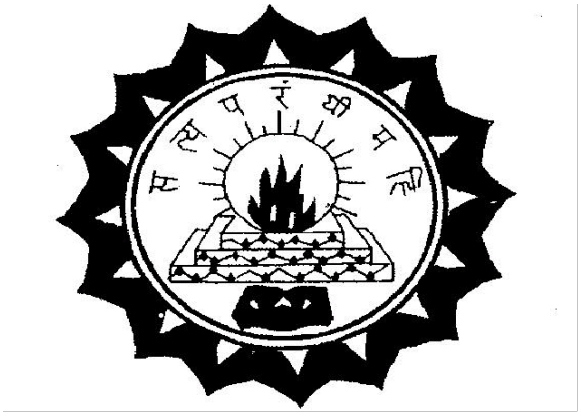
DWARAKA DOSS GOVERDHAN DOSS VAISHNAV EVENING COLLEGE (AUTONOMOUS)

ARUMBAKKAM, CHENNAI-600 106



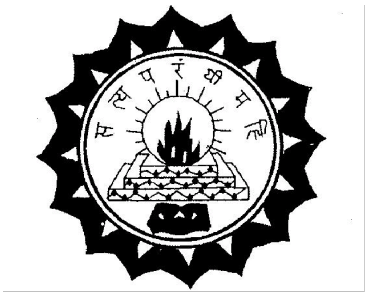
DEPARTMENT OF COMPUTER APPLICATIONS

Bachelor of Computer Applications

OBJECT ORIENTED PROGRAMMING LAB USING C++

DWARAKA DOSS GOVERDHAN DOSS VAISHNAV EVENING COLLEGE (AUTONOMOUS)

ARUMBAKKAM, CHENNAI-600 106



DEPARTMENT OF COMPUTER APPLICATIONS

OBJECT ORIENTED PROGRAMMING LAB USING C++

Certified that this is a record work done by ……. whose Register no. …… of I B.C.A., during the academic year 2018 – 2019.

Faculty In-charge **Head of the Department**

Submitted for practical examination held on …… at Dwaraka Doss Goverdhan Doss Vaishnav College, Chennai-106.

Internal Examiner **External Examiner**

Sl. No.	Date	Program Name	Page No	Signature
		Usage of Operators		
1.		Arithmetic Operators, Bitwise Operators, Ternary Operator, sizeof Operator		
2.		Mathematical Library Functions		
		Selection Structures		
3.		Relation between two numbers using simple-if		
4.		Check whether the given number is positive, negative or zero using if-else		
5.		Largest of 3 given numbers using nested-if		
6.		Print the result of a student using else-if ladder		
7.		Display the number of days in the given month using switch-case		
		Looping Structures		
8.		Reverse the given number and find out the sum of digits using while loop		
9.		Check whether the given number is prime or not using do-while loop		
10.		Generate Fibonacci Series using for loop		

11.		Inline function		
12.		Function Overloading		
13.		Class and Objects		
14.		Constructor and Destructor		
15.		Static data members and member functions		
16.		Passing Objects to functions		
17.		Binary Operator Overloading		
		Inheritance		
18.		Single Inheritance		

19.		Multilevel Inheritance		
20.		Multiple Inheritance		
21.		Virtual Function		

Ex.No: 1

Date:

PROGRAM USING OPERATORS

Aim: To write a C++ program to illustrate the usage of various operators.

Algorithm:

