

Information Technology and Programming–I

Class –XI

**BOARD OF SECONDARY EDUCATION
RAJASTHAN, AJMER**

Information Technology and Programming–I

Class –XI

Translators

Dr. Vishnu Goyal (Convenor)

Govt. R.C. Khaitan Polytechnic College, Jaipur

Dr. Anil Gupta

HOD Computer Science & Engg.

MBM Engineering College

Jai Narain Vyas University, Jodhpur

Dr. Priyadarshi Patni

Director

Lachoo Memorial College, Jodhpur

Maninder Singh Bhui

Assistant Professor

Centre for E-Governance, Jaipur

Yogesh Sharma

Assistant Professor

Centre for E-Governance, Jaipur

Arvind Sharma

HOD Computer Engg.

Sine International College of Engg., Jaipur

2018

SYLLABUS COMMITTEE

Information Technology and Programming–I

Class –XI

Dr. Vishnu Goyal (Convenor)

Govt. R.C. Khaitan Polytechnic College, Jaipur

Dr. Anil Gupta

HOD Computer Science & Engg.

MBM Engineering College

Jai Narain Vyas University, Jodhpur

Harjee Ram Choudhary

Asstt. Professor

Govt. Engg. College, Ajmer

Dalpat Singh Songara

Asstt. Professor

Govt. Women Engg. College, Ajmer

Amarjeet Poonia

Asstt. Professor

Govt. Women Engg. College, Ajmer

Vishnu Prakash Sharma

Asstt. Professor

Govt. Engg. College, Ajmer

Rajesh Kumar Tiwari

Principal

Govt. Sr. Sec. School, Bajunda, (Bhilwara)

ACKNOWLEDGEMENT

As per the directive of State Government, new syllabus has been prepared by Board of Secondary Education, Rajasthan on the basis of major social, historical and cultural events at National and State Level, for the students with a view to provide them a platform for an overall personality development and establishing a meaningful relationship between their roots and academics.

Under the programme, in the first phase, text books have been prepared for the session 2016-17 for the students of IX and XI, session 2017-18 for the students of X and XII standard, who are pursuing education in the schools affiliated to the Board of Secondary Education, Rajasthan

Along with an insight into the social, cultural and historical benchmarks, factual information, project-based task and activity-based exercises have also been effectively dealt with in the prescribed books. The books will promote creativity, original thinking, contemplation and expression among the students. The modern techniques and teaching aids will make the learning more effective, interesting and result oriented.

I, therefore, on my behalf and on behalf of the Board of Secondary Education, Rajasthan extend my deep gratitude to the writers and Rajasthan State Text book Board for their kind co-operation in our endeavour to undertake the important work of text book writing and hope to get the same co-operation in future also.

Prof. B.L. Choudhary
Chairman

Board of Secondary Education, Rajasthan, Ajmer

PREFACE

Information and communication techniques have come with great revolution in our life. We are continuously pursuing economic prosperity with the help of information technology. E-commerce as an electronic consortium, email for sending mail through internet, online governmental E-governance, E-banking by bank behavior, E-education for online educational material etc. is a medium of continuously developing information technology. With the multi-dimensional use of information technology new doors of development are opening up.

Today the information and communication technology is developing its effective utility in research, business, industry, agriculture, medical, education, entertainment, communication, transport, environment, meteorology, space science, etc. so this basic knowledge has become necessary for all of us today.

Keeping in view the inevitability, the Secondary Education Board, Rajasthan has given it a place in its curriculum. The book is written in accordance with the new curriculum of the Board for class 11 students. While keeping the technical terms in English, the effort has been made to make the language simple, clear and intellectualization. The technical terms have been given in accordance with the terminology accepted by the Government of India. The English word is also given in brackets as needed. This book is titled Information Technology and Programming-I.

This book contains complete information about C programming. Object Oriented Programming, Computer Networking, XML database management System (DBMS) and PHP are also included as per the curriculum. This book will provide complete information about programming to students. The authors explain the microcosm of programming in simple language. Programming constructs are explained in details through suitable example. At the end of the chapter, important points are given. Important questions which may be asked in the examinations are also given at the end of the chapter.

Your suggestions are welcome.

Convenor

Syllabus

Class – XI

(Information Technology and Programming–I)

Unit-1: Introduction to C Language

Introduction to 'C' Language, history characterisation of C Language, properties of C language, structure of a c program, pre process of directive, character set, constant, variable, identifiers. Data type-declaration, keyword, input-output function, operators, type conversion of expression, precedence of operators, control statements : if statement, if-else statement, nested if-else statement, switch statement, looping : for loop, while loop and do-while loop, nested loop, break statement, continue statement.

Unit 2: Programming in 'C' Language

Array: declaration and initialization, one dimensional array, two dimensional array and multi-dimensional array. Function: pre-defined functions, user defined functions – declaration and definition of a function, String and if functions, global and local variable, storage classes, actual and formal parameters, calling a function, call by reference, call by value, passing array to the function. Recursion, introduction to pointers, structures, array with structure.

Unit 3: Object Oriented Programming

Concept of object oriented programming, structures programming, procedural programming and object oriented programming: difference, advantages and limitations. Characteristics of OOP: Object and Classes, Data encapsulation, data abstraction, polymorphism and inheritance. Structure of a C++ program – tokens, keywords, constants, basic data types. Output using cout and input with cin directives.

Unit – 4: Computer Networking

Introduction to Networking- Network Characteristics. Components. Network Locations. Data Communication Model. Components of Data Communication system Performance. Consistency. Reliability. Recovery, Security; Network Topologies - Star, Bus, Ring, Mesh, Tree, Hybrid; Standardization & Protocols - Need for Standards, Standardization Committees, Internet standards; Transmission Modes & Data Transmission - Simplex, Half Duplex, Full Duplex, Parallel Transmission, Serial Transmission; Categories of Networks - LAN, MAN, WAN, PAN, Internetworks;

OSI Model - Physical, Data Link, Network, Transport, Session, Presentation, Application Layers; TCP/IP Model ; Signals in Networking - Analog, Digital, Periodic, Non Periodic Signals; Introduction of Transmission Media in Networks- Twisted Pair Cable, Coaxial Cable, Optical Fiber Cable, Radio Wave Transmission, Satellite Communication; Introduction about Networking Devices - Bridge, Hub, Switch, Router, Repeater etc ; Introduction of Addresses in Networking(MAC & IP) - Classes, IP Address Components, Subnet Mask, IP Address Planning, Introduction about MAC & IPv6; Introduction about Wireless - Wireless LAN Technologies; WLAN standards & security - RF Bands, 802.11, WLAN clients Access to the Network, Access Modes, Coverage Areas, Data Rates, WLAN Devices; Congestion Control & Quality of Services - Data Traffic, Congestion, Congestion Control, Load shedding, Jitter Control, QoS, Technique to improve QoS, Integrated Services, Differential Services; DNS & E-Mail- DNS, Name Space, DNS in the internet, DNS Message, Resolution Process in DNS, DNS Poisoning, E-Mail, Services of E-Mail, E-Mail Architecture, Mail Server;

Unit 5: XML

XML Basics: Introduction and Overview, Introduction of Markup Language, Structuring Data, Creating Well-Formed XML Documents, Creating Valid XML Documents, XML Features.

Creating XML Documents: The XML Declaration, Tags and elements, Attributes, References, Text. DTD Basics: The Need for XML Document Schema, Types of XML Document Schema, DTD Schema Definition, Validating the XML Document, Entity Declaration XML Schema Basics: XML Schema: Differences between XML Schema and DTD Definitions, XML Schema Structure. XML Namespaces: Introduction of XML Namespaces, Namespace Declaration, Default Namespace.

Unit – 6: Database Management System

Database Concepts: Introduction & definition of Database, usage of Database, Database Management System (DBMS), advantages of DBMS. Concept of domain, tuple, relation, key, primary key, alternate key, candidate key, example of database management system; Introduction to RDBMS

Structured Query Language: Introduction, Characteristics of SQL, Advantages of using SQL, data definition language (DDL), data manipulation language (DML).

MySQL commands: CREATE TABLE, DROP TABLE, ALTER TABLE,

UPDATE...SET..., INSERT, DELETE; SELECT, DISTINCT, FROM, WHERE, IN, BETWEEN, GROUP BY, HAVING, ORDER BY; SQL functions number, string, date.

Unit-7: PHP

Introduction to web development using PHP: Basics of scripting, client side vs server side scripting, need of web-development using php, Script tags, variable declaration & access.

Data types: Commenting style in code, Primitive data types, Constant declaration & access, Functions.

Types of Loop: for, while, do-while.

Array: One-Dimension array, Multi-Dimension array, foreach loop for extracting array, Inbuilt-array: `$_GET[]`, `$_POST[]`, `$_REQUEST[]` etc.

Conditional Constructs in php: Different types of conditional construct: if – else, and else-if, switch-case statement. Operators: arithmetic, comparison, assignment, logical, conditional. Precedence of operator.

Connectivity for Php with MySQL: Connectivity code in php for MySQL database, different MySQL operations perform using php script like: Data Query Language (DQL), Data Definition Language (DDL), Data Manipulation Language (DML), Data Control Language (DCL).

INDEX

Unit -1

Chapter - 1 Introduction to C Language 1-51

Unit-2

Chapter - 2 Programming in C Language 52-119

Unit-3

Chapter- 3 Object Oriented Programming 120-134

Unit-4

Chapter- 4 Computer Networking 135--183

Unit-5

Chapter - 5 XML 184-199

Unit-6

Chapter-6 Database Management System (DBMS) 200-236

Unit-7

Chapter - 7 PHP 237-274

Chapter-1

Introduction to 'C' Language

Different programming languages develop for the Computer. These are designed for specific purpose. Machine language was first language to be developed as Computer programming language. It is very hard to write a Computer program in this language because it has very large Instruction set. It can directly run hardware of Computer System.

FORTRAN was developed for solution of scientific and mathematical problems and COBOL was developed to solve commercial problems. 'C' language was developed for the need of the current scenario. This language can be used in all types of works.

In 1972, 'C' language was developed by Dennis Ritchie in the Bell Telephone Laboratories Industry (Now, it is AT & T Bell Laboratories) as the successor of BCPL (Basic Combined Programming Language) and 'B'. This language can be used with both Unix and Dos Operating systems. It requires different compiler program. The new version 'C++' of ANSI 'C' and TURBO 'C' was developed in 1980.

1.1 Characteristic of 'C' language

1. 'C' language is a flexible language so that it is easy to make different kinds of programs.
2. The program is written in this language, can run speedily.
3. 'C' language is a high level language but it also has the characteristics of low level language.
4. The program is written in this language can store in less memory.
5. 'C' language program can be arranged in blocks, so that it can easy to make large program.
6. Pointers are used to make the program in 'C' language.
7. Bit operators (0 or 1) are presented in the 'C' language, which makes easy to work with Bytes.

1.2 Structure of Program

The program of 'C' language is the group of different functions. Every function is a group of statements and it performs a special task. The first function called by the compiler is main() function. It must be present in every 'C' language program. It is very important to know the structure of the program, where, we have to write the statements in the program. The structure of the program is as following :

1. Pre-Processor directive or/ and Header file
2. Global declarations of variables and Function Prototype
3. main()
4. {
5. Local declarations of variables and Function Prototype of main function
6. Single or compound statements
7. }
8. Header of other function with its arguments
9. {
10. Local variables of function
11. Single or compound statements
12. }

Description

Line No. 1: The header file selects according to the library functions have to be used in which contain information that must be included in the program when it compiled. If we want to include maths library function (for example sin(), cos() and sqrt()) in the program then we have to write following statement.

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

A program can contain one or more header file and write in any order.

Pre-Processor Directive

The pre-processor directives are executed before the execution of the program by the compiler. They give the directions to the compiler. Some pre-processor directives

are #include, #define etc.

The #include pre-processor directive is used to include header files in the program and the #define is used to declare symbolic constants in the program.

Example :

```
#define A 10
```

A is a constant and its value is 10.

Line No. 2 : The variables are declared in this line. These variables can be used anywhere in the program. It can be used in function only if the same name variable is not declared in the function.

Example :

```
int A;
```

```
float B, C;
```

Here, the function prototypes are also be declared. And these functions can be called in any part of the program.

Line No. 3 : It is main() function. Every 'C' program must contain main() function because the first function called by compiler is main() function.

Line No. 4 : Start the boundary of the main() function with open curly bracket ({).

Line No. 5 : Variables are declared after the curly bracket, can be used within main() function.

These are also called local variables. These will be declared as global variables (Line No. 2). Here, the function prototypes are also be declared. And these functions can be called within the main() function.

Line No. 6 : Definition of main() function is written here. It includes single or compound statements.

Line No. 7 : End of the main() function boundary with close curly bracket (}).

Line No. 8, 9, 10, 11, and 12 : These lines are similar to line 3, 4, 5, 6, and 7. The function name and its arguments are written at line no. 8 that is also called header of the function.

These lines are optional and these lines write when a user defines a function in the program.

The comments are used for documentation. The compiler does not execute the comments. These are written between `/*` and `*/`.

Program 1 : Write a 'C' language program to findout the area of the rectangle.

```
#include<stdio.h>

/* To calculate the of Rectangle.*/

main()

{

    float a,b,area;

    printf("Enter Side of the Rectangle :\n");

    scanf("%f %f",&a,&b);

    area = a * b;

    printf("\nThe area of the Rectangle = %f",area);

}
```

Enter Side of the Rectangle : 12 14

The area of the Rectangle = 168

Program 2 : Write a 'C' language program to calculate the given formula.

```
#include<stdio.h>

main()

{

    float m,v,C;

    printf("Enter The value of m & v :\n");

    scanf("%f %f",&m,&v);

    C =0.5 * m * v * v;

    printf("\nThe value of C = %f",C);

}
```

Enter The value of m & v : 2.5 1.7

The value of C = 3.612500

1.3 Character set of 'C'

The 'C' language includes lower and upper case letters, digits, and special symbols.

Alphabets A, B, C, Z

 a, b, c, z

Digits 0, 1, 2, 3, 9

Some of the special characters are shown in the table 1.1

Table - 1.1

Symbol	Meaning	Symbol	Meaning
#	Hash	?	Question Mark
<	Less than	:	Colon
>	Greater than	;	Semicolon
=	Equal to	^	Caret Sign
+	Plus	&	Ampersand
-	Minus	~	Tiled
*	Asterisk	\$	Dollar sign
%	Percentage	,	Comma
/	slash	\	Back slash
@	At the rate	'	single quotes
"	Double quote	!	Exclamation sign
	Filter sign		

1.4 Identifier

Identifiers are the name given to various program elements such as variable name,

function name, and symbolic constant name etc. by the user.

The rules for writing identifier are following :

1. The first letter of identifier must be a letter.
2. After the first letter, the identifier can include both uppercase and lowercase letters, all digits and underscore (_).
3. The maximum size of the identifier is 31 characters.
4. Keyword (or reserve word) cannot be used as identifier.
5. 'C' language is case sensitive language, lowercase letters are different from uppercase letters. Mostly, should use lowercase letters.

1.5 Constant :

The value of the identifier does not change during the execution of the program. Constants are three types.

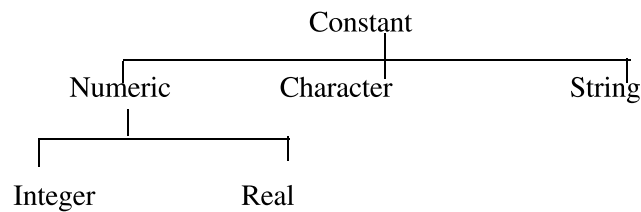


Fig. 1.1 Types of constants

1. Numeric Constant :

These are three types

(a) Integer Constant :

These are made of numbers and not included decimal points. These are shown in three different number system :

- (i) Decimal (Base 10)
- (ii) Octal (Base 8)
- (iii) Hexa decimal (Base 16)

Numbers and characters used in different number systems are shown in the Table-1.2.

Table-1.2

Type	Digits / alphabet	Example
Decimal	0, 1, 2, 9	0, 279, 972, 32767
Octal	0, 1, 3, 7	02, 027, 07777 Octal number starts with 0 (zero).
Hexa decimal	0, 1, 2, ... 9 and a, b, c (upper and lower case)	0x1, 0xab, 0xffff Hexa Decimal number starts with 0x or 0X.

(b) Real Constant :

The number which includes decimal points, they are called real number constant.

Example :

0.96, 872.127, 2.0 E-7

0.0693, 1.7676E + 10

2. Character Constant :

When a character written in a single quotation then it is called single character constant. It includes alphabets, numbers and special characters.

Example :

'A' '5' 'b' '\$'

3. String Constants :

When zero, one or more than one characters encloses within the double quotation then it is called string constant.

Examples :

"Home Loan"

"ALWAR"

" "

"2 * 6 * 7 - 5 + 2"

4. Back Slash Character Constant

These are also character constants, which are not used for printing character. These character constant are used for controlling the output. They are written with back slash (\). Some black slash constant and their meaning are shown in the table-1.3.

Table-1.3

Escape Sequence	Meaning
\a	bell (alert)
\t	horizontal tab
\v	vertical tab
\n	new line
\r	carriage return
\'	quotation mark
\b	back space
\0	NULL

1.6 Variable

Variable are identifiers, which store the value, and value can be changed during the execution of the program. The same rules are applicable on variable, which are used for writing identifier. Some valid variables are :

employee, result123, roll_num, min

And some invalid variables are :

short Reserve word

b' s Special character

5ort First letter must be alphabet

int eger blank space

1.7 Data Type

'C' programming language support different types of Data types. They takes different

size in memory to store the data. Generally, 'C' language uses four type of data types.

1. int To store integer value
2. char To store one character
3. float To store single precision floating point
4. double To store double precision floating point

Some of the datatype, size, keyword and range are shown in Table-1.4.

Table-1.4

Data type	key word	Size (in bits)	Range
Character	char	8	0 to 225
Integer	int	16	-32768 to 32767
	short int	16	-32768 to +32767
	long int	32	-2, 147, 483, 648 to +2,147, 483, 647
	unsigned int	16	0 to 65535
	unsigned long int	32	0 to 4, 294, 967, 295
	float	float	32
Double	double	64	1.7E-308 to 1.7E+308
	long double	80	3.4E-4932 to
	1.7E +4932		

1.8 Declaration

All variables must be declared before they are executed in the program. Variables are declared at the top (as shown in the program structure) of the program. The general format is :

<data type> <variable name>

Example :

```
int a, b, c;
```

```
char cl;
```

1.9 Keyword

The keywords (or reserve words) are the special words that have already defined in the 'C' language. The keywords can be used only for their intended purpose. The keyword cannot be used as identifier (User Defined Words) in program. 'C' language includes 32 keywords. They are shown in table-1.5.

Table-1.5

auto	float	const	struct
break	for	continue	switch
case	goto	default	typedef
char	if	do	union
double	int	short	unsinged
else	long	singed	void
enum	register	sizeof	volatile
extern	return	static	while

1.10 Operatorss

The symbols are used in the program for calculation by the computer. These symbols are called operators. The operator tells which operation to be performed. With the help of variable, constant and operator makes the expression. 'C' language includes Unary, Binary and Ternary operators.

Operators is used in the 'C' language are divided into different categories as follows:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Conditional Operators

5. Increment and Decrement Operator
6. Bitwise Operators
7. Assignment Operators

1.10.1 Arithmetic Operator

Arithmetic operators are used for the purpose of numerical calculations. 'C' language includes following Arithmetic operators are shown in table-1.6

Table-1.6

Operator	Purpose
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder (Remaining after division)

There is no exponentiation operator in 'C'. The library function `pow(x,y)` is used to calculate exponentiation which is available in the header file `<math.h>`. It's meaning X^y .

1.10.2 Arithmetic Expression

Variables and constants are interconnected with the help of operator called expressions. If both operands are integer variable or constant then it is called Integer Expression.

Example : if $a = 15$ and $b = 2$ then result of expression will be

Expression	Result
$a + b$	17
$a - b$	13
$a * b$	30
a / b	7
$a \% b$	1

The actual value of a/b is 7.5. If both operands are integer variable (or constant) then result will be in integer number. So the result of $15/2$ will be 7. It is shown in figure 1.2.

Integer division

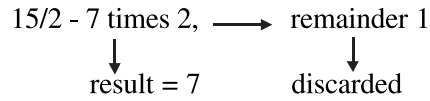


Figure 1.2

The modulus operator ($\%$), complement of the division operator in that it provides a means for us to obtain the remainder after integer division. It is shown in figure 1.3.

Modulus Operation

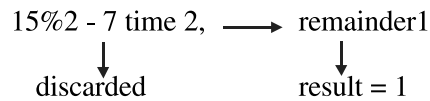


Figure 1.3

Other examples of modulus operator are as following :

$$- 15 \% 2 = -1$$

$$- 15 \% -2 = -1$$

$$15 \% -2 = 1$$

If both operands are real numbers then the result of the expression will be a real number. This type of expression are called real expression. If a operand is real number and other operand is integer number the result will be real number and this expression called a mixmode expression.

Example :

if $a = 15.5$, $b = 4.2$ then

$$a + b = 15.5 + 4.2 = 19.700000$$

$$a * b = 15.5 * 4.2 = 65.100000$$

above both operands are real numbers

Example :

if $a = 15.5$ (real), $b = 4$ (integer) then

$a + b = 15.5$ (real) + 4 (int) = 19.500000 (real)

$a * b = 15.5$ (real) * 4 (int) = 62.000000 (real)

above both are Mixmode expressions.

1.10.3 Relational Operators

Relational operators are used to compare two values. All operators are needed two operands. If it's value is true after comparison then it return 1 otherwise it returns 0 on false value. In the relational expression, both operands must have same data type. The relational operators used in 'C' language are as follows.

Table-1.7

Operator	Meaning
<	less than
<=	less than equal to
==	equal to
!=	not equal to
>	greater than
>=	greater than equal to

Example :

If p, q and r are integer variable and their values are 1,2 and 3 respectively. Result of the relational expression are given in the table.

Table 1.8

Relational expression	Result	value
$p < q$	true	1
$(p + q) >= r$	true	1
$(p + 5) == r + 4$	False	0
$r != 5$	true	1
$p * q > q * r$	False	0

1.10.4 Logical Operator

Logical operators are also used with two operators. These operators may be logical or relational expression. The result of logical operators are true or false. 'C' language includes three logical operators.

Table-1.9

Operators	Meaning
&&	and
	or
!	not

Logical AND (&&)

The result of the logical operator AND (&&) will be true only if both operands are true otherwise false.

Example :

if $i = 9$ and $j = 8.5$ then the result of the following expression will be true because both expressions are true.

$$(i > 6) \ \&\& \ (j < 9.5)$$

Logical OR (||) :

The result of the logical operator OR (||) will be true if one operand is true out of two operators otherwise false.

Logical NOT (!) :

Logical NOT is a Unary operator. There fore, it requires one operand. If operand is true then result will be false and if operand is false then result will be true.

1.10.5 Boolean Expression

The value of boolean expressions is true or False. A boolean expression is interconnected with boolean variable, constant and relational operators. Two boolean expressions can also be interconnected with logical operator.

Examples of logical operators are as follows :

- (i) age >55 && salary <1000
- (ii) number <0 && number >100
- (iii) !(status==1)

Example :

Let i=7, f=35 and c = 'W' ,here some of the complex logical expression are shown in the table.

Boolean Expression	Result	Value
(i >=6) && (c =='W')	True	1
(f >11) && (i >100)	False	0
C != 'a' (1 + f) <=10	True	1
C == i f > 20	False	0

1.10.6 Conditional Operator

This is a ternary operator, so it has three operands. This is write as follows :

variable = exp1 ? exp2 : exp3 ;

The exp1 is a boolean expression and it will evaluate first it, if exp1 is true then exp2 will evaluate and result will be stored in the variable otherwise exp3 will evaluate and result will be stored in the variable. Example :

if a = 5, b = 9

c = a > b ? a : b ;

The value of c will be 9. Because the value of expression a>b will be False and the value of exp3 will be stored in c which is b. So, the value of b will be stored in c.

1.10.7 Increment and decrement operators

'C' language has two very useful operators.

- (i) Increment Operator - ++
- (ii) Decrement Operator - --

These are unary operators. So, they required one operand. The increment operator (++) increase the value of the variable by one and decrement operator (--) decrease the value of the variable by one. These operators are written as follows :

++A or A++

and

--A or A--

If the operator is used before the operand (eg ++a), then the operand would be add or subtract one from the variable before it is utilized for its intended purpose within the program. If the operator follows the operand (eg a++) then the value of the operand would be add or subtract one from the variable after it utilized.

1.10.8 Assignment Operators

Assignment operators are used to assign the result of an expression to a variable. This is shown by '=' sign. This operator stores the value of the right hand side expression after calculation into the left hand side variable. The general format are as follows :

variable = Expression

Example :

These are the example of the assignment operator :

area = 2 * P1 * r1

x = y

a = 7.52

Total = A[1] + A[2]

In the above assignment statements, the left hand side must have only one variable (cannot be constant and expression).

'C' language also includes special assignment operator, these are as follows :

+=, *=, %=, -=, /=

Example :

x = x + 4 writes as x += 4

x = x * 5 writes as x *= 5

$x = x \% 4$ writes as $x \%= 4$

$x = x - 5$ writes as $x -= 5$

$x = x / 4$ writes as $x /= 4$

1.10.9 Type Conversion of expression

These are used to change the data type of the result of the expression. This writes as follows :

(Data type) Expression;

(datatype) expression, the value of the expression will be converted into the data type which is written in the bracket.

If a expression have more than one operand then the result of the expression automatically converted into higher precision data type, presented in the expression. This type of casting is called implicit type casting.

Example :

If $a = 7$ (integer) $b = 9.5$ (float) then the result of $2a + b$ will be 23.500000

Program 3

/ Program to convert a centimeter into meter-centimeter with integer Arithmetic*/*

```
#include<stdio.h>
```

```
main()
```

```
{
```

```
    int cent,meter;
```

```
    printf("Enter the value = ");
```

```
    scanf("%d",&cent);
```

```
    meter = cent / 100;
```

```
    cent = cent % 100;
```

```
    printf("\nMeters = %d and Centimeter = %d",meter,cent);
```

```
}
```

Enter the value = 1056

Meters = 10 and Centimeter = 56

Program 4

```
/* Program to explain explicit and implicit conversion */
```

```
#include<stdio.h>
```

```
main()
```

```
{
```

```
    int A=7,B=8;
```

```
    float X,Y=7.5;
```

```
    X = A + Y; /*Implicit Type Conversion*/
```

```
    printf("X = %f\n",X);
```

```
    X = (float)A/B; /*Explicit Type Convesion*/
```

```
    printf("X = %f\n",X);
```

```
}
```

Output:

X = 14.500000

X = 0.875000

1.11 Precedence of Operators

If an expression has more than one operator then it is necessary to know how they will execute in a sequence. It is predefined which operator will execute first. The precedence of the operators are predefined. The sequence in which same group operators in an expression are executed is determined by the associativity of the operator.

Table-1.10 :Precedence of Operators and associativity rules

Precedence group	Operator	Associativity
Function,array, struct member and pointer	() [] .→	L→R
Unary Operator	- ++ -- ~ * & sizeof(type)	R→L
Arithmetic multiplication, division and remainder	* / %	L→R
Arithmetic add and subtract	+ -	L→R
Bitwise shift operator	<<>>	L→R
Relational Operator	< <= > >=	L→R
Equality Operators	== !=	L→R
Bitwise AND	&	L→R
Bitwise XOR	^	L→R
Bitwise OR		L→R
Logical AND	&&	L→R
Logical OR		L→R
Conditional Operators	? :	R→L
Assignment Operators	= += -= *= /= %= &= ^= = <<= >>=	R→L
Comma Operators	,	L→R

Example :

If X = 7 y = 3.0 Z = 2 A = 2.5 B = 7 then solve the following expression

$$X + Y / (Z * A + B / Z)$$

Solution :

$$\begin{aligned} & X + Y / (Z * A + B/Z) \\ & = 7 + 3.0 / (2 * 2.5 + 7 / 2) \\ & = 7 + 3.0 / (5.0 + 7/2) \\ & = 7 + 3.0 / (5.0 + 3) \\ & = 7 + 3.0 / 8.0 \\ & = 7 + 0.375 \\ & = 7.375 \end{aligned}$$

Program 5

```
/* Program to calculate a expression */
#include<stdio.h>

main()
{
    float A,B=2.5,D=9.25;

    int i = 5,j=10;

    A = (float)i/j + (B*2)/(i + j) + D;

    printf("Result = %f\n",A);
}
```

Output:

A = 10.083333

1.12 Input-Output Function

Generally, a program do three work read data, process on data and gives output. Program uses functions to take input and to produce output. These functions are scanf(), printf(), getchar(), putchar(). The input/output are of two types formatted and unformatted. The function printf() and scanf() are formatted and putchar() and getchar() are unformatted I/O statement.

1.12.1 Console Input/Output

Console input/output functions are divided into two categories.

1. Unformatted console input/output functions
2. Formatted console input/output functions

The main difference between these two type of I/O statement is, the formatted console input takes the input from keyboard and print the output on the VDU according to the requirement of the user. But unformatted statement takes the data and print in normal form. For example we wants to print Total_item and sale_item on VDU. Where does it print on VDU and how much space is required between two data? This can be done using formatted console input/output statements. Both types of functions are shown in the figure given below.

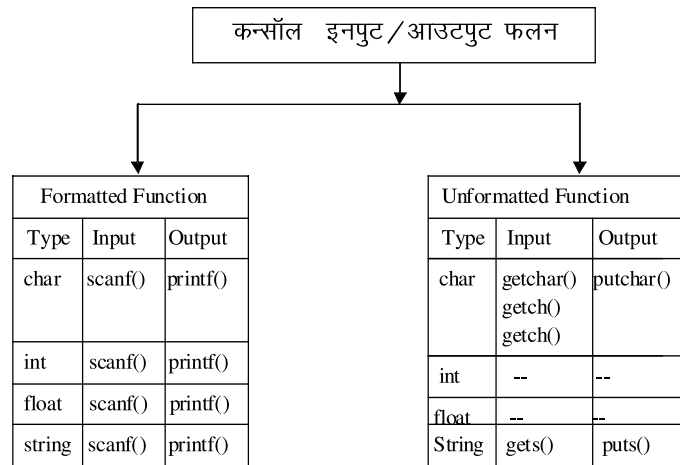


Fig. 1.4

1.12.2 Output Function printf()

The output function sends the values to output device or output file from the memory. The printf() function in the 'C' language sends data to output device from the memory. The printf() function write as follows :

```
printf("Control string", variable1,variable2,variable3);
```

The variable1,variable2,variable3 are variable name. The values of these variables will send to the output device. The control string writes between double quote (" "). The control string writes according to the data types to be printed. For example %d used for integer data type. The formats of the control string are shown in table-1.11.

Table-1.11

Format	Data type of variable
%d	int (Decimal)
%x	int (Hexa Decimal)
%o	int (Octal)
%h	short int
%u	unsigned int
%ld	long int
%f	float
%e	float (Exponent)
%lf	double float
%Lf	long double
%c	character
%s	string

Example :

```
A = 20, B = 30
```

```
printf("%d %d",A,B)
```

```
Output : 20 30
```

A and B are integer type variables and according to control string it print the value of A and B variables. The printf() function also prints a same string which is written in the double quotation.

Example :

```
printf ("C is a good programming language.")
```

```
Output : C is a good programming language.
```

1.12.3 Formatted printf() function

When printf() function print the value of the variables in the format form then we

have to place some extra character in the control string.

Output of integer numbers

The format specification for printing an integer number is :

`%wd`

Where *w* is an integer value that specifies the total number of column for the output.

Example :

```
int A=1476
main()
{
    printf("%7d",A);
}
```

The output of the above program is :

			1	4	7	6
--	--	--	---	---	---	---

This output will be written in seven columns but number will arrange in four columns. So, first three columns will be empty.

Output of real numbers

The format specification for printing a real number is :

`%w.pf`

Where *w* and *p* are two integer value and *w* specifies the total number of column for the output and *p* is the total number of column after decimal point.

Example

```
float x = 17.7927 ;
main()
{
    printf("7.3f", x) ;
}
```



```
}
```

The output of the above program is :

	1	7	.	7	9	2
--	---	---	---	---	---	---

This output will be written in seven columns. It will use three columns after decimal point, one for decimal point and two for number written before the decimal point. One column will be blank because it uses 6 column. Which is shown in the output.

Output of strings

The format specification for printing a string is :

`%w.ps`

Where `w` and `p` are two integer values and `w` specifies the total number of column for the output and `p` is the total number of characters to be printed.

Example :

```
char str[9] = "computer" ;
```

```
main()
```

```
{
```

```
printf("\n%12.5s", str) ;
```

```
printf("\n%.5s", str) ;
```

```
printfl("\n%12s", str) ;
```

```
}
```

The output of the above program is :

								c	o	m	p	u
c	o	m	p	u								
				c	o	m	p	u	t	e	r	

First `printf()` statement will write output in 12 column and only five characters of the string will be printed. Second `printf()` statement print only five character in first five column and third `printf()` statement print the "computer" string in 8 column out of 12 column and four column will be empty. Which is shown in the output.

The format specification for printing a single character is :

`%wc`

This character leaves blank (w-1) column and print in the wth column.

1.12.4 Input function scanf()

The scanf() function used to store data in the variable of the program from Input device (keyboard). The printf() function write as follows :

```
scanf(" Control String ",&variable1,&variable2,&variable3);
```

The &variable1,&variable2, and &variable3 are the address of the variables in the scanf where receive data from keyboard will stored. The ampersand sign (&) shows the address of the variable not value. The percentage sign (%) written with English alphabet in the control string as same in the printf() function. These are shown in the table-1.11.

Example :

```
scanf("%d%d%d", &NUM1, &NUM2, &NUM3) ;
```

NUM1, NUM2 and NUM3 are integer variable. All variable have to write different control string for different variable.

Program 6 : Write a 'C' language program to read three real numbers and findout the smallest number.

```
#include<stdio.h>
```

```
/* C program to read three float number and find smallest number.*/
```

```
main()
```

```
{
```

```
float a,b,c,small; /*Declaration of float numbers*/
```

```
printf("Enter three float Numbers :\n");
```

```
scanf("%f %f %f",&a,&b,&c); /*read three float numbers*/
```

```
if(a < b)
```

```
if(a < c)
```

```
small = a;
```

```

        else
            small = c;
    else
        if(b < c)
            small = b;
        else
            small = c;
    printf("\nThe Smallest Number = %f",small);
}

```

Input :Enter three float Numbers :

78.23 78.96 85.52

Output:The Smallest Numbers : 85.52

Input :Enter three float Numbers :

23.23 23.15 23.16

Output:The Smallest Numbers : 23.15

getchar() function

This function takes a character from the input device and gives to the computer. The general format is :

```
Variable Name = getchar( );
```

Example :

```

char c ;
.....
.....
c = getchar ( );

```

In the above example, the c variable define as char type. When second statement

c=getchar() is executed then computer waits for pressing a key on the keyboard(if data is not in a buffer) and store in the variable c. So, c = getchar() and scanf(" %c", &c) both are equivalent statement.

putchar() Function

This function puts a character from the memory to the output device. The general format is :

```
putchar(Name of char Variable);
```

The character variable defined previously which will print on the monitor.

Example :

```
char a;  
.....  
.....  
putchar(a)
```

In the above example, a variable defines as char type through the first statement. And second statement putchar(a) send a character to the output unit. The type of variable must be char. So, putchar(a) and printf("%c", c) both are equivalent statement.

Program 7 : Write a 'C' language program to read a line and print it in the uppercase.

```
#include<stdio.h>  
  
/* C program to read a line and print it into upper case.*/  
  
main()  
{  
  
    char c[80];/*Declaration of char array or string*/  
  
    int i;  
  
    printf("Enter A line:\n");  
  
    for(i=0;(c[i]=getchar()) != '\n';i++);/*This statement read a  
  
                                line*/
```

```

c[i]='\0'; /* Store NULL char at end*/

printf("The UpperCase line is:\n");

for(i=0;c[i]!='\0';i++)

{

    c[i] = toupper(c[i]);/*Convert into the UpperCase Letter*/

    putchar(c[i]); /*To print a Character on the screen*/

}

}

```

Input :Enter a line :

The string is a combination of character.

Output:The UpperCase line is:

- THE STRING IS A COMBINATION OF CHARACTER.

1.13 Control Statements

In the 'C' language, a statements executes in the sequential fashion in which sequence they are written in the program. Each statement is executed once and once only. But a logical condition decides which statement will execute and which will not execute in the program. The if and switch statements can fulfil the above requirement of 'C' language.

Some of the statements will be executed until the logical condition is true. It is called looping.

The control statements execute the program statement in the predefine sequence. These are the following control statements in the 'C' language :

1. Decision Making Statement
 - (a) if Statement (b) if else statement (c) switch statement
2. Loop statement
 - (a) for loop (b) while loop (c) do while loop
3. Other control statement

- (a) break (b) continue (c) goto

1.14 if statement

It is a powerful control statement which first check the condition than after it execute other statements according to result of the condition. The if statement are following four types.

- (i) Simple if statement
- (ii) if else statement
- (iii) Nested if else statement
- (iv) else if stairs

1.14.1 Simple if statement

First, it checks the condition in the statement. If the condition is True then it executes the group of statement otherwise it does not. One or more statement can be in the group of statement. The general format of the if statement is as following :

```
if(Condition)
{
    Group of statement
}
```

If control statement has one statement in the group of statement then there is no need to close within curly bracket. As shown in the example.

- (a) if (a > 10)

```
printf("a is greater than 10");
```

- (b) if (a>10|| b == 5)

```
printf("a is greater than10 or equal to 5");
```

- (c) if (n%2 ==0)

```
printf("N is a even number");
```

1.14.2 if-else statement

The control flow is bi-directional in this statement. First, it checks the condition. If the condition is true then it executes first group of statements. If it is false then it executes second group of statement. if-else statement will be written as follows :

```
if(Condition 1)
{
    Statement Group 1
}
else
{
    Statement Group 2
}
```

Examples of if else statements are as follows:

- (a) if (a > b)
- ```
 printf("a is greater than b");
else
 printf("b is greater than a");
```
- (b) if (n%2 == 0)
- ```
    printf("N is Even Number");
else
    printf("N is odd Number");
```
- (c) if (Basic > 50000)
- ```
{
 HRA = BASIC * 0.15 ;
 DA = BAISC * 0.61 ;
```

```

 }
else
{
 HRA = BASIC * 0.10
 DA = BASIC * 0.61 ;
}

```

The example (a) and (b) include the single statement so, there is no need to close with in curly bracket. But in the example (c) close the curly bracket due to two statements.

**Program 8 :** Write a 'C' language program to find the given year is leap year or not.

```

#include<stdio.h>
#include<conio.h>
/*It is C program to check for leap Year*/
main()
{
 int year,leap;
 printf("\nEnter the year :");
 scanf("%d",&year);
 if(year % 100 == 0) /* Checking for century*/
 if(year % 400 == 0)/* Century is leap year*/
 printf("It is century and leap year");
 else
 printf("It is century but not leap year");
 else if(year % 4 == 0) /* Checking for leap year*/
 printf("It is leap year");
 else

```



```
 printf("It is not leap year");
 getch();
}
```

Input/Output:

Enter the year :2000

It is century and leap year

Enter the year :2001

It is not leap year

Enter the year :2004

It is leap year

Enter the year :1900

It is century but not leap year

### **1.14.3 Nested if else statement :**

When an if else statement written within another if-else statement then it is called nested if else.

```
if (Condition 1)
{
 if (Condition 2)
 {
 Statement group1
 }
 else
 {
 Statement group 2
 }
}
```

```

Statement group 3
}
else
{
Statement group 4
}

```

First, it check the condition in the above control statement. If it is False then it starts to execute statement group 4. It left all the statements groups. If the condition 1 is true then condition 2 will be checked. If condition 2 is true then it execute statement group 1 otherwise on false it execute statement group 2. Then after control flow execute the statement group 3 and leave the if else statement and it will not execute statement group 4.

Example :

```

if (x >= 40)
{
if (x >= 65)
{
printf("FIRST DIVISION") ;
}
else
{
printf("SECOND DIVISION");
}
}
else
{

```

```
 printf("FAIL");
 }
```

#### **1.14.4 else if stairs**

This type helps to take decision on multi-way statement. This statement is used when the multiple condition is checked one by one. This is written as follows :

```
if (Condition 1)
{
 Statement Group 1
}
else if (Condition 2)
{
 Statement Group 2
}
else if (Condition 3)
{
 Statement Group 3
}
else
{
 Statement Group 4
}
```

#### **1.14.5 switch statement**

This is a multi-way conditional statement. It also have a condition similar to if else statement. Different statements will be executed according to expression or variable in the switch statement. The value of expression or variable must be integer or character. The format is as follows :

```

switch (Expression or variable)
{
 case 1 : Statement group 1
 break ;
 case 2 : Statement group 2
 break :
 case n : Statement group n
 break ;
 default : Statement group
}

```

Here switch, case, break and default are keywords. First of all, switch statement checks the value of expression or variable. It compares with the constant written with case keyword. If this value find equal to any case constant then it executes the statement group related to the case statement and it will be out from the switch statement after executing break statement otherwise it will execute all case statements. If the value of expression or variable does not match with any case value then it executes the default statement group. If default statement does not present in the switch statement then the control flow come out from the switch statement without executing any statement group if any case value does not match.

