1...

Introduction to C++

Objectives...

- To study basic concepts, features, advantages and applications of OOP.
- To learn applications and features of C++.
- To understand Input and Output operator in C++.
- To learn Simple C++ program.

1.1 INTRODUCTION

- During the late 1970's and early 1980's 'C' had become very popular language. C was a structural programming language. However, for large and complex program, the structural approach failed to show the desired results in terms of bug-free, easy-to-maintain and reusable programs. To deal with this a new way to program was invented "Object Oriented Programming".
- Object Oriented Programming (OOP) is playing an increasingly significant role in the analysis, design and implementation of software systems. OOP languages provide the programmer the ability to create class hierarchies, instantiate objects and send messages between objects to process themselves.
- Object oriented means to organize software as a collection of discrete, real-world objects that incorporate both data structure and behavior.
- Object Oriented Modeling and Design (OOMD) is a new way of thinking about problems using models organized around real-world concepts. The fundamental construct is the object, which combines both data structure and behavior in a single entity.
- Object oriented models are useful for understanding problems, communicating with application experts, modeling enterprises, preparing documentation and designing programs and databases.
- Object oriented development is a conceptual process independent of a programming language until the final stage.
- Object Oriented Programming is a new way of organizing code and data that promises increased control over the complexity of software development process.

- The prime purpose of C++ programming was to add object orientation to the C programming language, which is in itself one of the most powerful programming languages.

1.1.1 Need of OOP

- The use of structured programming made more complex programs, less complex and easier to understand. But even with structured programming, when a program reaches a certain size, its complexity exceeds that which a programmer can manage.
- To deal with this approach, a new way to program, was invented **"Object Oriented Programming"**. This method of programming reduces the program complexity by incorporate rating features like inheritance, encapsulation and polymorphism.
- OOP is a way of programming help the programmer to comprehend and manage large and complex programs.
- Object Oriented Programming models real-world objects with software counterparts. It takes advantage of class relationships where objects of certain class have the same characteristics.
- Object Oriented Programming (OOP) allows us to decompose a problem into a number of entities called objects.
- OOP takes advantages of inheritance and multiple inheritance relationship where newly created classes of objects are derived by absorbing characteristics of existing classes and adding unique characteristics of their own.
- OOP gives us a more natural way to view the programming process, namely by modeling real-world objects. C++ was the first object oriented programming language. It was invented by **"Bjarne Stroustrup"**. This language was initially called as **"C with classes"**.

1.1.2 Object Oriented Vs Procedure Oriented Programming [W-18]

Table 1.1: Difference between Object Oriented Programming and Procedure Oriented Programming

Object Oriented Programming (OOP)	Procedure Oriented Programming (POP)
1. Object oriented programming language is object oriented.	1. Procedural programming languages tend to be action oriented.
2. In OOP, the unit of programming is the class.	2. In procedural programming, unit of programming is the function.
3. OOP programmers concentrate on creating their own user-defined types called classes and components. Each class contains data as well as the set of functions that manipulate that data. The data components of a class are called data members. The function components of a class are called member functions. The instance of a class is called an object.	3. In procedural oriented programming programmers concentrate on writing functions. Groups of actions that perform some common task are formed into functions and functions are grouped to form programs.

contd. ...

4. OOP trace on data.	4. POP trace on procedures.
5. Follow bottom-up approach in program design.	5. Follow top-down approach in program design.
6. Basic building block of OOP is object.	6. Basic building block of POP is function.
7. OOP has access specifiers named Public, Private, and Protected.	7. POP does not have any access specifiers.
8. Examples of OOP are C++, JAVA, VB.NET, C#.NET.	8. Examples of POP are C, VB, FORTRAN, Pascal.
9. In OOP, importance is given to the data rather than procedures or functions because it works as a real world.	9. In POP, importance is not given to data but to functions as well as sequence of actions to be done.
10. OOP provide Data Hiding so it provides more security.	10. POP does not have any proper way for hiding data so it is less secure.
11. In OOP, overloading is possible in the form of Function Overloading and Operator Overloading.	11. In POP, Overloading is not possible.
12. In OOP, objects can move and communicate with each other through member functions.	12. In POP, data can move freely from function to function in the system.

1.2 OBJECT ORIENTED CONCEPTS

1.2.1 Classes

- A class is a group of objects with similar properties (attributes), common behaviour (operations), common relationship to other objects and common semantics.
- Classes are user defined datatypes and behave like the built-in types of programming languages.
- The entire set of data and code of an object can be made user defined data type with the help of class.
- Once, a class has been defined we can create any number of objects belonging to the same class. Person, company, process, window are some examples of classes.
- Fig. 1.1 shows class Person, with two objects i.e. Ram and Shyam.

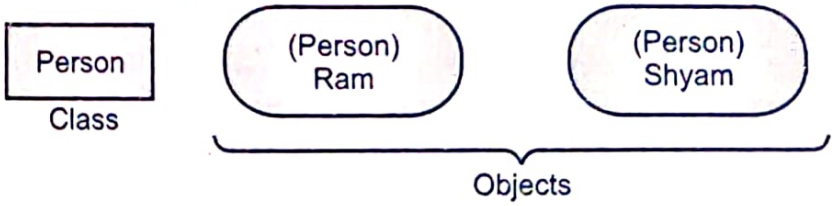


Fig. 1.1: Classes and Objects

1.2.2 Objects

- Objects are discrete, distinguishable entities.
- Objects are basic run-time entities.

- A paragraph in a document, a window on a workstation, the white queen in a chess game is some examples of Objects.
- Objects can be concrete such as a file in a file system or conceptual such as scheduling policy in a multiprocessing operating system.
- Each object has its own inherent identity.

Objects serve two purposes:

1. Objects promote understanding of the real-world and provide a practical basis for computer implementation.
2. Decomposition of a problem into objects depends on judgment and the nature of the problem. Program objects should be chosen such that they match closely with the real-world objects.

- Fig. 1.2 shows different styles of object representation.

