Chapter 1 :

Principles of Object-Oriented Programming

Reference : E. Balaguruswamy Object Oriented Programming With C++; chapter-1.

Topics:

- **Topic 1: Basics of procedure oriented programming & object oriented programming**
- **Topic 2: Basic concept of object oriented programming**
- **Topic 3: Basic C++ program**

Questions:

Topic 1: Basics of procedure oriented programming & object oriented programming

1. What is procedure(or structured) oriented programming(POP)? What is object oriented programming(OOP)? How does the POP(or structured oriented programming) differ from OOP? Write down the features of object oriented programming?
2. Write down the benefits of OOP.
3. Write down the application of OOP.
4. Basic organization of data and function in an OOP. or,Describe OOP paradigm.
5. Difference between object based and object oriented programming.
6. Explain system design. Explain briefly the steps involved in the OOP design approach.

Topic 2: Basic concept of object oriented programming

7. Write down the basic concepts related to OOP.
8. What is class and object? Differentiate between object and class.

9. What is encapsulation(data binding/data hiding)? Encapsulation reduces the complexity-justify your answer.
10. What is data abstraction? Or, what is abstract data types(ADT)?
11. Differentiate between data abstraction and encapsulation.
12. What is inheritance? What is reusability? How do you achieve reusability in C++?
13. What is polymorphism? Differentiate between polymorphism and inheritance.
14. What is dynamic binding(or late binding)? How is it useful in OOP?
15. What is message passing?
16. What is void data type?
17. Describe the major part of C++ programming language.

Topic 3: Basic C++ program

18. Write a c++ program that evaluate the following series :

$$sum = 1 + (1/2)^2 + (1/3)^2 + (1/4)^2 + \ldots\ldots\ldots\ldots\ldots$$

19. Write a c++ program that evaluate the following series :

$$sum = 1 + (1/2)^2 + (1/3)^3 + (1/4)^4 + \ldots\ldots + (1/n)^n$$

20. Write a c++ program that evaluate the following series :

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

21. Write a c++ program that evaluate the following series :

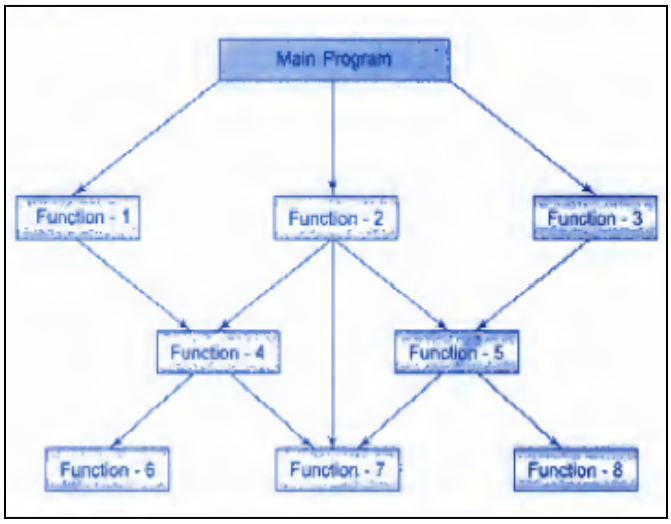
$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots\ldots$$

Q:1::What is procedure(or structured) oriented programming(POP)? What is object oriented programming(OOP)? Write down the striking features of object oriented programming? How does the POP(or structured oriented programming) differ from OOP?

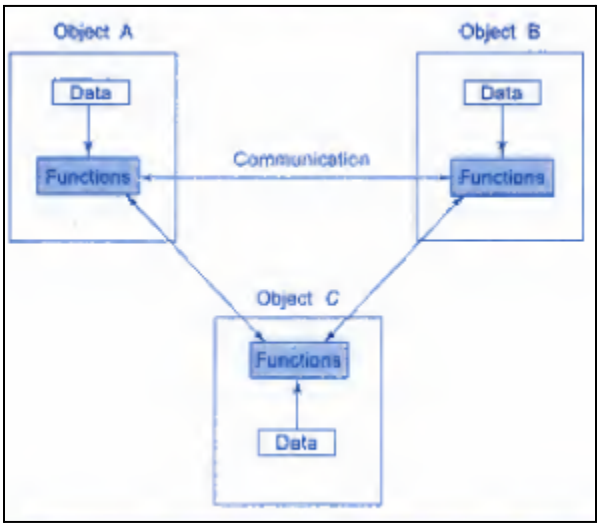
Answer:

Procedure(or structured) Oriented Programming(POP): Procedure-oriented programming basically consists of writing a list of instructions (or actions) for the computer to follow, and organizing these instructions into groups known as functions. Some characteristics exhibited by procedure-oriented programming are:

- Emphasis is on doing things(algorithms)
- Large programs are divided into smaller programs known as functions.
- Most of the functions share global data.
- Data move openly around the system from function to function.
- Functions transform data from one form to another.
- Employs top-down approach in program design.



