

Energized Study Material of Computer Science XII



BY



Mr. Javeed Hassan Sofi (MCA/BEd)

Teacher GMS Brar Bandipora



A complete energized Study material of Computer
Science for Class XII



Prepared by DSEK and SIE Kashmir



Copyright © 2020 by DSEK, SIE Kashmir

As per Latest Syllabus of JKBOSE

Contents

Structure	Page No's
Unit 1: Objected-Oriented Programing in C++	02
Classes & Objects	02
Array of Objects	09
Friend Functions	11
Unit 2: Constructors and Destructors	16
Constructors and Destructors	16
Types of Constructor	18
Destructors	24
Unit 3: Inheritance (Extending Class)	25
Inheritance	25
Types of Inheritance	26
Data file handling	36
Unit 4: Pointers	46
Pointers	46
Pointers and Arrays	48
Array of Pointers	49
This pointer	56
Unit 5: Data Structures	57
Basics	57
Stacks, Queues And Linked List	67
Exercise	79
Unit 6: Database and SQL	81
Basics	81
Syntax of some SQL commands	87
Practice work	92
Unit 7: Boolean Logic	97
Boolean Algebra & Theorems	100
Basic Logic Gates	109
Exercise	121
Unit 8: Networking and Cyber Security	123
Switching Techniques	126
Data Communication Terminologies	127
Transmission Media	128
Network Security Concepts	144

Unit 1

Objected-Oriented Programing in C++

Objectives:

In this Chapter you will learn about:

- Classes & Objects
- Advantages of OOPs
- polymorphism
- Access Specifiers



CLASSES AND OBJECTS:

Class:-

A class is a user defined data that contained data as well as function together.

or

A class is a way to bind the data and function together in a single unit.

or

A class is a group of objects that share common properties and relationships

Note:-

1. The variables and functions enclosed in a class are called data member and member functions.
2. The variable of class are called objects or instance of a class .
3. Classes are basic user defined data type of object oriented programming language
4. The difference between a class and structure is that ,by default ,all the members of a class are private while by default ,all the members of structure are public

Class Declaration:-

Class <classname>

{

private:

data members;

member functions;

protected:

data members;

member functions;

```
    public:
        data members ;
        member functions:
};
```

Note:- The data members and member functions of class are grouped under three sections.

1. private
2. protected
3. public

Explanation of private ,protected & public access modifier:-

Private :-

- By private we mean that members can be accessed only from within the class i.e member data can be accessed by the member function .
- The private data members are not accessible to the outside works (i.e outside the class)
- By default members of a class are private.
- Private members are not inheritable.

Protected:-

- The members which are declared as protected ,can only be accessed by member functions and friends function of that class.
- Protected members are similar to private members, the only difference is that protected members are inheritable.

Public:-

The members which are declared as public can be accessed from outside classes also.

Note:- By default ,the members of a class are private .if both the labels, are missing, then by default, all the members are private .Such a class is completely hidden from outside world.

Example:-

```
class Student
{
    int rollno;
    float marks;
    public:
        void getdata(int x,flot y);
```

```
void display(void);  
};
```

Array of Objects:- An array of variables that are of the type Class .such variables are called array of objects.

Example:

```
class employee  
{  
    char name[50];  
    int age;  
    public:  
        void getdata(void);  
        void showdata(void);  
};
```

The employee is a user defined data type and can be used to create objects that relate to different categories of employees.

```
employee manager[4];  
employee engineer[15];  
employee workers[200];
```

Creating Objects:-

- An object is an instance of class.
- In general a class is a user defined data type .while an object is an instance of class
- Once a class has been declared, we can create variables of that type by using class name.
- An object occupies memory space.

Let Student be the name of class

Student S1, S2;

or

Class Student

```
{  
-----  
-----
```

} S1,S2;

Accessing Class Members:- After creating the objects there is a need to access the class members .This can be done using (.) operator.

Syntax:

Object name . Data member
Object name. member function

As:

S1.getdata(129,100)
s1.display()
s1.rollno=129 // illegal statement

Note:- A variable declared public can be accessed by the objects directly.

//Example for public data members

#include<iostream.h>

#include<conio.h>

void main()

{

clrscr();

class date

{

public:

int day;

int month;

int year;

};

date d1;

d1.day=26;

d1.month=6;

d1.year=2012

cout<<" Date is="<<d1.day<<"/" <<d1.months<<"/" <<d1.year<<endl;

getch();

}

Example :- Write a program to make use of simple arithmetic operations such as addition, subtraction, multiplication division using the concept class.

#include <iostream.h>

```
#include<conio.h>
class calculator
{
    private :
    int x;
    int y;
    public :
        void getdata();
{
    cout<<" Enter the two numbers:" << endl;
    cin>> x >> y;
}

void display()
{
    cout<<"x="<<x<<endl;
    cout<<"y="<<y<<endl;
}

void sum()
{
    cout<<" addition of x & y ="<<x+y<<endl;
}

void diff()
{
    cout << " diff of x & y =" <<x-y<<endl;
}

void mul()
{
    cout<<" mul of x & y ="<<x*y<<endl;
}

void div()
{
    cout<< " div of x & y="<< x/y<<endl;
}
};

Void main()
```

```
{
    clrscr();
    calculator C1;
    c1.getdata();
    c1.display();
    c1.sum();
    c1.diff();
    c1.mul();
    c1.div();
    getch();
}
```

Class method definition:- The data members of a class must be declared within the body of the class ,whereas member functions of the class can be defined in following ways:

- 1.Out side the class definition
2. Inside the class definition

Member function outside the class definition:-

- declare function prototype within the body of a class and then define it outside the body of class.
- By using scope resolution operator (: :) we bind the function to the class.

Syntax:-

```
Return type classname :: functionname(argument declaration)
{
    function body
}
```

Example:

```
#include<iostream.h>
#include<conio.h>
class product
{
    int qty;
    float cost;
    Public:
        void getdata(int a, float b);
        void putdata(void);
```

```
};

void product :: getdata(int a,float b)
{
    qty= a;
    cost=b;
}

void product :: putdata(void)
{
    cout<<" qty="<< qty<<endl;
    cout<<"cost=" << cost<<endl;
}

void main()
{
    clrscr();
    product p;
    cout<<" The object ps is "<< endl;
    p.getdata(75,255.60);
    p.putdata();
    product Q;
    cout<<" The object Q is =" << endl;
    Q.getdata(100,125.25);
    Q.putdata();
    getch();
}
```

Member function inside the class definition:

Note:- When a function is defined inside a class ,it is treated as inline function. Normally small functions are defined inside the class definition.

Example:-

```
#include<iostream.h>
#include< conio.h>
class date
{
    private:
    int day;
```

```
int month;
int year;
public:
    void getdata(int x,int y,int z)
{
    day=x;
    month=y;
    year=z;
}

void display()
{
    cout<< day<<month<<year <<endl;
}
}

Void main()
{
    clrscr();
    date d1,d2,d3;
    d1.getdata(30,12,2012);
    d2.getdata(23,11,2011);
    d3.getdata(24,05,2010);
    cout<< " you have entered following date";
    d1.display();
    d2.display();
    d3.display();
    getch();
}
```

Array of Objects:-

An array of variables that are of the type Class .Such variables are called array of objects.

Example:

```
class employee
{
    char name[50];
    int age;
```

```
public:
    void getdata(void);
    void showdata(void);
};
```

The employee is a user defined data type and can be used to create objects that relate to different categories of employees.

```
employee manager[4];
employee engineer[15];
employee workers[200];
```

Example:-

```
#include<iostream.h>
#include<conio.h>
class employee
{
    char name[50];
    int age;
    public:
        void getdata(void);
        void showdata(void);
};
```

```
void employee:: getdata(void)
{
    cout<<"enter name=";
    cin>>name;
    cout<<"enter age=";
    cin>>age;
}
```

```
void employee::showdata(void)
{
    cout<<name<<endl<<age;
}
```

```
void main()
{
    clrscr();
    int size=4;
```

```
    employee manager[size];
for(int i=0; i<size;i++)
{
    cout<<"Enter Details of managers";
    manager[i].getdata();
}
cout<<endl;

for(i=0;i<size;i++)
{
    cout<<"Details of Managers is=";
    manager[i].putdata();
}
getch();
}
```

Friend Function:-

A friend function is not a member function ,has full access rights to the private members of the class.

Syntax :-

```
class ABC
{
    -----
    -----
    public:
        -----
        -----
    friend void xyz(void);
}
```

Characteristics of friend function:

- It is not in the scope of the class to which it has been declared as friend.
- Since it is not in the scope of the class ,it cannot be called using the object of that class.
- It can be invoked like a normal function without the help of any object.

- Unlike member functions, it cannot access the member names directly and has to use an object name and dot membership operator with each member name.
- It can be declared either in the public or the private part of a class without affecting its meaning.

Usually ,it has the objects as arguments.

Example:-

```
#include<iostream.h>
#include<conio.h>
Class sample
{
    int a;
    int b;
    public:
        void setvalue()
{
    a=25;
    b=40;
}

friend float mean(sample s);
};

float mean (sample s)
{
    return float (s.a+s.b)/2;
}

void main()
{
    sample x;
    x.setvalue();
    cout<<mean(x);
}
```

// A function friendly to two classes

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

```
{
    int x;
    public:
        void setvalue(int i)
{
    x=i;
}
friend void max(XYZ,ABC);
};

class ABC
{
    int a;
    public:
        void setvalue(int i)
{
    a=i;
}
friend void max(XYZ,ABC);
};

void max(XYZ m, ABC n)
{
    if (m.x>=n.a)
        cout<<m.x;
    else
        cout<<n.x;
}

void main()
{
    ABC abc;
    abc.setvalue(10);
    XYZ xyz;
    xyz.setvalue(20);
    max(xyz,abc);
}
```

Object as function Arguments:-

An object can be passed as an argument to functions in two ways.

- **pass by value**
- **pass by reference.**

// Example of call by value

```
#include<iostream.h>
#include<conio.h>
class class_2;
class class_1
{
    int value1;
public:
    void indata(int a)
    {
        value1=a;
    }
    void display(void)
    {
        cout<<value1<<"\n";
        friend void exchange(class _1 obj1, class_2 obj2);
    }
}

class class2
{
    int value2;
public:
    void indata(int a)
    {
        value2=a;
    }
    void display(void )
    {
        cout<<value2<<"\n";
    }
    friend void exchange (class _1 t1, class_2 t2);
};

void exchange(class_1 T1,class_2 T2)
{
    int temp;
    temp=T1.value1;
```

```
    T1.value1=T2.value2;
    T2.value2=temp;
}

void main()
{
    class_1 C1;
    class_2 C2;
    C1.indata(100);
    C2.indata(200);
    cout<<" values before exchange "<<"\n";
    C1.display();
    C2.display();
    exchange(C1,C2);
    cout<<" values after exchange"<<"\n";
    C1.display();
    C2.display();
}
```

Unit 2

Constructors and Destructors

Objectives:

In this Chapter you will learn about:

- Constructor And Destructor
- Special characteristics
- Declaration and Definition of constructors
- Types of constructors



CONSTRUCTOR AND DESTRUCTOR:

Constructor:-

A constructor is a special member function which is used to initialize the object of a class.

Note:-

- A constructor is distinct from other member functions because its name is same as the name of the class .It is executed automatically when an object is declared. The only restriction that applies to constructor is that it must not have a return type not even void.
- Constructor of a class is the first member function to be executed automatically when an object of the class is created.

The general syntax:

```
class classname
{
    -----
    -----
    -----
    classname( parameter list);
    -----
};

class name::classname(parameter list)
{
```

```
        -----  
    }
```

Example:

```
class Bank  
{  
    private:  
    int x,y;  
    -----  
    -----  
    public:  
        Bank( );  
};  
  
Bank :: Bank( )  
{  
    x=0;  
    y=0;  
}  
  
void main()  
{  
    Bank B;  
}
```

When an object B is created, constructor will be called automatically and it will initialize the data members x and y=0

Example:-

```
#include<iostream.h>  
#include<conio.h>  
class fibonacci  
{  
    private:  
    int fo,f1,fib;  
    public:  
        fibonacci();  
        void increment();  
        void display();
```

```
};
fibonacci::fibonacci()
{
    fo=0;
    f1=1;
    fib=fo+f1;
}

void fibonacci::increment()
{
    fo=f1;
    f1=fib;
    fib=fo+f1;
}

void fibonacci:: display()
{
    cout<< fib<<“\t”;
}

void main()
{
    clrscr();
    fibonacci F;
    for(int i=0;i<=15;++i)
    {
        F.display();
        F.increment();
        getch();
    }
}
```

Default Constructor:

A constructor that accepts no parameters is called default constructor.

Syntax:

```
class First
{
    private:
        -----
        -----
}
```

```
    public:
        first();
        -----
        -----
};
```

```
First :: First()
{
}
#include<iostream.h>
#include<conio.h>
class student
{
    private:
        char name[50];
        int roll_no;
        char sex;
        int age;
        float marks;
    public:
        student();
        void putdata();
};
```

```
student:: student()
{
    gets(name,"Donald");
    roll_no=12;
    sex='M';
    age=34;
    marks=90;
}
void student:: putdata()
{
    cout<<name<<endl;
    cout<<roll_no<<endl;
    cout<<sex<<endl;
    cout<<age<<endl;
    cout<<marks<<endl;
```

```
}  
void main()  
{  
    clrscr();  
    student S;  
    S.putdata();  
    getch();  
}
```

Parameterized constructor:-

The constructor that can take arguments are called Parameterized constructor.

Example:

```
class Bank  
{  
    private:  
    int x,y;  
    public:  
        Bank(int a, int b)  
        -----  
        -----  
};  
  
Bank:: Bank(int a, int b)  
{  
    x=a;  
    y=b;  
}
```

In the parameterized constructor we pass the initial values as arguments to the constructor in two ways:

- Implicit call
- Explicit call

Implicit call:

Bank b1(5,11);

Explicit Call:

Bank B1=Bank(5,11);

Example:-

```
#include<iostream.h>
#include<conio.h>
class Bank
{
    int x,y;
    public:
        Bank(int ,int);
        void putdata()
        {
            cout<<x<<"\n"<<y;
        }
};

Bank :: Bank(int a,int b);
{
    x=a;
    y=b;
}

void main()
{
    clrscr();
    Bank B1(5,11);
    Bank B2=Bank(6,34);
    cout<<" object 1";
    B1.putdata();
    cout<<"object 2";
    B2.putdata();
    getch();
}
```

Copy Constructor:-

A copy constructor takes a reference to its own class as parameter i.e a constructor having a reference to an instance of its own class as an argument is known as copy constructor. Copy constructor are used in following situations:

1. The initialization of an object by another object of the same class.

Example:

```
Sample S1;                                //default constructor
Sample S2(S1);
    or
Sample S2=S1;
```

The general syntax

```
classname ::classname(classname &)
```

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

```
class sample
```

```
{
    int x;
    public:
    sample()
    {
        x=0;
    }
    sample(int a)
    {
        x=a;
    }
    sample (sample & S)
    {
        x=s.x
    }
    void display(void)
    {
        cout<<x;
    }
};
```

```
void main()
```

```
{
    clrscr();
    Sample P(45);
    Sample Q(P);
    Sample R=P;
    Sample T;
    T=P;
    cout<<" \n value of p:";
```

```
P.display();
cout<<“\n value of Q:”;
Q.display();
cout<<“ \n value of R”;
R.display();
cout<<“\nValue of T:”;
T.display();
getch();
}
```

Constructor with default Arguments:-

Example:

```
item (int x,int y=0);           // declaration
item:: item(int x,int y)
{
    a=x;
    b=y;
}
```

when we make object of item

```
    item P(11);
```

11 will be assign to x and 0 to y;

```
    item Q(11,17);
```

11 to x and 17 to y;

Example:

```
class student
{
    private :
        char name[50];
        int age;
        char sex;
        float marks;
    public:
        student();           // Constructor
        ~ student();         // Destructor
        void getdata();
        void putdata();
};
```

DESTRUCTOR:

A destructor is a member function having same name as its class but preceded by ~ sign and it de-initializes an object before it goes out of scope.

Note:- Object Are Destroyed In The Reverse Order Of Creation. So destructor is Invoked in the reverse order of constructor

Program:-

```
#include<iostream.h>
#include<conio.h>
class sample
{
    public:

        sample( )
        {
            cout<<" object is born\n";
        }
        ~sample ()
        {
            cout <<" object is dead"
        }
};

void main()
{
    clrscr();
    sample S;
    cout<<" Main terminated "<<endl;
    getch();
}
```

Unit 3

Inheritance (Extending Classes)

Objectives:

In this Chapter you will learn about:

- Inheritance -Extending Classes
- Concept of inheritance
- Base Class, Derived Class
- Types of Inheritance



INHERITANCE -EXTENDING CLASSES

Inheritance:-

Inheritance is the process of creating new classes called derived classes from existing ones. i.e base classes.

Different form of Inheritance:

1. Single Inheritance
2. Multiple Inheritance
3. Hierarchical Inheritance
4. Multilevel Inheritance
5. Hybrid Inheritance.

Derived and Base Class:- A derived class extends its features by inheriting the properties of another class, called base class and adding features of its own.

The general syntax of derived class is :-

Class derived _class_name : visibility mode base class name

Visibility of Class Members

Base class visibility	Private derivation	Protected derivation	Public derivation
Private	Not inherited	Not inherited	Not inherited
Protected	Private	Protected	Protected
Public	Private	Protected	public

1. Single Level Inheritance:-

Single inheritance is the process of creating new class (derived class) from an base class .i.e it creates a copy of base class and can add refinements of its own .

Public Derivation:-

Example:-

```
#include <iostream.h>
#include<conio.h>
class item
{
    int x;
    public:
        int y;
        void getdata();
        int getvalue(void);
        void display(void);
};

class newitem : public item
{
    int z;
    public:
        void mul(void);
        void show(void);
};
```

```
void item :: getdata(void)
{
    x=9;
    y=17;
}

int item:: getvalue(void)
{
    return x;
}

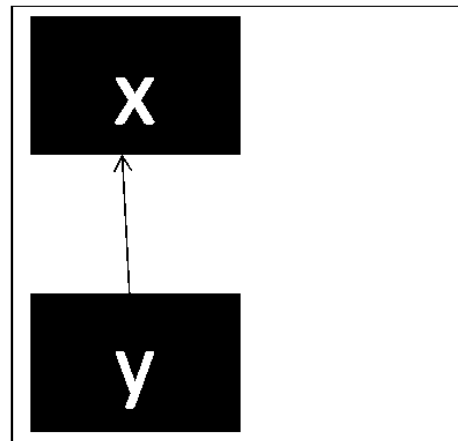
void item:: display()
{
    cout<<"x="<<x<<endl;
}

void new item :: mul()
{
    z=y*getvalue();
}

void newitem:: show()
{
    cout<<"x="<<getvalue()<<endl;
    cout<<"y="<<y<<endl;
    cout<<"z=" <<z<<endl;
}

void main()
{
    clrscr();
    newitem n;
    n.getdata();
    n.mul();
    n.display();
    n.show();
    n.y=45;
    n.mul();
    n.show();
    getch();
}
```

}



2. Multilevel Inheritance:-

Derivation of class from another derived class is called inheritance i.e when one derived class inherits from a class that itself inherits from another class is called multilevel inheritance.

Syntax:

```
Class X {.....};           // Base Class
Class Y: Public X;           // Y derived from X
Class Z:Public Y;            // Z derived from Y
```

```
#include<iostream.h>
#include<conio.h>
const int size=50;
class person
{
    char name[size];
    int age;
    char sex;
    public:
        void getdata()
        {
            cout<<" name=";
            cin>>name;
            cout<<"Age";
            cin>>age;
```

```
        cout<<"sex=";<<endl;
        cin>>sex;
    }
    void display()
    {
        cout<<"name =" << name <<endl;
        cout<<"Age="<<Age<<endl;
        cout<<"sex="<<sex<<endl;
    }
};
```

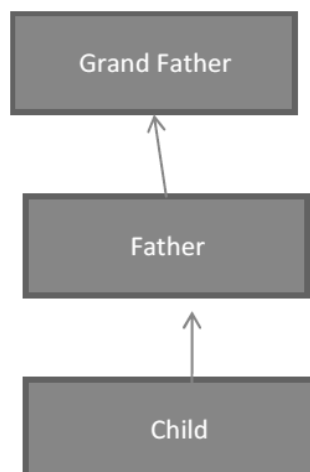
```
class student:Public Person
{
    private:
        int rollno;
        char branch[50];
    public:
        getdata()
        {
            Person::getdata();
            cout<<"Rollno=";
            cin>>rollno;
            cout<<"Branch";
            cin>>branch;
        }
};
```

```
void display()
{
    person:: display();
    cout<<"Rollno ="<<rollno<<endl;
    cout<<"Branch="<<branch<<endl;
}
};
```

```
class Exam : public Student
{
    protected:
        int marks1;
        int marks2;
    public:
        void getdata()
```

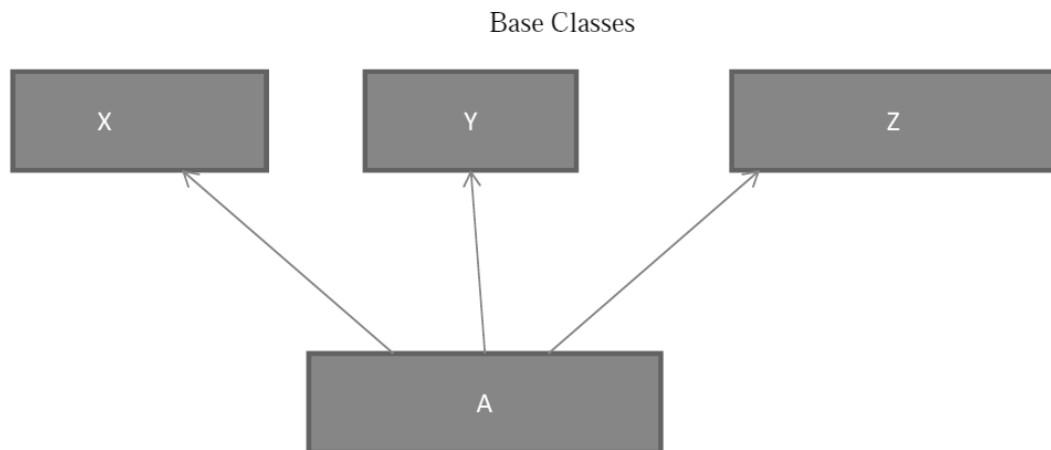
```
{
    Student:: getdata();
    cout<<"Marks 1=";
    cin>>marks1;
    cout<<"Marks2=";
    cin>>marks2;
}
void display()
{
    Student:: display()
    cout<<"Marks 1="<<marks1<<endl;
    cout<<"Marks2="<<marks2<<endl;
    cout<<"Total Marks="<<Totalmarks();
}
int Totalmarks()
{
    return marks1+marks2;
}
};

void main()
{
    clrscr();
    Exam e;
    e.getdata();
    e.display();
}
```



3. Multiple Inheritance:

A class can be derived from more than one base class i.e a class can inherit the properties of two or more than two classes .This is called multiple inheritance.