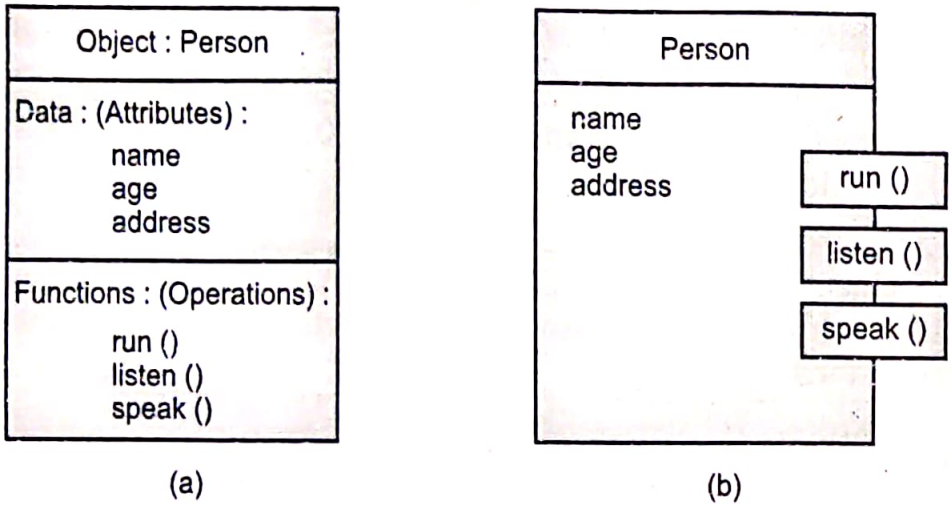


Fig. 1.2: Different styles to represent object

- In a program, objects interact by sending messages to one another.
- Each object contains data and code to manipulate the data.
- Objects can interact without having to know details of each other's data or code. It is sufficient to know the type of message accepted and the type of response returned by the objects.

1.2.3 Data Abstraction

[S-19, 22, 23; W-22]

- Abstraction is the selective examination of certain aspects of a problem.
- The goal of abstraction is to isolate those aspects that are important for some purpose and suppress those aspects that are unimportant.
- In other words, we can say that abstraction is the act of representing essential features without including the background details or explanations.
- Data abstraction is the process of defining a data type, often called Abstract Data Type (ADT), together with the principle of data hiding.
- The definition of an ADT involves specifying the internal representation of the ADTs data as well as the functions to be used by others to manipulate the ADT.
- In OOP, classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost and functions to operate these attributes. Classes are also known as ADTs.

1.2.4 Encapsulation

[S-22, 23; W-18, 22]

- The wrapping up of data and functions into a single unit i.e. class is called as Encapsulation.
- In other words, "the binding of data and function into a single unit (class) is called as encapsulation".
- The data is not accessible to the outside world and only those functions which are wrapped in the class can access it.
- These functions which are wrapped in the class provide the interface between the objects data and the program.
- Encapsulation is a mechanism that keeps the data and code safe from external interference and misuse. This insulation of the data from direct access by the program is called data hiding which is also known as data encapsulation.

1.2.5 Inheritance

- In inheritance process we can create new classes known as derived or child or sub-class from the existing class known as base or parent or super class.
- Inheritance is a process by which object of one class (derived class) can acquire the properties of objects of another class (base class).
- The concept of inheritance provides the idea of reusability. This means that we can add additional features to an existing class without modifying it. This is made possible by deriving a new class from the existing one. The new class will have the combined features of both the classes.
- Fig. 1.3 shows class of graphic geometric figures. Move, Select, Rotate and Display are operations inherited by all subclasses. Scale applies to one and two-dimensional figures. Fill applies only two-dimensional figures.

