

# **West Bengal State University**

***Registration No -1071711100060 of 2017***

***Roll -3201134 No -19162***

***Sub - C++ Practical***

***Part -III***

***Paper- CMSA VIII***

## **Index**

<b>ID X No</b>	<b>Assignment Name</b>	<b>Pages</b>	<b>Date</b>	<b>Signature</b>
1	Implementation of Fibonacci Series	4-5	05/11/2019	
2	Area and Circumference of a Circle.	6-7	05/11/2019	
3	Area of Triangle using function overloading.	8-9	05/11/2019	
4	Implementation of Constructor Overloading.	10-11	05/11/2019	
5	Implementation of Stack Operation.	13-19	05/11/2019	
6	Implementation of Queue Operation.	20-25	11/11/2019	
7	Array insertion deletion with linear search and binary search.	26-34	11/11/2019	
8	Implementation of Linked list	35-43	11/11/2019	
9	Addition of two complex number using Friend function.	44-47	11/11/2019	
10	Addition and Subtraction using operator overloading.	48-51	11/11/2019	
11	Implementation of Multilevel Inheritance	52-54	25/11/2019	
12	Implementation of String operation	55--56	18/11/2019	
13	Implementation of Bubble sort , Insertion sort and Selection sort.	57-62	18/11/2019	
14	Implementation of Binary Search Tree	63-73	18/11/2019	
15	Implementation of Simple Inheritance.	74-76	18/11/2019	
16	Implementation of Multiple Inheritance	77-80	25/11/2019	

# 1 . Implementation of Fibonacci Series

## Algorithm :

- ALGORITHM FOR CLASS "cl":  
Step:1 start  
Step:2 declare the following data member : n1, n2, l, f in integer  
Step:3 declare the following member function : void input(), void fib()  
Step:4 end of class "cl"
- ALGORITHM FOR FUNCTION "VOID INPUT()":  
Step:1 start  
Step:2 set n1=0, n2=1  
Step:3 print 'enter the limit'  
Step:4 print l  
Step:5 end
- ALGORITHM FOR FUNCTION "VOID FIB()":  
Step:1 start  
Step:2 print n1 and n2  
Step:3 for (i=2, i<1, i++)  
    Set f= n1+n2  
    Print f " "  
    Set n1=n2  
    Set n2=f  
End
- ALGORITHM FOR "MAIN" FUNCTION:  
Step:1 start  
Step:2 declare object 'ob' for 'cl'  
Step:3 call function input()  
Step:4 call function fib()  
Step:5 end of class main

## Program Code :

```
#include<iostream>
#include<conio.h>
using namespace std;
class cl
{
    int n1,n2,l,f;
    public :
        void input();
        void fib();
};
void cl::input()
{
    n1=0;n2=1;
    cout<<"Enter the limit"<<endl;
    cin>>l;
}
```

```

void cl::fib()
{
    cout<<n1<<" "<<n2<<" ";
    for(int i=2;i<l;i++)
    {
        f=n1+n2;
        cout<<f<<" ";
        n1=n2;
        n2=f;
    }
}
int main()
{
    cl ob;
    ob.input();
    ob.fib();
    return 0;
}

```

### **Output:**

Enter the limit

10

0 1 1 2 3 5 8 13 21 34

### **Discussion :**

The Fibonacci numbers are the numbers in the following integer sequence.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

The iostream header file helps to use the 'cin' & 'cout' library functions.

## 2 . Area and Circumference of a Circle.

### Algorithm :

ALGORITHM FOR CLASS "cl":

Step:1 start

Step:2 declare the following data member : r, A, C in integer

Step:3 declare the following member function : void input(), void area(), void circumference(), void output()

- ALGORITHM FOR FUNCTION "VOID INPUT()":

Step:1 start

Step:2 print 'enter the radius'

Step:3 print r

Step:4 end

- ALGORITHM FOR FUNCTION "VOID AREA()":

Step:1 start

Step:2 set  $A = (22/7) * r * r$

Step:3 end

- ALGORITHM FOR FUNCTION "VOID CIRCUMPHERENCE()":

Step:1 start

Step:2 set  $C = 2 * (22/7) * r$

Step:3 end

- ALGORITHM FOR FUNCTION "VOID OUTPUT()":

Step:1 start

Step:2 print 'area of circle=A'

Step:3 print 'circumference of circle=C'

Step:4 end

Step:4 end of class "cl"

ALGORITHM FOR "MAIN" FUNCTION:

Step:1 start

Step:2 declare object 'ob' for 'cl'

Step:3 call function input()

Step:4 call function area()

Step:5 call function circumference()

Step:6 call function output()

Step:7 end of class main

### Program Code :

```
#include<iostream>
using namespace std;
class cl
{
    int r,A,C;
    public :
        void input()
        {
            cout <<"Enter the redious ";
            cin >>r;
```

```

    }
    void Area()
    {
        A=(22/7)*r*r;
    }
    void circumference()
    {
        C=2*(22/7)*r;
    }
    void output()
    {
        cout<<"Area of circle = "<<A<<endl;
        cout<<"Circumference of circle - "<<C<<endl;
    }
};
int main()
{
    cl ob;
    ob.input();
    ob.Area();
    ob.circumference();
    ob.output();
    return 0;
}

```

## Output:

Enter the radius 10.0

Area of circle = 300

Circumference of circle = 60

## Discussion:

The iostream header file helps to use the 'cin' & 'cout' library functions.

The value of  $\pi$  is taken as (22/7).

The object o is used to call all the member function of the class cl.

### 3 . Area of Triangle using function overloading.

#### Algorithm :

- ALGORITHM FOR CLASS 'triangle':

Step:1 start

Step:2 declare the following data member : area in double

Step:3 declare the following member function : void triangle\_area(double, double),

void triangle\_area(double, double, double)

Step:4 end of class 'cl'

- ALGORITHM FOR FUNCTION 'void triangle\_area(double h, double b)':

Step:1 start

Step:2 set  $\text{area} = 0.5 * h * b$

Step:3 print "The area of the triangle is : "area

Step:4 end

- ALGORITHM FOR FUNCTION 'void triangle\_area(double a, double b, double c)':

Step:1 start

Step:2 set  $s = (a + b + c) / 2$

Step:3 set  $t = s * (s - a) * (s - b) * (s - c)$

Step:4 set  $\text{area} = \sqrt{t}$

Step:5 print "The area of the triangle is : "area

Step:6 end

- ALGORITHM FOR 'MAIN' FUNCTION:

Step:1 start

Step:2 declare object 'ob' for 'triangle'

Step:3 declare c as int

Step:4 print "Enter 1 to calculate the area of a triangle from height and base"

Step:5 print "Enter 2 to calculate the area of a triangle from three sides"

Step:6 print "Enter your choice"

Step:7 print c

Step:8 if(c=1)

    Declare double h, b

    Print "Enter the height of the triangle"

    Print h

    Print "Enter the base of the triangle"

    Print b

    Call ob.triangle\_area(h, b)

    else if(c=2)

        declare double a, b, c

        print "Enter the 1st side of the triangle"

        print a

        print "Enter the 2nd side of the triangle"

        print b

        print "Enter the 3rd side of the triangle"

        print c

        call ob.triangle\_area(a, b, c)

    else

```
        print "Wrong Choice"
    end if
Step:9 end of class main
```

## Program Code :

```
#include<iostream>
#include<cmath>
using namespace std;
class triangle
{
    double area;
public:
    void triangle_area(double, double);
    void triangle_area(double, double, double);
};
void triangle::triangle_area(double h, double b)
{
    area=0.5*h*b;
    cout<<"The area of the triangle is : "<<area<<endl;
}
void triangle::triangle_area(double a, double b, double c)
{
    double s=(a+b+c)/2;
    double t=s*(s-a)*(s-b)*(s-c);
    area=sqrt(t);
    cout<<"The area of the triangle is : "<<area<<endl;
}
int main()
{
    triangle obj;
    int c;
    cout<<"Enter 1 to calculate the area of a triangle from height and
base"<<endl;
    cout<<"Enter 2 to calculate the area of a triangle from three
sides"<<endl;
    cout<<"Enter your choice"<<endl;
    cin>>c;
    if(c==1)
    {
        double h, b;
        cout<<"Enter the height of the triangle"<<endl;
        cin>>h;
        cout<<"Enter the base of the triangle"<<endl;
        cin>>b;
        obj.triangle_area(h, b);
    }
    else if(c==2)
    {
```



```

        double a, b, c;
        cout<<"Enter the 1st side of the triangle"<<endl;
        cin>>a;
        cout<<"Enter the 2nd side of the triangle"<<endl;
        cin>>b;
        cout<<"Enter the 3rd side of the triangle"<<endl;
        cin>>c;
        obj.triangle_area(a, b, c);
    }
    else
        cout<<"Wrong Choice"<<endl;
}

```

### **Output:**

Enter 1 to calculate the area of a triangle from height and base

Enter 2 to calculate the area of a triangle from three sides

Enter your choice

2

Enter the 1st side of the triangle

10

Enter the 2nd side of the triangle

9

Enter the 3rd side of the triangle

5

The area of the triangle is: 22.4499

### **Discussion :**

The concept of function overloading has been taken into account in this program.

The cmath header file is used to take help of sqrt() library function.

The iostream header file helps to use the 'cin' & 'cout' library functions.