**Syntax:-**

class derived class : Visibility BaseClass1, Visibility Base Class2,E.g.

```
class X
{
    -----
    -----
};

class y
{
    -----
    -----
};

class Z
{
    -----
    -----
};

class A : Public X, Public Y, Public Z
```

Program: To demonstrate multiple inheritance

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

```
{
    protected:
        int a;
    public:
        void getdata(int);
};

class Y
{
    protected :
        int b;
    public:
        void getdata1(int);
};

class Z: public X, public Y
{
    public:
        void display(void );
};

void X :: getdata(int m)
{
    a=m;
}

void Y:: getdata1(int n)
{
    b=n;
}

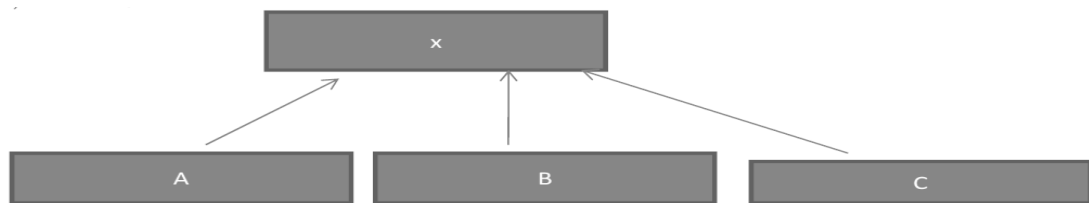
void Z::display(void)
{
    cout<<"a="<<a<<endl;
    cout<<"b="<<b<<endl;
    cout<<"a+b="<<a+b<<endl;
    cout<<"a/b="<<a/b<<endl;
}

void main()
{
```

```
clrscr();
Z d;
d.getdata(30);
d.getdata1(15);
d.display();
getch();
}
```

4. Hierarchical Inheritance:

When many derived classes are inherited from single base class it is called hierarchical Inheritance .



Syntax:-

```
class A: public X
class B : public X
class C: public X
```

Example:

```
# include<iostream.h>
#include< conio.h>
class personal
{
    private:
        int rollno;
        char name[15];
        char sex;
    public:
        void getdata();
        void putdata();
};
```

```
class physical : public personal
{
    private:
        float height ;
```

```
        float weight ;
    public:
        void getdata();
        void putdata();
};

class academic : public personal
{
    private:
        char trade[20];
        int semes;
    public:
        void getdata();
        void putdata();
};

void personal :: getdata()
{
    cout<<" Students's name :"; cin>>name;
    cout<<"Rollno:"; cin>> rollno;
    cout<<"Sex:"; cin>>sex;
}

void personal:: putdata()
{
    cout<<"Student's name :"<<name <<endl;
    cout<<"Rollno:"<<rollno<<endl;
    cout<<"Sex:"<<sex<<endl;
}

void physical :: getdata()
{
    personal :: getdata();
    cout <<"Height :"; cin>>height;
    cout<<"Weight:"; cin>> weight;
}

void physical :: putdata()
{
    personal :: putdata();
    cout<<"Height :"<<height << endl;
    cout<<"Weiight:"<<weight<<endl; }
    void academic :: getdata()
{
```

```
    personal :: getdata();
    cout<<" Trade :";
    cin>> trade;
    cout<<"Semester:";
    cin>>semes; }
void academic :: putdata()
{
    personal :: putdata();
    cout<<" Trade :"<< trade <<endl;
    cout<<"Semester :" << semes <<endl;
}
void main()
{
    class physical obj1;
    class academic obj2;
    clrscr();
    cout<<" ****enter the physical informatioon ****"<<endl;
    obj1.getdata();
    cout<<" ***Enter the academic information ****"<<endl;
    obj2.getdata();
    cout<<endl<< "***** The entered data is *****"<<endl;
    cout<<endl<<"physical fitness "<<endl;
    obj1.putdata();
    cout<<endl<<"Academic Fitness "<<endl;
    obj2.putdata();
    getch();
}
```

DATA FILE HANDLING IN C++

Key Facts:

- **Text file:** A text file stores information in readable and printable form. Each line of text is terminated with an EOL (End of Line) character.
- **Binary file:** A binary file contains information in the non-readable form i.e. in the same format in which it is held in memory.
- **Stream:** A stream is a general term used to name flow of data. Different streams are used to represent different kinds of data flow.

There are three file I/O classes used for file read / write operations.

- **ifstream** - can be used for read operations.
- **fstream** - can be used for write operations.
- **fstream** - can be used for both read & write operations.
- **fstream.h:**
 - This header includes the definitions for the stream classes ifstream, ofstream and fstream. In C++ file input output facilities implemented through fstream.h header file. It contain predefines set of operation for handling file related input and output fstream.h class ties a file to the program for input and output operation.
- **A file can be opened using:**
 - By the constructor of the stream. This method is preferred when single file is used with the stream. (only for input / only for output)
 - By the open() function of the stream.

File modes:

- **ios::out** - It creates file in output mode and allows writing into the file.
- **ios::in** - It creates file in input mode and permit reading from the file.
- **ios::app** - To retain the previous contents of the file and to append to the end of existing file.
- **ios::ate** - To place the file pointer at the end of the file, but you can write data any where in the file.
- **ios::trunc** - It truncates the existing file (empties the file).
- **ios::nocreate** - If file does not exist this file mode ensures that no file is created and open() fails.
- **ios::noreplace** - If file does not exist, a new file gets created but if the file already exists, the open() fails.
- **ios::binary** – Opens a file in binary.

- **eof():** This function determines the end-of-file by returning true for end of file otherwise returning false.
- **open():** If you want to manage multiple file with same stream use open().
Stream_object.open(“Filename”,(Filemode));
e.g., fstream fio;

- **fio.open(“book.dat”, ios::out | ios::in | ios::binary);**

- **close():** This function terminates the connection between the file and stream associated with it.
Stream_object.close();

- **read():** The read() function reads a fixed number of bytes from the specified stream and puts them in the buffer.

Stream_object.read((char *)& Object, sizeof(Object));

- **write():** The write() function writes fixed number of bytes from a specific memory location to the specified stream.

Stream_object.write((char *)& Object, sizeof(Object));

get pointer: A get pointer indicates the position in the file at which the next input is to occur.

put pointer: It indicates the position in the file at which the next output is to be placed.

seekg(): It places the get pointer to the specified position in a stream.

seekp(): It places the put pointer to the specified position in a stream.

tellg(): This function returns the current position of the get pointer in a stream.

tellp(): This function returns the current position of the put pointer in a stream.

Steps To Process A File

- Determine the type of link required.
- Declare a stream for the desired types of link.
- Attach the desired file to the declared stream.
- Process the file.
- Close the file link with stream.

General program structure used for operating a Text File

1. Write a function in a C++ to count the number of lowercase alphabets present in a text file “BOOK.txt”.

```
int countalpha()
{
    ifstream Fin("BOOK.txt");
    char ch;
    int count=0;
    while(!Fin.eof())
    {
        Fin.get(ch);
        if (islower(ch))
```

```
        count++;
    }
    Fin.close();
    return count;
}
```

2. Write a function in C++ to count and display the number of lines starting with alphabet 'A' present in a text file "LINES.TXT".

```
void counter( )
{
    char Aline[80];
    int Count=0;
    ifstream Fin ("LINES.TXT");
    while(!fin.eof())
    {
        Fin.getline(Aline,80, '\n')
        if (Aline[0]== 'A')
            Count++;
    }
    cout<<Count<<endl;
    Fin.close( );
}
```

3. Write a function COUNT_DO() in C++ to count the presence of a word 'do' in a text file "MEMO.TXT".

```
void COUNT_DO ( )
{
    fstream f;
    Clrscr();
    f.open ("MEMO.TXT", ios :: in);
    char are [80];
    char ch;
    int i = 0, sum = 0, n = 0 ;
    While (f)
    {
        f. get (ch);
        are [i] = ch;
        i++;
    }
}
```

```
        if (strcpy (ch, "do"))
        {
            i--;
            sum = sum + i;
            n++;
        }
    }
    cout<<"The total no. of the do is:"<<n;
}
```

4. Write a function in C++ to count the number of vowels present in a text file "STORY.TXT".

Ans: void vowelcount()

```
{
    ifstream file("STORY.TXT");
    int n=0;
    while(file.get(ch))
    {
        if(ch=='a' || ch=='A' || ch=='e' || ch=='E' || ch=='i' || ch=='I' || ch=='o' ||
        ch=='O' || ch=='u' || ch=='U')
            n++;
    }
    cout<<"\n Total no. of vowels are"<<n;
    file.close();
}
```

5. Write a function in C++ to count the number of lines present in a text file "STORY.TXT".

Ans : void CountLine()

```
{
    ifstream FIL("STORY.TXT");
    int LINES=0;
    char STR[80];
    while (FIL.getline(STR,80))
        LINES++;
    cout<<"No. of Lines:"<<LINES<<endl;
    FIL.close();
}
```

6. Write a function in C++ to count the word "this" /"THIS" present in Text file

"Diary.TXT"

Ans : void wordcount()

```
Fstream file;
file.open("Diary.TXT",ios::in);
char word[20];
int count=0;
while (!file.eof( ))
{
    file>>word;
    if(!strcmp(word,"this")||strcmp(word,"THIS")||!strcmp(word,"THIS"))
        count++;
}
cout<<" Number of this/This/THIS="<<count<<endl;
file. Close( );
}
```

7. Write a function in C++ to read the content of a text file "PLACES.TXT" and display all those lines on screen, which are either starting with 'P' or starting with 'S'.

```
Ans: void DispPorS ( )
{
    ifstream File ("PLACES.TXT");
    char STR[80];
    while(File.getline(STR,80))
    {
        if(STR[0]=='P' || STR[0]=='S')
            cout<<STR<<endl;
    }
    File.close();    //Ignore
```

General program structure used for operating a Binary File:

Q1. Write a function in C++ to add more new objects at the bottom of a binary file STUDENT.DAT", assuming the binary file is containing the objects of the following class.

```
class STUD
{
    int Rno;
    char Name[20];
    public:
        void Enter(){cin>>Rno;gets(Name);}
```

```
void Display(){cout<<Rno<<Name<<endl;}  
};
```

Ans: void Addnew()

```
{  
    ofstream FIL;  
    FIL.open("STUDENT.DAT", ios::out | ios::binary | ios::app);  
    STUD S;  
    char CH = 'y';  
    while(CH == 'Y' || CH == 'y')  
    {  
        S.Enter();  
        FIL.write((char*)&S, sizeof(S));  
        cout<<"More(Y/N)?">>CH;  
    }  
    FIL.close();  
}
```

Q2. Write a function in C++ to read & display the details of all the members whose membership type is 'L' or 'M' from a binary file "CLUB.DAT".

Assume the binary file contains object of class CLUB.

```
class CLUB  
{  
    int mno;  
    char mname[20];  
    char type;  
    public:  
        void register ( );  
        void dis();  
  
    char whattype( )  
    {  
        return type;  
    }  
};
```

Ans : void show()

```
{  
    ifstream file;  
    CLUB c1;  
    char s;  
    file.open("CLUB.DAT", ios::in | ios::binary);
```

```
        while( file.read((char*)&c1, sizeof (c1)))
        {
            s = c1.whattype( );
            if((s=='L') || (s=='M'))
                c1.dis( );
        }
        File.close();
    }
```

Q3. Given a binary file PHONE.DAT containing records of the following structure

```
type.class Phonlist
{
    char add;
    char Name[20];
    char area[5];
    public:
    void register();
    void show();
    int check(char ac[ ])
    {
        return strcmp(area,ac)
    }
};
```

Write a function TRANSFER() in C++, that would copy all records which are having area as "DEL" from PHONE.DAT to PHONBACK.DAT.

Ans:

```
void transfer( )
{
    ifstream Fin;
    ofstream Fout;
    Phonlist ph;
    Fin.open("PHONE.DAT", ios::in | ios::binary);
    Fout.open("PHONBACK.DAT", ios::out | ios:: binary);
    while(Fin.read((char*)&ph, sizeof(ph)))
    {
        if(ph.check("DEL") == 0)
```

```
        Fout.write((char*)&ph, sizeof(ph));
    }
    Fin.close();
    Fout.close();
}
```

4. Given a binary file STUDENT.DAT, containing records of the following class

Student type

```
class Student
{
    char S_Admno[10]; //Admission number of student
    char S_Name[30]; //Name of student
    int Percentage; //Marks Percentage of student
public:
    void EnterData()
    {
        gets(S_Admno);gets(S_Name);cin>>Percentage;
    }
    void DisplayData()
    {
        cout<<setw(12)<<S_Admno;
        cout<<setw(32)<<S_Name;
        cout<<setw(3)<<Percentage<<endl;
    }
    int ReturnPercentage()
    {
        return Percentage;
    }
};
```

Write a function in C++, that would read contents of file STUDENT.DAT and display the details of those Students whose Percentage is above

Ans :- void Distinction()

```
{
    Student S;
    fstream Fin;
    Fin.open("STUDENT.DAT", ios::binary|ios::in);
    while(Fin.read((char*)&S, sizeof(Student))
    if (S.ReturnPercentage()>75)
```

```
        S.DisplayData( );
        Fin.close();
    }
```

Questions based on file pointer

1. Observe the program segment given below carefully and fill the blanks marked as Line 1 and Line 2 using fstream functions for performing the required task. 1

```
#include <fstream.h>
class Library
{
    long Ano; //Ano – Accession Number of the Book
    char Title[20]; //Title – Title of the Book
    int Qty; //Qty – Number of Books in Library
public:
    void Enter(int); //Function to enter the content
    void Display(); //Function of display the content
    void Buy(int Tqty)
    {
        Qty+=Tqty;
    } //Function to increment in Qty
    long GetAno() {return Ano;}
};

void BuyBook (long BAno, int BQty)
//BAno ® Ano of the book purchased
//BQty ® Number of books purchased
{
    fstream File;
    File. open (“STOCK.DAT”, ios: : binary|ios: : in|ios: : out);
    int Position=-1;
    Librery L;
    while (Position == -1 && File. read ((char*) &L, sizeof (L)))
    if (L. GetAno() ==BAno)
    {
        L. Buy (BQty); //To update the number of Books
        Positions=File. tellg()-sizeof (L);
        //Line 1: To place the file pointer to the required position.
        _____;
```

```
//Line 2: To write the object L on to the binary file
_____;
}
if (Position==1)
cout<<"No updation done as required Ano not found...";
File. Close();
}
```

Ans:

```
File. seekp (position, ios :: beg);      // Line-1
File. write ((char *) & L, sizeof (L));  // Line-2
```

2. Observe the program segment given below carefully and fill the blanks marked as Statement 1 and Statement 2 using seekg() and tellg() functions for performing the required task.

```
#include <fstream.h>
class Employee
{
    int Eno;char Ename[20];
    public:
        _____ //Function to count the total number of records
        int Countrec();
};
int Item::Countrec()
{
    fstream File;
    File.open("EMP.DAT",ios::binary|ios::in);
    _____ //Statement 1
    int Bytes = _____ //Statement 2
    int Count = Bytes / sizeof(Item);
    File.close();
    return Count;
}
```

Ans:

```
File.seekg(0,ios::end);    //Statement 1
File.tellg();              //Statement 2
```

Unit 4

Pointers

Objectives:

In this Chapter you will learn about:

- Pointers
- Concept of a Pointer
- Declaration of Pointers
- Initialization of Pointers



POINTERS:

- Pointer is a variable that holds a memory address of another variable of same type.
- It supports dynamic allocation routines.
- It can improve the efficiency of certain routines.

C++Memory Map :

- **Program Code** : It holds the compiled code of the program.
- **Global Variables** : They remain in the memory as long as program continues.
- **Stack** : It is used for holding return addresses at function calls, arguments passed to the functions, local variables for functions. It also stores the current state of the CPU.
- **Heap** : It is a region of free memory from which chunks of memory are allocated via DMA functions.

Static Memory Allocation : The amount of memory to be allocated is known in advance and it is allocated during compilation, it is referred to as Static Memory Allocation.

e.g. `int a;` `// This will allocate 2 bytes for a during compilation.`

Dynamic Memory Allocation : The amount of memory to be allocated is not known beforehand rather it is required to be allocated as and when required during runtime, it is referred to as dynamic memory allocation.

C++ offers two operators for DMA – **new** and **delete**.

e.g., `int x = new int; float y = new float;` // dynamic allocation

`delete x; delete y;` //dynamic deallocation

Free Store : It is a pool of unallocated heap memory given to a program that is used by the program for dynamic memory allocation during execution.

Declaration and Initialization of Pointers :

Syntax :

Datatype *variable_name;

e.g., `int *p; float *p1; char *c;`

Two special unary operator * and & are used with pointers. The & is a unary operator that returns the memory address of its operand.

e.g., `int a = 10; int *p; p = &a;`

Pointer arithmetic:

Two arithmetic operations, addition and subtraction, may be performed on pointers. When you add 1 to a pointer, you are actually adding the size of whatever the pointer is pointing at. That is, each time a pointer is incremented by 1, it points to the memory location of the next element of its base type.

e.g. `int *p; p++;`

If current address of p is 1000, then p++ statement will increase p to 1002, not 1001.

Adding 1 to a pointer actually adds the size of pointer's base type.

Base address : A pointer holds the address of the very first byte of the memory location where it is pointing to. The address of the first byte is known as BASE ADDRESS.

Dynamic Allocation Operators :

C++ dynamic allocation allocate memory from the free store/heap/pool, the pool of unallocated heap memory provided to the program. C++ defines two unary

operators new and delete that perform the task of allocating and freeing memory during runtime.

Creating Dynamic Array :

Syntax : pointer-variable = new data-type [size];

e.g. `int * array = new int[10];` `array[0]` will refer to the first element of array, `array[1]` will refer to the second element. No initializes can be specified for arrays.

All array sizes must be supplied when new is used for array creation.

Two dimensional array :

```
int *arr, r, c;
```

```
r = 5; c = 5;
```

```
arr = new int [r * c];
```

Now to read the element of array, you can use the following loops :

```
For (int i = 0; i < r; i++)
```

```
{
```

```
    cout << "\n Enter element in row " << i + 1 << " : ";
```

```
    For (int j=0; j < c; j++)
```

```
        cin >> arr [ i * c + j];
```

```
}
```

Memory released with delete as below:

Syntax for simple variable : delete pointer-variable;

eg. delete p;

For array :

delete [size] pointer variable;

e.g. delete [] arr;

Pointers and Arrays :

C++ treats the name of an array as constant pointer which contains base address i.e address of first location of array. Therefore Pointer variables are efficiently used with arrays for declaration as well as accessing elements of arrays, because array is continuous block of same memory locations and therefore pointer arithmetic help to traverse in the array easily.

```
void main()
```

```
{
```

```
    int *m;
```

```
    int marks[10] = { 50,60,70,80,90,80,80,85,75,95};
```

```
    m = marks; // address of first location of array or we can write it as m=&marks[0]
```

```

for(int i=0;i<10;i++)
cout<< *m++;
// or
m = marks; // address of first location of array or we can write it as m=&marks[0]
for(int i=0;i<10;i++)
cout<< *(m+i);
}

```

Array of Pointers :

To declare an array holding 10 int pointers –

```
int * ip[10];
```

That would be allocated for 10 pointers that can point to integers. Now each of the pointers, the elements of pointer array, may be initialized. To assign the address of an integer variable phy to the forth element of the pointer array, we have to write `ip[3] = & phy;` Now with `*ip[3]`, we can find the value of phy. `int *ip[5];`

Index	0	1	2	3	4
address	1000	1002	1004	1006	1008
int	a = 12,	b = 23,	c = 34,	d = 45,	e = 56;

Variable	a	b	c	d	e
Value	12	23	34	45	56
address	1050	1065	2001	2450	2725

```
ip[0] = &a; ip[1] = &b; ip[2] = &c; ip[3] = &d; ip[4] = &e;
```

Index	ip[0]	ip[1]	ip[2]	ip[3]	ip[4]
Array ip value	1050	1065	2001	2450	2725
address	1000	1002	1004	1006	1008

ip is now a pointer pointing to its first element of ip. Thus ip is equal to address of ip[0], i.e. 1000

*ip (the value of ip[0]) = 1050
* (* ip) = the value of *ip = 12
* * (ip+3) = * * (1006) = * (2450) = 45

Pointers and Strings :

Pointer is very useful to handle the character array also. E.g :

```
void main()
{
    char str[] = "computer";
    char *cp;
    cp=str;
    cout<<str ;           //display string
    cout<<cp;             // display string
    for (cp =str; *cp != '\0'; cp++) // display character by character by character
        cout << "--"<<*cp;
                                // arithmetic

    str++;      // not allowed because str is an array and array name is constant pointer
    cp++;       // allowed because pointer is a variable
    cout<<cp;
}
```

Output :

```
Computer
Computer
--c--o--m--p--u--t--e--r
Computer
```

An array of char pointers is very useful for storing strings in memory.

```
Char *subject[] = { "Chemistry", "Phycics", "Maths", "CS", "English" };
```

In the above given declaration subject[] is an array of char pointers whose element pointers contain base addresses of respective names. That is, the element pointer subject[0] stores the base address of string "Chemistry", the element pointer subject[1] stores the above address of string "Physics" and so forth.

An array of pointers makes more efficient use of available memory by consuming lesser number of bytes to store the string.

An array of pointers makes the manipulation of the strings much easier. One can easily exchange the positions of strings in the array using pointers without actually touching their memory locations.