Fig(1.1): typical structure of POP



Fig(1.2): typical structure of OOP

Object Oriented Programming(OOP) : OOP allows decomposition of a problem into a number of entities called objects and then builds data and functions around these objects.OOP treats data as a critical element in the program development and does not allow it to flow freely around the system.

Some of the striking features of object-oriented programming :

- Emphasis is on data rather than procedure,
- Programs are divided into what are known as objects
- Data structures are designed such that they characterize the objects
- Functions are operate on the data of an object are tied together in the data structure.
- Data is hidden and can not be accessed by external functions
- Object may communicate with each other through functions
- New data and functions can be easily added whenever necessary
- Follows bottom-up approach in program design

Difference between OOP and POP:

Sr. No.	Key	OOP	POP
1	Definition	OOP stands for Object Oriented Programing.	POP stands for Procedural Oriented Programming.
2	Approach	OOP follows bottom up approach.	POP follows top down approach.
3	Division	A program is divided to objects and their interactions.	A program is divided into funtions and they interacts.
4	Inheritance supported	Inheritance is supported.	Inheritance is not supported.
5	Access control	Access control is supported via access modifiers.	No access modifiers are supported.
6	Data Hiding	Encapsulation is used to hide data.	No data hiding present. Data is globally accessible.
7	Example	C++, Java	C, Pascal

Q:2:: Write down the benefits of OOP.

Answer: OOP offers several benefits to both the program designer and the user. Object-orientation contributes to the solution of many problems associated with the development and quality of software products. The new technology promises greater programmer productivity, better quality of software and lesser maintenance cost. The principal advantages are:

• Through inheritance we can eliminate redundant code and extend the use of existing classes.

• We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads to save development time and higher productivity.

• The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program,

• It is possible to have multiple instances of an object to co-exist without any interference.

• It is possible to map objects in the problem domain to those in the program.

• It is easy to partition the work in a project based on objects.

• The data-centered design approach enables us to capture more details of a model in implementable form.

• Object-oriented systems can be easily upgraded from small to large systems.

• Message passing techniques, for communication between objects makes the interface descriptions with external systems are much simpler.

• Software complexity can be easily managed.

Q:3:: Write down the application of OOP.

Answer : Applications are beginning to gain importance in many areas. The most popular application of object oriented programming up to now has been in the area of user interface design such as windows.Hundreds of windowing systems have been developed, using the OOP techniques. Real-business systems are often much more complex and contain many more objects with complicated attributes and methods. OOP

is useful in these types of applications because it can simplify a complex problem. The-promising areas for application of OOP include:

• Real-time systems
• Simulation and modeling
• Object-oriented databases
• Hypertext,hypermedia.
• AI and expert systems
• Neural networks and parallel programming
• Decision support and office automation systems
• CIM/CAM/CAD systems

The richness of the OOP environment has enabled the software industry to improve not only the quality of software systems but also its productivity. Object-oriented technology is certainly changing the way software engineers think,analyze,design and implement systems.

Q:4:: Basic organization of data and function in an OOP. or,Describe OOP paradigm.

Answer:

OOP treats data a critical element in the program development and does not allow it to flow freely around the system. It ties data more closely to the functions that operate on it and protects it from accidental modification from outside functions. OOP allows us to decompose a problem into a number of entities called objects and then build data and functions around these entities. The organization of data and functions in OOP is shown in Fig. (1.3). OOP is an approach that provides a way of modularizing programs by creating partitioned memory areas for both data and functions that can be used as templates for creating copies of such modules on demand. The data of an object can be accessed by the functions associated with that object. However, functions of one object can access the functions of other objects.