## 4 . Implementation of Constructor Overloading.

### Algorithm :

- ALGORITHM FOR CLASS “cl”:  
Step:1 start  
Step:2 declare the following data member : a in integer  
Step:3 declare the following constructor cl(), cl(int x), cl(int x, int y)  
Step:4 declare the following member function : void output()  
Step:5 end of class “cl”
- ALGORITHM FOR DEFAULT CONSTRUCTOR ‘cl()’  
Step:1 start  
Step:2 set a=5  
Step:3 print “default constructor”  
Step:4 end
- ALGORITHM FOR PARAMETERIZED CONSTRUCTOR ‘cl(int x)’  
Step:1 start  
Step:2 set a=x  
Step:3 print “constructor with one parameter”  
Step:4 end
- ALGORITHM FOR PARAMETERIZED CONSTRUCTOR ‘cl(int x, int y)’  
Step:1 start  
Step:2 set a=x+y  
Step:3 print “constructor with two parameter”  
Step:4 end
- ALGORITHM FOR FUNCTION “VOID OUTPUT()”:  
Step:1 start  
Step:2 print ‘a’  
Step:3 end
- ALGORITHM FOR “MAIN” FUNCTION:  
Step:1 start  
Step:2 declare object for ‘cl’  
Step:3 call parameterized constructors  
Step:4 call function output()  
Step:5 end of class main

### Program Code :

```
#include<iostream>
using namespace std;
class cl
{
    int a;
    public :
        cl();
        cl(int x);
        cl(int x,int y);
        void output();
};
```

```

cl::cl()
{
    a=5;
    cout<<"default constructor"<<endl;
}
cl::cl(int x)
{
    a=x;
    cout <<"constructor with one parameter "<<endl;
}
cl::cl(int x,int y)
{
    a=x+y;
    cout<<"constructor with two parameter"<<endl;
}
void cl::output()
{
    cout<<a<<endl;
}
int main()
{
    cl ob1;
    cl ob2(6);
    cl ob3(7,8);
    ob1.output();
    ob2.output();
    ob3.output();
    return 0;
}

```

### **Output:**

default constructor

constructor with one parameter

constructor with two parameters

5

6

15

### **Discussion :**

Overloaded constructors essentially have the same name (name of the class) and different number of arguments.

A constructor is called depending upon the number and type of arguments passed

## 5 . Implementation of Stack Operation.

### Algorithm :

- ALGORITHM FOR CLASS 'st':  
Step:1 start  
Step:2 declare the following data type top, a[30]  
Step:3 declare the constructor 'st()'
  - ALGORITHM FOR CONSTRUCTOR 'st()':  
Step:1 start  
Step:2 set top= -1  
Sep:3 endStep:4 declare the following member function 'overflow()', 'underflow()' in type 'int' and 'push(int)', 'pop()', 'peep()', 'display()' in type 'void'  
Step:5 declare the destructor '~st()'
  - ALGORITHM FOR DESTRUCTOR '~st()':  
Step:1 start  
Step:2 print "end of program"  
Step:3 endStep:6 end of class 'st'
  - ALGORITHM FOR 'int' function 'overflow()':  
Step:1 start  
Step:2 if (top=29)  
    return 1  
    else  
    return 0  
    end if  
Step:3 end
    - ALGORITHM FOR 'int' function 'underflow()':  
Step:1 start  
Step:2 if(top=-1)  
    return 1  
    else  
    return 0  
    end if  
Step:3 end
      - ALGORITHM FOR 'void' function 'push(int x)':  
Step:1 start  
Step:2 if (overflow())=1  
    Print "stack overflow"  
    else  
    set top=top+1  
    set a[top]=x  
    end if  
Step:3 end
        - ALGORITHM FOR 'void' function 'pop()':  
Step:1 start  
Step:2 if(underflow())=1  
    Print "stack underflow"

```

        else
        print a[top]
        set top=top-1
        end if
Step:3 end
    • ALGORITHM FOR 'void' function 'peek()':
Step:1 start
Step:2 if(underflow()=1)
        Print "no element"
    else
        Print a[top]
    End if
Step:3 end
    • ALGORITHM FOR 'void' function 'display()':
Step:1 start
Step:2 if(underflow()=1)
        Print "no element"
    Else
        for(i=top, i>=0, i--)
            print a[i]
        end if
Step:3 end
    • ALGORITHM FOR 'main' FUNCTION:
Step:1 start
Step:2 declare object 'ob' for 'st'
Step:3 print "enter 1 for push"
        print "enter 2 for pop"
        print "enter 3 for peek"
        print "enter 4 for display"
        print "enter 5 for exit"
Step:4 declare x as int
Step:5 print x
Step:6 if(x=1)
        Declare value in int
        print "enter a value for push"
        print value
        call push(value) function
    end if
Step:7 if(x=2)
        Call pop() function
    End if
Step:8 if(x=3)
        Call peek() function
    End if
Step:9 if(x=4)
        Call display() function
    End if
Step:10 if(x=5)
        Print "Thank You "

```

```
        break
    End if
Step:11 end of main function
```

## Program Code :

```
#include<iostream>
using namespace std;
class st
{
    int top,a[30];
    public :
        st()
        {
            top=-1;
        }
        int overflow();
        int underflow();
        void push(int);
        void pop();
        void peep();
        void display();
        ~st()
        {
            cout<<"End of Program "<<endl;
        }
};
int st::overflow()
{
    if (top==29)
        return 1;
    else
        return 0;
}
int st::underflow()
{
    if(top== -1)
        return 1;
    else
        return 0;
}
void st::push(int x)
{
    if (overflow()==1)
    {
        cout<<"stack overflow"<<endl;
    }
    else
    {
        top=top+1;
        a[top]=x;
    }
}
```

```

void st::pop()
{
    if(underflow()==1)
    {
        cout<<"stack underflow"<<endl;
    }
    else
    {
        cout<<a[top]<<endl;
        top=top-1;
    }
}
void st::peek()
{
    if (underflow()==1)
    {
        cout<<"no element"<<endl;
    }
    else
    {
        cout<<a[top]<<endl;
    }
}
void st::display()
{
    if(underflow()==1)
    {
        cout<<"no element"<<endl;
    }
    else
    {
        for(int i=top;i>=0;i--)
        {
            cout<<a[i]<<endl;
        }
    }
}
int main()
{
    st ob;
    while (1)
    {
        cout<<"enter 1 for push"<<endl;
        cout<<"enter 2 for pop"<<endl;
        cout<<"enter 3 for peek"<<endl;
        cout<<"enter 4 for display"<<endl;
        cout<<"enter 5 for exit"<<endl;
        int x;
        cin>>x;
        if(x==1)
        {
            int value;

```



```

        cout<<"enter a value for push"<<endl;
        cin>>value;
        ob.push(value);
    }
    if(x==2)
    {
        ob.pop();
    }
    if(x==3)
    {
        ob.peep();
    }
    if(x==4)
    {
        ob.display();
    }
    if(x==5)
    {
        cout<<"Thank You "<<endl;
        break ;
    }
}
return 0;
}

```

### **Output:**

enter 1 for push

enter 2 for pop

enter 3 for peep

enter 4 for display

enter 5 for exit

1

enter a value for push

2

enter 1 for push

enter 2 for pop

enter 3 for peep

enter 4 for display

enter 5 for exit

1

enter a value for push

5

enter 1 for push

enter 2 for pop

enter 3 for peep

enter 4 for display

enter 5 for exit

1

enter a value for push

3

enter 1 for push

enter 2 for pop

enter 3 for peep

enter 4 for display

enter 5 for exit

4

3

5

2

enter 1 for push

enter 2 for pop

enter 3 for peep

enter 4 for display

enter 5 for exit

2

3

enter 1 for push

enter 2 for pop

enter 3 for peep

enter 4 for display

enter 5 for exit

4

5

2

enter 1 for push

enter 2 for pop

enter 3 for peep

enter 4 for display

enter 5 for exit

5

Thank You

End of Program

## **Discussion :**

Stacks are a type of container adaptors with LIFO (Last In First Out) type of working, where a new element is added at one end and (top) an element is removed from that end only. push() function is used to insert an element at the top of the stack. The element is added to the stack container and the size of the stack is increased by 1. pop() function is used to remove an element from the top of the stack (newest element in the stack). The element is removed to the stack container and the size of the stack is decreased by 1.