1. Start the program.

2. Read two integers num1 and num2.

3. Perform all arithmetic operators on them and print the result.

4. Perform bitwise AND, OR, One's complement, shift left and shift right.

5. Apply postincrement and preincrement on those numbers num1 and num2 and print the answer.

6. Apply the special operator sizeof() on the datatypes and check the bytes occupied.

7. Check the number num1 is odd or even by conditional or ternary operator.

8. Stop the program.

Program:

```
//Program using operators
#include<iostream>
using namespace std;
int main()
{
cout<<"\n\t\tTYPES OF OPERATORS";
cout<<"\n\t\t----------------";
//Arithmetic operators
int num1,num2,sum,sub,mult,div,mod;
cout<<"\nEnter the first number:";cin>>num1;
cout<<"\n Enter the second number:";cin>>num2;
cout<<"\n num1="<<num1<<" num2="<<num2;
sum = num1 + num2;
sub = num1 - num2;
mult = num1 * num2;
div = num1 / num2;
mod = num1 % num2;
cout<<"\n\nAddition is :"<<sum;
cout<<"\nSubtraction is :"<<sub;
cout<<"\nMultiplication is :"<<mult;
cout<<"\nDivision is :"<<div;
cout<<"\nModulus is :"<<mod;
//Bitwise Operators
```

```
cout<<"\n\nBitwise AND="<<(num1&num2);

cout<<"\nBitwise OR="<<(num1|num2);

cout<<"\nBitwise XOR="<<(num1^num2);

cout<<"\nBitwise Complement of num1="<<(~num1);

cout<<"\nRight Shift num1 by 2 bits="<<(num1>>2);

cout<<"\nLeft Shift num2 by 2 bits="<<(num2<<2);

//sizeof operator

cout<< "\n\n char:" <<sizeof(char);

cout<< "\tshort:" <<sizeof(short);

cout<< "\tint:" <<sizeof(int);;

cout<< "\tlong:" <<sizeof(long);

cout<< "\tfloat:" <<sizeof(float);

cout<< "\tdouble:" <<sizeof(double);

//ternary operator

num1%2==0?cout<<"\n\n"<<num1<<" is even":cout<<"\n\n"<<num2<<" is odd";

//Increment and Decrement Operators

cout<<"\n\nPostincrement of num1="<<num1++;

cout<<"\nPreincrement of num2="<<++num2;

return 0;

}
```

Input and Output:

```
Enter the first number : 3
Enter the second number : 5
```

```
num1 = 3  num2 = 5
Addition is : 8
Subtraction is : -2
Multiplication : +15
Division : 0
Modulus : 3
Bitwise AND =1
Bitwise OR = 7
Bitwise XOR = 6
Bitwise complement of num 1 = -4
Right shift num 1 by 2 bits = 0
Left shift num 2 by 2 bits = 20
Char : 1  short : 2  int:4  long:4  float:4  double:8
3 is odd
Post increment of num 1 = 3
Pre increment of num 2 = 6
```

Result:

Hence program using C++ operators is verified.

Ex. No: 2

Date:

PROGRAM USING MATHEMATICAL LIBRARY FUNCTIONS

Aim: To write a C++ program to show the functioning of various mathematical library functions.

Algorithm:

1. Start the program.

2. Read an integer number num.

3. Find the square root of number using sqrt() function.

4. Read a float number x.

5. Apply the functions like floor(),ceil() on x to round the value of x.

6. Print the absolute value of x irrespective of the sign using fabs() function.

7. Find sine, cosine and tangent values of x using sin(),cos() and tan() respectively.

8. Print the natural and logarithm of x using log() and log10()/

9. Find x^y using pow() function.

10. Stop the program.

Program:

```
// Mathematical Library Functions
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
intnum;
cout<<"\n\t\tMATHEMATICAL LIBRARY FUNCTIONS";
cout<<"\n\t\t----------------------------";
cout<<"\nEnter an integer:";cin>>num;
cout<<"\nSquareroot of num1="<<sqrt(num);
float x;
cout<<"\nEnter a float number:";cin>>x;
cout<<"\nFloor of x="<<floor(x);
cout<<"\nCeil of x="<<ceil(x);
cout<<"\nAbsolute value of x="<<fabs(x);
cout<<"\nSine value of x="<<sin(x);
cout<<"\nCosine value of x="<<cos(x);
cout<<"\nTan value of x="<<tan(x);
cout<<"\nNatural logarithm of x="<<log(x);
cout<<"\nLogarithm of x to base 10="<<log10(x);
cout<<"\nPower of 4 to 5="<<pow(4,5);
return 0;
```

}

Input and Output:

MATHEMATICAL LIBRARY FUNCTON

Enter an integer : 9
Square root of num 1 = 3
Enter a float number =4.1
Floor of x=4 ceil of x=5
Absolute value of x=4.1
Sine value of x=-0.818277
Cosine value of x = 0.574824
tan value of x =1.42353
natural logarithm of x =1.41099 logarithm of x to base10=0.612784
power of 4 to 5=1024

Result:

Hence program using mathematical library is verified.

Ex. No: 3

Date:

PROGRAM USING SIMPLE IF

Aim: To write a C++ program using simple if statement.

Algorithm:

1. Start the program.
2. Read two integers: num1 and num2.
3. Check using if statement whether num1>num2 or num2<num1 or num1=num2.
4. Display the output as “num1 is greater than num2” or “num2 is greater than num1” or “num1 is equal to num2” accordingly.
5. Stop the program.

Program:

```
// Usage of simple if to find relation between 2 numbers
#include<iostream>
using namespace std;
int main()
{
int num1,num2;
cout<<"\n\t\tRELATION BETWEEN 2 NUMBERS";
cout<<"\n\t\t-----------------------";
```

```
cout<<"\n Enter 2 integers:";

cin>>num1>>num2;

if (num1==num2)

cout<<"Both numbers are equal (ie) "<<num1;

if (num1>num2)

cout<<num1<<" is greater than "<<num2;

if (num2>num1)

cout<<num2<<" is greater than "<<num1;

return 0;

}
```

Input and Output:

RELATION BETWEEN 2 NUMBERS
Enter 2 integers:5
8
8 is greater than 5

Result:

Hence program using simple if is verified.

