EXERCISE-1

Aim:

a) Write a Simple Program on printing “Hello World”

Description: In this by using cout object we can print the output

Algorithm:

1. Start
2. Print Hello World using cout Object
3. end

Sample Input and output:

Hello world

Aim:

b) C++ Program to take input string from user and print it

Description: This Program for is Described How to use cout and cin statements

Algorithm:

1. Start
2. Print Enter your name
3. Read name
4. Print name

Sample Input and output:

Enter name:

Yugandhar

Hello Yugandhar

Aim:

c) Write a C++ Program to convert any two programs that are written in C into C++

i) Write a C++ Program for given year is leap year or not

Description: this Program Describes about how to use the control statements in C++

Algorithm:

1. Start
2. Read year
3. If year%4 =0
   3.1 print leap Year
4. else
   4.1 print not Leap year
5. end

Sample Input and Output:

Enter year

2000

Leap year

ii) Write a C++ program to print factorial of a number

Description: this Program Describes about how to use the conditional and loop control statements in C++

Algorithm:

1. **Start**
2. Declare t n,i,fact=1cout<<"enter no ";
3. Read n
4. for(i=1;i<=n;i++) do
   4.1 fact=fact*i;
   4.2 cout<<"the factorial of a no is"<<fact;
5. End for
6. **Stop**

Sample Input and output:

Enter no:

5

The factorial of a no is120

Viva-Voce:

1. Explain evolution of C++
2. What are the Steps for executing C++ Program
3. What are the control statements in C++
4. What the Key Concepts are of object oriented programming?

Conclusion:

Student can understand basics of the computer programming and implements the applications on Conditional control statements and repetition and this program attained with co1 and mapped with the PO1, PO2, PO3, PSO1, and PSO2**.**

EXERCISE-2

Aim:

a) Write a Program that computes the simple interest and compound interest payable on principal amount (in Rs.) of loan borrowed by the customer from a bank for a giver period of time (in years) at specific rate of interest. Further determine whether the bank will benefit by charging simple interest or compound interest.

Description: This program describes how to use Else-If Ladder in C++

Algorithm:

1. Start
2. Read n
3. if((n>=1)&&(n<=50))
   3.1 cout<<"bus fare is "<<n*5<<"rupees";
4. else if((n>=51)&&(n<=100))
   4.1 cout<<"bus fare is "<<n*10<<"rupees";
5. else if((n>=101)&&(n<=150))
   5.1 cout<<"bus fare is "<<n*15<<"rupees";
6. else
   6.1 cout<<"bus fare is "<<n*20<<"rupees";
7. end

Sample Input and output:

What distance will u travel

200

bus fare is4000

Aim:

b) Write a Program to calculate the fare forth e pass enters traveling in a bus. When a Passenger enters the bus, the conductor asks "What distance will you travel?" On knowing distance from passenger (as an approximate integer), the conductor mentions the fare to the passenger according to following criteria**.**

Description: How to use control switch statement in C++

Algorithm:

1. Start
2. Read p,t,r
3. Calculate si,ci
4. Enter choice
   4.1 If ch=1
   4.2 calculate si
   4.3 print si
5. if ch==2
   5.1 5.1 calculate ci
   5.2 print ci
6. end

Sample Input and output:

enter values of p,t,r:

200 2 1.5

1.si

2.ci

Enter your choice:

1

si is: 600

Viva-Voce:

1. Difference between while and do-while
2. What is the Use of Switch
3. What is the Use of Break statement

Conclusion:

Student can implement applications on different control statements and Multi-way selection statements this program attained with co1 and mapped with the PO1, PO2, PO3, PSO1 and PSO2.

EXERCISE-3

Aim:

a) Write a C++ Program for Scope resolution Operator

Description: The Scope Resolution Operator (::) is used to access the global Data

Algorithm:

1. Start
2. Declare and initialize global variable a=10
3. declare and initialize with a=10
4. print a //local
5. {
   5.1 initialize a with 20 // block
   5.2 print a
6. }
7. Print ::a//global

Aim:

b) Write a program to illustrate scope resolution, new and delete Operators. (Dynamic Memory Allocation)

Description: The new operator is used to allocate the memory and delete operator is delete the memory which is allocated by new

Algorithm:

1. Start
2. Declare num, i
3. Print “enter number of students whose percent is to be calculated::
4. Read num
5. Declare *stu_per
6. Initialize stu_per=new float[num];
7. Print “enter student details”
8. for(i=0;i<num;i++) do

    8.1 cout<<"student "<<i+1<<":";

    8.2 cin>>*(stu_per+i);

9. End for
10. Print “student information:";
11. for (i=0;i<num;i++) do

    10.1cout<<"student"<<i+1<<":";

    10.2 cout<<*(stu_per+i);

12. End for
13. delete [stu_per];
14. stop

Sample Input and output:

enter number of students whose percent is to be calculated:

2

enter student details:

student 1:90

student 2:90

student information:

student 1:90

student 2:90

Aim:

c) Write a C++ Program for Storage classes

Description: The Storage classes **are auto, register, static** and **extern**

Storage class	Scope	Default value
Auto	local	garbage
Register	local	garbage
Static	Local & global	0
extern	Program	0

Algorithm:

1. **Start**
2. Declare i as auto int
3. Declare n=5 as register int
4. for(i=0;i<5;i++)
   4.1 call func()
5. end for
6. print a
7. print i
8. print n;
9. void func(void)
   9.1 static int count=0;
   9.2 print the count is:"<< coun
10. end func
11. stop

Sample Input and output:

Value of extern variable is:5

Value of auto variable is: -1234

Value of Register variable is: -1234

Aim:

d) Write a program to illustrate Enumerations

Description: The enumeration operator (enum) is used to give naming constants and the value is incremented by 1 and starting value is 0

Algorithm:

1. **Start**
2. **Declare enum week{ sunday, monday, tuesday, wednesday, thursday, friday, saturday}**
3. enum week today
4. today=tuesday
5. Print today+1
6. Stop

Sample Input and output:

3

Viva-Voce:

1. What are the storage classes in C++
2. What is call by value and call by address
3. Differentiate between call by value and call by address
4. Why new and delete operators are used

Conclusion:

Student can implement call by value, call by reference concepts, Dynamic memory allocations, and Storage class, enumerations concepts .this Program attains with co1 and mapped with the PO1, PO2, PO3, PSO1 and PSO2.