Pointers and CONST:

A constant pointer means that the pointer in consideration will always point to the same address. Its address cannot be modified.

A pointer to a constant refers to a pointer which is pointing to a symbolic constant. Look the following example:

```
int m = 20;           // integer m declaration
int *p = &m;          // pointer p to an integer m
++ (*p);              // ok : increments int pointer p
int * const c = &n;    // a const pointer c to an integer n
++ (* c);             // ok : increments int pointer c i.e. its contents
++ c;                // wrong : pointer c is const – address can't be modified
const int cn = 10;    // a const integer cn
const int *pc = &cn;  // a pointer to a const int
++ (* pc);            // wrong : int * pc is const – contents can't be modified
++ pc;               // ok : increments pointer pc
const int * const cc = *k; // a const pointer to a const integer
++ (* cc);           // wrong : int *cc is const
++ cc;              // wrong : pointer cc is const
```

Pointers and Functions:

A function may be invoked in one of two ways:

- Call by value
- Call by reference

The second method call by reference can be used in two ways:

- By passing the references
- By passing the pointers

Reference is an alias name for a variable.

For ex : int m =23;

int &n = m;

int *p;

p = &m;

Then the value of m i.e. 23 is printed in the following ways :

cout <<m; // using variable name

```
cout << n;           // using reference name
cout << *p;          // using the pointer
```

Invoking Function by Passing the References:

When parameters are passed to the functions by reference, then the formal parameters become references (or aliases) to the actual parameters to the calling function. That means the called function does not create its own copy of original values, rather, it refers to the original values by different names i.e. their references.

For example the program of swapping two variables with reference method:

```
#include<iostream.h>
void main()
{
    void swap(int &, int &);
    int a = 5, b = 6;
    cout << "\n Value of a : " << a << " and b : " << b;
    swap(a, b);
    cout << "\n After swapping value of a : " << a << "and b : " << b;
}

void swap(int &m, int &n)
{
    int temp; temp = m;
    m = n;
    n = temp;
}
```

Output :

Value of a : 5 and b : 6

After swapping value of a : 6 and b : 5

Invoking Function by Passing the Pointers:

When the pointers are passed to the function, the addresses of actual arguments in the calling function are copied in to formal arguments of the called function.

That means using the formal arguments (the addresses of original values) in the called function, we can make changing the actual arguments of the calling function.

For example the program of swapping two variables with Pointers:

```
#include<iostream.h>
void main()
{
    void swap(int *m, int *n);
    int a = 5, b = 6;
    cout << "\n Value of a :" << a << " and b :" << b;
    swap(&a, &b);
    cout << "\n After swapping value of a :" << a << "and b :" << b;
}

void swap(int *m, int *n)
{
    int temp;
    temp = *m;
    *m = *n;
    *n = temp;
}
```

Input:

Value of a : 5 and b : 6

After swapping value of a : 6 and b : 5

Function returning Pointers:

The way a function can returns an int, an float, it also returns a pointer. The general form of prototype of a function returning a pointer would be Type * function-name (argument list);

```
#include <iostream.h> int
*min(int &, int &);
void main()
{
    int a, b, *c;
    cout << "\nEnter a :"; cin >> a;
    cout << "\nEnter b :"; cin >> b;
    c = min(a, b);
    cout << "\n The minimum no is :" << *c;
}
```

```
int *min(int &x, int &y)
{
    if (x < y )
        return (&x);
    else
        return (&y)
}
```

Dynamic structures :

The new operator can be used to create dynamic structures also i.e. the structures for which the memory is dynamically allocated.

```
struct-pointer = new struct-type;
student *stu;
stu = new Student;
```

A dynamic structure can be released using the de-allocation operator delete as shown below:

```
delete stu;
```

Objects as Function arguments:

Objects are passed to functions in the same way as any other type of variable is passed. When it is said that objects are passed through the call-by-value, it means that the called function creates a copy of the passed object.

A called function receiving an object as a parameter creates the copy of the object without invoking the constructor. However, when the function terminates, it destroys this copy of the object by invoking its destructor function.

If you want the called function to work with the original object so that there is no need to create and destroy the copy of it, you may pass the reference of the object. Then the called function refers to the original object using its reference or alias. Also the object pointers are declared by placing in front of an object pointer's name.

```
Classname * object-pointer;
```

e.g. Student *stu;

The member of a class is accessed by the arrow operator (->) in object pointer method. e.g :

```
#include<iostream.h>
class Point
{
    int x, y;
    public :
        Point()
        {
            x = y = 0;
        }
        void getPoint(int x1, int y1)
        {
            x = x1; y = y1;
        }
        void putPoint()
        {
            cout << "\n
            Point : (“ << x<< “, “<< y<< “)”;
        }
};

void main()
{
    Point p1, *p2;
    cout << “\n Set point at 3, 5 with object”;
    p1.getPoint(3,5);
    cout << “\n The point is :”;
    p1.putPoint();
    p2 = &p1;
    cout << “\n Print point using object pointer :”;
    p2->putPoint();
    cout << “\n Set point at 6,7 with object pointer”;
    p2->getPoint(6,7);
    cout<< “\n The point is :”;
    p2->putPoint();
    cout << “\n Print point using object :”;
    p1.getPoint();
}
```

If you make an object pointer point to the first object in an array of objects, incrementing the pointer would make it point to the next object in sequence.

student stud[5], *sp;

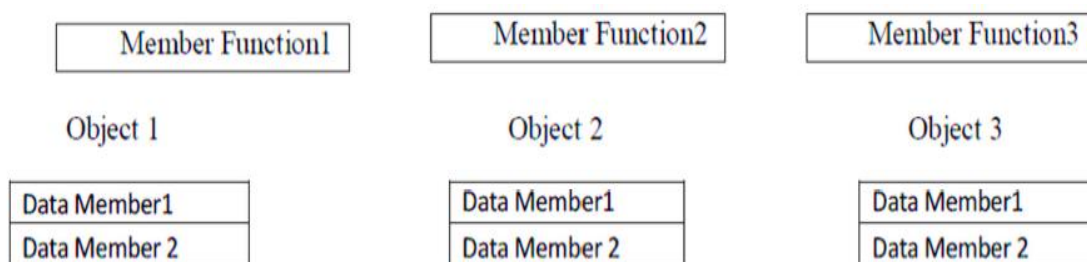
```
sp = stud;           // sp points to the first element (stud[0]) of stud
sp++;               // sp points to the second element (stud[1]) of stud
sp += 2;            // sp points to the fourth element (stud[3]) of stud
sp--;               // sp points to the third element (stud[2]) of stud
```

You can even make a pointer point to a data member of an object. Two points should be considered:

- A Pointer can point to only public members of a class.
- The data type of the pointer must be the same as that of the data member it points to.

this Pointer:

In class, the member functions are created and placed in the memory space only once. That is only one copy of functions is used by all objects of the class. Therefore if only one instance of a member function exists, how does it come to know which object's data member is to be manipulated?



For the above figure, if Member Function2 is capable of changing the value of Data Member3 and we want to change the value of Data Member3 of Object3. How would the Member Function2 come to know which Object's Data Member3 is to be changed?

To overcome this problem this pointer is used.

When a member function is called, it is automatically passed an implicit argument that is a pointer to the object that invoked the function. This pointer is called This. That is if object3 is invoking member function2, then an implicit argument is passed to member function2 that points to object3 i.e. this pointer now points to object3. The friend functions are not members of a class and, therefore, are not passed a this pointer. The static member functions do not have a this pointer.

Unit 5

Data Structure

Objectives:

In this Chapter you will learn about:

- Arrays
- Searching techniques
- Sorting Algorithms
- Stacks, Queues And Linked List
- Exercise



Data Structure:

In Computer Science, a data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently. Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks. For example, Stacks are used in function call during execution of a program, while B-trees are particularly well-suited for implementation of databases. The data structure can be classified into following two types:

Simple Data Structure: These data structures are normally built from primitive data types like integers, floats, characters. For example arrays and structure.

Compound Data Structure: simple data structures can be combined in various ways to form more complex structure called compound structures. Linked Lists, Stack, Queues and Trees are examples of compound data structure.

Array:

Array is a collection of elements of similar data type stored in continuous memory location under a single name.

Declaration of array:

Syntax:-

Datatype array name[total no of elements];

Eg. int arr[10];

The statement written above will create an array namely arr having 10 integer elements. To differentiate array elements index value is used. Index value is an integer value that starts from 0 and moves upto N-1, where N is total number of elements in the array. For example if we have an array of 10 elements then the index value will be from 0 to 9

Initialization of the array

Datatype array name[no of elements]={value1,value2,.....valueN};

Eg int arr[5]={10,20,30,40,50};

We can skip the total no elements and can initialize the array

Eg int arr[]={10,20,30};

Or

int arr[]={10,20,30,40,50,60};

Program to enter and display elements in an array

```
void main()
{
    int arr[10],i;
    cout<<"enter the elements in array"<<endl;
    for(i=0;i<10;i++)
    {
        cin>>arr[i];
    }
    cout<<"The Array elements are"<<endl;
    for(i=0;i<10;i++)
    {
        cout<<arr[i]<<endl;
    }
}
```

Searching methods in array

Linear Search: In this method each element of the array is compared with the number to be searched in linear order (from first to last). And where the number is matched the position is displayed.

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

```
int lsearch(int[],int,int);
int a[50],item,n,index;
clrscr();
cout<<"\n Enter size of array";
cin>>n;
cout<<"\n Enter array elements";
for(int i=0;i<n;i++)
{
    cin>>a[i];
}
cout<<"Enter the item to be searched";
cin>>item;
index=lsearch(a,n,item);
if(index== -1)
    cout<<"\n Element not found";
else
    cout<<"\n Element found at position "<<index+1;
getch();
}
```

```
int lsearch(int a[],int size,int item)
{
    int found=0;
    for(int i=0;i<size;i++)
    {
        if(a[i]==item)
        {
            return i;
            found=1;
            break;
        }
    }
    if(found==0)
        return -1;
}
```

Binary Search Method:

Binary search algorithm is applicable for already sorted array only. In this algorithm, to search for the given item from the sorted array (in ascending order), the item is compared with the middle element of the array. If the middle element is equal to the item then index of the middle element is returned, otherwise, if item is

less than the middle item then the item is present in first half segment of the array (i.e. between 0 to middle-1), so the next iteration will continue for first half only, if the item is larger than the middle element then the item is present in second half of the array (i.e. between middle+1 to size-1), so the next iteration will continue for second half segment of the array only. The same process continues until either the item is found (search successful) or the segment is reduced to the single element and still the item is not found (search unsuccessful).

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int bsearch(int[],int,int);
    int a[50], item, n, index;
    clrscr();
    cout<<"\n Enter total elements";
    cin>>n;
    cout<<"\n Enter array elements in sorted form:";
    for(int i=0;i<n;i++)
    {
        cin>>a[i];
    }
    cout<<"Enter the item to be searched";
    cin>>item;
    index=bsearch(a, n, item);
    if(index== -1)
        cout<<"\n Element not found";
    else
        cout<<"\n Element found at position "<<index+1;
    getch();
}

int bsearch(int a[], int size, int item)
{
    int beg, med, last;
    beg=0,found=0;
    last=size-1;
    int mid=(last+beg)/2;
    while(beg<=last)
    {
```

```
        mid=(beg+last)/2;
        if(item==a[mid])
        {
            return mid;
            found=1;
            break;
        }
        else if(item>a[mid])
            beg=mid+1;
        else
            last=mid-1;
    }

    If(found==0)
        return -1;
}
```

Sorting operation in the array:

Sorting means to arrange the array elements in Ascending order or in Descending order. There are various methods to do this but for the ease sake Bubble sort method is displayed here.

```
#include<iostream.h>
#include<conio.h>
void bubblesort (int[],int);
void main()
{
    int a[50],n;
    clrscr();
    cout<<"\nHow many elements do you want to create array with? ";
    cin>>n;
    cout<<"\nEnter array elements\n";
    for(int i=0;i<n;i++)
        cin>>a[i];
    bubblesort(a,n);
    cout<<"\n\nThe sorted array is as shown below\n";
    for(i=0;i<n;i++)
        cout<<a[i]<<"\n";
    getch();
}
```

```
void bubblesort(int a[],int n) //Function to perform bubble sort
```

```
{
    int temp;
    for(int i=0;i<n-1;i++)
    {
        for(int j=0;j<n-i-1;j++)
            if(a[j]>a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
    }
}
```

Exercise for One dimensional array:

Q1. Write a function in C++ which accepts an integer array and its size as arguments/parameters and reverses the array example : if the array is 1,2,3,4,5 then rearrange the array as 5,4,3,2,1

Ans :

```
void reverse(int arr[ ], int n)
{
    int temp;
    for(int i=0,j=n-1; i<=j; i++,j--)
    {
        temp= arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}
```

Q2. Write a function in C++ which accepts an integer array and its size as arguments/parameters and exchange the array in the given manner example : if the array is 1,2,3,4,5,6,7,8,9,10 then rearrange the array as 2,1,4,3,6,5,8,7,10,9

Ans :

```
void change(int arr[ ], int n)
{
    int temp;
```

```
for(int i=0; i<n; i=i+2)
{
    temp= arr[i];
    arr[i] = arr[i+1];
    arr[i+1] = temp;
}
}
```

Q3 Write a function in C++ to merge the contents of two sorted arrays A & B into third array C. Assuming array A is sorted in ascending order, B is sorted in descending order, the resultant array is required to be in ascending order.

Ans:

```
void MERGE(int A[ ], int B[ ], int C[ ], int M, int N, int &K)
{
    int I,J, K;
    for(I=0, J=N - 1, K=0; I<M && J>=0;)
    {
        if (A[I]<=B[J])
            C[K++]=A[I++];
        else
            C[K++]=B[J--];
    }
    for (int T=I;T<M;T++)
        C[K++]=A[T];
    for (T=J;T>=0;T--)
        C[K++]=B[T];
}
```

Two Dimensional Arrays:

In a two dimensional array the data is represented in form of rows and columns.

Declaration of two dimensional arrays

Datatype name_of_array[no of rows][no of columns]; eg

```
int arr[5][3];
```

In the example given above a two dimensional array is declared with the name arr where there are 5 rows and three columns i.e total 15 (5 x 3) elements

Program for entering and displaying data(in matrix form) in two dimensional array

```
void main()
{
    int arr[5][3],i,j;
    cout<<"Enter array elements "<<endl;
    for(i=0;i<5;i++)
    {
        for(j=0;j<3;j++)
        {
            cin>>arr[i][j];
        }
    }
    cout<<"the array elements are"<<endl;
    for(i=0;i<5;i++)
    {
        for(j=0;j<3;j++)
        {
            cout<<arr[i][j]<<"\t";
        }
        cout<<endl;
    }
    getch();
}
```

Exercise based on Two dimensional Array

Q1. Write a function in C++ that will accept a two D array and its row and column size as argument and find sum of rows and columns

Ans :

```
void rowcolsum(int A[ ][ ],int N, int M)
{
    for (int i=0;i<N;i++)
    {
        int SumR=0;
        for (int j=0;j<M;j++)
            SumR+=A[i][j];
        cout<<SumR<<endl;
    }
    for (int i=0;i<N;i++)
    {
```

```
        int SumC=0;
        for (int j=0;j<M;j++)
            SumC+=A[j][i];
        cout<<SumC<<endl;
    }
}
```

Q2. Write a function in C++ to find the sum of both left and right diagonal elements from a two dimensional array (matrix).

Ans :

```
void DiagSum(int A[ ][ ], int N)
{
    int SumD1=0,SumD2=0;
    for (int I=0;I<N;I++)
    {
        SumD1+=A[I][I];
        SumD2+=A[N-I-1][I];
    }
    cout<<"Sum of Diagonal 1:"<<SumD1<<endl;
    cout<<"Sum of Diagonal 2:"<<SumD2<<endl;
}
```

Address Calculation in Two Dimensional Array

Two dimensional array can be arranged in two manner

1. Row Major Order
2. Column Major Order

In Row Major Order the elements are shown Row Wise where as in Column Major Order the array elements are shown Column wise e,g

If we have following two dimensional array

1	2	3
4	5	6
7	8	9

The array elements in Row major Order will be

1,2,3,4,5,6,7,8,9

The array elements in Column Major Order will be

1,4,7,2,5,8,3,6,9

To find the address of a particular row and column the formula in Row Major Order is

$$\text{Address of } A[\text{row}][\text{column}] = B + w * (n(\text{row}) + \text{column})$$

Where

B= Base address of the array

w= Word size

n= total no of columns in the array

To find the address of a particular row and column the formula in Column Major Order is

$$\text{Address of } A[\text{row}][\text{column}] = B + w * (n(\text{Column}) + \text{row})$$

Where

B= Base address of the array

w= Word size

n= total no of rows in the array

Q1. An array x[30][10] is stored in the memory with each element requiring 4 bytes of storage. If the base address of x is 4500, find out memory locations of x[12][8] if the content is stored along the row.

Ans:

Here the array is stored in Row Major Order so

$$B = 4500$$

$$W = 4$$

$$N = 10$$

As per the formula

$$\text{Address of } A[\text{row}][\text{column}] = B + w * (n(\text{row}) + \text{column})$$

$$= 4500 + 4 * (10(12) + 8)$$

$$= 4500 + 4 * (128)$$

$$= 4500 + 512$$

$$= 5012$$

Q2. An array x[30][10] is stored in the memory with each element requiring 4 bytes of storage. If the base address of x is 4500, find out memory locations of x[12][8] if the content is stored along the row.

Ans:

Here the array is stored in Column Major Order so

$B=4500$

$W= 4$

$N= 30$

As per the formula

Address of $A[\text{row}][\text{column}] = B + w * (n(\text{Column}) + \text{row})$

99

$=4500+4*(30(8)+12)$

$=4500+4*(252)$

$=4500+1008$

$=5508$

Q 3. An array $P[20][30]$ is stored in the memory along the column with each of the element occupying 4 bytes, find out the Base Address of the array, if an element $P[2][20]$ is stored at the memory location 5000.

Ans :

Given, $W=4$, $N=20$, $M=30$, $\text{Loc}(P[2][20])=5000$

Column Major Formula:

$\text{Loc}(P[I][J]) = \text{Base}(P) + W * (N * J + I)$

$\text{Loc}(P[2][20]) = \text{Base}(P) + 4 * (20 * 20 + 2)$

$\text{Base}(P) = 5000 - 4 * (400 + 2)$

$= 5000 - 1608$

$= 3392$

STACKS, QUEUES AND LINKED LIST



Stack

In computer science, a stack is a Last in, First out (LIFO) data structure. It simply means that an element that is inserted at the end will be deleted first. To Manage a stack all the insertion and deletion takes place from one position called “top”.

One of the common uses of stack is in function call.

Operations on the Stack

There are two fundamental operations

1. Push
2. Pop

Push means to insert an element

Pop means to delete an element

Stack using array

```
#include<iostream.h>
const int size=5
class stack
{
    int a[size]; //array a can store maximum 5 item of type int of the stack
    int top; //top will point to the last item pushed onto the stack
public:
    stack() //constructor to create an empty stack, top=-1 indicate that no item is present in the
    {
        // array
        top=-1;
    }
    void push(int item)
    {
        if(top==size-1)
            cout<<"stack is full, given item cannot be added";
        else
            a[++top]=item; //increment top by 1 then item at new position of the top in the array a
    }
    int pop()
    {
        if (top== -1)
        {
            cout<<"Stack is empty ";
            return -1; //-1 indicates empty stack
        }
        else
            return a[top--]; //return the item present at the top of the stack then decrement top by 1
    }
};

void main()
{
    stack s1;
    s1.push(3);
    s1.push(5);
    cout<<s1.pop()<<endl;
    cout<<s1.pop()<<endl;
    cout<<s1.pop();
}
```

Queue

In computer science, a Queue is a First in, First out (FIFO) data structure. It simply means that an element that is inserted at the beginning will be deleted first. To Manage a queue all the insertion and deletion takes place from two different positions called “front” and “rear”. Every element is inserted from the rear position and deleted from the front position in the queue.



Queue using array

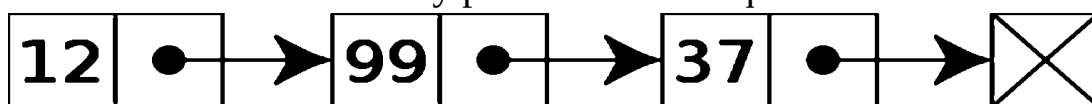
```
#include<iostream.h>
const int size=5
class queue
{
    int a[size]; //array a can store maximum 5 item of type int of the queue
    int front,rear;
public:
    queue() //constructor to create an empty queue
    {
        front=rear=-1;
    }
    void insert(int item)
    {
        If((front==0)&&(rear==size-1))
            cout<<"queue is full, given item cannot be added";
        else if((front==-1)||(rear==-1))
            front=rear=0;
        else
            rear++;
            a[rear]=item;
    }

    int deletion()
    {
        if (front== -1)
        {
            cout<<"queue is empty ";
            return -1;
        }
        else
            return a[front++];
    }
};
```

```
void main()
{
    queue q1;
    q1.insert(3);
    q1.insert(5);
    cout<<q1.deletion()<<endl;
    cout<<q1.deletion()<<endl;
}
```

Linked List

A linked list is a data structure consisting of a group of nodes which together represent a sequence. Under the simplest form, each node is composed of a data and a reference (in other words, a link) to the next node in the sequence; more complex variants add additional links. This structure allows for efficient insertion or removal of elements from any position in the sequence.



Here in the figure is an example of a linked list whose nodes contain two fields: an integer value and a link to the next node. The last node is linked to a terminator used to signify the end of the list. Linked lists are among the simplest and most common data structures. They can be used to implement several other common abstract data types, stacks, queues etc though it is not uncommon to implement the other data structures directly without using a list as the basis of implementation.

The principal benefit of a linked list over an array is that the list elements can easily be inserted or removed without reallocation or reorganization of the entire structure because the data items need not be stored contiguously in memory or on disk. Linked lists allow insertion and removal of nodes at any point in the list, and can do so with a constant number of operations if the link previous to the link being added or removed is maintained during list traversal. Linked list are dynamic structure where memory allocation takes place at run time.

Operation on a linked list

There are three basic operations on a linked list

1. Insertion
2. Deletion
3. Traversal



Inserting a node or element into Linked list:

Inserting an element into linked list contains 3 types .

1. Insertion at beginning of the Linked list
2. Insertion at the middle of the linked list
3. Insertion at the end of the linked list

1. Insertion at beginning of the Linked list :

An element can be inserted at the beginning of the Linked list by creating a new node at the beginning of the linked list and inserting an element into the node and assigning the address of the address of the next node or assigning the reference of the next node .

2. Insertion at the middle of the linked list:

An element can be inserted at the middle of the linked list. We need to know the data or element of the neighboring element to insert an element at middle.

3. Insertion at the End of the linked list :

An element can be inserted at the end by simply traversing the node to the end and creating a new node at the end . Also the address pointer of the new node must be assigned to NULL

Deleting a node from the Linked list.

Deleting an element is similar to pop operation in Stack and Queue. A node can be deleted in 3 ways similar to Insertion.

1. Deleting a Node from the beginning of the Linked List
2. Deleting a node from the middle of the Linked list.
3. Deleting a node from the end of the Linked List.

Program of linked list