Ex. No: 4

Date:

PROGRAM USING IF-ELSE

Aim: To write a C++ program using if-else statement.

Algorithm:

1. Start the program.
2. Read an integer namely ‘num’.
3. If num is greater than zero, display the number as positive.
4. If num is lesser than zero, display the number as negative.
5. If num is equal than zero, display the number as zero.
6. Stop the program.

Program:

```
// Usage of if else to check the number is +ive,-ive or zero
#include<iostream>
using namespace std;
int main()
{
```

```
floatnum;

cout<<"\n\t\tCHECK WHETHER THE GIVEN NUMBER IS POSITIVE, NEGATIVE OR ZERO";

cout<<"\n\t\t-------------------------------------------------------";

cout<<"\n Enter a number:";

cin>>num;

if (num>0)

cout<<"The given number is positive "<<num;

else

if (num<0)

cout<<"The given number is negative "<<num;

else

        cout<<"The given number is zero "<<num;

return 0;

}
```

Input and Output:

Check whether the given number is positive or negative or zero
Enter a number : -2
The given number is negative -2

Result:

Hence program using If-else is verified

Ex. No: 5

Date:

PROGRAM USING NESTED IF

Aim: To write a C++ program using nested if statement.

Algorithm:

1. Start the program.
2. Read three integers: num1, num2, num3.
3. If num1 is greater than num2, goto Step-4 else goto Step-5.
4. If num1 is greater than num3, assign big=num1 else assign big=num3. Goto Step-6.
5. If num2 is greater than num3, assign big=num2 else assign big=num3. Goto Step-6.
6. Display the value of big.
7. Stop the program.

Program:

```
//Usage of nested-if to find biggest of 3 numbers
#include<iostream>
using namespace std;
int main()
```

```
{
float num1,num2,num3,big;
cout<<"\n\t\tBIGGEST OF 3 NUMBERS";
cout<<"\n\t\t--------------------";
cout<<"\nEnter 3 numbers:";
cin>>num1>>num2>>num3;
if (num1>=num2)
{
if (num1>=num3)
        big=num1;
else
        big=num3;
}
else
if (num2>=num3)
        big=num2;
else
        big=num3;
cout<<"The biggest among the 3 given numbers is "<<big;
return 0;
}
```

Input and Output:

BIGGEST OF 3 NUMBERS

Enter 3 number :1.3
9.5
12.7
The Biggest among the given 3 is 12.7

Result:

Hence program using nested if is verified.

Ex.No:6

Date:

PROGRAM USING ELSE IF LADDER

Aim: To write a C++ program using else if statement.

Algorithm:

1. Start the program.
2. Read student's roll number, name and percentage.
3. If percentage is greater than or equal to 90, assign grade='D'.
4. If percentage is greater than or equal to 80 but less than 90, assign grade='A'.
5. If percentage is greater than or equal to 70 but less than 80, assign grade='B'.
6. If percentage is less than 70, assign grade='C'.
7. Display all the details of student like roll number, name, percentage and grade.
8. Stop the program.

Program:

```
//Usage of else-if ladder to display grade of a student
#include<iostream>
using namespace std;
int main()
```

```
{
intnum;
char name[20];
float per;
char grade;
cout<<"\n\t\tCALCULATING THE STUDENT GRADE";
cout<<"\n\t\t----------------------------";
cout<<"\nEnter the student's rollnumber:";
cin>>num;
cout<<"\nEnter the student's name:";
cin>>name;
cout<<"\nEnter the student's percentage:";
cin>>per;
if (per>=90)
        grade='D';
else if (per>=80 && per<90)
        grade='A';
else if  (per>=70 && per<80)
        grade='B';
else
        grade='C';
cout<<"\nTHE STUDENT'S ROLLNUMBER="<<num;
cout<<"\nTHE STUDENT'S NAME="<<name;
cout<<"\nTHE STUDENT'S PERCENTAGE="<<per;
```

```
cout<<"\nTHE STUDENT'S GRADE="<<grade;

return 0;

}
```

Input and Output:

CALCULATING THE STUDENTS GRADE
Enter the student's roll number:77291
Enter the student's Name: K.VYSHALI
Enter the student's percentage: 83

The students roll number=77291
The students name= K.VYSHALI
The students percentage=83
The students grade=A

Result:

Hence program using Else if ladder is verified.

