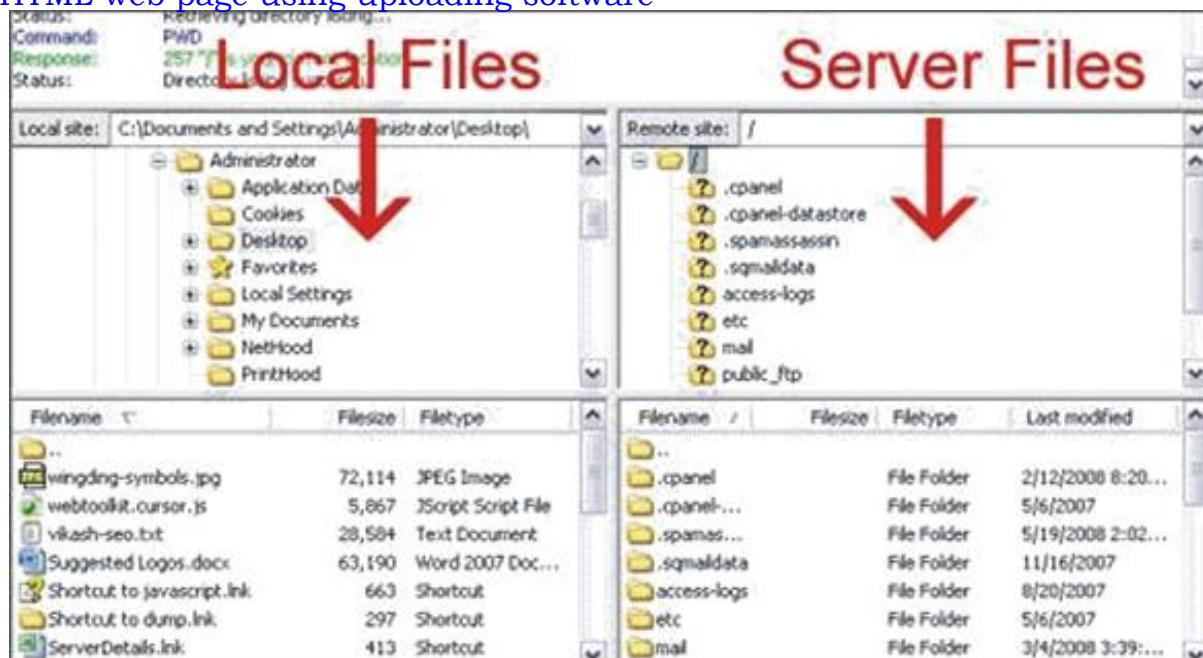
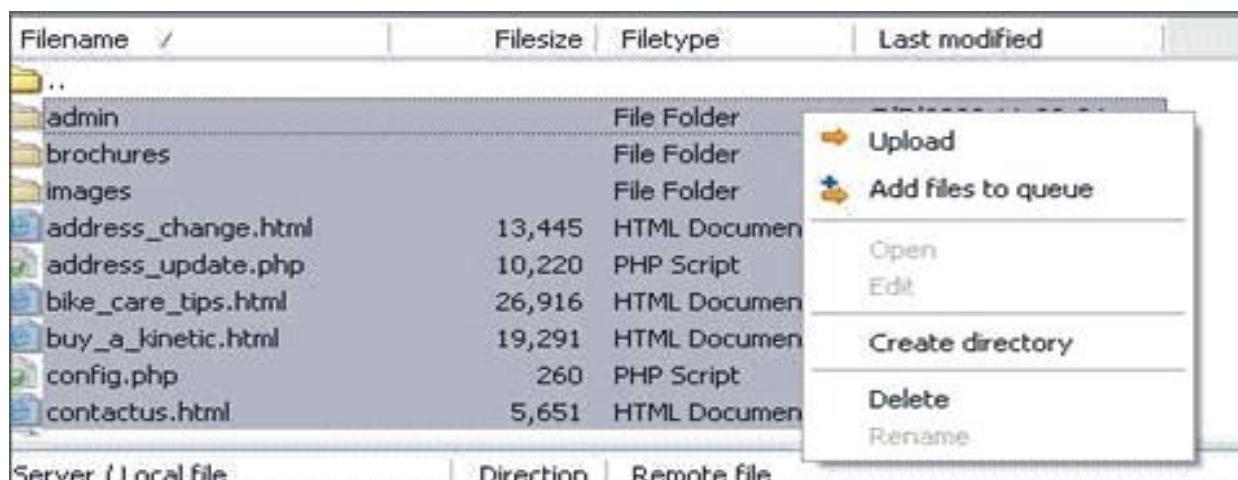