Example :

```

switch (N) /* N is an integer value */
{
 case 1 : printf("ONE");
 break ;
 case 2 : printf("TWO");
 break ;
 case 3 : printf("THREE");
}

```

```

 break ;
 case 4 : printf("FOUR");
 break ;
 case 5 : printf("FIVE");
 break ;
 default : printf("Enter Between 1-5")
 }

```

In the above example, it will print the numbers in words. The value of N compares with constant (1, 2, 3, 4 and 5). If there is any value matches with case constant then it prints a number in word otherwise it executes the default statement and control flow come out from the switch statement.

**Program 9 :** Write a 'C' program to print the grade of the students. Grades are as follows:

|          |                               |
|----------|-------------------------------|
| 100 - 90 | A +                           |
| 89 - 80  | A                             |
| 79 - 70  | B +                           |
| 69 - 60  | B                             |
| 56 - 50  | C                             |
| 50 - 0   | Fail (using switch statement) |

```

#include<stdio.h>
#include<conio.h>
#include<string.h>
/* It is C program to Use switch Statement.*/
main()
{
 int marks, Rollno;
 char Name[20], grade[5];

```

```

clrscr();

printf("Enter The Name of a Student : ");

gets(Name); /*Library function to read a string*/

printf("Enter The Roll No. of a Student : ");

scanf("%d",&Rollno);

printf("Enter The Marks Obtained : ");

scanf("%d",&marks);

switch(marks/10)

{

 case 10:

 case 9: strcpy(grade,"A+");break;

 case 8: strcpy(grade,"A");break;

 case 7: strcpy(grade,"B+");break;

 case 6: strcpy(grade,"B");break;

 case 5: strcpy(grade,"C");break;

 default: strcpy(grade,"Fail");

}

printf("\nName of a Student : %s",Name);

printf("\nThe Roll No. of a Student : %d",Rollno);

printf("\nThe Marks Obtained : %d",marks);

printf("\nThe Grade Obtained : %s",grade);

getch();

}

Result:

```

Input:

Enter The Name of a Student : Sunil Methi

Enter The Roll No. of a Student : 8542

Enter The Marks Obtained : 85

Output:

Name of a Student : Sunil Methi

The Roll No. of a Student : 8542

The Marks Obtained : 85

The Grade Obtained : A

Input:

Enter The Name of a Student : Karishma Methi

Enter The Roll No. of a Student : 8645

Enter The Marks Obtained : 45

Output:

Name of a Student : Karishma Methi

The Roll No. of a Student : 8645

The Marks Obtained : 45

The Grade Obtained : Fail

### **1.15 Looping**

Besides of decision-making statements, some statements are required to execute more than once in the program. When some statements execute again and again in the same order then this processing is called looping. for, while and do-while are three loop statement in the 'C' language. These all statements have two parts:

- (i) Looping body
- (ii) Condition

It checks the control statement in the loop and if this condition is true then loop

will be executed otherwise control flow will come out from the loop.

### 1.15.1 for loop

The general format of for statement is :

```
for (exp1 ; exp2 ; exp3)
{
 Loop body
}
```

Where exp1 : This tell the initial value of the counter variable and the initializing expression is evaluated once only, at the beginning of the for loop.

exp2 : This is a boolean expression. If it is true then loop will executing continuously.

exp3 : Increment/ decrement statement is used to increase/decrease the value of counter variable.

Example :

(a) for (i = 0 ; i < 10 ; i ++)

```
 printf("%d", i);
```

(b) for (j = 1 ; j <= 20 ; j += 2)

```
{
 printf("%d\n",j);
 SUM = SUM + j ;
```

```
{
 printf("SUM = %d", SUM);
```

(c) for (j = 1, i = 10; j < i ; j ++, i --)

```
 printf("%d %d", j, i);
```

**Program 10** : Write a program to add 10 numbers.

```
#include<stdio.h>
```



```
#include<conio.h>

main()
{
 int N,sum=0,i; /*Sum initialize by zero at declaration*/
 for(i=1; i<=10;i++)
 {
 printf("Enter %d Number :",i);
 scanf("%d",&N);
 sum += N; /*short hand assignment operator.*/
 }
 printf("\nSum of ten integer Number = %d",sum);
 getch();
}
```

Input/Output:

```
Enter 1 Number :23
Enter 2 Number :47
Enter 3 Number :98
Enter 4 Number :45
Enter 5 Number :85
Enter 6 Number :93
Enter 7 Number :123
Enter 8 Number :243
Enter 9 Number :24
Enter 10 Number :45
```

Sum of ten integer Number = 826

### 1.15.2 while loop

It is not known in advance, how many times a loop will be executed and it is depends on a condition. When the condition is true then the statements will be executed continuously. It written as follows :

```
while (Condition)
{
 Loop body
}
```

Example :

(a) while (i < 10) / \* initial value of i is one \*/

```
{
printf("%d", i);
i ++;
}
```

(b) while (j <= 20) / \* initial value of j is one \*/

```
{
printf("%d", j);
SUM = SUM + j;
j += 2;
}
printf("SUM = %d", SUM);
```

**Program 11 :** Write a program to add the digit of the given number.

```
#include<stdio.h>
```

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

```

/*It is C program to Add the digit of a given number*/

/*e.g. 12345 ,sum = 1 + 2 + 3 + 4 + 5 */

main()
{
 int m,N,sum=0;/* Sum initialize by zero at declaration*/

 printf("Enter a Number :");

 scanf("%d",&N);

 while(N != 0)
 {
 m = N % 10;

 sum = sum + m;

 N = N / 10;

 }

 printf("\nSum of digits = %d",sum);

 getch();

}

```

Input/Output:

Enter a Number :24563

Sum of digits = 20

### 1.15.3 do while loop

The do-while loop is similar to the while loop. First it check the condition in the while loop then after it executes the loop body. Whenever in the do-while loop, first, it executes the loop body then after it checks the condition. So, the do-while loop statement is always executed at least once. The general format of the do - while loop is :

```

do
{

```

```
Loop body
} while (Condition);
```

Example :

(a) do

```
{
 printf("%d", i); /* /initial value of i is one */
 i ++ ;
} while (i < 10) ;
```

(b) do /\* initial value of j is one \*/

```
{
 printf("%d", j);
 SUM + = j ;
 j + = 2 ;
} while (j < = 20) ;
```

**Program 12 :** Write a 'C' program to reverse a given number.

```
#include<stdio.h>
```

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

```
/*It is C program to reverse a given number*/
```

```
/*e.g. 12345 print as 54321*/
```

```
main()
```

```
{
 int m,N,rev=0;
 printf("Enter a Number :");
 scanf("%d",&N);
```

```

do
{
 m = N % 10;
 rev = rev * 10 + m;
 N = N / 10;
}while(N != 0);
printf("\nReverse Number is = %d",rev);
getch();
return 0;
}

```

Input/Output:

Enter a Number :32512

Reverse Number is = 21523

#### 1.15.4 Nested Loop

When a loop runs inside another loop, it is called a nested loop. For example: a for loop inside another for loop. It must be remembered that the inner loop must close first and then the outer loop will close. As shown in the figure.

```

for ()
{
 for ()
 {
 {
 }
}

```

Example :

(a) for (i = 0 ; i < 10 ; i ++)

```
for (j = i; j < 10 ; j ++)
```

```
 printf("%d", i);
```

printf statement print 55 integers.

(b) for (i = 1 ; i < 10; i ++)

```
{ j=1;
```

```
 while (j <= 10)
```

```
 {
```

```
 printf("%d", i*j);
```

```
 j++;
```

```
 }
```

```
 printf("\n");
```

```
}
```

The above loop will print 1 to 10 tables.

### 1.15.5 break statement

Generally, break statement is used with loops or switch statement. It is used to terminate a loop. If break statement executes in nested loop then the control flow will come out from the loop where the break statement is be executed.

Example :

(a) for (i = 1 ; i <= 10 ; i + +)

```
{
```

```
 scanf("%d", &N) ;
```

```
 SUM = SUM + N ;
```

```
 if (SUM>= 200)
```

```
 break ;
```

```
}
```

The loop can terminate in two way first, the value of i reaches to 10 or SUM exceeds 200. When the value of SUM exceed 200 then break statement will execute and it terminate the loop.

### 1.15.6 continue statement

The break and continue, both statements are used to stop execution. But break statement stop the execution of the loop and control flow come out from the loop. The loop does not terminate when continue statement is encounter then the remaining loop statements are skipped and computation proceeds directly start the next pass of the loop. It Is written as:

Example :

```
do
{
scanf("%d", &N) ;
it (N < 0)
{
printf("Error");
continue ;
}
/* other statements */
} while (N <= 50) ;
```

In the above example, If N read a negative value then continue statement will be executed and it cause to start new pass of the loop.

### Important Points

1. The semicolon separates the statements. This is indication of end of statement. The semicolon does not attach after the pre-processor directives.
2. 'C' language is a case sensitive language. It means, uppercase letters are different from lowercase letters.
3. The printf( ) function sends the message to the output devices.

4. The scanf( ) function also calls a function which takes the input from input device.
5. In the 'C' language, all 32 keywords are in lower case letter.
6. 'C' language is a operator rich language. It includes arithmetic, relational, logical, bitwise etc. operators.
7. The precedence and associativity rules are applicable to solve a expression.
8. The main( ) function is also a user define function and it must be presented in the program.
9. scanf(), getchar(), getch(), getche() and gets() are input function.
10. printf(), putchar() and puts() are output function.
11. The decision statements if, if-else and switch are used in the 'C' language.
12. The part of the program can be executed more than once by looping. The for, while and do while loop are used in the 'C' language.
13. The break statement is used to stop execution of switch and loop and control flow comes out from the switch or loop.
14. The continue statement is used to stop execution part of the loop and start new iteration.

## **Exercises**

### **Objective Type Questions**

1. if(1)
 

```
printf("TRUE")
else
printf("FALSE")
```

|                        |                       |
|------------------------|-----------------------|
| (a) TRUE               | (b) FALSE             |
| (c) statement is wrong | (d) None of the above |
2. Who was developed the 'C' language?
 

|                      |                   |
|----------------------|-------------------|
| (a) Ken Thomson      | (b) Dennis Rechie |
| (c) Martin Recharson | (d) Donavon       |
3. The symbol used to separate the two statement in the 'C' language.



- (a) & (b) !  
(c) ; (d) “
4. Which must be presented in the 'C' language program?  
(a) Global variable (b) main() function  
(c) Pointer (d) Function
5. The meaning and uses are predefined, these words are called.  
(a) Variable (b) Constant  
(c) Identifier (d) Reserve word
6. Which is not a reserve word?  
(a) auto (b) while  
(c) switch (d) total
7. The valid constant is :  
(a) 4, 78 (b) OX 23 A  
(c) 177A (d) 5.7, 632
8. Which is not a valid identifier?  
(a) 5XYZ (b) total\_subject  
(c) Stud\_mark (d) XSXY
9. The control string for long double data type in the printf statement.  
(a) %d (b) %lf  
(c) %ld (d) %Lf
10. The purpose of % (percentage) operator is  
(a) Addition (b) Division  
(c) Remainder (d) Multiplication
11. If  $i = 8$  and  $b = -i + 3$  then the value of b.  
(a) 11 (b) 10  
(c) 9 (d) 12
12. Which statement is used to stop the execution of the loop :  
(a) break (b) stop  
(c) end (d) None of these

```
13. for(i=0;i<=5;i++)
 for(j=i;j<=5;j++)
 printf("India");
```

In the above program, How many times will print India.

- (a) 6                      (b) 21  
(c) 36                     (d) 30

14. In which statement a case keyword is used-

- (a) switch                (b) for  
(c) do while              (d) while

15. Which loop will first terminate in the nested looping :

- (a) Outer loop            (b) Inner loop  
(c) Both at once         (d) None of these

### Very Short Type Questions

1. What is reserve word?
- 2.. How many types are arguments are there?
3. How many times main( ) function can be used in the 'C' language?
4. What is use of ++ operator.
5. What do you mean by getchar( ).
6. Which function is used for giving output?
7. In which structure the "default" statement is used?
8. How does nested if else statement written?
9. Where is continue statement used?
10. How many loop structures are presented in the 'C' Language?
11. Explain do while structure with the help of an example.

### Short Type Questions

1. What is the meaning of algorithm?
2. What is constant?
3. How does declare a variable?
4. Differentiate between string constant and character constant.
5. What is identifier?
6. Explain ternary operator.

7. What is precedence of the operators? Explain.
8. When does it require associativity rules for operators?
9. Why does 'C' language call middle level language?
10. What is nested loop?
11. Explain do-while statement.
12. Write all types of the if statement.
13. How does switch statement writes, write format.
14. Differentiate between continue and break.
15. Write the output of the following program :

```
(a) #include<stdio.h>
 main ()
 {
 int i = 1, x = 1 ;
 for (i = 1 ; i < 10, i++)
 {
 printf("%d\n", x+i);
 x = x + 1;
 }
 }
```

```
(b) # include <stdio.h>
 # define p 10
 {
 int k =1, w = p;
 while (k <=w)
 {
 printf("%d\n",k);
 }
 }
```

**Essay Type Questions**

1. Explain the structure of the 'C' language program.
2. What is meaning of program structure?
3. How many type of data types are there in C language?
4. Explain the input/output statements used in the 'C' language.
5. What do you understand by precedence and associativity of the operators?
6. Explain all input function with the help of an example.
7. How does you get the output from the output function? Explain with the help of an example.
8. Convert the following expression into 'C' equivalent expression.

(i)

$$a^2 + b^2$$

(ii)

(iii)

$$(iv) \quad 1 + \frac{a}{b + 1/c}$$

$$\frac{ab}{c} - \frac{b^2}{d} + d$$

9. Write a program to read three numbers for finding largest number with using conditional operator.
10. Which statements are used for looping? Explain.
11. Write a 'C' program for printing all odd numbers from 1 to 50.
12. Differentiate while loop and do-while loop with the help of examples.
13. Write a program to findout the sum of the following series.

$$1 + x + x^2 + x^3 \dots\dots\dots x^n$$

**Answer Key**

1. a    2. b    3. c    4. b    5. d    6. d    7. b    8. a    9.d  
10.c    11. b    12. a    13. b    14. a    15. b

## Chapter-2

### Programming in 'C' Language

#### 2.1 Array

When we have to make a list of 10 students then we have to declare 10 variables. If the number of students increases then variables will be increased and the program will be complex. In the 'C' language, the Array is used to solve this problem.

Array is a variable which can collect similar types of data. We have studied that a variable can store only one value at a time. But some time, it is required where we use similar and related data. It is typical to store all values in the different variables. So, a variable can store all values. In this group, every value indicate by its index value. The type of data can be char, int, float, double etc. If there are 50 students in the class then we have to use an array named marks which can store all marks and it is known as marks[50]. The number 50 is written in the square bracket([ ]),that it shows, it is a group of 50 elements. Every element recognized by their index value. For example marks[8] means the number is stored at position eight.

Example :

The ten elements of an array P[10] is as follows :

P[0], P[1], P[2], . . . P[8], P[9]

#### 2.1.1 Array Declaration

An array must be declared before it used in the program as same as the other variables, three thing are declared in an array declaration :

- (i) Type of an Array
- (ii) Name of an array
- (iii) Capacity of an array (Number of Elements)

When we declare an array then the memory will be allotted according to the size of the array. An array declares as follows :

< Type of array > < Name of array > < size >

Example :

```
int marks[50] ;
```

Here, int marks[50] tells us that the size of marks is 50 and it can store integer only.

### 2.1.2 Processing On Array

The processing on an array, is done on doing processing on each element. It is required a loop for processing on an array. When a loop executes it processed all element.

Example :

```
for (i = 0; i<10; i++)
 scanf("%d",&A[i]) ;
```

In the above statements, it takes 10 inputs for an array. It will be clear from the following example :

**Program 1 :** Write a 'C' language program to arrange 10 numbers.

```
#include<stdio.h>
#include<conio.h>
/*It is C program to sort 10 numbers using selection sort. */
int main()
{
 int A[10];
 int i,j,temp;
 printf("\nEnter Ten Values :\n");
 for(i=0;i<10;i++)
 scanf("%d",&A[i]);
 /* Sorting Algorithm */
 for(i=0;i<9;i++)
 for(j=i+1;j<10;j++)
```

```

 if(A[i] > A[j])
 {
 temp = A[i];
 A[i] = A[j];
 A[j] = temp;
 }

printf("\nThe Sorted Data are :\n");

for(i=0; i<10; i++)
 printf("%d ", A[i]);

getch();

return 0;

}

```

Result:

Enter Ten Values :

5 19 45 63 7 4 78 12 89 56

The Sorted Data are :

4 5 7 12 19 45 56 63 78 89

### **2.1.3 Type of Array :**

Array are two types :

1. One Dimensional Array
2. Multi Dimensional Array

#### **1. One Dimensional Array**

The one dimensional array has one row or one column. For example , there are 50 students in the class and uses one dimensional array to store their marks. If the array name is Marks then the values will be :

Marks [ 0 ] = 77

Marks [ 1 ] = 55

Marks [ 2 ] = 67

Marks [ 3 ] = 65

Marks [ 4 ] = 89

:

:

:

Marks [ 47 ] = 78

Marks [ 48 ] = 48

Marks [ 49 ] = 52

## 2. Multi Dimensional Array

It is required most of the time when the data can not be stored in the single dimensional array. For example, if there are 20 students in a class and store marks of five subject of each student then we require a table having 20 row and 5 column. The multi dimensional array is a kind of array which have two or more dimension. It is written as follows :

<Type of array><Name of array>[Max size 1] [Max size 2].....[Max size n]

If it is two dimensional array then two square bracket ( [ ] ) will be attached.

Example :

In the array A[5][5], its first element A[0][0], second element is A[0][1] and similarly last element is A[4][4]. It has 5 row and 5 column. It is shown in figure 3.1.

|   | 0  | 1  | 2  | 3  | 4  |
|---|----|----|----|----|----|
| 0 | 9  | 6  | 16 | 24 | 61 |
| 1 | 16 | 72 | 95 | 97 | 55 |
| 2 | 49 | 57 | 40 | 45 | 25 |
| 3 | 29 | 64 | 5  | 18 | 2  |
| 4 | 10 | 12 | 98 | 59 | 26 |

Figure 3.1 : Array A[5][5]



### 2.1.4 Initialization of Multi Dimensional Array

The initialization of multi dimensional array is similar to one dimensional array.

Example :

```
int MAT[3][3]={0,1,2,3,4,5,6,7,8,};
```

In The above initialization, the row and column are not shown separately. The above array can also be initialized in row and column format.

```
int MAT[3][3]={ {0,1,2,} {3,4,5,} {6,7,8,}};
```

The values of first row will be enclosed in first bracket. The value of second and third row will be enclosed in second and third bracket respectively. If the total values in the bracket is less than the size of an array then remaining values will be automatically initialized by zero.

Example :

```
int MAT[3][3] = { {1,2,3,} {4,5,6,} };
```

The capacity of the array is 9 and there are 6 values in the bracket. So, remaining three value will be zero.

The values of an array will be as following :

```
MAT[0][0]=1 MAT[0][1]=2 MAT[0][2]=3
```

```
MAT[1][0]=4 MAT[1][1]=5 MAT[1][2]=6
```

```
MAT[2][0]=0 MAT[2][1]=0 MAT[2][2]=0
```

**Program 2 :** Write a 'C' language program to multiply two 3 x 3 matrices.

```
#include<stdio.h>
```

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

```
main()
```

```
{
```

```
 /*It is C program to Multiply two 3 x 3 Matrix*/
```

```
 /*Array C is initialized by Zero*/
```

```

int i,j,k,A[3][3],B[3][3],C[3][3]={0};

/* Read Matrix A & B*/

printf("\nEnter Matrix A :\n");

for(i=0; i<3; i++)

 for(j=0; j<3; j++)

 scanf("%d",&A[i][j]);

printf("Enter Matrix B :\n");

for(i=0; i<3; i++)

 for(j=0; j<3; j++)

 scanf("%d",&B[i][j]);

/* Multiply Two Matrix A And B*/

for(i=0; i<3; i++)

 for(j=0; j<3; j++)

 for(k=0; k<3; k++)

 C[i][j] += A[i][k] * B[k][j];

printf("\nMultiply Matrix C :\n");

for(i=0; i<3; i++)

{

 for(j=0; j<3; j++)

 printf("%d ",C[i][j]);

 printf("\n");

}

getch();

}

```

Result :

Enter Matrix A :

1 2 3

3 2 1

6 8 4

Enter Matrix A :

1 4 5

6 2 1

5 3 8

Multiply Matrix C :

28 17 31

20 19 25

74 52 70

**Program 3 :** Write a 'C' language program to arrange 10 numbers.

```
#include<stdio.h>
```

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

```
/*It is C++ program to sort 10 numbers using Bubble sort.*/
```

```
int main()
```

```
{
```

```
 /*Array A declare with initialization.*/
```

```
 int a[10]={5,4,89,12,78,6,19,45,63,7};
```

```
 int i,j,temp;
```

```
 /* Sorting Algorithm*/
```

```
 for(i=0;i<9;i++)
```

```

for(j=0;j<9-i;j++)
 if(a[j] > a[j+1])
 {
 temp = a[j];
 a[j] = a[j+1];
 a[j+1] = temp;
 }
printf("\nThe Sorted Data are :\n");
for(i = 0; i<10;i++)
 printf("%d ",a[i]);
getch();
}

```

Result:

The Sorted Data are :

4 5 6 7 12 19 45 63 78 89

## 2.2 String

String is a group of characters. A string is defined as an array of character in the 'C' language. The group of character enclosed within double quotation then it is called string constant.

Example :

```
" C is better than other language." /* It is string constant*/
```

### 2.2.1 Declaration and Initialization of String Variable

A string is defined as an Array of char. String is defined as follows :

```
char <string name> [size]
```

The size is show the number of character in the string.

Example :

```
char name[20];

char address[30];
```

In the above example, the array can store 20 and 30 characters. But it can store 19 and 29 characters because last character of the string should be a NULL character ('\0'). NULL char is the indication of end of string.

Here, it is remembered that if we are processed whole string at a time then the NULL character automatically attached at the end of string. If we are processed a string char by char then programmer must be remembered to store NULL char at the end of string. Because in this case the compiler donot store NULL char. A string can be initialized as follows :

```
name[20]="RAMSHANKAER";

name[20]={ 'R','A','M','S','H','A','N','K','E','R', '\0'};
```

In the above initialization, we have remembered that if a string is used to initialize as a string constant then there is no need to assign NULL char otherwise a NULL character must be stored at the end of string. When we initialize a string then there is no need to write size in the square bracket.

Example :

```
name[]="RAM SHANKER";
```

In the above example, we omit the size of array. The size of array will be fixed according to the size of string constant. In the above example, the size of array will be 12.

### **2.2.2 Input a String**

There are many methods to read or input a string. A string can be read through scanf( ) function.

Example :

```
char name[20];

scanf("%s", name);
```

The ampersand (&) operator is not used with string variables because the name of string itself denotes the address of the string. In the above method, it can read a string upto the blank space. In the above scanf( ) function, gives an input "Ram Shanker" then it stores "Ram" in the string variable. If we want to solve the above problem then we have

to change the control string for the string.

Example :

```
scanf("%[^\\n]", Name);
```

It read the string before the new line character ('\n'). If we give an input and press Enter key then "Ram Shanker" will be stored in the string. We can also read a string char by char.

Example :

```
int i = - 1;

char name[20], ch;

while(ch!='\\n')
{
 ch=getchr();
 name[i]=ch;
}

name[i]='\\0' /*store NULL Char at end.*/
```

The above code can be written as follows with the help of for loop.

```
for(i=0; (name[i] = getchar())!='\\n'; i+ +);

name[i]='\\0';
```

In the above method, we can input a string upto the new line character. A library function (gets( )) is also available for inputting a string. It is also read a string upto the new line character.

Example :

```
gets(name);
```

### 2.2.3 Print a string

A string can be printed with the help of function printf( ) and puts( ).

Example :

```
printf("%s", name);
```

OR

```
gets(name);
```

In the above both methods, the string will be print upto NULL character.

#### 2.2.4 Library Function for string

'C' language includes library function for strings. Some of the important functions are shown in the table 3.1.

Table 3.1 : String Functions(string.h)

| Function  | Purpose                             |
|-----------|-------------------------------------|
| strcat( ) | Concatenates two strings            |
| strcmp( ) | Compare two strings                 |
| strcpy( ) | Copy one string over another string |
| strlen( ) | Find the length of a string         |

#### **strcat( ) Function :**

The syntax of strcat( ) function is :

```
strcat(string1, string2);
```

The string1 and string2 are character array. When the function strcat() will execute then string2 will append at the end of string1. The string2 remain unchanged.

Example :

```
char st1[10]="Computer";
```

```
char st2[10]=" System";
```

```
strcat (st1, st2);
```

After execution, the string will be :

```
st1="Computer System";
```

```
st2="System";
```

The string st2 append at the end of string st1. It is remembered that first argument must be a variable.

### **strcmp( ) Function**

The syntax of strcmp( ) function is :

```
n = strcmp(string1, string2);
```

It compares two strings 'string1' and 'string2' and gives the following result :

string1 == string2    The value of n will be zero.

string1 < string2    The value of n will be negative.

string2 > string2    The value of n will be positive.

Example :

```
strcmp("this","that");
```

In the above example, the string "this" is bigger because its first two characters are same and third character 'i' is bigger than character 'a' (ASCII value of i is 105 and a is 97). A string called a bigger string if its ASCII value is higher than other string. And it returns the difference of the ASCII values. In the above example, it returns value 8(105- 97 = 8).

Example :

```
strcmp("Kapil","Anish"); return -7
```

```
strcmp("Mohan","Mohan"); return zero
```

```
strcmp("Anish","Anil"); return +7
```

The strcmpi() function is also used for comparing two strings which ignore the case (upper and lower) of the string means both strings are same. The strcmp( ) function makes difference in both cases.

### **strcpy( ) Function**

The syntax of strcpy( ) function is :

```
strcpy(string1, string2);
```

The string1 must be a variable and string2 will be variable or constant. The characters of the string2 will be copied into the string1.



Example :

```
strcpy (State, "Rajasthan");
```

It will copy the string constant "Rajasthan" into the variable state.

### **strlen( ) Function**

The syntax of strcpy( ) function is :

```
n = strlen(String);
```

,

It stores the string length (total character) in the variable 'n'. String may be variable or constant.

Example :

```
n = strlen ("Rajasthan");
```

The value of n will be 10. It also count the NULL character.

### **2.2.5 Two Dimensional String Array**

If we want to store more than one string in an array then we have to use two dimensional array. It is declared as follows :

```
char name [10][20];
```

In the above array, it can store 10 string and their length may be 20.

This string can be read as follows :

```
for (i= 0; i < 10; i + +)
 scanf ("%s", name[i]);
/* or gets (name[i]);*/
```

And print as follows :

```
for (i= 0; i < 10; i + +)
 printf ("%s", name[i]);
/* puts (name[i]);*/
```

**Program 4 :** Write a program to find whether a given string is Palindrome or not.

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

main()

{

 int m,n,flag=1;

 char str[80];

 clrscr();

 printf("\nEnter a String :");

 gets(str);

 m=0;

 n= strlen(str) - 1;

 while(m <= n)

 {

 if(str[m] != str[n])

 {

 flag = 0;

 break;

 }

 m++;

 n--;

 }

 if(flag==1)
```

```

printf("\nString is Palindrome");
else
printf("\nString is not Palindrome");
getch();
return 0;
}

```

### 2.3 Function

It is typical to write a large (complex) program. So, a 'C' language program is divided into small parts. These small parts are completed a process. These all parts are combined in a sequence that it solves a complex problem.

All high level languages have this provision that the name is given to the block and written separately, when we require to call the block we can call it number of times. This independent block is called subprogram or function.

Different program blocks are made for every problem. These blocks are called function. Definition of the function can be written as follows :

**" A function is a self contained program segment that is used for some specific well defined task ."**

Function has important roll in the 'C' language. We have already used scanf( ), printf( ) functions through main( ) function. In the 'C' language, functions are divided into two parts.

#### 2.3.1 main( ) function :

All 'C' program start their execution from the main( ) function. The main() function define as follows:

```

main()
{
statements;
}

```

This function is perfectly valid in 'C' language because main( ) does not return any value but new compiler return int type value. If we donot want to return a value then

we have to use void key-word before the main( ) function.

Example :

```
void main()
{
 statements;
}
```

Therefore, explicitly defines main( ) as matching one of the following prototypes.

```
int main()
void main()
int main (int argc, char * argv[])
void main (int argc, char * argv[])
```

If we are not used int or void keyword then it will return integer value. Similarly, if the function does not specify return type value then it also return integer type value.

### **2.3.2 Advantages of Function**

1. The function is divided into small parts so, the large program can be written easily.
2. A program can read and understand easily using function.
3. We can add a new function in the program, which can easily change the program.
4. The complex problem can easily be solved by the function.
5. The error can be found out easily by the function.
6. All statements, which are repeated in the program, are written in the form of function so, that it is not required to write repeatedly in main function. It reduces the size of the program.
7. The complex program can easily be understood by using functions.
8. In the 'C' language, function is a subprogram which is use for special purpose.

In the 'C' language, a program has one or more functions. In the every program, the main( ) function must write once and once only. The function will not return any value

to the user. There are many function which are already available in 'C' language. For example, scanf(), pow( ) etc. Every function returns a value at a time. A user define function is called using in the main( ) function. When a function calls in a main function the control flow leave the current instruction and transfer to the function where it is written. After execution of the function again it return to the instruction where control flow leave the instruction. We can call a function several times in the main( ) function.

### 2.3.3 Types of Functions

Functions are divided into two parts :

1. Library Functions
2. User Defined Functions

#### 1. Library Functions :

The function have defined in the language, are called library function. They can use directly in the program. This facility is available in all high level language. It is very useful to collect general function in the library. This is helpful to make large program. The library function are shown in table 2.2. Library function - printf, scanf, sqrt, calloc, malloc, starcat & many more.

**Table 2.2 Library Function**

| <b>S. No.</b> | <b>Name of Function</b> | <b>Purpose of Function</b>                        | <b>Header file name</b> |
|---------------|-------------------------|---------------------------------------------------|-------------------------|
| 1.            | getchar()               | returns the next character typed on the keyboard. | stdio.h                 |
| 2.            | putchar()               | outputs a single character to the screen.         | stdio.h                 |
| 3.            | printf()                | outputs a single character to the screen.         | stdio.h                 |
| 4.            | scanf()                 | returns the next character typed on the keyboard. | stdio.h                 |
| 5.            | isdigit()               | returns non-0 if arg is digit 0 to 9              | ctype.h                 |
| 6.            | isalpha()               | returns non-0 if arg is a letter of the alphabet  | ctype.h                 |
| 7.            | isalnum()               | returns non-0 if arg is a letter or digit         | ctype.h                 |
| 8.            | islower()               | returns non-0 if arg is lowercase letter          | ctype.h                 |
| 9.            | isupper()               | returns non-0 if arg is uppercase letter          | ctype.h                 |
| 10.           | acos()                  | returns arc cosine of arg                         | math.h                  |
| 11.           | sqrt()                  | returns square root of num                        | math.h                  |
| 12.           | fabs()                  | returns absolute value of num                     | math.h                  |

|     |        |                              |        |
|-----|--------|------------------------------|--------|
| 13. | asin() | returns arc sine of arg      | math.h |
| 14. | atan() | returns arc tangent of arg   | math.h |
| 15. | cos()  | returns cosine of arg        | math.h |
| 16. | exp()  | returns natural logarithim e | math.h |

## 2. User Defined Functions

All type of library functions is not presented in the language which fulfil the requirement of the user. Many times it is required that we have to execute group of statement number of times and this type of function is not available in the library. Then we are required User Define Functions. The functions, which are written according to the requirement of the user, are called Use Defined Function (UDF). In a program, one or more user define function can be made. The User define function can be written before or after the main( ) function.

### 2.3.4 Declaration of Function and Definition a function

In the 'C' language, a user define function can be define as follows :

```
Return type Function Name (List of arguments)
{
 Local Variabals
 function body
 Expressions
}
```

Function name is similar to an identifier. When a function calls, it executes according to the statements are written in the function and it returns a value. This value must have a data type. So, at the time of declaration, we define the return value type which tell us the type of value to be return by the return expression. if we are not defined the return type value then it return integer type value. A function can use more than one return statement and it is not necessary to write the return statement at the end of the function, it can be written anywhere in the function according to the requirement.

**Program 5 :** Write a 'C' program to find out smaller number out of two numbers using function.

```
#include<stdio.h>
```

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

```

/* It is C program to illustrate Function*/

/* It is use to find biggest number between two numbers*/

main() /* Main function */
{
 int A,B,big; /*Local variable for main function*/

 printf("\n Enter two number :");

 scanf("%d %d",&A,&B);

 big = fun_big(A,B); /*Function Call By Value*/

 printf("The biggest number : %d",big);

 getch();
}

/* Function return an integer value */

int fun_big(int C,int D)
{
 if(C > D)
 return(C);
 else
 return(D);
}

```

Result :

Enter two number : 45 85

The biggest number : 85

Enter two number : 105 85

The biggest number : 105

First, it executes main( ) function. When the control flow executes call statement (fun\_big()) in the main( ) function then it will execute the statements which are written in it. When the control transfers from main( ) function to user define function then the values written with the name of the function in the bracket also copied in the variables of the user define function.

For example, when it complete the execution of the function then the control return back to the main( ) function and executes the remaining statements in the main( ) function.

The function where we call the user define function are called "Calling Function" and the user define function which calls are called "Called Function".

### **2.3.5 Actual Parameters and Formal/Dummy Parameters**

When we call user define function in the main( ) function then the values are written with the name of function in the bracket are called actual parameter. The actual parameter can be variables, constants, and addresses.

At the time of function definition the arguments are written in bracket, are called Formal/Dummy Parameter.

When a user define function call then the actual parameters will be copied in the formal parameters. It is remembered at the time of calling a function that the number of actual parameters must be equal to the formal parameters.

Example :

```
int a ;

main ()
{
 int x, y;
 int z;
 printf("Enter the first number\n");
 scanf("%d",&x);
 printf("Enter the second number\n");
 scanf("%d",&y);
 z = mul (x,y); /*Actual parameters (x, y)*/
```



```

printf("%d\n",z);
}
int mul(int a, int b)/* Formal parameter*/
{
int c;
c = a * b;
return(c) ;
}

```

The values at that time are  $a = x$  and  $b = y$ .

### 2.3.6 Calling a Function

We call a function in the `main()` function by its name. It should be remembered that the function name would be written right side of the expression. There are two methods to call a function.

1. Call by Value
2. Call by Address

#### 1. Call by Value

When we pass the variable or constants as arguments at the time of calling to the function then this type of calling is called 'call by value'.

Example :

```

main ()
{
int mark1,mark2,mark3;
float d;
printf("Enter the first marks\n");
scanf("%d",&mark1);

```

```

printf("Enter the second marks\n");
scanf("%d",&mark2);
printf("Enter the third marks\n");
scanf("%d",&marks3);
d=average(mark1, mark2, mark3); /*function call*/
printf("Average of three marks is %f\n",d);
}
float average(int x, int y, int z)
{
float f;
f = (x + y + z)/3;
return (f);
}

```

In this example, it is sending variables at the time of function calling.

## 2. Call by Reference

At the time of function calling, we pass the addresses of the variable in place of variables or constants then it is called 'Call by Reference'. Here, we have to remember that the formal argument must be a pointer type variable to store the address of the actual parameters. A simple variable cannot store the address of a variable.

Example :

```

main ()
{
int a,b;
float d;
printf("Enter the first number\n");
scanf("%d",&a);

```

```

printf("Enter the second number\n");

scanf("%d",&b);

d=average(&a,&b);

printf("Average of two number is %f\n",d);
}

float average(int*x, int *y)
{
float f;

f = (*x + *y)/2

return (f)
}

```

Here, the address of a and b will pass to the function through avg(&a,&b).

Let 1000 and 1050 are addresses, where the address of x and y will be 1000 and 1050 respectively. If we write int x, int y in place of int \*x, int \*y then it will be wrong because simple variables x and y can not store the addresses.

### 2.3.7 Categories of Functions

Functions are divided into three different categories according to their way of passing arguments to the function.

1. Function with No Arguments and No Return Values
2. Function with Arguments but No Return Values
3. Function with Arguments and Return Value

#### 1. Function with No Arguments and No Return Values :

In this types of functions, we do not pass any argument and the function does not return any value. In this type of category, the control go and come from main( ) function to user define function and user define function to main( ) function. They have no data with them. This type of category is used to transfer the control from one place to the other place.

Example :

```
main ()
{
 printf("You are in Main program\n");
 average();
}
void average()
{
 float f;
 printf("Enter the first marks\n");
 scanf("%d",&x);
 printf("Enter the second marks\n");
 scanf("%d",&y);
 printf("Enter the third marks\n");
 scanf("%d",&z);
 f = (x + y + z)/3;
 printf("Average of three marks is %f\n",d);
}
```

## 2. Function with Argument But No Return Values

In this category, we send the arguments at the time of calling from the main( ) function but function does not return any value. We read the data in the main( ) function and send them to the user define function later.

In this category, we have to remember that when the arguments pass from the main( ) function then the data type of the actual arguments should be same to the formal arguments. All actual arguments will be copied into the formal parameters one by one.

Example :

```

main ()
{
 int mark1,mark2,mark3;

 float d;

 printf("Enter the first marks\n");

 scanf("%d",&mark1);

 printf(Enter the second marks\n");

 scanf("%d",&mark2);

 printf("Enter the third marks\n");

 scanf("%d",&marks3);

 average(a,b,c);
}

void average(int x, int y, int z)
{
 float f;

 f = (x + y + z)/3;

 printf("Average of three marks is %f\n",f);
}

```

Whenever, the function average( ) are called then control transfer to the user define function and the values of a and b will be copied into the variable x and y (x = a and y = b) respectively. When the user define function is completed then control will return to the main( ) function without return a value after completing its execution.

### **3. Function with Arguments and Return Value**

In this category, the actual arguments are passed and stored in the formal arguments at the time of the calling of user define function. When the execution of the user define function completed then control return to the main( ) function with a return value. When all values of user define function will be used later in the main( ) function then this cat-

egory will be used. Here, the return statement is used to get a value from user define function. The return statement returns a value at a time. So, we can say that whenever we call a function then it returns a single value on each call. The return statement must write here and it return a float value automatically.

Example :

```
main ()
{
 int mark1,mark2,mark3;

 float d;

 printf("Enter the first marks\n");
 scanf("%d",&mark1);
 printf("Enter the second marks\n");
 scanf("%d",&mark2);
 printf("Enter the third marks\n");
 scanf("%d",&marks3);
 d=average(marks1, marks2, marks3);
 printf("Average of three marks is %f\n",d);
}

float average(int x, int y, int z)
{
 float f;
 f = (x + y + z)/3;
 return (f);
}
```

## 2.4 Scope rules

We are using the variable in the function, where are these variables can be used

in the program? It is decided by the scope of rules. The variables used in the program are two types.

1. Local variable
2. Global variable

### 1. Local variable

These variables are declared in the function. The variables declared in the header of the function also called local variables. The boundary of the local variable is the boundary of function. So, the local variable works within the function only.

Example :

```
#include<stdio.h>

main()
{
 int a=5, b=20; /*Local variables*/
 printf("%d", b);
 fun();
}

fun()
{
 int b=10; /* Local variable*/
 printf("%d", b);
 /*printf("%d", a); uncommented cause and error*/
}
```

output:

20 10

Here, three variables are declared. Variable a and b are declared in main( ) function and variable b is declared in fun( ) function. Both b are different from each other.

The function can be used within the boundary of the function where it is declared. For example, variable a cannot be used within the function fun().

## 2. Global Variable

These variables are declared at fixed place of the program and they are visible in whole program. The global variables are declared after the header files.

Example :

```
#include<stdio.h>

int a, b, c;/* Global declaration of vaiable*/

main()
{
 a=10; b=20; c=30;

 fun1();

 printf("a=%d b=%d c=%d", a , b, c);
}

funl()
{
 printf("\na=%d b=%d c=%d", a, b, c);

 a + = 5;

 c + = 10;

 return;
}
```

output:

a=10 b=20 c=30

a=15 b=20 c=40

The value of a, b and c can be changed in any function in the program. Here, it is remembered that if a variable declares as a global and local variable with a same name



then the local variable will be used in the function and global variable will not use in the function. The global variable will be used where local variable is not declared with the same name.

Example :