## 6 . Implementation of Queue Operation.

### Algorithm :

- ALGORITHM FOR CLASS 'Q':  
Step:1 start  
Step:2 declare the following data type front, rear, a[30]  
Step:3 declare the constructor 'Q()'
  - ALGORITHM FOR CONSTRUCTOR 'Q()':  
Step:1 start  
Step:2 set front= -1  
Step:3 set rear= -1  
Step:4 end  
Step:4 declare the following member function 'overflow()', 'underflow()' in type 'int' and 'insert(int)', 'delete()', 'display()' in type 'void'  
Step:5 declare the destructor '~Q()'
  - ALGORITHM FOR DESTRUCTOR '~Q()':  
Step:1 start  
Step:2 print "end of program"  
Step:3 end  
Step:6 end of class 'Q'
  - ALGORITHM FOR 'int' function 'overflow()':  
Step:1 start  
Step:2 if (rear==29)  
    return 1  
    else  
    return 0  
    end if  
Step:3 end
  - ALGORITHM FOR 'int' function 'underflow()':  
Step:1 start  
Step:2 if(rear==0)  
    return 1  
    else  
    return 0  
    end if  
Step:3 end
  - ALGORITHM FOR 'void' function 'insert(int x)':  
Step:1 start  
Step:2 if (overflow()==1)  
    Print "queue overflow"  
    else  
    set rear=rear+1  
    set a[rear]=x  
        if (front==0)  
        set front++  
    end if  
    end if  
Step:3 end

- ALGORITHM FOR 'void' function 'delete()':

Step:1 start

Step:2 if(underflow()==1)

Print "queue underflow"

else

set x=a[front]

print x

if(front==rear)

set front=-1

set rear=-1

else

set front++

end if

end if

Step:3 end

- ALGORITHM FOR 'void' function 'display()':

Step:1 start

Step:2 if(underflow()==1)

Print "no element"

Else

for(i=front, i<=rear, i++)

print a[i]

end if

Step:3 end

- ALGORITHM FOR 'main' FUNCTION:

Step:1 start

Step:2 declare object 'ob' for 'Q'

Step:3 print "enter 1 for insert"

print "enter 2 for delete"

print "enter 3 for display"

print "enter 4 for exit"

Step:4 declare n as int

Step:5 print n

Step:6 if(n==1)

Declare value in int

print "enter a value for insert"

print value

call insert(value) function

end if

Step:7 if(n==2)

Call delete() function

End if

Step:8 if(n==3)

Call display() function

End if

Step:9 if(x==4)

Print "Thank You "

break

End if  
Step:10 end of main function

## Program Code :

```
#include<iostream>
using namespace std;
class Q
{
    int front,rear,a[30];
    public :
        Q()
        {
            front=-1;
            rear=-1;
        }
        int overflow();
        int underflow();
        void insert(int);
        void del();
        void display();
        ~Q()
        {
            cout<<"End of program "<<endl;
        }
};
int Q::overflow()
{
    if (rear==29)
        return 1;
    else
        return 0;
}
int Q::underflow()
{
    if(front== -1)
        return 1;
    else
        return 0;
}
void Q::insert(int x)
{
    if(overflow()==1)
        cout<<"queue overflow"<<endl;
    else
    {
        rear=rear+1;
        a[rear]=x;
        if(front== -1)
            front++;
    }
}
void Q::del()
```

```

{
    if(underflow()==1)
        cout<<"queue unerflow"<<endl;
    else
    {
        int x=a[front];
        cout<<x<<endl;
        if(front==rear)
        {
            front=-1;
            rear=-1;
        }
        else
            front++;
    }
}
void Q::display()
{
    if(underflow()==1)
    {
        cout<<"No element"<<endl;
    }
    else
    {
        for(int i=front;i<=rear;i++)
        {
            cout<<a[i]<<endl;
        }
    }
}

int main()
{
    Q ob;
    while(1)
    {
        cout<<"enter 1 for insert"<<endl;
        cout<<"enter 2 for delete"<<endl;
        cout<<"enter 3 for display"<<endl;
        cout<<"enter 4 for exit"<<endl;
        int n;
        cin>>n;
        if(n==1)
        {
            int value;
            cout<<"enter a value for push"<<endl;
            cin>>value;
            ob.insert(value);
        }
        if(n==2)
        {
            ob.del();
        }
    }
}

```

```

        if(n==3)
        {
            ob.display();
        }
        if(n==4)
        {
            cout<<"Thank You"<<endl;
            break ;
        }
    }
    return 0;
}

```

## Output:

enter 1 for insert

enter 2 for delete

enter 3 for display

enter 4 for exit

1

enter a value for insertion

2

enter 1 for insert

enter 2 for delete

enter 3 for display

enter 4 for exit

1

enter a value for insertion

8

enter 1 for insert

enter 2 for delete

enter 3 for display

enter 4 for exit

1

enter a value for insertion



5

enter 1 for insert

enter 2 for delete

enter 3 for display

enter 4 for exit

3

2

8

5

enter 1 for insert

enter 2 for delete

enter 3 for display

enter 4 for exit

2

2

enter 1 for insert

enter 2 for delete

enter 3 for display

enter 4 for exit

4

Thank You

End of program

## **Discussion :**

Queue are a type of container adaptors which operate in a first in first out (FIFO) type of arrangement. Elements are inserted at the back (end) and are deleted from the front. insert() function is used to insert an element at the back of the queue. The element is added to the queue container and the size of the queue is increased by 1. del() function is used to remove an element from the front of the queue (oldest element in the queue). The element is removed to the queue container and the size of the queue is decreased by

## 7 . Array insertion deletion with linear search and binary search.

### Algorithm :

- ALGORITHM FOR CLASS 'cl':

Step:1 start

Step:2 declare the following data type len, idx, a[30], n

Step:3 declare the following member function : void input(), void lsearch(), void bsearch(), void insert(), void del(), void display()

Step:4 end of class 'cl'

- ALGORITHM FOR FUNCTION 'void input()':

Step:1 start

Step:2 print "enter the length of the array"

Step:3 print len

Step:4 print "enter the element of the array"

for ( i=0, i<len, i++)

print a[i]

set idx++

Step:5 end

- ALGORITHM FOR FUNCTION 'void lsearch()':

Step:1 start

Step:2 print "Linear Seach"

Step:3 declare value as int, set c=0;

Step:4 print "enter the value for search"

Step:5 print value

for( i=0, i<len, i++)

if(a[i]=value)

set c++

else if(c=0)

print "value is not found"

else

print "value is found "<<endl;

end if

end if

Step:6 end

- ALGORITHM FOR FUNCTION 'void bsearch()':

Step:1 start

Step:2 print "Binary Seach"

Step:3 declare value, l, h, m as int

Step:4 print "enter the value for search"

Step:5 print value

Step:6 set l=0

Step:7 set h=len-1

Step:8 set m=(l+h)/2

if(a[m]=value)

print "value is found "

break

```

end if
if(value<a[m])
    set h=m-1
else
    set l=m+1
end if
if(l>h)
    print "value is not found "
break
end if

```

Step:9 end

- ALGORITHM FOR FUNCTION 'void insert()':

Step:1 start

```

Step:2 if(idx=0)
    print "Array is empty" next line "For insertion "
    input()
else
    print "Enter the value "
    print n
end if
for( i=0, i<=idx, i++)
    if(i=idx)
        set a[idx]=n
        set idx++
    end if

```

Step:3 end

- ALGORITHM FOR FUNCTION 'void del()':

Step:1 start

```

Step:2 for( i=0, i<=idx, i++)
    if(i=idx)
        set idx--
    end if

```

Step:3 end

- ALGORITHM FOR FUNCTION 'void display()':

Step:1 start

```

Step:2 for( i=0, i<=idx, i++)
    Print a[i]

```

Step:3 end

- ALGORITHM FOR 'main' FUNCTION 'int main()':

Step:1 start

Step:2 declare 'ob' as object for 'cl' and declare k as int

```

Step:3 print "For insertion press 1"
    Print "For linear search press 2"
    Print "For binary search press 3"
    print "For deletion press 4"
    print "For display press 5"
    print "For exit press 0"
    print k
    step:1 if(k=1)
        call insert() function

```

```

step:2 else if(k=2)
    call Lsearch() function
step:3 else if(k=3)
    call Bsearch() function
step:4 else if(k=4)
    call del() function
step:5 else if(k=5)
    call dis() function
step:6 else if(k=0)
    break
    else
    print "Wrong choice"
    end if

```

Step:4 end

## Program Code :

```

#include<iostream>
#include<math.h>
using namespace std;
class cl
{
    int a[30],len,ins,pos,n,d;
    public :
        void input();
        void Lsearch();
        void Bsearch();
        void del();
        void dis();
};
void cl::input()
{
    int count;
    if(count==0)
    {
        cout<<"Enter the length of the Array : ";
        cin>>len;
        cout<<"Enter array elements : ";
        for(int i=0; i<len; i++)
        {
            cin>>a[i];
            count=1;
        }
    }
    else
    {
        cout<<"Enter element to be insert : ";
        cin>>ins;
        cout<<"At which position (Enter index number) ? ";
        cin>>pos;
        for(int i=len; i>pos; i--)
        {
            a[i]=a[i-1];

```

```

        }
        a[pos]=ins;
        cout<<"Element inserted successfully..!!\n";
        len++;
    }
}
void cl::Lsearch()
{
    cout<<"Linear Search "<<endl;
    int value,c=0;
    cout<<"enter the value for search"<<endl;
    cin>>value;
    for(int i=0;i<len;i++)
    {
        if(a[i]==value)
            c++;
    }
    if(c==0)
    {
        cout<<"value is not found"<<endl;
    }
    else
    {
        cout<<"value is found "<<endl;
    }
}
void cl::Bsearch()
{
    cout<<"Binary Search "<<endl;
    int value,l,h,m;
    cout<<"enter the value for search "<<endl;
    cin>>value;
    l=0;
    h=len-1;
    while(1)
    {
        m=(l+h)/2;
        if(a[m]==value)
        {
            cout<<"value is found "<<endl;
            break ;
        }
        if(value<a[m])
        {
            h=m-1;
        }
        else
        {
            l=m+1;
        }
        if(l>h)
        {
            cout<<"value is not found "<<endl;