```
#include<iostream.h>
class node
{
    public :
        int d;
        node *p;
};
class linked
{
    public :
        node *start;
    public :
        linked()
```

```
        {
            start=NULL;
        }
void insert_at_beg();
void insert_at_mid();
void insert_at_end();
int remove();
void display();
};
void linked :: insert_at_end()
{
    node *temp;
    if(start==NULL)
    {
        start=new node [1];
        start->p=NULL;
        cout<<"\n Enter Data to insert \n";
        cin>>start->d;
    }
    else
    {
        temp=start;
        while(temp->p!=NULL)
        temp=temp->p;
        temp->p= new node [1];
        temp=temp->p;
        cout<<"\n Enter Data to insert \n";
        cin>>temp->d;
        temp->p=NULL;
    }
}

void linked :: insert_at_mid()
{
    node *temp,*next;
    int x;
    temp=start;
    cout<<"\n Enter the Neighbour element \n";
    cin>>x;
    while(temp->d!=x)
    temp=temp->p;
```

```
    next=temp->p;
    temp->p=new node [1];
    temp=temp->p;
    temp->p=next;
    cout<<"\n Enter data to insert \n";
    cin>>temp->d;
}
```

```
void linked :: insert_at_beg()
{
    node *temp;
    temp=new node [1];
    temp->p=start;
    start=temp;
    cout<<"\n Enter element to insert \n";
    cin>>temp->d;
}
```

```
int linked :: remove()
{
    node *temp,*prev;
    int x;
    if(start==NULL)
    {
        cout<<"\n Underflow -- No element to Delete \n";
        return 0;
    }
    cout<<"\n Enter element to delete \n";
    cin>>x;
    if(start->d==x && start->p==NULL)
    {
        delete start;
        start=NULL;
        cout<<"\n Deletion successfull \n";
        return 0;
    }
    if(start->d==x)
    {
        temp=start->p;
        delete start;
        start=temp;
    }
}
```

```
        cout<<"\n Deletion Successfull \n";
        return 0;
    }
    temp=start;
    while(temp->d!=x)
    {
        prev=temp;
        temp=temp->p;
    }
    prev->p=temp->p;
    cout<<"\n Deletion Successfull \n";
    return x;
}
void linked :: display()
{
    node *temp;
    temp=start;
    cout<<"\n Linked List elements are \n";
    while(temp->p!=NULL)
    {
        cout<<""<<temp->d<<" ";
        temp=temp->p;
    }
    cout<<temp->d<<endl;
}
int main()
{
    int n=0,a;
    linked l;
    do
    {
        cout<<"\n ***** M E N U *****\n";
        cout<<"\n1.Insert at beginning\n2.Insert at middle\n3.Insert at
end\n4.Delete\n5.Display elements";
        cout<<"\n6.Exit\n";
        cout<<"Enter option \n";
        cin>>a;
        switch(a)
        {
            case 1:
                l.insert_at_beg();
```

```
        break;
    case 2:
        l.insert_at_mid();
        break;
    case 3:
        l.insert_at_end();
        break;
    case 4:
        l.remove();
        break;
    case 5:
        l.display();
        break;
    case 6:
        n=1;
        break;
    default :
        cout<<"\n Invalid Entry \n";
        break;
    }
}while(n!=1);
return 0;
}
```

Implementation of stacks using a linked list

The stack which is implemented using linked list is called linked stack or dynamic stack

```
#include<iostream.h>
#include<conio.h>
struct node
{
    int data;
    node * next;
};

class stack
{
    node *top;
public:
    stack()
    {
```

```
        top=NULL;
    }
    void stackpush();
    void stackpop();
    void displaystack();
};

void stack::stackpush()
{
    node *ptr;
    ptr=new node;
    cout<<"Enter the element to be pushed"<<endl;
    cin>>ptr->data;
    if(top==NULL)
        ptr->next=NULL;
    else
        ptr->next=top;
    top=ptr;
}

void stack ::stackpop()
{
    node *ptr;
    ptr=new node;
    ptr=top;
    cout<<"The popped element is "<<ptr->data;
    top=top->next;
    delete ptr;
}

void stack :: displaystack()
{
    node *ptr;
    ptr=new node;
    ptr=top;
    cout<<"The stack is "<<endl;
    while(ptr!=NULL)
    {
        cout<<ptr->data<<endl;
        ptr=ptr->next;
    }
}
```

```
}  
  
void main()  
{  
    clrscr();  
    char ans;  
    stack s1;  
    do  
    {  
        s1.stackpush();  
        cout<<"wish to continue "<<endl;  
        cin>>ans;  
    }while(ans=='y');  
    s1.displaystack();  
    cout<<"Press any key to pop an element"<<endl;  
    getch();  
    s1.stackpop();  
    getch();  
}
```

Implementation of queues using a linked list

The queue which is implemented using linked list is called linked queue or dynamic queue

```
#include<iostream.h>  
#include<conio.h>  
struct node  
{  
    int data;  
    node * next;  
};  
  
class queue  
{  
    node *front,*rear;  
    public:  
    queue()  
    {  
        rear=front=NULL;  
    }  
    void insqueue();
```

```
void delqueue();
void dispqueue();
};

void queue::insqueue()
{
    node *ptr;
    ptr=new node;
    cout<<"Enter the element to be insert"<<endl;
    cin>>ptr->data;
    ptr->next=NULL;
    if(rear==NULL)
        front=rear=ptr;
    else
    {
        rear->next=ptr;
        rear=ptr;
    }
}

void queue ::delqueue()
{
    node *ptr;
    ptr=new node;
    ptr=front;
    cout<<"The deleted element is "<<ptr->data;
    if(front==rear)
        front=rear=NULL;
    else
        front=front->next;
    delete ptr;
}

void queue :: dispqueue()
{
    node *ptr;
    ptr=new node;
    ptr=front;
    cout<<"The queue is "<<endl;
    while(ptr!=NULL)
    {
        cout<<ptr->data<<endl;
```

```
        ptr=ptr->next;
    }
}
void main()
{
    clrscr();
    char ans;
    queue q1;
    do
    {
        q1.insqueue();
        cout<<"wish to continue "<<endl;
        cin>>ans;
    }while(ans=='y');
    q1.dispqueue();
    cout<<"Press any key to delete an element ...."<<endl;
    getch();
    q1.delqueue();
    getch();
}
```

Exercise

Q1. Write a function in C++ to delete a node containing customer's information, from a dynamically allocated Queue of Customers implemented with the help of the following structure:

```
struct Customer
{
    int CNo;
    char CName[20];
    Customer *Link;
};
```

Ans:

```
struct Customer
{
    int CNo;
    char CName[20];
    Customer *Link;
} *Front, *Rear, *ptr;
```

```
void DELETE()
{
    if(Front == NULL)
        cout<<"\n Queue Underflow\n";
    else
    {
        ptr = Front;
        Front = Front@Link;
        delete ptr;
    }
}
```

Q2. Write a function in C++ to delete a node containing Book's information, from a dynamically allocated Stack of Books implemented with the help of the following structure.

```
struct Book
{
    int BNo;
    char BName[20];
    Book *Next;
};
```

Ans:

```
struct Book
{
    int BNo;
    char BName[20];
    Book *Next;
}*Front, *Rear, *ptr;
void POP()
{
    if(Front == NULL)
        cout<<"\n Stack Underflow\n";
    else
    {
        ptr = Front;
        Front = Front@Link;
        delete ptr;
    }
}
```

Unit 6

DATABASE AND SQL

Objectives:

In this Chapter you will learn about:

- Basic database concepts
- Key Constraints
- Syntax of some SQL commands



Introduction:

Database is a collection of related information that is organized in such a way that supports for easy access, modify and maintain data. The contents of a database are obtained by combining data from all the different sources in an organization. Generally, the database is managed by some special software packages known as Database Management Systems (DBMSs). DBMSs are specially designed applications to create connection between user and program, and to store data in an organized manner. The purpose of DBMSs software is to allow the user to create, modify and administration of database. Examples of database management systems are:

Ms-Access, MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP, dBASE, FoxPro, etc.

Relational data model:

The relational data model is a database model based on first-order predicate logic (First Order Predicate Logic is one where the quantification is over simple variables), formulated and proposed by Edgar F. Codd. in 1969. The first-order predicate logic is a symbolised reasoning, in which statement is broken down into a subject and a predicate. The predicate modifies the properties of the subject, while in the first order logic, a predicate can only refer to a single subject. In the relational data model, database is represented as collection of related tables. Each table is termed as relation and has its unique name in the relational data model. Tables are formed by using rows and columns. A row (horizontal subset) of a table represents a tuple or record, while column (vertical subset) of a table represents an attribute.

Relation:

In database, a relation means a 'table', in which data are organized in the form of rows and columns. Therefore in database, relations are equivalent to tables.

For example

Relation: Student

Ad No	Name	Class	Section	Average
101	Anu	12	A	85
105	Balu	12	D	65
203	Leena	11	B	95
205	Madhu	10	B	75
305	Surpreeth	9	C	70
483	Usha	6	A	60

Domain:

A domain is the original sets of atomic values used to model data. In data base management and database, a domain refers to all the possible unique values of a particular column.

For example:

1. The domain of gender column has a set of two possible values i.e, Male or Female.
2. The domain of marital status has a set of four possible values i.e, Married, Unmarried, Widows and Divorced.

Therefore, a domain is a set of acceptable values of a particular column, which is based on various properties and data types. We will discuss data types later in this chapter.

Basic Database concepts

Data :- Raw facts and figures which are useful to an organization. We cannot take decisions on the basis of data.

Information:- Well processed data is called information. We can take decisions on the basis of information

Field: Set of characters that represents specific data element.

Record: Collection of fields is called a record. A record can have fields of different data types.

File: Collection of similar types of records is called a file.

Table: Collection of rows and columns that contains useful data/information is called a table. A table generally refers to the passive entity which is kept in secondary storage device.

Relation: Relation (collection of rows and columns) generally refers to an active entity on which we can perform various operations.

Database: Collection of logically related data along with its description is termed as database.

Tuple: A row in a relation is called a tuple.

Attribute: A column in a relation is called an attribute. It is also termed as field or data item.

Degree: Number of attributes in a relation is called degree of a relation.

Cardinality: Number of tuples in a relation is called cardinality of a relation.

Example:

Relation: Student

AdNo	Name	Class	Section	Average
101	Anu	12	A	85
105	Balu	12	D	65
203	Leena	11	B	95
205	Madhu	10	B	75
305	Surpreeth	9	C	70
483	Usha	6	A	60

Fields (Attributes/Columns):- AdNo, Name, Class, Section and Average.

Tuples (Rows/Records):

101	Anu	12	A	85
-----	-----	----	---	----

Domain: Possible values of section are ('A','B','C','D')

Degree: 5 (Number of columns).

Cardinality: 6 (Number of rows).

KEYS:

Keys are an important part of a relational database and a vital part of the structure of a table. They help enforce integrity and help identify the relationship between tables. There are three main types of keys - candidate keys, primary keys, foreign keys and alternate keys.

Primary Key: Primary key is a key that can uniquely identifies the records/tuples in a relation. This key can never be duplicated and NULL.

Foreign Key: Foreign Key is a key that is defined as a primary key in some other relation. This key is used to enforce referential integrity in RDBMS.

Candidate Key: Set of all attributes which can serve as a primary key in a relation.

Alternate Key: All the candidate keys other than the primary keys of a relation are alternate keys for a relation.

DBA: Data Base Administrator is a person (manager) that is responsible for defining the data base schema, setting security features in database, ensuring proper functioning of the data bases etc.

Select Operation: The select operation selects tuples from a relation which satisfy a given condition. It is denoted by lowercase Greek Letter σ (sigma).

Project Operation: The project operation selects columns from a relation which satisfy a given condition. It is denoted by lowercase Greek Letter π (pi). It can be thought of as picking a sub set of all available columns.

Union Operation: The union (denoted as $\cup$) of a collection of relations is the set of all distinct tuples in the collection. It is a binary operation that needs two relations.

Set Difference Operation: This is denoted by $-$ (minus) and is a binary operation. It results in a set of tuples that are in one relation but not in another. Structured Query Language SQL is a nonprocedural language that is used to create, manipulate and process the databases(relations).

Characteristics of SQL

1. It is very easy to learn and use.
2. Large volume of databases can be handled quite easily.
3. It is nonprocedural language. It means that we do not need to specify the procedures to accomplish a task but just to give a command to perform the activity.
4. SQL can be linked to most of other high level languages that makes it first choice for the database programmers.

Processing Capabilities of SQL

The following are the processing capabilities of SQL

1. Data Definition Language (DDL)

DDL contains commands that are used to create the tables, databases, indexes, views, sequences and synonyms etc.

e.g: Create table, create view, create index, alter table etc.

2. Data Manipulation Language (DML)

DML contains command that can be used to manipulate the data base objects and to query the databases for information retrieval.

e.g Select, Insert, Delete, Update etc.

3. View Definition:

DDL contains set of command to create a view of a relation.

e.g : create view

4. Data Control Language:

This language is used for controlling the access to the data. Various commands like GRANT, REVOKE etc are available in DCL.

5. Transaction Control Language (TCL)

TCL include commands to control the transactions in a data base system. The commonly used commands in TCL are COMMIT, ROLLBACK etc.

Operators in SQL:

The following are the commonly used operators in SQL

- | | |
|-------------------------|---------------------|
| 1. Arithmetic Operators | +, -, *, / |
| 2. Relational Operators | =, <, >, <=, >=, <> |
| 3. Logical Operators | OR, AND, NOT |

Data types of SQL

Just like any other programming language, the facility of defining data of various types is available in SQL also. Following are the most common data types of SQL.

1. NUMBER e.g. Number(n,d) Number (5,2)
2. CHAR CAHR(SIZE)
3. VARCHAR / VARCHAR2 VARCHAR2(SIZE)
4. DATE DD-MON-YYYY

5. LONG
6. RAW/LONG RAW

Constraints:

Constraints are the conditions that can be enforced on the attributes of a relation. The constraints come in play whenever we try to insert, delete or update a record in a relation. Not null ensures that we cannot leave a column as null. That is a value has to be supplied for that column.

e.g name varchar(25) not null;

Unique constraint means that the values under that column are always unique.

e.g Roll_no number(3) unique;

Primary key constraint means that a column can not have duplicate values and not even a null value.

e.g. Roll_no number(3) primary key;

The main difference between unique and primary key constraint is that a column specified as unique may have null value but primary key constraint does not allow null values in the column.

Foreign key is used to enforce referential integrity and is declared as a primary key in some other table.

e.g cust_id varchar(5) references master(cust_id);

it declares cust_id column as a foreign key that refers to cust_id field of table master. That means we cannot insert that value in cust_id field whose corresponding value is not present in cust_id field of master table.

Check constraint limits the values that can be inserted into a column of a table.

e.g marks number(3) check(marks>=0);

The above statement declares marks to be of type number and while inserting or updating the value in marks it is ensured that its value is always greater than or equal to zero.

Default constraint is used to specify a default value to a column of a table automatically. This default value will be used when user does not enter any value for that column. e.g balance number(5) default = 0;

SYNTAX of some SQL COMMANDS :

1. SELECT column list FROM <table name> WHERE <condition> GROUP BY <column name(s)> HAVING <search condition> ORDER BY column name;
2. INSERT INTO <table name> [<column list>] VALUES (<value>, <value>, ...);
3. DELETE FROM <table name> [WHERE <predicate>];
4. UPDATE <table name> SET <column name> = <new value> [WHERE <predicate>];
5. CREATE TABLE <table name> (<column name><data type> [(size)] <column constraint>, <column name><data type> [(size)] <column constraint>, ... <table constraint> (<column name> [<column name> ...]) ...);
6. CREATE VIEW <view name> [(<column name>, <column name>,...)] AS <SELECT command>;
7. ALTER TABLE <table name> ADD / MODIFY <column name><data type><size>;
8. DROP TABLE <table name>;
9. DROP VIEW <view name>;



Consider the following Table:-

Roll_no	Name	Class	Marks	City
101	Ajaz	X	400	Srinagar
102	Farooq	XI	450	Budgam

103	Sajad	XI	430	Sopore
104	Javeed	XII	490	Bandipora
105	Humira	IX	330	Baramulla
106	Tahir	XI	404	Ganderbal

Eliminating Duplicate/Redundant data

DISTINCT keyword is used to restrict the duplicate rows from the results of a SELECT statement.

e.g. SELECT DISTINCT name FROM student;

The above command returns

Name

Ajaz

Farooq

Sajad

Conditions based on a range:

SQL provides a BETWEEN operator that defines a range of values that the column value must fall for the condition to become true.

e.g. SELECT Roll_no, name FROM student WHERE Roll_no BETWEEN 100 AND 103;

The above command displays Roll_no and name of those students whose Roll_no lies in the range 100 to 103 (both 100 and 103 are included in the range).

The above command returns

Roll_no	Name
101	Ajaz
102	Farooq
103	Sajad

Conditions based on a list:

To specify a list of values, IN operator is used. This operator select values that match any value in the given list.

e.g. `SELECT*FROM student WHERE city IN ('Srinagar','Sopore','Baramullah');`

The above command displays all those records whose city is either 'Srinagar','Sopore','Baramullah'

The above command returns

Roll_no	Name	Class	Marks	City
101	Ajaz	X	400	Srinagar
103	Sajad	XI	430	Sopore
105	Humira	IX	330	Baramulla

Conditions based on Pattern

SQL provides two wild card characters that are used while comparing the strings with LIKE operator.

- percent(%) Matches any string
- Underscore(_) Matches any one character

e.g `SELECT Roll_no, name, city FROM student WHERE Roll_no LIKE "%3";`

displays those records where last digit of Roll_no is 3 and may have any number of characters in front.

e.g `SELECT Roll_no, name, city FROM student WHERE Roll_no LIKE "1_3";`

displays those records whose Roll_no starts with 1 and second letter may be any letter but ends with digit 3.

ORDER BY Clause

ORDER BY clause is used to display the result of a query in a specific order(sorted order). The sorting can be done in ascending or in descending order. It should be kept in mind that the actual data in the database is not sorted but only the results of the query are displayed in sorted order.

e.g. `SELECT name, city FROM student ORDER BY name;`

The above query returns name and city columns of table student sorted by name in increasing/ascending order.

e.g. `SELECT * FROM student ORDER BY city DESC;`

It displays all the records of table student ordered by city in descending order.

Note:- If order is not specifies that by default the sorting will be performed in ascending order.

GROUP BY Clause

The GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

The syntax for the GROUP BY clause is:

```
SELECT column1, column2, ... column_n, aggregate_function (expression)
FROM tables
WHERE conditions
GROUP BY column1, column2, ... column_n;
```

Aggregate Function:

aggregate_function can be a function such as SUM, COUNT, MAX, MIN, AVG etc.

e.g `SELECT name, COUNT(*) as "Number of employees"
FROM student
WHERE marks>350
GROUP BY city;`

```
SELECT MIN(Sal), MAX(Sal), AVG(Sal),
FROM Emp
WHERE DeptNo = 'D1'
```

Example: (SUM, AVG, MAX, MIN)

```
SELECT SUM (fees)
FROM student;
```

```
SELECT AVG (fees)
FROM student;
```

```
SELECT MAX (fees)
FROM student;
```

```
SELECT MIN (fees)
FROM student;
```

The INSERT Command:

The tuples are added to relation using INSERT command of SQL.

Syntax:

```
INSERT INTO <table-name>[<column list>]
VALUES (<value>,<value>,<value>,.....);
```

Example :

Enter a new record in student table

```
INSERT INTO student (sid,sname,fname,age,class,address);
VALUES(107,"Jameel","Farooq",18,"XII","Pulwama");
```

<u>sid</u>	<u>sname</u>	<u>fname</u>	<u>age</u>	<u>class</u>	<u>address</u>
107	Jameel	Farooq	18	XII	Pulwama

The DELETE Command:

The delete command removes the tuples from the tables. This command remove the entire row from the table and not the individual field. So no filed argument is needed.

Syntax:

```
DELETE FROM <table-name>
WHERE <condition>;
```

Example:

Delete all the records of employee whose salary is less than 3000

```
DELETE FROM emp
WHERE sal<3000;
```

To delete all the record from the table:

```
DELET FROM<table-name>;
```

The UPDATE command is used to changes some values in existing rows. The UPDATE command specifies the rows to be changed using the WHERE clause, and new data using the SET keyword.

Example:

Update the salary of employee to 5000 whose employee code is 1011.

```
UPDATE emp  
SET sal=5000  
WHERE empno=1011;
```

The ALTER TABLE Command:

The ALTER command is used to change the definition of existing table.

1. It can be used to add columns to a table.

Syntax (to add a column to a table):

```
ALTER TABLE <table-name> ADD <column-name>  
<data type> <size>;
```

2. To modify existing columns of a table:

Syntax:

```
ALTER TABLE <table-name>  
MODIFY (Columnname newdatatype (news�));
```

Example:

To modify column job of table emp to have new width of 30 character

```
ALTER TABLE emp  
MODIFY (job char(30));
```

The DROP Command:

The DROP command is used to drop the table from the database. For dropping a table all the tuples should be deleted first i.e the table should be empty.

Syntax:

```
DROP TABLE <table-name>
```

Practice work:

Consider the following partial relation EMP. Let us create some sub-queries for them.

EMPNO	ENAME	JOB	SAL	DEPTNO
7566	Nirmal	MANAGER	2975	10
7788	Kailash	ANALYST	3000	10
7839	Karuna	PRESIDENT	5000	20
7902	Ashwin	ANALYST	3000	20
7905	Ashwini	MANAGER	4000	20

Example 1: Get the details of the person having the minimum salary.