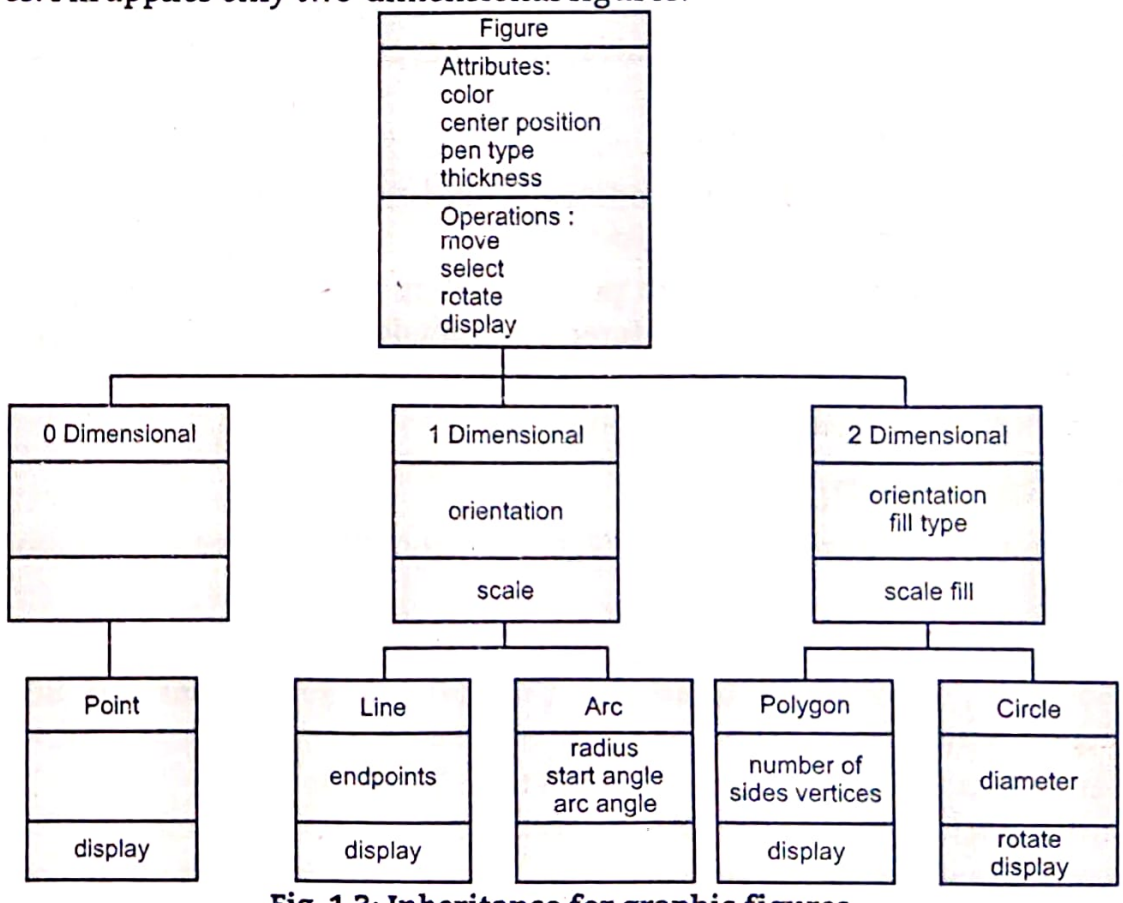


Fig. 1.3: Inheritance for graphic figures

1.2.6 Polymorphism

- In Polymorphism, "poly" means many and "orphism" means form which forms many forms of one things.
- In other words, "Polymorphism means one thing different or many forms" i.e. the ability to take more than one form.
- Polymorphism plays an important role in allowing objects having different internal structures to share the same external interfaces.
- In OOP, polymorphism refers to the fact that a single operation can have different behaviour in different objects. In other words, different objects react differently to the same message.
- For example, consider the operation of addition. For two numbers, the addition should generate the sum. The operation of addition is expressed by a single operator +. You can use the expression x + y to denote the sum of x and y for many different types of x and y integers, floating point numbers and complex numbers and even the concatenation of two strings.
- Similarly, suppose a number of geometrical shapes all respond to the message draw.
- Each object reacts to this message by displaying its shape on a display screen. Obviously, the actual mechanism for displaying the object differs from one shape to another, but all shapes perform this task in response to the same message.

1.2.7 Dynamic Binding [W-22]

- Binding refers to the linking of a procedure call to the code to be executed in response to the call. This link can be physical or conceptual connection between object instances.
- Link is an instance of an association.
- Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run-time.
- Dynamic binding is associated with polymorphism and inheritance. A function call associated with a polymorphic reference depends on the dynamic type of that reference.
- Dynamic binding also known as Late Binding.

1.2.8 Message Passing

- The act of communicating with an object to get something done is called as Message Passing.
- OOP consists of set of objects that communicate with each other.
- The process of programming in an object oriented language, therefore involves the following basic steps:
  1. Creating classes that define objects and their behaviour,
  2. Creating objects from class definitions,
  3. Establishing communication among objects.

- Objects communicate with one another by sending and receiving information.
- A message for an object is a request for execution of a procedure and therefore will invoke a function (procedure) in the receiving object that generates the desired result.
- Message passing involves specifying the name of the object, the name of the function (message) and the information to be sent.
- Example:

```
employee.salary (name);
```

Object Message Information

- Objects have a life cycle. They can be created and destroyed. Communication with an object is feasible as along as it is alive. The way of representing message passing is shown in Fig. 1.4.

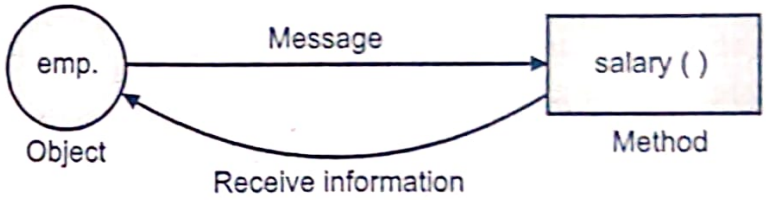


Fig. 1.4: Message Passing

1.3 FEATURES OF OOP [S-18]

- OOP treats data as 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 decomposition, (breaking into small modules) of a problem into number of entities called objects and then builds data and functions around these objects.
- The organization of data and functions in OOP is shown in Fig. 1.5. The data of an object can be accessed only by the functions associated with that object. However, functions of one object can access the functions of other objects.

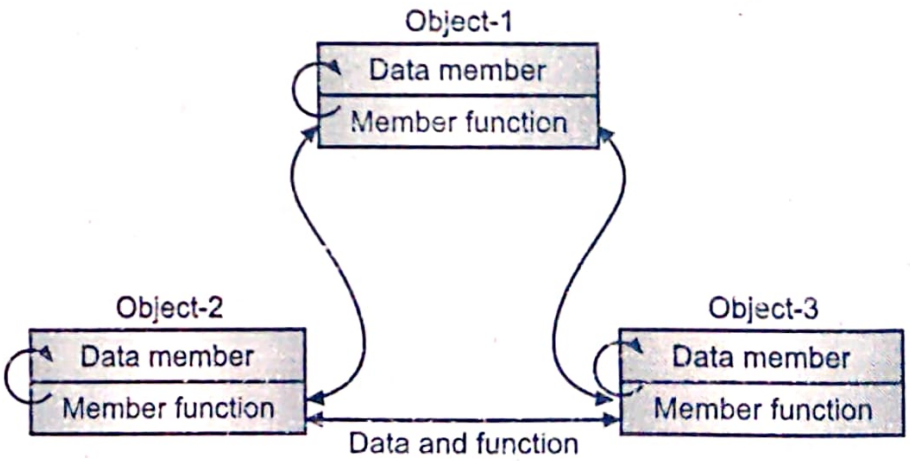


Fig. 1.5: Data and Function Organization in OOP

- The organization of data and function is payroll system is shown in Fig. 1.6.
- In Fig. 1.6, data and function organization is shown with respect to payroll system. We can consider three objects like employee, pay slip and other details.

- All of them have some data members and member functions which act on that data. The member functions of one object can communicate with the member function of another object.

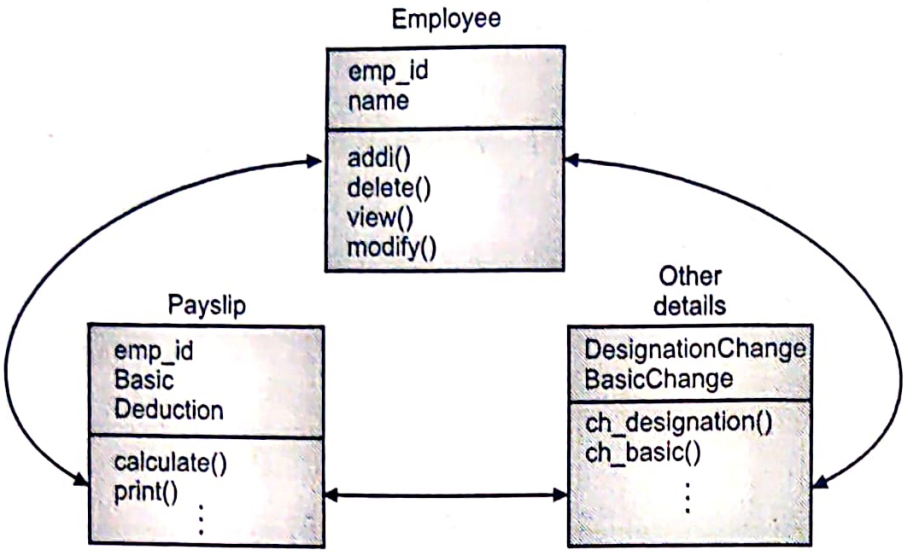


Fig. 1.6: Organization of Data and Member Function in Payroll

- Fig. 1.7 shows the important features of OOPs. It also shows various features offered by C++ programming language. From the Fig. 1.7 we can recognize that the feature persistence is not offered by C++ programming language.