```

```

        break;
    }
}
}
void cl::del()
{
    cout<<"Enter element to be delete : ";
    cin>>d;
    for(int i=0; i<len; i++)
    {
        if(a[i]==d)
        {
            for(int j=i; j<(len-1); j++)
            {
                a[j]=a[j+1];
            }
            len--;
        }
    }
}
void cl::dis()
{
    for(int i=0;i<len;i++)
    {
        cout<<a[i]<<endl;
    }
}
int main()
{
    cl ob; int k;
    while(1)
    {
        cout<<"For insertion press 1"<<endl;
        cout<<"For linear search press 2"<<endl;
        cout<<"For binary search press 3"<<endl;
        cout<<"For deletion press 4"<<endl;
        cout<<"For display press 5"<<endl;
        cout<<"For exit press 0"<<endl;
        cin>>k;
        if(k==1)
        {
            ob.input();
        }
        else if(k==2)
        {
            ob.Lsearch();
        }
        else if(k==3)
        {
            ob.Bsearch();
        }
        else if(k==4)
        {

```

```

        ob.del();
    }
    else if(k==5)
    {
        ob.dis();
    }
    else if(k==0)
    {
        break;
    }
    else
    {
        cout<<"Wrong choice"<<endl;
    }
}
return 0;
}

```

Output:

For insertion press 1

For linear search press 2

For binary search press 3

For deletion press 4

For display press 5

For exit press 0

1

Enter the length of the Array: 5

Enter array elements:

2

4

6

1

9

For insertion press 1

For linear search press 2

For binary search press 3

For deletion press 4

For display press 5

For exit press 0

2

Linear Search

enter the value for search

1

value is found

For insertion press 1

For linear search press 2

For binary search press 3

For deletion press 4

For display press 5

For exit press 0

3

Binary Search

enter the value for search

6

value is found

For insertion press 1

For linear search press 2

For binary search press 3

For deletion press 4

For display press 5

For exit press 0

5

2

4



6

1

9

For insertion press 1

For linear search press 2

For binary search press 3

For deletion press 4

For display press 5

For exit press 0

4

Enter element to be delete: 2

For insertion press 1

For linear search press 2

For binary search press 3

For deletion press 4

For display press 5

For exit press 0

5

4

6

1

9

For insertion press 1

For linear search press 2

For binary search press 3

For deletion press 4

For display press 5

For exit press 0

**Discussion :**

An array in C or C++ is a collection of items stored at contiguous memory locations and elements can be accessed randomly using indices of an array. They are used to store similar type of elements as in the data type must be the same for all elements. A simple approach is to do linear search, i.e. Start from the leftmost element of `arr[]` and one by one compare `x` with each element of `arr[]`. If `x` matches with an element, return the index. If `x` doesn't match with any of elements, return -1.

Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

## 8 . Implementation of Linked list

### Algorithm :

step 1: start

step 2: defining class node

step3 : declareing datatype

```
int data;
```

```
node *link;
```

step 4: defining class list public node

step 5:node \*head;

step 6:declaring public:

```
list();
```

```
void insertion(int);
```

```
void deletion();
```

```
void traverse();
```

```
void searching(int);
```

```
~list();
```

```
};
```

```
list::list()
```

```
{
```

```
    head=NULL;
```

```
}
```

```
void list::insertion(int x)
```

```
{
```

```
    node *p, *temp;
```

```
    if(head==NULL)
```

```
    {
```

```
        head=new node();
```

```
        head->data=x;
```

```
        head->link=NULL;
```

```
    }
```

```
    else
```

```
    {
```

```
        p=head;
```

```
        while(p->link!=NULL)
```

```
            p=p->link;
```

```
        temp=new node();
```

```
        temp->data=x;
```

```
        temp->link=NULL;
```

```
        p->link=temp;
```

```
    }
```

```
}
```

```
void list::deletion()
```

```
{
```

```
    node *p, *temp;
```

```
    int x;
```

```
    if(head==NULL)
```

```
    {
```

```
        cout<<"Empty List"<<endl;
```

```
    }
```

```

else if(head->link==NULL)
{
    x=head->data;
    head=NULL;
    cout<<"The deleted value is : "<<x<<endl;
}
else
{
    p=head;
    while(p->link->link!=NULL)
        p=p->link;
    temp=p->link;
    x=temp->data;
    p->link=NULL;
    delete(temp);
    cout<<"The deleted value is : "<<x<<endl;
}
}
void list::traverse()
{
    node *p;
    p=head;
    if(p==NULL)
    {
        cout<<"Empty List"<<endl;
    }
    else
    {
        cout<<"The contents of the list are : "<<endl;
        while(p!=NULL)
        {
            cout<<p->data<<endl;
            p=p->link;
        }
    }
}
void list::searching(int key)
{
    node *p;
    int c=0;
    p=head;
    if(p==NULL)
    {
        cout<<"Empty List"<<endl;
    }
    else
    {
        while(p!=NULL)
        {
            if(p->data==key)
                c++;
            p=p->link;
        }
    }
}

```

```

        if(c!=0)
        {
            cout<<"Search Successful"<<endl;
        }
        else
        {
            cout<<"Search Unsuccessful"<<endl;
        }
    }
}
list::~~list()
{
    cout<<"End of Program"<<endl;
}
int main()
{
    int n,x;
    list obj;
    while(1)
    {
        cout<<"Enter 1 to INSERT data into the LINKED LIST"<<endl;
        cout<<"Enter 2 to DELETE data from the LINKED LIST"<<endl;
        cout<<"Enter 3 DISPLAY the contents of the LINKED LIST"<<endl;
        cout<<"Enter 4 SEARCH data in the LINKED LIST"<<endl;
        cout<<"Enter 0 to TERMINATE"<<endl;
        cout<<"Enter Your Choice....."<<endl;
        cin>>n;
        if(n==1)
        {
            cout<<"Enter the data....."<<endl;
            cin>>x;
            obj.insertion(x);
        }
        else if(n==2)
            obj.deletion();
        else if(n==3)
            obj.traverse();
        else if(n==4)
        {
            cout<<"Enter the value to be searched....."<<endl;
            cin>>x;
            obj.searching(x);
        }
        else if(n==0)
            break;
        else
            cout<<"Wrong Choice!!"<<endl;
    }
}

```

## Program Code :

```
#include<iostream>
```

```

using namespace std;
class node
{
    public:
        int data;
        node *link;
};
class list : public node
{
    node *head;
    public:
        list();
        void insertion(int);
        void deletion();
        void traverse();
        void searching(int);
        ~list();
};
list::list()
{
    head=NULL;
}
void list::insertion(int x)
{
    node *p, *temp;
    if(head==NULL)
    {
        head=new node();
        head->data=x;
        head->link=NULL;
    }
    else
    {
        p=head;
        while(p->link!=NULL)
            p=p->link;
        temp=new node();
        temp->data=x;
        temp->link=NULL;
        p->link=temp;
    }
}
void list::deletion()
{
    node *p, *temp;
    int x;
    if(head==NULL)
    {
        cout<<"Empty List"<<endl;
    }
    else if(head->link==NULL)
    {
        x=head->data;
    }
}

```

```

        head=NULL;
        cout<<"The deleted value is : "<<x<<endl;
    }
    else
    {
        p=head;
        while(p->link->link!=NULL)
            p=p->link;
        temp=p->link;
        x=temp->data;
        p->link=NULL;
        delete(temp);
        cout<<"The deleted value is : "<<x<<endl;
    }
}

void list::traverse()
{
    node *p;
    p=head;
    if(p==NULL)
    {
        cout<<"Empty List"<<endl;
    }
    else
    {
        cout<<"The contents of the list are : "<<endl;
        while(p!=NULL)
        {
            cout<<p->data<<endl;
            p=p->link;
        }
    }
}

void list::searching(int key)
{
    node *p;
    int c=0;
    p=head;
    if(p==NULL)
    {
        cout<<"Empty List"<<endl;
    }
    else
    {
        while(p!=NULL)
        {
            if(p->data==key)
                c++;
            p=p->link;
        }
        if(c!=0)
        {
            cout<<"Search Successful"<<endl;
        }
    }
}

```

```

        }
        else
        {
            cout<<"Search Unsuccessful"<<endl;
        }
    }
}
list::~~list()
{
    cout<<"End of Program"<<endl;
}
int main()
{
    int n,x;
    list obj;
    while(1)
    {
        cout<<"Enter 1 to INSERT data into the LINKED LIST"<<endl;
        cout<<"Enter 2 to DELETE data from the LINKED LIST"<<endl;
        cout<<"Enter 3 DISPLAY the contents of the LINKED LIST"<<endl;
        cout<<"Enter 4 SEARCH data in the LINKED LIST"<<endl;
        cout<<"Enter 0 to TERMINATE"<<endl;
        cout<<"Enter Your Choice....."<<endl;
        cin>>n;
        if(n==1)
        {
            cout<<"Enter the data....."<<endl;
            cin>>x;
            obj.insertion(x);
        }
        else if(n==2)
            obj.deletion();
        else if(n==3)
            obj.traverse();
        else if(n==4)
        {
            cout<<"Enter the value to be searched....."<<endl;
            cin>>x;
            obj.searching(x);
        }
        else if(n==0)
            break;
        else
            cout<<"Wrong Choice!!"<<endl;
    }
}

```

## Output:

Enter 1 to INSERT data into the LINKED LIST

Enter 2 to DELETE data from the LINKED LIST



Enter 3 DISPLAY the contents of the LINKED LIST

Enter 4 SEARCH data in the LINKED LIST

Enter 0 to TERMINATE

Enter Your Choice.....

1

Enter the data.....