Ex.No: 7

Date:

PROGRAM USING SWITCH CASE

Aim: To write a C++ program using switch case statement.

Algorithm:

1. Start the program.
2. Read month from 1 to 12 and year in the format YYYY.
3. If month is 1,3,5,7,8,10, display the number of days as 31.
4. If month is 4,6,9,11, display the number of days as 30.
5. If month is 2, check whether the year is leap year or not. If leap year, display number of days as 29 else number of days as 28.
6. If month is not within 1 to 12, display as wrong month.
7. Stop the program.

Program:

```
//Usage of switch case to display days of a month
#include<iostream>
using namespace std;
int main()
{
```

```
intmonth,year;
cout<<"\n\t\tNUMBER OF DAYS IN A MONTH";
cout<<"\n\t\t----------------------";
cout<<"\nEnter the month between 1 to 12:";
cin>>month;
cout<<"\nEnter year as YYYY format:";
cin>>year;
switch (month)
{
case 1:
case 3:
case 5:
case 7:
case 8:
case 10:
case 12:cout<<month<<" has 31 days";break;
case 4:
case 6:
case 9:
case 11: cout<<month<<" has 30 days";break;
case 2: if (year%4==0)
        cout<<month<<" has 29 days since "<<year<<" is a leap year";
else
        cout<<month<<" has 28 days since "<<year<<" is a not a leap year";
```

```
break;

default:cout<<"Wrong month. Please retry!";

}

return 0;

}
```

Input and Output:

NUMBER OF DAYS IN A MONTH
Enter a month between 1 to 12 : 11
Enter a year as YYYY format : 1996
11 has 30 days.

Result:

Hence program using switch case is verified.

Ex.No: 8

Date:

PROGRAM USING WHILE LOOP

Aim: To write a C++ program using while statement.

Algorithm:

1. Start the program.
2. Read a positive integer num from user.
3. If num is greater than 0, divide number by 10 else goto Step-8.
4. Store the remainder of the division in rem and display rem.
5. Add rem to sum.
6. Assign the quotient to num.
7. Goto Step -3.
8. Display the value of sum.
9. Stop the program.

Program:

```
// Program to reverse a number and sum of digits using while loop
#include<iostream>
#include<cstdlib>
```

```
using namespace std;
int main()
{
intnum,sum=0,rem;
cout<<"\n\t\tREVERSE THE NUMBER AND FIND THE SUM OF DIGITS";
cout<<"\n\t\t----------------------------------------";
cout<<"\n\nEnter an Positive Integer:";
cin>>num;
if (num<0)
{
cout<<"Please enter only positive integer. Try again!”;
exit(0);
}
cout<<"\nReverse of the number=";
while (num>0)
{
rem=num%10;
sum+=rem;
cout<<rem;
num/=10;
}
cout<<"\nSum of the digits="<<sum;
return 0;
}
```

Input and Output:

REVERSE THE NUMBER AND FIND THE SUM OF DIGITS

Enter a positive number : 25
Reverse of the number : 52
Sum of the digits = 2.

Result:

Hence program using while loop is verified.

Ex.No: 9

Date:

PROGRAM USING DO-WHILE LOOP

Aim: To write a C++ program using do-while statement.

Algorithm:

1. Start the program.
2. Read the number to be checked in num.
3. Divide the number num from 2 to num-1.
4. If any number from 2 to num-1 divides num by giving remainder as 0, display the number num is not a prime number and goto Step-6.
5. If no number from 2 to num-1 divides num, display the number num is a prime number.
6. Stop the program.

Program:

```
//Program to check prime or not using do-while loop
#include<iostream>
using namespace std;
int main()
```

```
{
intnum,rem,flag=0,i;
cout<<"\n\t\tCHECK WHETHER THE GIVEN NUMBER IS PRIME OR NOT";
cout<<"\n\t\t----------------------------------------";
cout<<"\nEnter the number to be checked:";
cin>>num;
i=2;
do
{
rem=num%i;
if (rem==0)
  {
        flag=1;
        break;
}
i++;
}
while (i<num);
if (flag==0)
cout<<num<<" is a prime number";
else
cout<<num<<" is not a prime number";
return 0;
}
```

Input and Output:

CHECK WHETHER THE GIVEN NUMBER IS PRIME OR NOT
Enter the number to be checked : 15
15 is not a prime number

Result:
Hence program using Do-While loop is verified.

Ex.No: 10

Date:

PROGRAM USING FOR LOOP

Aim: To write a C++ program using for statement.