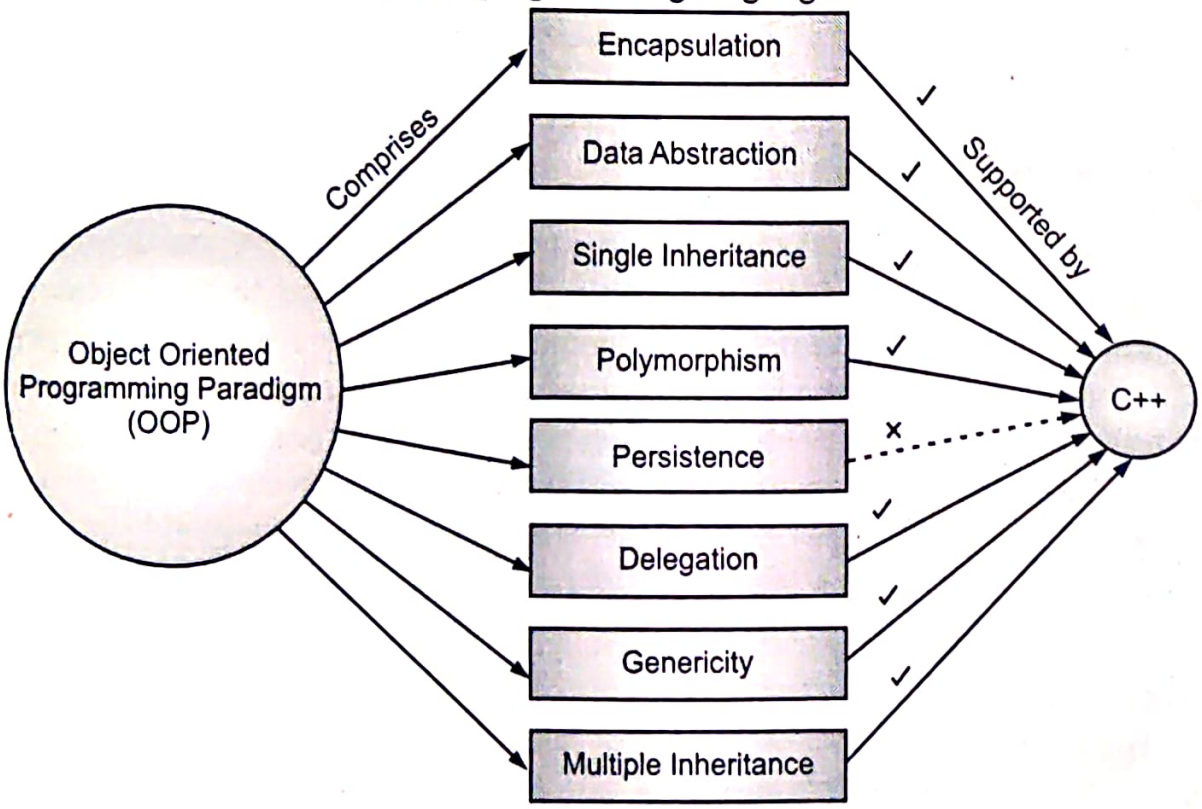


Fig. 1.7: Features of OOP's

1. Data Encapsulation:

- The wrapping up of data and functions into a single unit i.e. class is called as encapsulation. Data encapsulation is also known as data hiding.
- 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 objects data and the program. They are bound together.

- They are safe from external interference and misuse. This represents encapsulation (or seal). This feature is not available in any conventional procedure-oriented programming language.

  For example: The atmosphere encapsulates the earth, skin encapsulates the internal part of a body.

2. Data Abstraction:

- In object oriented programming, each object will have external interfaces through which it can be used. There is no need to look into its inner details.
- So the user needs to know the external interfaces only, to make use of an object. The internal details of the objects are hidden which makes them abstract. This technique or feature of hiding internal details in an object is known as Data Abstraction.
- Abstraction is the selective examination of certain aspects of a problem. The goal of abstraction is to isolate those aspects that are important for some purpose and suppress those aspects that are unimportant.
- Many different abstractions of the same thing are possible, depending on the purpose for which they are made. In other words, we can say that abstraction as the act of representing essential features without including the background details or explanation. Data abstraction is the process of defining a data type, often called Abstract Data Type (ADT), together with the principle of data hiding.
- The definition of an ADT involves specifying the internal representation of the ADTs data as well as the functions to be used by others to manipulate the ADT.
- In OOP, classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost and functions to operate these attributes. Classes are also known as ADTs.

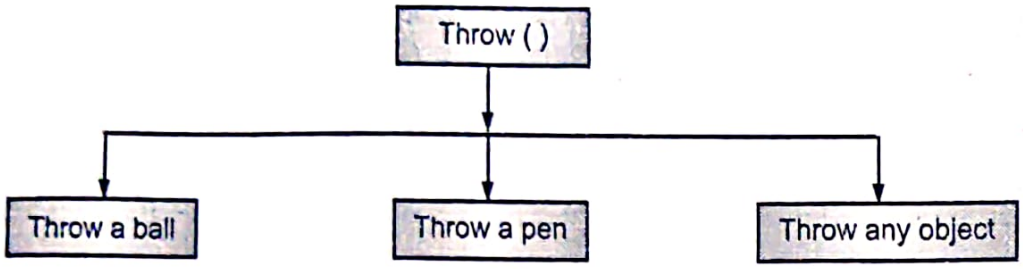
3. Inheritance:

- The notation of defining a new object in terms of an old one is an integral part of OOP. The term inheritance is used for this concept. Inheritance means one class of objects inherits the data and behaviour from another class.
- Inheritance imposes a hierarchical relationship among classes in which a child class inherits from its parents. The parent class is known as the base class and the child is the derived class. The base class is also known as *Superclass* and the derived class is known as *Subclass*.
- Attributes and operations common to a group of subclasses are attached to the superclass and shared by each subclass. Each subclass is said to inherit the features of its superclass. A child inherits the property of his father. He can acquire new properties or modify the inherited one.
- Same way, a new class can derive its properties from another existing class. The derived class inherits all the properties of the base (old) class. This allows the extension and reuse of existing code. This also enables the creation of hierarchy of classes which simulate the class and subclass concept of real world.