EXERCISE-4

Aim:

Write a program illustrating Inline Functions

Description: the Inline functions are act like Macros. The code replaced in the macro so avoid overheads, so increase the execution speed

Algorithm:

1. Declare inline int power(int x,int n);
2. Declare i
3. for(i=2;i<=3;i++) do
   3.1 Print "2 power "<<i<<" is "<<power(2,i)
4. End for
5. inline int power(int x,int n)
   5.1 return pow(x,n);
6. end power
7. Stop

Sample input and output:

2 power 2 is 4

2 power 3 is 8

Aim:

a) Write a program illustrates function overloading. Write 2 overloading functions for power.

Description: the function overloading is nothing more than one function having same prototype but different parameters

Algorithm:

1. Declare int power(int a,int b);
2. Declare float power(float x, float y);
3. Declare a,b as int
4. Declare x,y as float
5. Print “enter values of a and b:";
6. Read a,b
7. Print “power of two values:"<<power(a,b)
8. Print “enter the values of c and d:";
9. Read x,y
10. Print “power of two float values:"<<power(x,y)
11. int power(int a,int b)
    11.1 return pow(a,b);
12. End
13. float power(float x,float y)
    13.1 return pow(x,y);
14. End
15. Stop

Sample Input and output:

enter values of a and b:

2 3

power of two values:8

enter the values of c and d: 2.1 1.0

power of two float values 2.1

Aim:

b) Write a program illustrate the use of default arguments for simple interest function.

Description: A default argument is a value provided in function declaration that is automatically assigned by the compiler if caller of the function doesn't provide a value for the argument with default value

Algorithm:

1. Declare void display(char = '*', int = 1)
2. Print "No argument passed:\n"
3. Call display()
4. Print ” First argument passed:\n"
5. Call display('#')
6. Print "\both argument passed:\n"
7. Call display('$', 5);
8. void display(char c, int n)
   8.1 for(int i = 1; i <= n; ++i) do
      8.1.1 Print c;
   8.2 end for
9. end display()
10. Stop

.

Sample input and output:

No argument passed:

*

First argument passed:

#

Both arguments passed:

$$$$$

Viva-Voce:

1. What is default argument concept
2. What is function overloading
3. What is polymorphism in C++? How function overloading used in polymorphism

Conclusion:

Student can develop different types of programs to elaborate the use of functions using function overloading, inline functions, default arguments this Program attains with CO2 and mapped with the PO1, PO2, PO3, PO5, PO12, PSO1 and PSO3.

EXERCISE-5

Aim:

a) Write a program to illustrate function overloading. Write 2 overloading functions for adding two numbers

Description: the function overloading is nothing more than one function having same prototype but different parameters

Algorithm:

1. Declare int add(int a,int b);
2. Declare float add(float x,float y);
3. Declare a,b as int
4. Declare x,y as float
5. Print “enter values of a and b:"
6. Read a,b
7. Print ”sum of two values:"<<add(a,b)
8. Print “enter the values of c and d:"
9. Read x,y
10. Print “sum of two float values:"<<add(x,y)<<endl;
11. int add(int a,int b)
    11.1 return a+b;
12. end
13. float add(float a,float b)
    13.1 return a+b;
14. end
15. Stop

Sample Input and Output:

enter values of a and b:

2 3

sum of two values:5

enter the values of c and d: 2.1 1.0

sum of two float values:3.1

Aim:

b) Write a program illustrate function template for power of a number.

Description: Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type. A template is a blueprint or formula for creating a generic class or a function.

Algorithm:

1. Declare template<typename dt>
2. dt power(dt a,dt b)
   2.1 return pow(a,b);
3. End
4. Declare x,y as int
5. Declare a,b as float
6. Print" Enter two integers
7. Read x,y
8. Print "Power of integers is "<<power(x,y)
9. Read "Enter two float numbers"
10. Read a,b
11. Print "Power of float numbers is "<<power(a,b)

Sample Input and Output:

Enter two integers: 3 3

Power of integers is 27

Enter two float numbers: 2.1 1.0

Power of float numbers is 2.1

Aim:

c) Write a program to illustrate function template for swapping of two numbers.

Description: Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type. A template is a blueprint or formula for creating a generic class or a function

Algorithm:

1. Declare template <class T>
2. void swap(T&a,T&b) //Function Template
   2.1 T temp=a
   2.2 a=b
   2.3 b=temp
3. end
4. Declare x1=4,y1=7 as int
5. Declare x2=4.5,y2=7.5 as float
6. Print “Before Swap:”
7. Print “x1=”<<x1<<“ty1=”<<y1
8. Print ”x2=”<<x2<<“ty2=”<<y2
9. Call swap(x1,y1)
10. Call swap(x2,y2)
11. Print “After Swap
12. Print “x1=”<<x1<<“ty1=”<<y1
13. Print “x2=”<<x2<<“ty2=”<<y2
14. Stop

Sample Input and Output:

Before Swap:
x1 :4 y1: 7
x2:4.5 y2:7.5
After Swap:
x1: 7 y1:4
x2:7.5 y2:4.5

Viva-Voce:

1. What is template
2. What is the class template
3. What is the template function
4. What is a Member function
5. What is the member function overloading

Conclusion:

Student can apply the templates, function templates to implement generic programming understand this Program attains with CO2 and mapped with the PO1, PO2, PO3, PO5, PO12, PSO1 and PSO3.

EXERCISE-6

Aim:

a) Write a main function to create objects of DISTANCE class. Input two distances and output the sum.

Description: C++ classes. A class in C++ is a user defined type or data structure declared with keyword class that has data and functions (also called methods) as its members whose access is governed by the three access specifiers private, protected or public (by default access to members of a class is private).

Algorithm:

1. Declare Distance as class
2. Declare feet, inches;
3. public:
4. void input()
   - 4.1 Print “enter feet and distance:";
   - 4.2 read feet, inches
5. End
6. void output()
   - 6.1 Print “distance values are:";
   - 6.2 Print feet, inches
7. End
8. void sum()
   - 8.1 Calculate sum=feet*12+inches;
   - 8.2 Print sum;
9. End

1. Start
2. Declare Distance d;
3. Call d.input();
4. Call d.output();
5. Call d.sum();

6. End

Sample Input and Output:

enter feet and distance:2 3

distance values are:

feet=2

distance=3

Sum=5

Aim:

b) Write a C++ Program to illustrate the use of Constructors and Destructors (use the above program.

Description: A constructor is a kind of member function that initializes an instance of its class. A constructor has the same name as the class and no return value. A constructor can have any number of parameters and a class may have any number of overloaded constructors.

“Destructor" functions are the inverse of constructor functions. They are called when objects are destroyed (deallocated). Designate a function as a class's **destructor** by preceding the class name with a tilde (~).