Algorithm:

1. Start the program.
2. Read the range of Fibonacci series as n.
3. Initialize f1 as -1 and f2 as 1.
4. Execute the Steps from 5 to 7 using for loop from 1 to n and then goto Step-8.
5. Add f1 and f2 and assign to f3.
6. Display f3.
7. Reassign f2 to f1 and f3 to f2.
8. Stop the program.

Program:

```
//Program using for loop
#include<iostream>
using namespace std;
int main()
```

```
{
int f1=-1,f2=1,f3,n,i;
cout<<"\n\t\tGENERATION OF FIBONACCI SERIES";
cout<<"\n\t\t----------------------------";
cout<<"\nEnter the range of fibonacci series:";
cin>>n;
cout<<"\nThefibonacci series is\n";
for(i=1;i<=n;i++)
 {
        f3=f1+f2;
        cout<<f3<<endl;
        f1=f2;
        f2=f3;
 }
return 0;
}
```

Input and Output:

GENERATION OF FIBONACCI SERIES
Enter the range of fibonacci series: 10
The fibonacci series is
0
1
1
2

3
5
8
13
21
34

Result:

Hence program using for loop is verified.

Ex.No:11

Date:

INLINE FUNCTION

Aim:

To write a C++ program using the concept of inline function.

Algorithm:

Step 1: Declare a function for cube
Step 2: Write member functions to find the cube of a number
Step 3: print the result.

Program:

```
#include<iostream>
using namespace std;
inline double cube(double);
int main()
{
double x;
cout<<"\nEnter a number:";
cin>>x;
x=cube(x);
cout<<"\nCube of x is:"<<x;
return 0;
}
double cube(double x)
{
return x*x*x;
}
```

Input and Output:

```
Enter a number: 2
Cube of x is: 8
```

Result:

Hence program using inline function is verified.

Ex.No:12

Date:

FUNCTION OVERLOADING

Aim:

To write a C++ program using the concept of function overloading

Algorithm:

Step 1: Start the program

Step 2: Declare a function for volume with different parameters

Step 3: Write member functions to

1) find the volume of cube
2) find the volume of circle
3) find the volume of rectangle

Step 4: print the result.

Step 5: Stop the program

Program:

```
#include<iostream>
using namespace std;
int volume(int);
double volume(double,int);
long volume(long,int,int);
```

```
int main()
{
cout<<volume(10)<<”\n”;
cout<<volume(2.5,8)<<”\n”;
cout<<volume(1001,75,15)<<”\n”;
return 0;
}
int volume(int s)
{
return(s*s*s);
}
double volume(double r,int h)
{
return(3.14519*r*r*h);
}
long volume(long l,int b,int h)
{
return(l*b*h);
}
```

Input and Output:

```
1000
157.26
112500
```

Result:

Hence program using function overloading is verified.

EX.NO:13
DATE:

CLASS AND OBJECTS

Aim:

To write a C++ program using the concept of class and object.

Algorithm:

Step 1: Define a class “employee” with data membersemployee number, employee name,basic pay, da, it and net salary.
Step 2: Write member functions to
1) Read the data members and perform pay calculations
2) Display the result
Step 3: Create objects for the above class and print the result.

Program:

```
#include<iostream>
using namespace std;
classEmployee
{
intempno;
charempname[10];
floatbasic,da,it,ns;
public :
void read(),display();
};
voidEmployee::read()
 {
cout<<"\nEnter employee name : ";
cin>>empname;
cout<<"\nEnter employee number : ";
cin>>empno;
cout<<"\nEnter basic salary : ";
cin>>basic;
da=basic*0.52;
it=basic*0.30;
ns=basic+da-it;
 }

voidEmployee::display()
```

```
 {
cout<<"\n "<<empno;
cout<<" "<<empname;
cout<<" "<<basic;
cout<<" "<<da;
cout<<" "<<it;
cout<<" "<<ns;
 }
int main()
{
Employee e1,e2;
e1.read();
e2.read();
cout<<" Emp_no Name Basic DA IT Net Salary\n";
cout<<"-------------------------------------------\n";
e1.display();
e2.display();
cout<<"\n-------------------------------------------";
return 0;
}
```

**********OUTPUT**********

```
Enter employee name : mina
Enter employee number : 84
Enter basic salary : 500

Enter employee name :anju
Enter employee number : 69
Enter basic salary : 600

Emp_no Name Basic DA IT Net Salary
-----------------------------------------
 84 mina 500 260 150 610
 69 anju 600 312 180 732
-----------------------------------------
```

Result:

Thus a C++ program using class and objects has been executed successfully.

EX. NO: 14
DATE:

CONSTRUCTOR AND DESTRUCTOR

Aim:

To write a C++ program using constructor and destructor

Algorithm:

Step 1: Define a class “Example” with 2 integers a,b as data members.
Step2: Define 2 constructors – One default and one parameterized constructor to initialize two data members.
Step 3: Define a destructor for the same class.
Step 4: Create objects and print the results.