```
SELECT ENAME, JOB, SAL
FROM EMP
WHERE SAL = (SELECT MIN (SAL)
              FROM EMP);
```

OUTPUT:

ENAME	JOB	SAL
Nirmal	MANAGER	2975

Example 2: Display the employees whose job title is the same as that of employee 7566 and salary is more than the salary of employee 7788

```
SELECT ENAME, JOB
FROM EMP
WHERE JOB = (SELECT JOB
              FROM EMP
              WHERE EMPNO = 7566)
```

OUTPUT:

ENAME	JOB
Ashwini	MANAGER

Example 3: To find the minimum SAL in those departments whose minimum salary is greater than minimum salary of department number 10.

```
SELECT DEPTNO, MIN (SAL)
FROM EMP
GROUP BY DEPTNO
HAVING MIN (SAL)> (SELECT MIN (SAL)
                   FROM EMP)
```

WHERE DEPTNO = 10);

OUTPUT:

DEPTNO	SAL
10	2975
20	3000

Example

salary of employees drawing minimum salary in that department.

4: Find the name, department no and

```
SELECT ENAME, SAL, DEPTNO
FROM EMP
WHERE SAL IN = (SELECT MIN (SAL)
                FROM EMP
                GROUP BY DEPTNO);
```

OUTPUT:

ENAME	SAL	DEPTNO
Nirmal	2975	10
Ashwin	3000	20

Example 5: Find the AVERAGE salary in each department.

```
SELECT DEPTNO, AVG (SAL)
FROM EMP
GROUP BY DEPTNO;
```

OUTPUT:

DEPTNO	SAL
10	2987.5
20	4000

Example:

Table: Student

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	2500
222	Mohit Sharma	11	B	4500
333	K.P.Gupta	12	B	3000
444	Ajit Kumar	10	A	2000
555	Nandini	12	C	3000
666	Rohan Sharma	11	B	2500

Relational Operator (< less than , > greater than <= less than or equal to,> = greater than or equal to, = equal to, != not equal to)

Example 6: Display students' name, who are paying below 3000 fees.

```
SELECT name
FROM student
WHERE fees<3000;
```

Output:

Name
Anu Jain
Ajit Kumar
Rohan Sharma

Example 7: Display students' name, who are paying above or equal to 3000 fees.

```
SELECT name
FROM student
WHERE fees>=3000;
```

Output:

Name
Mohit Sharma
Nandini

Example 8: Display students' information, who are not in class 10

```
SELECT *
FROM student
WHERE class!= 10;
```

Output:

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	2500
222	Mohit Sharma	11	B	4500
333	K.P.Gupta	12	B	3000
555	Nandini	12	C	3000
666	Rohan Sharma	11	B	2500

Logical Operator (AND, OR, NOT)

Example 9: Display information of students in class 11B.

```
SELECT *  
FROM student  
WHERE class = 11 AND section = 'B';
```

Output:

Adno	Name	Class	Section	Fees
222	Mohit Sharma	11	B	4500
666	Rohan Sharma	11	B	2500

Example 10: Display 11th and 12th class students' information.

```
SELECT *  
FROM student  
WHERE class =11 OR class=12;
```

Output:

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	2500
222	Mohit Sharma	11	B	4500
333	K.P.Gupta	12	B	3000
555	Nandini	12	C	3000
666	Rohan Sharma	11	B	2500

Example 11: Display students' information, who are not in 10th class.