Algorithm:

1. Declare class Distance
2. Declare feet, inches
3. public:
4. Distance()
   4.1 Print "enter feet and distance:"
   4.2 Read feet, inches
5. void output()
   5.1 Print “distance values are:"
   5.2 Print feet, inches
6. End
7. void sum()
   7.1Calculate sum=feet*12+inches
   7.2 Print sum
8. ~Distance()
   8.1 cout<<"\ndestructor is called:"
9. End

1. Start
2. Declare Distance d;
3. Call d.output();

4. Call d.sum();
5. Stop

Sample Input and Output:

enter feet and distance:2 3

distance values are:

feet=2

distance=3

Sum=27

Aim:

c) Write a program for illustrating function overloading in adding the distance between objects (use the above problem)

Description: the function overloading is nothing more than one function having same prototype but different parameters

Algorithm:

1. Declare Distance class
2. public:
3. void sum(int a,int b)
   3.1 Calculate sum=a*12+b;
   3.2 Print Sum;
4. End
5. void sum(float a,float e)
   5.1 Calculate sum=a*12+e;
   5.2 print sum;
6. End

1. Start
2. Distance s;
3. Declare x,y as int
4. Declare f,k as float
5. Print “enter values of a and b”
6. Read x,y
7. Print ”enter values of f and k:";
8. Read f,k
9. Call s.sum(x,y);
10. Call s.sum(f,k);
11. Stop

Sample Input and Output:

enter values of a and b:2 3

enter values of f and k: 3.0 4.0

27

40.0

Aim:

d) Write a C++ program demonstrating a Bank Account with necessary methods and variables

Description: In bank application the use can create methods like account details, withdraw, balance and display methods in the class

Algorithm:

1. Declare class bank
2. protected:
3. Declare name[20], act_id, balance;
4. public:
5. void input()
   5.1 Print “enter account details:";
   5.2 Read name, act_id, balance;
6. void display()
   6.1 Print “account information:"
7. 6.2 Read name, act_id, balance
8. void withdraw(float n)
   8.1 if(n<balance)
   8.2 balance=balance-n;
   8.3 Print balance
9. End if
10. else
    10.1 balance=balance;
    10.2 Print balance
11. void deposit(float n)
    11.1 bal= balance+n;
    11.2 Print " balance

1. Start

2. Declare bank b;
3. Declare x,y;
4. Call b.input();
5. Print “enter amt to withdraw:";
6. Read x;
7. Call b.withdraw(x);
8. Print “enter amt to deposit";
9. Read y;
10. Call b.deposit(y);
11. Call b.display();
12. Stop

Sample Input and Output:

Enter account details: yuggu 101 1500

Enter amt to withdraw:1000

with draw is successful, remaining balance:500

enter amt to deposit:500

remaining Balance:1500

Account information:

Yuggu

101

1500

Viva-Voce:

1. What is a Class
2. What is the object? How do you create the objects
3. What is the constructor and destructor in C++
4. What is the Constructor overloading

Conclusion:

Student can understand Class and objects concept to implement classes, constructors and constructors overloading this Program attains with CO3 and mapped with the PO1, PO2, PO3, and PSO1.

EXERCISE-7

Aim:

a) Write a program implementing Friend Function

Description: In object-oriented programming, a friend function, that is a "friend" of a given class, is a function that is given the same access as methods to private and protected data. A friend function is declared by the class that is granting access, sofriend functions are part of the class interface, like methods.

Algorithm:

1. Declare class student
2. Declare name[20], reg_no,sem;
3. public:
4. void input()
   - 4.1 Print “enter details:";
   - 4.2 Read name, reg_no ,sem
   - 4.3friend void display(student);
5. End
6. void display(student k)
   - 6.1 Print name, reg_no ,sem

1. Declare student s;
2. Call s.input();
3. Call display(s);
4. Stop

Sample Input and Output:

enter details: yuggu 101 I

student information:

name: yuggu

redno:101

sem:I

Aim:

b) Write a program to illustrate this pointer

Description: Every object in C++ has access to its own address through an important pointer called this pointer. This pointer is an implicit parameter to all member functions. Therefore, inside a member function, this may be used to refer to the invoking object.

Algorithm:

1. Declare class demo
2. Declare num;
3. public:
4. demo() { }
5. demo(int i)
   5.1 num=i;
6. end
7. demo max(demo x)
   7.1 if(num<x.num)
      7.1.1 Return x;
   7.2 Else
      7.2.1 return *this;
8. void show()
      8.1 Print “num”
9. End
   1. Start
   2. Declare int a,b;
   3. Print enter values of a and b";
   4. Read a,b
   5. Declare demo d1(a),d2(b),d3;
   6. d3=d1.max(d2);
   7. Print “maximum number=";
   8. Call d3.show()
   9. Stop

Sample Input and Output:

enter values of a and b 2 3

maximum number=3

Aim:

c) Write a Program to illustrate pointer to a class

Description: A pointer to a C++ class is done exactly the same way as a pointer to a structure and to access members of a pointer to a class you use the member access operator -> operator, just as you do with pointers to structures. Also as with all pointers, you must initialize the pointer before using it.

Algorithm:

1. Declare class bank
2. protected:
3. Declare name[20], act_id,balance;
4. public:
5. void input()
   - 5.1 Print "enter account details:";
   - 5.2 Read name, act_id,balance
6. End
7. void display()
   - 7.1 Print ”account information:";
   - 7.2 Print name, act_id,balance
8. End

1. Start
2. Declare bank b;
3. Declare bank *b1;
4. b1=&b;
5. b1->input();
6. b1->display();

Sample Input and Output:

enter details: yuggu 101 I

student information:

name: yuggu

redno:101

sem:I

Viva-Voce:

1. What is this pointer
2. What is the friend function
3. How create a pointer to the class
4. How we can access member functions outside the class
5. Differentiate between macro and inline

Conclusion:

Student can understand Access specifiers concept useful to implement the Secured OOP applications. Friend functions, Inline functions which are very useful to implement programs to use more efficiently.this Program attains with CO2 & CO3 and mapped with the PO1, PO2, PO3, PO5, PO12, PSO1 and PSO3.

EXERCISE-8

Aim:

a). Write a program to Overload Unary, and Binary Operators as Member Function, and Non Member Function.
i. Unary operator as member function
ii. Binary operator as nonmember function

Description:

Operator overloading redefines the concept of usage of operator.
The syntax is

```
Return type Operator  operator symbol (parameters)
 {
Statements;
 }
```

The mechanism in which a non-member function has access permission to the private members of the class. This can be done by declaring a non-member function friend to the class whose private data is to be accessed.