4. Polymorphism:

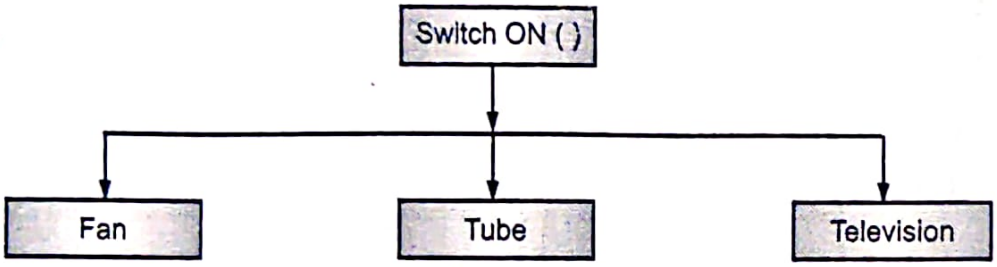
- It is very useful concept in OOP's. In simple terms it means one name, many duties. It provides common interfaces to carry out similar tasks.

- In other words, we can say that a common interface is created for accessing related objects. In C++, it is achieved by function overloading operator overloading and dynamic binding.
  For example:
  (i)



(a)

(ii)



(b)

Fig. 1.8

- Basically, polymorphism means the quality of having more than one form. In OOP, polymorphism refers to the fact that a single operation can have different behavior in different objects.
- In other words, different objects react differently to the same message. For example, consider the operation of addition. For two numbers, the addition should generate the sum.
- The operation of addition is expressed by a single operator +. You can use the expression x + y to denote the sum of x and y for many different types of x and y integers, floating point numbers and complex numbers and even the concatenation of two strings.
- Similarly, suppose a number of geometrical shapes all respond to the message draw. Each object reacts to this message by displaying its shape on a display screen. Obviously, the actual mechanism for displaying the object differs from one shape to another, but all shapes perform this task in response to the same message.

5. Message Passing:

- It is the process of invoking an operation on an object. In response to a message, the corresponding method (function) is executed in the object.
- OOP consists of objects and classes. It consists of set of objects that communicate with each other. It is very necessary to understand properly the classes, objects and also the message passing i.e. communication between objects.
- Objects communicate with one another by sending and receiving information. A message for an object is a request for execution of a procedure, and therefore will invoke a function (procedure) in the receiving object that generates the desired result.
- Message passing involves specifying the name of the object, the name of the function (message) and the information to be sent.

6. **Extensibility:**

- It is a feature, which allows the extension of the functionality of the existing software components. In C++, this is achieved through abstract classes and inheritance.

7. **Persistence:**

- The phenomenon where the object (data) outlives the program execution time and exists between executions of a program is known as Persistence.
- In C++, this is not supported. However, the user can build it explicitly using file streams in a program.

8. **Delegation:**

- It is an alternative to class inheritance. It is a way of making object composition as powerful as inheritance. In delegation, two objects are involved in handling a request.
- In C++, this approach takes a view that an object can be a collection of many objects and the relationship is called has-a relationship or containership.

9. **Genericity:**

- It is a technique for defining software components that have more than one interpretation depending on the data type of parameters.
- In C++, genericity is realised through function templates and class templates.

1.4 ADVANTAGES AND APPLICATIONS OF OOP

1.4.1 Advantages of OOP

1. Data abstraction concept consists of focusing on the essential, inherent aspects of an entity and ignoring its accidental properties. The use of abstraction preserves the freedom to make decisions as long as possible.
2. By the use of property inheritance, we can eliminate redundant code and extend the use of existing class.
3. The property of data encapsulation helps to programmer to build secure programs that cannot be invaded by code in other parts of the program.
4. We can have multiple instances of an object to co-exist without any interference.
5. We can map objects in the problem domain to the object in the program.
6. We can modularise the program by dividing it into classes and objects.
7. The data centered design approach enables to capture more details of a model in implementable form.
8. Object oriented programming supports reusability. Reusable software reduces design, coding and testing cost by amortizing effort over several design. Reducing the amount of code also simplifies understanding.
9. In OOP, complex softwares can be easily managed.
10. OOP saves development time and gives higher productivity.
11. In OOP, small projects (programs) can be easily upgraded to larger one.
12. Message passing in OOP provides the interface with external systems.

1.4.2 Applications of OOP

- 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 areas for application of OOP includes:
  1. Real time systems, for example, oil exploration plant.
  2. CAM/CAD systems.
  3. Object oriented databases.
  4. Artificial Intelligence (AI) and expert systems, for example, Disease knowledge base.
  5. Hypertext, hypermedia and expertext, for example, web page designing.
  6. Decision support and office automation systems.
  7. Neural networks and Parallel programming.
  8. Simulation and Modeling.

1.5 INTRODUCTION TO C++

- C++ is a hybrid language, in which some entities are objects and some are not.
- C++ is an extension of C language, implemented not only to add object-oriented capabilities but also to readdress some of the weaknesses of C language.
- Many added features are orthogonal to OOP, such as inline expansion of subroutines, overloading of functions and function prototypes.
- C++ is a strongly typed language developed by **Bjarne Stroustrup** at AT and T's Bell Laboratories. It was originally implemented as a preprocessor that translates C++ into standard C.
- C++ contains good facilities for specifying access to attributes and operations of a class. Access may be permitted by methods of any class (public), restricted to methods of subclasses of the class (protected), or restricts to direct methods of the class (private). In addition, instant access can be given to a particular class or function using the friend declaration.
- C++ also supports overloaded operators i.e. several methods that share the same name but whose arguments vary in number of type.
- C++ supports several memory allocation strategies for objects – statically allocated by the compiler, stack based and allocated at run-time from a heap.
- The programmer must avoid mixing objects of different memory types or dangling references that may cause run-time failures.
- Each class can have several constructor and conversion functions, which initialize new objects and convert between types for assignment and argument passing.
- To conclude, C++ is a complex, malleable language characterized by a concern for the early detection of errors, various implementation choices and run-time efficiency at the expense of some design flexibility and simplicity.
- C++ systems generally consist of several parts i.e. a program development environment, the language and the C++ standard library.