```
#include<stdio.h>

int a=10, b=20;

main()
{
 a=20; c=30;

 printf("a=%d b=%d c=%d",a, b,c);
}
```

output:

a=20, b=20, c=30

Here, it prints the value of local variable 'a' because the precedence of local variable is higher than global variable.

## 2.5 Storage classes type

There is two ways to characterize variables.

1. Data Type
2. Storage class

Data type refer to the type of data stores in the variable and storage class refer to the scope and life time of the variable with in the program.

There are four different type of storage class :

- (i) automatic
- (ii) external
- (iii) static and

(iv) register

They are declared by keyword auto, extern, static and register respectively.

Example :

```
auto int x, y, z;

extern float r1, r2;

register char ch1;

static int c, d;
```

The storage class keyword is specified a particular storage class must be placed at the beginning of the variable declaration. We have already read about local and global in the paragraph 3.4 of this chapter.

### 2.5.1 Automatic variables

Automatic variables are declared within the function and these are local variable to the function. Their scope is within the function only and its lifetime starts with the function calling and ends with the termination of the function. A variable can be declared in the different function with the same name. They are independent to each other. The automatic variables are required a keyword auto for declaration of the variables.

Example :

```
auto int A;
```

If the storage class is not written in the declaration statement then it makes the variable of automatic type (Default storage class is Automatic).

Example :

```
#include<stdio.h>

#include<conio.h>

main()
{

 /*Function main() scope start here*/

 auto int total1=10;
```

```

 auto int total2=30;

 printf("\ntotal1 = %d",total1);

 fun();

 printf("\ntotal1 = %d",total1);
}/*Function main() scope End here*/

fun()
{
 /*Function fun() scope start here*/

 auto int total1=30;

 printf("\ntotal1 = %d",total1);

 /*printf("total2 = %d",total2);

 uncommented to get an error*/

}/*Function fun() scope End here*/

```

Result :

```

 total1 = 10

 total1 = 30

 total1 = 10

```

In the above program, the variable total1 is declared in both function main( ) and fun( ). Both total1 are independent to each other and the value of total1 of main( ) function will not be changed before or after the calling of function fun( ) and the variable total2 can not be used within the function fun( ). It can be used within the main( ) function only.

### 2.5.2 External variable

External variables are not related to a single function as automatic variables. The external variable is declared in a function, all function can be accessed the variable which are written after the declared function. Their scope extends from point of declaration through the remainder of the program. They are using as the global variables. When we call a function, where it declares then its lifetime starts and end with the termination of the program. All function can be used the variable after declaration it. The decalaration of

variable uses extern keyword.

Example :

```
extern int A, B, C;
```

```
extern char D, E;
```

### 2.5.3 Static variables

The Static variables are also declared in the functions and therefore have the same scope as automatic variables. The Static variables are local to the function but they retain their value through out the life of the program. The first call of the function (where the static variable declares) declares the static variables and it initialize by zero if it is not initialized. It is lifetime start with the first execution of the function and died with the termination of the program. The static variables retain their last value and will use when it calls again. The declaration of variable uses static keyword.

Example :

```
#include<stdio.h>
```

```
main()
```

```
{
```

```
 int I;
```

```
 for (I = 1; I <= 10; I ++);
```

```
 fun();
```

```
}
```

```
void fun();
```

```
{
```

```
 static int K;
```

```
 printf("%d", K);
```

```
 K++;
```

```
}
```

The function fun( ) is called ten times in main() function by a loop. When we call

the function first time the initial value of K will be zero (i.e.  $K = 0$ ). And it print the value of K is zero and value of K will be increased by one. When the function is called second time then the last value of K will print (in first call) 1 and the value of K will be increased by one. When the function is called third time then the last value of K will print (in second call) 2 and the value of K will be increased by one. And it will prints ten values of the variable K. The output will be as follows :

0 1 2 3 4 5 6 7 8 9

### 2.5.4 Register variable

In the above three types of variables are stored at the memory address. But the register storage class variables store the value in the CPU registers. Mostly, those variables are declared which are used extensively in the program. This is helpful to reduce the execution time of the program.

Example :

```
register int A, B, C;
```

```
register cahr x, y, z;
```

These variables are local to the function and work as the automatic variables. Moreover declaring certain variables to be register variables does not guarantee that they will actually be treated as register variables. The declaration will be valid only if the requested register space is available if it is not so, then the variable will be treated as automatic variable. The summary of storage classes as shown in Table 2.3.

**Table 2.3**

| Class Type       | Storage Space | Default initial Value | Scope                                                   | Life Time                                                                      |
|------------------|---------------|-----------------------|---------------------------------------------------------|--------------------------------------------------------------------------------|
| Automatic Memory |               | Garbage Value         | Local to the function in which the Variable is defined. | Till the control remains within the function in which the variable is defined. |
| External Memory  |               | Zero                  | At the decla-                                           | After declaration                                                              |

|          |          |         |              |                    |
|----------|----------|---------|--------------|--------------------|
|          |          |         | ration point | to termination of  |
|          |          |         | to entire    | the program.       |
|          |          |         | program.     |                    |
| Static   | Memory   | Zero    | Local to the | Value of the       |
|          |          |         | function in  | variable persist   |
|          |          |         | which the    | between different  |
|          |          |         | Variable is  | function call.     |
|          |          |         | defined.     |                    |
| Register | CPU      | Garbage | Local to the | Till the control   |
|          | register | Value   | function in  | remains within     |
|          |          |         | which the    | the function in    |
|          |          |         | Variable is  | which the variable |
|          |          |         | defined.     | is defined.        |

---

## 2.6 Arrays and Functions

If we want to pass an array from main() function to user define function then we have to take name of the array as an argument. And also take another argument as refer to the size of the array.

Example :

```
average (a,n);
```

Here, average is the name of user define function. a and n are name and size of an array. The function will de declared as follows :

```
float average (float arr[], int n1);
```

When, average(a,n) statement executes in the main() function then the control will transfer from main() function to user define function with its arguments. This will be float arr [] = a and n1 = n.

The bracket ( [ ] ) with the name of an array arr show that arr is an array and n1 is the size of the array. Here, it is not required to write the size of an array in the bracket because it would decide by the actual parameters.

Example :

```
float average (float arr [],int n1);

main()
{
 float a[40],r ;
 int n;
 printf("Enter the size of the array\n");
 scanf("%d",&n);
 printf("Enter the element of the array\n");
 for(i=0;i<n;i++)
 {
 scanf("%d",&a[i]);
 }
 r=average(a,n);
 printf("The average of the element is%f",r);
}

float average(float arr[], int n1);
{
 int i, sum = 1;
 float ave;
 for(i=0;i<n1;i++)
 {
```

```

 sum = sum + arr[i];
 }
 ave = sum/n1;
 return(ave);
}

```

In the above example, we suppose that the maximum size of the array is 40. But we store 10 ( $n = 10$ ) values in the array. When `average(a,n)` executes in `main( )` function and its mean `average(a,10)` then control will transfer with both values to executes the function. Whenever, control goes to the function then it stores all values of array `a[ ]` into the array `arr[ ]` and value of `n1` equal to `n`. Then the function will execute according to the values and result will return to the `main( )` function

## 2.7 Recursion

When a function calls itself until some specified condition has been satisfied, this processing called recursion. To write a function recursively then we have to remember two things :

1. The function must be call itself.
2. The function must have a condition which stop the recursion on satisfaction (Termination condition).

Generally, functions are called in the another function. The `main( )` function is exception, it cannot be called anywhere in the program. The facilities available in many programming languages that a function calls itself, this is called recursive function and process is called recursion. A function calls a other function than the program control return back to the calling function. But function calls it self then it will come to the same function. It is dangerous that it may not be terminated from the function. In these conditions, program will run and if there is no resources available for computer it terminates abnormally and control will out from the program. It should be remembered that any function must have a termination condition. Otherwise it will cause an error message.

**Program 6 :** Write a function to findout the factorial of a given number using recursion function.

```
#include<stdio.h>
```

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



```
/* It is C program to calculate the factorial of N*/
```

```
main() /* Main function */
```

```
{
```

```
 int N,f1; /*Local variable for main function*/
```

```
 int fact(int N); /* Function Prototype*/
```

```
 printf("\n Enter a number :");
```

```
 scanf("%d",&N);
```

```
 f1 = fact(N); /* Function Call By Value*/
```

```
 printf("The Factorial of %d is = %d",N,f1);
```

```
 getch();
```

```
}
```

```
/* Function return an integer value */
```

```
int fact(int K) /* Function Defination */
```

```
{
```

```
 int i,f2=1; /*Local variable for fact function*/
```

```
 for(i=1;i<=K;i++)
```

```
 f2 = f2 * i;
```

```
 return(f2);
```

```
}
```

Result :

Enter a number : 5

The Factorial of 5 is = 120

Enter a number : 7

The Factorial of 7 is = 5040

Let number = 5

In the first term it return(5 \* fact(4))

In the second term it return (5 \* 4 \* fact(3))

In the third term it return (5 \* 4 \* 3 \* fact(2))

In the fourth term it return (5 \* 4 \* 3 \* 2 \* fact(1))

In the fifth and last term it return (5 \* 4 \* 3 \* 2 \* 1)

**Program 7 :** Similarly, we also understand the recursion using another recursive function.

Let findout the value of  $x^y$  using successive multiplication.

```
int power(int a, int b);

main()
{
 int x,y,z;

 printf("Enter the Number\n");

 scanf("%d\n",&x);

 printf("Enter the power\n");

 scanf("%d\n",&y);

 z = power(x,y);

 printf("The result is\n");

 printf("%d\n",z);
}

int power(int a, int b)
{
 if (b==1)

 return a;
```

```

else
 return(a*power(a,b-1));
}

```

**Program 8 :** Write a program to findout the n<sup>th</sup> fibbonacci number. #include<stdio.h>

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

```
/* It is C program to calculate Nth Fibonacci Number*/
```

```
/* Fibonacci Number Start with 1,1,2,3,5,8,13,21,34,55*/
```

```
main()
```

```

{
 int N,f1;

 int fib(int N);/* Function Prototype*/

 printf("\nEnter a number :");

 scanf("%d",&N);

 f1 = fib(N); /* Function Call By Value*/

 printf("The Nth Fibonacci Number is = %d",f1);

 getch();
}

/* Function return an integer value */

int fib(int K) /* Function Definition */

{
 int i,f1=1,f2=1,f3; /*Local variable for fact function*/

 for(i=1;i<=K-2;i++)

 {
 f3 = f1 + f2;

```

```

 f1 = f2;

 f2 = f3;

 }

 return(f3);

}

```

Result :

Enter a number : 5

The Nth Fibonacci Number is = 5

Enter a number : 10

The Nth Fibonacci Number is = 55

**Program 9 :** Write a function to solve the quadratic equation  $ax^2+bx+c=0$ .

```

#include<stdio.h>

#include<conio.h>

#include<math.h>

main()

{

 void Q_EQ(int p,int q,int r);

 int a,b,c;

 printf("Enter the value of a,b and c :");

 scanf("%d %d %d",&a,&b,&c);

 Q_EQ(a,b,c);

 getch();

}

void Q_EQ(int p,int q,int r)

```

```

{
 float r1,r2,D;
 D = q * q - 4.0 * p * r;
 if(D < 0)
 printf("Roots are imaginary");
 else
 {
 if(D == 0)
 {
 printf("Roots are Equals\n");
 r1 = r2 = - q / (2.0 * p);
 }
 else
 {
 r1 = (- q - sqrt(D)) / (2.0 * p);
 r2 = (- q + sqrt(D)) / (2.0 * p);
 }
 printf("\nRoots are : %f %f", r1,r2);
 }
}

```

Program 10 : Write a function 'prime' which return 1 if the argument is prime otherwise it return 0.

```
#include<stdio.h>
```

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

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

```

main()
{
 int prime(int p);
 int flag,N;
 printf("\nEnter the value of N :");
 scanf("%d",&N);
 flag = prime(N);
 if(flag == 1)
 printf("\nNumber is prime");
 else
 printf("\nNumber is not prime");
 getch();
}

int prime(int p)
{
 int i,r1;
 r1 = (int)sqrt(p);
 for(i=2; i<=r1; i++)
 if(p % i == 0)
 return(0);
 return(1);
}

```

## 2.8 Pointer

Pointer is a powerful tool. Every variable stores in the primary memory that is called the address of the variable. Every variable has a memory address.

A pointer is a variable that stores the address of the data item (Address of the data in memory). The pointers are used to make the good and complex program in the 'C' language.

Every variable allocates the memory according to the type of the data. The char type data stores in 1 Byte, int type data store in 2 Bytes and float type data store in 4 Bytes. There is one memory address for a group of Bytes and that is stored in the pointer variable associate with the group. For example : int type data stores in 2 Bytes and there addresses are FAB and FAC respectively the address of the variable will be FAB. This is the address of first memory cell.

Let d is a variable, which represent integer data type. The compiler allocates the memory according to the type of data where it stores its value. It can easily accessible. Let we store value 10 in the variable d.

K is a pointer type variable, it can store address only. So that , K stores the address of the memory 4500 (Address of variable d).

There are two operators perform the above work in the 'C' language. The \* operator is used to declare a pointer variable and it is also used to retrieve the value from the address stored in the pointer variable. This is also called value operator and the & operator is used to retrieve the memory address of any variable. For example we want to know the address of the variable d then we have to write &d. So, we write the following statement to store the address of d in pointer variable.

$$K = \&d;$$

According to the above description \*K and d shows the same value.

**Program 1 :** Write a program to add five integer number using pointers.

```
#include<stdio.h>

#include<conio.h>

#include<alloc.h>

main()

{

 int I, *A,sum=0, B;

 A = &B;
```

```

for(I=1; I <+ = 5, I++)
{
 printf("Enter a number:");
 scanf("%d", A);
/*don't use ampersand sign with pointer variable.*/
 sum=sum + *A;
}
printf("\nSum of Five Numbers = %d', sum);
getch();
}

```

### 2.8.1 Pointer Expression

The pointer expression is the expressions, which uses pointer variables. If P1 and P2 are two pointer variables. They are declared and initialized.

#### 1. Assignment Expressions :

These expressions are used to assigned values, are called assignment expressions.

Example :

```

P2++;
--P1;
P1 = P1 + 1;
*P2 = *p2 + 1

```

First three expressions are used to increase or decrease the memory address stored in pointer variables. And fourth expression increases the value stored at memory location stored in variable P2.

#### 2. Arithmetic Expressions :

```

P2 = P2 + 1;

```



$P2 = P1 + 1;$

In the pointer variable, a memory location can be increase or decrease. The following arithmetic operations are not valid on pointers :

$P1 / P2$  Two pointer can be subtract if they are indicate same type

$P1 / 50$  of elements. Generally, two pointers can subtract when

$P1 + P2$  only they are indicated the elements of an array.

$P1 * P2$

$P1 * 17$

### 3. Relational Expression

The relational operators are used on pointer variables.

Example :

$P2 > P1$

$P1 == P2$

$P2 >= P1$

$P1 != P2$

#### 2.8.2 Advantage of pointers

1. The pointers efficiently control an array.
2. The pointer can pass the information to the function, it automatically change the value of variables which are passed by the calling statement.
3. They speedup the execution of the program.
4. Pointer also provides an alternative way to access individual array elements.
5. The variable outside the function can be accessed with the help of pointers.

### 2.9 Dynamic Memory Allocation

Here, the memory can be assigned two types :

- (1) Static memory assignment

## (2) Dynamic memory assignment

1. Static memory is assigned before the execution of the program. It is used in array and it cannot be changed at the time of execution.
2. Dynamic memory can be assigned the memory after the execution of the program. It is assigned according to requirement of the programmer. The function available in 'C' are used to assign dynamic memory are shown in the table 4.1

Table 2.4

| Function       | Return value | Purpose                                                                                                      |
|----------------|--------------|--------------------------------------------------------------------------------------------------------------|
| malloc (size)  | Pointer void | Allocate required size of block.                                                                             |
| calloc (size)  | Pointer void | Allocate required size of block which divided into equal size of parts and each part is initialized by zero. |
| free(pointer)  | No value     | Free previous allocate memory.                                                                               |
| realloc (size) | Pointer void | Modify the size of memory.                                                                                   |

### **malloc( ) Function**

The block of memory may be allocated using the function malloc(). The malloc() function is allocated a block of specified size and its type will be void pointer, we can change the void type memory in another data type.

The malloc function can be return as follows :

```
Str = malloc (size in byte)
```

If we change the memory type which is allotted by malloc function (if str stores the address of int type data) then malloc will be written as follows :

```
Str=(int*)malloc(size in byte)
```

Example :

```
int *integer;

integer=(int *) malloc (sizeof(int) * 10);
```

The above statement allocates 20 Bytes memory block. A integer number stores in 2 Bytes and return a integer type pointer.

Example :

```
char *cptr;
cptr=(char *) malloc(20);
```

It is also allocate 20 Bytes and return char type pointer.

### **calloc( ) Fuction**

The memory can be allocated by the calloc() function. It divided the memory in the same size of blocks. It is written as follows :

```
Str = (type cast*) calloc(n, sizeof(element));
```

The above statement allocate configures space for same size of n blocks with same size and each block store zero value.

Example :

```
int *cptr,
cptr = (int *) calloc(10, size of (int));
```

It allocate 10 blocks of 2 Bytes and each block stores value zero.

### **Releasing Memory**

When we does not use the memory block or no longer need the memory in the program then we may release that block of memory for future use to allocate memory of any other data. It uses free( ) function for the above purpose. It is written as follows :

```
free(Str);
```

Str is a pointer which had assigned the memory by the function calloc( ) or malloc( ).

### **Altering the size of memory block**

The realloc( ) function can increase or decrease the memory block.

Example :

Allocate memory by function malloc( )

```
Str = malloc(size)
```

Then its size can be increased or decreased by the function `realloc()`.

```
Str=realloc(ptr, new_size)
```

The memory will be allocated of new size.

Note : if there is not enough memory to allocate then `malloc()` or `calloc()` function return the NULL value.

## 2.10 Pointer and Array

We have read that the space for array is reserved at the time of declaration of the array. In this space we can store the elements of the array. The base address of the array is the address of the first element of the array.

Let an array declares as follows :

```
int A[5] = {7, 9, 15, 12, 14};
```

It is also assumed that the base address of the array is 2000. Because integer data, stores in 2 Bytes. So, the five elements of the array will store as follows :

We have known that the name of an array also shows the base address of the array. So, `A` and `&A[0]` shows the base address of an array and elements of an array stores in the continuous memory location.

Similarly, pointer variable also stores the base address of the memory block. So, an array and a pointer variable can be interchanged.

Example :

```
int*ptr;
int A[5] = {7, 9, 15, 12, 14};
ptr = A /* ptr = &A[0]*/
```

In the above example, `ptr` and `A` both are worked as an array. Both can be used as pointer notation or traditional way.

For example the value of `ptr[1]` is 9 and the value of `*(A + 1)` will also be 9. It can be understood by the following example.

### Program 12

```
#include<stdio.h
```

```

#include<conio.h>

main()
{
 int *ptr,I;

 int A[5] = {7, 9, 1, 2, 4};

 ptr = A;

 clrscr();

 for(I=0;I<+5;I++)
 {
 printf("\nI = %d ptr[I] = %d",I,ptr[I]);

 printf(" *(A+I) = %d &ptr[I] = %x",*(A+I),&ptr[I]);

 printf(" (A+I) = %x",ptr[I]);
 }

 getch();
}

```

The output will be following :

|       |            |            |                |           |
|-------|------------|------------|----------------|-----------|
| I = 0 | ptr[I] = 7 | *(A+I) = 7 | &ptr[I] = ffca | (A+I) = 7 |
| I = 1 | ptr[I] = 9 | *(A+I) = 9 | &ptr[I] = ffcc | (A+I) = 9 |
| I = 2 | ptr[I] = 1 | *(A+I) = 1 | &ptr[I] = ffce | (A+I) = 1 |
| I = 3 | ptr[I] = 2 | *(A+I) = 2 | &ptr[I] = ffd0 | (A+I) = 2 |
| I = 4 | ptr[I] = 4 | *(A+I) = 4 | &ptr[I] = ffd2 | (A+I) = 4 |

In the above program, the pointer variable is written in form of a traditional array and an array variable is written in form of pointer type. Which is shown that a one dimensional array can be interchange with pointer variable. Here, it is necessary to remember that if we add 1 in the pointer variable then it increases 2 Bytes. It is depended upon the data stored in the variable. If it is long int then we add 1 in the pointer variable

then it increases four Bytes.

**Program 13 :** Write a 'C' program to arrange 10 integer numbers using selection sort.

```
#include<stdio.h>

#include<conio.h>

#include<alloc.h>

#include<string.h>

/*It is C program to sort 10 Numbers. */

int main()

{

 int *A;

 /* Array A declared as pointer type.*/

 int i,j,temp;

 A = (int *)malloc(20);

 printf("\nEnter Ten Numbers :\n");

 for(i=0;i<+10;i++)

 scanf("%d",(A+i));

 /* Sorting Algorithm */

 for(i=0;i<+9;i++)

 for(j=i+1;j<+10;j++)

 if(*(A+i) > *(A+j))

 {

 temp = *(A+i);

 *(A+i) = *(A+j);

 *(A+j) = temp;

 }

}
```

```

 }
printf("\nThe Sorted Numbers are :\n");
for(i = 0; i<+10;i++)
 printf("%d\n", *(A+i));
getch();
return 0;
}

```

Result:

Enter Ten Numbers :

20 10 40 60 50 30 80 70 90 75

The Sorted Numbers are :

10 20 30 40 50 60 70 75 80 90

**Program 14 :** Write a program to add two strings using pointers.

```

#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<string.h>
main()
{
 char *s1,*s2,*s3;
 int n1,i,j;
 /*Memory Allocation */
 printf("\nEnter the length of first string : ");
 scanf("%d",&n1);

```

```

s1 = (char *)malloc(n1+1);

printf("\nEnter the length of second string : ");

scanf("%d",&n1);

s2 = (char *)malloc(n1+1);

s3 = (char *)malloc(strlen(s1) + strlen(s2) -1);

printf("\nEnter Two strings :\n");

scanf("%s %s",s1,s2);

/* Algorithm */

strcpy(s3,s1);

i=strlen(s3);

j=0;

while(s2[j] != '\0')

s3[i++] = s2[j++];

s3[i] = '\0';

printf("Strings are : \n %s %s %s",s1,s2,s3);

getch();

}

```

## 2.11 Structures

We have already read about an array which has similar types of elements. But the different type of data can be written in the structure. So, a structure can use different types of data int, float, double, char and array according to a user. Every element in the structure is called a member. If we have to store Name, Basic, Address and HRA of 40 employees then we have read that array is the best method. But Name, Basic, Address and HRA are stored in different arrays. It can be made easy to use structures. It declares a structure named Employee and their members will be Name, Basic, address and HRA. We can store the information of 40 employees to make an array of the structure.

### 2.11.1 Declaration of Structure :



The structure declaration is more difficult than an array declaration. Because every element have to be declared in the structure. The general term, the composition of structure may be define as follows:

```
struct tag
{
 <data type> member1;
 <data type> member2, member3;

 <data type> membern;
}
```

In the above declaration, struct is a keyword, tag is a name that identifies structures of this type. And member1, member2, member3 .....member n are an individual member declaration. The individual members can be ordinary variables, pointers, array or other structures. The same type of members can be specified in a single statement. It is declared in the above example.

```
<datatype> member2, member3;
```

Example :

Define an employee structure and the members are as following:

```
name, basic, address, HRA
```

```
struct employee
{
 char name[20];
 int basic;
 char address[25];
 int HRA;
```

```
};
```

The above structure employee is called the user define structure. Now we can declare multiple copies of employee structure.

Example :

```
struct employee emp1, emp2;
```

The emp1 and emp2 are similar to the structure employee. The above two statements can be combined in a single statement.

Example :

```
struct tag
{
 <data type> member 1
 <data type> member2

 data type member n;
} variable1, variable2;
```

In this case, it is not necessary to write tag in the declaration. The above statement is written as follows.

```
struct employee
{
 char name[20];
 int basic;
 char address[25];
 int HRA;
} emp1, emp2;
```

In the above example, we are used simple variables and array variables.

Example :

```
struct employee
{
 char name[20];
 int basic;
 char address[25];
 int HRA;
}emp1, emp2;
```

We can declare a structure member with in a structure.

Example :

```
struct date
{
 int day;
 int month;
 int year;
};

struct employee
{
 char name[20];
 int basic;
 char address[25];
 int HRA;
 struct date DOB;
```

```
}emp1;
```

In the above example DOB member is a date type structure. This is written with in the employee structure.

We can initialize a structure when it declares.

Example :

```
struct employee emp1 = {"Sangeeta Gupta", 8000, "153, Arya Nagar",
1200, 25, 01, 1976};
```

First value will be stored in the first member of structure, second value will be stored in the second member of the structure and so on.

### 2.11.2 Memory Map

The different memory block will be allocated for different members continuously in the memory. The size of the structure is equal to the sum of the each member size. The memory map of the employee structure is as follows :

The memory will use in the structure :

$20 + 2 + 25 + 2 + 2 + 2 + 2 = 55$  bytes

The size of structure can be calculated by a sizeof operator.

Example :

```
sizeof(employee);
```

output : 55

### 2.11.3 Processing a Structure

The member of structure is process individually. The structure member can be accessed by writing following statement :

```
<Name of structure> l <Member of structure>
```

It uses the . (dot) operator. Please see the following example :

```
struct employee
{

char name[20];
```

```

 int basic;

 char address[25];

 int HRA;

 struct date DOB;

 }emp1;

```

The member of emp1 will be accessed as follows.

```

emp1.basic = 8000

emp1.HRA = 1200

strcpy(emp1.name, "Yashika");

```

Similarly, we can read and write all members of the structure. F o r  
 reading of the member of the structure :

```

scanf("%d", &emp1.basic);

scanf("%d", &emp1.HRA);

scanf("%s", emp1.name);

```

For writing of the member of the structure :

```

printf("%d", emp1.basic);

printf("%d", emp1.HRA);

printf("%s", emp1.name);

```

If a structure is declared with in a structure then the . (dot) operator is used to access the sub member of the structure. It is written as follows :

```

<Name of struct> | <Member of struct> | <Sub-member of struct>

```

In the above example, DOB member declares as date type structure which is declared as follows.

```

struct date
{

```

```

 int day;

 int month;

 int year;

};

```

The member of DOB access as follows :

```

emp1.DOB.day= 25

emp1.DOB.Month = 01

emp1.DOB.Year = 1976,

```

If we declare two variables of a structure then we can store the value of a variable into the another variable using a assignment operator.

Example :

```

struct student emp1={"Sunil Methi", 12000, "Alwar", 1200, 24, 11, 1968},
emp2;

```

If we use the assignment operator then the statement will be written as follows :

```

emp2 = emp1;

```

The value of member of emp1 will be copied into the member of emp2. There is no need to copy from member to member for the same structure.

## 2.12 Structure and Array

We can declare the array as the member of the structure. Which is done in the above examples.

We can also declare structure of array. If we want to store the data of 100 employee then we declare a structure of array. It is declared as follows :

```

struct employee
{
 char name[20];

 int basic;

```

```

 char address[25];

 int HRA;

 struct date DOB;

};

struct employee emp1[100];

```

In the above example, emp1 is declared as an array of structure. It is accessed as follows :

Name of structure[subscript Value]Member of structure

Example :

```

emp1[0].basic = 12000;

emp1[1].basic = 8000;

```

### 2.13 Structure and Function

We can pass the member of structure as the simple variable.

Example :

```

Calculate(emp1.basic,emp1.HRA);

```

Basic and HRA are the member of emp1. We can pass the whole structure at a time. For example emp1 is declared as employee structure then

```

Calculate(emp1); /* calling */

```

Example :

```

void Calculate (struct employee Temp)

{

 Statements;

}

```

All member of emp1 will be copied into members of temp.

**Program 15 :** Declare a employee structure and its members are as follows

name, address, basic pay, HRA and DA and store their values and calculate Gross Pay.

```
#include<stdio.h>

#include<conio.h>

main()
{
 struct employee
 {
 char name[20];
 char address[25];
 int basic_pay;
 float da,hra;
 float gross_pay;
 };
 struct employee e1;
 /* Read structure*/
 printf("\nEnter Employee Data :");
 printf("\nName : ");
 gets(e1.name);
 printf("Address :");
 gets(e1.address);
 printf("Basic Pay : ");
 scanf("%d",&e1.basic_pay);
 printf("DA(%) : ");
 scanf("%f",&e1.da);
```



```

printf("HRA(%) : ");
scanf("%f",&e1.hra);

/* Gross Pay Calculation*/

e1.gross_pay = e1.basic_pay + e1.basic_pay * e1.da/100 +
 e1.basic_pay * e1.hra/100;

/* Printing */

printf("\nEmployee Data :");

printf("\nName : %s",e1.name);

printf("\nAddress : %s",e1.address);

printf("\nBasic Pay : %d",e1.basic_pay);

printf("\nDA : %5.2f%",e1.da);

printf("\nHRA : %5.2f%",e1.hra);

printf("\nGross Pay : %8.2f",e1.gross_pay);

getch();

}

```

Results:

```

Enter Employee Data :
Name : Sunil Methi
Address : 153, Arya Nagar, Alwar.
Basic Pay : 9375
DA(%) : 43
HRA(%) : 15
Employee Data :

```

Name : Sunil Methi

Address : 153, Arya Nagar, Alwar.

Basic Pay : 9375

DA : 43.00%

HRA : 15.00%

Gross Pay : 14812.50

**Program 16 :** Write a program to store student name, roll\_no, and marks obtained in four subjects of a class in a structure and print name, roll\_no, total marks obtained by every students.

```
#include<stdio.h>
```

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

```
main()
```

```
{
```

```
 int i,j,tot;
```

```
 struct
```

```
 {
```

```
 char name[20];
```

```
 int roll_no;
```

```
 int sub1,sub2,sub3,sub4;
```

```
 }student[5]; /* In this case tag not required*/
```

```
 /* Read structure*/
```

```
 for(i=0; i<+=4; i++)
```

```
 {
```

```
 printf("\nEnter Student %d Data :",i+1);
```

```
 printf("\nName : ");
```

```

scanf(" %[\n]",student[i].name);

printf("Roll Number :");

scanf("%d",&student[i].roll_no);

printf("Marks Subject 1 : ");

scanf("%d",&student[i].sub1);

printf("Marks Subject 2 : ");

scanf("%d",&student[i].sub2);

printf("Marks Subject 3 : ");

scanf("%d",&student[i].sub3);

printf("Marks Subject 4 : ");

scanf("%d",&student[i].sub4);

}

/* Printing */

for(i=0; i<+=4; i++)

{

printf("\nName : %s",student[i].name);

printf("\nRoll Number : %d",student[i].roll_no);

tot=student[i].sub1+student[i].sub2

+student[i].sub3+student[i].sub4;

printf("\nTotal Marks : %d",tot);

}

getch();

}

```

### **Important Points**

1. Arrays are two types. One dimensional array and two dimensional array.
2. The part of the program, which is group of statements, and known by the separate name, called function.
3. The main ( ) function is also a user define function and it is must be presented in the program.
4. The arguments define at the time of function declaration are called formal /dummy parameter.
5. When a function calls, the arguments written with the calling statement is called actual parameters.
6. The function can call two types :
  - (i) call by value
  - (ii) call by reference
7. When a function calls itself, this process is called recursion.
8. The function must have a condition, which stop the recursion on satisfaction (Termination condition).
9. Pointer stores the address of a simple variable.
10. Dynamic memory can be assigned by the pointer variables.
11. The calloc, malloc are used to allocate the memory.
12. The free function is used to release the memory.
13. We can collect the different types of data using the structure.
14. The member of the structure are accessed by the . (dot) operator.
15. We can make a link list using structure.

## EXERCISE

### Objective Type Questions :

- The group of similar types data are called
  - Array
  - Function
  - String
  - None of these
- How many elements are in the array `float arr[3][2]` :
  - 2
  - 3
  - 6
  - 9
- Which is false statement :
  - Global variable can be used anywhere in the program.
  - Auto variables are declared in the `main()` function.
  - Local variables are not work within the function or block.
  - Local variable can be declared in the different function with the same name.
- Which is the correct statement to call a function `int add(int x)` :
  - `add();`
  - `add(x);`
  - `add(int x);`
  - `int add (int x);`
- How many types a function can call :
  - 2
  - 1
  - 3
  - 4
- How many types of function are :
  - 1
  - 2
  - 4
  - 3
- The value stores in the pointer variable
  - Integer Value
  - Any Value

- (c) Address of the another variable (e) None of these
8. `int B = 10;`  
`int A = &B;`  
`printf("%d", B)` will print.
- (a) 10 (b) Address of variable A  
(c) Address of variable B (d) Print according to program
9. `int*A;`  
`A=(int *)malloc(sizeof(int)*10);`  
`printf("%d",A);` will print.
- (a) Address of variable A (b) Address of first element of an Array  
(c) First value of an array (d) Not print any value
10. The member of a structure can be
- (a) Pointer variable (b) Integer Variable  
(c) Floating variable (d) All of these
11. Which symbol is used to access the member of structures.
- (a) . (dot) (b) \*  
(c) ® (d) &
12. `a=1011`  
`b=1111` and  
`x = a & b` then value of x -
- (a) 10 (b) 11 (c) 12 (d) 13
13. If nay Union have int float and double data type then how much momory for this Union.
- (a) 2 bytes (b) 4 bytes (c) 10 bytes (d) 8 bytes

**Very Short Answer Type Questions :**

1. Where is a return statement written?
2. How many types are arguments?
3. What are the local variables?
4. Which header file includes the '\0' character?
5. How can you declare a one dimensional array?
6. How is declared a pointer variables?
7. `int A;`  
How is print the address of a variable A?
8. How is release the memory by the function `free()`?
9. How is declared a structure the member?
10. How is written the statement to initialize the member of the structure?
11. which file contains the `malloc()` function?
12. How we declare member of Function?
13. How we initialize the member of Function?
14. How we allocate bits for a variable?
15. Which keyword is use to create new data type?

**Short Answer Type Questions :**

1. Which is the local variable?
2. Write a example to declare a two dimensional array.
3. How is a function declares?
4. How many types are functions in the 'C' language?
5. How is a array declared?

6. Write two condition of the recursion.
7. What is pointer variable?
8. `int *A={ 10, 20, 30};`  
If the base address of A is 2000 the write all addresses.
9. How are an array declared by a pointer type variable.
10. What is structure? Explain.
11. Write student structure whose member are name, address and roll\_No.
12. How we declare bit field to store 0 or 1 for male or female?
13. Write an example of .....data.

**Eassy Type Questions :**

1. Write advantages of printers.
2. How with declare Arry by pointer? explain with example.
3. Store 10 name, address and phone number using structure. agange these names according to Alphabat. Write a programe to print the this order.
4. write a programe to read one string and count the frequency of Alphbet.
5. How with declare pointers to the function. explain with example.
6. Write structure for following members showing bit field.
  - i. be male or female
  - ii. have one of the eight different hobbies.
  - iii. be single, married, divorcded or widowed.
7. What is drived data type? how can be written drived data type of 12 months?

**Answer Key**

1. a    2. c    3. b    4. c    5. a    6. b    7. b    8. a    9. b  
 10. d    11. a    12. b    13.d



## Chater-3

# Object Oriented Programming

### Program and Programing

We know that the group of instructions written in order to complete a certain task is called a program and an operation in a computer is called an instruction. Generally, a program can be seen as a logical process in which data is given (input), that data is processed and then, as a result, we get data. To solve a problem by computer, it has to be changed in such a way that computer can understand, this process is called programming. To prepare this, we use some languages, such as Fortran, Pascal, Cobol etc. These languages are procedural languages. By the year 1960-70s, these languages were used very effectively, but by this time the programs were becoming big and complex.

### 3.1 Unstructured programming

While learning programming, we prepare small and simple programs at the initial level, they have one main program. Main program means - a set of instructions in which the data is available throughout the program and those data can be changed. See Figure 3.1

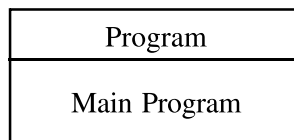


Figure 3.1: Unstructured Programming - program processes data directly

When size of a program increases, this kind of programming leads to a lot of problems. For example, if a group of same instructions has to be re-used, we will have to re-write that group, this unnecessarily will increase the size of program. It is natural to have difficulties to improve such a program. The solution to this problem came out that as, this group of instructions should be taken out from the main program and given a new name, and reference of that name should be used in the main program. Thus, formed the basis of procedural programming, in which other programs can be used in a main program. These programs can be understood as helpful processes, which we call procedures.

### 3.2 Procedural Programming

With the help of procedural programming, the size of program can be reduced, chances of errors are less and, maintenance is also easier. When a procedure is called in the main program, the control of the program goes to that procedure, the computer operates that group of instructions (procedure). After the completion of that group, the control goes back to the instruction written next in the main program. See Figure 3.2

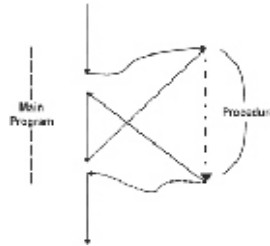


Figure 3.2: Execution of Procedure

In the main program, the control of program transfers to the procedure, instructions written in procedure are executed, control comes to the next statement in the main program.

With the help of procedures, we are able to reduce the size of program, and make it an error-free.

Now, we can assume that one program is a series of many procedures. Each procedure is called from the main program, the data is processed and the results are transmitted to the main program. Thus, when the whole series is completed, the main program gets the final result. The data is transferred from main program in the form of parameters. See figure 3.3

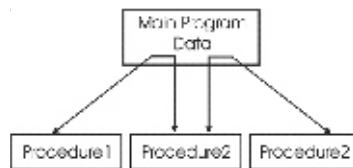


Figure 3.3: Procedural Programming - In the control of main programs - Procedure execution and data exchange.

So now we can say that the main program is divided into small segments which are called procedures. Common types of procedures that can be used in other programs, and likewise, the processors created by the user themselves can also work in another program, it is necessary that the procedures can be stored separately.

### 3.3 Modular Programming

In modular programming, the procedures are collectively stored in groups, which are called modules. In general, the procedures which are meant for similar kind of processing are kept in one module. The purpose is to get all of them at one place. These stored procedures can be used directly in the main program. The main program is divided into very small pieces and data processing is done through the procedures. See figure 3.4

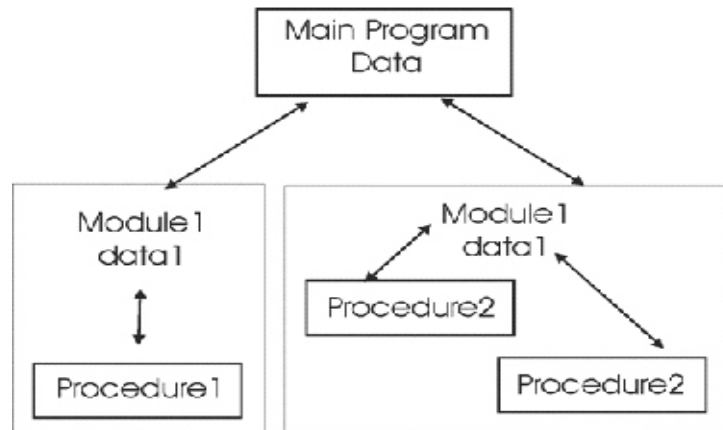


Figure. 3.4 : Modular Programming -Calls of procedures from different modules are coordinated in the main program, appropriate data is handled as parameters.

Each module has its own data. When the procedures of a module are used, each module maintains its internal system of data. But a module can be present in a program only once at a time.

### 3.4 Benefits of Procedural and Modular Programming Languages

- o Very good for general programming
- o Solution code may be available for many problems, so it may not be necessary to repeat coding for the same.
- o As required in case of machine languages, knowledge of hardware is not necessary.
- o Programs written in such languages can be run on another CPU after translating with the help of different compilers.

#### Limitations

- o Availability of many languages, the programmer has to stick to one language, and different languages require different types of expertise.

- o In-depth knowledge of language-specific programming instructions is necessary.
- o For problems requiring artificial intelligence, where logic is fuzzy (logical), procedural languages are not suitable.

### 3.5 Concept of Object-Oriented Programming

Object-Oriented Programming is different from procedural programming. Programs are based on classes and objects in such languages. Objects-classes are used with the help for methods written in them. Before the development of OOP, procedural languages were used. Programming instructions and data are separate entities in such languages and for this reason, chances of errors are high in such programming. This problem becomes more serious when the programs become lengthy, i.e. the number of instructions is higher or the problem itself is a complex one.

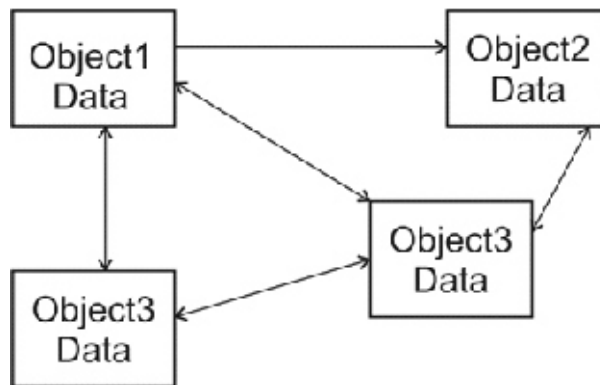


Figure 3.5: Object-Oriented Programming

#### 3.5.1 Object

In OOP, an object is considered as an entity. Everything that is in existence and which can be seen, touched or felt logically is an object. It is quite possible that an object is a group of other small objects. One object can perform many activities. These activities define the behavior of the object. For example, a two-wheeler is an entity and this two-wheeler is made up of many other entities like wheels, seat, handle, pedals etc. We know that there are many two-wheelers which have some or all features, for example, bicycle, bike or scooter. At the same time pedals will not be present in all two wheelers. Another example, if we consider that "Student" is an object, to which we can relate information like student's name, address, class. Similarly, StudentStatus can be another object that which may contain details about presence, subject, exam\_score, exam\_results etc. Objects store data and instructions, and they are created for special processing. Object can be used through the available instructions in the language. An object can also call another object to perform an action through messages. The object to request the object to act through the data or the instruction is called Sender Object and the object that receives is called the Receiving Object.



Figure. 3.6 Object

The control of the implementation now comes to the receiving object and it remains until the instruction is completed. After execution of the instruction, the control goes back to the sender object. For example, if we consider two objects -exam and student. Exam object, uses the student object to get the name of the student.

Sender object can also send information to the recipient object through message, it is called argument. The recipient object generally returns a value to the sender object. This value is used by the sender object in further processing. For example, the StudentStatus object wants to change the value of the student's attendance, for that he sends the new value into a message as a parameter. In this example the value returned by the receiving object will be ignored.

### 3.5.2 Method

How does the receiving object understand messages received from the sender object? How are those messages processed, and the parameters obtained with them are processed? When an object receives a message, the program code that accompanies that message is executed. In other words, these messages determine how the behavior of an object will be and the program code written in the object determines what the object will do. The code associated with the message is called method.

When the object receives a message, that message also contains the name of the method. It determines which program code will be executed and control is transferred to that code for execution.

We know, in procedural languages, a procedure is a group of instructions. method is also a group of instructions. The program code written in a method is similar to code in a procedure. Sending a message to an object is like invoking or calling a procedure.

### 3.5.3 Class

An object comes in to picture when execution of program takes place. While writing the code, we define it through class. Class is a data type and object is a variable. After defining the class, we can make its object variable as per the requirement. The data and program code that we write in definition, are consistent with the need for programming. They are called member data and member functions.

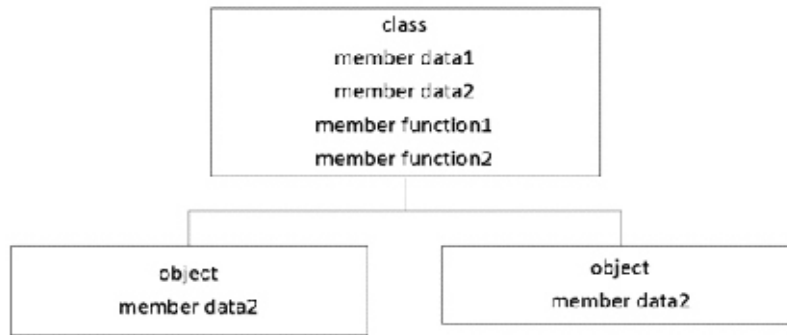


Figure. 3.7 Class

### 3.6 Characteristics of OOP

#### Reusability

Assume that we write a program to solve one problem and we have some similar more problems. There are all the chances that we need to write programs again to solve such problems because one program written to solve a problem may not be useful in solving similar other problems. This happens because in traditional programming, data and programs are kept separate. For example, the new problem for which we need to write program is exactly same except the data types. We need different data types in our new problem. Similarly, there may be slight changes required in the logic, most of the code remains the same. Still, we need to write the program again as the old program cannot work in such situations. Object-Oriented programming is a system in which solutions can be written for new problems while changing the available solutions. It means we do not have to try again from the beginning to program for the new problem. We have already developed programs. What we need to do is make some changes according to our need. Reusability is possibly one of the most important features of Object Oriented Programming.

#### Encapsulation

In procedural languages, the data and the processes which will use that data are defined separately. In object-oriented programming, as we go about defining the class, we write the data and the processes together. These processes are called methods. This is called encapsulation. The abstraction implies that whatever data is defined in a class, it can be used without further explanation and details. The advantage of encapsulation and abstraction is that the use of data structures and symbols is the same as the programmer has thought of while defining them. In this way, the information given in the class is hidden (information hiding) by which the programming becomes more secure.

## **Polymorphism**

The behaviour and implementation of an object are different. Many objects may work for the same message or the same object can be used in different forms. The form of an object may vary according to the requirement at the time of execution. For example, we define a class as 'addition' for mathematical addition operation. In this class we define a method called 'add', in which we write the code to add two integer variables. In the same class there is another method called add, in which we write the code to add two float variables. Now, at the time of execution of the program, if two integer variables are used while the add method, the first method will come in action, and if two float variables are used, the second method will come in action. Here the point to note is that, there are two methods with the same name but the desired method will be used depending on the data sent at the time of execution. The same name is being used in different forms. Due to arrangement of polymorphism, the exchange of messages between the sender object and the receiving object becomes possible.

## **Inheritance**

Inheritance is another important concept in object-oriented programming. After defining a class, if we have to define another class - a class whose properties are the same as the first class and we want to add some other properties - we can do it by use of inheritance. In such a system, we are also using reusability, due to this reason there is no need to do all the work again which we did while defining the first class. This also saves time and facilitates maintenance of the program. The class taking the property of a class is called a Sub Class or Derived Class, and the class from which properties are forwarded is called Super Class or Base Class.

### **3.7 Object-Oriented Problem Solving Approach**

Object-oriented programming approach is similar to the approach we follow in our daily life to find solutions to problems. We identify the real-life objects that are helpful in solving the problem and use them in a certain order. We usually do it to solve our problems. Think about the objects in problem and use them to solve the problem. While programming, we create objects that can solve the problem. Based on the messages sent to the object, there are different operations are performed which solve the problem.

The process of resolving an object-oriented problem can be divided into four phases.

1. Identify the problem
2. Identify the objects needed for the solution
3. Identify messages to be sent to the objects
4. Create a sequence of messages to the objects that solve the problem.

### 3.8 Benefits of Object-Oriented Programming

- o Programming is easy. After examining a class thoroughly, the convenience of using in the work decreases the likelihood of errors.
- o Class can be considered as a "black box". We don't need to have the internal details of a class to use it. Methods available in a class can be used without the knowledge of details.
- o Unnecessary effort of writing the same type of programs is avoided. A class can be used directly as per our requirement. In other words, program code can be reused.
- o With the help of different compilers, code can be made in conformance with another CPU, which means code-portability is available.

### 3.9 C++ Basic Elements

C ++ uses the class to use the concept of objects. Keyword 'class' is used to create a class. Class is a user-defined data type that a programmer can use to define a real-time object for its programming, and then programming through a resource based on it. While defining we do not create objects, but make a model which is called a class. We make objects using class as per the requirement. For example, we define a class named student, and, in the program, can create different objects for different courses according to our need.

Each class has some features defined in it, these are the characteristics which we want in an object. They are called attributes of an object. It is not necessary that all the attributes are required in each object. We can create another class which has all the attributes the first class and some new attributes too. This feature is available to us through inheritance.

Example