```
SELECT *  
FROM student  
WHERE NOT class = 10;
```

Output:

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	2500
222	Mohit Sharma	11	B	4500
333	K.P.Gupta	12	B	3000
555	Nandini	12	C	3000
666	Rohan Sharma	11	B	2500

Unit 7

Boolean Logic

Objectives:

In this Chapter you will learn about:

- Boolean Algebra & Theorems
- Basic Logic Gates
- Canonical and Standard Forms (SOP & POS)



BOOLEAN ALGEBRA:

Boolean Algebra was introduced by George Boole in 1854. He was an English mathematician, who primarily worked on differential equations and the calculus of variations. But the work for which he is best known is - application of algebraic systems to non-mathematical reasoning. He proposed that logical propositions should be expressed as algebraic equations, so now logic was reduced to algebra. In his logical deductions he replaced the operation of multiplication by AND, and addition by OR. However, it is Claude Shannon, whose work on using Boolean Algebra to design Switching Circuits in late 1930s, became the foundation of computer Logic and Design. As Boolean logic allows things to be mapped into bits and bytes, it has application in telephone switching circuits, electronics, computers (both hardware and software), etc.

Boolean Algebra, which is algebra of two values may be (True, False) or (Yes, No) or (0, 1), is an important tool in analyzing, designing and implementing digital circuits.

Boolean Algebra is made up of:

- ❖ Elements - which are variables or constants with value 1 or 0.
- ❖ Operators - which are And, Or and Not.
- ❖ Axioms and Theorems.

A Boolean variable is a symbol used to represent a logical quantity. It will take value from the domain $\{0, 1\}$, and Boolean constant is single digit binary value (bit) viz. 0 or 1.

There are three fundamental operators- AND, OR and NOT. AND is a binary operator, to perform logical multiplication, it is represented by '.' OR is also a

binary operator, to perform logical addition. It is represented by '+'. NOT is a unary operator, to complement the operand. Not is represented as ' or $\bar{}$. Complement is the inverse of a variable/ constant. In case of Boolean algebra, since the variable/constant can have value 0 or 1 so complement will be 1 or 0.

Note: A binary operator is applied on two operands and unary to one.

Just like algebra, Boolean algebra also have axioms and theorems, which describe how logical quantities behave. We know that axiom is a statement which is considered to be true, and theorems are to be proved.

First axiom is Closure Property, it states that Boolean operations (+ or .) on Boolean variables or constants will always result into a Boolean value.

Following are other axioms and theorems of Boolean algebra:

Identity: states that, sum of anything and zero, or product of anything and one, is same as the original anything. So identity with respect to + is 0, and with respect to . is 1

$$A + 0 = A \text{ and } A.1 = A$$

Commutative: property tells us, we can reverse the order of variables, that are either ORed together or ANDed together without changing the truth of the expression.

$$A + B = B + A \text{ and } A . B = B . A$$

Distributive: law states that, ORing variables and then ANDing the result with single variable is equivalent to ANDing the result with a single variable with each of the several variables and then ORing the products. Vice versa with respect to operators and terms is also true.

$$A . (B + C) = A . B + A . C \text{ and } A + (B . C) = (A + B) . (A + C)$$

Complement: states that sum of a Boolean quantity with its complement or product of a Boolean quantity with its complement results into identity

$$A + A' = 1 \text{ and } A . A' = 0$$

Idempotency: states that when we sum or product a boolean quantity to itself, the resultant is original quantity.

$$A + A = A \text{ and } A . A = A$$

Null Element: No matter what the value of Boolean variable, the sum of variable and 1 will always be 1. Also the product of Boolean variable and 0 will always be 0.

$$A + 1 = 1 \text{ and } A \cdot 0 = 0$$

Involution: states that complementing a variable twice, or any even number of times, results in the original Boolean value.

$$(A')' = A$$

Associative: this property tells us that we can associate, group of sum or product variables together with parenthesis without altering the truth of the expression

$$A + (B + C) = (A + B) + C \text{ and } A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

Absorption: is also known as covering, have three forms

1. $A + AB = A$ and $A \cdot (A + B) = A$
2. $A + (A' \cdot B) = A + B$ and $A \cdot (A' + B) = A \cdot B$
3. $(A + B) \cdot (A' + C) \cdot (B + C) = (A + B) \cdot (A' + C)$ AND $(A \cdot B) + (A' \cdot C) + (B \cdot C) = A \cdot B + A' \cdot C$

De Morgan's Theorem: states that the complement of a sum / product, equals the product / sum of the complements.

$$(A + B)' = A' \cdot B' \text{ and } (A \cdot B)' = A' + B'$$

Apart from these axioms and theorems, there is a principle. Duality Principle which states that starting with Boolean relation, you can device another Boolean relation by

1. Changing each OR operation to AND
2. Changing each AND operation to OR
3. Complementing the identity (0 or 1) elements.

For example dual of relation

$$A + 1 = 1 \text{ is } A \cdot 0 = 0.$$

You must have noticed that second part of axioms and theorems is actually the result of this principle only. Practical consequence of this theorem is, it is sufficient to prove half the theorem, the other half comes gratis from the principle.

If we compare Boolean Algebra with Arithmetic algebra, we will find that

1. In Boolean algebra + can distribute over .

2. Logical subtraction and logical division are not available in Boolean algebra, as there is no additive or multiplicative inverse available.
3. Complement is available in Boolean algebra
4. Two valued Boolean algebra contains 0 & 1 only.

Boolean Function - is an expression formed with boolean variable(s), boolean constant(s), boolean operator(s), parenthesis and equal sign. It will always result into a boolean value, as specified by closure property. There are three ways to represent a boolean expression/function viz.

1. Algebraic
2. Truth Table
3. Circuit Diagram

Algebraic Representation: Axioms and Theorems written above are examples of algebraic representation, but let's have some more examples:

$$F_1 = a.b$$

formed using two boolean variables a & b with AND operator. This function will be equal to 1, for a = 1 and b = 1, otherwise it will be 0.

$$F_2 = xyz' \quad 2$$

formed using three boolean variables x, y and z with AND operator. Function will be 1 if x = 1, y = 1, and z' = 1 i.e. z = 0, in all other situations F will be 0. 2

They can also be represented as

$$F_1(a, b) = a.b \quad 1$$

$$F_2(x, y, z) = x.y.z' \quad 2$$

These are examples of some simple Boolean functions. Complex boolean functions can be formed using simple expressions. Following are the examples of such functions:

$$F_3(a, b, c) = (a+b).c \quad 3$$

$$F_4(x, y) = (x+y).(x'+y') \quad 4$$

$$F_5(x, y, z) = x.(y+z).(x+y') \quad 5$$

$$F_6(w, x, y, z) = y'z + wxy' + wx'z \quad 6$$

When an expression/function contains two or more operators, the order of operations is decided by parenthesis and precedence of operators. Moving from high to low precedence, following table gives the precedence of Boolean operators.

Operator	Precedence
()	High
NOT	
AND	
OR	
	Low

Truth Table representation:

A truth table is a chart of 1's and 0's, arranged to indicate the results of all possible input options. Number of columns in the table is decided by the number of Boolean variables in the function. Each variable will be represented in a column. For n variable expression, there will be 2^n rows in the table. For filling these rows in the table, they will be the binary representation of the integers from 0 to (2^n-1) , in the same order.

Let's fill the table for one variable – **a**

a
0
1

Since the table is for one variable there will be only two rows $2^1 = 2$ rows. For two variables - a & b, the table will have 4 rows ($2^2 = 4$), listed below

a	b
0	0
0	1
1	0
1	1

Three Variables - a, b & c, 8 rows

a	b	c
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

and so on. The trick for filling the table is, starting from right most column of the table - 1st column will be combination of 0,1,0,1,..... (2^0 zeros and then same number of 1s. In 2nd column it will be 0,0,1,1,0,0,..... 2^1 zeros and then same number of ones. In 3rd column 0,0,0,0,1,1,1,1,0,0, (2^2 number of zeros and ones), and so on.

Now let's use one variable table, to represent a simple boolean function $f(a) = a'$

a	F(a)
0	1
1	0

Second column of table gives the value of Boolean function i.e. a' . This table tells that if a is true, a' will be false and vice versa, as is evident from 2nd column of table.

Truth table for the boolean function applied on two variables with all basic Boolean operators i.e. AND, OR will be

a	b	F=a.b	F=a+b
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Now let's represent some more functions written earlier in algebraic form using truth table:

$$F_2(x,y,z) = xyz'$$

x	y	z	f2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

From our earlier explanation we know that F will be one for $x = 1, y = 1$ and $z = 0.2$

$$f_3(a, b, c) = (a+b).c$$

a	b	c	(a+b)	f3 = (a+b).c
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

$$f_4(x, y) = (x+y). (x'+y')$$

x	y	x'	y'	(x+y)	(x'+y')	f4 = (x+y).(x'+y')
0	0	1	1	0	1	0
0	1	1	0	1	1	1
1	0	0	1	1	1	1
1	1	0	0	1	0	0

For now let's move on to proving the various theorems of Boolean algebra:
Theorems can be proved in two ways:

- Algebraic Method:** In this method, we use axioms & theorems. This method is similar to what you do in mathematics.
- Truth Table:** Here we use truth table to show that expression given on LHS results into same values for expression on RHS. This actually is verification of theorem, not proving.

Associative:

Theorem states that $A + (B+C) = (A+B) + C$ and $A.(B.C) = (A.B).C$

Let's algebraically prove it:

$$\text{Let } X = [A + (B + C)] [(A + B) + C]$$

then

$$\begin{aligned}
 X &= [(A + B) + C] A + [(A + B) + C] (B + C) \\
 &= [(A + B) A + C . A] + [(A + B) + C] (B + C) \\
 &= A + [(A + B) + C] (B + C) \\
 &= A + [(A + B) + C] B + [(A + B) + C] C \\
 &= A + (B + C)
 \end{aligned}$$

But also

$$\begin{aligned}
 X &= (A + B) [A + (B + C)] + C [A + (B + C)] \\
 &= (A + B) [A + (B + C)] + C \\
 &= A [A + (B + C)] + B [A + (B + C)] + C \\
 &= (A + B) + C
 \end{aligned}$$

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

We know, second form of the theorem will also be true. (Duality Principle)

Truth table method:

A	B	C	B+C	LHS $A+(B+C)$	(A+B)	RHS $(A+B)+C$
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

As LHS= RHS, hence verified

Null Element states that $A + 1 = 1$ and $A . 0 = 0$

Algebraic method:

Taking LHS

$$\begin{aligned}
 A+1 &= (A+1).1 \text{ by Identity} \\
 &= (A+1). (A+A') \text{ by Complement} \\
 &= A + (1.A') \text{ by Distribution} \\
 &= A+ (A'.1) \text{ by Idempotancy \& Commutative} \\
 &= A+A' \text{ by Identity} \\
 &= 1 \text{ by Complement}
 \end{aligned}$$

By applying the principle of Duality, we get $A \cdot 0 = 0$

Truth Table method:

A	1	LHS $A+1$	RHS
0	1	1	1
1	1	1	1

As LHS= RHS, hence verified

Involution states that $((A'))' = A$

Taking **LHS**

$$\begin{aligned}
 (A')' &= (A')' + 0 && \text{by Identity} \\
 &= (A')' + (A' \cdot A) && \text{by Complement} \\
 &= ((A')' + A') \cdot ((A')' + A) && \text{by Distribution} \\
 &= 1 \cdot ((A')' + A) && \text{by Complement} \\
 &= (A')' + A && \text{by Identity} \quad (1)
 \end{aligned}$$

Taking **LHS** again

$$\begin{aligned}
 (A')' &= (A')' \cdot 1 && \text{by Identity} \\
 &= (A')' \cdot (A + A') && \text{by Complement} \\
 &= (A')' \cdot A + ((A')' \cdot A') && \text{by Distribution} \\
 &= (A')' \cdot A + 0 && \text{by Complement} \\
 &= (A')' \cdot A && (2)
 \end{aligned}$$

Substituting the value of $(A')' = (A')' \cdot A$ in (1), we get

$$\begin{aligned}
 (A')' + A &= (A')' \cdot A + A \\
 &= A + (A')' \cdot A && \text{by Commuting} \\
 &= A + A \cdot (A')' && \text{by Commuting} \\
 &= A && \text{by Absorption}
 \end{aligned}$$

We know, second form of the theorem will also be true. (Duality Principle)

Truth Table method:

A	(A')	LHS $(A')'$	RHS A
0	1	0	0
1	0	1	1

As LHS= RHS, hence verified

Idempotency states that $A+A=A$ and $A \cdot A = A$

Algebraic method:

Taking L.H.S

$$A+A = (A+A) \cdot 1 \text{ by identity}$$

$$= (A+A) \cdot (A+A') \text{ by complement}$$

$$= (AA+AA') + (AA + AA') \text{ by distribution}$$

$$= AA+0$$

$$= A$$

We know, second form of the theorem will also be true. (Duality Principle)

Truth Table method:

A	RHS A	LHS A+A
0	0	0
1	1	1

As LHS = RHS, hence verified

Absorption Law:

i) $A + AB = A$ and $A \cdot (A + B) = A$

Algebraic method:

Taking LHS

$$A + AB = (A \cdot 1) + (A \cdot B) \text{ by Identity}$$

$$= A \cdot (1+B) \text{ by Distribution}$$

$$= A \cdot 1 \text{ by Null Element}$$

$$= A$$

Using Duality, we know that $A \cdot (A+B) = A$ is also true

Truth Table method:

A	B	A.B	LHSA+AB	RHS A
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	1	1

As LHS=RHS, hence verified

ii) $(A + A'B) = A + B$ and $A \cdot (A' + B) = A \cdot B$

Algebraic method:

Taking LHS

$$A + A'B = (A + A') \cdot (A + B) \text{ by Distribution}$$

$$= 1 \cdot (A + B) \text{ by Complement}$$

$$= A + B \text{ by Identity}$$

Because of duality, $A \cdot (A' + B) = A \cdot B$ is also true.

Truth Table method:

A	B	A'B	LHS $A + A'B$	RHS $A + B$
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

As LHS = RHS, hence verified

iii) $((A+B) \cdot (A'+C) \cdot (B+C)) = (A+B) \cdot (A'+C)$ AND $((A \cdot B) + (A' \cdot C) + (B \cdot C)) = A \cdot B + A' \cdot C$

Taking LHS

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

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

by Idempotency

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

by Commuting

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

by Commuting

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

by Distributive

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

by Distribution

$$= B \cdot C + A \cdot C \cdot C + B \cdot A' \cdot B + A \cdot C \cdot A'B$$

by Distribution

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

by Associativity and Idempotency

$$= B \cdot C + A \cdot C + B \cdot A' + 0$$

by Complement and Idempotency

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

by Complement and Distributive

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

by Distributive

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

by Distributive

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

by Commuting

Using Duality Principle we can say that $A \cdot B + A' \cdot C + B \cdot C = A \cdot B + A' \cdot C$ is also true

Truth Table method:

A	B	C	A+B	A'+C	B+C	LHS (A+B).(A'+C). (B+C)	RHS (A+B). (A'+C)
0	0	0	0	1	0	0	0
0	0	1	0	1	1	0	0
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	1	0	0	0	0
1	0	1	1	1	1	1	1
1	1	0	1	0	1	0	0
1	1	1	1	1	1	1	1

As LHS= RHS, hence verified

De Morgan's Theorem:

States that $(A+B)' = A'.B'$ and $(A.B)' = A' + B'$

Algebraic method:

Assuming that DeMorgan's theorem is true, then through this we can find complement of a function/expression.

We also know that

$$X+X' = 1 \text{ and } X.X'=0$$

So by showing that

$$\text{i) } (A+B) + A'.B' = 1$$

$$\text{ii) } (A+B). (A'.B') = 0$$

We shall establish the DeMorgan's theorem.

$$\text{i) } (A+B) + A'.B' = 1$$

Taking LHS

$$= (A+B) + (A'. B') = ((A+B)+ A'). ((A+B)+ B')$$

By Distribution

$$= (A+(B+ A')). (A+(B+ B'))$$

By Associativity

$$= (A+(A'+B)). (A+(B+ B'))$$

By Commutativity

$$= (A+ (A'+B)). (A+1)$$

By Complement

$$= ((A+ A')+B).(A+1)$$

By Commutativity

$$= (1+B). (A+1)$$

By Complement

$$= 1.1$$

By Null element

$$= 1$$

$$\text{ii) } (A+B). (A'.B')= 0$$

Taking LHS

$$= (A.(A'.B') + (B. (A'.B'))$$

$$= ((A.A').B') + ((B.A'). B')$$

By Associativity

$$= 0 + ((B'.A'). B')$$

By Null Element

$$= 0 + ((A'.B).B')$$

By Commutativity

$$= 0 + (A'+(B.B'))$$

By Associativity

$$= 0 + 0$$

By Null Element

$$= 0$$

Truth Table method:

A	B	A+B	LHS (A+B)'	A'	B'	RHS A'.B'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Hence De Morgan's theorem is established. Using Duality we can say that
 $(A . B)' = A' + B'$

BASIC LOGIC GATES



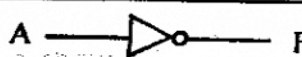
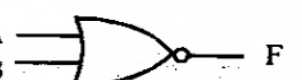
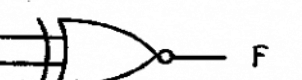
Logic Gates:

A gate is simply an electronic circuit, which operates on one or more signals to produce an output signal. Gates are digital circuits because the input and output signals are either low (0) or high (1). Gates also called logic circuits.

This logic circuit diagram can be composed of basic gates AND, OR and NOT or advance gates, NAND and NOR, etc.

Logic circuits that perform logical operations such as AND, OR and NOT are called gates. A gate is block of hardware that produces a logic 0 or a logic 1 output, in response to the binary signal(s) applied to it as input.

Let's look at the symbols used to represent digital logic gates

Name	Graphic Symbol	Algebraic function	Truth Table															
NOT		$F = \bar{A}$ or $F = A$	<table> <tr> <th>A</th> <th>F</th> </tr> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </table>	A	F	0	1	1	0									
A	F																	
0	1																	
1	0																	
AND		$F = A.B$ or $F = AB$	<table> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = A + B$	<table> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NAND		$F = \overline{A.B}$	<table> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = \overline{A + B}$	<table> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0
A	B	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
Exclusive-OR (XOR)		$F = A\bar{B} + \bar{A}B$ $F = A \oplus B$	<table> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
Exclusive-NOR (XNOR)		$F = \overline{A \oplus B}$	<table> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

Canonical and Standard Forms:

In a boolean function, a binary variable can appear in any form, normal (a) or complemented (a'). Now consider a boolean expression formed on two variables a & b using AND operator. Since each variable may appear in any form, there are four possible ways of combining these two variables viz. a'b', ab', a'b, ab.

Similarly if we combine them using OR operator, again there are four possible ways viz. a+b, a'+b, a+b', a'+b'

For now we will concentrate on the terms formed using AND operator only. Each of them is called a minterm or Standard Product Term. A minterm, is obtained from AND term, with each variable being complemented, if the corresponding bit of binary number (in the truth table) is 0 and normal (i.e. non complemented) if the bit is 1. These terms are designated using **m**.

Following table provide minterm for 3 variables:

a	b	c	Minterm	Designation
0	0	0	a'b'c'	m ₀
0	0	1	a'b'c	m ₁
0	1	0	a'bc'	m ₂
0	1	1	a'bc	m ₃
1	0	0	ab'c'	m ₄
1	0	1	ab'c	m ₅
1	1	0	abc'	m ₆
1	1	1	abc	m ₇

For four variables, there will be 16 minterms designated by m₀ to m₁₅.

Similarly, OR terms are called maxterms or Standard Sum Term. A maxterm is obtained from OR operator, variable being complemented if the corresponding bit of binary number (in the truth table) is 1 and normal if the bit is 0. These terms are designated by M

We already have maxterms for 2 variables, maxterms for 3 variables will be:

a	b	c	maxterm	designation
0	0	0	$a+b+c$	M_0
0	0	1	$a+b+c'$	M_1
0	1	0	$a+b'+c$	M_2
0	1	1	$a+b'+c'$	M_3
1	0	0	$a'+b+c$	M_4
1	0	1	$a'+b+c'$	M_5
1	1	0	$a'+b'+c$	M_6
1	1	1	$a'+b'+c'$	M_7

Maxterm for four variables can be obtained in similar manner and they are designated as M_0 to M_{15} . If we put both maxterms and minterms together, we will have following table

a	b	c	minterm	maxterm
0	0	0	$a'b'c'$	$a+b+c$
0	0	1	$a'b'c$	$a+b+c'$
0	1	0	$a'bc'$	$a+b'+c$
0	1	1	$a'bc$	$a+b'+c'$
1	0	0	$ab'c'$	$a'+b+c$
1	0	1	$ab'c$	$a'+b+c'$
1	1	0	abc'	$a'+b'+c$
1	1	1	abc	$a'+b'+c'$

By now you might have noted that each maxterm is complement of its corresponding minterm. For n variables, there will be 2^n minterms (i.e. Standard Product Term) denoted as m_i , $0 < i < 2^n - 1$, and 2^n maxterms (i.e. Standard Sum Term), their complement, denoted as M_i , $0 < i < 2^n - 1$.

Now let's use this information for algebraic representation of boolean function from the truth table. This is done by forwarding a minterm for each combination of variables which produces a 1 in the function, and then by ORing these minterms. For e.g.

a	b	c	F1	minterm
0	0	0	0	a'b'c'
0	0	1	1	a'b'c
0	1	0	0	a'bc'
0	1	1	0	a'bc
1	0	0	1	ab'c'
1	0	1	0	ab'c
1	1	0	1	abc'
1	1	1	1	abc

Since a'b'c, ab'c', abc' and abc minterms are resulting into 1, in function (F₁) column, so algebraic representation of the function F₁ will be

$$F_1(a,b,c) = a'b'c + ab'c' + abc' + abc$$

This is a boolean expression, expressed as sum of minterms and , will give value 1. This form of expression is known as Sum of Products or SOP expression. F₁ in SOP form can also be represented as

i) $m_1 + m_4 + m_6 + m_7$

Instead of using Product terms, we can just mention the minterm designation.

ii) $\sum (1, 4, 6, 7)$

This one is mathematical representation of the above expression.

Let's take some more examples:

x	y	z	F ₂	F ₃	F ₄
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	1	1	1
1	1	1	0	0	1

Algebraic representation in SOP form, of the three functions, represented in the truth table is

$$F_2(x,y,z) = x'yz + xy'z' + xyz'$$

OR

$$m_3 + m_4 + m_6$$

$$F_3(x,y,z) = x'y'z + x'yz + xy'z' + xy'z + xyz'$$

OR

$$m_1 + m_3 + m_4 + m_5 + m_6$$

$$F_4(x,y,z) = x'yz' + x'yz + xyz' + xyz$$

OR

$$m_2 + m_3 + m_6 + m_7$$

All of them will result into value 1.

What if we have to represent the function in complement form i.e. for its value 0. Simple, we will OR the

minterms, from the truth table, for which function value is 0. So we have,

$$F_1' = x'y'z' + x'yz' + x'yz + xy'z \text{ (the remaining minterms)}$$

On evaluating it for complement, we get

$$F_1 = (x+y+z). (x+y'+z) . (x+y'+z') . (x'+y+z')$$

What we get is the **Products of Sum form (POS)** of the function F_1 .

POS form of the expression gives value 0, and we work on the complements. POS is formed by ANDing maxterms.

POS also can be represented in many ways. One way we have seen, others are

$$\text{i) } F_1 = M_0 . M_2 . M_3 . M_5$$

Instead of using maxterms, we can use designation of it.

$$\text{ii) } F_1 = \prod(0, 2, 3, 5)$$

This one is mathematical representation of the above expression.

Let's look both, SOP and POS expression for same function

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

SOP expression for F will be

$$x'y'z + x'yz'$$
 or $\sum (1,2)$

POS equivalent of it will be found using complement of minterms with value zero, we know this

$$(F')' = (x'y'z' + x'yz + xy'z' + xy'z + xyz' + xyz)'$$

$$= (x + y + z) \cdot (x + y' + z') \cdot (x' + y + z) \cdot (x' + y' + z') \cdot (x' + y' + z) \cdot (x' + y' + z')$$

OR

$$= \prod (0,3,4,5,6,7)$$

Conversion from one form to another is easy. Short cut for it is:

- i) Interchange the symbol (OR to AND / $\sum$ to $\prod$)
- ii) List those terms, which are missing in original.

Expressions which were written till now, are in Canonical form. In this form, every term of the function should have all the literals. When, minterms or maxterms from truth table, are used for representation, the function is in canonical form only, as terms contains all variables (may be in any form). Since we can use either minterms or maxterms to form expression, we can have Canonical SOP expression or Canonical POS expressions.

Till now, we were given truth table and we were representing Boolean Function in algebraic form. Vice versa of it, i.e. for a given Boolean function in algebraic representation, we can represent it in Truth table.

What we have worked on was Canonical Boolean functions. There is another form of representation, also - both for SOP & POS. It is known as reduced / simplified form.

Simplified expressions are usually more efficient and effective when implemented in practice. Also simpler expressions are easier to understand, and less prone to errors. So the Canonical expressions are simplified/minimized to obtain a reduced expression (Standard form).

Minimization of Gates:

Boolean function may be reduced in many ways, but we will be taking up the following two ways of reduction in this course.

1. Algebraic Manipulations

2. Karnaugh Map.



Algebraic reduction of Boolean function/expression: in this reduction, we apply certain algebraic axioms and theorems to reduce the number of terms or number of literals in the expression. In this way of reduction, there are no fixed rules that can be used to minimize the given expression. It is left to individual's ability to apply axioms or theorems for minimizing the expression.

Example 1:

$$\begin{aligned} F(a, b) &= (a+b)' + ab' \\ &= a'b' + ab' && \text{by De Morgan's theorem} \\ &= (a'+a)b' && \text{by Distribution} \\ &= 1.b' && \text{by Complement} \\ &= b' && \text{by Identity} \end{aligned}$$

Example 2:

$$\begin{aligned} F(a, b) &= ab + ab' + a'b \\ &= a(b+b') + a'b && \text{by Distribution} \\ &= a.1 + a'b && \text{by complement} \\ &= a + a'b && \text{by Identity} \\ &= a+b && \text{by absorption} \end{aligned}$$

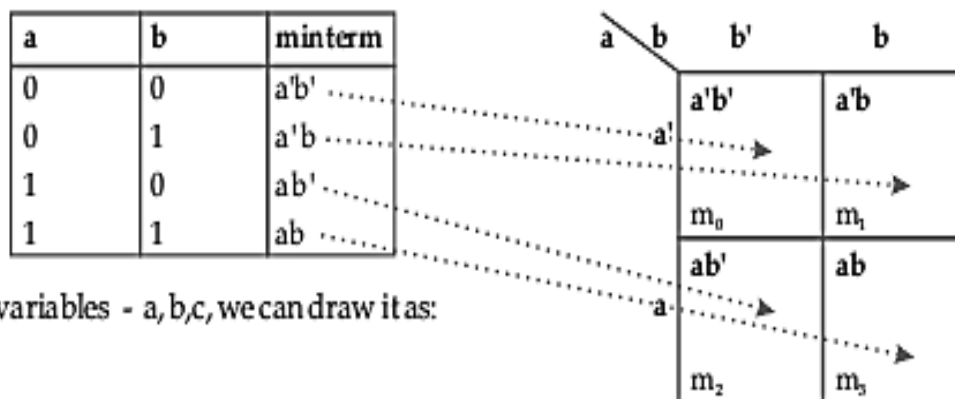
Using Karnaugh Map :

A Karnaugh map is graphical display of the fundamental products in a truth table. The map reduces boolean expression more quickly and easily, as compared to algebraic reduction.

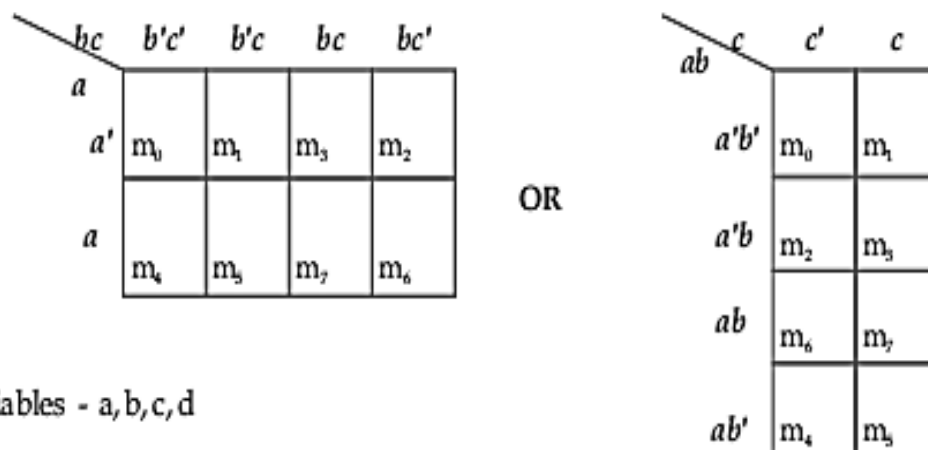
