Lesson 1: Introduction To Object-Oriented Programming

Objectives

After reading this chapter, you will understand:

- The procedure oriented programming concepts and object oriented paradigm.
- The basic concepts of object oriented programming.
- Benefits of object oriented programming.
- Applications of object oriented programming.
- Structure of a C++ program, writing a sample C++ program, compiling and running a program.
- The different types of errors in programming.

Structure Of the Lesson

1.1 Introduction

C++ is an object oriented programming language. It was initially named as C with classes. However in 1983, it was renamed as C++. C++ was developed by Bjarne Stroustrup at AT & T Bell laboratories, New Jersey, U.S.A. It is an enhancement of the C programming language with the major addition of class construct feature of Simula67. It was built upon C and hence all standard C features are also available in C++. Thus C++ is the superset of C. Almost all C programs are also C++ programs.

The three important facilities that C++ have are C with classes, function overloading and operator overloading. These features help to create abstract datatype, inheritance from existing datatype and polymorphism. C++ allows the programmer to build programs with clarity, extensibility and ease of maintenance.

1.1.1 Procedure Oriented Programming

Procedure oriented programming is viewed as a sequence of things to be done, such as reading, calculating, printing etc. A number of functions are written to accomplish these tasks. The primary importance is given to functions and little is given to data that are being used by various functions.

Data is placed as global, so that they may be accessed by all the functions. Each function can have its own data. Global data can be used by any function so that it is difficult to identify what data is used and by which function.

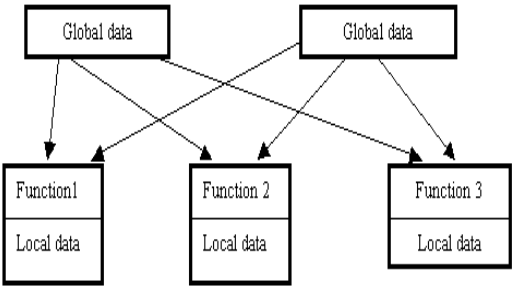
Procedure Oriented programming does not model real world problems very well.

Some Important Characteristics Are:

- Importance is given in doing the algorithms.
- Large programs are divided into smaller programs known as functions.
- Most of the functions share global data.
- Data move over the system from function to function freely.
- Functions transforms the data from one form to another.
- Employs top-down approach in program design.

1.1.2 Object Oriented Programming

Object oriented programming is an approach that provides a way of modularizing the programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules in demand.

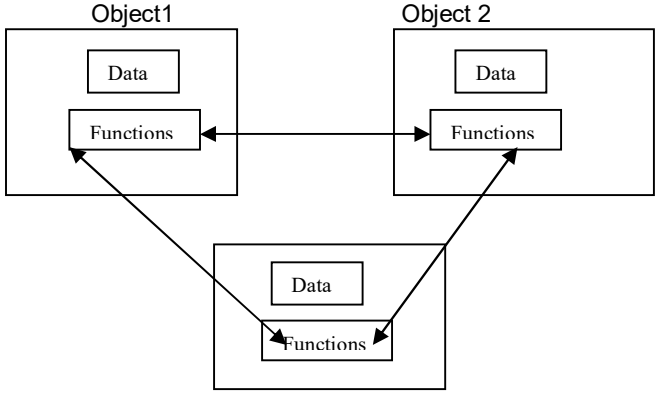


OOPS treats data as a critical element in the program development and does not allow it to freely flow around the system.

It ties data more closely to the function that operates on it and protects it from accidental modifications from outside functions. OOP allows us to decompose the problem into number of entities called objects and build data and function around them. Data of an object can be accessed only by functions associated with that object. Functions of one object can access the functions of another object.

Important Features:

- Importance is given to data then the functions.
- Programs are divided into objects.
- Data Structures characterize the objects.
- Functions that operate on the data of the object are tied together in the data structure.
- Data is hidden and cannot be accessed by the external functions.
- Objects may communicate with each other through functions.
- New data and functions can be easily added.
- Follows bottom-up approach is used in the program design.



Object3

1.2 Concepts Of OOPS

The major Concepts of Object Oriented Programming are:

1. Class
2. Object
3. Abstraction
4. Encapsulation
5. Data Hiding
6. Inheritance
7. Reusability
8. Polymorphism
9. Virtual Functions
10. Message passing

Class: Class is an abstract data type (user defined data type) that contains member variables and member functions that operate on data. It starts with the keyword class. A class denotes a group of similar objects.

e.g.:
```
class employee
{
  int empno;
  char name[25],desg[25];
  float sal;
public:
  void getdata ();
  void putdata ();
};
```

Object: An object is an instance of a class. It is a variable that represents data as well as functions required for operating on the data. They interact with private data and functions through public functions.

e.g.: employee e1, e2;

In the above example employee is the class name and e1 and e2 are objects of that class.