figure 17.4 showing the loacl and server files.



1. Open the FileZilla application and from the top menu, select File and click on Site Manager (ctrl + S).
2. In the Site Manager window, click on the New Site button. Enter the name of your site, like Signature Solutions.
3. Click in the text field for Host and your Host IP, like ftp.signature.co.in
4. Select Normal from the drop down for Logon type. (You can leave the field for Port empty and Server type as it is)
5. In the User text field, enter the FTP Username.
6. In the Password text field, enter your FTP Password.

7. Now click on Connect button to establish a connection with your hosting server. If the connection is successful, you should see the status message in the status window as Directory listing successful.

If the connection is not successful, the status window will show an error message as Could not connect to server. Check if you have entered the details correctly. If you are still unable to connect, contact your web hosting service provider for assistance.

8. After successfully connecting to the server, FileZilla will list all the files on your computer in the left window and the files on your server in the right window.

9. To upload files, browse to the destination directory on the server in the right window. Then, browse to the source directory in your computer in left window and select the directories and files that you want to upload to the server. Now right click in the selected area and select Upload.

10. You should be able to see the upload progress in the bottom window.

11. To download files, select the destination directory from the left window and the source directories & files from the right window, right-click on the selection and select Download.

12. Files related to your website is kept under public_html, www or documents directory.

13. Always remember to Disconnect after you finish the upload or download. To disconnect, select Server from the top menu and click on Disconnect (ctrl + D).

14. To connect again, go to Site Manager (ctrl + S), select the account and click on the Connect button.

17.5.1 XML

XML is a eXtended Markup Language for documents containing structured information. Structured information contains both content (words, pictures, etc) and some indication of what role that content plays. Almost all documents have some structure.

XML is a text-based markup language that is fast becoming the standard for data interchange on the web. As with HTML, you identify data using tags (identifiers enclosed in angle brackets: <...>). Collectively, the tags are known as markup. But unlike HTML, XML tags identify the data rather than specify how to display it. Whereas an HTML tag says something like, “Display this data in bold font” (...), an XML tag acts like a field name in your program. It puts a label on a piece of data that identifies it (for example, <message>...</message>).

One big difference between XML and HTML is that an XML document is always constrained to be well formed. There are several rules that determine when a document is well formed, but one of the most important is that every tag has a

closing tag. So, in XML, the </to> tag is not optional. The <to> element is never terminated by any tag other than </to>

Note: Another important aspect of a well-formed document is that all tags are completely nested. So you can have <message>..<to>..</to>..</message>

17.5.1 DHTML (Dynamic HTML)

DHTML refers to web content that changes each time it is viewed. For example, the same URL could result in a different page depending on any number of parameters, such as:

1. Geographic location of the reader
2. Time of day
3. Previous pages viewed by the reader
4. Profile of the reader

DHTML refers to new HTML extensions that will enable a Web page to react to user input without sending requests to the web server.