Algorithm:

Step 1: Start the program.
Step 2: Declare the class unary_op .
Step 3: Declare the variables and its member function.
Step 4: Using the constructor unary_op() to get the two numbers.
Step 5: Define the function operator ++ to increment the values
Step 6: Define the function operator - -to decrement the values.
Step 7: Define the show function.
Step 8: Declare the class object.
Step 9: Call the function show to display before increment values.
Step 10: Call the function operator ++() by incrementing the class object.
Step 11: Call the function show to display after increment values
Step 12: Call the function operator - -() by decrementing the class object.
Step 13: Call the function show to display after decrement values
Step 14: Stop the program.

Sample input & output:

```
before increment
2       3
after increment
3       4
after decrement
```

2 3

Algorithm for Binary operator as nonmember function:

Step 1: Start the program.
Step 2: Declare the class number .
Step 3: Declare the variables and its member function.
Step 4: Using the member function input() to get the two numbers.
Step 5: Define the display() function.
Step 6: Define the function operator *() to perform multiplication and make friend to class number using keyword friend.
friend number operator *(number a,number b);
Step 7: Declare the class objects n1,n2,n3.
Step 8: Call the function input() using class object n1 to get the two numbers.
Step 9: Call the function input() using class object n2 to get the two numbers.
Step 10: Call the function operator *() to perform multiplication.
Step 11: Call the function display() to display the values.
Step 12: Stop the program

Sample input & output:

enter x and y : 2 3
enter x and y : 3 4

x=6 y=12

Aim:

b). Write a c ++ program to implement the overloading assignment = operator

Description:

Operator overloading redefines the concept of usage of operator.
The syntax is

```
Return type Operator  operator symbol (parameters)
 {
Statements;
 }
```

The mechanism in which a non-member function has access permission to the private members of the class. This can be done by declaring a non-member function friend to the class whose private data is to be accessed.

Algorithm:

Step 1: Start the program.
Step 2: Declare the class demo .
Step 3: Declare the variables and its member function.
Step 4: Using the constructor to get the two numbers.
Step 5: Define the show() function.
Step 6: Define the function operator = () to perform assignment operator overloading.
Step 7: Declare the class object d1 and d2.
Step 8: Call the function show to display the value of d1 object.
Step 9: Call the function show to display the value of d2 object.
Step 10: Call the function operator =() to assign d2 object value into d1.
Step 11: Call the function show to display the value of d1 object.
Step 12: Call the function show to display the value of d2 object.
Step 13: Stop the program

Sample input & output:

values of d1
X=2
values of d2
X=3
values of d1
X=3
values of d2
X=3

Aim:

c).Write a case study on Overloading Operators and Overloading Functions (150 Words)

C++ allows you to specify more than one definition for a function name or an operator in the same scope, which is called function overloading and operator overloading respectively.

When you call an overloaded function or operator, the compiler determines the most appropriate definition to use by comparing the argument types you used to call the function or operator with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function or operator is called overload resolution.

Function overloading in C++:

```
using namespace std;
#include<iostream>
#include<math.h>
int power(int a,int b);
float power(float x,float y);
main()
{
        int a,b;
        float x,y;
        cout<<"enter values of a and b:";
        cin>>a>>b;
        cout<<"power of two values:"<<power(a,b)<<endl;
        cout<<"enter the values of c and d:";
        cin>>x>>y;
        cout<<"power of two float values:"<<power(x,y)<<endl;
}
int power(int a,int b)
{
        return pow(a,b);
}
float power(float x,float y)
{
        return pow(x,y);
}
```

Operators overloading in C++:

```
using namespace std;
#include<iostream>
class unary_op
{
        int x,y;
```

```
        public:
        unary_op(int i,int j)
        {
                x=i;
                y=j;
        }
        void show()
        {
        cout<<"\nx="<<x;
        cout<<"\ny="<<y;
        }
        void operator ++()
        {
                ++x;
                ++y;
        }
        void operator --()
        {
        --x;
        --y;
        }
};
main()
{
        unary_op u(2,3);
        cout<<"\nbefore increment";
        u.show();
        ++u;
        cout<<"\nafter increment";
        u.show();
        --u;
        cout<<"\n after decrement";
        u.show();
}
```

Viva-Voce:

1. Can we overload default operators? How we can overload it?
2. How can we overload pre and post incerement operators?
3. If operator overloading function returing something then what is the return type in operator overloading?

Conclusion:

Student can implement unary and binary operators on object while performing operator overloading. So this program attains with CO3 and mapped with the PO1, PO2, PO3, PSO1.

EXERCISE-9

Aim:

a) Write C++ Programs and incorporating various forms of Inheritance
i) Single Inheritance
ii) Hierarchical Inheritance
iii) Multiple Inheritances
iv) Multi-level inheritance
v) Hybrid inheritance

Description:

Deriving the features of Super class into Sub class is defined as inheritance. Multiple, Multilevel ,Single, Hybrid are the some of the inheritances. Multiple, Multilevel and Hybrid

Virtual Base class avoids the ambiguity occurring due to multipath inheritance. When a class is delared as virtual , the compikler takes essential caution to avoid the duplication of member variable.

Algorithm:

Step 1: Start the program.
Step 2: Declare the base class student_det.
Step 3: Declare the variables name, email_id, ph no, gender.
Step 4: Declare the derived class edu_det.
Step 5: Declare the variables course and percentage.
Step 6: Declare and define the function read() to get the values.
Step 7: Declare and define the function display() to display the values.
Step 8: Create the derived class object.
Step 9: Call the function read() and display().
Step 10: Stop the program

Sample input & output:

enter details:
raju
raju@gmail.com
2788562
B.Tech
71.2
student information:
name= raju
email_id= raju@gmail.com
phone_no= 2788562
course= B.Tech
percentage= 71.2

Algorithm for Hierarchical Inheritance:

Step 1: Start the program.
Step 2: Declare the base class player.
Step 3: Declare the variables name, age, country, matches.
Step 4: Declare and define the function read1() to get the player details.
Step 5: Declare and define the function show1() to display the player details
Step 6: Declare the derived class cricket.
Step 7: Declare the variables runs.
Step 8: Declare and define the function read() to get the values.
Step 9: Declare and define the function show() to display the values.
Step 10: Declare the derived class footballt.
Step 11: Declare the variables goals.
Step12: Declare and define the function read() to get the values.
Step 13: Declare and define the function show() to display the values.
Step 14: Create the derived class cricket object.
Step 15: Call the function read() and show() to get the cricket information.
Step 16: Create the derived class football object.
Step 17: Call the function read() and show() to get the football information
Step 18: Stop the program

Sample input & output:

```
enter details
ABC
23
India
50
name= ABC
 age=23
 country=India
matches=50
enter runs: 350
average score= 7
enter details
XYZ
20
Brazil
20
name= XYZ
 age=20
 country=Brazil
matches=20
enter goals: 40
average score: 2
```