Abstraction: Abstraction refers to the process of concentrating on the most essential features and ignoring the details. There are two types of abstraction

i) Procedural Abstraction
ii) Data Abstraction

Procedural Abstraction: Procedural abstraction refers to the process of using user-defined functions or library functions to perform a certain task, without knowing the inner details. The function should be treated as a black box. The details of the body of the function are hidden from the user.

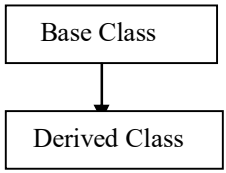
Data Abstraction: Data Abstraction refers to the process of formation of user defined data type from different predefined data types.
e.g: structure, class.

Encapsulation: Encapsulation is the process of combining data members and member functions into a single unit as a class in order to hide the internal operations of the class and to support abstraction.

Data Hiding: All the data in a class can be restricted from using it by giving some access levels (visibility modes). The three access levels are private, public, protected.

Private data and functions are available to the public functions only. They cannot be accessed by the other part of the program. This process of hiding private data and functions from the other part of the program is called as data hiding.

Inheritance: Inheritance is the process of acquiring (getting) the properties of some other class. The class whose properties are being inherited is called as base class and the class which is getting the properties is called as derived class.



Reusability : Using the already existing code is called as reusability. This is mostly used in inheritance. The already existing code is inherited to the new class. It saves a lot of time and effort. It also reduces the size of the program.

Polymorphism: Polymorphism means the ability to take many forms. Polymorphism allows to take different implementations for same name.

poly → many
morphism → forms

There are two types of polymorphism, Compile time polymorphism and run time polymorphism. In Compile time polymorphism binding is done at compile time and in runtime polymorphism binding is done at runtime.

e.g.: Function overloading, operator overloading

Function Overloading: Function overloading is a part of polymorphism. Same function name having different implementations with different number and type of arguments.

Operator Overloading: Operator overloading is a part of polymorphism. Same operator can have different implementations with different data types.

Virtual Functions: Virtual functions are special type of functions which are defined in the base class and are redefined in the derived class. When virtual function is called with a base pointer and derived object then the derived class function will be called. A function can be defined as virtual by placing the keyword virtual for the member function.

Message Passing: An object-oriented program contains a set of objects that communicate with one another. The process of object oriented programming contains the basic steps:

1. Creating classes
2. Creating objects
3. Communication among objects

This communication is done with the help of functions (i.e., passing objects to functions)

1.2.1 Benefits Of OOPS

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

- Programs can be built from the standard working modules that communicate with one another , rather than writing the code from scratch. This leads to saving of development time and higher productivity.

- Principle of data hiding helps the programmer to build secured programs. It is possible to have multiple instances of objects to co-exist without any inheritance.

- Easy to partition the work in project based objects.

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

- Software complexity can be easily managed.

- Message passing techniques for communication makes the interface descriptions with external systems much simpler.

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

1.2.2 Applications of OOPS

The promising areas for application of OOP includes:

- Real-time systems
- Simulation and modeling
- Object-oriented databases
- Hypertext,hypermedia and experttext
- AI and expertsystems
- Neural networks and parallel programming
- Decision support and Automation system
- CIM/CAM/CAD systems

1.3 The Software Life Cycle

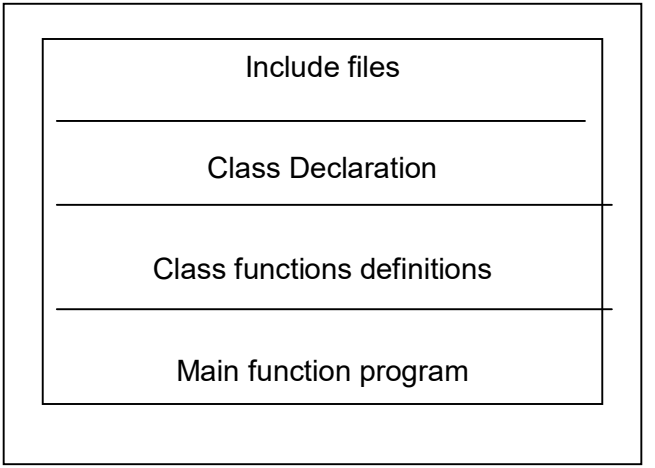
The software development process is divided into six phases known as software life cycle. The six phases of this life cycle are:

- Analysis and specification of the task (problem definition)
- Design of the software (algorithm design)
- Implementation (coding)
- Testing
- Maintenance and evolution of the system
- Obsolescence

1.4 Structure of C++ Program

C++ program contains 4 sections. These may be placed in separate code files and then compiled independently or jointly. A program is commonly organized into 3 separate files. The class declarations are placed in a header file and the definitions of member functions go into another file.