2

Enter 1 to INSERT data into the LINKED LIST

Enter 2 to DELETE data from the LINKED LIST

Enter 3 DISPLAY the contents of the LINKED LIST

Enter 4 SEARCH data in the LINKED LIST

Enter 0 to TERMINATE

Enter Your Choice.....

1

Enter the data.....

5

Enter 1 to INSERT data into the LINKED LIST

Enter 2 to DELETE data from the LINKED LIST

Enter 3 DISPLAY the contents of the LINKED LIST

Enter 4 SEARCH data in the LINKED LIST

Enter 0 to TERMINATE

Enter Your Choice.....

1

Enter the data.....

3

Enter 1 to INSERT data into the LINKED LIST

Enter 2 to DELETE data from the LINKED LIST

Enter 3 DISPLAY the contents of the LINKED LIST

Enter 4 SEARCH data in the LINKED LIST

Enter 0 to TERMINATE

Enter Your Choice.....

3

The contents of the list are :

2

5

3

Enter 1 to INSERT data into the LINKED LIST

Enter 2 to DELETE data from the LINKED LIST

Enter 3 DISPLAY the contents of the LINKED LIST

Enter 4 SEARCH data in the LINKED LIST

Enter 0 to TERMINATE

Enter Your Choice.....

2

The deleted value is : 3

Enter 1 to INSERT data into the LINKED LIST

Enter 2 to DELETE data from the LINKED LIST

Enter 3 DISPLAY the contents of the LINKED LIST

Enter 4 SEARCH data in the LINKED LIST

Enter 0 to TERMINATE

Enter Your Choice.....

3

The contents of the list are :

2

5

Enter 1 to INSERT data into the LINKED LIST

Enter 2 to DELETE data from the LINKED LIST

Enter 3 DISPLAY the contents of the LINKED LIST

Enter 4 SEARCH data in the LINKED LIST

Enter 0 to TERMINATE

Enter Your Choice.....

4

Enter the value to be searched.....

5

Search Successful

Enter 1 to INSERT data into the LINKED LIST

Enter 2 to DELETE data from the LINKED LIST

Enter 3 DISPLAY the contents of the LINKED LIST

Enter 4 SEARCH data in the LINKED LIST

Enter 0 to TERMINATE

Enter Your Choice.....

0

End of Program

## **Discussion :**

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers.

The function insert() inserts the data into the beginning of the linked list. It creates a new\_node and inserts the number in the data field of the new\_node. Then the

new\_node points to the head. Finally the head is the new\_node i.e. the linked list starts from there. In order to delete the node, which is present after the specified node, we need to skip the desired number of nodes to reach the node after which the node will be deleted.

We need to keep track of the two nodes. The one which is to be deleted the other one if the node which is present before that node.

## 9 . Addition of two complex number using Friend function.

### Algorithm :

- ALGORITHM FOR CLASS 'complex':  
Step:1 start  
Step:2 declare the following data member : r, i in integer  
Step:3 declare the following member function : void input(int, int), void output()  
Step:4 declare friend function 'friend void add(complex, complex)  
Step:5 end of class 'complex'
  - ALGORITHM FOR FUNCTION 'void input(int x, int y)':  
Step:1 start  
Step:2 set r= x  
Step:3 set i= y  
Step:4 end
  - ALGORITHM FOR FUNCTION 'void output()':  
Step:1 start  
Step:2 print r "+i" i  
Step:3 end
  - ALGORITHM FOR FRIEND FUNCTION 'void add(complex ob1, complex ob2)':  
Step:1 start  
Step:2 declare 'ob' object for class 'complex'  
Step:3 set ob3.r=ob1.r+ob2.r  
Step:4 set ob3.i=ob1.i+ob2.i  
Step:5 call output() function  
Step:6 end
  - ALGORITHM FOR 'main' FUNCTION 'int main()':  
Step:1 start  
Step:2 declare x1,x2,y1,y2 as integer  
Step:3 Declare object 'ob1', 'ob2' for class 'complex'  
Step:4 Print "Enter the value of real and imaginary part "  
Step:5 Print x1 and y1  
Step:6 Call input(x1,y1) function for 'ob1'  
Step:7 Print "Enter value of real and imaginary part of 2nd no."  
Step:8 Print x2 and y2  
Step:9 Call input(x2,y2) function for 'ob2'  
Step:10 Call function add(ob1,ob2)  
Step:11 End of main function

### 1. ADDITION AND SUBSTACTION OF TWO COMPLEX NO.S USING OPERATOR OVERLOADING:

- ALGORITHM FOR CLASS 'cl':  
Step:1 start  
Step:2 declare the following data member : a, b in integer  
Step:3 Declare default constructor 'cl()'

- ALGORITHM FOR DEFAULT CONSTRUCTOR 'cl()':  
 Step:1 start  
 Step:2 print " "  
 Step:3 end  
 Step:4 declare parameterized constructor 'cl(int, int)'  
 Step:5 declare the following member function : void output()  
 Step:6 declare operators as 'cl operator+(cl)' and 'cl operator-(cl)'  
 Step:7 end of class 'cl'
  - ALGORITHM FOR PARAMETERIZED CONSTRUCTOR 'cl(int x, int y)':  
 Step:1 start  
 Step:2 set a=x  
 Step:3 set b=y  
 Step:4 end
    - ALGORITHM FOR FUNCTION 'void op()':  
 Step:1 start  
 Step:2 print "a= " a next line "b= " b  
 Step:3 end
      - ALGORITHM FOR +operator 'operator+(cl ob2)':  
 Step:1 start  
 Step:2 call object 'ob3' for class 'cl'  
 Step:3 print "Operator +"  
 Step:4 set ob3.a=a+ob2.a  
 Step:5 set ob3.b=b+ob2.b;  
 Step:6 return the value of 'ob3'  
 Step:7 end
        - ALGORITHM FOR -operator 'operator-(cl ob2)':  
 Step:1 start  
 Step:2 call object 'ob4' for class 'cl'  
 Step:3 print "Operator +"  
 Step:4 set ob4.a=a-ob2.a  
 Step:5 set ob4.b=b-ob2.b;  
 Step:6 return the value of 'ob4'  
 Step:7 end
          - ALGORITHM FOR 'main' function 'int main()':  
 Step:1 start  
 Step:2 call object 'ob1(5,10)' for 'cl'  
 Step:3 call op() function for 'ob1'  
 Step:4 call object 'ob2(2,4)' for 'cl'  
 Step:5 call op() function for 'ob2'  
 Step:6 call object 'ob3' for 'cl', call object 'ob4' for 'cl'  
 Step:7 set ob3=ob1+ob2  
 Step:8 call op() function for 'ob3'  
 Step:9 set ob4=ob1-ob2  
 Step:10 call op() function for 'ob4'  
 Step:11 end

## Program Code :

```
#include<iostream>
```

```

#include<math.h>
using namespace std;
class complex
{
    int r,i;
    public :
        void input(int ,int);
        void output();
        friend void add(complex, complex);
};
void complex::input(int x, int y)
{
    r=x;
    i=y;
}
void complex::output()
{
    cout<<r<<"+"i"<<i<<endl;
}
void add(complex ob1, complex ob2)
{
    complex ob3;
    ob3.r=ob1.r+ob2.r;
    ob3.i=ob1.i+ob2.i;
    ob3.output();
}
int main()
{
    int x1,x2,y1,y2;
    complex ob1,ob2;
    cout<<"Enter the value of real and imaginary part "<<endl;
    cin>>x1>>y1;
    ob1.input(x1,y1);
    cout<<"Enter value of real and imaginary part of 2nd no."<<endl;
    cin>>x2>>y2;
    ob2.input(x2,y2);
    add(ob1,ob2);
    return 0;
}

```

## Output:

Enter the value of real and imaginary part

10

50

Enter value of real and imaginary part of 2nd no.

30

30

40+i80

## **Discussion :**

A friend function of a class is defined outside that class' scope but it has the right to

access all private and protected members of the class. Even though the prototypes for

friend functions appear in the class definition, friends are not member functions.

The cmath header file is used to take help of sqrt() library function.

The iostream header file helps to use the 'cin' & 'cout' library functions.