Algorithm for Multiple Inheritances:

Step 1: Start the program.
Step 2: Declare the base class student_details .
Step 3: Declare the variables name,roolno.
Step 4: Declare the derived class physical_details.
Step 5: Declare the variables height,weight.
Step 6: Declare the derived class sports.
Step 7: Declare the variables sportname.
Step 8: Declare and define the function read() to get the student details.
Step 9: Declare and define the function output() to display the student information.
Step 10: Create the derived class object.
Step 11: Call the function read() and output() to get the information.
Step 12: Stop the program

Sample input & output:

Enter details:
Raju
501
5.5
70
Cricket
sports information
name= raju
rollno= 501
height= 5.5
weight= 70
sportname= cricket

Algorithm for Multi-level inheritance:

Step 1: Start the program.
Step 2: Declare the base class student_details .
Step 3: Declare the variables name,emailed,phone number,gender.
Step 4: Declare the derived class physical_details.
Step 5: Declare the variables height,weight.
Step 6: Declare the derived class edu_det.
Step 7: Declare the variables course,percentage.
Step 8: Declare and define the function read() to get the student details.
Step 9: Declare and define the function display() to display the student information.
Step 10: Create the derived class object.
Step 11: Call the function read() and display() to get the information.
Step 12: Stop the program

Sample input & output:

enter details:
raju
raju@gmail.com
2788562
B.Tech
71.2
5.5
70
student information:
name= raju
email_id= raju@gmail.com
phone_no= 2788562
course= B.Tech
percentage= 71.2
height= 5.5
weight= 70

Algorithm for Hybrid inheritance:

Step 1: Start the program.
Step 2: Declare the base class student_details .
Step 3: Declare the variables name,regno.
Step 4: Declare the derived class physical_details.
Step 5: Declare the variables height,weight.
Step 6: Declare the base class edu_det.
Step 7: Declare the variables course,percentage.
Step 8: Declare the derived class sports
Step 9: Declare the variables sportname.
Step 10: Declare and define the function read() to get the student details.
Step 11: Declare and define the function show() to display the student information.
Step 12: Create the derived class object.
Step 13: Call the function read() and show() to get the information.
Step 14: Stop the program

Sample input & output:

enter details:
raju
B.Tech
71.2
5.5
70
Cricket

student information:
name= raju
course= B.Tech
percentage= 71.2
height= 5.5
weight= 70
sportname =cricket

Aim:

b) Write a program to show Virtual Base Class

Description:

Deriving the features of Super class into Sub class is defined as inheritance. Multiple, Multilevel ,Single, Hybrid are the some of the inheritances. Multiple, Multilevel and Hybrid

Virtual Base class avoids the ambiguity occurring due to multipath inheritance. When a class is delared as virtual , the compikler takes essential caution to avoid the duplication of member variable.

Algorithm:

Step 1: Start the program.
Step 2: Declare the base class A .
Step 3: Declare the variables.
Step 4: Declare the derived class B .
Step 5: Declare the variables.
Step 6: Declare the derived class C .
Step 7: Declare the variables
Step 8: Declare the derived class D .
Step 9: Declare the variables
Step 10: Declare and define the function input() to read the values.
Step 11: Declare and define the function display() to display the values.
Step 12: Create the derived class object.
Step 13: Call the function input() and display() to get the information.
Step 14: Stop the program

Sample input & output:

Enter values: 2 3 4 5

X=2 y=3 z=4 w=5

Aim:

c) Write a case study on using virtual classes (150 Words)

When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class. This can be achieved by preceding the base class' name with the word virtual.- Consider the following example :

```
class A
{
  public:
     int i;
};

class B : virtual public A
{
  public:
     int j;
};

class C: virtual public A
{
  public:
     int k;
};

class D: public B, public C
{
  public:
     int sum;
};
int main()
{
  D ob;
  ob.i = 10; //unambiguous since only one copy of i is inherited.
  ob.j = 20;
  ob.k = 30;
  ob.sum = ob.i + ob.j + ob.k;
  cout << "Value of i is : "<< ob.i<<"\n";
  cout << "Value of j is : "<< ob.j<<"\n"; cout << "Value of k is :"<< ob.k<<"\n";
```

```
  cout << “Sum is : ”<< ob.sum <<”\n”;
  return 0;
}
```

Viva-Voce:

1. Illustrate the execution process of construcors in multiple inheritance?
2. In which way we can rectify the problem occur in multipath inheritance?
3. Demonstrate multilevel inheritance using real time examples?

Conclusion:

Student can get knowledge and understand and analyze to perform different types of inheritance like multiple, multilevel and hybrid. So this program attains with CO4 and mapped with the PO1, PO2, PO3, PO4 and PSO1, PSO2.

EXERCISE-10

Aim:

a) Write a Program in C++ to illustrate the order of execution of constructors and destructors in inheritance

Description:

Base class constructors are always called in the derived class constructors. Whenever you create derived class object, first the base class default constructor is executed and then the derived class's constructor finishes execution.

Whether derived class's default constructor is called or parameterised is called, base class's default constructor is always called inside them.

To call base class's parameterised constructor inside derived class's parameterised constructo, we must mention it explicitly while declaring derived class's parameterized constructor.

Algorithm:

Step 1: Start the program.
Step 2: Declare the base class student_details.
Step 3: Declare the variables name,rollno.
Step 4: Declare constructor to read the student details.
Step 5: Declare destructor to display the student details.
Step 6: Declare the derived class ug_details.
Step 7: Declare the variables course,percentage.
Step 8: Declare constructor to read the student ug details.
Step 9: Declare destructor to display the student ug details.
Step 10: Declare the derived class pg_details.
Step 11: Declare the variables specilization,percentage.
Step 12: Declare constructor to read the student pg details.
Step 13: Declare destructor to display the student pg details.
Step 14: Create derived class object.
Step 15: Stop the program.

Sample input & output:

```
enter name and rollno          raju     501
enter course and percentage    B.Tech          71.2
enter specilization and percentage     networks        80.2
specilization= networks
percentage=80.2
course= B.Tech
percentage=71.2
name= raju
rollno=501
```

Aim:

b) Write a Program to show how constructors are invoked in derived class

Description:

Base class constructors are always called in the derived class constructors. Whenever you create derived class object, first the base class default constructor is executed and then the derived class's constructor finishes execution.

Whether derived class's default constructor is called or parameterised is called, base class's default constructor is always called inside them.

To call base class's parameterised constructor inside derived class's parameterised constructo, we must mention it explicitly while declaring derived class's parameterized constructor.