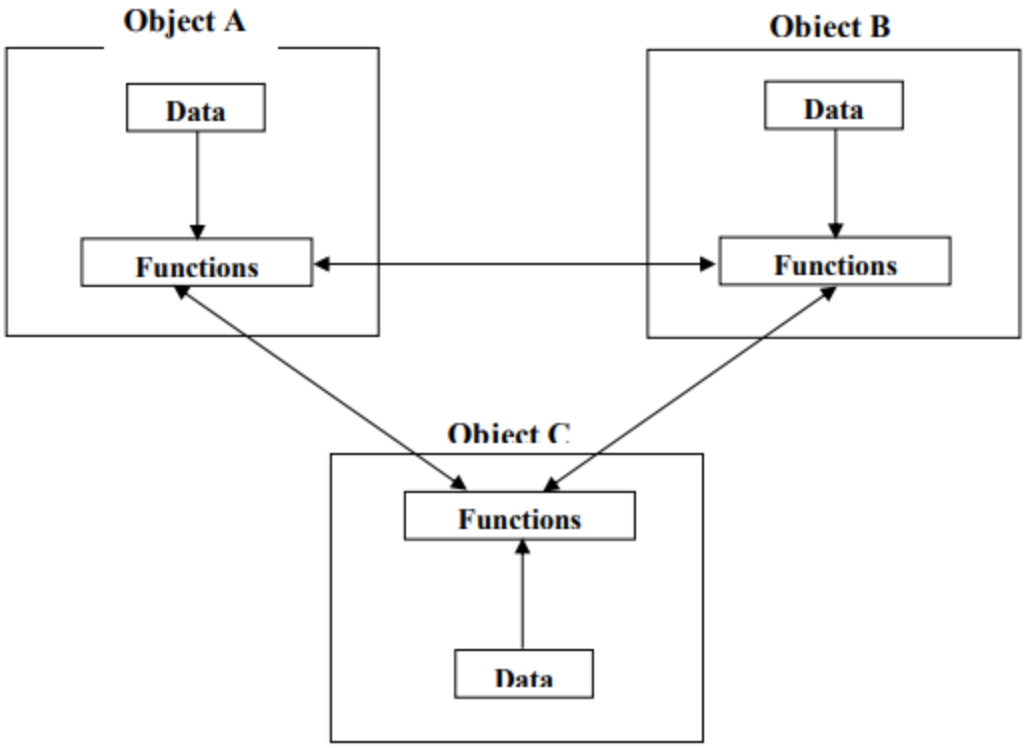


Fig. (1.3) organization of data and function in OOP

Q:5:: Difference between object based and object oriented programming.

Answer:

	object oriented programming		**object based programming**
1.	Object-oriented languages follow all the concepts of OOPs like - encapsulation, abstraction, inheritance, polymorphism.	1.	The object-based language doesn't follow all the concepts of OOPs like inheritance and polymorphism.
2.	Object-oriented languages do not have the inbuilt objects.	2.	Object-based languages have the inbuilt objects, for example, JavaScript has window object.
3.	Examples of object-oriented programming are Java, C#, C++, Smalltalk, etc.	3.	Examples of object-based languages are JavaScript, VBScript, etc.

Q:6:: Explain system design. Explain briefly the steps involved in the OOP design approach.

Answer: Design is concerned with the mapping of objects in the problem space into objects in the solution space and creating an overall structure(architectural model) and computational model of the system.This stage normally uses the bottom up approach to build the structure of the system and the top down functional decomposition approach to design the class member functions that provide services.The object oriented design (OOD) approach may involve the following 6 steps:

1. Review of objects created in the analysis phase
2. Specification of class dependencies
3. Organization of class hierarchies
4. Design of classes
5. Design of member functions
6. Design of driver program (i.e. main function())

Q:7:: Write down the basic concepts related to OOP.

Answer: Basic concepts related to OOP are:

1. Class
2. Object
3. Data abstraction(abstract data type)
4. Encapsulation(data binding and data hiding)
5. Inheritance
6. Polymorphism
7. Dynamic Binding
8. Message Passing

Q:8::What is class and object? Differentiate between object and class.

Answer:

Class : Classes are user-defined data types that act as the prototype or template or blueprint for individual objects.

Objects : Objects are instances or variables of a class created with specifically defined data. Objects can correspond to real-world objects or an abstract entity.