## **10 . Addition and Subtraction using operator overloading.**

### **Algorithm :**

ALGORITHM FOR CLASS 'cl':

Step:1 start

Step:2 declare the following data member : a, b in integer

Step:3 Declare default constructor 'cl()'

ALGORITHM FOR DEFAULT CONSTRUCTOR 'cl()':

Step:1 start

Step:2 print " "

Step:3 end

Step:4 declare parameterized constructor 'cl(int, int)'

Step:5 declare the following member function : void output()

Step:6 declare operators as 'cl operator+(cl)' and 'cl operator-(cl)'

Step:7 end of class 'cl'

ALGORITHM FOR PARAMETERIZED CONSTRUCTOR 'cl(int x, int y)':

Step:1 start

Step:2 set a=x

Step:3 set b=y

Step:4 end

ALGORITHM FOR FUNCTION 'void op()':

Step:1 start

Step:2 print "a= " a next line "b= " b

Step:3 end

ALGORITHM FOR +operator 'operator+(cl ob2)':

Step:1 start



Step:2 call object 'ob3' for class 'cl'

Step:3 print "Operator +"

Step:4 set ob3.a=a+ob2.a

Step:5 set ob3.b=b+ob2.b;

Step:6 return the value of 'ob3'

Step:7 end

ALGORITHM FOR -operator 'operator-(cl ob2)':

Step:1 start

Step:2 call object 'ob4' for class 'cl'

Step:3 print "Operator +"

Step:4 set ob4.a=a-ob2.a

Step:5 set ob4.b=b-ob2.b;

Step:6 return the value of 'ob4'

Step:7 end

ALGORITHM FOR 'main' function 'int main()':

Step:1 start

Step:2 call object 'ob1(5,10)' for 'cl'

Step:3 call op() function for 'ob1'

Step:4 call object 'ob2(2,4)' for 'cl'

Step:5 call op() function for 'ob2'

Step:6 call object 'ob3' for 'cl', call object 'ob4' for 'cl'

Step:7 set ob3=ob1+ob2

### **Program Code :**

```
#include<iostream>
using namespace std;
class cl
{
    int a,b;
    public :
```

```

        cl()
        {
            cout<<" ";
        }
        cl(int,int);
        void op();
        cl operator+(cl);
        cl operator-(cl);
};
cl::cl(int x, int y)
{
    a=x;
    b=y;
}
void cl::op()
{
    cout<<"a= "<<a<<endl<<"b= "<<b<<endl;
}
cl cl::operator+(cl ob2)
{
    cl ob3;
    cout<<"Operator +"<<endl;
    ob3.a=a+ob2.a;
    ob3.b=b+ob2.b;
    return ob3;
}
cl cl::operator-(cl ob2)
{
    cl ob4;
    cout<<"Operator -"<<endl;
    ob4.a=a-ob2.a;
    ob4.b=b-ob2.b;
    return ob4;
}
int main()
{
    cl ob1(5,10);
    ob1.op();
    cl ob2(2,4);
    ob2.op();
    cl ob3;cl ob4;
    ob3=ob1+ob2;
    ob3.op();
    ob4=ob1-ob2;
    ob4.op();
    return 0;
}

```

## Output:

a= 5

b= 10

a= 2

b= 4

Operator +

a= 7

b= 14

Operator -

a= 3

b= 6

### **Discussion :**

Overloaded operators are functions with special names: the keyword "operator" followed by the symbol for the operator being defined. Like any other function, an overloaded operator has a return type and a parameter list.

Most overloaded operators may be defined as ordinary non-member functions or as class member functions.

## 11 . Implementation of Multilevel Inheritance

### Algorithm :

Algorithm for class cl1

Step1. Start

Step2. Declare the following data type under “protected” data member  
int a

Step3. Declare the following function under “public” data member  
void val1()

- Algorithm for void val1()

Step1. Start

Step2. Print ‘enter the value of b’

Step3. Scan b

Step4. Print b [value of b]

Step5. End

Step4. End of class cl1

- Algorithm for class cl2 -&gt; cl2 is inherit from cl1 publicly

Step1. Start

Step2. Declare the following data type under “protected” data member  
int b

Step3. Declare the following function under “public” data member  
void val2()

- Algorithm for void val2()

Step1. Start

Step2. Print ‘enter value for b’

Step3. Scan b

Step4. Print b [value of b]

Step5. Print ‘a inherited from cl1’

Step6. Print a [value of a]

Step7. End

Step4. End of class cl2

- Algorithm for class cl3 -&gt; cl3 is inherit from cl2 publicly

Step1. Start

Step2. Declare following data type under protected data member  
int c

Step3. Declare following function under public data member  
void val3()

- Algorithm for void val3()

Step1. Start

Step2. Print ‘enter value of c’

Step3. Scan c

Step4. Print c [value of c]

Step5. Print ‘a & b inherited from cl2’

Step6. Print b [value of b]

Step7. Print a [value of a]

Step4. End of class cl3

- Algorithm for main function

Step1. Start

Step2. Declare object cl

Step3. Call function val1();

Step4. Call function val2();

Step5. Call function val3() ;

Step6. End of main function

## Program Code :

```
#include<iostream>
using namespace std;
class cl1
{
    protected : int a;
    public :
        void val1()
        {
            cout<<"Enter the value of a "<<endl;
            cin>>a;
            cout<<"a= "<<a<<endl;
        }
};
class cl2 : public cl1
{
    protected : int b;
    public :
        void val2()
        {
            cout<<"Entaer the value of b "<<endl;
            cin>>b;
            cout <<"b= "<<b<<endl;
            cout<<" a inherited from cl1 "<<endl;
            cout<<"a= "<<a<<endl;
        }
};
class cl3 : public cl2
{
    protected : int c;
    public :
        void val3()
        {
            cout<<"Enter the value of c"<<endl;
            cin>>c;
            cout<<"c= "<<c<<endl;
            cout<<" a and b inherited from cl2 "<<endl;
            cout <<"b= "<<b<<endl;
            cout<<"a= "<<a<<endl;
        }
};
int main()
{
    cl3 ob3;
```

```
        ob3.val1();  
        ob3.val2();  
        ob3.val3();  
        return 0;  
    }
```

### **Output:**

Enter the value of a

10

a= 10

Enter the value of b

50

b= 50

a inherited from cl1

a= 10

Enter the value of c

60

c= 60

a and b inherited from cl2

b= 50

a= 10

### **Discussion :**

The capability of a class to derive properties and characteristics from another class is called **Inheritance**. Inheritance is one of the most important features of Object-Oriented Programming.

**Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class.

**Super Class:** The class whose properties are inherited by sub class is called Base Class or Super class.

In **Multilevel Inheritance** , a derived class is created from another derived class.

## 12 . Implementation of String operation

### Algorithm :

Algorithm for class str

Step1: Start

Step2: Declare the following under public data member

String s1, s2, marge str, void strinput()

- Algorithm for void strinput()

Step1: Start

Step2: print "enter string s1"

Step3: Scan s2

Step4: Print "enter string s2"

Step5: Scan s2

Step6: End

- Algorithm for strcmpare()

Step1: Start

Step1: if (s1=s2)

then

Print"s1 is equal to s2"

endif

if(s1>s2)

then

print 's1 is greater than s1'

endif

Step3: end

- Algorithm for strconcat()

Step1: start

Step2: margestr=s1+s2

Step3: Print Result and string

Step4: End

Step3: End of class str

- Algorithm for main function()

Step1: Declare the object str

Step2: Call the function strinput()

Step3: Call the function strcmpare()

Step4: Call the function strconcat()

Step5: Return 0

Step6: End of main function

### Program Code :

```
#include <iostream>
using namespace std;
class str
{
    public : string s1, s2, margestr;
            void strinput()
            {
```

```

        cout << "Enter string s1: ";
        getline (cin, s1);
        cout << "Enter string s2: ";
        getline (cin, s2);
    }
    void strcompare()
    {
        if (s1 == s2)
            cout << s1 << " is equal to " << s2 << endl;
        if (s1 > s2)
            cout << s1 << " is greater than " << s2 << endl;
        else
            cout << s2 << " is greater than " << s1 << endl;
    }
    void strconcat()
    {
        margestr = s1 + s2;
        cout << "Resultant String = " << margestr;
    }
};
int main()
{
    str ob;
    ob.strinput();
    ob.strcompare();
    ob.strconcat();
    return 0;
}

```

## Output:

Enter string s1: Computer

Enter string s2: Science

Science is greater than Computer

Resultant String = ComputerScience

## Discussion :

The iostream header file helps to use the 'cin' & 'cout' library functions.

The C-style character string originated within the C language and continues to be supported within C++. This string is actually a one-dimensional array of characters which is terminated by a null character '\0'. Thus, a null-terminated string contains the characters that comprise the string followed by a null.



## 13 . Implementation of Bubble sort , Insertion sort and Selection sort.

### Algorithm :

Algorithm for class search

Step1: Declare the following data type- int a[30],n

Step2: Declare the following function under public data member-  
Search(),Void input(),Void bubble sort();Void insertion sort(),void selection  
sort(),Void display()

Step3: End of class search

- Algorithm for Search()

Step1: Start

Step2: Print "enter the total no of elements"

Step3: Scan n;

Step4: End

- Algorithm for input()

Step1: Start

Step2: Print "enter the element..."

Step3: for( int i=0; i<n, i++)

then

scan a[i]

end for loop

Step4:End

- Algorithm for bubble sort()

Step1: Start

Step2: for(int i=0,i<n-1,i++)

then

for( int j=0,j<n-i-1,j++)

then

if(a[j]>a[j+1])

then

int temp=a[j+1]

a[j+1]=temp

endif

end for loop

end for loop

Step3: print "the array after bubble sort is ...."

