Chapter 2. **Object Oriented Programming**

Programming Paradigm

A *programming paradigm* defines the methodology of designing and implementing programs using the key features and building blocks of a programming language.
It gives you an idea how problems are generally analysed and solved in a particular programming language.

Procedural programming:-

Procedural programming paradigm separates the function and data manipulated by them.

Object based programming:-

It is the latest programming paradigm that implement some features of object oriented programming but not all.

Object-oriented programming (**OOP**):
Object Oriented Programming is a programming paradigm that uses "objects" – data structures consisting of data fields and methods together with their interactions – to design applications and computer programs. Programming techniques may include features such as data abstraction, encapsulation, modularity, polymorphism, and inheritance.

Class:
A Class is a user defined datatype which contains the variables, properties and methods in it.
OR
A class is a group of objects that share common properties and relationships.

Object:
An object is an identifiable entity with some characteristic and behavior.

Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com

Basic concepts of oop

- Data abstraction
- Data encapsulation
- Modularity
- Inheritance
- Polymorphism

Data Abstraction:
It refers to the act of representing essential features without including the background details .

Data Encapsulation:
It means wrapping up data and associated functions into one single unit called class..
A class groups its members into three sections :public, private and protected, where private and protected members remain hidden from outside world and thereby helps in implementing data hiding.

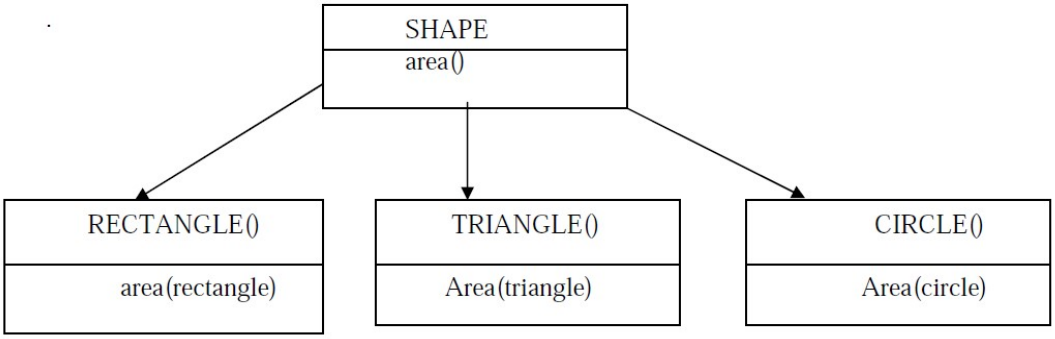
Modularity :
The act of partitioning a complex program into simpler fragments called modules is called as modularity.

Inheritance:
Inheritance is the capability of one class of things to derive capabilities or properties from another class.
Base class:- the class from which properties are being inherited is known as base class.
base class is also known as parent class or super class.

Derived class is also known as a child class or sub class. Inheritance helps in reusability of code , thus reducing the overall size of the program

Polymorphism:
• **Poly** means many and **morphs** mean form, so polymorphism means one name multiple forms.
• It is the ability for a message or data to be processed in more than one form.
• C++ implements Polymorhism through Function Overloading

Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com



Advantage of OOP

1. **Reuse of code:-** encapsulation allow class definition to be reused in other application.
2. **Ease of comprehension:**- oops codes are more near to the real world model's than other programming methodologies codes.
3. **Ease of fabrication and maintenance:**- the concept such as encapsulation , data abstraction allow for very clean design .

Disadvantage of OOP

1. With OOP classes tends to be overly generalized.
2. The relation among classes becomes artificial at time.
3. The oop program design is tricky.
4. Need proper planning and proper design for oop programming.

Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com

Chapter 3. Function overloading

Function overloading:- function having same name but different signature. Polymorphism is implemented by function overloading
Function argument list is known as signature.
Signature must be different in number and datatypes and not in return type only.
Every overloaded function must be defined separately.

Early binding or static binding?

Function calling matches the types and number of parameters at compile time ,this is known as compile time polymorphism.

Wap to calculate the area of circle, rectangle, square?

```
void area(float);
void area(int);
void area(int,int);
void main()
{
int  l,b,s,
float r;
cout<<"enter length of
radius";
cin>>r;
area( r) ;
cout<<"enter l and b of
rectangle";
cin>>l>>b;
area(l,b);
cout<<"area of square";
cin>>s;
}
area(s);
getch();
}
void area(int l,int b)
{
cout<<"area of rectangle
is"<<l*b;
}
void area(int r)
{
cout<<"area of circle
is"<<3.14*r*r;
}
void area(int s)
{
cout<<"area of square
is"<<s*s;
```


Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com

Calling overloaded functions:-

Overloaded function are called just like other functions. The number and type of argument determine which function should be invoked.