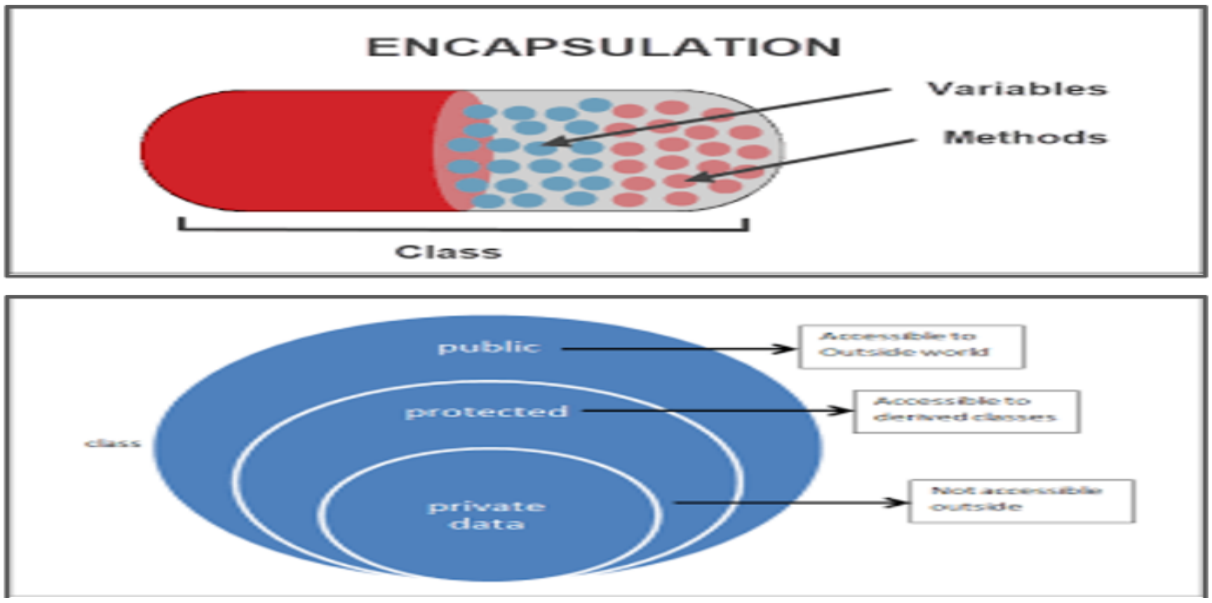
Differentiate between object and class:

Class	Object
A class is a blueprint from which you can create the instance, i.e., objects.	An object is the instance of the class, which helps programmers to use variables and methods from inside the class.
A class is used to bind data as well as methods together as a single unit.	Object acts like a variable of the class.
Classes have logical existence.	Objects have a physical existence.
A class doesn't take any memory spaces when a programmer creates one.	An object takes memory when a programmer creates one.
The class has to be declared only once.	Objects can be declared several times depending on the requirement.

Q:9:: What is encapsulation(data binding/data hiding)? Encapsulation reduces the complexity-justify your answer.

Answer: chapter 2+5- Q::3

The binding or wrapping up of data and functions, into a single unit (called class) is known as encapsulation. Data encapsulation is the most striking feature of a class.The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it. These functions provide the interface between the object's data and the program.This insulation of the data from direct access by the program is called data hiding or information hiding.



Fig(1.4) : Encapsulation(data or information hiding/binding)

Every class is an example of encapsulation because we write everything within the class only that binds variables and functions together and hides their complexity from other classes as well as safes from outside interference and misuse.In the encapsulation technique, we declare fields as private in the class to prevent other classes from accessing them directly. The required encapsulated data can be accessed by using the public member. If the field is declared private in the class then it cannot be accessed by anyone from outside the class and hides the field within the class. Therefore, it is also called data hiding.

Complexity mainly arises from a lot of connections between various parts of the system. Encapsulation hides some parts of the system so they cannot be connected to, hence, less complexity in the system.

Q:10:: What is data abstraction? Or, What are abstract data types(ADT)?

Answer: Abstraction refers to the act of representing essential features without including details or background. Class helps us to group data members and member functions using available access specifiers. A class can decide which data member will be visible to the outside world and which is not.Since classes use the concept of data abstraction they are known as abstract data types(ADT).

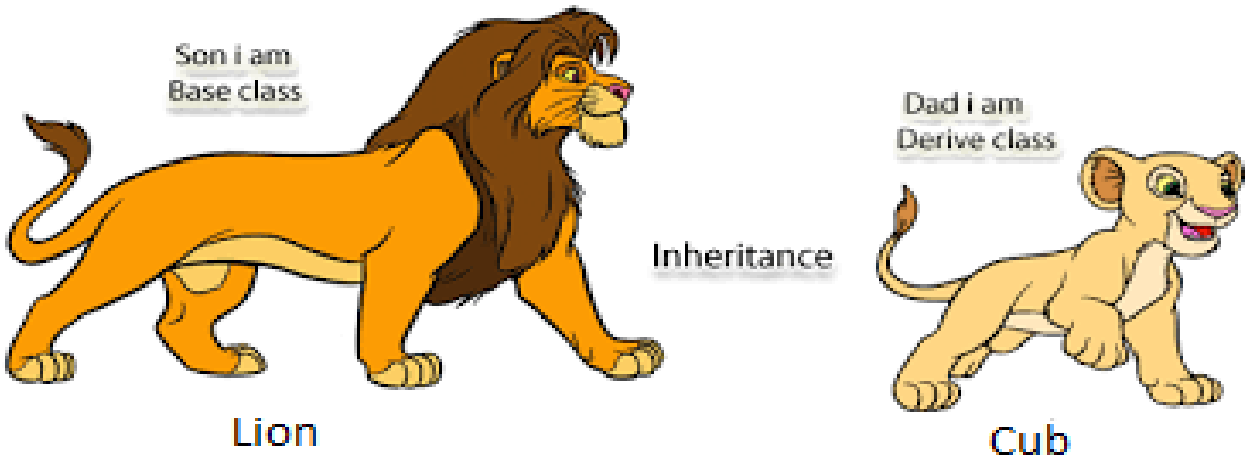
Q:11:: Differentiate between data abstraction and encapsulation.