```
class class_name
{
private:
 Data_Members;
 Member_Functions;
public:
 Data_Members;
```



Member\_Functions;

};

We create object to use class, which we call instance of a class. Class is a definition and an object a variable. A class contains variables known as data members, and procedures to handle these variables known as member functions.

Each programming language is a set of symbols, specific words and rules. A program is written using all of these. There are some elements that are found in every language. We now discuss some of the elements available in C++.

### Character Set

Group of characters which is recognized in a language is known as Character Set.

|                                     |                                                                             |
|-------------------------------------|-----------------------------------------------------------------------------|
| Letters                             | A-Z, a-z                                                                    |
| digits                              | 0-9                                                                         |
| Special Characters<br>(विशेष अक्षर) | Space + - * / ^ \ ( ) [ ] { } = ! = > < ' " \$ , ; : % ! & ? # < =<br>> = @ |
| Formatting characters               | backspace, horizontal tab, vertical tab, form feed and carriage return      |

### 3.10 Tokens

A token is a group of characters. Programmer writes program using these tokens. Tokens available in C++ language are: Keywords, Identifiers, Literal, Integer Constants.

#### Keywords

C++ language has certain words which are already defined in the language. They are reserved words. Meaning of these words is already known to the compiler and programmer cannot change that meaning.

#### Identifiers

C++ provides a feature of using symbolic names (identifiers) for various data elements (variable, function, class, module, or any other user-defined item) in C++ is available to the programmer. These names are created by taking the letters from the C++ character set. The rules for making the name are as follows:

- o An identifier may have alphabets (A-Z, a-z), digits (0-9), and / or under score ( \_ ) in the name.

- o The initial character cannot be a number.
- o It should not be a reserved word.

### Literals

Those data elements whose value cannot be changed in the program are called literals.

- ◆ Integer constant : Integer constants mean whole numbers which do not have any fractional part. C++ provides three types of integer constants.
- ◆ Decimal Integer Constant: Numbers, first digit cannot be zero. Example: 78, -168, +4
- ◆ Octal Integer Constant: Numbers, first digit is zero. Example: 014.
- ◆ Hexadecimal Constant: Numbers, first two characters are ox or OX. Example: OX24C

**Character Constant:** One or more character which are written within single quotation marks, for example 'A', '4', '\t'. The characters which cannot be printed directly using keyboard, like tab, backspace, are printed using escape sequence.

**Floating Constant:** Fractional numbers. These can be written in fractional form or exponential form, example -0.342, 314159E-5

**String Constant:** Sequence of characters written in double quotation marks is known as string constant. '\0' is added at the end of string, which denotes the end of string. Example, "TECHNOLOGY" will be stored in memory as "TECHNOLOGY\0" and its size will be 12 characters.

### Data Types

#### Basic Data Types

| Type                  | Keyword |
|-----------------------|---------|
| Boolean               | bool    |
| Character             | char    |
| Integer               | int     |
| Floating point        | float   |
| Double floating point | double  |
| Valueless             | void    |
| Wide character        | wchar_t |

Following data type modifiers can be used to change the basic data types.

- ◆ signed
- ◆ unsigned
- ◆ short
- ◆ long

Various variable types, how much memory would be needed to store the value in memory, and what is maximum and minimum value which can be stored in those variables, is shown in the following table.

| Type               | Typical Bit Width | Typical Range                   |
|--------------------|-------------------|---------------------------------|
| char               | 1byte             | -127 to 127 or 0 to 255         |
| unsigned char      | 1byte             | 0 to 255                        |
| signed char        | 1byte             | -127 to 127                     |
| int                | 4bytes            | -2147483648 to 2147483647       |
| unsigned int       | 4bytes            | 0 to 4294967295                 |
| signed int         | 4bytes            | -2147483648 to 2147483647       |
| short int          | 2bytes            | -32768 to 32767                 |
| unsigned short int | Range             | 0 to 65,535                     |
| signed short int   | Range             | -32768 to 32767                 |
| long int           | 4bytes            | -2,147,483,648 to 2,147,483,647 |
| signed long int    | 4bytes            | same as long int                |
| unsigned long int  | 4bytes            | 0 to 4,294,967,295              |
| float              | 4bytes            | +/- 3.4e +/- 38 (~7 digits)     |
| double             | 8bytes            | +/- 1.7e +/- 308 (~15 digits)   |
| long double        | 8bytes            | +/- 1.7e +/- 308 (~15 digits)   |
| wchar_t            | 2 or 4 bytes      | 1 wide character                |

Depending upon the compiler and the computer you are using, the size of data types in memory may differ from what is shown here.

## **Input-Output**

In the standard library of C++, there is one header file called - iostream.h, which may be used to read data from keyboard and display it on the screen.

Following C++stream objects may be used for input-out

cout console output

cin console input

### **cout object**

cout is used to display message on screen using << insertion operator.

```
cout<< "Hello World";displays Hello world on screen
```

```
cout<< 250; // displays number 250 on screen
```

```
cout<< sum; // displays the value of variable sum on screen
```

If variable and constant are to be displayed in combination, << can be used more than once. Example

```
cout<< "Area of Crop Field is "<< area<< " square meter" ;
```

### **cin object**

cin is used to input value using keyboard by the user. To store the value in memory >> extraction operator is a must.