Step4: call display() Function

Step5: End

- Algorithm for insertion sort()

Step1: start

Step2: declare the following data type

int i , key , j

Step3: for(i=1,i<n,i++)

then

key=a[i];

j=i-1;

```

while ( j>=0 && a[j]>key)
do
a[j+]=a[j]
j=j-1
done
a[j+]=key;
end of for loop
Step4: print the array after insertion sort is...
Step5: call display () function
Step6: End
    • Algorithm for selection sort()
Step1: start
Step2: declare the following datatype- int idx;
Step3: for(int i=0,i<n-1,i++)
then
idx=i
for(int j=i+1,j<n,j++)
then
if(a[j]<a[j+])
then
a[j+]=temp
endif
end for loop
end for loop
Step4: print the array after selection sort is....
Step5: call display () function
Step6: End
    • Algorithm for display()
Step1: start
Step2: for(int i=0,i<n,i++)
then
print a[i]
end for loop
Step3: End
    • Algorithm for main function()
Step1: start
Step2: declare the object obj;
Step3: declare the data type int c;
Step4: call the function input()
Step5: print 'enter 1 to sort the array with bubble sort'
Step6: print 'enter 2 to sort the array with insertion sort'
Step7: print 'enter 3 to sort the array with selection sort'
Step8: print 'enter 0 to exit'
Step9: scan c
Step10: if (c=1)
then
call the function bubble sort();
else if(c=2)
then

```

```

call the function insertion sort()
else if(c==3)
call the function selection sort()
else if(c==0)
then
print 'end of program'
endif

```

```
endif
```

```
endif
```

```
Step11: return 0
```

```
Step12: End of main function
```

## Program Code :

```

#include<iostream>
using namespace std;
class search
{
    int a[30], n;
    public:
        search();
        void input();
        void bubble_sort();
        void insertion_sort();
        void selection_sort();
        void display();
};
search::search()
{
    cout<<"Enter the total no. of elements"<<endl;
    cin>>n;
}
void search::input()
{
    cout<<"Enter the elements..."<<endl;
    for(int i=0;i<n;i++)
    {
        cin>>a[i];
    }
}
void search::bubble_sort()
{
    for(int i=0;i<n-1;i++)
    {
        for(int j=0;j<n-i-1;j++)
        {
            if(a[j]>a[j+1])
            {
                int temp=a[j];
                a[j]=a[j+1];

```

```

        a[j+1]=temp;
    }
}
}
cout<<"The array after bubble sort is..."<<endl;
display();
}
void search::insertion_sort()
{
    int i, key, j;
    for(i=1;i<n;i++)
    {
        key=a[i];
        j=i-1;
        while(j>=0&& a[j]>key)
        {
            a[j+1]=a[j];
            j=j-1;
        }
        a[j+1]=key;
    }
    cout<<"The array after insertion sort is..."<<endl;
    display();
}
void search::selection_sort()
{
    int idx;
    for(int i=0;i<n-1;i++)
    {
        idx=i;
        for(int j=i+1;j<n;j++)
        {
            if(a[j]< a[idx])
            {
                int temp=a[j];
                a[j]=a[idx];
                a[idx]=temp;
            }
        }
    }
    cout<<"The array after selection sort is..."<<endl;
    display();
}
void search::display()
{
    for(int i=0;i<n;i++)
        cout<<a[i]<<" ";
    cout<<endl;
}
int main()
{
    search obj;
    int c;

```

```

obj.input();
cout<<"Enter 1 to sort the array with bubble sort"<<endl;
cout<<"Enter 2 to sort the array with insertion sort"<<endl;
cout<<"Enter 3 to sort the array with selection sort"<<endl;
cout<<"Enter 0 to exit"<<endl;
cin>>c;
if(c==1)
    obj.bubble_sort();
else if(c==2)
    obj.insertion_sort();
else if(c==3)
    obj.selection_sort();
else if(c==0)
    cout<<"End of program"<<endl;
return 0;
}

```

## Output:

Enter the total no. of elements

5

Enter the elements...

2

1

4

5

3

Enter 1 to sort the array with bubble sort

Enter 2 to sort the array with insertion sort

Enter 3 to sort the array with selection sort

Enter 0 to exit

1

The array after bubble sort is...

1 2 3 4 5

Enter the total no. of elements

5

Enter the elements...

2

1

4

5

3

Enter 1 to sort the array with bubble sort

Enter 2 to sort the array with insertion sort

Enter 3 to sort the array with selection sort

Enter 0 to exit

1

The array after selection sort is...

1 2 3 4 5

## **Discussion :**

1.Insertion sort is used when number of elements is small. It can also be useful when

input array is almost sorted, only few elements are misplaced in complete big array.

2.The selection sort algorithm sorts an array by repeatedly finding the minimum

element (considering ascending order) from unsorted part and putting it at the

beginning.

3.Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the

adjacent elements if they are in wrong order.

## 14 . Implementation of Binary Search Tree

### Algorithm :

p-1: Declare data

Step-2: Declare left link

Step-3: Declare right link

1.Algorithm for class list

Step-1: Declare the following functions under public access type:

list()

struct node \*new\_node(int)

void inorder(struct node \*)

struct node \*insert(struct node \*, int)

struct node \*minValueNode(struct node \*)

struct node \*deleteNode(struct node \*, int)

2.Algorithm for list()

Step-1: Initialize the root as NULL

3. Algorithm for new\_node(int x)

Step-1: struct node \*temp=(struct node \*)malloc(sizeof(struct node))

Step-2: temp->data=x

Step-3: temp->l\_link=NULL

Step-4: temp->r\_link=NULL

Step-5: return temp

□ Algorithm for inorder(struct node \*temp)

Step-1: if(temp!=NULL)

then

inorder(temp->l\_link)

print temp->data

```
inorder(temp->r_link)
```

```
end if
```

4. Algorithm for insert(struct node \*temp, int x)

Step-1: if(temp=NULL)

then

```
return new_node(x);
```

```
end if
```

Step-2: if(x<temp->data)

then

```
temp->l_link=insert(temp->l_link,x);
```

else if(x>temp->data)

then

```
temp->r_link=insert(temp->r_link,x);
```

```
end if
```

Step-3: return temp

5. Algorithm for minValueNode(struct node \*p)

Step-1: struct node\* current = p;

Step-2: while (current->l\_link!=NULL)

do

```
current=current->l_link
```

```
end while
```

Step-3: return current

6. Algorithm for deleteNode(struct node \*root, int key)

Step-1: if (root == NULL)

then

```
return root
```



```

end if
Step-2: if (key<root->data)
then
root->l_link=deleteNode(root->l_link,key)
else if (key>root->data)
then
root->r_link=deleteNode(root->r_link,key)
else
then
if (root->l_link == NULL)
then
struct node *temp = root->r_link;
free(root);
return temp;
else if (root->r_link == NULL)
then
struct node *temp = root->l_link;
free(root);
return temp;
end if
struct node* temp = minValueNode(root->r_link);
root->data = temp->data;
root->r_link = deleteNode(root->r_link, temp->data);
end if
Step-3: return root
7.Algorithm for main()

```

```
Step-1: Initialize ob as class object
Step-2: Initialize n and x as integer
Step-3: while(1)
do
print 'Enter 1 to INSERT data into the BST'
print 'Enter 2 to DISPLAY the BST in INORDER'
print 'Enter 3 to DELETE a value from the BST'
print 'Enter 0 to EXIT'
print 'Enter Your Choice.....'
if(n==1)
then
print 'Enter the data....'
read x
if(root==NULL)
then
root=ob.insert(root,x)
else
then
ob.insert(root,x)
end if
else if(n==2)
then
ob.inorder(root)
else if(n==3)
then
int key
```

```
print 'Enter the value to be deleted'
```

```
read key;
```

```
ob.deleteNode(root, key)
```

```
else if(n==0)
```

```
then
```

```
break;
```

```
else
```

```
then
```

```
cout<<"Wrong Choice"<<endl
```

```
end if
```

```
step-4: return 0
```

## **Program Code :**

```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct node
{
    int data;
    struct node *l_link;
    struct node *r_link;
} *root;
class list
{
    public:
        list();
        struct node *new_node(int);
        void inorder(struct node *);
        struct node *insert(struct node *, int);
        struct node *minValueNode(struct node *);
        struct node *deleteNode(struct node *, int);
};
list::list()
{
    root=NULL;
}
struct node *list::new_node(int x)
{
    struct node *temp=(struct node *)malloc(sizeof(struct node));
    temp->data=x;
    temp->l_link=NULL;
    temp->r_link=NULL;
```

```

        return temp;
    }
    void list::inorder(struct node *temp)
    {
        if(temp!=NULL)
        {
            inorder(temp->l_link);
            cout<<temp->data<<" ";
            inorder(temp->r_link);
        }
    }
    struct node *list::insert(struct node *temp, int x)
    {
        if(temp==NULL)
            return new_node(x);
        if(x<temp->data)
            temp->l_link=insert(temp->l_link,x);
        else if(x>temp->data)
            temp->r_link=insert(temp->r_link,x);
        return temp;
    }
    struct node *list::minValueNode(struct node *p)
    {
        struct node* current = p;
        while (current&&current->l_link!=NULL)
            current=current->l_link;
        return current;
    }
    struct node *list::deleteNode(struct node *root, int key)
    {
        if (root == NULL)
            return root;
        if (key<root->data)
            root->l_link=deleteNode(root->l_link,key);
        else if (key>root->data)
            root->r_link=deleteNode(root->r_link,key);
        else
        {
            if (root->l_link == NULL)
            {
                struct node *temp = root->r_link;
                free(root);
                return temp;
            }
            else if (root->r_link == NULL)
            {
                struct node *temp = root->l_link;
                free(root);
                return temp;
            }
            struct node* temp = minValueNode(root->r_link);
            root->data = temp->data;
            root->r_link = deleteNode(root->r_link, temp->data);
        }
    }

```

```

    }
    return root;
}
int main()
{
    list ob;
    int n, x;
    while(1)
    {
        cout<<"Enter 1 to INSERT data into the BST"<<endl;
        cout<<"Enter 2 to DISPLAY the BST in INORDER"<<endl;
        cout<<"Enter 3 to DELETE a value from the BST"<<endl;
        cout<<"Enter 0 to EXIT"<<endl;
        cout<<"Enter Your Choice....."<<endl;
        cin>>n;
        if(n==1)
        {
            cout<<"Enter the data...."<<endl;
            cin>>x;
            if(root==NULL)
                root=ob.insert(root,x);
            else
                ob.insert(root,x);
        }
        else if(n==2)
        {
            ob.inorder(root);
            cout<<endl;
        }
        else if(n==3)
        {
            int key;
            cout<<"Enter the value to be deleted"<<endl;
            cin>>key;
            ob.deleteNode(root, key);
        }
        else if(n==0)
            break;
        else
            cout<<"Wrong Choice"<<endl;
    }
    return 0;
}

```

## Output:

Enter 1 to INSERT data into the BST

Enter 2 to DISPLAY the BST in INORDER

Enter 3 to DELETE a value from the BST

Enter 0 to EXIT

Enter Your Choice.....