Algorithm:

Step 1: Start the program.
Step 2: Declare the base class A.
Step 3: Declare the variables.
Step 4: Declare constructor to read and display the values.
Step 5: Declare the base class B.
Step 6: Declare the variables.
Step 7: Declare constructor to read and display the values.
Step 8: Declare the derived class C.
Step 9: Declare the variables.
Step 10: Declare constructor to read and display the values and invoke the constructors of base class A and B.

```
public: C():A(),B()
        {
                ---------
                --------
        }
```

Step 11: Create object for derived class.
Step 12: Stop the program.

Sample input & output:

X=10
Y=20
Z=30

Viva-Voce:

1.Can we overload constructors as normal member functions?
2. It is possible to write a constructor without arguments?
3. How the constructors are executed implicitly?

Conclusion:

Students can implement constructors and destructors in inheritance and analyze the order of execution of constructors and destructors invoked in derived class. So this program attains with CO3 & CO4 and mapped with the PO1, PO2, PO3, PO4 and PSO1, PSO2.

.

EXERCISE-11

Aim:

a) Write a program to illustrate runtime polymorphism

Description:

The pointer this is transferred as an unseen parameter in all calls to non-static member functions. The keyword this is a local variable that is always present in the body of any non-static member function. The pointer is rarely referred to explicitly in a function definition

Dynamic binding or late binding can be achieved through virtual keyword. The member function followed by the vrtual keyword is called a virtual function. These are used with inheritance.

Algorithm:

Step 1: Start the program.
Step 2: Declare the base class A .
Step 3: Declare the variables.
Step 4: Declare and define the virtual function input() to read the values.
Step 5: Declare and define the virtual function display() to display the values.
Step 6: Declare the derived class B .
Step 7: Declare the variables
Step 8: Declare and define the function input() to read the values.
Step 9: Declare and define the function display() to display the values.
Step 10: Create object for base class and also pointer object.
Step 11: Assign the address of base class object into pointer object.
Step 12: Call the function input() and display() to get the information.
Step 13: Create object for derived class.
Step 14: Assign the address of derived class object into pointer object.
Step 15: Call the function input() and display() to get the information.
Step 16: Stop the program.

Sample input & output:

```
Enter x      2
Enter y      3
X=2
Y=3
```

Aim:

b) Write a program to illustrate this pointer

Description:

The pointer this is transferred as an unseen parameter in all calls to non-static member functions. The keyword this is a local variable that is always present in the body of any non-static member function. The pointer is rarely referred to explicitly in a function definition

Dynamic binding or late binding can be achieved through virtual keyword. The member function followed by the vrtual keyword is called a virtual function. These are used with inheritance.

Algorithm:

Step 1: Start the program.

Step 2: Declare the class number

Step 3: Declare the variable n.

Step 4: Declare and define constructor to initialize the values.

Step 5: Declare and define the function input() to read the values.

Step 6: Declare and define the function display() to display the values.

Step 7: Declare and define the function

number max(number &a)

Step 8: if(n>a.n)

8.1: return a

8.2: else return *this

Step 9: Create object n1,n2,n3

9.1: n1.input()

9.2: n2.input()

9.3: n3=n1.max(n2)

9.4: n3.show()

Step 10: Stop the program.

Sample input & output:

Enter value: 285

Enter value: 297
Maximum value= 297

Aim:

c) Write a program illustrates pure virtual function and calculate the area of different shapes by using abstract class.

Description:

The pointer this is transferred as an unseen parameter in all calls to non-static member functions. The keyword this is a local variable that is always present in the body of any non-static member function. The pointer is rarely referred to explicitly in a function definition

Dynamic binding or late binding can be achieved through virtual keyword. The member function followed by the vrtual keyword is called a virtual function. These are used with inheritance.

Algorithm:

Step 1: Start the program.
Step 2: Declare the base class shape .
Step 3: Declare pure virtual functions
virtual void input()=0;
virtual void area()=0;
Step 4: Declare the derived class rectangle
Step 5: Declare the variables length,breadth.
Step 6: Declare and define the function input() to read the values.
Step 7: Declare and define the function area() to display the values.
Step 8: Declare the derived class triangle
Step 9: Declare the variables base,height.
Step 10: Declare and define the function input() to read the values.
Step 11: Declare and define the function area() to display the values.
Step 12: Declare the derived class circle
Step 13: Declare the variables radius.
Step 14: Declare and define the function input() to read the values.
Step 15: Declare and define the function area() to display the values
Step 16: Create pointer object for base class.
Step 17: Create object for derived class.
Step 18: Assign the address of derived class object into pointer object.
Step 19: Call the function input() and area() to get the information.
Step 20: Create object for derived class.
Step 21: Assign the address of derived class object into pointer object.
Step 22: Call the function input() and area() to get the information.
Step 23: Create object for derived class.

Step 24: Assign the address of derived class object into pointer object.
Step 25: Call the function input() and area() to get the information.\
Step 26: Stop the program.

Sample input & output:

Enter length and breath 50 30
Area= 150
Enter base and height 30 50
Area=75
Enter radius 3
Area=28.6

Aim:

d) Write a case study on virtual functions (150 Words)

A virtual function is a member function that you expect to be redefined in derived classes. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class's version of the function.

Virtual functions ensure that the correct function is called for an object, regardless of the expression used to make the function call.

Suppose a base class contains a function declared as virtual and a derived class defines the same function. The function from the derived class is invoked for objects of the derived class, even if it is called using a pointer or reference to the base class. The following example shows a base class that provides an implementation of the `Print Balance` function and two derived classes.

```
#include<iostream.h>
#include<conio.h>
class first
{
  int b;
  public:
  first() { b=10;}
  virtual void display()
  {
            cout<<"\n b="<<b;
  }
};
class second : virtual public first
{
  int d;
  public:
  second() { d=20;}
  virtual void display ()
  {
            cout<<"\n d="<<d;
  }
};


main()
{
  clrscr();
  first f,*p;
  second s;
  p=&f;
```

```
    p->display();
    p=&s;
    p->dispaly();
    getch();
    return 0;
}
```

Viva-Voce:

1. When base class and derived class member functions with same name then what is the order of execution of those functions?
2. How can we excute base class member function when derived class have same name?
3. How can we protect base calss member variable?

Conclusion:

Student can able to implement this pointer, runtime polymorphism, virtual and pure virtual functions. . So this program attains with CO4 and mapped with the PO1, PO2, PO3, PO4 and PSO1, PSO2.

EXERCISE-12

Aim:

a) Write a C++ Program to illustrate template class

Description:

A template function is overloaded with generic types . Function overloading is the process of defining a function with same name but with multiple parameters.

