	SVKM's NMIMS Mukesh Patel School of Technology Management & Engineering / School of Technology Management & Engineering	
B. Tech/MBA Tech	**Workbook**	**Academic Year- 2024-25**
Year:-First	**Subject:-** Programming for Problem Solving	**Semester: - First**

Experiment: 10

PART A

(PART A: TO BE REFERRED BY STUDENTS)

Aim: Programming using object-oriented programming (using data members, member functions, constructors, overloading & inheritance)

Learning Outcomes: The learner would be able to

1. Solve problems using object-oriented programming
2. Implement programming using overloading
3. Write programs using inheritance

Theory:

Note: - Theory part is continued from page number two.

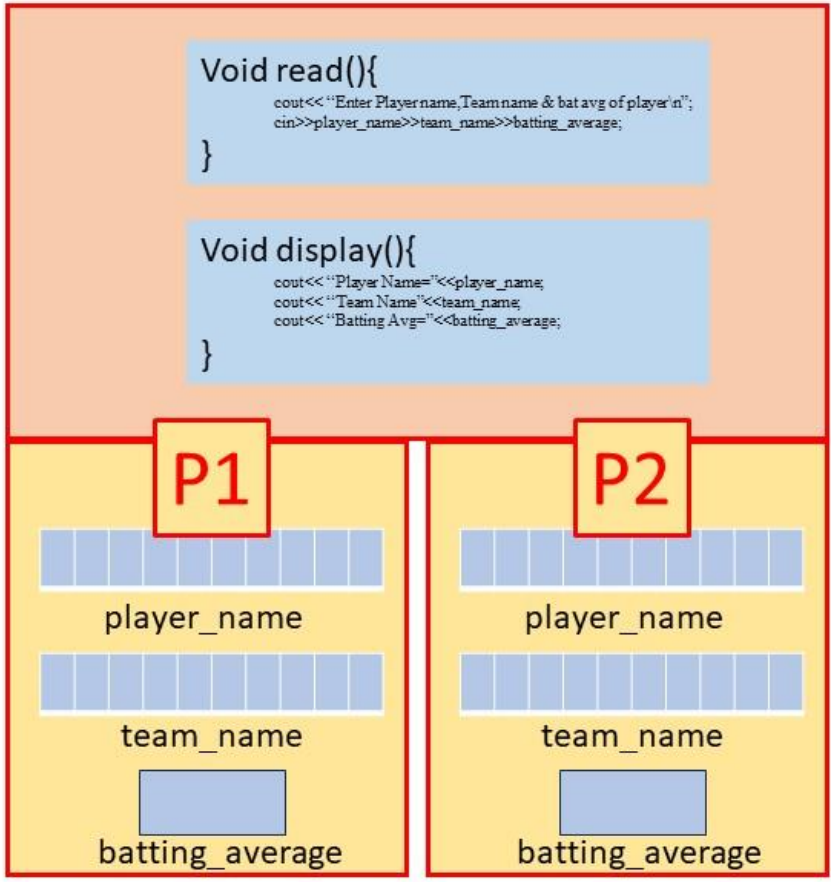
Tasks:

1. Class "Employee" has data members: Emp_id, Emp_name and Emp_sal and this class uses a parameterized constructor to accept the details of 2 employees and display the results using the display () function.
2. Define class Complex with real and imaginary as data members, define default and parameterized constructor to initialize two complex numbers. Define add (Complex, Complex) member function to add two complex numbers and show( ) function to display both the complex numbers with their addition.
3. Rewrite above question to add two complex numbers using overloaded + operator.
4. Create a class rectangle with (length, width), derive a class box with additional member (depth). In both the classes write parameterized constructor to initialize data member and area () function to find area. Define main () and create appropriate objects to call area () function.
5. Declare a class employee having data members as id, name and member function as get Data() & display( ). Derive class manager from employee class. Manager class has data members as salary and member functions as getdata () & display (). Again, derive class project manager from manager. Project manager class have data members as total experience, number of projects handled and member function as getdata () & display (). Write a program using multilevel inheritance to display all details of project manager.