1

Enter the data....

55

Enter 1 to INSERT data into the BST

Enter 2 to DISPLAY the BST in INORDER

Enter 3 to DELETE a value from the BST

Enter 0 to EXIT

Enter Your Choice.....

1

Enter the data....

2

Enter 1 to INSERT data into the BST

Enter 2 to DISPLAY the BST in INORDER

Enter 3 to DELETE a value from the BST

Enter 0 to EXIT

Enter Your Choice.....

1

Enter the data....

4

Enter 1 to INSERT data into the BST

Enter 2 to DISPLAY the BST in INORDER

Enter 3 to DELETE a value from the BST

Enter 0 to EXIT

Enter Your Choice.....

1

Enter the data....

3

Enter 1 to INSERT data into the BST

Enter 2 to DISPLAY the BST in INORDER

Enter 3 to DELETE a value from the BST

Enter 0 to EXIT

Enter Your Choice.....

1

Enter the data....

59

Enter 1 to INSERT data into the BST

Enter 2 to DISPLAY the BST in INORDER

Enter 3 to DELETE a value from the BST

Enter 0 to EXIT

Enter Your Choice.....

1

Enter the data....

70

Enter 1 to INSERT data into the BST

Enter 2 to DISPLAY the BST in INORDER

Enter 3 to DELETE a value from the BST

Enter 0 to EXIT

Enter Your Choice.....

2

2 3 4 55 59 70

Enter 1 to INSERT data into the BST

Enter 2 to DISPLAY the BST in INORDER

Enter 3 to DELETE a value from the BST

Enter 0 to EXIT

Enter Your Choice.....

3

Enter the value to be deleted

55

Enter 1 to INSERT data into the BST

Enter 2 to DISPLAY the BST in INORDER

Enter 3 to DELETE a value from the BST

Enter 0 to EXIT

Enter Your Choice.....

2

2 3 4 59 70

Enter 1 to INSERT data into the BST

Enter 2 to DISPLAY the BST in INORDER

Enter 3 to DELETE a value from the BST

Enter 0 to EXIT

Enter Your Choice.....

0

## **Discussion :**

Binary Search Tree, is a node-based binary tree data structure which has the following

properties:

- 1.The left subtree of a node contains only nodes with keys lesser than the node's key.



2. The right subtree of a node contains only nodes with keys greater than the node's key.
3. The left and right subtree each must also be a binary search tree.

There must be no duplicate nodes.

In case of binary search trees (BST), Inorder traversal gives nodes in non-decreasing order.

To get nodes of BST in non-increasing order, a variation of Inorder traversal where Inorder

traversal is reversed can be used.

## 15 . Implementation of Simple Inheritance.

### Algorithm :

Algorithm for class cl1

Step1. Start

Step2. Declare following data type under 'protected' data member  
int a

Step3. Declare following function under 'public' data member  
void ip1(), void op1()

- Algorithm for void ip1()

Step1. Start

Step2. Print 'enter the value of n'

Step3. Scan a

Step4. End

- Algorithm for void op1()

Step1. Start

Step2. Print a= value of a

Step3. End

Step4. End of class cl1

- Algorithm for class cl2 -&gt; cl2 is inherit from cl1 publicly

Step1. Start

Step2. Declare following data type under 'protected' data member  
int b

Step3. Declare following function under 'public' data member  
void ip2(), void op2()

- Algorithm for void ip2()

Step1. Start

Step2. Print 'enter the value of b'

Step3. Scan b

Step4. End

- Algorithm for void op2()

Step1. Start

Step2. Print 'Single inheritance'

Step3. Print a= value of a

Step4. Print b= value of b

Step5. Print a + b= value of (a + b)

Step6. End

Step4. End of class cl2

- Algorithm for main function

Step1. Start

Step2. Declare object cl1

Step3. Declare object cl2

Step4. Call the function ip1()

Step5. Call the function op1()

Step6. Call the function ip2()

Step7. Call the function op2()

Step8. Return 0

Step9. End of main function

## Program Code :

```
#include<iostream>
using namespace std;
class cl1
{
    protected : int a;
    public :
        void ip1()
        {
            cout<<" Enter the value of a"<<endl;
            cin>>a;
        }
        void op1()
        {
            cout<<"a= "<<a<<endl;
        }
};
class cl2 : public cl1
{
    protected : int b;
    public :
        void ip2()
        {
            cout<<"Entyer the value of b "<<endl;
            cin>>b;
        }
        void op2()
        {
            cout<<"Single inheritance "<<endl;
            cout<<"a= "<<a<<endl;
            cout<<"b= "<<b<<endl;
            cout<<"a+b= "<<a+b<<endl;
        }
};
int main()
{
    cl1 ob1;
    cl2 ob2;
    ob2.ip1();
    ob2.op1();
    ob2.ip2();
    ob2.op2();
    return 0;
}
```

## Output:

Enter the value of a

10

a= 10

Enter the value of b

50

Single inheritance

a= 10

b= 50

a+b= 60

## Discussion :

The capability of a class to derive properties and characteristics from another class is called **Inheritance**. Inheritance is one of the most important features of Object-Oriented Programming.

**Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class.

**Super Class:** The class whose properties are inherited by sub class is called Base Class or Super class.

In single inheritance, a class is allowed to inherit from only one class. i.e. one sub class is

inherited by one base class only.

## 16 . Implementation of Multiple Inheritance

### Algorithm :

Algorithm for class cl1

Step1. Start

Step2. Declare following data type under 'protected' data member  
int a

Step3. Declare following function under 'public' data member  
void ip1(), void op1()

- Algorithm for void ip1()

Step1. Start

Step2. Print 'enter the value of a'

Step3. Scan a

Step4. End

- Algorithm for void op1()

Step1. Start

Step2. Print a= value of a

Step3. End

Step4. End of class cl1

- Algorithm for class cl2

Step1. Start

Step2. Declare following data type under 'protected' data member  
int b

Step3. Declare following function under 'public' data member  
Void ip2(),void op2()

- Algorithm for void ip2()

Step1. Start

Step2. Print 'enter the value of b'

Step3. Scan b

Step4. End

- Algorithm for void op2()

Step1. Start

Step2. Print b= value of b

Step3. End

Step4. End of class cl2

- Algorithm for class cl3 -> cl3 is inherit from cl1 & cl2 publicly

Step1. Start

Step2. Declare following function under 'public' data member  
void op3()

- Algorithm for void op3()

Step1. Start

Step2. Print value of 'a' inherited from cl1

a= value of a

Step3. Print value of 'b' inherited from cl1

b= value of b

Step4. Print sum= value of a + b

Step5. End

Step3. End of class cl3

- Algorithm for main function

Step1. Start

Step2. Declare object cl1, cl2, cl3

Step3. Call function ip1()

Step4. Call function ip2()

Step5. Call function op1()

Step6. Call function op2()

Step7. Call function op3()

Step8. return 0

Step9. End of main function

## Program Code :

```
#include<iostream>
using namespace std;
class cl1
{
    protected : int a;
    public :
        void ip1()
        {
            cout<<" Enter the value of a"<<endl;
            cin>>a;
        }
        void op1()
        {
            cout<<"a= "<<a<<endl;
        }
};
class cl2
{
    protected : int b;
    public :
        void ip2()
        {
            cout<<"Entyer the value of b "<<endl;
            cin>>b;
        }
        void op2()
        {
            cout<<"b= "<<b<<endl;
        }
};
class cl3 : public cl1, public cl2
{
    public :
        void op3()
        {
            cout<<"value of a inherited from cl1 \n"<<"a= "<<a<<endl;
            cout<<"value of b inherited from cl2 \n"<<"b= "<<b<<endl;
            cout<<"sum = "<<a+b<<endl;
        }
};
```

```

    }
};
int main()
{
    cl1 ob1; cl2 ob2;
    cl3 ob3;
    ob3.ip1();
    ob3.ip2();
    ob3.op1();
    ob3.op2();
    ob3.op3();
    return 0;
}

```

### Output:

Enter the value of a

50

Enter the value of b

60

a= 50

b= 60

value of a inherited from cl1

a= 50

value of b inherited from cl2

b= 60

sum = 110Enter the value of a

50

Enter the value of b

60

a= 50

b= 60

value of a inherited from cl1

a= 50

value of b inherited from cl2

b= 60

sum = 110

## **Discussion :**

The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object-Oriented Programming.

**Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class.

**Super Class:** The class whose properties are inherited by sub class is called Base Class or Super class.

Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes.

i.e. one **sub class** is inherited from more than one **base classes**.