Class template (<class T>) tells the compiler that the following class declaration can use the template data type. T is avariable of template type that can be used in the class to define a variable of template type.

Function template holds arguments of various types and the compiler identifies the data type according to the requirement. Genericity is attained using templates

Algorithm:

Step 1: Start the program.
Step 2: Create class template using
template <class t>
Step 3: Declare the base class demo .
Step 4: Declare the variables.
Step 5: Declare and define constructor initialize and display the values.
Step 6: Create object
demo<int>d1(5);
demo<char>d2('A');
demo<float>d3(1.0);
Step 7: Stop the program.

Sample input & output:
value of i:5
value of i:A
value of i:1.0

Aim:

b) Write a Program to illustrate class templates with multiple parameters

Description:

A template function is overloaded with generic types . Function overloading is the process of defining a function with same name but with multiple parameters.

Class template (<class T>) tells the compiler that the following class declaration can use the template data type. T is avariable of template type that can be used in the class to define a variable of template type.

Function template holds arguments of various types and the compiler identifies the data type according to the requirement. Genericity is attained using templates

Algorithm:

Step 1: Start the program.
Step 2: Create class template using
template < class t1,class t2>
Step 3: Declare the base class demo .
Step 4: Declare the variables.
Step 5: Declare and define constructor initialize and display the values.
Step 6: Create object
demo<int,float>d1(2,2.3);
demo<char,int>d2('A',3);
Step 7: Stop the program.

Sample input & output:

a=2
b=2.3
a=A
a=3

Aim:

c) Write a Program to illustrate member function templates

Description:

A template function is overloaded with generic types . Function overloading is the process of defining a function with same name but with multiple parameters.

Class template (<class T>) tells the compiler that the following class declaration can use the template data type. T is avariable of template type that can be used in the class to define a variable of template type.

Function template holds arguments of various types and the compiler identifies the data type according to the requirement. Genericity is attained using templates

Algorithm:

Step 1: Start the program.
Step 2: Create class template using
template < class t1,class t2>
Step 3: Declare the base class demo .
Step 4: Declare the variables.
Step 5: Declare and define constructor initialize and display the values.
Step 6: Declare and define function initialize and display the values.
Step 7: Create object
demo<int>d1(3);
d1.show(5);
demo<char>d2('A');
d2.show('f');
demo<float>d3(1.0);
d3.show(2.3);

Step 8: Stop the program.

Sample input & output:

a=3
b=5
a=A
b=f
a=1.0
b=2.3

Viva-Voce:

1. Illustrate generic programming?
2. Differentiate function template with class template?
3. When one function name is same as template function name then which function will be executed?

Conclusion:

Student can apply the templates; function templates concept to implement generic programming get knowledge and analyze generic programming enhances the efficiency of object oriented approach. . So this program attains with CO5 and mapped with the PO1, PO2, PO3, PO5 and PSO1, PSO2.

EXERCISE-13

Aim:

a).Write a Program for Exception Handling Divide by zero

Description:

An exception is an object. The exception-handling technique passes the control of a program from a location of exception in a program to an exception-handler routine linked with the try block.

Algorithm:

Step 1: Start the program.
Step 2: Declare the variables x,y.
Step 3: Read the values x,y.
Step 4: Inside the try block check the condition.
    a. if(y==0) then throw the exception.
    b. otherwise calculate the quotient x/y and display the value.
Step 5: Catch the exception and display the appropriate message “err:divided by zero exception”.
Step 6: Stop the program.

Sample input & output:

enter the value of x and y 4 0
err:divided by zero exception

Aim:

b). Write a Program to rethrow an Exception

Description:

An exception is an object. The exception-handling technique passes the control of a program from a location of exception in a program to an exception-handler routine linked with the try block.

Algorithm:

Step 1: Start the program.
Step 2: Declare and define the function void sub(int a,int b)
Step 3: Inside the try block
Step 4: if(a==0) then throw 'A' else perform subtraction and display the value.
Step 5: Catch the exception and display the appropriate message "caught an exception in sub()".
Step 6: Rethrow the exception.
Step 7: Display the message "end of sub()".
Step 8: Inside the try block
sub(5,2);
sub(0,9);
Step 9: Catch the rethrow exception and display the appropriate message "caught an exception"
Step 10: Display the message "end of main".
Step 11: Stop the program.

Sample input & output:

subtraction is 3
end of sub()
caught an exception in sub()
caught an exception
end of main

Viva-Voce:

1. How exception is different from error?
2. Illustrate the exception handling mechanism?
3. It is possible to re-throw an exception, if yes how it is possible?

Conclusion:

Student can get knowledge and understand the exceptions occur in a program and how to handle those exceptions. So this program attains with CO5 and mapped with the PO1, PO2, PO3, PO5 and PSO1, PSO2.

EXERCISE-14

Aim:

a) Write a Program to implement List and List Operations

Description:

Lists are sequence containers that allow constant time insert and erase operations anywhere within the sequence, and iteration in both directions. List containers are implemented as doubly-linked lists; doubly linked lists can store each of the elements they contain in different and unrelated storage locations.

Algorithm:

Step 1: Start the program.
Step 2: Declare and define the function void show (list<int>&num)
Step 3: Display "list elements:".
Step 4: Create object for iterator class
    list<int>::iterator it=num.begin();
Step 5: if(it!=num.end()) then goto step6 otherwise goto step 11.
Step 6: Display *it and increment the it goto step 5.
Step 7: Create object for list class
    list<int>li;
Step 8: Insert the values into list using push_back()
    li.push_back(5);
    li.push_back(15);
    li.push_back(25);
Step 9: Call the function show().
Step 10: Insert the values into list using push_front ()
    li.push_front(35);
    li.push_front(45);
Step 11: Stop the program.

Sample input & output:

list elements
5
15
25
list elements
5
15
25
35
45

Aim:

b) Write a Program to implementVector andVector Operations

Description:

The storage of the vector is handled automatically, being expanded and contracted as needed. Vectors usually occupy more space than static arrays, because more memory is allocated to handle future growth. This way a vector does not need to reallocate each time an element is inserted, but only when the additional memory is exhausted.

Algorithm:

Step 1: Start the program.
Step 2: Create object for vector class
vector<int>ve;
Step 3: Display “vector size:”.
ve.size();
Step 4: Decalre the variables i,n.
Step 5: if(i<5) goto step 6 otherwise goto step 7.
Step 6: Read the value of n and insert into vector
ve.push_back(n);
goto step 5.
Step 7: Display “vector size:”.
ve.size();
Step 8: Create object for iterator class
vector<int>::iterator it=num.begin();
Step 9: if(it!=num.end()) then goto step10 otherwise goto step 11.
Step 10: Display *it and increment the it goto step 9.
Step 11: if(i<5) goto step 12 otherwise goto step 13.
Step 12: Display the vector values.
ve[i];
goto step 11.
Step 13: Stop the program.