17.6.1 Dynamic HTML

Dynamic HTML is a collective term for a combination of Hypertext Markup Language (HTML) tags and options that can make Web pages more animated and interactive than previous versions of HTML. Much of dynamic HTML is specified in HTML 4.0. Simple examples of dynamic HTML capabilities include having the color of a text heading change when a user passes a mouse over it and allowing a user to “drag and drop” an image to another place on a Web page. Dynamic HTML can allow Web documents to look and act like desktop applications or multimedia productions.

The Concepts and Features in Dynamic HTML

- An object-oriented view of a Web page and its elements
- Cascading style sheets and the layering of content
- Programming that can address all or most page elements
- Dynamic fonts

An Object-Oriented View of Page Elements

Each page element (division or section, heading, paragraph, image, list, and so forth) is viewed as an “object.” (Microsoft calls this the “Dynamic HTML Object Model.” Netscape calls it the “HTML Object Model.” W3C calls it the “Document Object Model.”) For example, each heading on a page can be named, given attributes of text style and color, and addressed by name in a small program or “script” included on the page. This heading or any other element on the page can be changed as the result of a specified event such as a mouse passing over or

being clicked or a time elapsing. Or an image can be moved from one place to another by “dragging and dropping” the image object with the mouse. (These event possibilities can be viewed as the reaction capabilities of the element or object.) Any change takes place immediately (since all variations of all elements or objects have been sent as part of the same page from the Web server that sent the page). Thus, variations can be thought of as different properties of the object.

Not only can element variations change text wording or color, but everything contained within a heading object can be replaced with new content that includes different or additional HTML as well as different text. Microsoft calls this the “Text Range technology.”

Although JavaScript, Java applet, and ActiveX Web pages, dynamic HTML implies an increased amount of programming in Web pages since more elements of a page can be addressed by a program.

A feature called dynamic fonts Web page designers include font files containing specific font styles, sizes, and colors as part of a Web page and to have the fonts downloaded with the page. That is, the font choice no longer is dependent on what the user’s browser provides.

17.7.1 Web Scripting

The process of creating and embedding scripts in a web page is known as Web-Scripting. A script or a computer-script is a list of commands that are embedded in a web-page normally and are interpreted and executed by a certain program or scripting engine. Scripts may be written for a variety of purposes such as for automating processes on a local-computer or to generate web-pages on the web.

The programming languages in which scripts are written are called scripting languages. There are many scripting languages available today. Most common ones are VBScript, JavaScript, ASP, PHP, PERL, JSP etc.

Types of Scripts

Scripts are broadly of following two types:

1. Client-Side Scripts

Client-Side scripting enables interaction within a web page. The client-side scripts are downloaded at the client-end and then interpreted and executed by the browser. The client side scripting is browser dependent. That is, the client side browser must be scripting enabled in order to run scripts.

Client-side scripting is used when the client-side interaction is used. Some sample uses of client-side scripting may be

1. To get data from users screen or browser
2. Online games
3. Customizing the display of page in browser without reloading the page.
Example rollover a hyperlink highlights that link without reloading the page.

Server-side scripts

Server-side scripting enables the completion or carrying out a task at the server end and then sending the result to the client end. In server side script, the server does all the work, so it doesn't matter which browser is being used at client end.

Server side scripting is used when the information is sent to a server to be processed at the server end. Some sample uses of server side scripting may be

1. Password protection
2. Browser customization
3. Form processing
4. Building and displaying pages created from a database.
5. Dynamically editing changing or adding content to a web page.

Some popular server side scripting languages are PHP (Hypertext Pre Processor),

Summary

- >HTML structure
- > HTML text,background, layout, numbering
- > links
- > Web hosting
- > Domain reistration
- > XML
- >domain
- >Web scripts

Review questions

One mark questions:

1. What is HTML?
2. What will be the extension of hypertext markup language file?
3. What is the use of web page?
4. What do you mean by domain?
5. What do you mean by hosting?
6. What is XML?
7. What is web scripting?
8. What is DHTML?