Define a class Cricket with data members player_name, team_name & batting_average and Member Function as read( ) and display( ). Write C++ program to read and display information of two players of a team.

```
class Cricket{
        private:
                char player_name10];
                char team_name[10];
                float batting_average;
        public:
                void read( ){
                        cout<< "Enter Player name,Team name & bat avg of player\n";
                        cin>>player_name>>team_name>>batting_average;
                }
                void display( ){
                        cout<< "Player Name="<<player_name;
                        cout<< "Team Name"<<team_name;
                        cout<< "Batting Avg="<<batting_average;
                }
};
int main( ){
        Cricket p1,p2;
        p1.read( );
        p1.display( );
        p2.read( );
        p2.display( );
        return 0;
}
```

Memory Allocation For Objects



Constructor

Constructor:-

- A constructor is a special member function whose task is to initialize the objects. (having no return type (no void as well)
- **It is special because its name is same as the class name.**
- The constructor is invoked whenever an object of its associated class is created.
- It is called constructor because it constructs the values of data members of the class.
- Like member function constructor can be defined within or outside of class.

Types of Constructors:-

- Default Constructor
- Parameterized Constructor
- Copy Constructor

Question:- Define a class Point with x and y as data member. Declare three objects of the Point and initialize them using default, parameterized and copy constructor. Display all the points by defining show( ) function.

```
#include <iostream>
using namespace std;
```

```
class Point{
    private:
        int x,y;
    public:
        Point(){
            x=y=0;
        }
        Point(int m,int n){
            x=m;
            y=n;
        }
        Point(Point &p){
            x = p.x;
            y = p.y;
        }
        void show(){
            cout<<"\nPoint is=("<<x<<","<<y<<")";
        }
};
int main() {
    Point p1,p2(10,30);
    Point p3=p2;
    p1.show();
    p2.show();
    p3.show();
    return 0;
}
```

Constructor Overloading

Calling Default Constructor

Calling Parameterized Constructor

Calling Copy Constructor

	SVKM's NMIMS **Mukesh Patel School of Technology Management & Engineering / School of Technology Management & Engineering**	
B. Tech/MBA Tech	**Workbook**	**Academic Year- 2024-25**
Year:-First	**Subject:-** Programming for Problem Solving	**Semester: - First**

Operator Overloading

Write a program to find factorial of a number

Program without using operator overloading function

```
#include <iostream>
using namespace  std;

int fact(int n){
        int f=1;
        for(int i=1;i<=n;i++)
                f=f*i;
        return f;
}

int main() {
        int n,f;
        cout<<"Enter no";
        cin>>n;
        f = fact(n);
        cout<<"Factorial of "<<n<<" is "<<f;
        return 0;
}
```

Program using operator overloading function

```
#include <iostream>
using namespace std;

class OpTest{
        private:
                int n;
        public:
                OpTest(){
                        cout<<"Enter no";
                        cin>>n;
                }

                int operator !( ){
                        int f=1;
                        for(int i=1;i<=n;i++)
                                f=f*i;
                        return f;
                }
};

int main() {
        OpTest op;
        int f = !op;
        cout<<"Factorial is"<<f;
        return 0;
}
```

Operator Function

As we are using operators as function name, operator keyword is required

! is logical not operator. We changed its behavior to find factorial…

Calling operator ! function

To work with operator overloading, at least one operand should be object…

	SVKM's NMIMS **Mukesh Patel School of Technology Management & Engineering / School of Technology Management & Engineering**	
B. Tech/MBA Tech	**Workbook**	**Academic Year- 2024-25**
Year:-First	**Subject:-** Programming for Problem Solving	**Semester: - First**

…Polymorphism…

Overloading		
Two or more function have same name and different signature.		
May belongs to one class, Inheritance is not required		
It is refinement		
It is static polymorphism / compile time polymorphism /early binding		
PROGRAMMING Example		
Method Overloading	**Constructor Overloading**	**Operator Overloading**
- Multiple methods in a class with same name and different signature	- Multiple constructor in a class	- Changing the behavior of operator

class Test{
public:
void add(int a, int b){
cout<<a+b;
}
void add(float a, float b){
cout<<a+b;
}
};
	class Test{
	public:
	Test( ){
	cout<< "Constr1"
	}
	Test(int a){
	cout<< "Constr2"
	}
	};
		void operator+(int l)
		{
		// method body
		}
		- Class is required for overloading operator

	SVKM's NMIMS **Mukesh Patel School of Technology Management & Engineering / School of Technology Management & Engineering**	
B. Tech/MBA Tech	**Workbook**	**Academic Year- 2024-25**
Year:-First	**Subject:-** Programming for Problem Solving	**Semester: - First**

Inheritance

Why Inheritance?