```
cin >> marks; // Data will be received from keyboard and will be stored in variable marks.
```

## Important Points

1. With the help of procedural programming, the size of the program can be reduced, thereby reducing the chances of errors.
2. In modular programming, the procedures are collectively stored in modules.
3. Procedural languages are suitable for general programming.
2. For problems requiring artificial intelligence, where logic is fuzzy, procedural languages are not compatible.
3. Class is a data type and object a variable.
4. To program for a new problem, we do not need to try again from the beginning, by making changes in the program already made, we can get the same solution.
5. When defining a class, we write data and code together. This is called encapsulation.
6. Abstraction means that whatever data or function is defined in a class, it can be used without further explanation and explanation.
7. At the time of execution of the program, the object may be as per our requirements.
8. A method in object-oriented programming is similar to what is a procedure in procedural programming.
9. In the context of procedural Programming methods are group of those instructions. Method and programme code written is code of procedure.

## Exercises

### Objective Type Questions

1. The ability to have different forms
  - A. Inheritance
  - B. Polymorphism
  - C. Member function
  - D. Encapsulation
2. The process of taking out an object's properties is called
  - A. Polymorphism
  - B. Inheritance
  - C. Reusability
  - D. Data hiding
3. What facilities in C++ makes it a strong language?
  - A. Easy implementation
  - B. Reusing old code

- C. Writing new code                      D. All of the above
4. Which facility is not available in C++?
- A. Encapsulation                      B. Abstraction
- C. Polymorphism                      D. Exceptions
5. To be called 'object-oriented', a programming language must have facility for
- A. Encapsulation                      B. Abstraction
- C. Polymorphism                      D. All of these
6. Which statement is not true?
- A. To perform a specific task, part of the code is called a function
- B. With the help of a function, large and complex programs can be divided into small and simple.
- C. Functions are helpful to carry out common tasks using the available code.
- D. A function can be used only once in a program.
7. A function defined in a class is called
- A. Member Variable                      B. Member function
- C. Class function                      D. Classic function
8. In OOP which concept means "Only the information required is communicated outside the object"
- A. Encapsulation                      B. Abstraction
- C. Data hiding                      D. Data binding
9. Which approach is better for saving information and understanding real-time examples, like information about vehicle or employee?
- A. Procedural Approach                      B. Object Oriented Approach
- C. Modula Approach                      D. None of the above
10. Advantages of object-oriented programming
- A. Re-use of code                      B. Better use of code
- C. Error free                      D. All of the above

### Very Short Type Questions

1. Likelihood of errors in procedural programming are less because the size of the program can be ..... (reduced)
2. First O in OOP is ..... (Object)

3. Data and ..... may be taken in a new class, in hierarchy.
4. In C++, class is a ..... data type.
5. An advantage of Reusability is..... (Ability to use already written code)

### **Short Type Questions**

1. Differentiate between class and object.
2. Differentiate between procedure and method.
3. What are the tokens in C ++?
4. What are the character constants in C ++?
5. What are the data type modifier?

### **Essay Type Questions**

9. Explain the characteristics of OOP.
10. Describe basic data types of C++.
11. Explain the use of 'cin' and 'cout' with the help of examples.
12. Differentiate between Encapsulation and Abstraction.

### **Answer Key**

1. B    2. D    3. D    4. D    5. D    6. D    7. B    8. C    9. B    10. D

## Chapter 4

### Computer Networking

#### 4.1 Data Communication Model

Introduction: At present, the computer has emerged as an essential service so many computers interconnect and share information between each other and the exchange of data between each room exits has increased phenomenally in the power of the compute. is Networking. The network is basically a group of all the elements that are used to connect computers and exchange data between multiple computers. The basic purpose of networking is to increase productivity.

Properties of Networking: While building a network the following basic qualities are kept in mind:

1. Price: This category comes with useful elements, their adjustment and maintenance.
2. Security: All elements and statistics are protected in this category.
3. Speed: The speed at which the data is exchanged so that the network gets quality comes in this category
4. Positioning: In this category all the elements of the network are interconnected in physical condition.
5. Scalability: In this category how the network accepts new change is kept in mind.

#### Data Communication

The exchange of data by any medium (wire or wireless) between the two devices used in the network comes in this category. The direct communication of data is with its exchanges rather than the generation of data and the effect of the exchange of data depends on three factors.



1. Delivery - Mechanism to exchange data by keeping in mind the right target i.e. the data is correctly achieved by goal only.
2. Purity - If the medium makes any changes in the data then it is unusable.
3. Timeliness - It is important only if the mechanism exchanges data at the right time.

### Information System



Figure 4.1: Information System

Each section of this picture is a part of the communication system. Each part has its own meaning and its utility.

1. **Origin** - The tool / Application that produces data.
2. **Transmitter** - The medium does not accept the data to send it to the target in the form it produces, so the transmitter is the device that prepares the data to be sent to the target.
3. **Sender System** - When two devices exchange data between each other the communication system between them is called the sender system.
4. **Recipient** - Recipient is the device that receives data from the sender system before moving on to the target.
5. **Goal:** The data on the last device which crosses the sender system from the originating device is called the target.

Part of the data communication system - the data communication system has 5 major parts-

1. Message - Message is the data or information that we want to transmit on the given medium. The message can be of various forms such as text, image, video
2. The sender - This is the device that sends the message. It can be a computer, telephone device or camera etc.

3. Receiver - This is the device that receives the message at the end.
4. Media - From the source the message reaches its goal by using media. This is the path which leads to the goal from the origin through the medium. Medium can be an optical fiber, coaxial wire, twisted pair or radio waves.
5. Protocol - Protocol is a set of rules that govern data transmission. It gives an agreement between two instruments (Source and Destination).

### **Definition of Network**

The network is a group of some electronic devices that are connected to some wires or wireless. The work of this medium should carry information from one end to the other. This medium allows all the consumers of this network to share these devices with each other. These devices which are used in the network are called nodes. The number of nodes in a network can be as per requirement. To be qualified and effective for a network it should meet many criteria mainly the following-

- (1) **Error free work** - Job supplement means an exchange of information without error. It involves the time taken in the decision taken by the goal of reaching the goal of information and after it. The editing of a network depends on the following:
  - a. Consumers number - If there are more consumers in the network then their impact falls on the speed of exchange of information in the network.
  - b. Communication - This means that the speed of transmitting data is at speed. It is measured in the Kbps / Mbps/ Gbps.
  - c. Types of medium - It is meant to be used in the exchange of data through the medium of work. The quality of communication of data depends on the medium used.
  - d. Types of equipment- Equipment used in the network affect both speed and efficiency of data exchange. A high-performance computer works effectively when editing the work.
  - e. Software - software that operates in the network of devices that operate on the network also affects the quality of the work of the network.
- (2) **Reconciliation** - Reconciliation means predicting the accuracy of time and data to be re-reacted by the goal. For example, we can assume that a network printer

of a network predicted 3 seconds to print a page but it takes 30 seconds which means that there is no harmony with the target in my network.

- (3) **Trustworthiness** - A standard of the usefulness of the reliability Network.
- (4) **Recovery / Restoration** - How soon after a network gets spoiled its utility reaches its real state, its usefulness depends on it.
- (5) **Security** - Protecting against the use of our network's devices, software and data in unauthorized use is the only security. Wire protection is also done in this category.

## 4.2 Network Topology

The style of connecting computers with each other is called topology. Choosing topology style is an important step in the process of networking. Before choosing a topology it is necessary to consider many aspects. Details of some of these aspects are presented below-

- (1) **Cost**- To reduce the cost of a network, it is necessary that we try to keep the costs of its construction low. To do this, it is necessary to ensure the medium of changing the information signals. The length of the transmission on which the transmission is to be done can also be changed as required.
- (2) **Flexibility**- Because the number of nodes cannot be ascertained as the network's expansion is not pre-determined. For this reason, our chosen topology should be such that there is a capacity to expand.
- (3) **Reliability**- There can be two types of fault in a network. Deterioration of one of the first nodes, the second, the whole network is getting worse. We should look at the fact that when one node gets spoiled, it will not affect the rest of the network.

Topology can be of three types, they are:

**4.2.1 Bus Topology** - All computers in bus topology are connected to the same cable. Every node (computer) is connected to two other nodes, one can send the same computer message at a time in this genre.

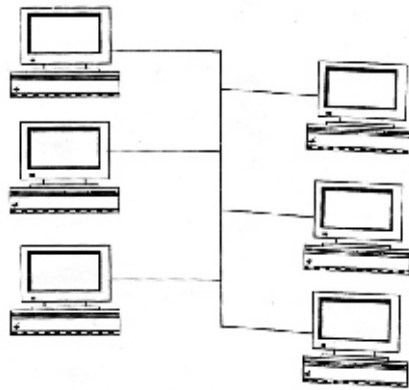


Figure 4.2 BUS Topology System

### **Benefits of Bus Topology**

1. This technique is simple, easy to understand and easy to use.
2. In this, less cable is required to connect computers. So, it falls cheap.
3. It can be easily added to many computers.

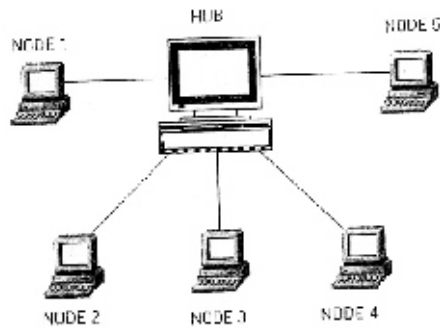
### **Disadvantages of Bus Topology**

1. When computers are unable to establish coordination of each other, all computers start transmissions simultaneously. This causes the network to slow down.
2. When the cable breaks two computers cannot interconnect.

**4.2.2 Star Topology:** Computer is connected to a hub or switch in this topology. The hub or switch is in the center of the network and sends the signals of one node to all the nodes.

### **Benefits of Star Topology**

1. It's easy to add new computers. The computer is connected to the hub by the cable and in the network.
2. It can be easily detected by hubs or switches.
3. Turning off the work of a computer does not have any effect on the rest of the network.



चित्र 11

Figure 4.3 Star Topology

### Disadvantages of Star Topology

1. When the hub or switch goes down the entire network stops working.
2. Star Topology is expensive because every computer is connected to a cable separated by the hub or switch located in the center.

**4.2.3 Tree topology:** In this topology, computers are added in such a way that they are the branches of tree. The node located at the top is called root. Root is connected to zero or more child nodes (breastpunct thumbs).

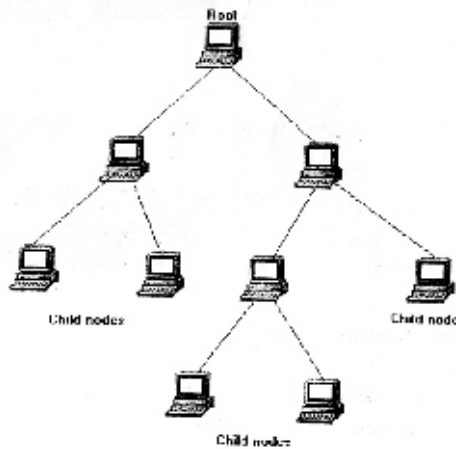


Figure 4.4 Tree Topology

The root node is the father of the child node. Every child node also has many child nodes. In this way, the parent node of every node consists of only the parent node

of the root node. In this type of topology, there is only one path from one node to the other node.

### **Benefits of Tree Topology**

1. It is easy to add new computer to child node in this topology.
2. It does not require a centered hub.

### **Disadvantages of Tree Topology**

1. Adding a computer to a root node is not easy.
2. Adding or removing the computer to the root node can be interrupted by the network.

## **4.3 Concept of LAN ,WAN, MAN**

Networks are broadly divided into three types:

**4.3.1 Local Area Network** - This network is spread within a building or within a few kilometers area. It is used for the exchange of information technology resources by various computers of any office or factory.

LANs are usually spread in smaller areas such as in a department or a building. The LANs are small so it is easy to handle them. Generally there are some flaws from short circuits and unwanted signals. LAN connects all nodes from the same cable. Telephone cables are used for transmission so they are cheap.

### **Features of LAN**

1. The most important feature of LAN is its speed. Generally, its data transmission speeds are between 10 to 100 mbps. Currently it is 1 Gbps and more.
2. LAN is a flexible network, without disturbing all networks the computer can be added and removed.
3. Since LAN is limited in the area, we can take several types of topology work in it.

**4.3.2 Metropolitan Area Network** - Metropolitan means the city. This network is spread over a very large area such as the city or town. MAN is a major form of

LAN, It also uses the same technique as LAN. Creating this is more complex than LAN. It can be spread around up to 60 kilometres. It connects branches and business houses in different areas of the same city. Examples - Cable TVs of cities Network.

### Features of MAN

1. The entire network is operated and controlled by a centralized machine.
2. MANs main goal is to use software and hardware resources together.
3. MAN can transmit both data and sound.

**4.3.3 Wide Area Network (WAN)** - In comparison to other networks, the WAN works in a very large area, it can be spread throughout the country or even the peninsula. All the computers in the countries or peninsula in the WAN are connected to each other. They can also exchange computer data and give a centrally controlled transmission. WAN can be from transmission, wired media (telephone line, fiber optics) or wireless medium (microwave). In all three types of networks, it is spread over the largest area. Adding two WAN network is very complicated. They may down due to short circuit, wire breakdown or other circuit.



Figure 4.5 WAN System

### Features of the WAN

1. The WAN uses several network devices for transmission such as router, switch, and gateway.
2. WAN uses two types of switching method for data transmission.
  - a. packet switching
  - b. circuit switching

3. Their data transmission speed is slower than other networks.

#### **4.4 Standardization and protocols**

In the electronic devices there is a need for a group of rules for communicating that acts as the reason of communication between the sender and the recipient. The protocols ensure the correct communication between the sender and the recipient. The sender and recipient of no protocol will not be able to understand the computing accurately. There are many topics in which the protocols are used.

**Protocol:** In the computer network, there is a lot of system requirements for transmission of the entities, it is necessary to agree with these protocols. Groups of rules that govern data computation are called protocols. The following element of the protocol is-

1. Syntax - Tells the format of data.
2. Semantics - These are related to every part of a bit how it has been interpreted.
3. Timing - When the data has been sent and at what speed

Regardless of any technique or system as per rules the regulation is known as a standard.

**The need for Standards:** The standers create a competitive and open market for equipment makers and make tools useful for working around the globe so that no company can make its arbitrariness.

#### **Standardization Committees**

- 1) ISO (International Organization for Standardization)
- 2) ANSI (American National Standards Institute)
- 3) IEEE (Institute of Electrical and Electronics Engineers)
- 4) ITU (International Telecommunication Union)

#### **4.5 Transmission**

Transmission is the transfer of data between two or more devices under which data is sent and received between one or more devices.



## Methods of Data Transmission

Data on transmission medium can be sent in two ways - asynchronous and synchronous

**1. Asynchronous Data Transmission** - Asynchronous transmission is also called start-stop transmission. In Asynchronous Transmission Start Stop Bit between each letter. The sender can always send a letter which the receiver receives. As long as the line's hardware is ready to send the data, the asynchronous communication line remains open. To tell about the data being sent, a series of bits is sent to the receiver device because the line remains open. After sending the entire data, the recipient of the data is informed that the data has been exhausted. After this the stop bit is sent so that the line can be broken. The interval between the two letters in the asynchronous transmission is undefined, i.e., to computer destination, a line of letters can be sent or the data can be sent at irregular intervals.

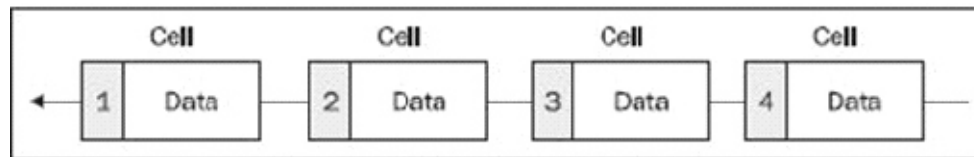


Figure 4.6 Asynchronous Data Transmission

The main advantage of this is that the computer does not need storage because transmission is literally done. One of the disadvantages of its transmission is that the line between sending two letters remains open.

**2. Synchronous Data Transmission** - There are two channels in Synchronous Communication to send a data and to keep all the links together for the start. To start the two computers together, the clock hardware is used when the computer is ready to send the data, then it sends a mixture of bit towards the receiver which is called the sync character because the first letter is bad therefore the second letter is sent along with it so that it can be ensured that all links can be started together.

A block of synchronous transmission letter is formed. Every block give header and trailer information by which the computer receiving it matches the clock to the sender's computer. The header is the information for identifying the sender and receiving computer. After the header, there is a group of letters which are the actual information and in the end of the block it is trailer. The trailer sends information about the end of the message. After this there is a check letter which helps in finding the defects during transmission.

The advantage of synchronous transmission is its efficiency. It eliminates the need for a start stop bit with every letter. It provides high data transmission speed compared to asynchronous. The interval between sending the block is very low and the block is sent at maximum line speed. The only drawback of synchronous transmission is that the storage device needs buffer memory for storage which collects the blocks of the line.

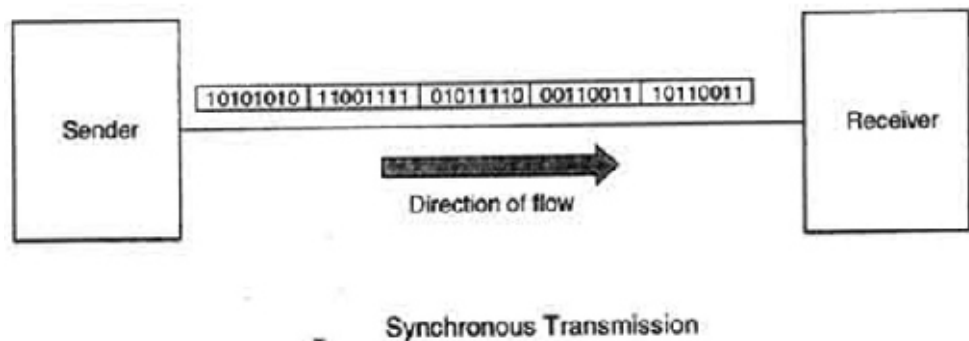


Figure 4.7 Synchronous Data Transmission

### Data Transmission Methods

Data transmission methods are

- (1) Simplex
- (2) Half duplex
- (3) Full duplex

**Simplex** : In simplex transmission method data communicates on one side. Example of Simplex is Television Communication. In this, the main transmitter sends the signal but does not expect the answer. The recipient cannot respond to the transmitter. The example of Simplex translation is the keyboard because it is connected to the computer and can give only data to the computer. In one-sided transmission it also requires a message that can tell that the recipient has obtained the data. Simplex transmission is not used in the work because a back path is required to send self, control and error signals.

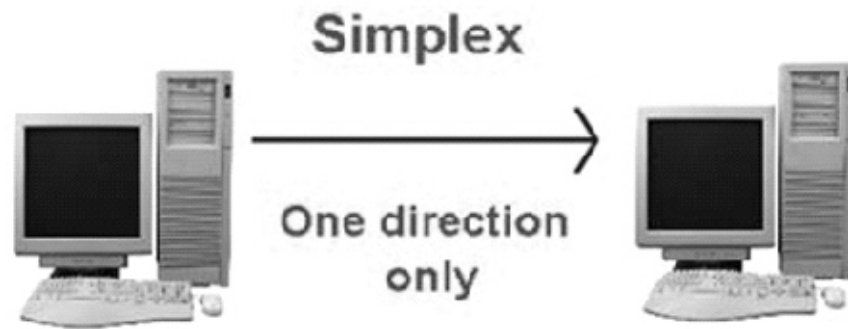


Figure 4.8 Simplex data Transmission

**Half Duplex:** In Half Duplex Method, both units do communication through the same medium but only one unit can transmit data at a time, when one unit is sending the data and the other is receiving the data so that Half duplex line sends and receives data in an alternate manner. It requires two cables. This is a very common way. The transmission is become sound. It can speak the same person at a time in that it is supposed to serve as the unit of the computer can transmission of data and computer sends a message of acceptance. In Half Duplex, if both devices try to send and receive data together, packets collide.



Figure 4.9 Half-Duplex Data Transmission

**Full duplex:** In the full duplex, the information mission flows in both directions at the same time. In full duplex method, the direction of the translation is changed as well as the line is diverted.

In such a case, the speed of the line in a computer is very fast. Using the full duplex method we can improve the efficiency of the translation. In Full Duplex Communication, both devices can send and receive data in both directions simultaneously. Its main example is telephone system.

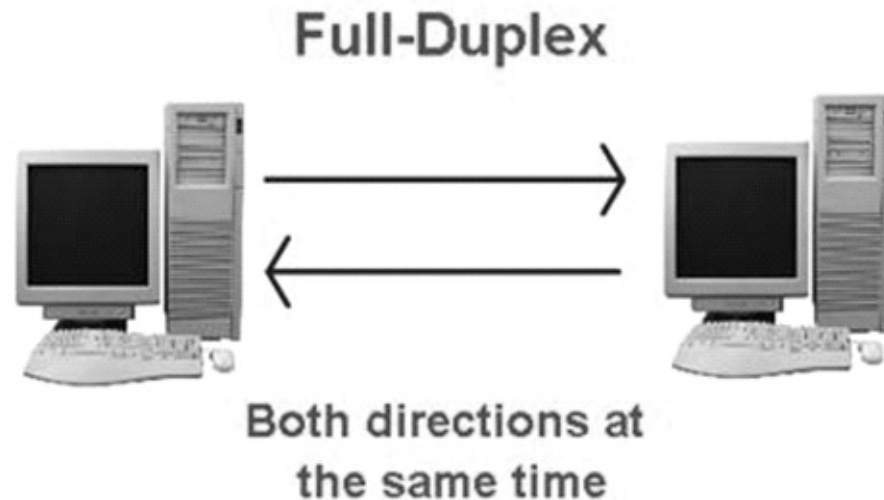


Figure 4.10 Full Duplex Data Transmission

**Parallel Transmission:** In this transmission more than two bits can transmit from source to destination at a time by using multiple cables.

#### 4.6 Networking Signals

Signal is the electrical electronics and optical representation of Data, to which can be sent to the communications medium. Data transmission, Digital translation or Digital communication is point-to-point and point to multi physical transfer on channel. For example, the sound signal that convert from a microphone or converts to a microphone converts

There are two types of networking signals

(1) Analog Signals

(2) Digital Signals

**Analog signals:** Analog signal is a continuous signal that changes according to time. The voltage, current and frequency changes in the electronics signal are changed to present. Most signals remain in the form of analog signals, which are later converted to digital signals because analog signals sending devices are expensive. The IC circuit is

used to convert analog to digital. Later these signals have to be received in analog so they are converted from digital to analog. While converting signals and sending them some errors are added that spoil the main signal, they are called impairments. As the signal moves forward on the media, its ability or energy slowly starts to decrease, in such an anal signal we use the amplifier to increase.



Figure 4.11 Analog signals

**Digital Signals:** A digital signal is an electronic signal that is converted to bit patterns. Every point of the digital signal is the discrete value. Digital Signals are represented by zero or one. Like analog signals, even with digital signals, there are errors during transmission due to which the form of real signals changes. And as the signal progresses over the media, its energy gets reduced. At least no matter how much the energy of any signal should be so that the receiver can understand the pattern between them, otherwise the receiver will not be able to detect the correct information. To correct the reduced energy, the digital signals uses repeats. HDMI technology is used to translate audio and video simultaneously, which is an example of digital signals.



Figure 4.12 Digital Signals

#### 4.7 Transmission Media in Networks

Transmission medium is the path on which transmitters and receivers exchange signals or transmit data to each other. The transmission media determines the physical path between two devices Transmission medium can be divided into two parts.

(1) Guided Transmission Media

(2) Unguided Transmission Media

**Guided Transmission Media:** Guided media is the medium in which the signals run according to the physical path or not confused with the path. The media's capacities in the directed media are related to its length and how it is connected and depends on it.

Examples: Twisted pair cable, coaxial cable and optical fiber cable

**Twisted Pair Cable:** Two cables are wrapped in each other in this technology. The two wires are wrapped in each other, reducing the electrical interference of each other. And having a couple wrapped in each other reduces the electric interference of the other couple. Twisted Pairs can only transmit digital and analog signals. The twisted pair cable is used for local telephone transmission and for digital data transmission between the main computer and the network computer in a short distance of up to 1 kilometer, the speed of data transmission can be 9600 bits per second up to 100 meters.

Figure 4.13 Twisted Pair Cable

Twisted pair cable is of two types

(1) **Unshielded Twisted Pair (UTP):** This is the most popular twisted pair cables and it is also becoming popular in cabling of local area network. UTP is generally used only in the form of telephone system and in many office buildings already. In this kind of cable there is possibility of a cross-talk in the workplace.



Figure 4.14 Unshielded twisted pair cable

The main problem of the UTP cable is the mixing of the signals of a wire which is called cross talk. Shielding is used to reduce cross-talk.

(2) **STP:** The STP uses high-quality tuned wire jackets that are more secure than the UTP's jacket. The STP is wrapped around the wires and around the wires with the call, making it very effective to make STP brilliant. By which transmitting data is protected against heavy crimes, therefore, STP transmits long distances of data in more speed than UTP and is more secure.



Figure 4.15 Shielded twisted pair cable

**Coaxial Cable:** There is a hard copper coil called core which is now wrapped with insulator. It is covered with unstained clusters of fine wire.

There is a protective plastic shell on top of these clusters. Symbols are exchanged by the core of copper. The outer armor is made from clustered wires.

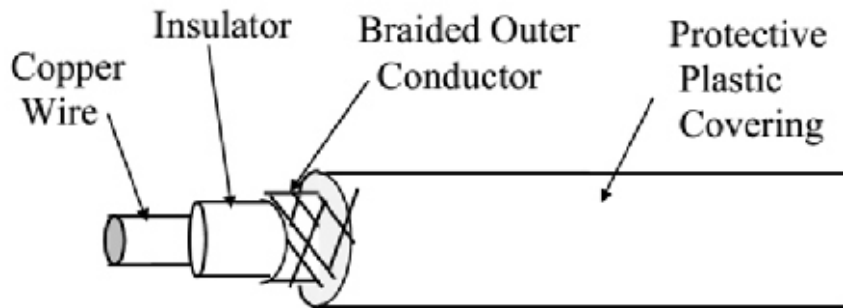


Figure 4.16 Coaxial Cable

Coaxial cable is commonly used for television networks. Coaxial cable can transmit data at a faster distance than a twisted cable. It is cheaper than fiber optic and can be used easily. There are two types of coaxial cable.

(1) **Thicknet:** This cable is used only in the TV network. It is thick and cannot be

easily folded, so it is difficult to use.

- (2) **Thinnet:** It is mostly used in networking. It is only flexible and cheap and can be use easily.

**Fiber Optic Cable:** This is the most innovative technology of without wire. It is the most excellent in handling security and data. It only transmits the light waves in place of electrical signals. It is the safest to send data because the data cannot be stolen in this cable. Inner part of the cable are made of plastic or glass in which the transmission of light occurs. up of inner part, which sends in the hinterland after hitting the light is up a plastic flower on top of the glass layer sends.

The sender is connected to a data transmission device that converts the electrical signals into light waves. These light waves are transmitted by fiber optic cable, the second data transmission device transmits these light waves into the widest signals before the receiving computer. Later this fundamental signal is sent to the receiving computer. Fiber optical cables are more empensive than coaxial cable. It tranmit data by 60 megabytes to two gigabytes per second.

Fiber optic is more suitable for more long distance transmission of data but this is the most expensive, speed and transmission medium of without wire.

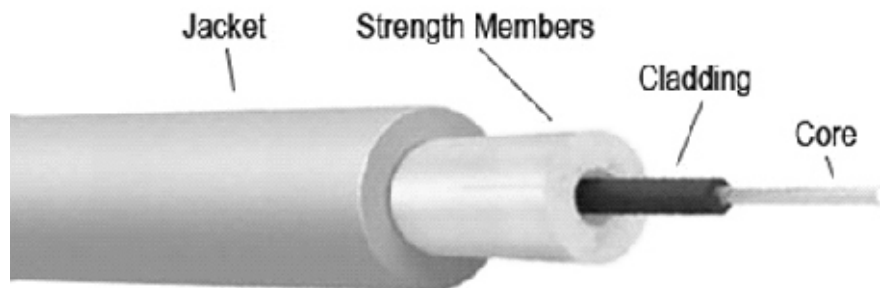


Figure 4.17 Fiber Optic Cable

**Unguided Transmission Media:** Unguided transmission media is the medium in which radio waves are used and the signals do not move according to the physical path and it is used where it is not possible to reach

**Radio transmission medium:** Radio waves can easily be generated. It can reach at long distances and it can easily cross the buildings so they are taken more for the work of data transmission. After the transmission radio waves can more in all directions, so the sender and the recipient are not required to be in the same line. In the past radio communication was used in telegrams. Transmissions of countries were used in the work of this radio transmission, on low transmission speed and low frequency (Fre-



quency) but now the transmission of data transmission at fast transmission on VHF (High High Frequency) and UHF (Ultra High Frequency) use the frequencies in many ways.

|         |                      |                                                                      |
|---------|----------------------|----------------------------------------------------------------------|
| 30 KHZ  | Low Frequency        | Long Distance Telegraph                                              |
| 3 MHZ   | Medium Frequency     | Used for sending medium-range non-wire signals.                      |
| 30 MHZ  | High Frequency       | Used for sending long-range signals.                                 |
| 300 MHZ | Very High Frequency  | Used for F. M. (FM) transmission and short-range mobile transmission |
| 3 GHZ   | Ultra-High Frequency | In television broadcast                                              |
| 30 GHZ  | Super High Frequency | Use in Radar and Microwave Transmission                              |

**Microwave Transmission Media:** These signals are also sent and received without the help of cable like TV and radio signals. Microwave signals are broadcasted by antenna mounted on buildings. The sender antenna and the recipient antenna are placed in the same line as the microwave signal only proceeds in one direction. To set the sender antenna and the recipient antenna in the same line is called line of power transmission. Microwave stations located on the ground are set in a type which imposed that they can exchange information with each other only if they can exchange information only when one these microwave stations are placed on the roof of buildings or on the mountain, hence the transmission path remains free. Stations need to be in the same line for microwave transmission.



Fig. 4.18 Microwave Communication  
(152)

Microwave systems use the repeater stations to diagnose the problem of the line off site. The distance of the microwave transmission is limited, so after every 25-30 kilometers the repeaters are placed. The sender station sends the microwave signals that the repeaters receive, than repeaters give them power so that they can reach the distance and signal after providing power Programmed again is transmitted. Microwave transmission cheaper fiber optic cable so microwave signals are used for the transmission of high-speed local and long distance .

**Infrared Transmission Media:** It is used for short range transmisson. Infrared transmission is cheap, easily used and there is no legal obstacle in using them as it works within the building. There should be no barrier between the sender and the recipient. Infrared medium uses infrared light for the transmission of signals. Light emitting diode (LED) transmits light signals and photo diodes receive light signals. Because infrared signals work very fast, so their data transmission speeds are very fast.

Examples of infrared transmission are remote of TV. Examples: In a large room, many computers with equipment to obtain infrared transmission are equipped with local network such as compute infrared can connect with.



Figure 4.19 Infrared Communication

**Satellite Transmission:** First time, transmission in satellite was used in 1960 when NASA launched Eco satellite. Satellite transmission can happen only when both antennas in a line. The main difference in satellite transmission and microwave transmission is that an antenna is mounted on the satellite, which is located approximately 3600 kilometers above the equator, the Geo synchronous orbit. This planet remains stable compared to Earth and remains at a point relative to the Earth, due to this, transmissions can be done by satellite system in mobile devices and all the locations can be reached.

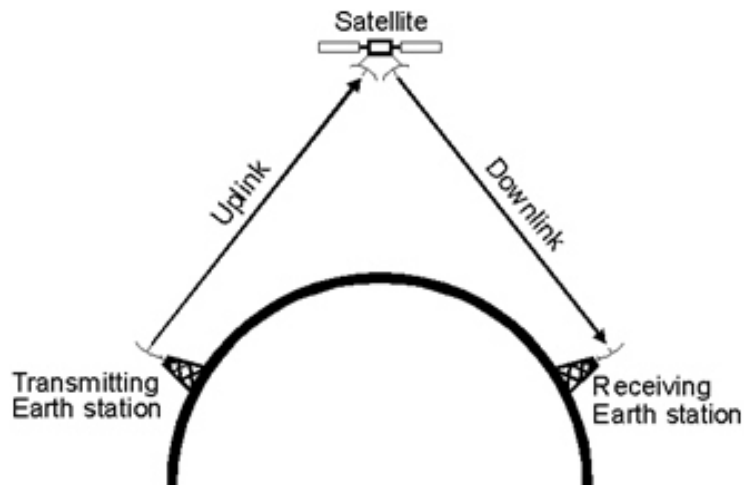


Fig. 4.20 Satellite Communication

In satellite transmission 6 GHz signals are transmitted by the transmitter in space. The signal becomes impaired due to long distances. By making this signal powerful by a transponder on the planet and sending it back to the earth at 4 GHz frequency, the receiver receives these signals on the earth. Installing satellites in the classroom is very expensive.

### Satellite Transmission Features

1. Life cycle of every satellite takes between 7 to 10 years.
2. The stations located on earth are at a distance from the user. The signals have to be fixed by the high speed transmission medium.
3. Anyone can tap the satellite transmission.
4. In many situations, satellites cannot work because they run by solar energy. In the state of the solar eclipse and in the middle of the sun, due to which the satellite ceases to meet solar energy. If the capacity decreases, then they stop working.
5. The shape that sends the satellite and the return frequency of the signals on the earth may be of C band (4/6 GHz) or K4 (11/4 GHz). The same frequency requires large antenna and blocking stations due to rain and environment have to face.

## 4.8 OSI & TCP / IP Model

Before knowing about the OSI and the TCP / IP model we also need to know how the Internet came.

If we talk about history, then the history of the Internet begins with the Department of Defense United States. While there were already computers in their different places, there was no exchange of information between them and the information that was exchanged was sent in an external storage, so the Department of Defense took the initiative that the computers of different places can be added together, they named this project as DARPA.

The department of defense and various other supportive departments formed research and some protocols in organizations which could exchange information by connecting computers. The whole process gave the name of the DOD model, which had four layers. After it changed its name to ARPANET and ARPANET presented a model called TCP / IP Model In addition to this, there were 4 floats like DOD, whose names are -

- 1 Application layer
- 2 Transport Layers
3. Internet Layer
4. Network Access Layer

The concept of the layer was given to spread the work of network communication in different parts.

Along with Arpanet, another organization, ISO (International Organization for Standardization) presented a model named OSI model, was a model of 7 layers. After the conversation, it was given the status of reference model and the TCP / IP model was given the protocol model.

The TCP / IP model is applicable only in computer networks and if the information is to be read or read about its different layers, then the OSI model is kept in mind..

| OSI Model          | TCP/IP Model         |
|--------------------|----------------------|
| Application Layer  |                      |
| Presentation Layer | Application Layer    |
| Session Layer      |                      |
| Transport Layer    | Transport Layer      |
| Network Layer      | Internet layer       |
| Data Link Layer    |                      |
| Physical Layer     | Network Access layer |

### OSI model facility

1. The big picture of the network can be understood.
2. It is possible to see hardware and software working together.
3. It is easy to know what new technology has developed in it.
4. Easy to troubleshoot problems of different networks.
5. Can be used to compare basic functional relationships on different networks.

### Information about various Layers of OSI Model:(Top to Down)

**1. Application layer:** This is the top level. Various methods are used to manipulate data (information) in this layer. The result for the user is the transfer of disturbing files also used in this layer. Mail services, directory services, network resources etc. are provided by the application layer.

**2. Presentation Layer:** The presentation layer takes care that the data should be change in such a way that the information of the receiver is understood and able to use the data. This layer plays the role of translator.

**3. Session Layer:** This layer works as synchronization of interactions between two devices. In order to avoid any harm to the data, the presentation layer properly synchronizes data from the presentation layer on the other side.

**4. Transport Layer:** This is primarily the most important layer, whose main task is to transfer the data from one computer to another computer responsibly. This layer decides that the transmission of data will be on parallel path or on single path. The main functions of this layer are multiplexing, segmentation and addressing. This layer breaks the data into small pieces.

By which the data can be transmitted correctly is called the Protocol Data Unit (PDU), The Protocol Data Unit on the Transport Layer has been given the name of Segment. The Transport Layer also works along with addressing all these, which helps in recognizing the applications in the network. Port numbers are used as addressing on the Transport Layer, which ranges from 0 to 65535.

If the data sent by the transport layer does not reach the recipient correctly, then it is the responsibility of the transport layer to resend that data and ensure that the data has reached the correct type and shape correctly.

**5. Network Layer:** Network Layer transforms the packet into the segments connected to the transport layer by adding information to the packet. The main function of the network layer is to send the data from one network to the other network, which is called routing. The table which is used by the network layer is called routing table, from which it determines which way would be better to send data if there is no way right for any reason is not the router selects one way of your writing table.

Network layer routing also gives addressing which we call IP addresses.

**6. Data Link Layer:** Data Link Layer Network layer transforms the packet from network layer to its frame by converting it into a frame that is a bits of bits. Data link layer also detects errors and also works for correction along with this, the Data Link Layer does two important functions and the following are:

1. How to use the medium
2. Assign the MAC Address

**7. Physical Layer:** Physical Layer transforms frames from data link layer into physical signals. It is also responsible for turning on the layer link, maintaining it smoothly and closing it.

In the computer networks, the data is generated from the application layer and processed from different layers and accesses the physical layer, it is later converted into signals. These signals move with the media to reach another device. Here the device returns these signals changes to the required information. This information comes back from the other layers of the application layer, which the person working or the user gets,

the application of data is taken from the layer to the physical layer called encapsulation and moving data from the physical layer to the application layer is called decapsulation.

Every layer in the network fulfills its work responsibly.

#### 4.9 Advancements of Networking

When two or more computers are added to share information and use of resources, then it becomes the network. A local area network, sometimes too far beyond the capacity of the transmission medium it has to be done that this network equipment helps local area network to reach long distances. This network device can be used for transmission of two different networks. Can also add common network equipment is as follows: -

**Modem:** Translation can happen to anyone from analog and digital when digital data is to be operated from the telephone line, then the digital data has to be converted into analog data because the analog data is transmitted faster Occurs. The sender's technique here that wants to convert digital messages to analog messages is called Modulation, the message recipient receives the inverse work, i.e. the analog signals are replaced by the demodulation technique digital signature.

Modem changes the digital signals of the computer to analog singles so that their translation can be transmitted from the telephone line to the model's signals in the sender's digital signals, thus with the help of Modern, two different computers can be translated by phone line. Modem's data transmission speed bits are measured in seconds.

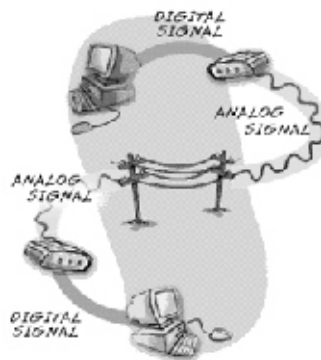


Figure 4.21 Modem Communication

**Hub:** Hub is a tool by which many computer in a network can be physically added. In this device first information is collected and later it is transmitted to computers so it is said interface unit. Only a few bits of data can be collected, so its transmission through more speed. It is a device that gets data first makes them powerful and lets them

transmit to the other computer because HUB make the data powerful so it also makes the signals unencumbered with the data powerful. It is a powerful wiring center which can be used printers, scanners, computer to connects with LAN. One device can only transmit data by hub at a time. Hub is a point by which the problem can be diagnosed by detecting it and its data transmission speed is 10 megabits per second. HUB are of two types -

1. **PASSIVE HUB:** This is a simple hardware tool. It can get data signals from multiple cables in the network.
2. **ACTIVE HUB:** It is a complex hardware tool that can test and control the information given by all different networks.

**Switch (SWITCH):** After seeing the address of the receipt of the frame, the switch starts sending frames to their destination. Do not wait for the entire frame to arrive before send it. It removes the errors of signals of data packets when data packets come in, the receiver is detected from their header and then they are sent to the receiver. In the frame the bits are applied in a predetermined order in which bits are detected for destination detection, fault control, receiver data, and information about the end of the frame.

Depending on how many computers can be added to a switch, it is known at its speed and depending on how many ports it has in it. Switches are the fastest and give different bandwidth for each device. The switches increase the efficiency of the network by reducing the additional traffic, and also ensure that all the devices can be transmitted at the same time. The switch keeps separate small buffer memory for all those devices connected. When it receives the data packet, it saves it in the buffer then after seeing its address, it converts it to its destination. If the location from which the data package has arrived and its address is the same, then the switch removes the data package. And does not transmit them, thus reducing the transmission of unnecessary data packets, it reduces data traffic and in other words the switch is a sensible hub .

### **Router:**

This device is used to transmit data between two networks. When any network is built after connecting two or more small network than router makes a table of all routes/paths which is used to identify route between two networks. Router uses best path to sent packets.

If any path goes down it sends packet via another route.

Router not only sends data it also chose best path. They send information to their neighbours which sent to his neighbours. It also works as firewall which stop un-



wanted packets.

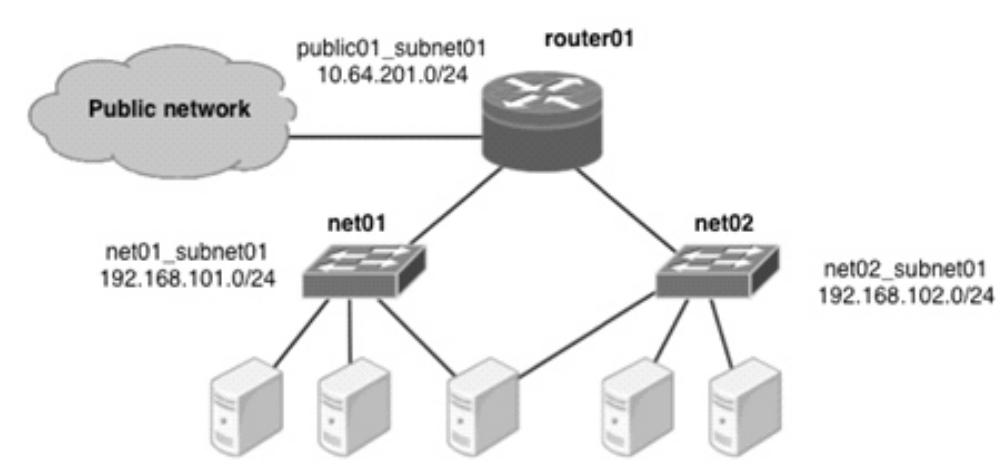


Figure 4.22 Router

**GATEWAY:** This tool connects to uneven network or some networks want information about how they can be streamlined after data arrives. When we connect two or more networks with two different operating systems, then the gateway are required. The address of the gateway messages and the necessary protocols change, they send them from one network to another.

The gateway transmits the group of instructions from the sender network to the instructions of the receiver network. The gateway usually has software in the router, the gateway can understand the instructions to all the networks connected to it, and it can change them from one to the other. The requests of the computers connected to the server are changed into instructions conveying the message of the server to the receiver the computer changes to the understanding of computers.

#### 4.10 Internet Protocol (IP)

IP addressing is an important function of the network layer which can make transmission possible even if it is in any network of its device or another network. IP (Version 4) and IP (Version 6) provide graded addressing for carrying data through both packets. Design, implementation and an effective IP plan will ensure the network will work effectively and efficiently.

It is also important to know that computers only work in Binary which is displayed by Zero (0) or One (1). The computer user / operator change the language given to it in the binary.

For example, the binary code for A written by ASCII standard is 01000001.

It is not necessary to know how words change in binary at this time, we need to

know the use of binary for IP address.

**IP (Version 4):** IP (Version 4) is a binary address of 32 bit. Packets senders and recipients on this network layer include this unique information. As a result, packet get its 32-bit sender and 32 bit recipient's address. Changing binary to decimal is based on mathematical positional notation. Positional notation, that is a digit gives different value according to the location. In the positional notation, the number is called the radix. Radix is 10 in decimal system.

We write the 32-bit binary address in the Dotted Decimals format because humans cannot read and remember the binary properly and the computer does all its work in binary.

This 32-bit binary address is shown in a group of 8-bit or octet, and the group of each 8 bit remains separate from a dot (dot).

For example, 11000000 10101000 00001010 00001010 can write binary address in dotted decimals 192.168.10.10. Binary number system have Radix 2, so the number is either 0 or 1. Posts in these 8-bit binary / binary numbers represent these quantities:

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| 128   | 64    | 32    | 16    | 8     | 4     | 2     | 1     |

|                   |     |    |    |    |   |   |   |   |
|-------------------|-----|----|----|----|---|---|---|---|
| Radix             | 2   | 2  | 2  | 2  | 2 | 2 | 2 | 2 |
| Exponent          | 7   | 6  | 5  | 4  | 3 | 2 | 1 | 0 |
| Octet Bit Values  | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| Binary Address    | 1   | 1  | 0  | 0  | 0 | 0 | 0 | 0 |
| Binary Bit Values | 128 | 64 | 0  | 0  | 0 | 0 | 0 | 0 |

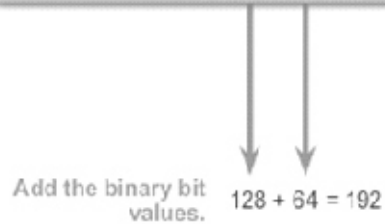


Figure 4.23 Internet Protocol

Every octet is made of 8 bits in which the bit is either 0 or 1. The quantity of 4 bits or 8 bits is in a series of 0 to 255. The value of the location of each bit is directly

opposite direction i, 1, 2, 4, 8, 16, 32, 64 and 132. If decimals are to be replaced by binary, then divide the given value according to the numbers mentioned above, in place of the number of those obtained from them, place 0 and 1 in place of others.

For example, if the 155 is to change binary I would have the following split

| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 128   | 64    | 32    | 16    | 8     | 4     | 2     | 1     |
| 128   |       |       | 16    | 8     |       | 2     | 1     |
| 1     | 0     | 0     | 1     | 1     | 0     | 1     | 1     |

**155's binary will be 10011011**

If any binary change in decimals then they can change according to above. Binary values are the value of the bit I value, keeping it according to its position and add it to the end.

The IP address has two parts in which one part is related to its network and the host.

IP Addresses are divided into 5 classes

Class A

Class B

Class C

Class D (Multicasting)

Class E (Reserved for the future Development and Research)

The class of that IP is detected from the first octet of any IP. The range of IP ranges from 0 to 255, which is as per the classes.

Class A: 0-127

Class B: 128-191

Class C: 192-223

Class D: 224-239

Class E: 240-255

For example, class of 10.10.12.50 is A because its first octet is 10 which come in the range of class A.

The subnet mask determines how much of the IP address belongs to the network, and how much the host. According to each class, different subnet masks are defined as follows.

Class A: 255.0.0.0

Class B: 255.255.0.0

Class C: 255.255.255.0

Class D: Not defined

Class E: Not defined

In class A the first octet is 255 which means values of 8 bits are 1, so the first octet of the IP address will be associated with the network, after which the remaining 24 bit will remain associated with the host. Similarly, the first and second octets of class B are 255, which mean values of 16 bits are 1, so the first and second octet of IP address will be associated with the network, after which the remaining 16 bit will remain associated with the host. Similarly, the first, second and third octets in class C are 255 means the values of 24 bits are 1, so the first, second and third octet of the IP address will be associated with the network, after which the remaining 8 bit will remain associated with the host.

IP (Version 6): Slowly as the Internet user started to grow, the address coming in the IP V4 started to end and the need started to increase so the IP (Version 6) was developed. The IP V6 does not have the concept of a class. This is the address of 128-bit which is written in Hexadecimal.

This is an example of the 2001: 0db8: 85a3: 0000: 0000: 8a2e: 0370: 7334 IP V6 address. This address can be further shortened leading zeros in a group can be removed. One or more consecutive groups of zero values can be replaced with a single empty group using two colon consecutively.

2001: db8: 85a3: 0: 0: 8a2e: 370: 7334

2001: db8: 85a3:: 8a2e: 370: 7334

## 4.11 MAC Address

MAC Address is a unique and physical address given to any network card for physical network for communication. The MAC address is used by IEEE Network technologies such as Ethernet and Wireless and works on the data link layer. The MAC address is given by the organization that creates any interface card and the read-only chip on the hardware is stored it and cannot be changed. MAC are made according to the standards of Institute of Electrical and Electronics Engineers (IEEE) and is the following - MAC-48, EUI-48 and EUI-64

The MAC address is also called a burn address or hardware address. MAC Address is the address of 48-bit which is arranged by 6 groups of 2 numbers of hexadecimal. These groups are separated from Hyphen (-). Mac address is usually divided into parts, the first 3 groups are given to an organization by the Institute of Electrical and Electronics Engineers (IEEE) and the last 3 groups are like serial numbers on the cards created by any organization, thus any MAC Address does not match any other MAC address.

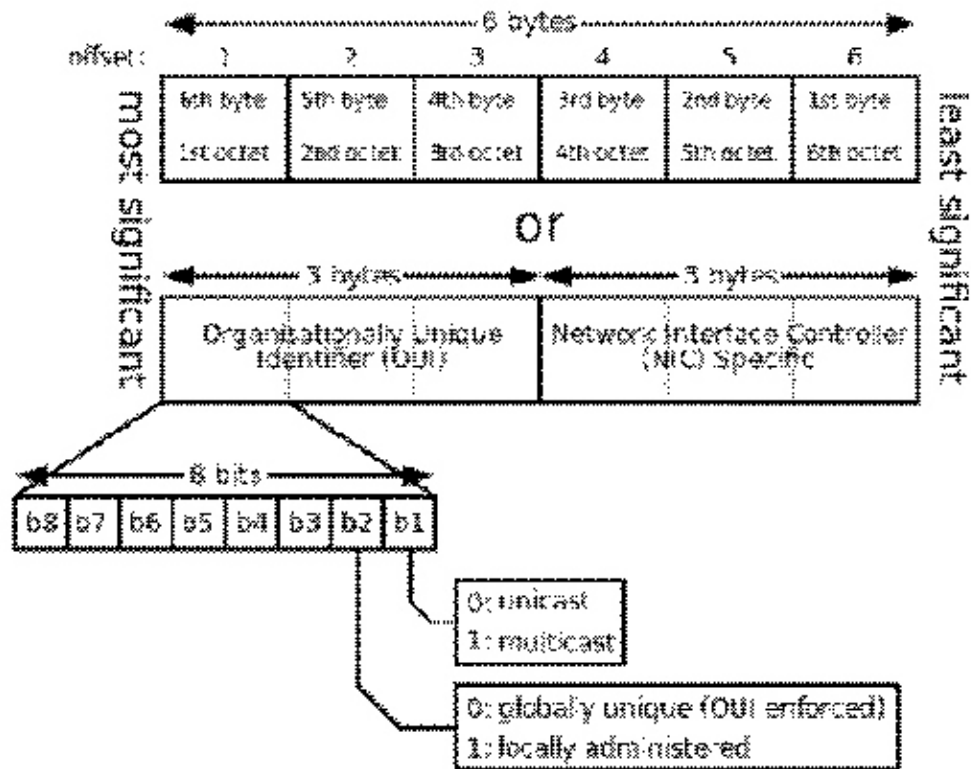


Figure 4.24 MAC Address

## 4.12 Subnetting

It is already known that the network runs well and smoothly with good planning, implementation and good management of IP. IPV4 have two types of hierarchy that are related to the host and the network. Any router sends any packet from one network to another network with the information related to the network. Once the network portion is detected, the host address only works for the address of the computer to which to send the packet. As the number of networks increases in organizations like this, the two types of taxation seem insignificant. It is now necessary to redistribute the network gradation. Which network can now divide three orders - Network, Subnetwork and Host data packets are transmitted rapidly by dividing the various modules of any network and it creates a cosmic network.

It is also known to us that the part of the IP is related to the network and which part is related to the host, and the subnet mask is done according to the classes and the subnet masks are already there.

**Netting of any network:** Every network has a valid series of host addresses. All the computers or devices or hosts present in the same network keep the same subnet mask and are the members of that network. IPV4 address has 32 binary bits that are related to the network and the host. The Subnetting is done by giving the host bits to the network. How many subnetworks of any network will be made depends on how many bits are given from the host by network or how many network networks have taken over the host.

For example, if 1 bit is taken then 2 subnetworks, 2 bit is taken then 4, 3 bit is taken then 8 subnetworks will be created. As host bits will be reduced, any network address will also be reduced. A network of class C consists of 24-bit network and 8 bit host if host is given 8 bits to 1-bit network, then the host will now be able to save 128 host addresses, which means that earlier this network was 256 hosts and this was a network. Now this network has broken into two pieces and hosts are now broken into pieces. There will be 128 - 128 host addresses in every subnetwork and now the network will be 25 bits than 24. Network bits can be written as slash (/) after any address such as 192.168.1.0/24.

"In the simple language it said, that the subnetting is transfer of bit from host network".

Any network is the first unique network addresses that represents any network and the final address is Broadcast Address which is used to send information to all the devices in any network. According to Manako, network address and Broadcast address cannot even be assigned to the device. The address of this can be assigned to

computers or other devices. According to this, 2 addresses cannot be taken from any network.

For example, if the host part of a network is 8 bit, then there will be a total of 256 host addresses, 2 of which cannot work, then only 254 addresses can be used. All these are calculated according to their class because in every class there are different types of bits hosted by class.

The question of subnetting can be asked in two ways - according to the network requirement and according to the host's requirements.

The question is according to what is required it is calculated according to how many bits are to be placed in the host part or how many bits have to be given to the network part. Calculate the number of which it is needed in the power of 2, giving or keeping the same bit according to it.

We will consider subnetting in some examples -

1. 192.168.1.0/24 Network subnetting in such a way that its four networks will be created

Network class:

Subnet mask of class:

How much is to keep or give:

How many networks were created?

How many host networks are left?

When submitting this network, we have to answer the questions written above.

**Remedy:** The class of the network is C, which is identified by the first octet of this address. The subnet Mask of Class C is 255.255.255.0 which we have already read. We will estimate how much bit to take or take, we need to create 4 subnetworks, and we need to give 2 bit network to 4, 2, 2. This network of sub-networking has 24 bit network and 8 bit host, now we have to give it to the network from 2 bit host, then the network will be 26 bit and the host will be 6 bit. Network address can now be written as follows 192.168.1.0.26.

Now 4 parts of this network or subnetwork are created and host address is equal to equal parts. The host part has 6 bit remaining, that means 2 to the power 6,

each network will have a total of 64 addresses, 2 of which cannot be used, 62 addresses can be assigned to computers or devices.

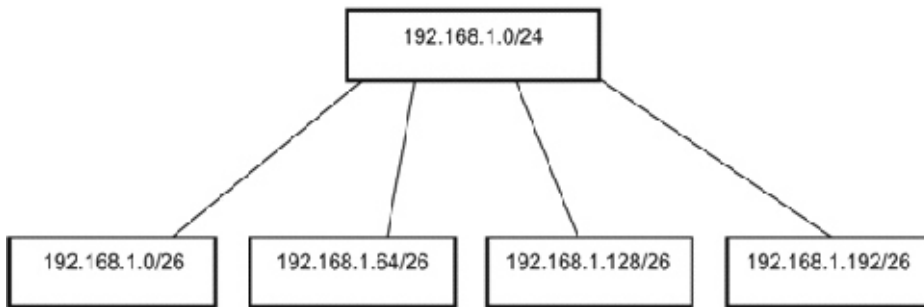


Figure 4.25 Subnetting

### 4.13 Planing of IP address

An IP address plan is an important task that needs to be done in the right way. IP address is a very valuable resource in the network of any organization, which needs to be used as per the need. New computers or devices cannot be added if any network address is over.

In the conventional subnetting all the subnetworks are given the same number of addresses. It is good only when all the essentials of the networks are usually not done.

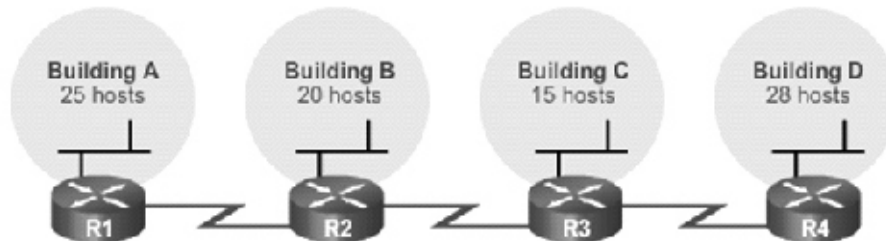


Figure 4.26 Networks with the need of a different host

According to Figure 4.26 a total of 7 networks are needed with 3 wide area networks (WAN) and 4 local area networks. According to conventional subnetting, if divided, the network will be required to deliver from the 3 bit host which will create total 8 networks and every network has a power of 5, 32 addresses and our need is met.

But by dividing this way many addresses are useless which will not be used. According to Figure 4.26, there are 3 wide area network in which 2 router is con-



nected, 2 addresses will be required but every interface of the router becomes a separate network, then 2 other address (network and broadcast), total 4 is required but 32 addresses are being given, resulting in 28 addresses being lost.

It can be saved from variable length subnet mask (VLSM) technology. VLSM technology helps to divide any network in an unequal manner. VLSM technology is similar to traditional subnetting, but in the past it is subnetting a network, and then subnetting the subnet network again, which gives network of uneven subnet masks and the absence of usage of the addresses ceases to an extent is there.

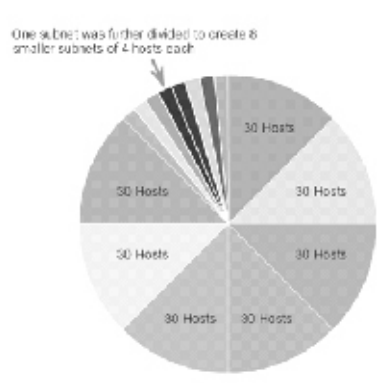


Fig. 4.27 Network subnetting by VLSM

#### 4.14 Wireless

Wireless networks allow any device to bundle without any cable. Wireless LAN (WLAN) is a classified part of wireless technology that is usually used in homes, offices and campus. This technique works on radio waves for instead of wires. It can be added to an already-connected LAN network, so that the user can use the wireless facility anytime in that area. This technique is similar to Ethernet.

**Wireless technology and standards:** Productivity in today's time is no longer restricted to a certain place or work for a set time period people are now connected to the office at any time and place for the airport or home now employees can see their emails, voice messages, and new information anytime. Users now expect someone to be connected to the Internet without interrupting the connection.

Wireless has the following advantages -

1. Flexibility to do work, increase productivity, move forward and create a friendly environment as needed.

2. Any user can connect anytime to the internet. There is no need to be in any one area or place.
3. Reduction in the cost of productivity coming from wireless.

**It has the following technology**

**1. Wireless Personal Area Networks (WPAN):** This technique works only in the area of few feet. Bluetooth is an example. With the help of Bluetooth, the user who has the mobile can always send the data to each other.

**2. Wireless LANs (WLANs):** This technique works only in the area of some 100 meters. This can be done separately from the internet at home, office and campus.

**3. Wireless Wide-Area Networks (WWANs):** This technique works in a very long range area. This technique is connected to home, city, and country. Satellite and mobile communications are examples of this.

All wireless devices work in the range of radio waves of electromagnetic spectrum. Regulation and allocation of radio frequency is the responsibility of the International Telecommunication Union (ITU). Variations of different frequencies and bands are allocated for various purposes. This frequency comes mostly after payment, while some frequency is free, such as Industrial, Scientific and Medical (ISM) and the National information infrastructure (NII) frequency bands.

WLAN (Wireless LAN) works at 2.4 GHz of ISM band and 5 GHz of NII. Wireless Communication works in the magnetic spectrum of 300 GHz from 3 HZ of radio waves.

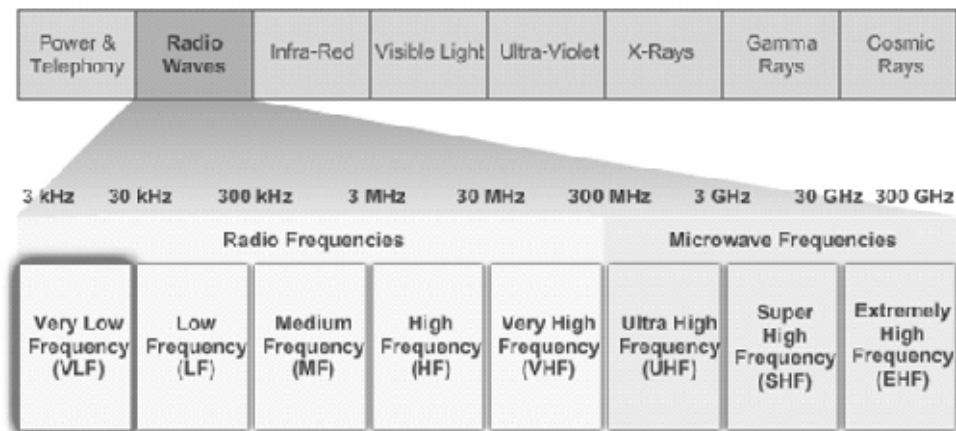


Fig. 4.28 Magnetic spectrum

Wireless lenses near these devices have transmitters and receivers to work on select frequencies which are low -

2.4 GHz (802.11b / g / n / ad)

5 GHz (802.11a / n / ac / ad)

60 GHz (802.11ad)

#### 4.15 Standards of Wireless

**IEEE 802.11:** The WLAN standard defines how the unrecognized ISM frequency band (RF) is used for the mac layer of the physical layer and wireless link in the frequency band.

Various implementation of IEEE 802.11 standard has been developed over the years. The following are highlighted on these standards: -

**802.11:** This was developed in 1997 and obsolete. This technique works at 2.4 GHz and provides a speed of 2 Mbps normally LAN used to work on 100 mbps while it worked on 2 Mbps and an antenna that used to be the sender and recipient of the equipment used in it.

**IEEE 802.11a:** It was released in 1999. It works on a low crowd frequency of 5 GHz and speeds up to 54 Mbps. It works on more frequencies so its area is reduced and it lessens the walls.

**IEEE 802.11b:** It was released in 1999. It works on the frequency of 2.4 GHz and provides speed of 11 Mbps.

**IEEE 802.11g:** It was released in 1993. It works on the frequency of 2.4 GHz and speeds up to 54 Mbps. This previous technology is compatible with IEEE 802.11b.

**IEEE 802.11n:** It was released in 2009. This works on both frequencies. 2.4 and 5 GHz. This technique speeds 150 to 600 Mbps. It has more than one antenna on the device coming to work which has MIMO (Multi in Multi Out) technology so that they transmit extra data transmission. It supports up to 4 antennas. This is compatible with previous technologies 802.11a / b / g.

**IEEE 802.11ac:** It was released in 2013. It works on the frequency of 5 GHz and gives speeds of up to 450 Mbps to 1.3 Gbps. This is compatible with previous technologies 802.11a / n.

**IEEE 802.11ad:** It was released in 2014 and is also called WiGig. It works on the frequency of 2.4, 5 and 60 GHz. And can speed up to 7 Gbps so this is the fastest wireless technology.

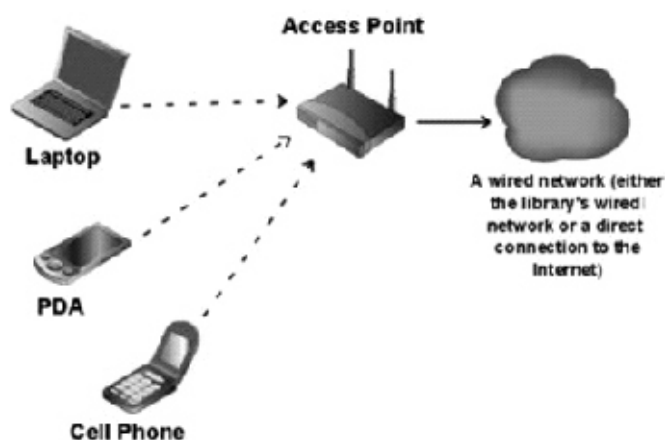


Figure 4.29 Wireless Network

**Wireless Security:** It is difficult to secure a network connected to a wireless network. Security is an important priority for anyone who is using network administration.

The wireless network was related to the intelligence information as far as the network is approaching or remains open. An attacker may not have to physically enter the workplace to gain access to a WLAN. Therefore it is very important to protect the wireless network.

Wireless attacks can occur as follows -

- Wireless intruders
- Rogue apps
- Interception of data
- DoS attacks

### **Ways to Protect Wireless (Securing WLANs)**

Security is always a matter of concern as the range of network limit varies. Wireless signals can travel through the concrete medium of the roof, wall, outside of the house or office. Wireless without any security is equal to leaving any Ethernet network in open use.

The following security arrangements to provide wireless security -

**SSID cloaking:** In this, the beacon frames left by the wireless device are discontinued so that the wireless network is not visible in any device. The device to connect to the wireless network can be added only after complete information of the wireless so no network can detect it.

**MAC address filtering:** In the wireless router, you can create a list of physical MAC Address so that any computer or device can be switched off to accommodate that network.

Both of the above methods can be broken so that the wireless network is required to encrypt and authenticate, therefore 802.11 standard has two types of methods which are low -

**Open system authentication:** Any customer or user can easily connect to it where security does not matter much. This kind of safety cafe, as the hotel offers free internet access in places, where security is not kept safe.

**Shared key authentication:** In order to encrypt the communication between the computer and the router, a key or password is inserted in the router without which any customer or user can connect without informing. The following security arrangements are for encrypting or hiding data.

**WEP (Wired Equivalent Privacy):** This is the first generation of authentication. It uses the key as a password. This takes the RC4 algorithm to work.

**WPA:** This uses the Temporal Key Integrity Protocol (TKIP) encryption algorithm for strong security.

**WPA2:** This is a way of providing security to the wireless network according to the industry standard; it takes AES (Advance Encryption Standard) instead of TKIP, which is a powerful way.

#### **4.16 Congestion Control**

Congestion control is the technique and method which fixes before the crowd of packets in the network meaning it does not happen or it fixes it after it is done. It is of two types

1. Open loop Congestion Control (Prevention)
2. Close Loop Congestion Control (Removal)

**Open Loop Congestion Control:** In different ways different strategies are used to prevent congestion. In this technique the congestion is either stopped by the sender or the recipient's place.

The following policies are used in this technique:

1. **Retransmission Policy:** In this policy if the sender feels that the packet sent by him has been destroyed on the way then that packet is re-sent but it can generate congestion. Good policy is required to stop this. In this a clock is used so that there is no congestion. The TCP protocol is already made in such a way that the possibility of congestion will be less.

2 **Acknowledgment Policy:** In this policy if the sender expects a confirmation of every packet sent, if any packet is not confirmed on receipt of the recipient, and then reduces the speed of sending the sender packets. This reduces the probability of congestion.

**Close loop congestion control:** After the conjunction of open loop congestion control it is rectified, various techniques are used by different protocols.

1. **Backpressure:** The node congestion that is in this technique stops receiving the data from the node before it which can increase the angle on the first node, but do the same as the first node is there. This is called node-node congestion control. This echnique is only used in the virtual circuit network

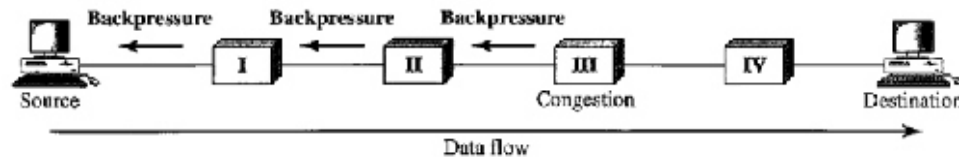


Figure 4.30 Backpressure Congestion Control

2 **Choke Packet:** In this technique a special type of packet choke packet is sent to the sender. It is almost similar to backpressure technology but sent to the node before it in Bekpresure, the information is sent directly to the sender and information is not sent to the middle node.

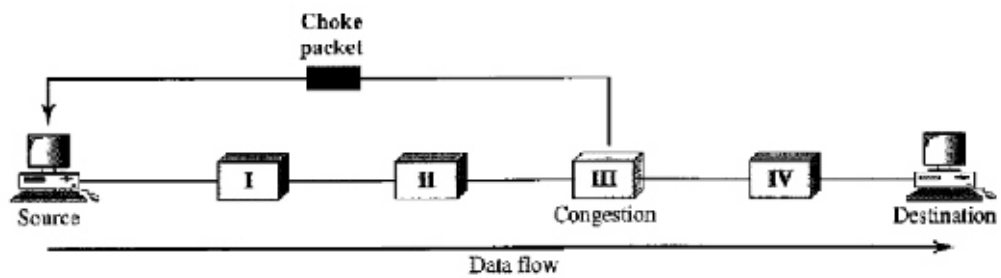


Figure 4.31 Choke packet congestion control

## 4.17 Quality of Service

The quality of service in computer networks means Resource Reservation for any type of communication. The quality of service is to give priority to high quality of any user, application and data flow. According to quality of service certain factors such as bit rate, delay, jitter and bit error rate should be according to quality.

Generally speaking the quality of service is meant to provide a high level of service.

**Flow characteristics:** There are mainly 4 types of flow characteristics

1. **Reliability :** Reliability is a feature. If the reliability is low then its direct intention is not to destroy the packet / acknowledgment which inspires retransmission.
2. **Delay:** The delay between the recipients from the sender can be tolerated to a certain extent in some year. But some applications such as telephony and audio conferencing cannot be delayed.
3. **Jitter:** The relation of jitter is delayed in the same packet related to the packets. If all the packets are reaching the distributor from the sender to the seller at the same time, not the jitter, but if the packets are reaching different delays, then the jitter is for the packets going to the network and not good for the user.
4. **Bandwidth:** The direct connection of bandwidth is at the speed of packets sent at one time. Separate different types of bandwidth are needed. If the bandwidth is more than or equal to the required bandwidth, the application will work properly otherwise the packets will start to be destroyed, which in turn will result in the packets that are not good for the network and computer users.

### 4.17.1 Techniques for improving the quality of service:

1. Scheduling

2. FIFO Queuing
3. Priority Queuing
4. Weighted Fair Queuing

#### **4.18 DNS (Domain Name Service)**

**4.18.1 Introduction :** DNS is a hierarchal distributed system which changes the domain name to IP. The computer does not understand the man made name and works in the binary while the human does not understand binary and works in own language so it becomes necessary that there is a system that can convert the names of human language into the address of the computer network address.

For example, IP 216.58.196.3 of www.google.co.in. It is possible from DNS.

**4.18.2 History:** In the days of internet, there were some computers in the world whose IP address was known to each other. Later, these computers were given the names of human or human languages which were converted to the IP from the computer's operating system File Hosts (Hosts.txt) in which the IP address was written in front of the computer's name.

But as the Internet promotion started, this work also started to become smile so now there was a need for a system to do this easily.

Paul Mocarpetris created the Domain Name System in the University of California, Irvine in 1983 and wrote the first implementation at the request of John Postell from ISI.

**4.18.3 Domain Name Space:** Domain Name System is a tree-like data structure. This tree structure is divided into several joints which start with a root or dot (.). The root domain or dot (.) is divided into several categories called top level domains.

For example com, org, net

Under the top level domain, there are secondary level domains in the pedestal structure that the user can choose their own requirements. A domain name occurs in all categories if a domain name is not in a category it can be taken from another category. Secondary level domains are divided into subdomains which the user can do according to their wish.

mail.google.com



Here is the mail subdomain, Google is the secondary level domain name and com dot top level domains, and all of this is divided by dot (.)

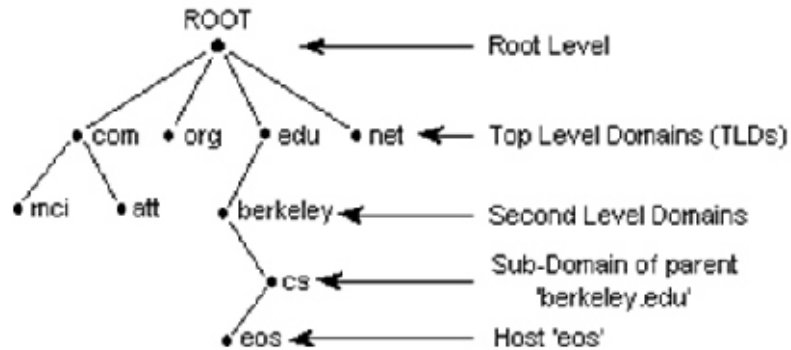


Figure 4.32 Domain Name System

Each DNS server stores information about any domain in the zone file. Any server has the following resource record (Resource Record).

1. NS record: This record keeps information on any domain's authority.
2. A record: This record keeps the details of the IP of any domain.
3. CNAME record: This record keeps information about the domain name of any domain, its other name.
4. MX records: It keeps information about its mail record.

Every computer or network device is inserted into the DNS entry so that it can locate the IP from the domain name. If no computer or network device has the entry of DNS, it will not be able to locate or contact any other computer or device by the domain name. The following is the method of IP address of any computer or network device.

Any computer contacts the DNS server in its DNS entry for any other computer, and asks its own question (domain's IP) if that DNS server has the address of that domain then it answers.

If that server does not respond to that domain, then it asks for itself or the latter of the DNS server.

If any server does not respond then the root server is asked.

#### 4.19 Email System

Email is the most popular service of internet services. In the early days of the internet only small and text messages could be sent by email. But now a days email is a very complex system. With the help of mail, text, images, video can also be sent. Email information can be sent simultaneously to more than one recipient.

**Email service:** Email has the following services:

1. It can be easily used.
2. Information can be sent to any corner of the world in a few seconds, so it works very fast.
3. When someone has to answer an email it keep up with the information already coming.
4. If a user is not able to use his email or is out of work he can use the email sending technique in which the user who has email will automatically go to the mail, whatever information the first store is.
5. Older postal system used paper while email uses electronic data so it is environmentally friendly.
6. Email can be read and respond to any device such as computer, laptop, mobile, tablet.

**Architecture of Email :** Before understanding the architecture of email it is necessary to understand some of the key elements.

1. **Mail User Agent:** This is an application or program that lets the user write and send mail.
2. **Mail Transfer Agent:** This is usually a server that serves to send email to each other.
3. **Mail Delivery Agent:** This is the mail server's technology in which the address of the receiver of the mail is given to him in the inbox.
4. **Post Office Protocol :** This is the protocol to receive mail from the server or the user's application or program from the server.
5. **SMTP (Simple Mail Transfer Protocol):** It comes with a user's program or application to send email to server and server to another server.

6. **Internet Message Access Protocol:** This is a new way of achieving a new one which uses some modern technology.

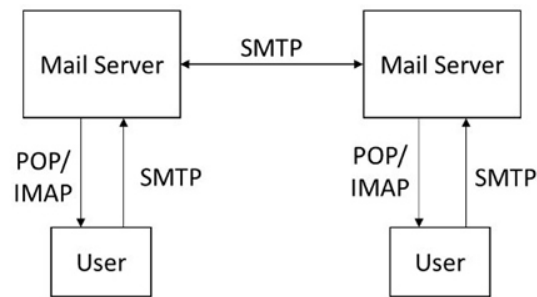


Figure 4.33 Mail system

We will try to understand the architecture of the email from Figure 4.33. as written below -

7. Writes a user with any user program or application or adds an image that is called attachment. When writing an email some important information such as the address of the recipient, addresses the subject of the email which is called the email header.
8. Now this created message is sent with the help of SMTP protocol to the email server whose service is taking the user.
9. Now the server matches the address of the recipient's domain to the domain of the email header sent by the sender. If the recipient's domain matches the domain of the server then the server finds the user's name in its database. After getting the name the server enters the mail in the Inbox's Inbox while working as a mail delivery agent, which is read by receiving it via the POP / IMAP protocol.
10. If recipient and recipient's domain is not available then the server sends that email to its related server via SMTP protocol. Now the second server also uses the above method to put email in the inbox of the user.

**Difference between POP / IMAP:** Both POP / IMAP protocols are used to bring email from the server to the user. But there are some technical differences which are below: -

1. As soon as the user application or program opens the user name and password, then the POP protocol downloads the entire email to the server from the server / client (Client) while POP works in two types, either one of the email the copy keeps the server or after download it destroys the copy. POP downloads the

entire email with an attachment.

2. IMAP is more powerful and complicated protocol than POP and gives more features to the user.
3. IMAP only downloads the email header instead of downloading the entire email so that the user knows that the mail has come from and what is his subject.
4. If the user can download an email without any email in his inbox, he can find it.
5. Do not download the entire mail saves bandwidth.
6. Users can create new folders according to their convenience.

### **Important points**

1. The style of connecting computers is called topology.
2. Network can be broadly divided into three types - LAN, MAN, WAN
3. Broadcasting medium is the path on which the sender and recipient exchange information.
4. Fiber optic medium transmission is the fastest path.
5. Protocol is a set of rules.
6. All the tools in the star topology are connected to a central device.
7. OSI is a reference model and TCP / IP is protocol model.
8. In a simple mode the data can go in the same direction, the recipient can't send data back to the sender.
9. Microwave signals are broadcasted by antenna mounted on buildings.
10. OSI models carry 7 layers and 4 layers in TCP / IP.
11. Coaxial cable used in the television.
12. An analog signal is used to send the sound.
13. Computer does its work in binary.
14. Router is used to send packets from one network to another.

15. IPV4 is 32 bit binary address and IPV6 is the address of 128 bit.
16. The IP address of the network layer is the addressing scheme.
17. MAC address is a physical address which does not change. It will be on read only chip on the hardware.
18. Subnetting is done to split a large network into smaller networks.
19. Modem changes analog signals into digital signals and digital signals into analog signals.
20. In Network, hub is a powerful wiring center.
21. The gateway transmits the group of instructions from the sender network into the instructions of the recipient network.
22. 32bit binary address in IPV4 is shown in a group of 8 bits or octets, and each 8 bit group remains separate from a dot (dot).
23. IPV4 has 5 classes. These are A,B,C,D and E
24. Wireless network allows any device to interact without any connection. Wireless LAN (WLAN) is a classified part of wireless technology that is usually used in homes, offices and campus.
25. Conjunction control is the technique and method which fixes the crowd of packets in the network even before it does not mean it does not happen or it fixes it after it is done.
26. The quality of service is to give priority to high quality of any user, application and data flow.

### **Exercises**

#### **Objective Type Questions**

1. Which one is the transmission medium?  
(A) Modem (B) Multiplexer  
(C) Hub (D) Coaxial Cable
2. The oldest and more usable transmission line is?  
(A) Coaxial Cable (B) Fiber Optic  
(C) Twisted Pair (D) None of the above

3. What is WAN?  
(A) Wire Area Network (B) Local Area Network  
(C) Wide Area Network (D) Wire Accessible Network
4. How much layers the OSI model have?  
(A) 4 (B) 2  
(C) 7 (d) 5
5. Which tool sends packets from one network to another?  
(A) Router (B) Hub  
(C) Switch (D) Gateway
6. What kind of transmissions does the wave go in all directions?  
(A) Radio link (B) Microwave  
(C) Infrared (D) Satellite
7. Which layer of TCP / IP model works as transportation?  
(A) Application (B) Transport  
(C) Network Access (D) Internet
8. Which device do we use to increase the power of analog signal?  
(A) Amplifier (B) Transmitter  
(C) Repeater (D) Transponder
9. Which of the following equipment is used in the telephone line?  
(A) Router (B) Modem  
(C) Switch (D) Hub
10. Which of the following can also work as a firewall?  
(A) Router (B) Modem  
(C) Switch (D) Hub
11. In which numbers system computers writed IP address?  
(A) Binary (B) decimal  
(C) Hexadecimal (D) None of these
12. How much bits in IPV6?  
(A) 128 (B) 64



2. Write about two types of signals?
3. Why is planning of IP address is required?
4. Why we do subnetting.
5. Explain the MAC address.
6. Explain the choke packet system.
7. What is the jitter in the network?
8. Explain the mail transfer agent.
9. Why is DNS required?

### **Essay Type Questions**

1. Write the details of OSI Model in short.
2. Explain the data transmission.
3. Explain the transmission media.
4. What is satellite transmission? Explain.
5. Explain the IP in detail.
6. Explain modem's methodology with Diagram.
7. Explain subnetting with examples.
8. What is Congestion Control? Explain in detail.
9. Explain the process of sending email.

### **Answer Key**

1. D   2.C   3. C   4. C   5. A   6. A   7. B   8. A   9.B   10. A  
11. A   12. A   13. B   14. C   15. C



### 5.1 Introduction of Markup Language

Markup language is concern about code and Tokens, which is arrange in a single document. which describe the data. In other words markup is used to organize Data. For Example HTML is a markup language which is used to develop web pages.

Example 5.1