Two marks questions:

1. What are text files?
2. With the help of syntax include images in web page.
3. Write the steps for creating web page?
4. Write the opening and closing tags?
5. What is the use of netscape?

Three marks questions:

1. Explain the program to include tables in Web page.
2. What are steps used in creating Web Hosting?
3. How do you register an domain?
4. What is web scripting?
5. What is use of PHP files?
6. Give the features of XML?
7. Give the features of DHTML?
8. Write the differences of Client-side scripts?
9. Write the server-side scripting?
10. Create an web page for creating your college time table?
11. Create an Web page using forms?
12. What are advantages of web designing?
13. What are the advantages and disadvantages of www?
14. Write a note on URL?

Model Question Paper -I
PART – A

NOTE: Answer all the questions.

Each question carries one mark.

$10 \times 1 = 10$

1. What is a microprocessor?
2. Write the standard symbol for AND gate.
3. What are data structures?
4. Is it possible to access data outside a class?
5. How do we initialize a pointer?
6. What is normalization?
7. Expand ARPANET.
8. What are cookies?
9. What is URL?
10. Define a domain.

PART – B

NOTE: Answer any five questions.

Each question carries two marks.

$5 \times 2 = 10$

11. What are the fundamental products for each of the input words
 $ABCD = 0010$, $ABCD = 0110$, $ABCD = 1110$.
 Write SOP expression.
12. Draw a general K-map for 4-variables A, B, C and D.
13. Explain data encapsulation.
14. Why are constructors needed in a program? Justify.
15. Differentiate between stream class and ofstream class.
16. What is relational algebra?
17. What are the logical operators in SQL?
18. What is a SIM card?

PART – C

NOTE: Answer any five questions.

Each question carries three marks.

$5 \times 3 = 15$

19. Explain the characteristics of motherboard.
20. Draw the logic gate diagram to implement NOT gate using NAND and NOR gate.
21. Explain the memory representation of queue using arrays.
22. What is new operator in C++? Give example.
23. Mention the types of file. Explain any one.
24. Give the different notations for E-R diagram.
25. Write the advantages of WWW.
26. What are the steps involved in hosting a webpage.

PART – D**NOTE:** Answer any seven questions.

Each question carries five marks.

 $7 \times 5 = 35$

27. State and prove De Morgan's theorem algebraically.
28. What are the operations performed on linear data structures.
29. Write an algorithm to insert an element into the array.
30. Explain the advantages of OOP.
31. Illustrate with an example how an array of objects can be defined.
32. Explain the features of copy constructor.
33. Write a C++ program to find the volume of cone, cube and cylinder using overloaded function.
34. What are the types of inheritance? Explain any two.
35. Explain Codd's rules for database management.
36. Write the differences between orderby and groupby commands with example.
37. Give the measures for preventing virus.

Cmputer Science (41) March 2015 Total Marks :70 Time :3:15 Minutes

PART – A

NOTE: Answer all the questions.

Each question carries one mark.

$10 \times 1 = 10$

1. What is data bus? ch1
2. Which basic gate is also called as inverter?ch3
3. What is meant by primitive data structure?ch4
4. What is a member function?ch7
5. Write the declaration syntax for a pointer.ch11
6. Define primary key.ch13
7. Expand FTP.ch15
8. What is network topology?ch15
9. What is freeware?ch16
10. Mention the use of HTML.ch17

PART – B

NOTE: Answer any five questions.

Each question carries two marks.

$5 \times 2 = 10$

11. Prove that $X+ XY=X$. ch2
12. Define minterm and maxterm.ch2
13. Briefly discuss the classes in OOP.ch6
14. What is a destructor? Write its syntax.ch9
15. Differentiate between ifstream and ofstream.ch12
16. What is data independence? Mention the types of data independence.ch13
17. Give the syntax and example for DELETE command in SQL.ch14
18. Explain half duplex communication mode.ch15

PART – C

NOTE: Answer any five questions.