- Reuse
- Avoids redundant coding
- Allow function Overriding.

Types of Inheritance:

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

Super Class:-

- A class which is getting extended by sub class is known as super class.
- Super class is also called as Base Class or Parent Class

Sub Class:-

- A class which is extending [taking] another class properties is known as sub class.
- Also called as extended class or Derived Class or Child Class.

Single Inheritance:

- One class as a parent class & other class as Child class.
- Extended class can inherit [take] properties from its Super class.

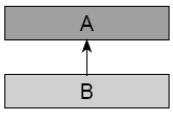


Fig:- Single Inheritance

- We can achieve single inheritance of interfaces also.

Multilevel Inheritance: -

- There is no limit to this chain of inheritance (as shown below) but getting down deeper to four or five levels makes code excessively complex.

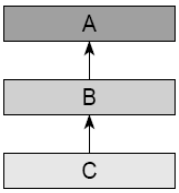


Fig:- Multilevel Inheritance.

- In above figure,
  - class A is Parent class of B
  - class B is child class of A & B is Parent of C.
  - class C is Child of B

Multiple Inheritance:-

- A class can inherit from more than one unrelated class.

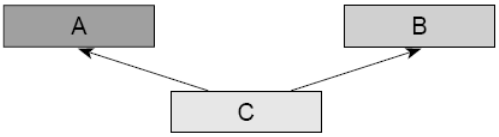


Fig:- Multiple Inheritance

Hierarchical Inheritance:-

- In hierarchical inheritance, more than one class can inherit from a single class.

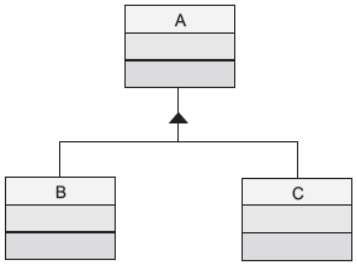


Fig:- Hierarchical Inheritance

- Class A is parent of both B and C.

Hybrid Inheritance: -

- is any combination of the above defined inheritances

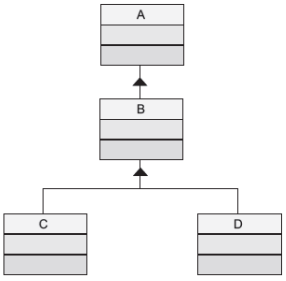


Fig:- Hybrid Inheritance.

Example of Multilevel Inheritance…

```
#include<iostream.h>
class student
{
        protected:
                        int roll_number;
        public:
                void get_number(int a){
                      roll_number=a;
                }
                void put_number( ){
                      cout<<" Roll number:"<<roll_number<<"\n";
                }

};
class test : public student
{
        protected:
                float sub1;
                float sub2;
        public:
                void test:: get_marks(float x, float y)
                {
                      sub1=x;
                      sub2=y;
                }
                void test:: put_marks()
                {
                      cout<<"marks in sub1 = "<<sub1<<"\n";
                      cout<<"marks in sub2 = "<<sub2<<"\n";
                 }
};
class result : public test
 {
         float total;
         public:
                 void result::display(void)
                 {    total=sub1+sub2;
                      put_number();
                      put_marks();
```

```
                cout<<"total = "<<total <<"\n";
            }
}
int main( )
{           result student1;
            student1.get_number(111);
            student1.get_marks(75.0,59.5);
            student1.display();
            return 0;
}
```

Output:-

Roll number:111
marks in sub1 = 75
marks in sub2 = 59.5
total =1

	SVKM's NMIMS **Mukesh Patel School of Technology Management & Engineering / School of Technology Management & Engineering**	
B. Tech/MBA Tech	**Workbook**	**Academic Year- 2024-25**
Year:-First	**Subject:-** Programming for Problem Solving	**Semester: - First**