```
<html>
 <head>
 <title>hello from HTML</title>
 </head>
 <body><center><H1>HTML document</H1></center>
 This is an HTML document
 </body>
</html>
```

In above example represents that HTML is using to interpret the important part of HTML pages such as header ,body and footer etc. This HTML page may be interpreted by Internet explorer, Google chrome and firefox. HTML markup may created using HTML tags which are predefined.

HTML tags directed to browser to represent data of document. In the same way XML is also used to store and represent data.

## 5.2 XML Introduction

XML language is developed by World Wide Web Consortium. XML is kind of markup language. XML Language is also used to represent and store the data as well as HTML. Data transformation at the internet is also accomplished by XML. XML Text is written in a such a way that human and machine can easily understand. It is self-descriptive language and a subset of standard generalized markup language (SGML). This kind of markup language is used in storage of large data.

### 5.2.1 Difference between HTML and XML

HTML	XML
HTML is an abbreviation for HyperText Markup Language.	XML stands for eXtensible Markup Language.
HTML was designed to display data and used to make attractive web pages.	XML used to transport and store data
HTML is a markup language itself.	XML provides a framework for defining markup languages.
HTML is case insensitive	XML is case sensitive.
HTML is used for designing a web page to be rendered on the client side at client browser.	XML is used basically to transport data between the application and the database.
HTML has its own predefined tags.	custom tags can be defined and the tags are invented by the author of the XML document.
HTML is about displaying data, hence static.	XML is about carrying information, hence dynamic.

### 5.2.2 XML data Structure

XML is either storage of data or also capable to define structure of data. XML structure is also capable to work with Complex data.

For Example if we store large amount of account details in HTML, which in-

crease changes of HTML errors. In XML we can create rules of syntax which instruct to definition of document. Due to this causes of errors in documents are almost removed. XML processor examines XML texts and two important factors are also examined by XML processor. which are as follows.

1. XML must be well formed.
2. XML must valid.

### **5.2.3 XML Features**

1. Excellent for handling data with a complex structure or a typical data.
2. Text data description
3. Excellent for long-term data storage and data reusability
4. Human- and computer friendly format
5. Data described using markup language
6. Handles data in a tree structure having one and only one-root element

## **5.3 Development of XML document**

### **5.3.1 XML declaration**

XML document declaration is compulsory which represent as follows.

```
<? xml version = "1.0" encoding = "UTF-8" ?>
```

In above declaration represent XML version and encoding is represent to useful characters encoding.

XML declaration helps to XML processor to parse XML documents. XML declaration is compulsory which is always written in first line of document.

### **5.3.2 Rules of XML Declaration**

- If the XML declaration is present in the XML, it must be placed as the first line in the XML document.
- If the XML declaration is included, it must contain version number attribute.
- The Parameter names and values are case-sensitive.

- The names are always in lower case.
- Either single or double quotes may be used.
- The XML declaration has no closing tag i.e. `<?xml>`

### 5.3.3 XML Tags and elements

For XML elements three types of tags are used majorly.

1. Opening Tags
2. Closing tags
3. Empty Tag

#### 1. Opening tags:

The beginning of every non-empty XML element is marked by a start tag. Following is an example of start tag –

Example: `<localaddress>`

#### 2. Closing tags:

Every element that has a start tag should end with an end tag. Following is an example of end tag .

Example: `</localaddress>`

#### 3. Empty Tags

The text that appears between start tag and end tag is called content. An element which has no content is termed as empty. An empty element can be represented in two ways as follows -

A start tag immediately followed by an end-tag as shown below -

`<hr></hr>`

A complete empty element tag is as shown below -

`<hr />`

Empty element tags may be used for any element which has no content.

### 5.3.4 XML Attributes

Various attributes are included in XML element. Attributes are created for various information of XML elements. XML attributes consist of name and its value.

Syntax :-

```
<element-name attribute1 attribute2 content.....>
</element-name>
```

Example

```
<student firstname="Mahesh">
 <name>Sharma</name>
 <grade>A+</grade>
</student>
```

### 5.3.5 XML comments

XML comments are same as HTML comments. To understand XML documents comments are used. XML comments do not play any role in XML code execution.

Syntax –

Single line Comments

```
<!-------Your Comment----->
```

Multi Line Comments

```
Comment <!------->
```

Example –

```
<?XML Version = "1.0" encoding = "UTF-8"/>
<!--Student grades are updated monthly----->
<classlist>
```

```
<student>
 <name>Ramesh</name>
 <grade>A+</grade>
</student>
<name>Girish</name>
<grade>A-</grade>
</student>
</classlist>
```

### 5.3.6 XML Entities

An XML documents are created by different storage units which are called entities. Each XML document has one entity which is called document entity. XML entity work for XML processor as starting point and placeholder.

There are different types of entities in XML but here we will learn only character entities.

They are introduced to avoid the ambiguity while using some symbols. For example, an ambiguity is observed when less than (<) or greater than (>) symbol is used with the angle tag (<>). Character entities are basically used to delimit tags in XML. Following is a list of pre-defined character entities from XML specification. These can be used to express characters without ambiguity.

- Ampersand - &amp;
- Single quote - &apos;
- Greater than - &gt;
- Less than - &lt;
- Double quote - &quot;

In this way main characters are represented through character entities.

## Types of Character Entities

There are three types of character entities -

1. Predefined Character Entities
2. Numbered Character Entities
3. Named Character Entities

### 5.3.7 XML Features

1. XML separates data from HTML
2. XML simplifies data transport
3. XML simplifies data sharing
4. XML is not dependent on operating system.
5. XML can be used to create new internet languages

## 5.5 XML DTD

The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements. XML DTD may be used by following two ways.

An XML DTD can be either specified inside the document, or it can be kept in a separate document and its reference will be passed into original XML document.

Syntax –

```
<!doctype element DTP identifier
```

```
[
```

```
 Declaration 1
```

```
 Declaration2
```

```

```

```
]>
```

In the above syntax,

- The DTD starts with `<!DOCTYPE delimiter>`
- An element tells the parser to parse the document from the specified root element.
- DTD identifier is an identifier for the document type definition, which may be the path to a file on the system or URL to a file on the internet. If the DTD is pointing to external path, it is called External Subset.
- The square brackets [ ] enclose an optional list of entity declarations called Internal Subset.

DTD declaration is two types, as follows

1. Internal DTD declaration: If the DTD is declared inside the XML file, it must be wrapped inside the `<!DOCTYPE>` definition.

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Mahesh</to>
<from>Jaipur</from>
<heading>Reminder</heading>
<body>Please remember it</body>
```



```
</note>
```

In above example.

- !DOCTYPE note defines that the root element of this document is note.
- !ELEMENT note defines that the note element must contain four elements: "to,from,heading,body".

## 2. External DTD Declaration

If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file:

```
<?xml version="1.0"?>

<!DOCTYPE note SYSTEM "note.dtd">

<note>

 <to>Mahesh</to>

 <from>Jaipur</from>

 <heading>Reminder</heading>

 <body>Please remember it</body>

</note>
```

### Note-dtd

```
<!ELEMENT note (to,from,heading,body)>

<!ELEMENT to (#PCDATA)>

<!ELEMENT from (#PCDATA)>

<!ELEMENT heading (#PCDATA)>

<!ELEMENT body (#PCDATA)>
```

In above example Note-dtd is an external DTD file whose reference is given in XML file. Different attributes of Note-dtd like Note ,to, from ,heading, body are used in original XML document.

## 5.6 DTD Schema

XML document schema types are dependent on data types of data. On the basis of data types XML document schema categorized in following two ways.

1. Simple type
2. Complex type

XML Schema is commonly known as XML Schema Definition (XSD). It is used to describe and validate the structure and the content of XML data.

### XML Schema Structure

XML schema defines the elements, attributes and data types. It is similar to a database schema that describes the data in a database.

#### Syntax :-

You need to declare a schema in your XML document as follows –

```
<XS : Schema XMLns : xs = http://www.abc.com/2001/XML Schema>
```

Example:

```
<?XML version = "1.0" encoding = "UTF-8"?>
<XS: Schema XMLS:XS = "http://www.abc.com/2001/XML Schema"
<XS: element name= "contact">
<xs : ComplexType>
 <xs: sequence>
 <xs: elements name="name" type="xs: string"/>
 <xs: elements name="Company" type="xs: string"/>
 <xs: elements name="phone" type="xs: int"/>
 </xs: sequence>
</xs :ComplexType>
```

</xs: element>

</xs: Schema>

### 5.6.1 Difference between XML Schema and DTD

XML Schema	DTD
1. XML Schema is designed using XML	1. DTD is developed using SGML Syntax
2. Supports datatypes for elements and attributes.	2. DTD doesn't support data types.
3. XML may be replaced using DOM.	3. Replacement is not possible using DOM.

### 5.7 XML Validation

Validation is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration (DTD), and if the document complies with the constraints expressed in it.

Validation is dealt in two ways by the XML parser. They are -

- 1 Well formed XML document
- 2 Valid XML document

#### 1. Well formed XML Document

An XML document is said to be well-formed if it adheres to the following rules

- ◆ XML documents must have a root element
- ◆ XML elements must have a closing tag
- ◆ XML tags are case sensitive
- ◆ XML elements must be properly nested
- ◆ XML attribute values must be quoted

#### Example

**Unformed XML Code:-**

```
<email>
 <to>Mr. John
 <body> Hello there !</to>
 </body>
</email>
```

### **Well Formed XML Code**

#### **Example:-**

```
<email>
 <to>Mr. Garcia </to>
 <body> Hello there </body>
</email>
```

## **2. Valid XML Document**

If an XML document is well-formed and has an associated Document Type Declaration (DTD), then it is said to be a valid XML document.

### **5.8 XML Namespace**

At the time of XML document creation, it may be possible that two elements have same name which increase chances of name conflicts.

Example

```
<?XML version = "1.0" encoding = "ISO-8859-15"?>
<html>
 <body>
 <P>Welcome</P>
 </body>
 <body>
```

```
<height>6th </height>
<Weight>155 </Weight>
</body>
</html>
```

In above example application <body> is used as per requirements which demonstrate human body but <body> is used two times in same document. As per rules of XML any element may not be used two times. To resolve this issue namespaces are developed.

Namespace is a mechanism by which element and attribute name can be assigned to a group. XML Namespaces provide a method to avoid element name conflicts.

### **Namespace declaration**

A Namespace is declared using reserved attributes.

Example

```
<elements XMLs : name = "URL">
```

- The Namespace starts with the keyword xmlns.
- The word name is the Namespace prefix.
- The URL is the Namespace identifier.

Example

```
<root>
 <h: table XMLs : h "http://www.abc.com/TR/>
 <h :tr>
 <h:td>Aries</h:td>
 <h:td>Bingo</h:td>
 </h:tr>
```

```

</h:table>
 <h: table XMLs : f“http://www.xyz.com/furniture/>
<h :name>
 <f:width>80 </width>
 <f:length>120</f:length>
</f:table>
</root>

```

In above example h and f are two different attributes of namespace which are added in single xml element (h: table).h and f are representing different URL whose definitions are different.

**Default Namespace :** It is such kind of namespace who does not use namespace prefix.

Default namespace may be declared using xmlns and without using prefix.

```
<xhtml xmlns=http://www.w3c.org/1999/xhtml>
```

### **Important points**

1. XML is developed by wide web consortium (W3c). It is subset of SGML (Standard generally markup language).
2. XML is used for storage of document data and information exchange.
3. XML have three types of tags – opening tags, ending tags and empty tags
4. XML documents is created by various storage units which are called Entities
5. XML have such symbols which may not be used as contents or XML text therefore these symbols are used to define XML Text and contents.
6. XML DTD is used in creation of valid structure of XML.
7. XML Schema definition is used for basic XML structure, describe XML Contents and to validate the XML documents.
8. XML validation is the process to understand XML document structure.

## Exercises

### Objective Type Questions

- XML Means?
  - X-Markup Language
  - Extensible markup language
  - Extra Markup language
  - Example markup language
- XML data is described by
  - Through XML Description node
  - By use of XSL
  - By Use of DTD
  - None of the above
- Which syntax is used for describe the version of XML
  - `<XML version="1.0"/>`
  - `<?XML version="1.0"?>`
  - `<?XML version="1.0"/>`
  - `<XML version="1.0"?/>`
- DTD Means
  - Dynamic Type Definition
  - Document Type definition
  - Do the Dance
  - Direct Type Definition
- XML data is used to store and \_\_\_\_\_.
  - Data Exchange
  - XML creation.
  - XML verification
  - None of the above.
- XML is similar to-
  - Java Script
  - Cprogramming
  - CSS
  - HTML
- XML does not work for-
  - Information sharing
  - To Store data
  - Exchanges of information.
  - Structure of Information
- XML valid syntax is \_\_\_\_\_
  - Mature
  - Well Parameterized
  - Well Formed
  - None of the above
- XML documents is used to validate-
  - CGG
  - DTD
  - Jquery
  - Parses

10. XML developed for-

a) By W3C

b) By W2C

c) By HTML

d) None of the above

**Short Type Questions.**

1. How to define syntax of XML version.

2. Describe briefly XML and HTML.

3. What is mean of valid XML document?

4. What is DTD?

5. What is use of XML attribute in XML?

**Essay Type Questions.**

1. Describe XML main work and advantages.

2. Describe DTD with example.

3. Describe XML Schema.

**Answer Key**

1. B    2. C                    3. B                    4. B                    5. A

6. D    7. C                    8. C                    9. B                    10 A



## Chapter- 6

### DATABASE MANAGEMENT SYSTEM (DBMS)

The database can normally be described as a repository for data. A database is a collection of any related data. A database is a collection of seamless, logically consistent, naturally meaningful and real data.

Database is a collection of related data. Data is a collection of facts and figures that can be processed for providing information. Most data represent recordable facts. Data is helpful in providing information that is based on facts.

If we have the marks obtained by all the students, then we can extract the list of toppers and other desired information. Data is stored by the data management system in such a way that the information can easily be obtained and updated.

#### 6.1 Data Base Management System (DBMS)

Data base management system is a software that has the functions of database storage, access, security, backup and such other features. Examples of some commonly used DBMS are: MySQL, Postgres SQL, SQL Server, Oracle etc.

DBMS is a collection of programs that enables the users to create and maintain the database. This software system gives us the following main features:

**Defining:** Facility for specifying type of data, structure of data and the constraints for the data being stored.

**Constructing:** Facility for storing the data in a storage device.

**Manipulating:** Facility for retrieving and updating the data present in the database and preparing reports.

#### 6.2 Database Abstraction

Database system provides available data to users according to their requirement and hides the details about how data is stored and managed in the hardware. This concept/process is known as database abstraction.

### 6.2.1 DBMS Architecture

Data base management systems is described by three-level schemas (schema architecture). There are three layers (levels), which are as follow:

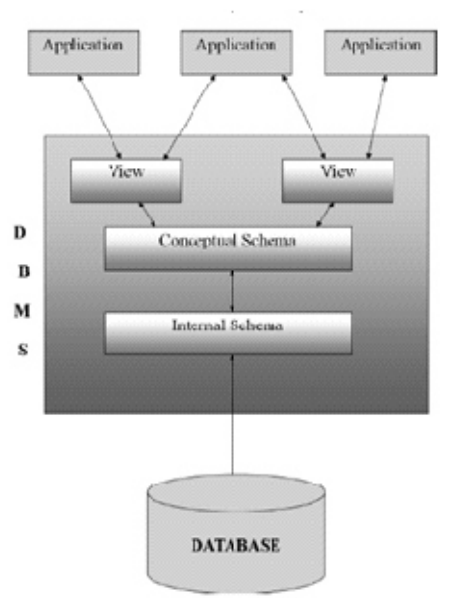


Fig. 6.1 Database Management System Architecture

#### 1. PHYSICAL LEVEL or INTERNAL LEVEL

It denotes how the data is stored in the database. It is the lowest level of abstraction.

#### 2. CONCEPTUAL or LOGICAL LEVEL

Conceptual abstraction is the next higher level of abstraction. This level represents what stored data and their relationship in a database. It is the job of database manager.

#### 3. EXTERNAL or VIEW LEVEL

Most of the database users do not use the complete database. It means that partial data is used by them. This external level is used for this purpose. It is the highest level of abstraction. What is needed by a user, or on what part of data the user is working, only that part of the database is visible to the user. In this way, there can be many views of one database.

Database can be categorized on different criterion. Database Management System is categorized on different basis, which are as follow:

1. On the basis of Data Models

1. Traditional Models: Relational, Network and Hierarchical Models

2. Emerging Models: Object-oriented and Object-relational Models

2. On the basis of users

Single and Multiple

3. On the basis of location

Centrlised and Distributed databases

4. On the basis of purpose

### **6.2.2 Database System Structure**

Databases system is divided into modules and each module has some or other responsibility of system control.

1.Storage Manager

2.Query Processor

1. Storage Manager

Storage manager is the module which makes interactions between low level data and application programs and queries. Storage manager is responsible for storing, retrieving and updating data in the database. It has following components.

1. Authorization and Integrity Manager:It tests the integrity constraints and checks the authorization of users to access data.
2. Transaction Manager: It ensures that no kind of change will be brought to the database until a transaction has been completed totally.
3. File Manager: It manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
4. Buffer Manager: It decides which data is in need to be cached in main memory and then fetch it up in main memory. This is very important as it defines the speed in which the database can be used.

The storage manager implements various data structures.

- (1) Data Files - where the data is stored.
- (2) Data Dictionary – where data about the data is stored.
- (3) Indices –which help in accessing the data faster.

## 2. Query Processor

As query is very much necessary to find out only the data user need from loads of data of the database, query processor is very important to process these query requests. Query processor has following components.

1. DDL Interpreter: It interprets the DDL statements and records the definitions in data dictionary.
2. DML Compiler: It translates the DML statements in low-level instructions that the query evaluation understands. It also performs query optimization which actually picks up the lowest cost evaluation plan from various alternatives.
3. Query Evaluation Engine: It executes the low-level instructions compiled by the DML compiler.

## 6.3 Advantages of DBMS

1. **No data redundancy and inconsistency:** In an organization, it is quite normal to have a set of data being used by different departments for different purposes. When each department keep its own data, initially it is a copy of the same data. After some time, changes may occur in the data which may not be updated everywhere. This leads to inconsistency. It increases the storage requirements and cost too. Data base management system saves us from this problem, because it reduces the issues of data redundancy.
2. **Restricting Unauthorized Access and Security:** All data present in the database should not be accessible to all users. Data Base Administrator restricts unauthorised access and provides security to the database.
3. **Data Integrity:** Security and integrity is maintained in th database. To maintain the integrity of data, some constraints or conditions are imposed.
4. **Simple Access :** Data is easily accessible in DBMS.

## **6.4 Characteristics of DBMS**

Following are the characteristics of DBMS:

1. Most important characteristic of DBMS is that data redundancy can be controlled.
2. Data can be shared.
3. Data is more secure.
4. Better and fast processing of data.
5. Data is independent in DBMS.
6. Examples of Data Base Management System Applications

## **6.5 Examples of Data Base Management System Applications**

Database is being in many organizations. Here are few examples:

1. In banking, database is used extensively. Any customer can do transactions at any branch of his/her bank. Earlier, when databases were not in use, customer had to go to his/her branch. Because of Database, customers are getting net-banking and such other facilities.
2. Airline reservation system is also based on database. Not only customer can book flight as per his/her convenience, but also the airlines employees can generate various reports which help in taking quick decisions.
3. In a university, information about teachers and students is stored in database. Details about various courses, how many students are studying in a course and other such details can be easily queried. Jobs related to examinations are also accomplished quickly.
4. Be it railway reservation or bus reservation, both are based on database. Passengers get the information quickly and at the same time, respective departments can get various reports as per their requirements.
5. Database makes it convenient to keep details of doctors, patients, various lab-test reports of patients, their medication, so that analysis-diagnosis of disease can take place easily.

## **6.6 Relational Data Base Management System**

Relations are created and maintained in various tables stored in database system, this system is called Relational Data Base Management System (RDBMS). DBMS

and RDBMS are used to physically store and maintain information in database. When the data increases in large amount, RDBMS is required for storing and managing in a better way. Relational data model contains information about indexes, keys, tables, and details about relationship among tables.

Edgar Frank Codd presented the principle of relational database in 1970s. Codd proposed a set of thirteen rules for relational model or principle. Relations among different data is the main requirement of relational model.

RDBMS can be considered as of next generation of DBMS. DBMS is used as a base model to store data in a relational database system. Although, instead of DBMS, RDBMS is used for complex business applications.

### 6.6.1 Entity Relationship Model (ER Model)

Entity relationship model is based on the concept of real-world entities and their relationships. While developing a database model of a real scenario, ER model creates entity set, relation set, normal attributes and constraints.

ER Model is used for conceptual design. It is based on

1. Entities and Attributes
2. Relationships among entities

**Entity:** In ER model, an entity has attributes which belong to real world. In each attribute, a set of its values is defined by domain.

For example, in a school database, student is an entity. Attributes of student can be name, age, course etc.

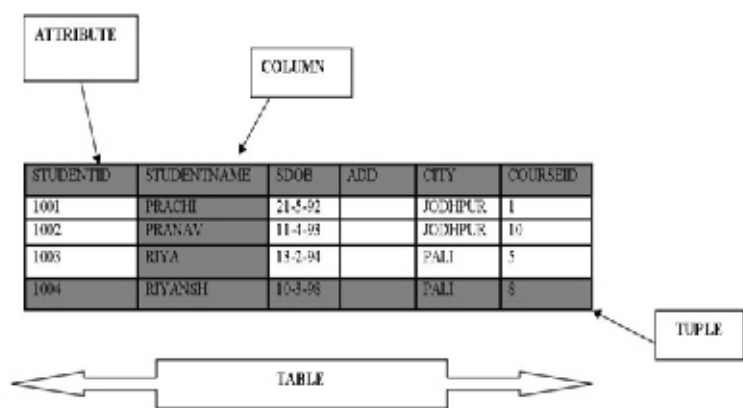


Fig 6.2 Entity Relationship Model

**Relationship:** Logical connection among more than one entities is called relationship. Relations in entity are mapped with different methods. Mapping cardinalities provides the number of relations between two entities.

Following cardinalities are there:

One to One

One to Many

Many to One

Many to Many

One element of entity A is connected to maximum one element of entity B and one element of entity B is connected to maximum one element of entity A.

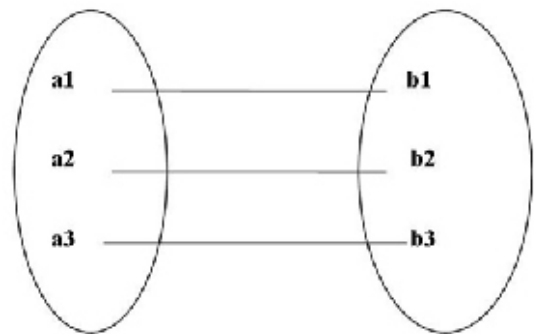


Fig. 6.3 One to One Relationship

One element of entity A may be connected to more than one element of entity B, but one element of entity B can be connected to maximum one element of entity A.

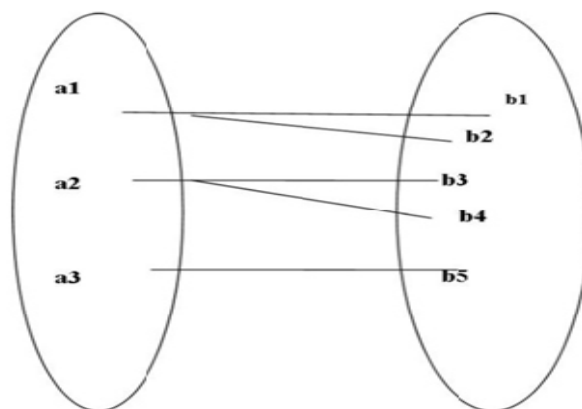


Fig. 6.4 One to Many Relationship

One element of entity A is connected to maximum one element of entity B, but one element of entity B may be connected to more than one element of entity A.

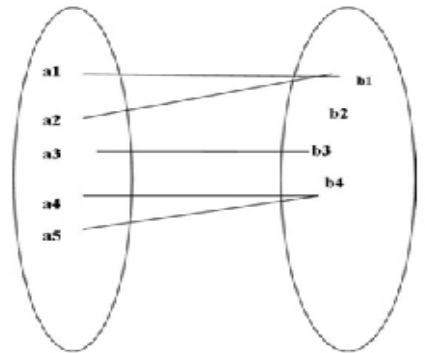


Fig. 6.5 Many to One Relationship

One element of entity A may be connected to more than one element of entity B and one element of entity B may be connected to more than one element of entity A.

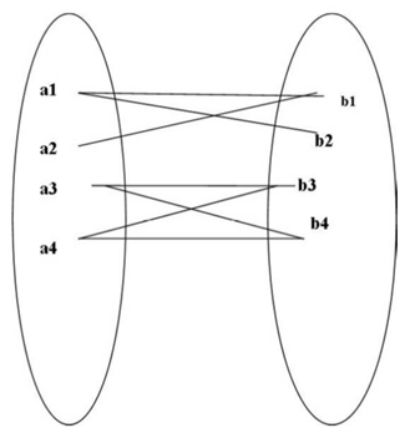


Fig. 6.6 Many to Many Relationship

### Some important definitions

- ◆ Data: Raw facts and figures which are useful for an organization are called data. We can not make decisions based on data.
- ◆ Information: Well-processed data is called information. We can decide on the basis of information.



- ◆ **Field/Attribute:** It represents a particular element of data. It is also called data item
- ◆ **Record:** Collection of fields is called a record. A record can have fields of different data types
- ◆ **File:** The collection of similar types of records is called a file.
- ◆ **Table:** Group of rows and columns is called a table, in which useful data / information is stored. It is generally considered a passive unit which is stored on secondary device.
- ◆ **Relation:** Relation (collection of rows and columns) on which we can perform different operations.
- ◆ **Database:** It is a collection of logically related data.
- ◆ **Tuple:** A row in a relation is called a tuple.
- ◆ **Domain:** The database domain is the group of all acceptable values. Example: A field for gender may have permissible domain to have three values {male, female, unknown} in that column.

### 6.6.2 Keys

Any attribute in the table which uniquely identifies each record in the table is called key.

#### 1. Primary Key

In a relational table, it is the first and foremost key which is used to uniquely identify a record. The primary key is of two types.

1. **Simple primary key:** The primary key which has one attribute is called simple primary key.
2. **Composite primary key:** The primary key which has more than one attribute is called composite primary key.

#### Defining primary key

1. The primary key is always unique.
2. There may be only one primary key in any table.

3. Composite primary key can have upto 16 fields combination for identifying a record uniquely.

## **2. Foreign Key**

In relational database, a foreign key is a group of column(s) that provide link between the two or more tables. It is also known as referencing key. In other words, when a primary key of one table is used as a key in another table, it is called a foreign key. Foreign key provides the method for maintaining integrity in data.

## **3. Composite Primary Key**

The composite primary key in the relational database table is a group of two or more columns that uniquely identifies each row in the table.

## **4. Super Key**

Super Key is defined as a set of attributes within a table that uniquely identifies each record within a table. Super Key is a superset of Candidate key.

## **5. Candidate Key**

Candidate keys are defined as the set of fields from which primary key can be selected. It is an attribute or set of attribute that can act as a primary key for a table to uniquely identify each record in that table.

## **6.7. SQL**

Structured Query Language(SQL) is the language of database. Relational databases like MS-Access, MS-SQL Server and Oracle use SQL.

### **6.7.1 Database Languages**

In any system, database languages are used to create and manage databases. Two types of languages are used in databases.

#### **Data Definition Language (DDL)**

DDL is used to define conceptual schema. It also provides information about how this schema is implemented in physical devices. The most important DDL statements in SQL are given below.

1. CREATE- To create objects in database

2. ALTER- To change in the structure of database
3. DROP- to remove objects from database
4. COMMENT- To put comment in data dictionary
5. RENAME- To rename the objects in database

### **Data Manipulation Language (DML)**

DML is used for manipulating the data in database. Few of the statements used in DML are given below.

1. SELECT- To retrieve data from database.
2. INSERT- To insert data in table.
3. UPDATE- To update the present data in a table.
4. DELETE- To delete records from a table.

### **6.8 MySQL**

MySQL is a database management system which is used for managing relational databases. It is an open source software.