Each question carries three marks.

$5 \times 3 = 15$

19. What are ports? Explain serial port.ch1
20. Write the logic diagram and truth table for NOR gate.ch3

21. Write an algorithm for PUSH operation in stack.ch4
22. What are the operations performed on pointers? ch11
23. Give the function of put(), get() and getline() with respect to text files.ch12
24. Briefly explain one-tier database architecture.ch13
25. What is web browser? Mention any two web browsers.ch16
26. Explain any three text formatting tags in HTML.ch17

PART – D

NOTE: Answer any seven questions.

Each question carries five marks.

7 x 5 = 35

27. Given the Boolean function $F(A,B,C,D)=\Sigma(0,4,8,9,10,11,12,13,15)$. Reduce it by using Karnaugh map.ch2
28. Explain any five basic operations performed on arrays.ch4
29. Write an algorithm to delete a data element from the queue.ch4
30. Explain the advantages of Object Oriented Programming.ch6
31. What is a class definition? Write its general syntax and example.ch7
32. What is an inline function? Write a simple program for it.ch8
33. What is a constructor? Give the rules for writing a constructor function.ch9
34. What is inheritance? Explain any two types of inheritance.ch10
35. What is a data warehouse? Briefly explain its components.ch13
36. Explain various group functions in SQL.ch14
37. Explain any five networking devices.ch15

Computer Science (41) July 2015

Total Marks :70 Time :3:15 Minutes

PART – A

NOTE: Answer all the questions.

Each question carries one mark.

10 x 1 = 10

1. what is motherboard? ch1
2. what is a logic gate ?ch3
3. Give an example for linear data structure ?ch4
4. What is a class ?ch7
5. Mention any one advantage of pointer.ch11
6. What is a database? ch13
7. Expand URL.ch15
8. Define bus topology?ch15
9. What is freeware?ch16
10. Mention any one HTML tag.ch17

PART – B

NOTE: Answer any five questions.

Each question carries two marks.

5 x 2 = 10

11. State and Prove involution law. ch2
12. What is principle of duality ? Give an example.ch2
13. Differentiate between base class and derived class.ch6
14. Mention different types of constructors.ch9
15. What is a stream? Mention any one stream used in C++.ch12
16. Write any two advantages of database system.ch13
17. Mention any two datatypes used in SQL. ch14
18. Explain circuit switching technique.ch15

PART – C

NOTE: Answer any five questions.

Each question carries three marks.

5 x 3 = 15

19. What is the function of UPS? Mention different types of UPS.ch1
20. Write the logic diagram and truth table for a NAND gate.ch3

- 21.Explain the various operations performed on queue data structure.ch4
 22.What is array of pointers? Give an example. ch11
 23.List the different modes of opening a file with their meaning in C++.ch12
 24.Write the symbols used in E-R diagram, with their significance.ch13
 25.What is E-commerce? Explain any two types.ch16
 26.Explain is web-hosting? Mention different types of web-hosting.ch17

PART – D

NOTE: Answer any seven questions.

Each question carries five marks.

$7 \times 5 = 35$

- 27.Reduce $F(A,B,C,D)=\Sigma(1,2,3,4,5,7,9,11,12,13,15)$ using Karnaugh map.ch2
 28.Explain the memory representation of stack data structure using arrays.ch4
 29.Write an algorithm for Binary search. ch4
 30.Mention any five applications of OOP. ch6
 31.What are access specifiers? Explain any two with examples. ch7
 32.What is function overloading? Explain the need for overloading. ch8
 33.Explain destructor with syntax and example. ch9
 34.What is inheritance? Mention its advantages.ch10
 35. Define the following database terms :
 a) Data Model
 b) Tuple
 c) Domain
 d) Primary key
 e) Foreign Key ch13
 36. What is data definition language ? Explain SELECT and UPDATE commands
 ch14
 37. EGive the measures for preventing virus. ch15