Sample input & output:

```
Vector size : 0
Enter value:    10      20      30      40      50
Vector size : 5
displaying values using iterators
10      20      30      40      50
displaying values using vector
10      20      30      40      50
```

Viva-Voce:

1. Discuss about different types of containers?
2. What are the list operations?

Conclusion:

Student can get knowledge and implement the different operations of list and vector So this program attains with CO6 and mapped with the PO1,PO2, PO3,PO4, PO5 and PSO1, PSO2.

EXERCISE-15

Aim:

a) Write a Program to implement Deque and Deque Operations

Description:

Specific libraries may implement deques in different ways, generally as some form of dynamic array. But in any case, they allow for the individual elements to be accessed directly through random access iterators, with storage handled automatically by expanding and contracting the container as needed.

Algorithm:

Step 1: Start the program.
Step 2: Decalre the variables name.
Step 3: Create object for deque class
    deque<string>dq;
Step 4: if(true) goto step 5 otherwise goto step 6.
Step 5: Read the string if it is not equal to 'a' insert into deque
    dq.push_front(name);
    goto step 4.
Step 6: Create object for iterator class
    deque<string>::iterator it
Step 7: if(it!=dq.end()) then goto step8 otherwise goto step 9.
Step 8: Display *it and increment the it goto step 7.
Step 9: Delete the beginning and ending value in deque
    dq.pop_front();
    dq.pop_back();
Step 10: if(it!=dq.end()) then goto step11 otherwise goto step 12.
Step 11: Display *it and increment the it goto step 10.
Step 12: Stop the program.

Sample input & output:

enter name and to stop give a
joj
jack
jal
a
elements are:
joj
jack
jal
after deleting are
jack

Aim:

b) Write a Program to implement Map and Map Operations

Description:

Maps are associative containers that store elements formed by a combination of a key value and a mapped value, following a specific order. In a map, the key values are generally used to sort and uniquely identify the elements, while the mapped values store the content associated to this key. The types of key and mapped value may differ, and are grouped together in member type value_type, which is a pair type combining both.

Algorithm:

Step 1: Start the program.
Step 2: Decalre the variables name,code,i.
Step 3: Create object for map class
map<string,int>m;
Step 4: if(i<3) goto step 5 otherwise goto step 6.
Step 5: Read name and code , insert into map
m[name]=code;
goto step 4.
Step 6: Create object for iterator class
map<string,int>::iterator it;
Step 7: if(it!=m.end()) then goto step8 otherwise goto step 9.
Step 8: Display “(*it).first” to print first value and "(*it).second” to print second value goto step 7.
Step 9: Stop the program.

Sample input & output:

Enter name and code
TV 223
Fridge 401
Cooker 302
Name code
TV 223
Fridge 401
Cooker 302

Viva-Voce:

1. Map is related to which type of container?
2. Analyze mapping of value with key?

Conclusion:

Student can get knowledge and implement the different operations of deque and map So this program attains with CO6 and mapped with the PO1,PO2, PO3,PO4, PO5 and PSO1, PSO2.

Experiments beyond the Syllabus

Aim:

Write a C++ program illustrating selection sort

Description:

The algorithm divides the input list into two parts: the sublist of items already sorted, which is built up from left to right at the front (left) of the list, and the sublist of items remaining to be sorted that occupy the rest of the list. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list.

Algorithm:

Step 1: Start the program.
Step 2: Declare class array.
Step 3: Declare the variables arr,count.
Step 4: Declare and define constructor array()

```
count = 0 ;
for ( int i = 0 ; i < MAX ; i++ )
arr[i] = 0 ;
```

Step 5: Declare and define function add()

```
if ( count < MAX )
{
        arr[count] = item ;
        count++ ;
}
else
        cout << "\nArray is full" ;
}
```

Step 6: Declare and define function sort( )

```
int temp ;
for ( int i = 0 ; i <= count - 2 ; i++ )
{
        for ( int j = i + 1 ; j <= count - 1 ; j++ )
        {
                if ( arr[i] > arr[j] )
                {
                        temp = arr[i] ;
                        arr[i] = arr[j] ;
                        arr[j] = temp ;
                }
        }
}
```

Step 7: Declare and define function display( )

```
for ( int i = 0 ; i < count ; i++ )
        cout << arr[i] << "  " ;
cout << endl ;
```

Step 8: Create object for the class.

Step 9: Call the function add()

```
a.add ( 25 ) ;
a.add ( 17 ) ;
a.add ( 31 ) ;
a.add ( 13 ) ;
a.add ( 2 ) ;
```

Step 10: Display Array before sorting. Call the function Display().

Step 11: Call the function sort().

Step 12: Display Array after sorting. Call the function Display().

Step 13: Stop the program.

Sample input and output:

Selection sort

Array before sorting

25 17 31 13 2

Array after selection sorting

2 13 17 25 31

Viva-Voce:

1. How generic function concept useful for sorting?
2. Illustrate different types of sorting techniques and enhance which one is efficient and faster?
3. Difference between space and time complexity?

Conclusion:

Student can get knowledge and implement different sorting operations like bubble, quick and merge sorting techniques. So this program attains with CO1 and mapped with the PO1,PO2, PO3 and PSO1,PSO2.

Aim:

Write a C++ program illustrating Copy Constructor.

Description:

When an object is passed by value into a function, a temporary copy of that object is created . All copy constructors require one argument, with reference to an object of that class.

Algorithm:

Step 1:Start

Step 2:Read int x,float y

Step 3:data(int xx,float yy)

{x=xx;y=yy}

data operator=(data &d)

Step 4:x=d.x

y=d.y

Step 5:copy constuctor executed

Step 6:data d1(12,5.8)

data d2,d4

Step 7:d4=d2=d1

data d3=d1

Step 8:print the objects

Step 9:stop

Sample input and output:

Assignment operator executed
Assignment operator executed
copy constructor executed
object d1: x=12 y=5.8
object d2: x=12 y=5.8
object d3: x=12 y=5.8
object d4: x=12 y=5.8

Viva-Voce:

1. In how many ways can we call copy constructor?
2. In which way we can de-allocate the memory of an object?
3. When the destructors will be executed?

Conclusion:

Student can get knowledge and understand and analyze different applications and types of constructors one of the important one is copy constructors. So this program attains with CO3 and mapped with the PO1,PO2, PO3 and PSO1.