A function call first match the prototype available with number and type of arguments provided with the function call and then call the appropriate function for execution.

STEPS involved in matching best match:-

Function are called through argument matching. Argument matching invoves comparing the actual arguments of the call. There are three possible case

(A) A match:- in this a match is found.
(B) No match:- no match is found in this case.
(C) Ambiguous match:- more than one match is found

The compiler tries to find the best possible match for the function call. In order to find the best possible match the compiler follows the following steps:-

1. **Search for an exact match**:- if the type of actual argument exactly matches the type of one defined instance, the compiler invoke that particular instance.

   Eg void sum(int,int);
   Void sum(int,int,int);

   Sum(5,7);// exact match for void sum(int,int);

2. **A match through promotion:-** if no exact match is found , an attempt is made to achieve a match through promotion of actual argument

   Eg void fun(int);
   Void fun(float);

   fun('c'); match trough promotion, here char will be converted to int.

Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com

default argument vs function overloading

default argument gives the appearance of overloading, because the function may be called with an optional number of arguments

```
#include<iostream.h>
#include<conio.h>
Void sum(int a=5, int b=8)// here a and b have default argument
 {
Cout<<a+b<<"\n";
}
Void main()
{
 Sum( );
 Sum(4);
 Sum(3,6);
Getch();
}
```

sum( ) function can be called with an optional number of arguments

Output:

13
12
9

Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com

Chapter 4. Class and objects

Object:- it is an identifiable entity which has some characteristics and behaviours.
Class:- class is a way to bind variables and member function into single unit called as class.it is a collection of similar objects that share common properties.

Access specifiers in Classes:

Access specifiers are used to identify access rights for the data and member functions of the class.
There are three main types of access specifiers in C++
- private
- public
- protected

Private :-members declared as private can be used only by member functions and friends (classes or functions) of the class.
By default, functions and data declared within a class are private to that class and may be accessed only by members of the class.

Protected :-members declared as protected can be used by member functions and friends (classes or functions) of the class. Additionally, they can be used by classes derived from the class.

Public :-members declared as public can be used by any function.

Syntax:

```
Class  class_name
{
Private:
Data members;
Member function();
```


Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com

```
Public:
Data member;
Member function();
};
```

class member function definitions:-

Member function inside the classes are defined in two ways

(1) Inside the class (2) outside the class

Inside the class:- the function that are declared inside the classes are called as inline function

Outside the class:- A member function defined outside the class are done by

```
Return type Class name::function_name(parameter list)
{
  Body of function;
}
```

Accessing members of a class:-

The dot ('. ') used above is called the dot operator or class member access operator. The dot operator is used to connect the object and the member function. The private data of a class can be accessed only through the member function of that class.

Inline function:

The function whose body is defined inside the class is called inline function.

INLINE FUNCTIONS

- Inline functions definition starts with keyword inline
- The compiler replaces the function call statement with the function code itself(expansion) and then compiles the entire code.
- They run little faster than normal functions as function calling overheads are saved.

 A function can be declared inline by placing the keyword inline before it.

Example

```
inline  void Square (int a)
{ cout<<a*a;}
void main()
{.
 Square(4);        ⟶   { cout <<4*4;}
 Square(8)  ;      ⟶   { cout <<8*8; }
}
```

In place of function call , function body is substituted because Square () is inline function

Difference between class and structure

Class	structure
1. It is created by word class	1. It is created by word struct
2. It contains both data member and member Functions.	2. It contains only variables
3. All members are private by default	3. All members are public

Array within class

A class which contains array as its member variables.it can be either public or private.

```
Class abc
{
 Int roll;
Int age;
Char name[20];
Float Marks[5];// array within class
};
```

Scope rule and classes:-

The scope of a class depends upon its access specifiers(public,private, protected)

Global class:- the class that definition occurs outside the bodies of all functions in a program.

Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com

Local class:- these class are defined inside a function body. the object of this class cannot be used outside the function.

Global object:- an object is said to be global if it is declared outside the function body.