1.5.1 Features of C++

- C++ supports all features of structure programming and object oriented programming, they are listed below:
  1. C++ provides overloading of functions and operators.
  2. Exception handing in C++ is done by the try, catch and throw keywords.
  3. C++ provides static methods, friends, constructors and destructors for the class object.
  4. It focuses on function template and class template for handling parameterized data types.
  5. C++ gives the simplest and easiest way to handle encapsulation and data hiding with the help of powerful keywords such as private, class, public and protected and so on.
  6. Inheritance concept in C++, one of the most powerful and important design concept is supported with single inheritance and multiple inheritances of base class and derived class.
  7. In C++, the polymorphism is done through virtual functions, virtual base classes and virtual destructors give the late binding of the compiler.

1.5.2 Applications of C++

- C++ is a flexible language, capable for handling very large programs.
- C++ is suitable for almost any programming task as well as development of databases, editors, communication systems, compiler and any complex real life application systems.
- C++ language is able to map the real world problem properly, effectively and efficiently the C part of C++ language gives the language the ability to get close to the machine level details.
- In C++ language programs are easily expandable and maintainable when a new feature needs to be implemented. It is very easy to add to the existing structure of an objects.
- C++ language allows user to create hierarchy related object, he/she can built special object oriented libraries which can be used later by many programmers.

1.6 INPUT AND OUTPUT OPERATOR IN C++

- C++ support a set of functions for performing input and output operations. The new feature of C++ is called **streams** which is used to handle I/O operations.
- C++ streams are of two types:
  (i) Input stream, and
  (ii) Output stream.
- Stream refers to the flow of data from a particular source to a specified destination.
- There are classes representing each stream **istream** is the class representing the input stream (istream) and **ostream** is the class representing the output stream.
- To achieve the console Input/Output operations, we use the objects of these stream classes.

(i) Input stream:

- The input stream is handling all input operations or reading operations with input devices like keyboard, disk etc. **"cin"** is the predefined object of the istream class. It is used for all console input operations.
- Operator **>>**, (or the right shift operator in C) is called the **extraction** operator, it gets the value from the stream object **cin** on its left and places it in the variable on its right.

Syntax:

```
cin>>variable;
```

- The use of cin is shown in Fig. 1.9.

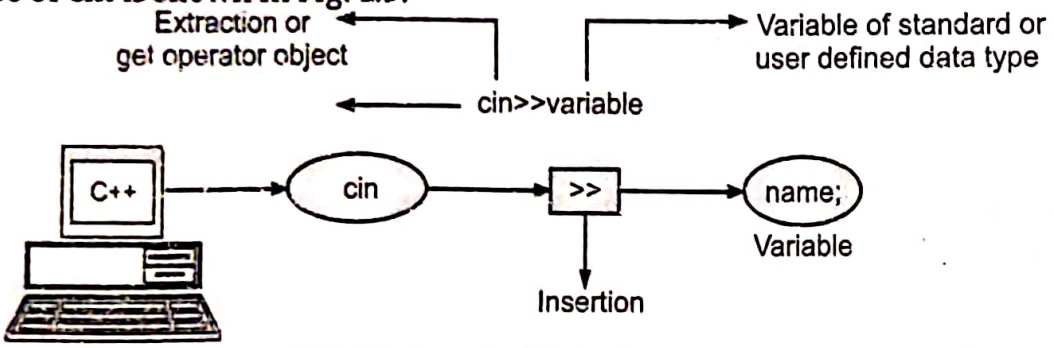


Fig. 1.9: Input with cin Operator

Examples:

1. `int a;`
   `cin>>a;`
2. `float salary;`
   `cin>>salary;`
3. `double area;`
   `cin>>area;`
4. `char name[15];`
   `cin>>name;`

- We can input more than one item using cin object. Such input operations are called cascaded input operations.
- cin will read all the items from left to right.

Syntax for more than one variable as:

```
cin>>var1>>var2>>var3..... >>varn;
```

Examples:

1. `cin>>x>>y;` `//x read first then y`
2. `cin>>name>>address;`
3. `cin>>roll>>name>>marks;`

(ii) Output stream:

- The output stream handles the write operations on output devices like screen, disk etc. **cout** is the predefined object of the **stream** class and is used for all console output operations.
- Operator **<<**, (or the left shift operator in C) is called **insertion** operator. It directs the contents of the variable on its right to the object (cout) on its left.

Syntax:

```
cout<<variable;
```

- The use of cout is shown in Fig. 1.10.

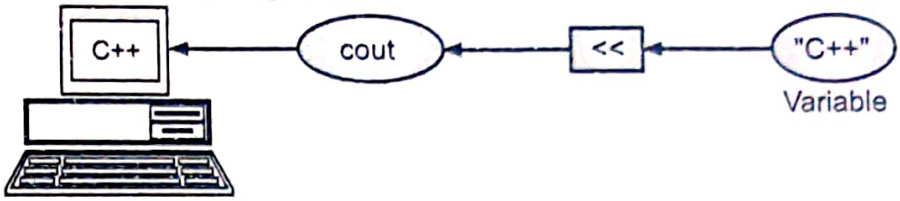


Fig. 1.10: Output with cout Operator

Examples:

1. `int a;`
   `cout<<a;`
2. `float salary;`
   `cout<<salary;`
3. `char name[15];`
   `cout<<name;`

- We can display more than one outputs called cascaded outputs.

Syntax:

```
cout<<var1<<var2<<.....<<varn;
```

Examples:

1. `cout<<"x="<<x;`
2. `cout<<roll<<"  "<<name;`
3. `cout<<roll<<"\t"<<name;`
4. `cout<<name<<endl;`

1.7 A SIMPLE C++ PROGRAM [W-22]

- A computer program is a sequence of instructions that tell the computer what to do.
- A typical C++ program would contain four sections:
  1. Include files
  2. Class declaration
  3. Member function definition
  4. Main function
- Fig. 1.11 shows structure of a C++ program.