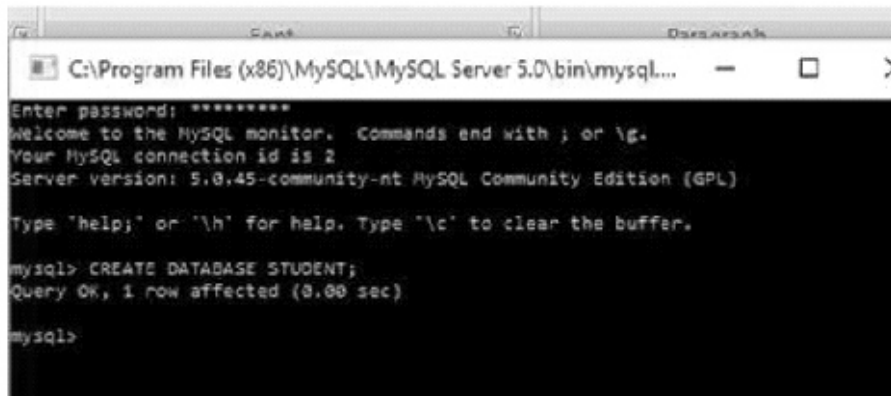


Fig. 6.7 MySQL Interface

## 6.8.1 MySQL Commands

### Create Database

There is a need to create a database before doing anything with the data. Database is a container of data. It stores details as per our requirements, for example about students, vendors, employees, customers or anything. Database in MySQL is a collection of tables, database views, triggers, stored procedures, etc., by which data is stored and manipulated. Following statement syntax is used to create a database in MySQL.

```
CREATE DATABASE [IF NOT EXISTS] [Name of Database]
```

'Name of Database' is the name by which we wish to create the database. Database name should be meaningful and descriptive as far as possible.

IF NOT EXISTS is an optional clause. It checks for the name of database, if it already exists in the database server. If it already exists it stops the user to create a new database with the same name. Database server cannot have two databases with the same name.

For example, to create a database with name 'student', the command is

```
CREATE DATABASE STUDENT;
```



```
C:\Program Files (x86)\MySQL\MySQL Server 5.7\bin>mysql.exe
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)
Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h;' for help. Type '\c;' to clear the current input statement.

mysql> show databases;
+-----+
| database |
+-----+
| information_schema |
| mysql |
| test |
+-----+
3 rows in set (0.00 sec)

mysql> CREATE DATABASE STUDENT;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW DATABASES;
+-----+
| database |
+-----+
| information_schema |
| mysql |
| student |
| test |
+-----+
4 rows in set (0.00 sec)

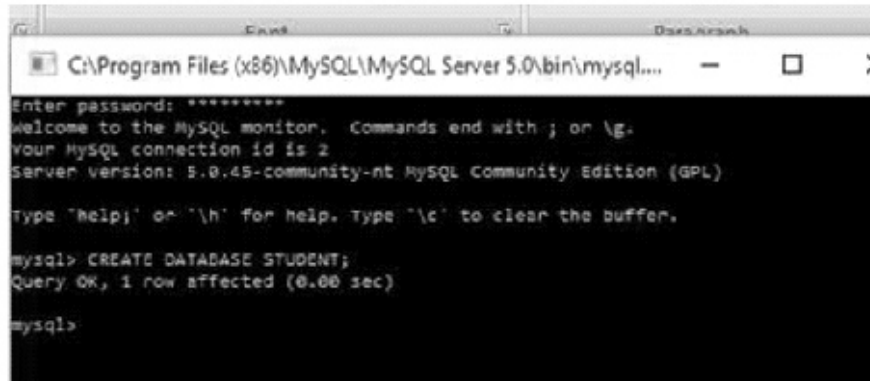
mysql> =
```

Fig. 6.8

After successful execution of the command, MySQL displays a message saying if the database is created successful or not.

## **SHOW DATABASES;**

This statement displays all database present in server. You can use this statement to check all databases or database you have created in the server. In the fresh installation of MySQL, when this statement is executed we get



```
C:\Program Files (x86)\MySQL\MySQL Server 5.0\bin\mysql...
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE DATABASE STUDENT;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Fig.2.9 Making Database

## **USE DATABASE**

Before working with a particular database, you have to tell which database will be used. You can work with the database that you want to use by using this statement, you can select the database named as 'student' using the following statement as follows.

## **USE STUDENT;**

Now creating new table, placing data query or calling a stored procedure etc. shall be applicable to database STUDENT.

## **DATABASE DROP**

Dropping a database means removing the database permanently. All data and related objects inside the database are permanently removed and this removal can not be reverted. Therefore, when this statement is given, a warning message appears on the screen. One should be extra careful while executing this statement. The syntax is

```
DROP DATABASE [IF EXISTS] database_name;
```

If we wish to remove STUDENT database, we write following statement.

e.g. DROP DATABASE STUDENT;

## MySQL Data Types

The columns of a table may contain different types of values, such as numeric or string. MySQL provides more different data types than numeric or string. Each data type can be determined by the following specifications in MySQL

- ◆ The value represents what kind of values?
- ◆ Is the value of fixed length or variable length?
- ◆ Can the datatype of value be indexed?

### Numeric Data

MySQL provides following numeric data types like SQL. Also, it provides facility to store single bit value with a data type called BIT. Numeric data types can be signed and unsigned. The table shows numeric data types available in MySQL.

Numeric Types	Description
TINYINT	A very small integer
SMALLINT	A small integer
MEDIUMINT	A medium-sized integer
INT	A standard integer
BIGINT	A large integer
DECIMAL	A fixed-point number
FLOAT	A single-precision floating point number
DOUBLE	A double-precision floating point number
BIT	A bit field

### String Data

In MySQL, the string data type can be from simple text to binary data such as images and files. The string data type can be used for matching and searching values. The following table shows the string data type in MySQL

<b>String Types</b>	<b>Description</b>
CHAR	A fixed-length nonbinary (character) string
VARCHAR	A variable-length non-binary string
BINARY	A fixed-length binary string
VARBINARY	A variable-length binary string
TINYTEXT	A very small non-binary string
TEXT	A small non-binary string
MEDIUMTEXT	A medium-sized non-binary string
LONGTEXT	A large non-binary string
ENUM	An enumeration; each column value may be assigned one enumeration member
SET	A set; each column value may be assigned zero or more set members

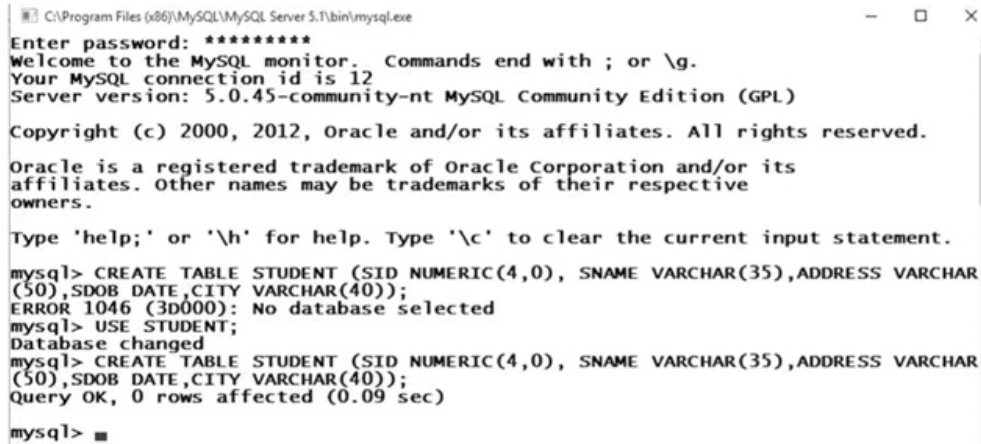
### **Date and Time Data Types**

MySQL provides date and time data types as well as a combination of date and time data type. In addition, MySQL timestamp data type also provides the facility to track the changes made in a row in a table. To store only year without date and month, year data type can be used. The following table shows date and time data type in MySQL

<b>Date and Time Types</b>	<b>Description</b>
DATE	A date value in ;YYYY-MM-DD' format
TIME	A time value in ;hh:mm:ss' format
DATETIME	A date and time value in ;YYYY-MM-DD hh:mm:ss' format
TIMESTAMP	A timestamp value in ;YYYY-MM-DD hh:mm:ss' format
YEAR	A year value in YYYY or YY format

## CREATE NEW TABLE IN A DATABASE

To make a new table within the database, the statement is 'create table' in MySQL. The statement of making tables is a complex statement, which is as follows.



```
C:\Program Files (x86)\MySQL\MySQL Server 5.1\bin\mysql.exe
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)
Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> CREATE TABLE STUDENT (SID NUMERIC(4,0), SNAME VARCHAR(35),ADDRESS VARCHAR
(50),SDOB DATE,CITY VARCHAR(40));
ERROR 1046 (3D000): No database selected
mysql> USE STUDENT;
Database changed
mysql> CREATE TABLE STUDENT (SID NUMERIC(4,0), SNAME VARCHAR(35),ADDRESS VARCHAR
(50),SDOB DATE,CITY VARCHAR(40));
Query OK, 0 rows affected (0.09 sec)
mysql> ■
```

Figure 6.10

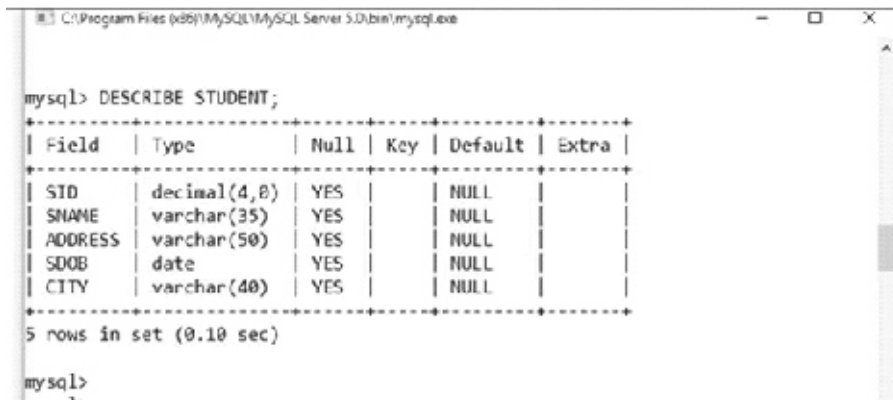
## DESCRIBE TABLE

To see the attributes and their data types of a table, DESCRIBE command is used.

DESCRIBE TABLE\_NAME

For example

DESCRIBE STUDENT;



```
C:\Program Files (x86)\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql> DESCRIBE STUDENT;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| SID | decimal(4,0) | YES | | NULL | |
| SNAME | varchar(35) | YES | | NULL | |
| ADDRESS | varchar(50) | YES | | NULL | |
| SDOB | date | YES | | NULL | |
| CITY | varchar(40) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.10 sec)
mysql>
```

Figure 6.11 Describe Database



## **DROP TABLE**

To remove an existing table, the following statement is used.

```
DROP [TEMPORARY] TABLE [IF EXISTS] table_name [, table_name]
```

For example, to remove STUDENT table, we write

```
DROPTABLE STUDENT;
```

DROP TABLE statement permanently deletes a table and its data from the database. This statement can be used for multiple tables also, in that case each table is separated by a comma.

## **ALTER TABLE**

To change the structure of an existing table, the alter table statement is used. It allows to add a column, delete a column, change data type of a column, add a primary key, rename the table and do many other tasks. The syntax is

```
ALTER TABLE table_name action1[,action2,...]
```

To change the structure of an existing table, specify

- the name of table in which we wish to make changes.
- actions you want to apply to the table, any action, such as adding a new column, adding primary key, changing the name of the table, etc.

More than one action can be applied with a single 'alter table' statement. Each action is separated by a comma.

## **ADD COLUMN TO TABLE**

After creating a table, due to some new requirements, there may be a requirement to add a new column, for example, to add a date\_of\_birth column to the STUDENT table, we write

```
ALTER TABLE STUDENT ADD COLUMN DOB date;
```

DOB column will be added after the last column in the table.

If the column to be added after a particular column, the name of that particular column will be written in the statement. For example, to add PINCODE column, after

CITY, the statement will be

```
ALTER TABLE STUDENT ADD COLUMN PINCODE AFTER CITY;
```

### **DROP COLUMN**

If we want to remove a column from the table, we can drop it, for example, suppose the table has a PINCODE column and now it is no more needed in the STUDENT table. The following statement will be written for the purpose.

```
ALTER TABLE STUDENT DROP COLUMN PINCODE;
```

### **RENAMING TABLE**

To change the name of the table, ALTER can be used, the statement will be

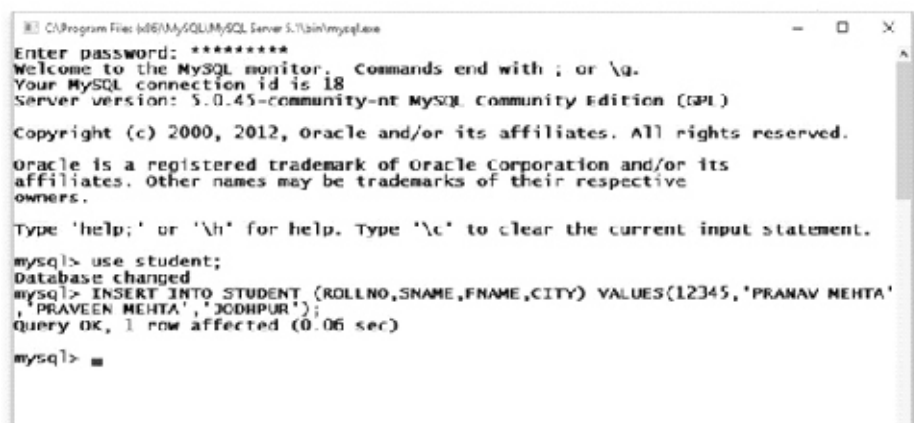
```
ALTER TABLE STUDENT RENAME TO STUINFO;
```

### **INSERT Statement**

Insert statement is used to insert one or more rows into a table. The syntax of the insert statement is

```
INSERT INTO TABLE [COLUMN1, COLUMN2,.....] VALUES
(VALUE1,VALUE2.....)
```

The list of columns is optional. After INSERT INTO, write list of columns of the specified table separated by commas within parentheses. Insert the values of columns separated by commas in the parentheses after the VALUES keyword.



```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 18
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use student;
Database changed
mysql> INSERT INTO STUDENT (ROLLNO,SNAME,FNAME,CITY) VALUES(12345,'PRANAV MEHTA',
'PRAVEEN MEHTA','JODHPUR');
Query OK, 1 row affected (0.06 sec)

mysql> █
```

Figure 6.12 Inserting data in a table

For example:

If we wish to put in data of a student in the STUDENT table then the following INSERT statement will be

```
INSERT INTO STUDENT (ROLLNO,SNAME,FNAME,CITY)
VALUES (12345,'PRANAV MEHTA','PRAVEEN
MEHTA','JODHPUR');
```

List of columns is optional, hence, following is also valid.

```
INSERT INTO STUDENT VALUES (12345,'PRANAV
MEHTA','PRAVEEN MEHTA','JODHPUR');
```

### **INSERT MULTIPLE ROWS**

To add more than one rows in a table, at the same time, following syntax may be used.

```
INSERT INTO table (column1,column2...) VALUES (value1,value2,...),
(value1,value2,...),
```

For example, if we wish to add data of Pranav, Prachi and Rudra Pratap, we can write as

```
INSERT INTO STUDENT VALUES (1 2 3 4 5 , ' P R A N A V
MEHTA','PRAVEENMEHTA','JODHPUR'),(12346,'PRACHI MEHTA','PRAVEEN
MEHTA','JODHPUR'), (12347,'RUDRA PRATAP','RAKESH
MEHTA','JODHPUR');
```

Here, values of each rows are placed in brackets which are separated by comma.

In the INSERT statement, the value can also be inserted by the SELECT statement. It is possible to copy (partially, or full) the contents of a table to another.

```
INSERT INTO table_1 SELECT c1, c2, FROM table_2;
```

For example, to copy complete data of STUDENT, to another table TEMPSTU, we can write the following statement

```
INSERT INTO TEMPSTU SELECT * FROM STUDENT;
```

## UPDATE

One of the important job while working with a database is data updating. To update some existing data in a table, we use UPDATE statement. We can use this statement to update one row, group of rows or all rows in the table. The UPDATE statement is as follows.

```
UPDATE table_name
```

```
SET
```

```
column_name1 = expr1,
```

```
column_name2 = expr2,
```

```
...
```

```
WHERE
```

```
condition;
```

Firstly, after the update keyword, specify the table name you want to update. Second, specify the column which you want to modify and give new value with the SET statement. To update multiple columns, separate them by commas.

Third, specify which rows will be updated using a condition in the where clause. This clause is optional. If we leave the clause, all rows in the table will be updated.

Where clause is extremely important. It should be used with caution, missing this clause, will update all the row in the database.

### Single Column Example:

In this example, we want to update Pranav's email with the new e-mail pranav@boser.edu.in.

```
UPDATE STUDENT SET EMAIL='pranav@boser-edu-in' WHERE
SNAME='PRANAV';
```

### Multiple Column Example:

In this example, we want to update email and lastname of rollno 205502.

```
UPDATE SET LASTNAME='MOHNOT', EMAIL=' pranav@boser-edu-
in' WHERE
```

```
ROLLNO=205502;
```

### **DELETE Statement**

To remove data from one or more tables, the DELETE statement is used. A single DELETE statement can be used to remove records from multiple tables.

```
DELETE FROM table
```

```
[WHERE conditions]
```

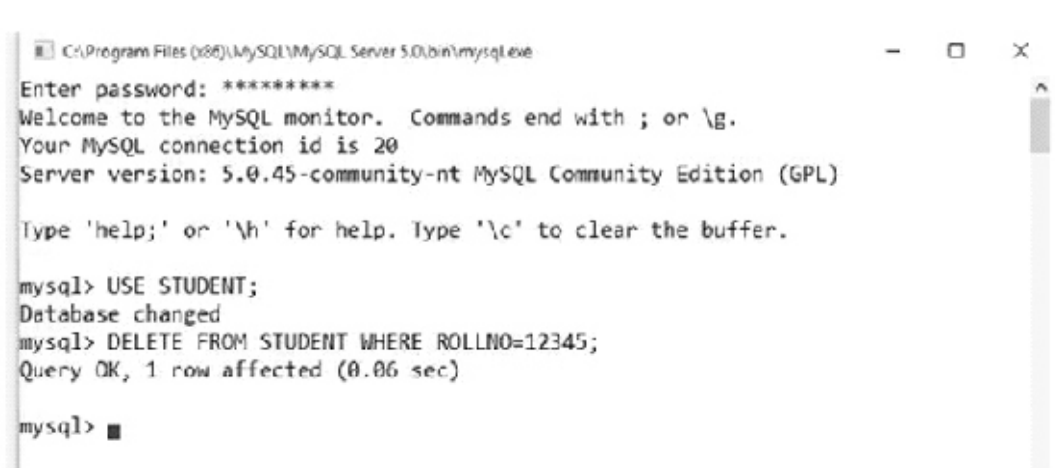
```
[ORDER BY ...]
```

```
[LIMIT rows]
```

WHERE Clause describes which row(s) you want to remove. If the record fulfills the condition given in WHERE clause, the row is deleted permanently from the table. If we do not use the WHERE clause, all records of that table will be removed. ROW\_COUNT() function shall return the number of records deleted by the preceding statement.

For example, to delete the record of student who has rollno 123456, we write

```
DELETE FROM STUDENT WHERE ROLLNO=123456
```



```
C:\Program Files (x86)\MySQL\MySQL Server 5.0\bin\mysql.exe
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> USE STUDENT;
Database changed
mysql> DELETE FROM STUDENT WHERE ROLLNO=12345;
Query OK, 1 row affected (0.06 sec)

mysql> █
```

Figure 6.14 Removing record from database

To delete all records from the table, the statement is

```
DELETE FROM STUDENT;
```

All records from the stable STUDENT are deleted by this statement.

### **Removing Records from more than one table**

```
DELETE table_1, table_2,...
```

```
FROM table-refs
```

```
[WHERE conditions]
```

```
DELETE FROM table_1, table_2,...
```

```
USING table-refs
```

```
[WHERE conditions]
```

For example, in a situation where one course is closed, to remove the record of that course and all records of students studying that course, the DELETE statement shall be

```
DELETE STUDENT, COURSE
```

```
FROM STUDENT, COURSE
```

```
WHERE STUDENT.COURSEID = COURSE.COURSEID AND
COURSE.COURSEID=1;
```

### **SELECT statement**

The SELECT statement is used to obtain data from the table. A table is a combination of rows and columns which is like a spreadsheet. Most of all, we want to see a group of rows or group of columns or a combination of both of them. The yield of the SELECT statement is called RESULT SET.

The result set is a list of group rows that have the same number of columns.

For example, see table STUDENT, which has columns – rollno, stuname, dob, email, coursecode

ROLLNO	STUNAME	DOB	EMAIL	COURSE CODE
100201	Pranav	2000-2-15	pranav@boser.edu.in	1
100202	Prachi	2000-3-10	prachi@boser.edu.in	2
100203	Riya	2001-5-21	riya@boser.edu.in	1
100204	Rudra Pratap	2000-4-25	rudra@boser.edu.in	1
100205	Riyansh	2002-3-11	riyansh@boser.edi.in	2

Which columns and rows we want to see we use SELECT statement. For example, if wish to see only name of student and email or name of student and date of birth, SELECT statement helps in doing so. The syntax of SELECT statement is

```

SELECT
 column_1, column_2, column_3 ...
FROM
 table_1
 [INNER | LEFT | RIGHT] JOIN table_2 ON conditions
WHERE
 Conditions
GROUP BY column_1
HAVING group_conditions
ORDER BY column_1

```

SELECT statement has different clauses, which are mentioned below.

- ◆ The SELECT statement allows for partial query in a table of data, for that a list of columns separated by commas is specified in the SELECT clause or a \* (asterisk) which means that you want to see all the columns.
- ◆ JOIN is used to receive data from other tables based on the conditions of relations.

- ◆ WHERE is a filter of ROWS in WHERE RESULT SET.
- ◆ GROUPBY creates a group of rows and the AGGREGATE function is held on each group.
- ◆ HAVING clause filtersthe groups made by GROUPBY.
- ◆ ORDER BY specifies a list of columns to sort.
- ◆ SELECT and FROM clauses are required in the statement, the other part is optional.
- ◆ SELECT statement permits to query partially about data in a table; for this purpose a list of columns separated by commas is specified.

#### Example

To see student name and email, we write the query as

```
SELECT
 STUNAME,EMAIL
FROM STUDENT;
```

To see all columns, we can write name of each column or simply \*.

```
SELECT ROLLNO,STUNAME, FNAME,EMAIL,CITY
FROM STUDENT;
```

OR

```
SELECT *
FROM STUDENT;
```

These two queries will result the same.

#### **GROUPBY**

This part is optional in SELECT statement. It groups and summarises a number of rows. For every group there is one row as the output. In other words, it reduces the number of rows in the result set.



We often use aggregate functions such as SUM, AVG, MAX, MIN and COUNT with GROUP BY. The aggregate functions provides information about individual groups, selected by SELECT statement. The syntax is

```
SELECT
 c1, c2, ..., cn, aggregate_function(ci)
FROM
 table
WHERE
 where_conditions
GROUP BY c1 , c2, ..., cn;
```

The GROUP BY segment should be written after the FROM and WHERE clauses. After GROUP BY, columns separated by commas are written or an expression that you want to use as criteria is written.

Aggregate function gives a single value after processing a group of rows. The GROUP BY is generally used with an aggregate function which provides a single value for each subgroup.

For example,

If we want to know how many courses are there, we can write GROUP BY with COUNT as

```
SELECT COURSEID, COUNT(*)
FROM COURSE
GROUP BY COURSEID
```

If we want a group according to the courses then the following statement will be written

```
SELECT *
FROM STUDENT
GROUP BY COURSEID
```

If we want to know how many students in the course, we use the GROUP BY segment with the COURSE column as the following query.

```
SELECT COURSEID,COUNT(*)

FROM STUDENT

GROUP BY COURSEID
```

We use the HAVING block to filter the group given by GROUP BY. The following query will be written using the HAVING section to find the number of students born after year 2003.

```
SELECT DOB, YEAR(DOB) AS YEAR, COUNT(*)

FROM STUDENT

GROUP BY YEAR

HAVING YEAR(DOB) > 2003;
```

The HAVING segment is used to filter the set or group of rows in the SELECT statement. The filter works on the column(s) in the GROUP BY block, if the GROUP BY block is omitted, then the HAVING segment acts like FROM.

The filter applies to each group of rows in the HAVING section, whereas the filter condition in WHERE section applies to every different row.

When you use the SELECT statement to query the data from a table, the result set does not occur in any sequence. To order result set, we use the ORDER BY. By using ORDER BY segment

- Result set can be sorted by a single column or multiple columns
- Result set, different columns, can be sorted in either ascending or descending order.

ORDER BY block is used as

```
SELECT column1, column2,

FROM table_name

ORDER BY column1 [ASC|DESC], column2 [ASC|DESC], ...
```

For example, we wish list the names of the students in ascending alphabetical order, the query will be as follows:

```
SELECT SNAME, ROLLNO
FROM STUDENT
ORDER BY SNAME;
```

If we wish list the names of the students in descending order, the query will be as follows:

```
SELECT SNAME, ROLLNO
FROM STUDENT
ORDER BY SNAME DESC;
```

If we need merit list of the candidates on the basis of total marks obtained, the query will be as follows.

```
SELECT ROLLNO, SNAME, TOTALMKS
FROM STUDENT
ORDER BY TOTALMKS;
```

### **6.8.2 MySQL FUNCTIONS**

Commonly used functions are described below in brief.

#### **AVG: Calculates the average of a set**

For example, to find the average score of the first subject of the students

```
SELECT AVG(SUB1) FROM STUDENT;
```

To find the average score of the first subject of the students who secured first division.

```
SELECT AVG(SUB1) FROM STUDENT WHERE DIV='I';
```

Using the DISTINCT operator, you can use the average function to calculate the average of different scores.

```
SELECT AVG(DISTINCT SUB1) FROM STUDENT;
```

We often use the average function to compute the average value for each group of rows in a table in conjunction with the GROUP BY segment. According to the syllabus, the average score of the first subject of the student is to be calculated as

```
SELECT AVG(SUB1) FROM STUDENT GROUP BY COURSE;
```

In order to set conditions for the average values for groups, the function AVG can be used in the HAVING section. To find average score of the marks of the first subject of the first class students for all courses, the statement will be

```
SELECT AVG(SUB1) FROM STUDENT GROUP BY COURSE
HAVING AVG(SUB1)>59;
```

**COUNT: Tells the number of rows in a table.**

COUNT Function returns the number of rows in a table. It counts the rows which are satisfying given condition.

COUNT function works in many different ways. It provides value in BIGINT data type. It returns zero when no matching row is detected.

COUNT(\*) returns the number of rows in a table. It includes the rows which have NULL values.

For example, to find total number of rows in table Students, we write

```
SELECT COUNT(*)

FROM STUDENTS;
```

To calculate, we can apply some condition using where, like

```
SELECT COUNT(*)

FROM STUDENTS

WHERE DIV='FIRST';
```

To calculate unique rows in a table, we use DISTINCT with COUNT, like

```
COUNT (DISTINCT expression)

SELECT COUNT(DISTINCT COURSEID)
```

FROM STUDENT;

Many a times, COUNT function is used with GROUP BY clause to identify data in groups.

**SUM: Calculates the sum of a set.**

SUM function is used for adding a set of values or expressions. If there is no value matching in the SELECT statement, the result of SUM function will be zero. We can use DISTINCT with SUM to find out the sum of only those values which are distinct. SUM function ignores NULL values for calculation.

SUM(DISTINCT expression)

For example, to find the sum of fee deposited by students, the value will be stored in TOTALFEE.

SELECT SUM(FEE) AS TOTALFEE

FROM STUDENT;

If we wish to sum the fee, course wise

SELECT SUM(FEE) AS TOTALFEE

FROM STUDENT

GROUP BY COURSEID;

SUM can be used with HAVING function to filter results based on a specific condition. The SUM function ignores NULL values in the calculation.

**MIN: Finds the minimum from the set.**

MIN function gives the minimum value in the set. It is very useful function in many situations like, finding the minimum marks obtained, or cheapest product etc. The syntax is

MIN(expression);

If we wish to find out the minimum marks obtained in first subject, we write

SELECT SNAME, MIN(SUB1)

FROM STUDENT;

### **MAX: Finds the maximum from the set.**

MAX function gives the maximum value in the set. It is very useful function in many situations like, finding the highest marks obtained, or costliest product etc. The syntax is

```
MAX(expression);
```

If we wish to find out the maximum marks obtained in first subject, we write

```
SELECT SNAME, MAX(SUB1)
```

```
FROM STUDENT;
```

### **STRING Functions**

CONCAT and CONCAT\_WS are used to add two or more strings at once using the function.

#### **CONCAT(expr1, expr2...)**

The CONCAT function takes one or more strings and combines them into a single string. The CONCAT function, there is minimum requirement of one string otherwise it will give an error. The syntax is

```
CONCAT(sting1,string2,string3,.....);
```

```
SELECT CONCAT('Suncity','Jodhpur');
```

The result will be SuncityJodhpur

```
SELECT CONCAT('Suncity',' ','Jodhpur');
```

The result will be Suncity Jodhpur

```
SELECT CONCAT('Suncity','.', 'Jodhpur');
```

The result will be Suncity.Jodhpur

CONCAT\_WS combines two or more string values with a predefined separator.

```
CONCAT_WS(separator,string1,string2,string3,...)
```

```
SELECT CONCAT_WS('-', 'Pranav', 'Mehta');
```

Output will be :Pranav.Mehta

```
SELECT CONCAT_WS('-', 'Pranav', 'Mehta');
```

Output will be : Pranav-Mehta

**LENGTH function:** It is used for finding the length of string, i.e. how many letters, characters, spaces in the string. It is measured in bytes.

**CHAR LENGHT:** This function also works to get the number of letters, characters, spaces in the string. It is measured in characters.

```
SELECT LENGTH('prachi');
```

Output will be 5

```
SELECT LENGTH('riyamehta');
```

Output will be 10

### **LEFT() function**

It is used to return a string of specified length from left a string. LEFT function accepts two arguments. The syntax is

```
LEFT(str, length)
```

1. The str string is the one from which we have to get the substring.
2. The length specifies the number of characters which will be a positive integer. The str string from the function gives the left-most number of letters written in LENGTH. If the value of str or LENGTH is NULL, then it returns a NULL value.

If the LENGTH is zero or negative, the LEFT function returns an empty string. If LENGTH is greater than the length of the string, then the LEFT function returns the entire string.

### **RIGHT() function**

It is used to return a string of specified length from right of a string. RIGHT Function accepts two arguments. The syntax is

```
RIGHT(str, length)
```

1. The str string is the one from which we have to get the substring.

2. The length specifies the number of characters which will be a positive integer. The str string from the function gives the right-most number of letters written in LENGTH. If the value of str or LENGTH is NULL, then it returns a NULL value.

If the LENGTH is zero or negative, the RIGHT function returns an empty string. If LENGTH is greater than the length of the string, then the RIGHT function returns the entire string.

## **TRIM**

TRIM function is used to remove unwanted spaces from the beginning or end of a string.

```
SELECT TRIM ('Sample Text ');
```

Result: Sample Text

## **DATE**

The date function is marked in YYYY-MM-DD format. The values ??of DATE can also be given in a string or number. This function accepts the value between 1000-01-01 to 9999-12-31.

## **MONTH**

The MONTH function returns the value of the month from the value of DATE, it is between 1 and 12.

```
SELECT MONTH('2016-02-03')
```

Result: 2

## **YEAR**

The YEAR function provides value for the year from the value of DATE.

```
SELECT YEAR('2016-02-03')
```

Result: 2016

## **DAY**

The DAY function returns the value of the day of the month from the value of



DATE, it is between 1 and 31. This is synonymous with DAYOFMONTH ().

```
SELECT DAY ('2016-02-15')
```

Result: 15

### **SYSDATE**

SYSDATE () provides current date and time value of the functioning system according to the string or number data as YYYY -MM-DD HH: MM: SS or YYYYMMDDHHMMSS.

```
SELECT SYSDATE ();
```

Result: 2016 -01 -16 13 Rs 47 per Rs 36

### **DATEDIFF**

The DATEDIFF function calculates the number of days between the two dates, the date time, or the timestamp value.

### **Important Points**

1. The database can normally be described as a repository for data.
2. Data is a collection of facts and figures that can be processed for providing information.
3. Data base management system is a software that has the functions of database storage, access, security, backup and such other features.
4. Data base management systems is described by three-level schemas (schema architecture).
5. Databases system is divided into two modules - Storage Manager and Query Processor.
6. Most important characteristic of DBMS is that data redundancy can be controlled.
7. Well-processed data is called information.
8. In any system, database languages are used to create and maintain a database.

9. Two types of languages are used in database – Data Definition Language and Data Manipulation Language.
10. Structured Query Language (SQL) is database language.
11. MySQL is a database management system used for managing relational database. It is open source software.

### **Exercises**

#### **Objective Type Question**

1. Raw facts and figures are  
A. Data      B. Information      C. Snapshot      D. Report
2. The facility which permits accessing only a few records in database.  
A. Form      B. Report      C. Queries      D. Tables
3. What are relational databases?  
A. To store relational information at one place.  
B. Relation of one database with another  
C. A database to store human relations  
D. Any of the above
4. With this key, each record is identified uniquely  
A. Primary Key      B. Key Record  
C. Unique Key      D. Name of Field
5. Database language related to definition of schema and complete database structure is  
A. DCL      B. DML      C. DDL      D. All of the Above

#### **Short Type Questions**

1. What do you understand with database?
2. Explain database architecture.
3. How many data types are available in MySQL?

4. How records are inserted in MySQL? Explain.
5. What is the utility of UPDATE function?

### **Essay Type Questions**

1. Explain database management system and its advantages.
2. Write the characteristics of database management system.
3. How many types of database languages are there? Explain in detail.
4. How a database of students can be created in MySQL? Explain in detail.
5. Explain string function in MySQL.

### **Answers Key**

1.A 2. C 3. A 4. A 5. B

## Chapter 7

### PHP

#### 7.1 Basics of Scripting

A scripting or script language is a programming language that support programs written for special run time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator. Most of the times scripting languages are often interpreted rather than compilation. The term scripting language is also used to refer to dynamic high level general purpose language such as Perl, TCL, Python, Php etc. Scripts often used for small programs (up to few thousand lines of code) in such languages, or in domain-specific languages such as the text processing language like: sed and AWK.

Scripting standards are types of:

1. BASH Shell Scripting using in Linux / Unix Operating system to perform some certain task.
2. For web-development we use: Php, Jsp, ASP etc.

##### 7.1.1 PHP History and Introduction :

PHP as it is known today is actually the successor to a product named PHP/FI (Hypertext Interpreter / Form interpreter). Created in 1994 by Rasmus Lerdorf, the very first incarnation of PHP was a simple set of Common Gateway Interface (CGI) binaries written in the C programming language. Originally used for tracking visits to his online resume, he named the suite of scripts “Personal Home Page Tools,” more frequently referenced as “PHP Tools.” Over time, more functionality was desired, and Rasmus rewrote PHP tools and producing a much larger and richer implementation. This new model was capable of database interaction and more, providing a framework upon which users could develop simple dynamic web applications such as guest books. In June of 1995, Rasmus released the source code for PHP tools to the public, which allowed developers to use it as they saw fit.

PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. Instead of lots of commands to output HTML (as seen in C or Perl), PHP pages contain HTML with embedded code that does “something” (in this case, output “Hello Students am in PHP script !”). The PHP code is enclosed in special start and end processing instructions `<?php // Your Code here ?>` that allow you to jump into and out of PHP mode.

```
<?php
 echo “Hello Students am in Php Script !”;
?>
```

### **Example-1 an introductory example**

#### **7.1.2 Use of interpreter in PHP**

Basically PHP language is interpreted and it is scripting type of language. The binary that lets you interpret PHP is compiled, but what you write is interpreted. The Apache web server used to interpret the code. Basically PHP code stored at server side (Under the apache web root directory) and interpreted by apache server into HTML and send back to client (here client term refers to web-browser like: Internet Explorer / Google Chrome / Firefox etc).

#### **7.1.3 There are two main types of website**

a) Static Website

b) Dynamic Website

**a) Static Website:** A static website is the simplest type of website you can design & developed easily. Mostly we code once a time and change very less times with such type of websites. Static websites are mostly written in HTML, CSS (Cascading Style Sheet) and Java- Script (JS) code only. The only form of interactivity on a static website is hyperlinks (To jump / access on another webpage which given in hyperlinks).

If you want to create your own website to be a small one (3 pages or less), then a static website approach might be the easiest way to go. Static websites are easier to make than dynamic websites, because they require less coding and technical knowledge. To write static website code, we can use any editor to write them, like as Notepad, Dream viewer IDE etc. In this chapter, we will use Dream viewer editor for coding purpose. Static website also known as “client side scripting” because the web page which written in HTML, CSS and JS and they can be run / execute by just double click,

so it open / run directly in the browser. Finally we can say that web browser only understand client side scripting like: HTML , CSS and JS.

**b) Dynamic website:** Mostly such types of websites are also known as web-applications, because the website you are which have with the database and written in server side script (like PHP), so only interpreter can execute / run such code and on the web browser only HTML page shown in the form of output. For example we code in HTML form tags, which gets information from the user and this information can be sent into table of database. (In the earlier chapters we have learnt about database and its tables). The table of database, store information of user in the form of rows and columns. Such kind of information can be use for the future aspects.

In this chapter we will learn how php connect with the mysql database, and how to code in HTML form tags (<form> </form>) and such form receive information. (The entered information in HTML form, may belongs to school / college / company, from user and this information can store in MySQL database).

We can divide user interface of website into three parts:

- a) Type of webpage
- b) style of webpage
- c) Behaviour of webpage

To create any website layout and its structure most of the times we use HTML and how the website look likes and how will be the interface look of website mostly we use: CSS. To validate the data entered by the user in HTML form, we use JS (Java Script).

#### **7.1.4 Use of PHP in website development:**

There is assumption that PHP is mainly focused on server-side scripting (dynamic web pages), It's not completely true. We can use PHP scripting in GTK(Genome Tool Kit) development to create desktop application for any kind of operating system. GTK based desktop applications can be execute independently without any dependency. We can run / execute php on command line interface (CLI) in Microsoft based operating system as well as open source OS. PHP works under various kind of web servers like: wamp, xampp, apache etc web-servers are available freely over the internet. In this chapter we will use XAMPP bundled software as server which consist of: apache webserver, PHP and mysql. The following figure shows you how to turn on / off the services of apache and mysql server.

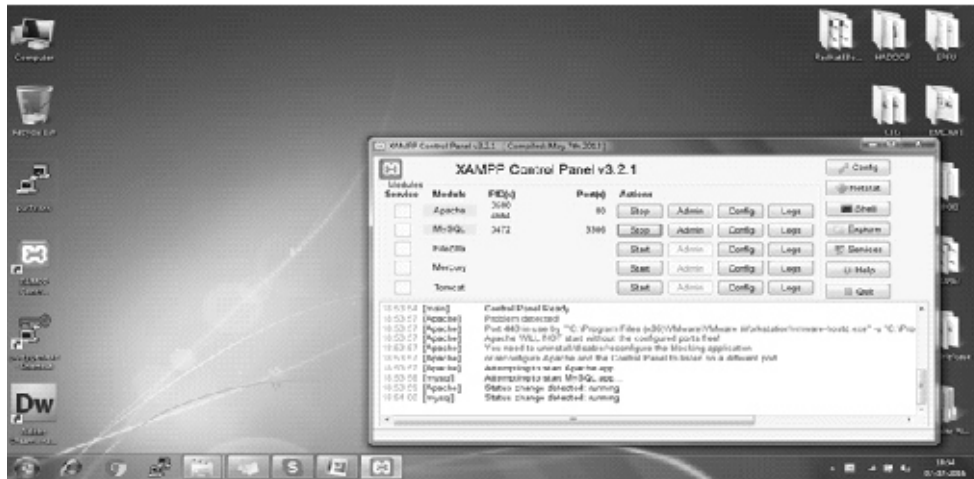


Figure 7.1 Control panel of XAMPP Software

After running the services of apache and mysql, we can test the web-server (apache) and data-base i.e. mysql by typing URL address in browser **http://localhost** , you will see there a welcome screen of XAMPP software. The apache web server works on TCP (Transmission control protocol) with port number is 80 and MySQL server also works under TCP service and having port number 3306. When we test the web services of XAMPP in browser, **http://** is representing as a protocol.



Figure 7.2 Welcome screen of xampp software in web browser

## 7.2 Web root directory in apache web-server:

The web root directory named as 'htdocs' located under the file system of XAMPP software. Under the **htdocs** directory all the websites are stored with their related own files and directory in windows the file system, and path is **c:\xampp\htdocs**. To create a new website project in xampp, we have to create a sub directory under the htdocs. For example we choose named as **myceg/** directory name. This directory behaves like website location which have path **c:\xampp\htdocs\myceg**, (example implemented in Microsoft Windows 7 OS). To create web-pages in PHP we may use editor like dream-viewer / Notepad / Eclipse. In following example we listed the web root directory by typing URL in the web browser as: **http://localhost/myceg**



Figure 7.3 Web root directory location in XAMPP software

### 7.2.1 Writing first PHP scripts and its tags

In Php we code all the statements in between following tags:

```
<?php
```

```
//—————Your code here—————
```

```
?>
```

Only above php tags are read by the interpreter of php, so whenever we create php web pages always write code in same tags. The example shown in the following figure 7.4 shows how to code and where to save the php document.



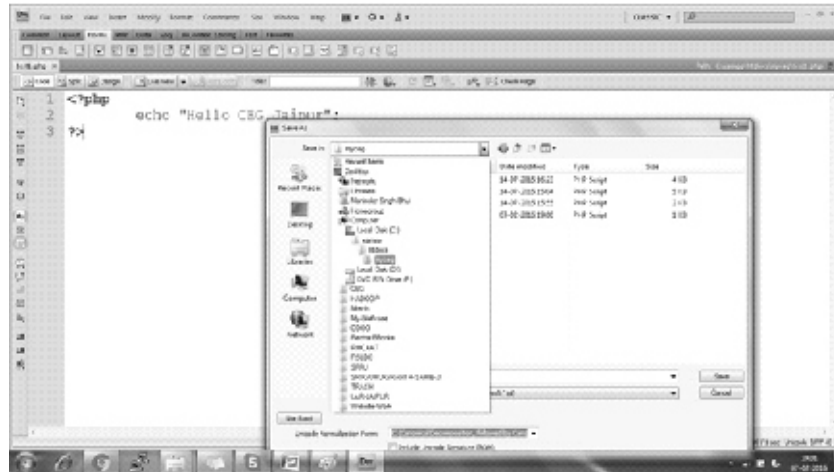


Figure 7.4 Php script saving

Remember that the extension of php page must be **\*.php**, here \* sign means name of web page and **.php** is extension. For example we have write “Hello World” program to execute this php script, we use here **echo** keyword for print purpose each and every statements are terminated by semicolon (;) only. The code is as follows:

**<?php**

**echo “Hello World”;**

**?>**

### 7.2.2 Variable & Identifiers in Php script

In PHP, when we define a variable in PHP it starts with the \$ sign, followed by the name of the variable:

#### Identifiers rule:

- \* An identifier can consist of one or more characters and must begin with a letter or an underscore. It means, identifiers can consist of only letters, numbers, underscore characters.
- \* Identifiers are case sensitive. Therefore, a variable named \$recipe is different from a variable named \$Recipe, \$rEciPe, or \$recipE.
- \* Identifiers can be of any length. This is advantageous because it enables a programmer to accurately describe the identifier’s purpose by using the identifier name.

- \* An identifier name can not be identical to any of PHP's predefined keywords. You can find a complete list of these keywords in the PHP manual appendix.