Program:

```
#include<iostream>
using namespace std;
class Example
 {
// Data members Declaration
inta,b;
public:
//Constructor without Argument
Example()
{
a=50;
b=100;
cout<<"\n I am Default Constructor";
}
//Constructor with Argument
Example(intx,inty)
{
a=x;
b=y;
cout<<"\nI am Parameterized Constructor";
}
void display()
```

```
{
        cout<<"\nValues :"<<a<<"\t"<<b;
}
        // Destructor
~Example()
{
        cout<<”\n I am Destructor”;
}
};

int main()
{
// Parameterized Constructor invoked
Example object1(10,20);
// Default Constructor invoked
Example object2;
object1.display();
object2.display();
return 0;
}
```

**********OUTPUT**********

```
I am Parameterized Constructor
I am Default Constructor
Values :      10            20
Values :      50            100
I am Destructor
I am Destructor
```

Result:

Thus a C++ program for constructor and destructor has been executed successfully.

EX.NO:15
DATE:

STATIC DATA MEMBER AND STATIC MEMBER FUNCTION

Aim:

To write a C++ program for studying the working of static data members and static member functions.

Algorithm:

Step 1: Declare a class BOX with data members length, breadth and height.

Program:

```
#include <iostream>
using namespace std;
class Box
{  floatl,b,h;
public:
        staticint count;
        void get()
          {
                cout<<"Enter length, breadth and height of the box:";
                cin>>l>>b>>h;
                // Increase every time when object is created
                count++;
        }
        float volume()
        {
                return (l*b*h);
        }
        staticintgetcount()
        {
                return count;
        }
};



// Initialize static member of class Box
```

```
int Box::count = 0;

int main()
{

        // Print total number of objects before creating object.
        cout<< "Inital Stage Count: " << Box::getcount() <<endl;

        Box b1,b2;
        b1.get();
        b2.get();

        cout<<"\n The volume of Box 1="<<b1.volume();
        cout<<"\n The volume of Box 2="<<b2.volume();

        // Print total number of objects after creating object.
        cout<< "\nFinal Stage Count: " << Box::getcount() <<endl;
        return 0;
}
```

**********OUTPUT**********

```
Initial Stage Count: 0
Enter length, breadth and height of the box: 2.5 3.6 7.8
Enter length, breadth and height of the box: 1.2 5.9 10.2

The volume of Box 1= 70.2
The volume of Box 2= 72.216
Final Stage Count: 2
```

Result:

Thus a C++ program using static data members and member functions has been executed.

EX.NO:16
DATE:

PASSING OBJECTS TO FUNCTION

Aim:

To write a C++ program for studying the working of passing objects to functions

Algorithm:

Step 1: Declare a class timeof with data hours and minutes.
Step 2: Calculate hours and minutes
Step 3: Print the results.

Program:

```
#include<iostream>
using namespace std;
class timeof
{
private:
int hours,minutes;
public:
void gettime(int h,int m)
{
hours=h;
minutes=m;
}
void puttime(void)
{
cout<<hours<<”hours”;
cout<<minutes<<”minutes”;
}
void sum(timeof,timeof);
};
```

```
void sum(timeof T1,timeof T2)
{
minutes=T1.minutes+T2.minutes;
hours=minutes/60;
minutes=minutes%60;
hours=hours+T1.hours+T2.hours;
}
int main()
{
timeof a,b,c;
a.gettime(2,15);
b.gettime(3,55);
c.sum(a,b);
cout<<"a=";
a.puttime();
cout<<"b=";
b.puttime();
cout<<"c=";
c.puttime();
return 0;
}
```

**********OUTPUT**********

```
a=2hours15minutes
b=3hours55minutes
c=6hours10minutes
```

Result:

Thus a C++ program using passing objects to functions has been executed.

EX.NO:17
DATE:

BINARY OPERATOR OVERLOADING

Aim:

To write a C++ program for matrix addition by overloading "+" operator.

Algorithm:

Step 1: Define a class Complex with two variables x and y for storing real and imaginary values of a complex number.
Step 2: Read two complex numbers.
Step 3: Define an operator member function +for adding two complex numbers.
Step 4: Define an operator friend function - for subtracting two complex numbers.
Step 5: Display the result in complex number format.