Answer:

S.No	Abstraction	Encapsulation
1.	It is the process of gaining information.	It is a method that helps wrap up data into a single module.
2.	The problems in this technique are solved at the interface level.	Problems in encapsulation are solved at the implementation level.
3.	It helps hide the unwanted details/information.	It helps hide data using a single entity, or using a unit with the help of method that helps protect the information.
4.	It can be implemented using abstract classes and interfaces.	It can be implemented using access modifiers like public, private and protected.

Q:12:: What is inheritance? What is reusability? How do you achieve reusability in C++?

Answer:



Inheritance: Inheritance is the process by which object of one class acquire the properties of objects of another class.It supports the concept of hierarchical classification.For example, the bird 'robin' is a part of the class 'flying bird" which in again a part of the class "bird". The principle behind this sort of division is that each derived class shares common characteristics with the class from which it is derived as illustrated in Fig. 1.5.

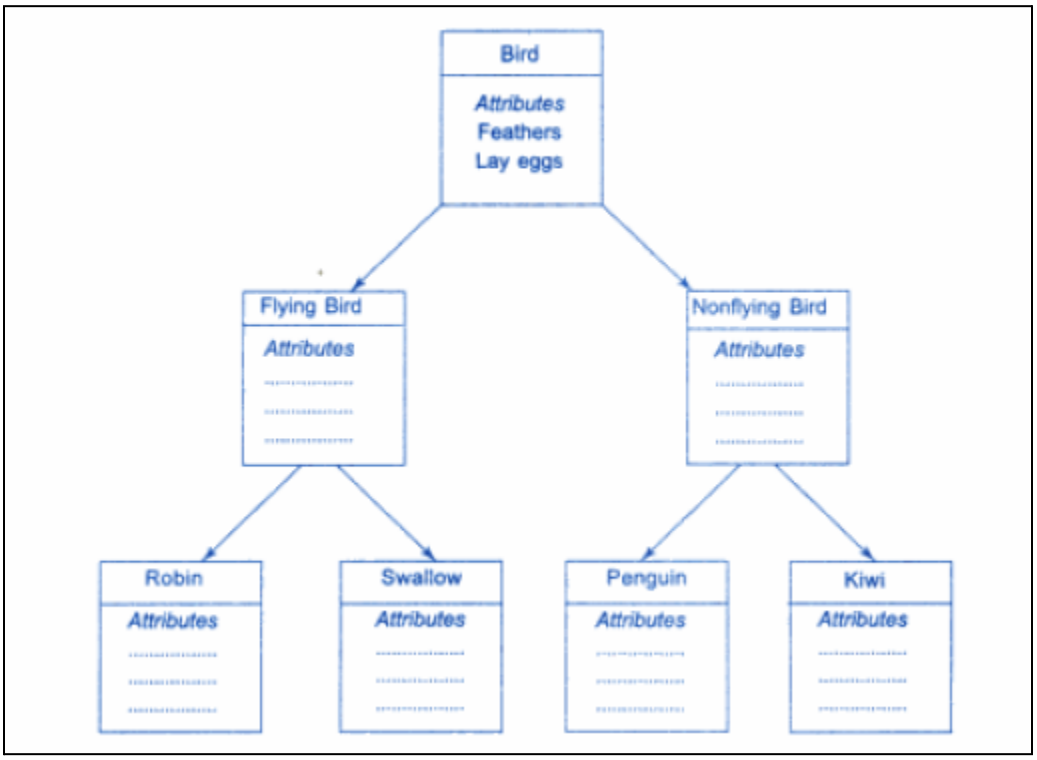


Fig. 1.5 : inheritance

Reusability: Inheritance provides the idea of reusability, this means that we can add extra features to an existing class without modifying it by deriving a new class from the class. The new class will have the combined features of both classes.The real appeal and power of the inheritance mechanism is that it allows the programmer to reuse a class that is almost but not exactly, what he wants and to tailor the class in such a way that it does not introduce any undesirable side-effects into the rest of the classes.Note that each subclass defines only those features that are unique to it. Without the use of classification each class would have to explicitly include all of its features.

Q:13:: What is polymorphism? Differentiate between polymorphism and inheritance.