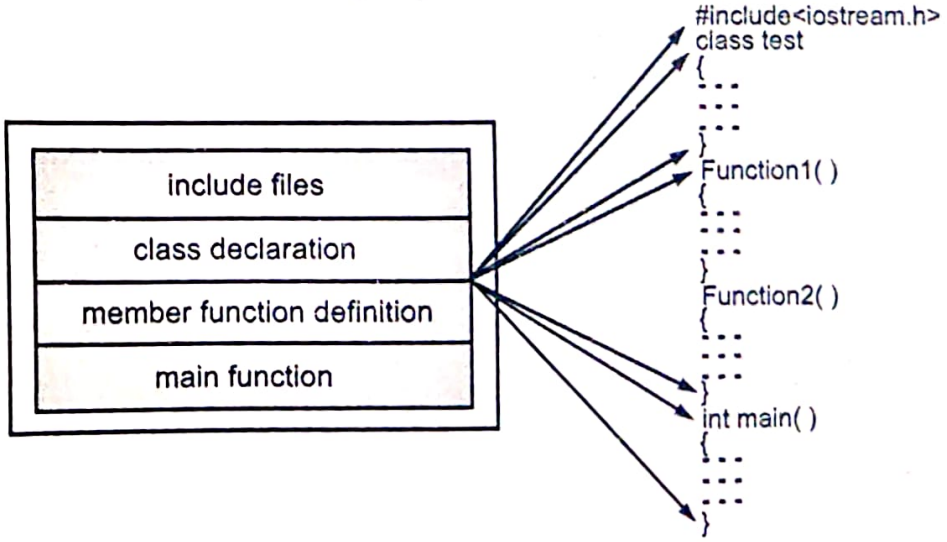


Fig. 1.11: Structure of C++ Program

- C++ program is a collection of functions.

For example:

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5.     cout << "Hello World!" <<  endl;
6.     return 0;
7. }
```

Output:

```
Hello World!
```

- Line 1 `#include<iostream>` is a special type of statement called a preprocessor directive and it start from #. Preprocessor directives tell the compiler to perform a special task. In this case, we are telling the compiler that we would like to use the iostream library. The iostream library contains code that tells the compiler what cout and endl do. In other words, we need to include the iostream library in order to write to the screen.
- Line 2 `using namespace std;` defines a scope for the identifiers that are used in the program. As you learned in the explanation for line 1, cout and endl live inside the iostream library. However, within iostream, they live inside a special compartment named std (short for standard). This using statement tells the compiler to look inside a compartment named std if it cannot find cout or endl defined anywhere else. In other words, this statement is also necessary so the compiler can find cout and endl, which we use on line 5.
- Line 3 `int main()` declares the main() function which is the point from where all C++ program begin their execution. Every program must have a main() function.
- Lines 4 ({ is the opening curly brace marks the start of a code block) and 7 (} is the closing curly brace, marks the end of function main()) tell the compiler which lines are part of the main function. Everything between the opening curly brace on line 4 and the closing curly brace on line 7 is considered part of the main() function.
- Line 5 `cout <<"Hello World!" << endl;` is our output statement. cout is a special object that represents the console/screen. The << symbol is an operator (much like + is an operator) called the output operator. cout understands that anything sent to it via the << operator should be printed on the screen. endl is a special symbol that moves the cursor to the next line.
- Line 6 `return 0;` is a new type of statement, called a return statement. When an executable program finishes running, it sends a value to the operating system that

indicates whether it was run successfully or not. The return value of main() is used for this purpose. This particular return statement returns the value of 0 to the operating system, which means "everything went okay!".

1. Comments:

- Comments are parts of the source code disregarded by the compiler. They simply do nothing.
- Their purpose is only to allow the programmer to insert notes or descriptions embedded within the source code.
- C++ supports two ways to insert comments:
  (i) // line comment, and
  (ii) /* */ block comment.
- The first of them, known as line comment, discards everything from where the pair of slash signs (//) is found up to the end of that same line.
- The second one, known as block comment, discards everything between the /* characters and the first appearance of the */ characters, with the possibility of including more than one line.

Program 1.1: Program using comments.

```
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello World!" << endl;          //Prints Hello World!
   cout << "WelCome to C++ !" << endl;      //Prints Welcome to C++!
   return 0;
}
```

Output:

```
Hello World!
Welcome to C++ !
```

- If you include comments within the source code of your programs without using the comment characters combinations //, /* */, the compiler will take them as if they were C++ expressions, most likely causing one or several error messages when you compile it.

2. Header Files:

- Each standard library has a corresponding header file containing the function prototypes for all the functions in that library and definitions of various data types and constants needed by those functions.
- Some common C++ header files that may be included in standard C++ program, are given below:

Table 1.2: Standard Library Header Files

Standard library header files	Explanation
<assert.h>	Contains macros and information for adding diagnostics that aid program debugging.
<ctype.h>	Contains function prototypes for functions that test characters for certain properties and that can be used to convert lowercase letters to uppercase letters.
<float.h>	Contains the floating point size limits of one program.
<limits.h>	Contains integral size limits of the system.
<math.h>	Contains function prototype for math library function.
<stdio.h>	Contains function prototype for the standard Input/Output library functions and information used by them.
<stdlib.h>	Contains function prototype for conversions of numbers to text, text to numbers, memory allocation, random numbers and various other utility functions.
<string.h>	Contains function prototype for C style string processing function.
<time.h>	Contains function prototype for manipulating time and date.
<iostream.h>	Contains function prototype for standard input and output functions.
<iomanip.h>	Contains function prototype for the stream manipulators that enable formatting of streams of data.
<fstream.h>	Contains function prototype for functions that perform input from files on disk and output to files on disk.

- The programmer can create custom header files. Programmer defined files should end in .h, which can be included by using the #include preprocessor directive.
- For example, the header file `square.h` can be included in our program by the directive, `#include "square.h"` at the top of the program.

1.8 C VS C++

- C++, as the name suggests is a superset of C. As a matter of fact, C++ can run most of C code while C cannot run C++ code.
- Below are the major differences between C and C++.
  1. **C follows the procedural programming paradigm while C++ is a multi-paradigm language (procedural as well as object oriented):** In case of C, importance is given to the steps or procedure of the program while C++ focuses on the data rather than the process. Also, it is easier to implement/edit the code in case of C++ for the same reason.
  2. **In case of C, the data is not secured while the data is secured (hidden) in C++:** This difference is due to specific OOP features like Data Hiding which are not present in C.

3. **C is a low-level language while C++ is a middle-level language:** C is regarded as a low-level language (difficult interpretation and less user friendly) while C++ has features of both low-level (concentration on what's going on in the machine hardware) and high-level languages (concentration on the program itself) and hence is regarded as a middle-level language.
4. **C uses the top-down approach while C++ uses the bottom-up approach:** In case of C, the program is formulated step by step, each step is processed into detail while in C++, the base elements are first formulated which then are linked together to give rise to larger systems.
5. **C is function-driven language C++ is object-driven language:** Functions are the building blocks of a C program while objects are building blocks of a C++ program.
6. **C++ supports function overloading while C does not:** Overloading means two functions having the same name in the same program. This can be done only in C++ with the help of Polymorphism.
7. **We can use functions inside structures in C++ but not in C:** In case of C++, functions can be used inside a structure while structures cannot contain functions in C.
8. **The namespace feature in C++ is absent in case of C:** C++ uses namespace which avoid name collisions.
9. **The standard input and output functions differ in the two languages:** C uses scanf and printf while C++ uses cin (>>) and cout (<<) as their respective input and output functions.
10. **C++ allows the use of reference variables while C does not:** Reference variables allow two variable names to point to the same memory location. We cannot use these variables in C programming.
11. **C++ supports exception handling while C does not:** C does not support it "formally" but it can always be implemented by other methods. Though you don't have the framework to throw and catch exceptions as in C++.
12. **C is a structure oriented language** where as **C++ is an object oriented language** where the coding is done by using the user defined objects.
13. C files are stored with an extension **'.c'** whereas C++ files are stored with **'.cpp'** extension.

Summary

- OOP stands for Object Oriented Programming. OOP is an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand.
- Examples of OOP are C++, Java, Eiffel, Smalltalk, VB.Net, C#.Net etc.
- Basic object-oriented concepts are objects, classes, encapsulation, inheritance, and polymorphism.
- Object is the basic unit of object-oriented programming. In OOP, a problem is considered as a collection of a number of entities called objects. Objects are instances of classes.

- In OOP a class gives the blueprint for a set of similar objects. A class is a user defined type consisting of data members and member functions.
- Data abstraction increases the power of programming language by creating user defined data types. Data abstraction also represents the needed information in the program without presenting the details.
- In OOP insulation of data from direct access by the program is called data hiding.
- Inheritance is the process of forming a new class from an existing class or base class. The base class is also known as parent class or super class. The new class that is formed is called derived class. Derived class is also known as a child class or sub class.
- Data encapsulation combines data and functions into a single unit called class. When using data encapsulation, data is not accessed directly; it is only accessible through the functions present inside the class.
- Data encapsulation enables the important concept of data hiding possible. Inheritance helps in reducing the overall code size of the program.
- In OOP polymorphism means one name, multiple forms. It allows us to have more than one function with the same name in a program.
- Persistence feature of OOP allows the object (data) outlives the program execution time and survives several executions of a program.
- Extensibility feature of OOP allows extension of the functionality of the existing software components.
- In OOP, dynamic binding means that the code associated with a given procedure is not known until the time of the call at run-time.
- Binding refers to the linking of a procedure call to the code to be executed in response to the call. When memory is allocated at run-time it is known as dynamic binding.
- In OOP, message passing involves specifying the name of the object, the name of the function (message) and the information to be sent.
- Object oriented approach has many benefits like Ease in software design, Ease in software maintenance and Reusability of software.
- Application of OOP has gained importance in almost all areas of computing including real-time business systems.

Check Your Understanding

1. The word "oriented" means showing .....
   (a) ambiguity (b) functionality
   (c) methodology (d) none
2. OOP decompose the problem into smaller unit known as .....
   (a) object (b) class
   (c) function (d) none
3. Object-oriented programming language gives emphasis on .....
   (a) flowchart (b) algorithm
   (c) data (d) none

4. Object-oriented programming language follows the ..... approach.
   (a) top-down (b) bottom-up
   (c) top-bottom (d) none
5. The function that operate on the data of an object are tied together in the data structure, this concept is known as .....
   (a) Polymorphism (b) inheritance
   (c) encapsulation (d) none
6. ..... allows creation of new classes from old one.
   (a) Polymorphism (b) inheritance
   (c) class (d) none
7. ..... are basic run-time entities.
   (a) data (b) classes
   (c) objects (d) none
8. The object with the attributes and methods are grouped together to form a .....
   (a) data (b) class
   (c) object (d) none
9. An object is an ..... of a class.
   (a) abstraction (b) operation
   (c) instance (d) none
10. The code redundancy can be eliminated through .....
    (a) polymorphism (b) inheritance
    (c) class (d) none
11. C++ was developed by .....
    (a) Stroustrup (b) Denis Richie
    (c) Pascal (d) none
12. The statement which starts with the symbol // is treated as ..... comment.
    (a) one (single) line (b) multiline
    (c) miniline (d) none

Answers

1. (b)	2. (a)	3. (c)	4. (b)	5. (c)	6. (b)	7. (c)	8. (b)	9. (c)	10. (b)
				11. (a)	12. (a)				

Practice Questions

Q.I Answer the following questions in short:

1. What is meant by OOP?
2. List out different concepts in OOP?
3. What is Class?
4. Define the term Polymorphism in detail.
5. What is C++?

Q.II Answer the following questions:

1. Explain the term Data Abstraction in detail.
2. What are the benefits of OOP?
3. Write short note on: Message Passing.
4. What are the Applications of OOP?
5. Enlist various Applications of C++.
6. Explain the following terms:
   (a) Object
   (b) Class.
7. Write a C++ program of print "Welcome to BCA-IV".
8. Compare C and C++.
9. List various features of C++.
10. State advantages and disadvantages of C++.

Q.III Define the term:

1. Encapsulation
2. Data abstraction
3. Polymorphism
4. Inheritance
5. Object
6. Classes
7. Dynamic binding.

Previous Exam Questions

April 2018

1. List any four features of OOPs. **[2 M]**

Ans. Refer to Section 1.3.

October 2018

1. What is encapsulation? **[2 M]**

Ans. Refer to Section 1.2.4.

2. Differentiate between object oriented programming and procedure oriented programming. **[4 M]**

Ans. Refer to Section 1.1.2.

April 2019

1. What is data abstraction? **[2 M]**

Ans. Refer to Section 1.2.3.