Program:

```
//Binary operator overaloding
#include<iostream>
using namespace std;
classComplex
{
        floatx,y;
public:
        voidgetdata(),putdata();
        Complex operator +(Complex);
        friendComplex operator –(Complex,Complex);
};



void Complex::getdata()
        {
                cout<<"Enter real and imaginary values of a complex number:";
                cin>>x>>y;
        }
```

```
void Complex::putdata()
	{
		if (y>0)
			cout<<x<<"+j"<<y<<endl;
		else
		{
			y=-y;
			cout<<x<<"-j"<<y<<endl;
		}
	}
Complex Complex::operator +(Complex c)
{
	Complex t;
	t.x=x+c.x;
	t.y=y+c.y;
	return t;
}
Complex operator -(Complex t1,Complex t2)
{
	Complex t3;
	t3.x=t1.x-t2.x;
	t3.y=t1.y-t2.y;
	return t3;
}

int main()
{
	Complex c1,c2,c3;
	c1.getdata();
	c2.getdata();
	c3=c1+c2;
	cout<<"\n ADDITION OF 2 COMPLEX NUMBERS \n";
	c3.putdata();
	c3=c1-c2;
	cout<<"\n SUBTRACTION OF 2 COMPLEX NUMBERS \n";
	c3.putdata();
	return 0;
}
```

**********OUTPUT**********
Enter real and imaginary values of a complex number: 3.5 -2.6
Enter real and imaginary values of a complex number: -6.4 1.3

ADDITION OF 2 COMPLEX NUMBERS
-2.9-j1.3
SUBTRACTION OF 2 COMPLEX NUMBERS
9.9-j3.9

Result:

Thus a C++ program for addition and subtraction of two complex numbers in support of binary operator overloading has been executed successfully.

EX.NO:18

DATE:

SINGLE INHERITANCE

Aim:

To write a C++ program for single inheritance.

Algorithm:

Step 1: Definethe base class Student with roll number and name.
Step 2: Invoke the derived class Marks which is inherited by the class Student with 3 marks and total.

Step 3: Create an object stud for the derived class Marks.

Step 4: Call all the member functions through stud object and display all the details.

Program:

```
#include<iostream>
using namespace std;
 classStudent
{
        intrno;
        char name[20];
        public:
                void getdata1();
                void display1();
};
```

```
voidStudent::getdata1()
        {
        cout<<"\nEnter RollNumber : ";
        cin>>rno;
        cout<<"\nEnterStudent Name : ";
        cin>>name;
        }



voidStudent::display1()
        {
        cout<<"\n Roll Number          ="<<rno;
        cout<<"\n Student Name="<<name;
        }
classMarks : public Student
{
int m1,m2,m3,tot;
public:
        void getdata2();
        void display2();
};

void Marks::getdata2()
{
        getdata1();
        cout<<"Enter Mark1 : \t";
        cin>>m1;
        cout<<"Enter Mark2 : \t";
        cin>>m2;
        cout<<"Enter Mark3 : \t";
        cin>>m3;
        tot=m1+m2+m3;
}
voidMarks::display2()
{
        display1();
        cout<<"\n Mark1="<<m1;
        cout<<"\n Mark2="<<m2;
        cout<<"\n Mark3="<<m3;
```

```
        cout<<"\n Total   ="<<tot;
}
};
int main()
{
        Marks std;
        std.getdata2();
        std.display2();
        return 0;
}
```

**********OUTPUT**********

```
Enter Roll Number:101
Enter Student Name:Anitha
Enter Mark1:40
Enter Mark2:65
Enter Mark3:59

Roll Number    = 101
Student Name = Anitha
Mark1 = 40
Mark2 = 65
Mark3 = 59
Total    = 164
```

Result:

Thus a C++ program for implementing single inheritance has been executed successfully.

EX.NO:19

DATE:

MULTILEVEL INHERITANCE

Aim:

To have a program to illustrate multilevel inheritance.

Algorithm:

Step 1: Invoke the base class Student
Step 2: Invoke the derived class Test which is inherited by the class Student
Step 3: Invoke the derived class Result which is inherited by the class Test
Step 4: Create an object r for the Result class
Step 5: Call the member functions using object r and display the results

Program:

```
#include<iostream>

using namespace std;

class Student // Base Class

{

protected:

introllno;

char *sname;
```

```
public:
voidgetdata(intnum,char *name)
{
rollno = num;
sname = name;
}
voidputdata(void)
{
cout<< " The Name of Student \t: "<<sname<<endl;
cout<< " The Roll No. is \t: "<<rollno<<endl;
}
};



classTest:public Student // Derived Class 1
{
protected:
int mark1,mark2;
public:
voidgettest(int m1,int m2)
{
```

```
mark1=m1;
mark2=m2;
}
voidputtest()
{
cout<< " Marks in Music is \t: "<< mark1<<endl;
cout<< " Marks in Drawing is \t: "<< mark2<<endl;
}
};

classResult:public Test // Derived Class 2
{
protected:
float total;
public:
voiddisplayresult(void)
{
total = mark1 + mark2;
putdata();
puttest();
cout<< " Total of  the Two \t: "<< total<<endl;
}
```

```
};


int main()
{
intx,y,z;
char n[20];
Result r;
cout<<"Enter the Roll number:";
cin>>x;
cout<<"Enter the name:";
cin>>n;
cout<<"Enter the marks in Music:";
cin>>y;
cout<"Enter the marks in Drawing:";
cin>>z;
r.getdata(x,n);
r.gettest(y,z);
cout<<endl<< "********** RESULT **********"<<endl;
r.displayresult();
return 0;
}
```