Answer: <u>Polymorphism:</u> Polymorphism, a Greek term means, the ability to take more than one form. An operation may represent different behaviors in different instances. The behavior depends upon the types of data used in the operation.

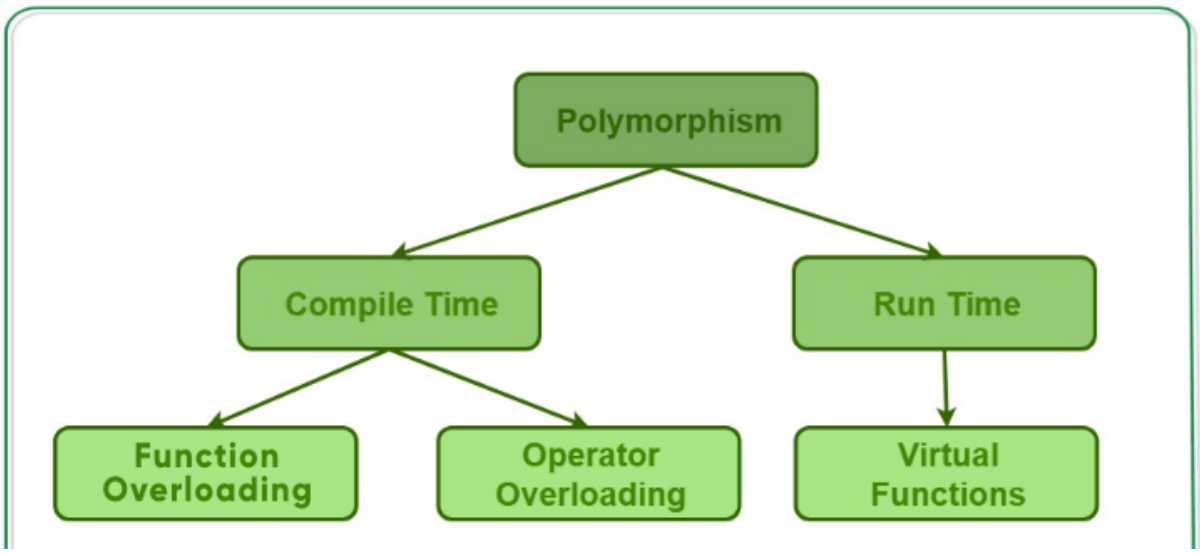


Figure 1.6 illustrates that a single function name can be used to handle different number and different types of arguments. This is something similar to a particular word having several different meanings depending on the context. Using a single function name to perform different types of tasks known function overloading.

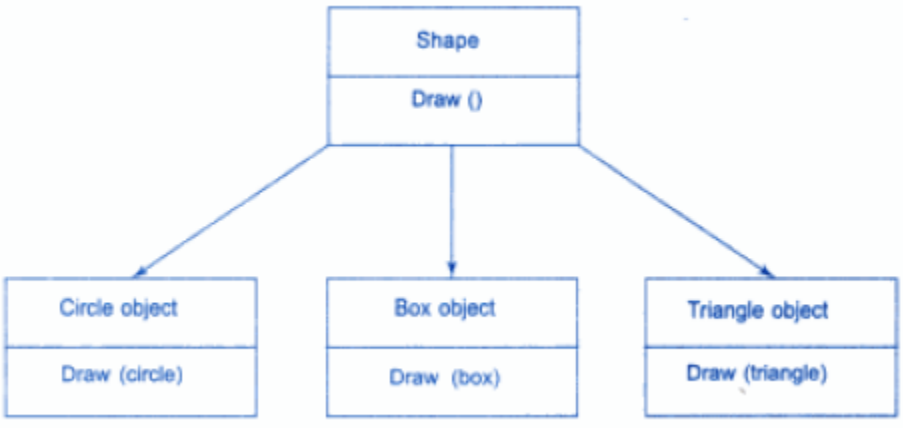


Fig. 1.6 : polymorphism

Differentiate between polymorphism and inheritance :

BASIS FOR COMPARISON	INHERITANCE	POLYMORPHISM
Basic	Inheritance is creating a new class using the properties of the already existing class.	Polymorphism is basically a common interface for multiple forms.
Implementation	Inheritance is basically implemented in classes.	Polymorphism is basically implemented on function / methods.
Use	To support the concept of reusability in OOP and reduce the length of code.	Allows object to decide which form of the function to be invoked when, at compile time(overloading) as well as run time(overriding).
Forms	Inheritance may be a single inheritance, multiple inheritance, multilevel inheritance, hierarchical inheritance and hybrid inheritance.	Polymorphism may be a compile-time polymorphism (overloading) or run-time polymorphism (overriding).
Example	The class 'table' can inherit the feature of the class 'furniture', as a 'table' is a 'furniture'.	The class 'study_table' can also have function 'set_color()' and a class 'Dining_table' can also have function 'set_color()' so, which form of the set_color() function to invoke can be decided at both, compile time and run time.