```
<?php
```

```
$a = "Hello World";
```

```
$b = 123;
```

```
$c = 10.5;
```

```
?>
```

While using PHP script, when we want to store some values in the code, which vary time to time with its value, the identifier denotes the variable name. Such identifier known as variable identifier or variable. In above script \$a , \$b and \$c are three variables and \$ symbol represent them as an variables.

A variable can have a short name (like a or b) or a more descriptive name (age, carname, total\_volume).

Rules to declare PHP variables:

- \* A variable can starts with the \$ sign, followed by the name of the variable.
- \* A variable name must start with a letter or the underscore character.
- \* A variable name cannot start with a number.
- \* A variable name can only contain alpha-numeric characters and underscores like A-z, 0-9, and \_.
- \* Variable names are case-sensitive (\$age and \$AGE are two different variables).

**Output of Variables:**

```
<?php
```

```
$a = "Hello World";
```

```
$b = 123;
```

```
$c = 10.5;
```

```
echo $a . $b . $c;
```

```
?>
```

**The output of above script is:**

**Hello World 123 10.5**

**NOTE:** We use (.) Operator in above script between the variable names to concatenate the variable output of variables.

### **7.2.3 Output statements in Php**

Without the output statements we can't proceed because each and every program that have three essential parts. In every programming we can define the life cycle of program in three phases:

- a) INPUT
- b) PROCESS
- c) OUTPUT

For example:- Like as in C, C++ programming the inbuilt functions are used printf() and cout<< same in php the following two functions / keywords are used to get output.

i) **echo statement:** It is an type of statement which is mostly used for output in php script. It can also use as function , and even it does not have limit of arguments also accept mix data types of arguments. So that we can pass **n** number of arguments in the parenthesis ( ) of function name.

**Important:** The echo statement can be used with or without parentheses: echo or echo().

Syntax: void echo(String \$arg, , , )

In above syntax there is no limit in passing arguments in the function / statement of echo. void is return type of function in syntax, it means no return data types are needed.

ii) **print() statement:** The print statement can be used with or without parentheses: print or print().

syntax: int print( string \$argument)

In above syntax **int (integer)** is a return data type of print() function and only single arguments are used.

### 7.3 Commenting styles in PHP

Comment styles are very important part in any of the programming or scripting, where programmer can hide some line of code that does not require to be executed. By commenting one or more lines of code, the commented part never seen in output window. Comment can be use for describe the programme type, date of written code, author name and those line(s) of code which seems to be bad logic as output. It means the commented area cannot be executed by compiler / interpreter.

PHP uses following two commenting styles:

**1) C / C++ Comment Style:** Single or multiple lines of code can be hide like C and C++ programming.

**(a) Single line comment:** We can put double forward slashes (//) in front of line or code pattern. So that we can hide / comment the particular line.

Ex: `<?php`

```
$a = 1234;
// $b = "hello world";
$c = 55.5;
echo $a;
echo $b;
echo $c;
```

`?>`

**Output is: 1234 55.5**

**Note:** \$b variable has been commented so it not showing in output.

**(b) Multiple line comment:** We can use `/* — */` pattern to hide / comment the multiple lines of code.

`<?php`

```
for($i=0; $i<=5;$i++)
{
```

```

 echo $i ;
 /* echo $i++;
 echo $i+2;
 echo "Hi"; */
}
?>

```

**Output is: 0 1 2 3 4 5**

**Note:** Remaining three lines of code has been commented so they are not showing in output.

**2) Unix style comment:** The “one-line” comment styles only comment to the end of the line or the current block of PHP code, whichever comes first.

```

<?php
 echo 'This is a test'; // This is a one-line c++ style comment
 /* This is a multi line comment
 yet another line of comment */
 echo 'This is yet another test';
 echo 'One Final Test'; # This is a one-line Unix shell-style
 comment
?>

```

#### **7.4 Data types in Php:**

Php have mainly three types of data type:

- 1) Scalar data type
- 2) Compound data type
- 3) Special data type

**1) Scalar data type:** Such types of data types are also known as scalar / single data types. It means such data types can store single value. The scalar variables are those which containing only integer, float, string or boolean data types.

**a) Integer:** An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647. Rules for integers:

- \* An integer must have at least one digit
- \* An integer must not have a decimal point
- \* An integer can be either positive or negative
- \* Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

In the following example of code \$a is an integer type of variable. The PHP function i.e. inbuilt function `var_dump()` function returns the data type and value:

```
<?php
 $a = 44785;
 var_dump($a); # It will show the data type of variable.
?>
```

**b) float / double:** A float (floating point number) / double data types are numbers with a decimal point or a number in exponential form.

In the next example \$b variable is a float / double type of variable . The PHP **var\_dump()** inbuilt function returns the data type and value:

**For Example:**

```
<?php
 $b= 447.56;
 var_dump($b); # It will show the data type of variable.
?>
```

**c) string data type:** A string is a sequence of characters, like “Hello Rajasthan “. A string can be any text inside quotes. You can use single or double quotes:

```

 <?php
 $a = "Hello Rajasthan ";
 $b = 'Hello Jaipur ';
 echo $a;
 echo "
"; # It will insert a new line after print the value $a variable
 echo $b;
?>

```

**d) Boolean data type:** A Boolean represents two possible states: true or false.

```

$x = true;
$y = false;

```

Booleans are often used in conditional testing.

**2) Compound Data Type:** Compound data can contain multiple values. PHP has 2 compound data types: array and object.

**a) Array data type:** An array stores multiple values in one single variable.

In the following example to print full name of student, variable name is \$fullname type of an array. The PHP var\_dump() function returns the data type and value:

```

<?php
$fullname = array("Ekam","Jeet","Singh");
var_dump($fullname); # Output is: EkamJeet Singh
?>

```

**b) Object data type:** An object is a data type which stores data and information on how to process that data. In PHP, an object must be declared explicitly.

A class is a structure that can contain properties and methods. For example:

```

<?php
class Bike {

```

```

function Bike() {
 $this->model = "Welcome to Indian brand of bike";
}
}

// create an object

$bajaj = new Bike();

// show object properties

echo $bajaj->model;

?>

```

**Output is : Welcome to Indian brand of bike**

In above example \$bajaj declared as object using **new** keyword.

**3) Special Data Type:** There are two special data types in PHP that have special meanings:

**Resource:** It holds a reference (link) to an external resource, e.g. file or database (mysql connectivity) connection.

**Null:** If a variable is null, it means the variable does not contain any value.

### 7.5 Constant declaration in PHP :

A constant is an identifier name for a simple value in any programming or scripting. As the name suggests, that value can't change during the execution of the script (except for magic constants, which are not actually constants). A constant is case-sensitive by default. By convention, constant identifiers are always defining in uppercase. We use define() function to declare constant in php.

The name of a constant follows the same rules as any label in PHP.

```

<?php

// Following are valid constant names

define("RAJ1", "education");

```



```
define("RAJ2", "is");
define("BOARD3", "essential for everyone");
```

**// Following are Invalid constant names**

```
define("2FOO", "something");
```

**// Following is valid, but should be avoided:**

```
define("__RAJ4__", "Hello World");
```

?>

## 7.6 function in php:

PHP functions are similar to other programming languages like C , C++ and Java. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value. Where in C programming explain with many functions like **fopen()** and **fread()** in file handling. Same in php built-in functions gives you option to create your own functions as well.

### \* Creating a PHP Function

In fact you hardly need to create your own PHP function because there are already more than 1000 of built-in library functions created for different area and you just need to call them according to your requirement.

It is very easy to create our own PHP function. Let's take an example that we want to create a PHP function which will simply gives output as simple message "Welcome in Rajasthan" on your browser wall when you will call it. In Following example creates a function called **printMessage()** and then calls it just after creating it. Note that while creating a function its name should start with keyword function and all the PHP code should be put inside { and } braces as shown in the following example below:

```
<?php

/* Defining a PHP Function */

function printMessage() {
 echo "Welcome in Rajasthan";
}

/* Calling a PHP Function */
```

```
printMessage ();
```

```
?>
```

---

**OUTPUT: Welcome in Rajasthan**

Example of passing arguments in function:

```
<?php
```

```
function multiplex($x,$y)
```

```
{
```

```
 $z = ($x * $y);
```

```
 return($z);
```

```
}
```

```
 multiplex(5,4); # calling the function with arguments
```

```
?>
```

**OUTPUT is: 20**

### **7.7 Control statements:**

We can divide different types of control statements in four parts-

- 1) Sequential statements
- 2) Conditional statements
- 3) Loop Statements
- 4) Jumping statements

**1) Sequential statements:** Sequential statements are executed one after the another as they appear in the design from the top of the process body to the bottom sequentially.

**2) Conditional statements:** PHP scripting also allows you to write code that perform different actions based on the results of a logical or comparative test conditions at run time. This means, you can create test conditions in the form of expressions that

evaluates to either true or false and based on these results you can perform certain actions.

There are several statements in PHP that you can use to make decisions:

- \* The if statement
- \* The if...else statement
- \* The if...else, if...else statement
- \* The switch...case statement

◆ **if statement:** this statement have very importance in programming because by using execution of statement we can control the flow of our program. The if construct is one of the most important features of many languages, PHP included. It allows for conditional execution of code fragments. PHP features an if structure that is similar to that of C.

**Syntax:**

```
if (expr) {
 code to be executed if condition is true;
}
```

For Example:

```
<?php
 if ($a > $b)
 echo "a is bigger than b";
?>
```

◆ **if-else statement:** The if....else statement executes some code if a condition is true and another code if that condition is false.

**Syntax:**

```
if (condition) {
 code to be executed if condition is true;
} else {
```

code to be executed if condition is false;

}

For example:

```
<?php
 $a = 40;
 $b = 50;
 if($a > $b)
 {
 echo "$a is greater than $b";
 }else{
 echo "$b is greater than $a";
 }
?>
```

◆ **Nested if—else statement:** Nested if statements means an if block inside another if block. Shortly a control structure inside another control structure.

**Syntax:**

```
if (expression 1)
{
 if (expression 2)
 {
 // statements 1
 }
 else
 {
 // Statements 2
 }
}
```

```

}
}
else
{
if (expression 2)
{
// Statements 3
}
else
{
// Statements 4
}
}

```

◆ **if—elseif—else ladder statement:** When PHP evaluates your If...elseif...else statement it will first see if the If statement is true. If that tests comes out false it will then check the first elseif statement. If that is false it will either check the next elseif statement, or if there are no more elseif statements, it will evaluate the else segment, if one exists.

**Syntax:**

```

if(expression or condition 1)
{
statement 1;
}elseif(expression or condition 2)
{
statement 2 ;
}else{

```

```
 statement 3;
 }
```

**For example:**

```
<?php
$employee = "Maninder singh";
if($employee == "Maninder singh"){
 echo "Hello Sir";
} elseif($employee == "kirti"){
 echo "Good Morning mam";
} else {
 echo "Morning ekam";
}
?>
```

◆ **switch—case statement:** Use the switch statement to select one of many blocks of code to be executed.

Syntax:

```
switch (n) {
 case label1:
 code to be executed if n=label1;
 break;
 case label2:
 code to be executed if n=label2;
 break;
 case label3:
 code to be executed if n=label3;
```

```
 break;
 ...
default:
 code to be executed if n is different from all labels;
}
```

**For example:**

```
<?php
 $percentage = 55.85;
 $a = (int) $per;
 Switch($a)
 {
 Case ($a =<85 && $a >= 60):
 Echo "Ist division";
 Break;
 Case ($a =<45 && $a >= 59):
 Echo "IInd division";
 Break;
 Case ($a =<36 && $a >= 44):
 Echo "IIird division";
 Break;
 Default:
 Echo "fail";
 Break;
 } ?>
```

**3) Iterative statements:** Often when we write code, and we need the same block of code to run over and over again in a row. Instead of this by adding several almost equal code-lines in a script, we can use iteration / loops to perform such task like this. It means Loops in PHP are used to execute the same block of code a specified number of times.

In PHP, we use the following looping statements:

- \* for - loops through a block of code a specified number of times.
- \* while - loops through a block of code if and as long as a specified condition is true.
- \* do -while - loops through a block of code once, and then repeats the loop as long as a special condition is true.
- \* foreach - loops through a block of code for each element in an array.

To use any loop / iterations in the programming having three essential parts:

- i) **Initialization:** Initialize the loop counter value, for this most of the times we use = assignment operator to start the loop.
  - ii) **Condition:** Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
  - iii) **size increment / decrement:** Increases or decrease the loop counter value
- ◆ **for loop:** In programming most of the times when we talk about loops, the name of for loop comes first of all. In this loop, we use three segments and separate by ; semi colon each other. We use such loop when we know how many time we need to repeat the code. It means when certainty occurs for loop come forward.

**Syntax:**

```
for(initialization ; condition ; step size part)
{
 statements
}
```

**For example:**

```
<?php
```

(255)



```

for($i=1 ; $i<=20 ; $i++)
 {
 echo $i . "
";
 }

```

?>

**Output is :**

```

1
2
3

```

◆ **while loop:** while loops are the simplest type of loop in PHP. They behave just like their C counterparts. The basic form of a while statement is:

Syntax:

```
while (expr)
```

```
 Statement
```

The meaning of a while statement is simple. It tells PHP to execute the nested statement(s) repeatedly, as long as the while expression evaluates to TRUE. The value of the expression is checked each time at the beginning of the loop, so even if this value changes during the execution of the nested statement(s), execution will not stop until the end of the iteration (each time PHP runs the statements in the loop is one iteration). Sometimes, if the while expression evaluates to FALSE from the very beginning, the nested statement(s) won't even be run once. Overall while loop when, condition is uncertain, means we cannot guess when will be the condition true or available, it continue till then.

**For Example:**

```
<?php
```

```
 /* example 1 */
```

```
 $i = 1;
```

```
 while ($i <= 10) {
```

```
 echo $i++; /* the printed value would be
```

```

 $i before the increment
 (post-increment) */
 }
 /* example 2 */
 $i = 1;
 while ($i <= 10):
 echo $i;
 $i++;
 endwhile;
?>

```

◆ **do-while loop:** The PHP do-while loop statement allows you to execute a block of code repeatedly based on a condition. Unlike the while loop statement whose the condition is checked at the beginning of each iteration, the condition in the PHP do-while statement is checked at the end of each iteration. It means the loop will execute at least once even the condition is evaluated to false.

**Syntax:**

```

 do {
// code block to be executed
 } while (expression);

```

For example:

```

 <?php
 $i = 0;
 do {
 echo $i;
 } while ($i > 0);
?>

```

◆ **foreach loop:** The foreach construct provides an easy way to iterate over arrays. foreach works only on arrays and objects, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variable. There are two syntaxes:

```
foreach (array_expression as $value)
```

```
statement
```

```
foreach (array_expression as $key => $value)
```

```
statement
```

**Note:** The first form loops over the array is given by **array\_expression**. On each iteration, the value of the current element is assigned to \$value and the internal array pointer is advanced by one.

The second form will additionally assign the current element's key to the \$key variable on each iteration.

**4) Jumping statements:** Sometimes we want to let the loops being without any condition, and allow the statements to go inside the brackets to decide when to exit the loop. There are two special statements that can be used inside the loops: **Break** and **Continue**.

Now we will try to understand a brief knowledge of break and continue keywords:

**Break** - ends execution of the current structure. Break accepts an optional numeric argument which tells it how much execution of nested structures to be interrupted.

**Continue** - is used to stop processing the current block of code in the loop and goes to the next iteration.

## 7.8 Array:

By the performance point of view array is an strongest data types in programming, because array is a special variable, which can hold more than one value at a time means An array stores multiple values in one single variable.

Syntax to create array in php:

```
array([$mixed data types ,$......]);
```

### 7.8.1 Types of Array:

In PHP, there are three types of array:

- i. Indexed arrays - Arrays with a numeric index

- ii. Associative arrays - Arrays with named keys
- iii. Multidimensional arrays - Arrays containing one or more arrays

**i. Indexed arrays:**

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

```
<?php
 $cars = array("Rajasthan", "higher", "education");
 print_r($cars)
?>
```

Or

```
<?php
 $cars[0] = "Rajasthan";
 $cars[1] = "higher";
 $cars[2] = "education";
 print_r($cars);
?>
```

**ii. Associative arrays :** Associative arrays are arrays that use named keys that you assign to them. There are two ways to create an associative array:

```
<?php
 $age = array("maninder"=>"34", "guurpreet"=>"38", "jagpreet"=>"40");
 print_r($age);
?>
```

Or

```
<?php
 $age[maninder] = "34";
```

```
$age['gurupreet'] = "38";
$age['jagpreet'] = "40";
print_r($age)
```

?>

**iii. Multidimensional array:** A multidimensional array is an array containing one or more arrays. PHP understands multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

The dimension of an array indicates the number of indices you need to select an element.

- \* For a two-dimensional array you need two indices to select an element
- \* For a three-dimensional array you need three indices to select an element

<?php

```
$cars = array(
 array("Volvo",22,18),
 array("BMW",15,13),
 array("Saab",5,2),
 array("Land Rover",17,15));

echo $cars[0][0].": In stock: ".$cars[0][1].", sold:
".$cars[0][2]."
";

echo $cars[1][0].": In stock: ".$cars[1][1].", sold:
".$cars[1][2]."
";

echo $cars[2][0].": In stock: ".$cars[2][1].", sold:
".$cars[2][2]."
";

echo $cars[3][0].": In stock: ".$cars[3][1].", sold:
".$cars[3][2]."
";
```

?>

### 7.8.2 Super global array in php:

Several predefined variables in PHP are known as “superglobals”, which means that they are always accessible, regardless of scope - and we can access them from any function, class or file without having to do anything special in the source code of php script.

The PHP superglobal variables are:

- \* `$GLOBALS[]`
- \* `$_SERVER[]`
- \* `$_REQUEST[]`
- \* `$_POST[]`
- \* `$_GET[]`
- \* `$_FILES[]`
- \* `$_ENV[]`
- \* `$_COOKIE[]`
- \* `$_SESSION[]`

This chapter we will emphasis some of the super global array variables, like : `$GET[]`, `$POST[]` only. All above super global variables are associative types of array. As we mentioned we will discuss about only `$_GET` and `$_POST`, so let us see:

`$_GET[]` : PHP `$_GET` can also be used to collect form data after submitting an HTML form with `method="get"`.

`$_GET` can also collect data sent in the URL.

**Note:** Assume we have an HTML page that contains a hyperlink with parameters:

```
<html>
```

```
<body>
```

```

```

```
 Click $GET
```

```

```

```
</body>
```

```
</html>
```

In above script the saved webpage name is : test1.php

**Note:** When someone click on Click hyperlink shown in above code of HTML, it will automatically redirect to ceg\_register.php and print values like :

```
<?php
```

```
echo "Course Name You selected: " . $_GET['coursename'] . " at URL " .
$_GET['website'];
```

```
?>
```

In above script the saved webpage name is: ceg\_register.php

\$\_POST: In PHP, \$\_POST is widely used to collect form data after submitting an HTML form with method="post". \$\_POST is also widely used to pass variables. The data which passes through URL, are always hide. Let us take an example of \$\_POST, for this we create a webpage named as : test2.php and pass some argument in URL of browser. In following example as we use action as a same page for testing purpose.

```
<html>
```

```
<body>
```

```
<form method="post" action="test2.php">
```

```
Student Name: <input type="text" name="sname">
```

```
<input type="submit" value="Save" name="b1">
```

```
</form>
```

```
<?php
```

```
if(isset($_POST['b1']))
```

```
{
```

```
 echo "Welcome ".$_POST['sname'];
```

```
}
```

```

?>

</body>

</html>

```

**Note:** When user execute this page in xampp server, it shows you a HTML webform, and when user enter the name and click on submit button which seen as Save name, for example you enter string as **ekam**, then you will get output: **Welcome ekam**.

### 7.9 Operators:

In any programming we try to obtain different types of results for any arithmetic / mathematical problem, so then we perform logical / mathematical operations in computer. To resolve all types of calculations we use symbolic notations with operands. These symbols are used to control the programming to solve the complex computations, it means these symbolic notation refer as Operators and both side of values in the calculation, are known as Operands. The associativity of operator in PHP are Right to Left, and when we solve a problem of expression in programming of PHP, the precedence are showing below (in figure) that which one of the operation will occur first and later respectively. To explain precedence of the operator, we need to understand following table:

## Operator Precedence

Associativity	Operators	Additional Information
non-associative	new	new
left	[	array()
non-associative	++ --	increment/decrement
non-associative	~ - (int) (float) (string) (array) (object) @	types
non-associative	instanceof	types
right	!	logical
left	* / %	arithmetic
left	+ - .	arithmetic and string
left	<< >>	bitwise
non-associative	< <= > >=	comparison
non-associative	== != === !==	comparison
left	&	bitwise and references
left	^	bitwise
left		bitwise
left	&&	logical
left		logical
left	? :	ternary
right	= += -= *= /= .= %= &=  = ^= << >>=	assignment
left	and	logical
left	xor	logical
left	or	logical
left	, (comma)	many uses



**7.9.1 Arithmetic Operator:** To solve different types of arithmetic expressions. We use five types of operator with their priority sequence:

\* , % , + , - , /

Arithmetic Operators		
Example	Name	Result
-\$a	Negation	Opposite of \$a.
\$a + \$b	Addition	Sum of \$a and \$b.
\$a - \$b	Subtraction	Difference of \$a and \$b.
\$a * \$b	Multiplication	Product of \$a and \$b.
\$a / \$b	Division	Quotient of \$a and \$b.
\$a % \$b	Modulus	Remainder of \$a divided by \$b.
\$a ** \$b	Exponentiation	Result of raising \$a to the \$b'th power. Introduced in PHP 5.6.

**7.9.2 String Operator:** In PHP mainly, The string may be define with two PHP operators. The first one is period / dot ( . ) operator, and second one is used to demonstrate decimal types of values. For example 15.6 is floating point number and we want to print this float type number with spaces in php script so that we will try with the following script:

```
<?php
 $a = 15.6;
 echo "Hello world ". $a. "
 ". "Thanks a lot";
?>
```

**Output is :** Hello world  
15.6  
Thanks a lot

**Note:** In above example we use dot operator in floating point number and also use it for concatenation purpose to concat different types of operands like string , break line code and then string again.

**7.9.3 Increment and Decrement Operator:** These are same as other operators, we denote increment operator as ++ and — for decrement operator. The importance of operator is they only use with single Operand. The operator listed in the following table format showing along with their description.

Operator	Name	Description
++\$x	Pre-increment	Increments \$x by one, then returns \$x
\$x++	Post-increment	Returns \$x, then increments \$x by one
--\$x	Pre-decrement	Decrements \$x by one, then returns \$x
\$x--	Post-decrement	Returns \$x, then decrements \$x by one

```

<?php
 echo "<h3>Postincrement</h3>";
 $a = 5;
 echo "Should be 5: “ . $a++ . “
\n”;
 echo "Should be 6: “ . $a . “
\n”;
 echo "<h3>Preincrement</h3>";
 $a = 5;
 echo "Should be 6: “ . ++$a . “
\n”;
 echo "Should be 6: “ . $a . “
\n”;
 echo "<h3>Postdecrement</h3>";
 $a = 5;
 echo "Should be 5: “ . $a— . “
\n”;
 echo "Should be 4: “ . $a . “
\n”;
 echo "<h3>Predecrement</h3>";
 $a = 5;
 echo "Should be 4: “ . —$a . “
\n”;
 echo "Should be 4: “ . $a . “
\n”;
?>

```

**7.9.4 Equal operator:** Such types of operators are used to compare two values in scripting. The description given in the following table

Comparison Operators

Example	Name	Result
\$a == \$b	Equal	TRUE if \$a is equal to \$b after type juggling.
\$a === \$b	Identical	TRUE if \$a is equal to \$b, and they are of the same type.
\$a != \$b	Not equal	TRUE if \$a is not equal to \$b after type juggling.
\$a <> \$b	Not equal	TRUE if \$a is not equal to \$b after type juggling.
\$a !== \$b	Not identical	TRUE if \$a is not equal to \$b, or they are not of the same type.
\$a < \$b	Less than	TRUE if \$a is strictly less than \$b.
\$a > \$b	Greater than	TRUE if \$a is strictly greater than \$b.
\$a <= \$b	Less than or equal to	TRUE if \$a is less than or equal to \$b.
\$a >= \$b	Greater than or equal to	TRUE if \$a is greater than or equal to \$b.

**7.9.5 Relational Operator:** In php and other scripting, a relational operator is a programming language construct or operator that tests or defines some kind of relation between two entities as listed following:

Operator	Name	Example	Description
==	Equal	\$x == \$y	Returns true if x and y are equal
===	Identical	\$x === \$y	Returns true if x and y are equal and of same type
!=	Not equal	\$x != \$y	Returns true if x and y are not equal
!==	Not identical	\$x !== \$y	Returns true if x and y are not equal and of same type
<	Less than	\$x < \$y	Returns true if x is less than y
<=	Less than or equal to	\$x <= \$y	Returns true if x is less than or equal to y
>	Greater than	\$x > \$y	Returns true if x is greater than y
>=	Greater than or equal to	\$x >= \$y	Returns true if x is greater than or equal to y
<>	Not equal	\$x <> \$y	Returns true if x and y are not equal

**7.9.6 Logical Operator:** Such operator is used to check logical calculations and its results, whether they are equal or not. They also work with two operands, and return true or false value respectively.

Example	Name	Result
\$a and \$b	And	TRUE if both \$a and \$b are TRUE.
\$a or \$b	Or	TRUE if either \$a or \$b is TRUE.
! \$a	Not	TRUE if \$a is not TRUE.
\$a && \$b	And	TRUE if both \$a and \$b are TRUE.
\$a    \$b	Or	TRUE if either \$a or \$b is TRUE.

**7.9.7 Bitwise Operator:** Such operators work on binary digit means bit. Rest of all the operators are work on value of the byte.

### Bitwise Operators

Example	Name	Result
$\$a \& \$b$	And	Bits that are set in both $\$a$ and $\$b$ are set.
$\$a   \$b$	Or (inclusive or)	Bits that are set in either $\$a$ or $\$b$ are set.
$\$a \wedge \$b$	Xor (exclusive or)	Bits that are set in $\$a$ or $\$b$ but not both are set.
$\sim \$a$	Not	Bits that are set in $\$a$ are not set, and vice versa.
$\$a \ll \$b$	Shift left	Shift the bits of $\$a$ $\$b$ steps to the left (each step means “multiply by two”)
$\$a \gg \$b$	Shift right	Shift the bits of $\$a$ $\$b$ steps to the right (each step means “divide by two”)

**7.9.8 Assignment operator:** In Php, we use = operator to denote as assignment operator. In any programming, the calculation is being solved in the expression from right to left according to precedence rule and solve the expression and save into a variable. For example:

```
<?php
 $a = $b + $c ;

 # In above = operator use for assign the addition result in
 $a variable

?>
```

### 7.10 MySQL Database

As describe in previous chapters we have learnt about database such as MySQL / MS-SQL in details and have been practices on command line. We also have learnt how to code on command line in SQL. In this chapter we will use the GUI (Graphical User Interface) of MySQL and also try to create database and its table as graphically. Once again we define the MySQL database “A database is a collection of information that is organized so that it can be easily accessed, managed and updated”. Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is

added. Databases process workloads to create and update themselves, querying the data they contain and running applications against it.

In the tables of database, all the columns are also known as fields of the table. Remember that all the records of the tables are denotes the common feature of the database table.

The database and its table structure are as depict as following:



Figure 7.5 Sample MySQL Database and listing of tables

For example, we would like to take example of any school and its students. When we want to store the information of the students in the database and its table, for this we will use SQL code to insert the records in it. So it means, in future when we require to print data on HTML page in bulk as output. At that time we will use Php to fetch data from database and MySQL as back end.

**7.10.1 Phpmysql:** phpmysql is free and open source software for mysql access. Actually mysql come with command line, but when we use phpmysql it means we can access mysql platform as GUI also. The phpmysql project comes along with XAMPP software which included: Apache, MySQL and Php. In this bundled software the MySQL comes in PhpMyadmin project which can be access over the browser with **http://** protocol. The method is showing in following snapshot:

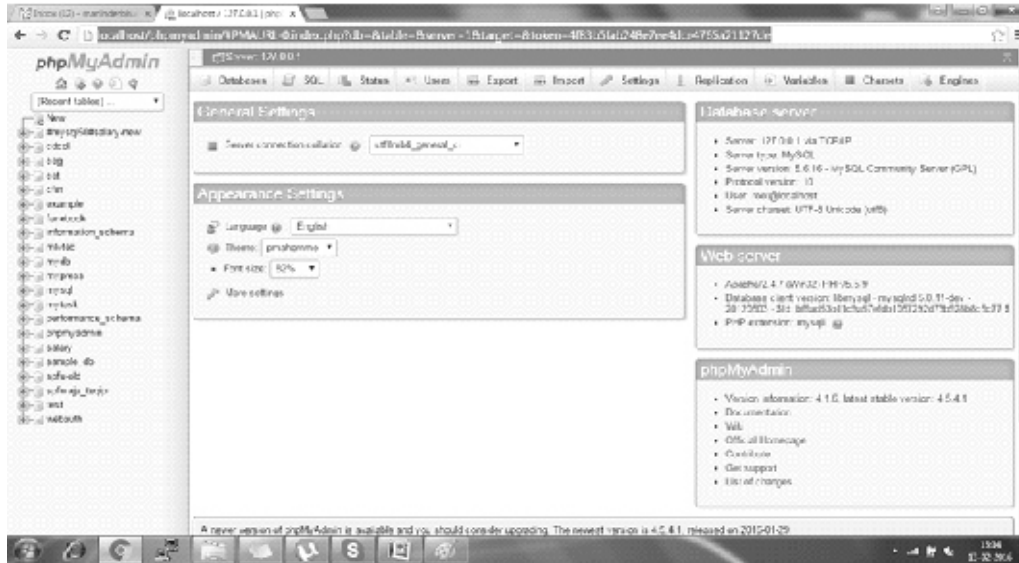


Figure 7.6 Welcome screen of GUI MySQL (i.e. phpmyadmin screen for MySQL access)

We can access GUI mysql, in web browser by typing URL section as `http://localhost/phpmyadmin` and just open it. Remember it the XAMPP software Apache and MySQL services are running with the TCP port number 80 and 3306 respectively, if they are not running then try to run / execute by click on start the services.

The following diagram illustrates how the constraints are apply on columns of the mysql table. We use different-different types of SQL for applying constraint, by which we can store unique data in columns, so that no one can insert record as repeatedly. For this we use following types of constraints (Criteria) :

- 1) primary key
- 2) unique key
- 3) foreign key
- 4) null / not null
- 5) check

In the next diagram the database in form of tables showing record (data) of the students. The table and its data gives us information how it was stored in row and column format. For example, name of the student, parents name and his / her belonging information can be store for the future aspect.

**For Example:**

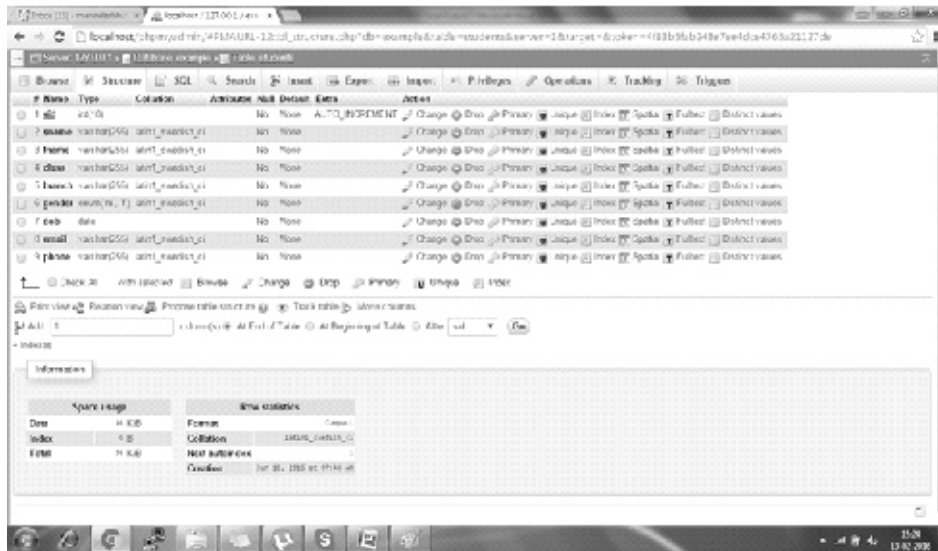


Figure 7.7 constraints and data types for the fields of mysql table

In above diagram the tables shows us every features of the database table which keeps the information also the tables gives us information how the constraint(Criteria) and data types are used simultaneously.

**7.10.2 Connectivity code for Php with MySQL:** When we want to connect php script and mysql database for getting stored data in the database for many sql operations like: Fetch , Insert, Delete , Update and Search etc. That's why we need to connect php webpage with mysql database. Every time while connecting php with mysql, remember that mysql user name and password must be secure and not shared with anyone. The following code is an example for connectivity of php script and mysql database:

```
<?php
 Mysql_connect("localhost","root","") or die(mysql_error());
 Mysql_select_db("ceg") or die(mysql_error());
?>
```

**Note:** in above php script we create an database named as **ceg** and its table name is student. Now when we want to connect php page with myql for fetching students record, we must use such code for the connectivity purpose, and save above script as connect.php and save this under the file system, **c:\xampp\htdocs\myceg**.

In above location of the project is cegjaipur and we need to create a directory under the file system c:\xampp\htdocs\ and then we need to save connect.php code there. After this we can run this script by typing in the URL section of the browser as follows:

**`http://localhost/myceg/connect.php`**

When a another php page require an connection establishment with database, then we can call the same connect.php code by using include() or require() function in php script. It means no need to create connectivity code again and again.

**For example:**

```
<?php
 include("connect.php");
 $a = mysql_query("select * from students");
 echo "Hello Students how are you ";
?>
```

In above example the mysql\_query() function will execute the query, when the connect.php is working fine, It means connectivity working properly otherwise we get an error message.

**7.10.3 Data types in mysql:** In the previous chapter we learnt about the data types in details. So we are listing some of the basic data types available in mysql :

- 1) date & time
- 2) timestamp
- 3) boolean
- 4) varchar
- 5) char
- 6) enum
- 7) int
- 8) float



9) double

11) text

**7.10.4 Database attribute:** There are in-built types of attribute which are available in the mysql database, such as **auto\_increment**, which increment the value of integer type of data type in the table of mysql. For example when we store data in the table of mysql, just imagine about students records, in which we want to identified the row number and that particular number never repeat or null / blank type, so that we use auto increment with constraint of primary key. Always remember that the particular key must be integer type and being auto incremental. This feature also allow us to count all the number of records have been inserted in the table.

**7.10.5 Index attributes:** As finding one word from the dictionary which contains thousand of words, similarly in database when we trying to find a particular data out of thousands sets of records in database, we mostly used index attribute by implement this feature on a particular table. Many times the database administrator apply such attribute manually on table, but it implement automatically while we use primary and unique key on fields of the table. Always remember that when a newly tabled created in mysql database, it always have constraint of unique and not null types.

### **Important Points**

- 1 Always install XAMPP / WAMP software, to use apache, mysql and php.
- 2 Always create web projects under the file system: c:\xampp\htdocs\
- 3 Php script can be write in any of the IDE ( Interface development environment)
- 4 Always write the php code in between **<?php ?>** tags.
- 5 **echo** keyword can be used for output statement.
- 6 Php was developed by Rasmus Lerdorf.
- 7 We comment single line of code in php by using **//** and by using **#**.
- 8 In php there are four types of control statements: Sequential, conditional, iterative, jumping.
- 9 To show logical expression, we use operator and operands.
- 10 Mysql is a Relational database management system.

- 11 By connectivity code in php we may perform many operations like: DQL (Data Query Language), DML(Data Manipulation Language), DDL(Data Definition Language), and many more.
- 12 The TCP port number for access apache webserver normally (http://) and securely (https://) is 80 and 443 respectively. The mysql use TCP port number 3306.

### Exercises

#### Objective Type Questions

1. Php was developed by:
 

a) Radarford	b) Bill gates
c) Rasmus Lerdorf	d) Jems Robert
2. How to initialize a variable in php:
 

a) #	b) @
c) \$	d) *
3. Which one of the Open Source bundled software is used for php and mysql :
 

a) JDK	b) Microsoft .NET
c) Eclips	d) XAMPP
4. Which port number is used by the Apache webserver:
 

a) 82	b) 85
c) 90	d) 80
5. Which port number is used by the mysql server
 

a) 3122	b) 8080
c) 3000	d) 3306
6. To start php script which one of the scriptlet tags are being used ?
 

a) <?php _____ ?>	b) <!php _____ !>
c) <%php _____ %>	d) <%php _____ ?>
7. Which one is example of loop / iteration in php script ?
 

a) while	b) float
c) int	d) long
8. Which one is the conditional statement type ?
 

a) if—else	b) while
c) for	d) foreach

9. Which one statement is used for the collection of similar data types ?

- a) array
- b) union
- c) constructor
- d) class

10. Which one is the type of associative array of type ?

- a) \$\_GET
- b) array()
- c) for each
- d) class

### Short Type Questions

1. How to write a php script.?
2. How to declare and create a function in php?
3. How to create array in php?
4. Create an expression in php ?
5. What is order of operators in php?
6. Prepare the list of operators.
7. Explain **echo** and print() function.
8. Differentiate between array and class of php.
9. Brief about php and mysql connectivity.
10. Explain the inbuilt function mysql\_connect()

### Essay Type Questions

1. Create a php script to declare and initialize the variable.
2. How to connect php and mysql and also explain inbuilt function: mysql\_connect() and mysql\_select\_db()
3. Create a php script to perform all the arithmetic operations along with the function.
4. What types keys are available in mysql ?
5. Explain in details of all operators available in PHP.

### Answer Key

- |      |       |      |      |
|------|-------|------|------|
| 1. c | 2. c  | 3. d | 4. d |
| 5. d | 6. a  | 7. a | 8. a |
| 9. a | 10. a |      |      |