Using K-map for reduction is 4 step process

1. Drawing K-map
2. Filling it, i.e. representing the boolean expression in map
3. Grouping the terms for reducing purpose
4. Reading the reduced expression from map

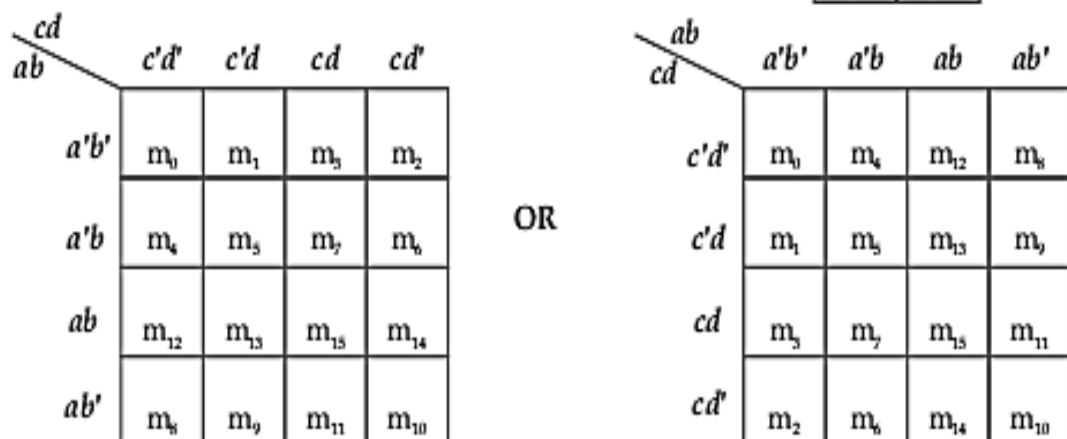
1. Drawing K-map:



For three variables - a, b, c, we can draw it as:



For four variables - a, b, c, d



The cells of the map are marked by the minterms as explained in first map. For e.g. If you look at the square assigned m (minterm), it corresponds to binary no. 0100 of the truth table.

Now to understand, why the squares have been marked, as they are?

If you look at any two adjacent cell in the map, they differ only by one variable, which is normal in one cell and complemented in other or vice-versa. For eg. From m to m only a changes its state from complemented to normal. And we know, using Boolean axiom ($a + a' = 1$) that we can eliminate the variable from the resultant reduced term.

2. Filling the K-map:

Once a K-map is drawn, next is to fill the map i.e. mark the function in the map. For representation, the function should be in canonical SOP form. If we have the function represented in truth table, then it will be in canonical form only. Otherwise, we know how to convert it. We will mark 1 in all those cells of K-map, for which there is a minterm in the Boolean function/expression.

3. Grouping the minterms:

Next is grouping the 1s in the map. We group 1s of adjacent cells.

Following are the rules for grouping. Remember our goal should be to maximize the size of group (i.e. no. of 1s included in the group) and to minimize the number of groups.

1. A group must contain 1, 2, 4, 8, 16, cells, i.e. in the power of 2.

1	1
0	0

Correct

0	0	0	0
0	1	1	1

Incorrect

2. Cells which are grouped, should be adjacent cells, i.e. horizontal or vertical, not diagonal.

0	1
1	1

Correct

0	1
1	0

Incorrect

3. Groups should not include any cell containing zero

0	0
1	1

Correct

0	0
1	0

Incorrect

4. Each group should be as large as possible

1	1	1	1
0	0	1	1

Correct

1	1	1	1
		1	1

Incorrect

5. Groups may overlap

1	1	1	1
0	0	1	1

Correct

1	1	1	1
		1	1

Incorrect

6. Groups may wrap around the table. The left most cell(s) in a row may be grouped with right most cell(s), and the top cell(s) in a column may be grouped with the bottom cells(s). Cells in corners of the map are also adjacent, for that purpose.

0	1	1	0
0	0	0	0
0	0	0	0
0	1	1	0

Correct

0	0	0	1
0	0	0	0
0	0	0	0
1	0	0	0

Incorrect

7. There should be as few groups as possible, as long as this does not contradict any of the previous rule. Once the grouping is done, last step is.

4. Reading the reduced expression from K-map:

- Each group corresponds to one product term, from K-map.
- The term is determined by finding the common literals in that group, i.e. the variables which change their state (from normal to complement or vice-versa) are to be omitted in resultant product term.

What we have to do is, find the variable(s) listed on top and / or side, which are the same for entire group and ignore variable(s) which are not same in the group.

Using the above strategy, write the result. Let's look at some examples

Simplify the Boolean Function $F(x,y,z) = x'yz + xy'z' + xyz + xyz'$

Step 1: Drawing the K-map

	yz	$y'z'$	$y'z$	yz	yz'
x					
x'					
x					

Step 2: Marking of function in map

	yz	$y'z'$	$y'z$	yz	yz'
x					
x'		0	0	1	0
x	1	0	1	1	1

Step 3: Grouping

	yz	$y'z'$	$y'z$	yz	yz'
x				1	0
x'				1	
x	1	0	1	1	1

Step 4: Reading reduced expression from map

For Group 1, variable x is changing its state and for Group 2, variable y is changing its state. So x in 1st group and y in 2nd group will be omitted. The resultant expression will contain yz , because of 1st group and xz' for 2nd. As this is SOP expression so both the terms will be joined through OR operator.

Resultant reduced function will be

$$F = yz + xz'$$

Exercise

Q1: Reduce the following Boolean Expression using K-Map:

$$F(A, B, C, D) = \sum (0, 3, 5, 6, 7, 8, 11, 15)$$

Ans:

	C+D			
A+B	0	1	3	2
A+B'	4	5	7	6
A'+B'	12	13	15	14
A'+B	8	9	11	10

There are 1 quad and 3 pairs after eliminating the redundant groups.

Quad (M₃, M₇, M₁₁, M₁₅)

reduces to $C' + D'$

Pair (M₅, M₇)

reduces to $A + B' + D'$

Pair (M₆, M₇)

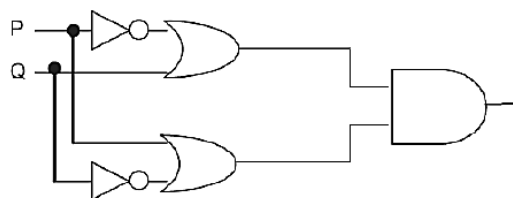
reduces to $A + B' + C'$

Pair (M₀, M₈)

reduces to $B + C + D$

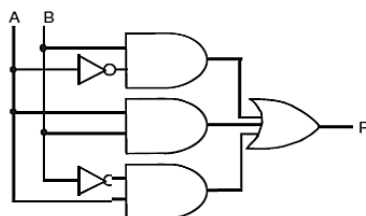
Hence, $F(A, B, C, D) = (C' + D') \cdot (A + B' + D') \cdot (A + B' + C') \cdot (B + C + D)$

Q2. Write the equivalent Boolean Expression for the following Logic Circuit



Ans: $F(P, Q) = (P' + Q) \cdot (P + Q')$

Q3. Write the equivalent Boolean Expression F for the following circuit diagram:



Ans: $A'B + AB + AB'$

Q4. Convert the following Boolean expression into its equivalent Canonical Sum of Product Form((SOP)

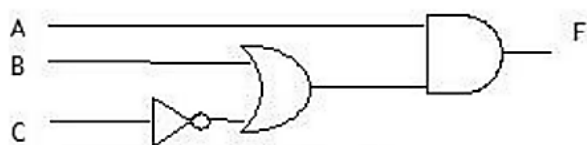
$$(X' + Y + Z')(X' + Y + Z)(X' + Y' + Z)(X' + Y' + Z')$$

Ans: $F(X, Y, Z) = \pi(4, 5, 6, 7)$
 $= \Sigma(0, 1, 2, 3)$
 $= X' \cdot Y' \cdot Z' + X' \cdot Y' \cdot Z + X' \cdot Y \cdot Z' + X' \cdot Y \cdot Z$

Q5. Draw a Logical Circuit Diagram for the following Boolean expression:

$$A.(B+C')$$

Ans:



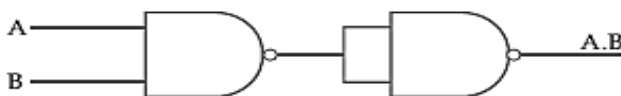
Q6. Draw a Logical Circuit Diagram for all logical Gates:

Ans:

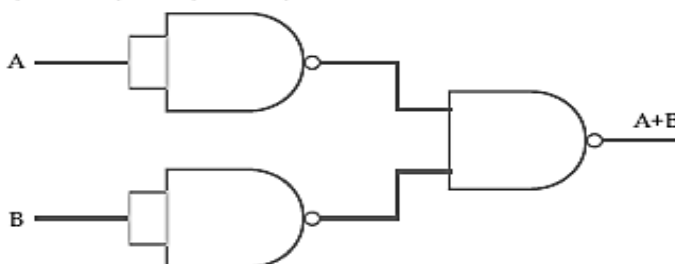
NOT $A' = (A.A)'$
 i.e. $A \text{ nand } A$



AND $A.B = ((A.B))'$
 i.e. $(A \text{ nand } B) \text{ nand } (A \text{ nand } B)$



OR $= A+B$
 $= ((A+B))'$
 $= (A'.B')'$
 $= ((A.A)'.(B.B))'$
 i.e. $(A \text{ nand } A) \text{ nand } (B \text{ nand } B)$



Unit 8

Networking and Cyber Security

Objectives:

In this Chapter you will learn about:

- Switching Techniques
- Data Communication Terminologies
- Transmission Media
- Network Security Concepts
- Cyber Security



Communication:

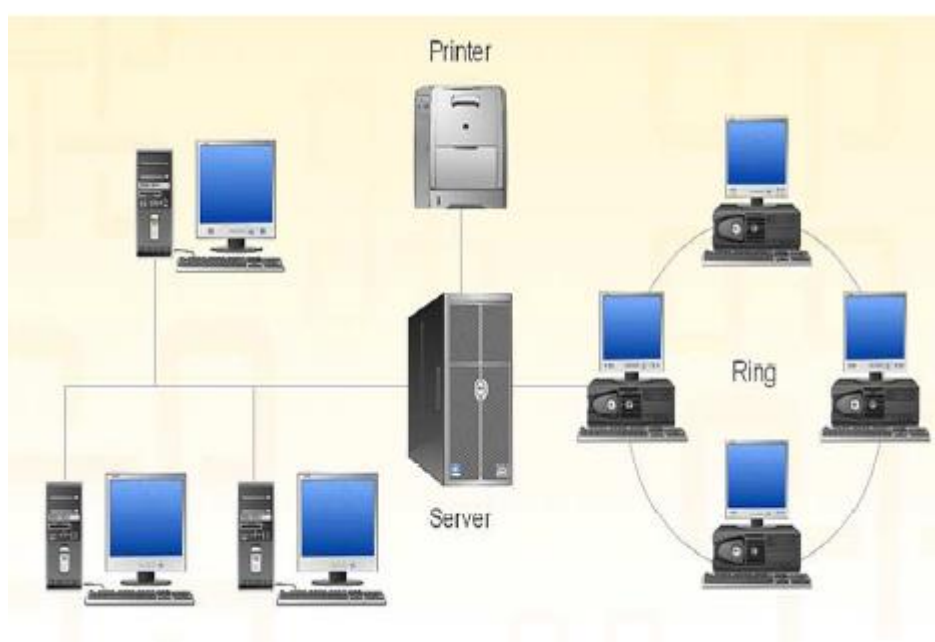
Communication means to convey. In today's fast changing world, data needs to be transferred efficiently, frequently and speedily. To move data at a fast speed and that too with minimum data loss, networking plays an important role. Computer networks have made a major impact on the society as a whole. As we are moving ahead in the 21st century, the world seems to be converging. The merging of computers and communication technology has changed the very perspective of communication today. It has had a profound influence on the way the computer systems are organized and used today. The old model of a single computer serving all of the organization's computational needs has been replaced by one in which large number of separate but interconnected computers do the same job. These systems together form a computer network.

Network:

We often need peripheral devices and data to be shared among various computers. In fact, in your school's computer lab, you must have seen one printer which is connected to only one computer, serving to the needs of all the computers in the lab. How does this happen? This happens because all your lab's computers and peripherals are forming a network. They are interconnected with each other enabling you to send and receive data from one computer to another. Hence it can be said that two computers are interconnected if they are able to exchange information.

A network is any collection of independent computers that communicate with one another over a shared network medium. In simple terms, a computer network is a collection of two or more computers linked together for the purpose of sharing

information and resources. When these computers are joined in a network, people can share files and peripherals such as modems, printers, backup drives, or CD-ROM drives. Each computer on the network is called a node and hence, a network is a series of points or nodes interconnected by communication paths (transmission media). A network can be as small and simple as two computers that share a printer or as complex as the world's largest network, the Internet. When networks at multiple locations are connected using services available from phone companies, people can send e-mail, share links to the global Internet, or conduct video conferences in real time with other remote users. As companies rely on applications like electronic mail and database management for core business operations, computer networking becomes increasingly more important.



Needs:

1. **Resource Sharing**
2. **Reliability**
3. **Cost Factor**
4. **Communication Medium**

Resource Sharing means to make all programs, data and peripherals available to anyone on the network irrespective of the physical location of the resources and the user.

Reliability means to keep the copy of a file on two or more different machines, so if one of them is unavailable (due to some hardware crash or any other) then its other copy can be used.

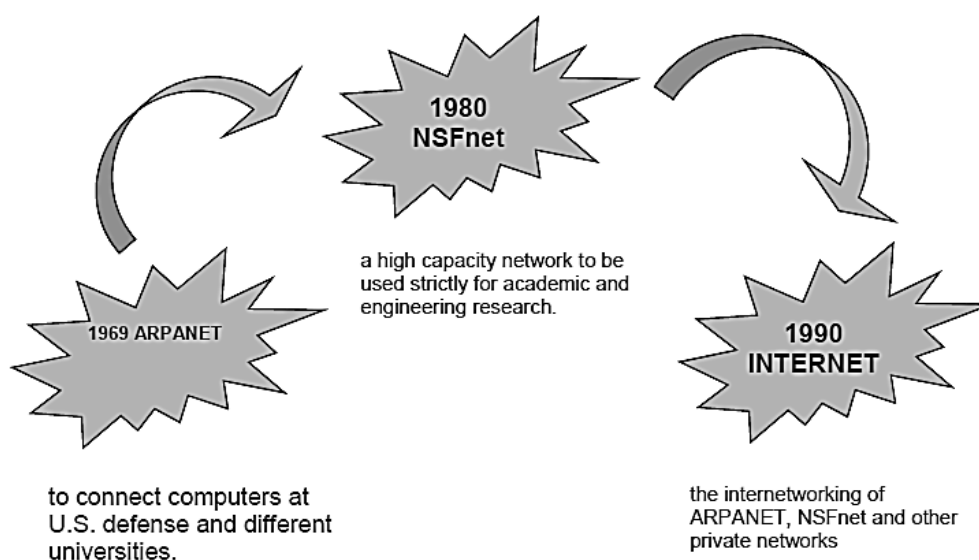
Cost factor means it greatly reduces the cost since the resources can be shared.

Communication Medium means one can send messages and whatever the changes at one end are done can be immediately noticed at another.

Evolution Of Networking:

Networking started way back in 1969 with the development of the first network called the ARPANET. The U.S. department of defense sponsored a project named ARPANET (Advanced Research Projects Agency Network) whose goal was to connect computers at different universities and U.S. defense. Soon engineers, scientists, students and researchers who were part of this system began exchanging data and messages on it. Gradually they could play long distance games and also socialize with people. Hence ARPANET expanded rapidly. In mid 80s, the National Science Foundation created a new high capacity network called NSFnet which allowed only academic research on its network. So many private companies built their own networks, which were later interconnected along with ARPANET and NSFnet to form Internet - a network formed by linking two or more networks.

Diagrammatic presentation



Internet:

The Internet is a system of linked networks that are worldwide in scope and facilitate data communication services such as remote login, file transfer, electronic mail, the World Wide Web and newsgroups. The internet is made up of many networks each run by a different companies and are interconnected at peering points. It is really a network of networks spread across the globe, all of which are connected to each other. This super network is a glorified WAN in many respects. It connects many smaller networks together and allows all the computers to exchange information with each other through a common set of rules for communication. These rules are called protocols and the internet uses Transmission Control Protocol/Internet Protocol (TCP/IP). Programs such as web browsers, File Transfer Protocol (FTP) clients, and email clients are some of the most common ways through which the users work on the Internet. With the meteoric rise in demand for connectivity, the Internet has become a communications highway for millions of users. The Internet was initially restricted to military and academic institutions, but now it is a full-fledged conduit for any and all forms of information and commerce. Internet websites now provide personal, educational, political and economic resources to every corner of the planet.

Inter Space:

It is a client/server software program that allows multiple users to communicate online with real time audio, video and text chat in dynamic 3D environments. It provides the most advanced form of communication technology available today. It is a vision of what internet will become tomorrow. The users will be able to communicate in multiple ways and from multiple sources instantly.

SWITCHING TECHNIQUES:

Switching techniques are used for transmitting data across networks.

Different types are :

1. Circuit Switching
2. Message Switching
3. Packet Switching

Circuit Switching

- Circuit switching is the transmission technology that has been used since the first communication networks in the nineteenth century.
- First the complete physical connection between two computers is established and then the data are transmitted from the source computer to the destination.
- When a call is placed the switching equipment within the system seeks out a physical copper path all the way from the sender to the receiver.

- It is must to setup an end-to-end connection between computers before any data can be sent.
- The circuit is terminated when the connection is closed.
- In circuit switching, resources remain allocated during the full length of a communication, after a circuit is established and until the circuit is terminated and the allocated resources are freed.

Message Switching

- In this the source computer sends data or the message to the switching circuit which stores the data in its buffer.
- Then using any free link to the switching circuit the data is send to the switching circuit.
- Entire message is sent to the destination. It reaches through different intermediate nodes following the “store and forward” approach.
- No dedicated connection is required.

Packet Switching

- Conceived in the 1960's, packet switching is a more recent technology than circuit switching.
- Packet switching introduces the idea of cutting data i.e. at the source entire message is broken in smaller pieces called packets which are transmitted over a network without any resource being allocated.
- Then each packet is transmitted and each packet may follow any rout available and at destination packets may reach in random order.
- If no data is available at the sender at some point during a communication, then no packet is transmitted over the network and no resources are wasted.
- At the destination when all packets are received they are merged to form the original message.
- In packet switching all the packets of fixed size are stored in main memory.



DATA COMMUNICATION TERMINOLOGIES:

Before we proceed with our study of networks, let us learn about some data communication terminologies being used.

Channel: A communication channel is a medium that is used in the transmission of a message from one point to another. In simple terms we can say that it is a pathway over which data is transferred between remote devices. It may refer to the entire physical medium, such as a telephone line, optical fibre, coaxial cable or

twisted pair wire, or, it may refer to one of the several carrier frequencies transmitted simultaneously within the line.

Depending on their speed, we have three broad categories of communication channels - narrow band which is slow and used for telegraph lines and low speed terminals; voice band used for ordinary telephone communication and broad band which is fastest and is used for transmitting large volumes of data at high speeds.

Bandwidth: In electronic communication, bandwidth refers to the range of frequencies available for transmission of data. It is expressed as the difference in Hertz(Hz) between the highest frequency and the lowest frequency. For example, a typical voice signal has a bandwidth of approximately 3KHz. Wider the bandwidth of a communication system, greater is the capacity and hence greater is the amount of data that can be transmitted over a period of time.

In computer networking, bandwidth is often used as a synonym for data transfer rate about which you will read in the next subtopic.

Data channel	• The information / data carry from one end to another in the network by channel.
Baud & bits per second (bps)	• It's used to measurement for the information carry of a communication channel. • Measurement Units :- • bit • 1 Byte= 8 bits • 1 KBPS (Kilo Byte Per Second)= 1024 Bytes , • 1 Kbps (kilobits Per Second) = 1024 bits, • 1 Mbps (Mega bits Per Second)=1024 Kbps
Bandwidth	• It is amount of information transmitted or receives per unit time. • It is measuring in Kbps/Mbps etc unit.

TRANSMISSION MEDIA:

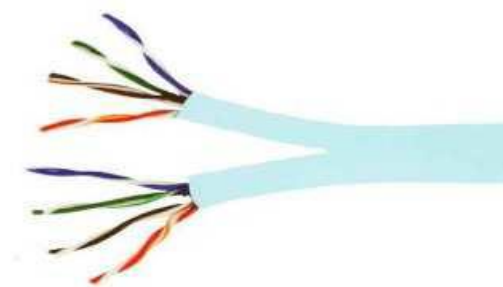
A transmission medium (plural media) is one which carries a signal from one computer to another. It is also known as communication channel. Transmission medium can be wired or wireless. We also name them as Guided and Unguided Media respectively. Wired transmission media includes twisted pair cable, Ethernet cable, coaxial cable and optical fibre whereas wireless transmission media includes microwave, radio wave, satellite, infrared, Bluetooth, WIFI etc.

Wired transmission media:

The wired or guided transmission media physically connects the two computers. The data signal physically gets transferred from the transmitting computer to the receiving computer through the wired transmission medium. Some of the wired transmission media are discussed below:

1. Twisted Pair Cable

- These cables consist of two insulated copper wires twisted around each other in a double helix.
- Twisting of wires reduces crosstalk which is bleeding of a signal from one wire to another.



Types:

- Unshielded Twisted Pair (UTP)
- Shielded Twisted Pair (STP)

STP offers greater protection from interference and crosstalk due to shielding. But it is heavier and costlier than UTP.

USE:

- In local telephone communication
- For digital data transmission over short distances upto 1 km

Advantages:

- Easy to install and maintain
- Simple
- Inexpensive
- Low weight
- Suitable for small (Local) Networks

Disadvantages:

- Not suitable for long distance due to high attenuation.
- Low bandwidth support.
- Low Speed

2. Coaxial cable:

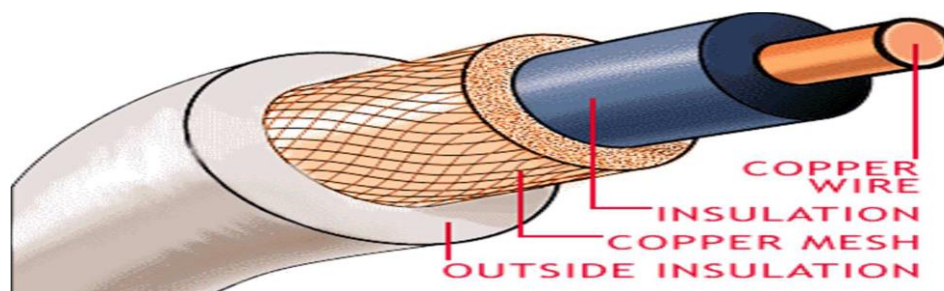
- Coaxial cable consists of a solid copper wire core surrounded by plastic cladding shielded in a wire mesh.
- Shield prevents the noise by redirecting it to ground.

Types:

Coaxial cable comes in two sizes which are called thinnet and thicknet.

- Thicknet : segment length upto 500 m
- Thinnet : segment length upto 185 m

USE: In TV channel communication



Advantages:

- Better than twisted wire cable.
- Popular for TV networks.
- Offers higher bandwidth & Speed

Disadvantages:

- Expensive than twisted wires.
- Not compatible with twisted wire cable.

3. Optical Fibres:

- Thin strands of glass or glass like material designed to carry light from one source to another.
- Source converts (Modulates) the data signal into light using LED (Light Emitting Diodes) or LASER diodes and send it over the Optical fiber.

It consists of three parts:

1. The core: glass or plastic through which the light travels.
2. The cladding : covers the core and reflects light back to the core
3. Protective coating : protects the fiber

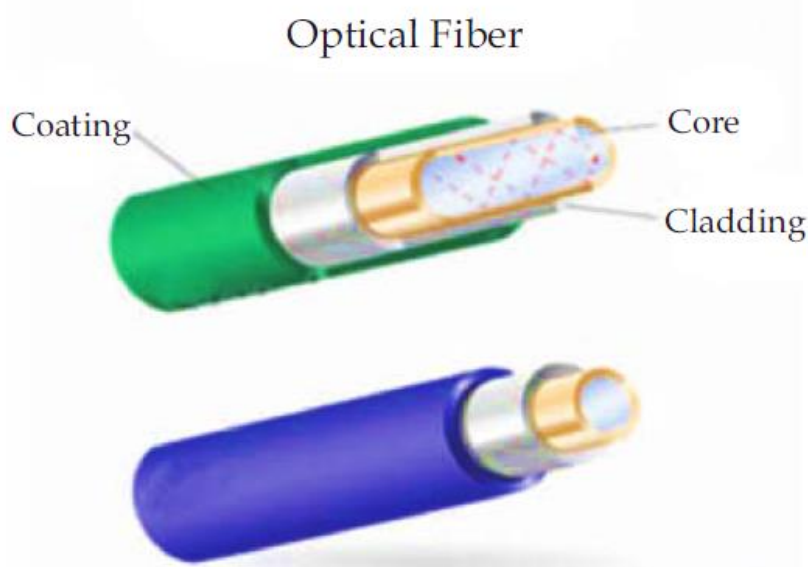
Advantages

- Not affected by any kind of noise.

- High transmission capacity
- Speed of Light
- Suitable for broadband communication

Disadvantages

- Installation requires care.
- Connecting two Optical fibers is difficult.
- Optical fibers are more difficult to solder
- Most expensive

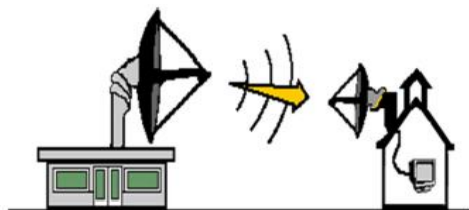


Wireless Transmission Media:

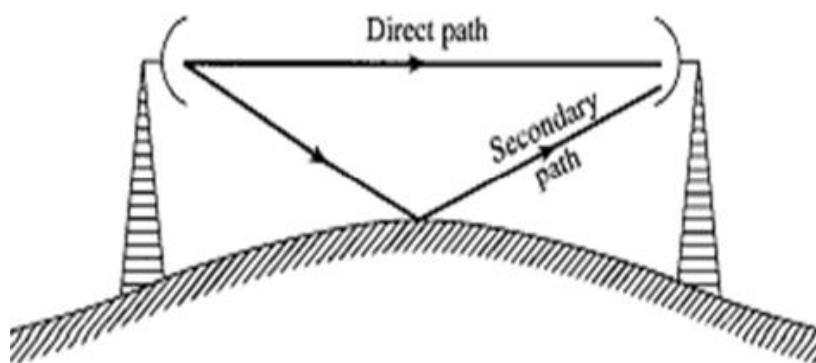
Wireless or unbounded or unguided media transport electromagnetic waves without using a physical conductor. The signals are broadcasted through air or water and thus are available to anyone that has a device capable of receiving them. Some of the wireless media are:

Microwaves:

Microwaves are transmitted from the transmitters placed at very high towers to the receivers at a long distance.



Microwaves are transmitted in line of sight fashion, and also propagated through the surfaces.



Advantages:

- Maintenance easy than cables.
- Suitable when cable cannot be used.

Disadvantages:

- Repeaters are required for long distance communication.
- Less Bandwidth available.

Satellite:

Geostationary satellites are placed around 36000 KM away from the earth's surface. In satellite communication transmitting station transmits the signals to the satellite. (It is called up-linking). After receiving the signals (microwaves) it amplifies them and transmit back to earth in whole visibility area. Receiving stations at different places can receive these signals. (It is called down-linking).



Advantage

- Area coverage is too large

Disadvantage

- High investment

Network devices:



For efficient working of any network, many devices are required. Some of the common network devices are discussed below:

Modem

- A modem is a computer peripheral that allows you to connect and communicate with other computers via telephone lines.
- Modem means Modulation/ Demodulation.
- Modulation: A modem changes the digital data from your computer into analog data, a format that can be carried by telephone lines.
- Demodulation: The modem receiving the call then changes the analog signal back into digital data that the computer can digest.
- The shift of digital data into analog data and back again, allows two computers to speak with one another.



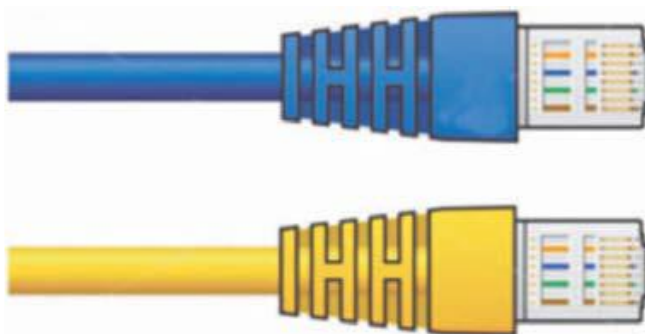
External Modem



Internal Modem

RJ- 45 Connector:

RJ-45 is short for Registered Jack-45. It is an eight wire connector which is commonly used to connect computers on the local area networks i.e., LAN.



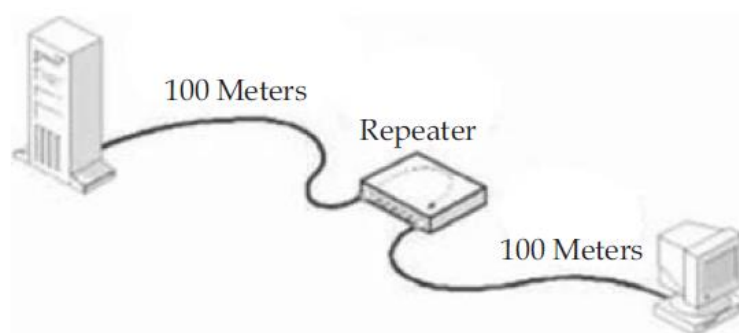
Network Interface Cards (Ethernet Card):

A network card, network adapter or NIC (network interface card) is a piece of computer hardware designed to allow computers to communicate over a computer network. It provides physical access to a networking medium and often provides a low-level addressing system through the use of MAC addresses. It allows users to connect to each other either by using cables or wirelessly.



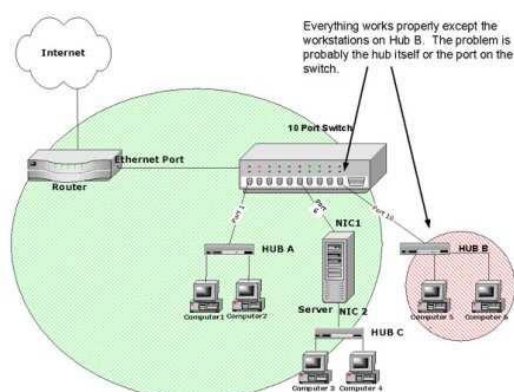
Repeaters:

A repeater is an electronic device that receives a signal and retransmits it at a higher level or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances without degradation. In most twisted pair Ethernet configurations, repeaters are required for cable runs longer than 100 meters.



Hubs:

A hub contains multiple ports. When a packet arrives at one port, it is copied to all the ports of the hub. When the packets are copied, the destination address in the frame does not change to a broadcast address. It does this in a rudimentary way, it simply copies the data to all of the Nodes connected to the hub.



Bridges:

A network bridge connects multiple network segments at the data link layer (layer 2) of the OSI model. Bridges do not promiscuously copy traffic to all ports, as hubs

do, but learn which MAC addresses are reachable through specific ports. Once the bridge associates a port and an address, it will send traffic for that address only to that port. Bridges do send broadcasts to all ports except the one on which the broadcast was received.

- Bridges learn the association of ports and addresses by examining the source address of frames that it sees on various ports. Once a frame arrives through a port, its source address is stored and the bridge assumes that MAC address is associated with that port. The first time that a previously unknown destination address is seen, the bridge will forward the frame to all ports other than the one on which the frame arrived.
- Bridges come in three basic types:
 - Local bridges: Directly connect local area networks (LANs)
 - Wireless bridges: Can be used to join LANs or connect remote stations to LANs.



Switches:

Switch is a device that performs switching. Specifically, it forwards and filters OSI layer 2 datagrams (chunk of data communication) between ports (connected cables) based on the Mac- Addresses in the packets. This is distinct from a hub in that it only forwards the datagrams to the ports involved in the communications rather than all ports connected. Strictly speaking, a switch is not capable of routing traffic based on IP address (layer 3) which is necessary for communicating between network segments or within a large or complex LAN. Some switches are capable of routing based on IP addresses but are still called switches as a marketing term. A switch normally has numerous ports with the intention that most or all of the network be connected directly to a switch, or another switch that is in turn connected to a switch.



Routers:

- • Routers are networking devices that forward data packets between networks using headers and forwarding tables to determine the best path to forward the

packets. Routers work at the network layer of the TCP/IP model or layer 3 of the OSI model. Routers also provide interconnectivity between like and unlike media (RFC 1812).

- A router is connected to at least two networks, commonly two LANs or WANs or a LAN and its ISP's network.

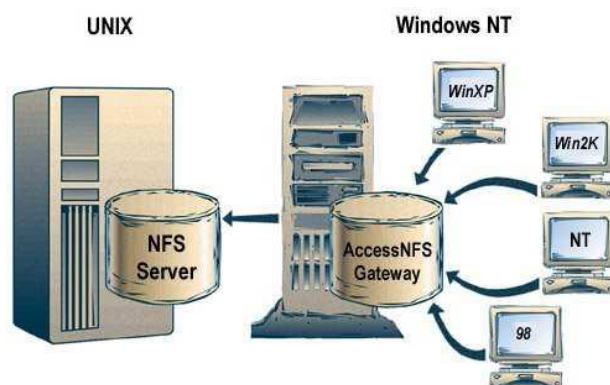
Gateway:

A Gateway is a network device that connects dissimilar networks. It established an intelligent connection between a local area network and external networks with completely different structures.

The computers that control traffic within your company's network or at your local Internet Service Provider (ISP) are gateway nodes. A network gateway can be implemented completely in software, completely in hardware, or as a combination of both. In the network for an enterprise, a computer server acting as a gateway node is often also acting as a proxy server and a firewall server. Here a proxy server is a node that is not actually a server but just appears to be so and a firewall is a system designed to prevent unauthorized access to or from a private network. A gateway is often associated with both a router, which knows where to direct a given packet of data that arrives at the gateway, and a switch, which furnishes the actual path in and out of the gateway for a given packet. It expands the functionality of the router by performing data translation and protocol conversion.

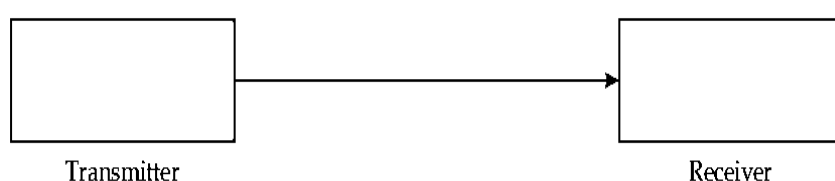
You will sometimes see the term default gateway on network configuration screens in Microsoft Windows.