Q:14:: What is dynamic binding(or late binding)? How is it useful in OOP?

Answer:

Dynamic binding(or late binding): Dynamic Binding also known as late binding means that the code associated with a given procedure call is not known until the time of the call at run time. It is associated with inheritance and polymorphism.

A function call associated with a polymorphic reference depends on the dynamic type of that reference.Consider the procedure "draw" in Fig. 1.7. By inheritance, every object will have this

procedure. Its algorithm is, however, unique to each object and so the draw procedure will be redefined in each class that defines the object. At run-time, the code matching the object under current reference will be called.

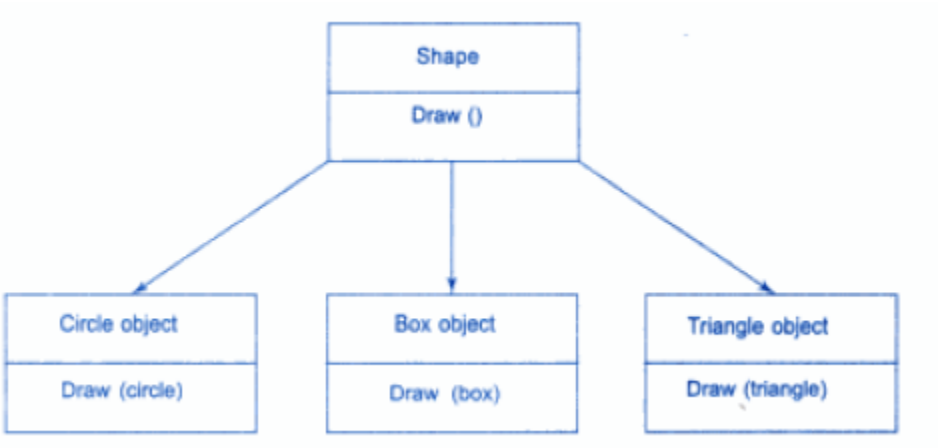
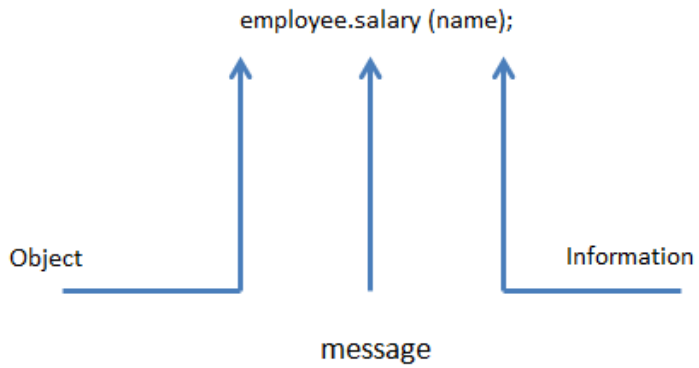


Fig. 1.7 : polymorphism as dynamic binding

Q:15:: What is message passing?

Answer: A message passing involves specifying the name of the object,name of the function(message) and the information to be sent. A message for an object is a request for execution of a procedure(function) and therefore will invoke a function in the receiving order that generates the desired result. Example :



Q:16:: What is void data type?

Answer: The void type specifies that no value is available. It is used in three kinds of situations:

i)When it is used as a function return type: Void return type specifies that the function does not return a value. Example: void add(int a,int b);

ii)Function arguments as void : Void parameter specifies that the function takes no parameters. Example : int add(void);

iii)Pointers to void : It specifies that the pointer is "universal" and it can point to anything. When we want to access data pointed by a void pointer, first we have to type cast it.

Example : void *a;

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

int main()
{
    int b=10;
    char c='M';
    void *a;
    //void pointer to integer pointer
    a= &b;
    cout<<"void a pointer point to :"<<*(int*)a<<endl;
    cout<<"void a pointer value :"<<a<<endl;
    cout<<"address of b variable,&b = "<<&b<<endl<<endl;
    //void pointer to character pointer
    a= &c;
    cout<<"void a pointer point to :"<<*(char*)a<<endl;
    cout<<"void a pointer value :"<<a<<endl;
    cout<<"address of c variable,&c = "<<(void*)&c<<endl<<endl;
    /*by using & with a character type variable,we can't get its
    address ,to display character type variable address,syntax:
    (void*)&character-variable */
    return 0;
}
```