Input and Output:

**********OUTPUT**********

Enter the Roll number: 138957

Enter the name: Sunitha

Enter the marks in Music: 76

Enter the marks in Drawing: 89

**********RESULT**********

The Name of Student : Sunitha

The Roll No. is : 138957

Marks in Music is : 76

Marks in Drawing is : 89

Total of the two is : 165

Result:

Thus a C++ program for multilevel inheritance has been executed successfully.

EX. NO:20
DATE:

MULTIPLE INHERITANCE

Aim:

To write a C++ program for illustrating the concept of multiple inheritance.

Algorithm:

Step 1: Define a class “A” with name and number as data members.
Step 2: Define a class “B” with mobile number as data member.
Step 3: Define a class “C” with mail address as the data member derived from the base classes as both “a” and “b”.
Step 4: Define the member functions to read and write all the data members.

Program:

```
#include<iostream>
using namespace std;
// Base class 1
classA
{
        char name[25];
        intnum;
public:
        voidgetdata();
        void display();
};

// Base class 2
classB
{
        longlongintmobile;
        public:
        voidgetdata();
        void display();
};
```

```
//Derived class
classC : public A, public B
{
        char mail[50];
public:
        voidgetdata();
        void display();
};

void A::getdata()
{
        cout<<"Enter yourName: ";
        cin>>name;
        cout<<"Enter yourRollNumber :";
        cin>>num;
}

voidA::display()
{
        cout<<” \nName=”<<name<<” \nRoll number=”<<num;
 }

voidB::getdata()
{
        cout<<"Enter yourMobileNumber: ";
        cin>>mobile;
}

voidB::display()
{
cout<<” \nMobile Number=”<<mobile;
}

voidC::getdata()
{
A::getdata();
B::getdata();
cout<<"Enter yourE-mail Address: ";
cin>>mail;
}
```

```
voidC::display()
{
A::display();
B::display();
cout<<"\nE-Mail address= "<<mail;
}

int main()
{
Cobject;
object.getdata();
object.display();
return 0;
}
```

**********OUTPUT**********

```
Enter yourName: Kumar
Enter your RollNumber: 131201
Enter your Mobile Number:9667743322
Enter yourE-mail Address:kumardgvc@yahoo.com

Name= Kumar
Roll number=131201
Mobile Number=966774322
E-Mail Address= kumardgvc@yahoo.com
```

Result:

Thus a C++ program for implementing multiple inheritance has been executed successfully.

EX.NO:14
DATE:

VIRTUAL FUNCTIONS

Aim:

To write a C++ program for virtual functions.

Algorithm:

Step 1: Define a class “Shape” with a pure virtual function display_area.
Step 2: Create two derived class of Shape ----Triangle and Rectangle .
Step 3: Read length and breadth of a rectangle and calculate area as length*breadth.
Step 4: Read base and height of a triangle and calculate area as ½*base*height.
Step 5: Create objects for the two classes and invoke the virtual function by pointer object of base class Shape.

Program:

```
#include<iostream>
using namespace std;
classShape
{
protected:
        double x, y;
public:
        voidgetdata(double a, double b)
        {
                x=a;
                y=b;
        }
        virtual void display_area()=0;
};
```

```
classTriangle:publicShape
{
        double area;
public:
        voiddisplay_area()
        {
        area=(x*y)/2;
        cout<<"Area of triangle is:"<<area<<endl;
        }
};

classRectangle:public shape
{
        double area;
public:
        voiddisplay_area()
        {
        area=x*y;
        cout<<"Area of rectangle is:"<<area;
        }
};

int main()
{
Shape *p;
Triangle t;
Rectangle r;
p=&t;
p->getdata(10,30);
p->display_area();
p=&r;
p->getdata(20,30);
p->display_area();
return 0;
}
```

Input and output :

**********OUTPUT**********

Area of triangle is: 150

Area of rectangle is: 600

Result:

Thus a C++ program for computing area of shapes using virtual functions has been executed successfully.