In computer networking, a default gateway is the device that passes traffic from the local subnet to devices on other subnets. The default gateway often connects a local network to the Internet, although internal gateways for local networks also exist.



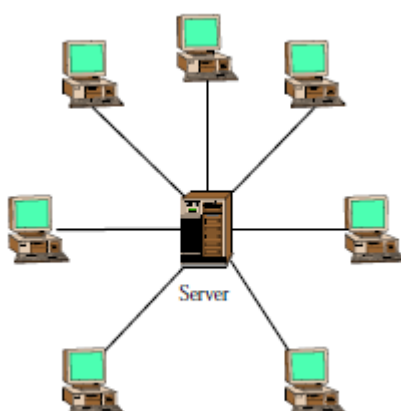
Network topologies and types:

Topology is the pattern of interconnection of nodes in a local area network(LAN). The topology used helps to select the communication medium and the other network devices. While choosing a topology, care has to be taken that the installation cost is minimum, the network so designed should be reliable and flexible. In simple terms the addition or reduction of nodes should be easy and also fault detection and removal should be simple. Before we talk about topologies in detail, let us learn about point to point link which has two ends transmitter and receiver. The main characteristic of Point to Point link is that each transmitter transmits to exactly one receiver and each receiver receives exactly form one transmitter. The transmission might occur on a single medium i.e. single wire or over separate wires.



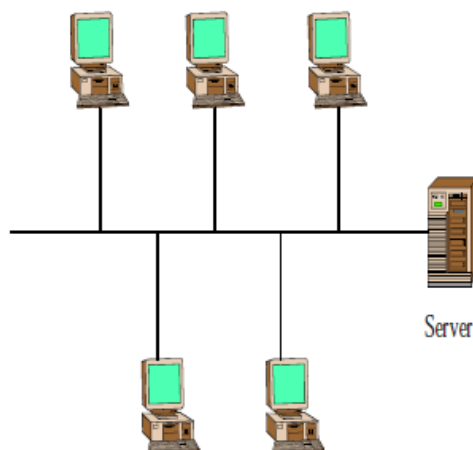
Star Topology:

The type of network topology in which each of the nodes of the network is connected to a central node with a point-to-point link in a 'hub' and 'spoke' fashion, the central node being the 'hub' and the nodes that are attached to the central node being the 'spokes' (e.g., a collection of point-to-point links from the peripheral nodes that converge at a central node) – all data that is transmitted between nodes in the network is transmitted to this central node, which is usually some type of device that then retransmits the data to some or all of the other nodes in the network, although the central node may also be a simple common connection point (such as a 'punch- down' block) without any active device to repeat the signals.



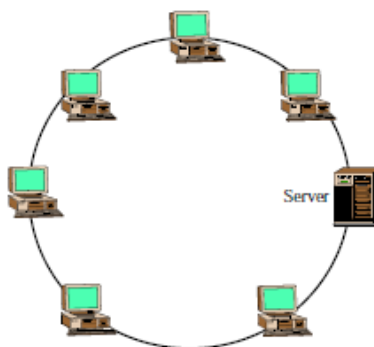
Bus Topology:

The type of network topology in which all of the nodes of the network are connected to a common transmission medium which has exactly two endpoints (this is the 'bus', which is also commonly referred to as the backbone, or trunk) – all data that is transmitted between nodes in the network is transmitted over this common transmission medium and is able to be received by all nodes in the network virtually simultaneously (disregarding propagation delays).



Ring Topology:

The type of network topology in which each of the nodes of the network is connected to two other nodes in the network and with the first and last nodes being connected to each other, forming a ring – all data that is transmitted between nodes in the network travels from one node to the next node in a circular manner and the data generally flows in a single direction only.



Computer Networks:

A communications network is two or more computers connected to share data and resources are “networked.” The simple idea behind computer networking is to allow users to access more information and give them access to devices not directly attached to their “local” system, such as printers or storage devices.

Local Area Network (LAN):

A network covering a small geographic area, like a home, office, or building. Current LANs are most likely to be based on Ethernet technology. For example, a library will have a wired or a communications network is two or more computers connected to share data and resources are “networked.” The simple idea behind computer networking is to allow users to access more information and give them access to devices not directly attached to their “local” system, such as printers or storage devices.

Metropolitan Area Network (MAN):

A Metropolitan Area Network is a network that connects two or more Local Area Networks or Campus Area Networks together but does not extend beyond the boundaries of the immediate town, city, or metropolitan area. Multiple routers, switches & hubs are connected to create a MAN.

Wide Area Network (WAN):

WAN is a data communications network that covers a relatively broad geographic area(i.e. one city to another and one country to another country) and that often uses transmission facilities provided by common carriers, such as telephone companies.

WAN technologies generally function at the lower three layers of the OSI reference model: the physical layer, the data link layer, and the network layer.

Network Protocols:

In information technology, a protocol is the special set of rules that two or more machines on a network follow to communicate with each other. They are the standards that allow computers to communicate. A protocol defines how computers identify one another on a network, the form that the data should take in transit, and how this information is processed once it reaches its final destination. A protocol is needed every time we want to perform any task on a network. It may be transferring data or taking a printout on a network printer or accessing the central database.

Although each network protocol is different, they all share the same physical cabling. This common method of accessing the physical network allows multiple protocols to peacefully coexist over the network media, and allows the builder of a network to use common hardware for a variety of protocols. This concept is known as "Protocol Independence." Thus it is clear that:

- A protocol means the rules that are applicable for a network.
- It defines the standardized format for data packets, techniques for detecting and correcting errors and so on.
- A protocol is a formal description of message formats and the rules that two or more machines must follow to exchange those messages. E.g. using library books.

Types of protocols are:

1. HTTP
2. FTP
3. TCP/IP
4. SLIP/PPP

Hypertext Transfer Protocol (HTTP) is a communications protocol for the transfer of information on the intranet and the World Wide Web. Its original purpose was to provide a way to publish and retrieve hypertext pages over the Internet.

HTTP is a request/response standard between a client and a server. A client is the end-user; the server is the web site.

FTP (File Transfer Protocol) is the simplest and most secure way to exchange files over the Internet. The objectives of FTP are:

- To promote sharing of files (computer programs and/or data).
- To encourage indirect or implicit use of remote computers.
- To shield a user from variations in file storage systems among different hosts.
- To transfer data reliably, and efficiently.

TCP/IP (Transmission Control Protocol / Internet Protocol):

The Transmission Control Protocol(TCP) breaks the data into packets that the network can handle efficiently. It manages the assembling of a message or file into smaller packets that are transmitted over the Internet. It verifies all the packets when they arrive at the destination computer and then reassembles them in proper order. Data can be lost in the intermediate network. So TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.

The Internet Protocol(IP)handles the address part of each packet so that it reaches to the right destination. It gives distinct address (called IP address) to each data packet. Each gateway computer on the network checks this address to see where to forward the message. Even though some packets from the same message are routed differently than others, they'll be reassembled at the destination. An IP address is a unique identifier for a node or host connection on an IP network. An IP address is a 32 bit binary number usually represented as 4 decimal values, each

representing 8 bits, in the range 0 to 255 (known as octets) separated by decimal points. This is known as "dotted decimal" notation.

Example:

140.179.220.200

The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world. TCP/IP uses the client/server mode of communication in which a computer user (a client) makes a request and the server provides the requested service such as sending a Web page. Also TCP/IP communication is primarily point-to-point transmission of data which means each communication is from one computer in the network to another computer. TCP/IP and the higher-level applications that use it are collectively said to be "stateless" because each client request is considered a new request unrelated to any previous one. Till the time complete message or all packets in a message have been delivered, the connection between the two computers remains intact but after that the transmission path is available freely. So unlike ordinary phone conversations that require a dedicated connection for the entire call duration, no dedicated connection is required. This makes the network paths freely available for everyone to use.

SLIP/PPP (Serial Line Internet Protocol / Point to Point Protocol)

SLIP/PPP provides the ability to transport TCP/IP traffic over serial line between two computers. The home user's computer has a communications link to the internet. The home user's computer has the networking software that can speak TCP/IP with other computers on the Internet. The home user's computer has an identifying address (IP address) at which it can be contacted by other computers on Internet. E.g. dial up connection.

SMTP :- SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end, it is usually used with one of two other protocols, POP3 or IMAP, that let the user save messages in a server mailbox and download them periodically from the server. In other words, users typically use a program that uses SMTP for sending e-mail and either POP3 or IMAP for receiving e-mail. On Unix-based systems, send mail is the most widely used SMTP server for e-mail. A commercial package, Send mail, includes a POP3 server. Microsoft Exchange includes an SMTP server and can also be set up to include POP3 support.

POP3:- POP3 (Post Office Protocol 3) is the standard way which has been around for decades. It is very similar regular mail. Messages are delivered to our computer, put in our mailbox, and are then our responsibility.

Advantage of POP3:

1. Email is available when we are offline
2. Email is not stored on the server, so our disk usage on the server is less
3. Just about any email client (software) supports POP3

IMAP:- IMAP (Interactive Mail Access Protocol).It is not like the mailbox on your house. With IMAP mail is delivered to the server, and we connect to the server to see our mail. The mail is not stored on our machine. When a message is marked as read, it is marked as read on the server, not on our computer.

Advantages of IMAP:-

1. Email is available from any machine we happen to use
2. Email is stored on the server, so our email cannot be deleted/destroyed if our computer should happen to crash, be stolen, or destroyed
3. We can access IMAP mail via the web, without even needing a mail client installed. This means We can check our mail from someone else's machine or even a public terminal and not have to worry about the security of your passwords.

Telnet- It is an older internet utility that lets us log on to remote computer system. It also facilitates for terminal emulation purpose. Terminal emulation means using a pc like a mainframe computer through networking.

1. Run telnet client- Type telnet in run dialog box.
2. Connect to telnet site -specify the host name, port and terminal type.
3. Start browsing- surf the shown site with provided instruction.
4. Finally disconnect-press Alt+F4.

Wireless/Mobile Computing

Wireless communication is simply data communication without the use of landlines. Mobile computing means that the computing device is not continuously connected to the base or central network.

1. **GSM(Global System for Mobile communication):** it is leading digital cellular system. In covered areas, cell phone users can buy one phone that will work anywhere the standard is supported. It uses narrowband TDMA, which allows eight simultaneous calls on the same radio frequency.
2. **CDMA(Code Division Multiple Access):** it is a digital cellular technology that uses spread-spectrum techniques. CDMA does not assign a specific

frequency to each user. Instead ,every channel uses the full available spectrum.

3. **WLL(Wireless in Local Loop) :** WLL is a system that connects subscribers to the public switched telephone network using radio signals as a substitute for other connecting media.
4. **Email(Electronic Mail):** Email is sending and receiving messages by computer.
5. **Chat:** Online textual talk in real time , is called Chatting.
6. **Video Conferencing:** a two way videophone conversation among multiple participants is called video conferencing.
7. **SMS(Short Message Service):** SMS is the transmission of short text messages to and from a mobile phone, fax machine and or IP address.
8. **3G and EDGE:** 3G is a specification for the third generation of mobile communication of mobile communication technology. 3G promises increased bandwidth, up to 384 Kbps when a device is stationary.
9. **EDGE(Enhanced Data rates for Global Evolution)** is a radio based high speed mobile data standard.
10. **Wi-Fi---**Wi-Fi is short for Wireless Fidelity . Wi-Fi is a wireless technology that uses radiofrequency to transmit data through the air. An Access Point is a separate wireless unit, which has the ability to extend from the router to maximize wireless reception. Up to 16 users can connect to one access point .It allows several wireless clients to connect to a single device.Wi-fi hot spot is defined as any location in which wireless technology both exists and is available for use to consumers. Wi-fi was intended to be used for wireless devices and LANS, but is now often used for internet access.
11. **IPR:-** The term "Intellectual Property Rights" refers to the legal rights granted with the aim to protect the creations of the intellect. These rights include Industrial Property Rights (e.g. patents, industrial designs and trademarks) and Copyright (right of the author or creator) and Related Rights (rights of the performers, producers and broadcasting organizations
12. **Cloud Computing :-** The practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer.

Advantages:-

- 1) Cost efficient
- 2) Backup & Restore
- 3) Almost unlimited storage
- 4) Automatic Software Integration
- 5) Easy access of Information

Disadvantages:-

- 1) Technical Issue
- 2) Security in the cloud
- 3) Prone to attack



NETWORK SECURITY CONCEPTS:

Protection methods:

1. Authorization - Authorization confirms the service requestor's credentials. It determines if the service requestor is entitled to perform that operation.
2. Authentication - Each entity involved in using a web service the requestor, the provider and the broker(if there is one) - is what it actually claims to be.
3. Encryption – conversion of the form of data from one form to another form.
4. Biometric System - involves unique aspect of a person's body such as Fingerprints, retinal patterns etc to establish his/her Identity.
5. Firewall - A system designed to prevent unauthorized access to or from a private network is called firewall. it can be implemented in both hardware and software or combination or both.

There are several types of firewall techniques-

- Packet filter- accepts or rejects of packets based on user defined rules.
- Application gateway- security mechanism to specific application like FTP and Telnet servers.
- Circuit level gateway - applies security mechanism when a connection is established.
- Proxy Server - Intercepts all messages entering and leaving the network.
- Cookies - Cookies are messages that a web server transmits to a web browser so that the web server can keep track of the user's activity on a specific web site. Cookies have few parameters name, value, expiration date.

Hackers and crackers :

Hackers are more interested in gaining knowledge about computer systems and possibly using this knowledge for playful pranks.

Crackers are the malicious programmers who break into secure systems.

Cyber Law -

It is a generic term, which refers to all the legal and regulatory aspects of internet and the World Wide Web.

Network Security Concepts

Network security deals with policies adopted by network administrator to protect the network from unauthorized access and misuse of network resources. It also ensures that the authorized users have adequate access to all the network resources.

Virus

If you observe that your system

- takes longer time to load applications
- shows unpredictable program behaviour
- shows inexplicable changes in file sizes
- has inability to boot,
- has strange graphics appearing on your screen

This could be because of your computer being infected by a virus.

Virus is a malicious program that attaches itself to the host program. It is designed to infect the host program and gain control over the system without the owner's knowledge. The virus gets executed each time the host program is executed. Also it has the tendency to replicate. They can spread through external media such as CDs, browsing infected internet sites and from email attachments.

Types of Viruses:

- **File Virus:** These viruses infect and replicate when it gets attached to MS-DOS program files with EXE or COM extensions.
- **Boot sector virus:** These viruses infect the boot sector of floppy disks or hard drives. Boot sector of a drive contains program that participates in booting the system. A virus can infect the system by replacing or attaching itself to these programs
- **Macro virus:** These viruses infect and replicate using the MS Office program suite, mainly MS Word and MS Excel. The virus inserts unwanted words or phrases in the document.

Worm:

Worm is also a malicious program like a virus. But unlike viruses, it does not need to attach itself to a host program. A worm works by itself as an independent object. It uses security holes in a computer networks to replicate itself. A copy of the worm scans the network for another machine that has a specific security hole. It copies itself to the new machine using the security hole, and then starts replicating from there, as well.

Trojan horse:

A Trojan horse is a program that contains hidden malicious functions. Trojan Horses trick users into installing them by appearing to be legitimate programs. Once installed on a system, they reveal their true nature and cause damage. Some Trojan horses will contact a central server and report back information such as passwords, user IDs, and captured keystrokes. Trojans lack a replication routine and thus are not viruses by definition.

Spam:

The term spam means endless repetition of worthless text. In other words, unwanted messages or mails are known as Spam. At times internet is flooded with multiple copies of the same message, it is nothing but spam. Most spam is commercial advertising. In addition to wasting people's time, spam also eats up a lot of network bandwidth.

Cookies:

When the user browses a website, the web server sends a text file to the web browser. This small text file is a cookie. Generally a cookie contains the name of the website from which it has come from and a unique ID tag.

Some cookies last only until the browser is closed. They are not stored on your hard drive. They are usually used to track the pages that you visit so that information can be customized for you for that visit. On the other hand, some cookies are stored on your hard drive until you delete them or they reach their expiry date. These may, for example, be used to remember your preferences when you use the website.

Firewall:

A firewall is hardware or software based network security system. It prevents unauthorized access (hackers, viruses, worms etc.) to or from a network.

Firewalls are used to prevent unauthorized internet users to access private networks connected to the Internet. All data entering or leaving the Intranet pass through the firewall, which examines each packet and blocks those that do not meet the specified security criteria.

A firewall examines all traffic routed between the two networks to see if it meets certain criteria. If it does, it is routed between the networks, otherwise it is stopped. A firewall filters both inbound and outbound traffic. A firewall may allow all traffic through unless it meets certain criteria, or it may deny all traffic unless it meets certain criteria.

Cyber Crime:

Cybercrime is defined as a crime in which a computer and internet is used in an illegitimate way to harm the user. Cyber criminals may use computer technology to access personal information, business trade secrets, or use the internet for exploitive or malicious purposes. Cybercrimes can be against persons or against property or against the government.

The list of Cyber Crimes includes:

- harassment by computer (Cyber Stalking, defamation)
- pornography
- illegal downloads, plagiarism
- software piracy/counterfeiting, copyright violation of software, counterfeit hardware, black market sales of hardware and software, theft of equipment and new technologies
- fraud (credit card fraud, fraudulent use of ATM accounts, stock market transfers, telecommunications fraud), theft of (electronic) money

Cyber Law:

Cyber law is an attempt to integrate the challenges presented by human activity on the internet with legal system of laws applicable to the physical world.

There was no statute in India for governing Cyber Laws involving privacy issues, jurisdiction issues, intellectual property rights issues and a number of other legal questions. With the tendency of misusing of technology, there has arisen a need of strict statutory laws to regulate the criminal activities in the cyber world and to protect the true sense of technology. "INFORMATION TECHNOLOGY ACT, 2000" [ITA-2000] was enacted by Parliament of India to protect the field of e-commerce, e-governance, e-banking as well as penalties and punishments in the field of Cyber Crimes. The above Act was further amended in the form of IT Amendment Act, 2008 [ITAA-2008].

In the IT Act the word 'computer' and 'computer system' have been so widely defined and interpreted to mean any electronic device with data processing capability, performing computer functions like logical, arithmetic and memory functions with input, storage and output capabilities and therefore any high-end programmable gadgets like a washing machine or switches and routers used in a network can all be brought under the definition.

Some of the CYBER OFFENCES UNDER THE IT ACT:

- Tampering with computer source documents - Section 65
- Hacking -Section 66

- Publishing of information which is obscene in electronic form -Section 67

Intellectual property rights (IPR) Issues:

Intellectual property rights are the rights given to an individual over the invention of their own. They usually give the creator an exclusive right over the use of his/her creation for a certain period of time.

There are only three ways to protect intellectual property:

1. Patents:

A Patent is a term used for a specific product designed by an individual. The designer is given exclusive rights over the patent for a limited period of time. With help of the patent right, the owner can stop others from making, using or selling the product design. The owner can take a legal action if someone uses the patent without his/ her permission.

In order to obtain a patent, the following conditions should be met:

- ✚ The product should be new
- ✚ It should be capable of being made or used in some kind of industry
- ✚ It should not be a scientific or mathematical discovery
- ✚ It should not be a dramatic, musical dramatic or artistic work

2.Trademarks

Trademark can be defined as a name or a different sign or a device identifying a product or a service. The product or the service is produced or provided by a specific person or a company. A Trademark is also known as brand name. It should be officially registered and legally restricted to the use of the specific person or the company.

3. Copyrights

Copyright is the term used for a written document. A legal action can be taken, if copyrights are violated. The following category of work can be considered for copyrights.

- ✚ literary works
- ✚ musical works, including any accompanying words
- ✚ dramatic works, including any accompanying music
- ✚ pantomimes and choreographic works
- ✚ pictorial, graphic and sculptural works
- ✚ motion pictures and other audio visual works
- ✚ sound recordings

- ✚ architectural works
- ✚ computer programs and websites

Hacking:

The term hacking was first used at M.I.T during 1950s and 1960s. The term was used for people who engaged themselves in harmless technical experiments and fun learning activities.

A computer enthusiast, who uses his computer programming skills to intentionally access a computer without authorization is known as hacking. The computer enthusiast involved in this activity is known as a hacker. A hacker accesses the computer without the intention of destroying data or maliciously harming the computer.

Another term commonly used with hacking is cracking. Cracking can be defined as a method by which a person who gains unauthorized access to a computer with the intention of causing damage.

Book References

1. C++ By E Balagurusamy
2. IGNOU Study Material
3. MCGRAW Abraham Silberschatz, Henry F. Korth, S. Sudarshan
4. Ronald J. Tocci Neal S. Widmer

QR Code References

1. E- Content developed by Sajad Akbar Rather (I/C ICT DIET SOPORE)
2. E- Content developed by Javeed Hassan Sofi (Tr. GMS Brar Bandipora)
3. [www. easytuts4you.com](http://www.easytuts4you.com)
4. SuccessCDs (<http://youtu.be/Kelm63Q-LO8>)
5. Knowledge Gate (<http://youtu.be/C-CzmtqMVfo/>)
6. <http://youtu.be/SJXTjSqjcLg>