```
void a pointer point to :10
void a pointer value :0x61fe14
address of b variable,&b = 0x61fe14

void a pointer point to :M
void a pointer value :0x61fe13
address of c variable,&c = 0x61fe13
```

Q:17::Describe the major part of C++ programming language.

Answer: C++ is an object oriented programming language. Here is the basic structure or major parts of a C++ program:

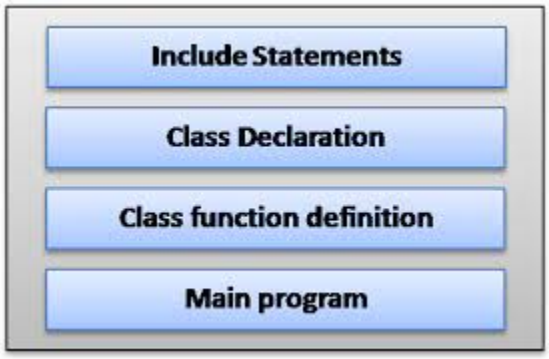


Fig 1.8 : major parts of a C++ program

Include statement: All required header files are included in this section.

Class declaration: All classes related to the program are declared(optional) and properly defined in this section.

Class function definition: Class functions are known as member functions.The member functions which are only declared within the class definition section, those member functions can be defined properly in this section.

Main program: It denotes the main() function section of the program where we create objects of classes and through objects we access the members of the classes.Within main() function we create another function or call another function which is already exists and can also perform any type of operation as we do in structured oriented programming.

Example: Here “item” is a class where t1 is an object of the class “ item”. Through the t1 object we can access the member variable and member function of the class “item”.

```cpp
//include section
#include <iostream>
using namespace std;
//class definition section
class item
{
    int a;
public:
    void getdata()
    {
        cout<<"Enter an integer : ";
        cin>>a;
    }
    void putdata();
};
//member function definition section
void item::putdata()
{
    cout<<"a = "<<a;
}
//main program section
int main()
{
    item t1;
    t1.getdata();
    t1.putdata();
    return 0;
}
```

```
Enter an integer : 5
a = 5
Process returned 0 (0x0)   execution time : 2.969 s
Press any key to continue.
```

Q:18:: Write a c++ program that evaluate the following series :

$$sum = 1 + (1/2)^2 + (1/3)^2 + (1/4)^2 + \ldots\ldots\ldots\ldots$$

Answer :

```cpp
#include<iostream>
#include<math.h>
#include <iomanip>
using namespace std;

int main()
{
    int i,n;
    double f,temp = 0.0,sum = 0.0,p = 2.0;

    cout<<"Enter the no. of terms :";
    cin>>n;
    for (i = 1; i<=n ; i++)
    {
        f = 1.0/i;
        temp = pow(f,p);  //double pow(double,double);
        sum = sum + temp;
        temp = 0.0;
    }
    cout<<"Sum of the series upto "<<n<<"th terms = ";
    std::cout << std::fixed << std::setprecision(6) << sum;
    return 0;
}
```

```
Enter the no. of terms :100
Sum of the series upto 100th terms = 1.634984

Process returned 0 (0x0)   execution time : 8.179 s
Press any key to continue.
```

Github code -

https://github.com/SyedaJannatul/DIIT_Object-Oriented-Programming/blob/e9d5de53292169e11a96fb965125175867111eed/chapter%201_principles%20of%20Object-Oriented%20programming/chapter%201_class_code/question_18_code.cpp

Q:19::Write a c++ program that evaluate the following series :

$$sum = 1 + (1/2)^2 + (1/3)^3 + (1/4)^4 + \ldots\ldots + (1/n)^n$$

Answer:

```
#include<iostream>
#include<math.h>
#include <iomanip>
using namespace std;

int main()
{
    int i,n;
    double f,temp = 0.0,sum = 0.0;

    cout<<"Enter the no. of terms : ";
    cin>>n;
    for (i = 1; i<=n ; i++)
    {
        f = 1.0/i;
        temp = pow(f,i);  //double pow(double,double);
        sum = sum + temp;
        temp = 0.0;
    }
    cout<<"Sum of the series upto "<<n<<"th terms = ";
    std::cout << std::fixed << std::setprecision(6) << sum<<endl;
    return 0;
}
```

```
Enter the no. of terms : 6
Sum of the series upto 6th terms = 1.291285

Process returned 0 (0x0)   execution time : 2.828 s
Press any key to continue.
```

Github code-

https://github.com/SyedaJannatul/DIIT_Object-Oriented-Programming/blob/e9d5de53292169e11a96fb965125175867111eed/chapter%201_principles%20of%20Object-Oriented%20programming/chapter%201_class_code/question_19_code.cpp