Local object:- this object is declared inside the function, so it can't be used outside the body of function.

```
Class Abc// global class
{
  Private:
   Int x;
   Int y;
Public:
 Void getdata();
 Void showdata();
};
Abc obj1;//global object

Void main()
{

   Class local// local class
   {
      Int age;

Public:
      Void takedata();
  };

Abc obj2;// local object

Obj1.getdata();
Obj1.showdata ();
}
```


Type of class functions

The member functions of a class can be categorized into three categories:

(i) Accessor functions.
(ii) Mutator functions.
(iii) Manager functions.

i) Accessor function allows us to access the values of objects. Accessor function cannot change the value of data members.

Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com

ii) Mutator functions:- these functions are used to change the value of object.

iii) Manager function:- constructors and destructors.

Nested class:- a class declared within another class is called as nested class.

```
class Nest
  {
    public:
     class Display
      {
       private:
        int s;
       public:
        void sum( int a, int b)
          {  s =a+b; }
        void show( )
          { cout << "\nSum of a and b is:: " << s;}
      };
  };
 void main()
  {
     Nest::Display x;
     x.sum(12, 10);
     x.show();
  }
```

The scope resolution operator(::)

There are two uses of the scope resolution operator in C++.
The first use being that a scope resolution operator is used to unhide the global variable that might have got hidden by the local variables. .
e.g.

```
int i = 10;
int main ()
{
int i = 20;
cout << i; // this prints the value 20
cout << ::i; // in order to use the global i one needs to prefix it with the scope resolution operator.
}
```

The second use of the operator is used to access the members declared in class scope.

Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com

The scope resolution operator (::) in C++ is used to define the already declared member functions .

Using objects:-

When a class is defined, no memory is allocated, memory is allocated during creation of its object. Object is the variable of any class.

It can be local object and global object.

Each members can be accessed by the help of object.

ARRAY OF OBJECTS:

An array having class type elements is known as array of objects, an array of objects is declared after the class definition is over.

Static classs member:

The member of a class may be declared as static. There may be static data member and static member function.

Static data member:- these data member are globally available for all the objects of that class type. these member usually store values common to the entire class.

Static data members must be defined outside the class definition.

```
Class x
{
  Static int count;
};
Int static count=0;
```


Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com

Function returning object:

Objects can not only be passed to functions but functions can also return an object

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

class data
{
    int a,b;
    public:
        void get();
        data sum (data,data);
        void show();
};

data sum(data a1,data a2)
{
    data a3;
    a3.a=a1.a+a2.a;
    a3.b=a1.b+a2.b;
    return a3;
}

main()
{
    clrscr();
    data a,b,c;
    a.get();
    b.get();
    c=sum(a,b);
    c.show();
}

void data::get()
{
    cout<<"PLEASE ENTER FOR A:->";
    cin>>a;
    cout<<"PLEASE ENTER FOR B:->";
    cin>>b;
}
void data::show()
{
    cout<<"A= "<<a<<endl;
    cout<<"B= "<<b<<endl;
}
```

Array of objects:- an array having class type object is known as array of objects. An array of objects is declared after the class definition is over…..

Static class members:

Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com

The members(data member and member function) of a class may be declared as static.
Static data member:- it is just like global variable for its class. this data member is available for all the objects of that class.
The static data member are usually maintained to store values common to the entire class.

Difference between static and ordinary data member

1. There is only one copy for entire class which is shared by all the objects of that class.
2. It is visible only within the class.

Two things must be done to make a static data members
It must be declared within the class
It must be Defined outside the class.
It is created by using word static .

```
Class x
{

Static int count;//declared inside the class

};
Int x:: count=0;// defined outside the class
```

Note:- We cannot have a static data member inside a local class.

Static member function
A member function that access only the static members of a class is declared as static.

```
Class x
{
 Static int count;
 Static void show()
{
 Cout<<count;} };
```

Difference between static function and ordinary function

Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com

1. Static function can access only static members of the same class
2. A static member function is invoked by using the class name instead of its objects.
   Class_name:: function_name();

```
Class x
{
 Int codeno, price;
Static int count;
Public:
Void getval(int I,int j)
{
 Codeno=I;Price=j;Count++;
}
Void display()
{
 Cout<<"code no is"<<codeno<<price is"<<price;
}
Static void disp()//static member function
{
 Cout<<"count is"<<count;
}
};
Int x::count=0;//definition of static data member outside class.
Void main()
{
 X ob;
Ob.getval(101,23);
Ob.display();
X::disp();//calling of static function
}
```


Ravinder kumar 8102060777 email:- ravinder.hzb@gmail.com