This helps the programmer to separate the abstract specification of the interface (class definition) from the implementation details (member function definition). The main program is placed in a third file which includes the previous two files as well as any other files required.



Structure of C++ program

1.4.1 A Sample C++ Program

```
//sum of two integers
#include <iostream.h>     //include header file
int  main()
{
        int x,y,sum;
        cout <<"Enter any 2 numbers: ";
        cin>>x>>y;
        sum = x + y;
        cout << "The given numbers are ";
        cout<<x;
        cout<<" and ";
        cout<<y;
        cout<<"\n";
        cout<<"Their sum is "<<sum<<"\n";
        return 0;
}               //End of example
```

Output:

```
Enter any 2 numbers: 4 5
The given numbers are        4 and 5
Their sum is 9
```

In order to make a program understandable, some explanatory notes is included at key places in the program. Such notes are called comments. In C++ the symbols // are used to indicate the start of the comments. The comment starts with a // and terminate at the end of the line. A comment may start any where in the line, and whatever follows till the end of the line is ignored. // is a single line comment.

Example:

```
//This is a
//sample C++
//program
```

The C comment symbols /*-------*/ is also valid and are suitable for multiline comments. Either or both of the styles can be used in the programs.

Example:

```
/*This is a       sample C++ program*/
```

The program begins with the line:

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

This is called include directive. It tells the compiler where to find information about certain items that are used in your program. iostream is the name of the library that contains the definitions of the routines that handle input from the keyboard and output to the screen. iostream.h is the file that contains the information about the library.

Directives begin with the symbol # at the very start of the line and no space is included between # and include. A C++ program is a collection of functions. The above example contains one function, main(). The program starts with

```
int main()
{
and ends with
return 0;
}
```

As the return type of the function is integer, 0 is returned here.The lines between the beginning and ending {} are the heart of the program.

```
int x,y,sum;
```

This line is called variable declaration. The variable declaration tells the computer that x,y and sum are the name of the three variables used in the program. int word tells the computer the numbers named by these variables are integers.

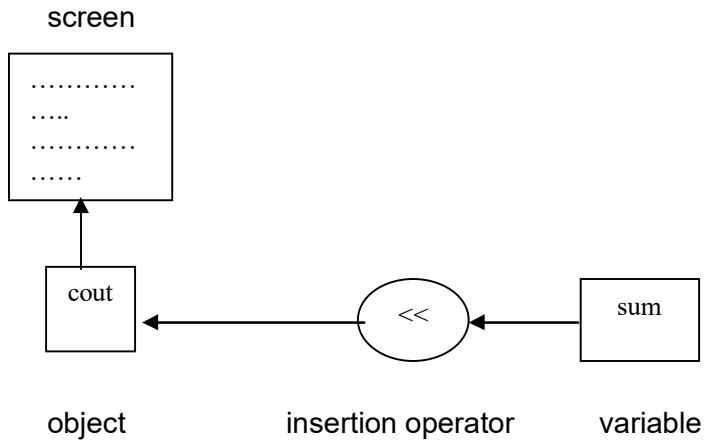
The remaining lines are the instructions that tell the computer to do the corresponding work. These instructions are called executable statements or statements. Every statement should end with a semicolon.

Most of the statements begin with the word cout or cin. These statements are the input and output statements. The << and >> arrows are the operators which tell the direction in which the data is moving.

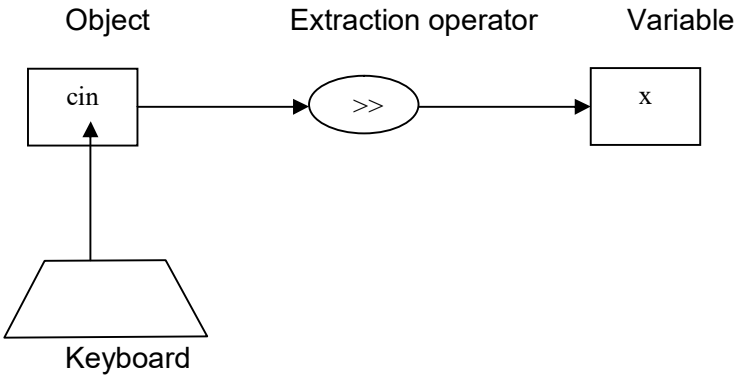
The operator << is called the insertion operator or put to operator. It inserts (or sends) the contents of the variable on its right to the objects on its left. Cout is a predefined object that represents the standard output stream in C++.

Here the standard o/p stream represents the screen. << operator can be overloaded.

Ex: cout<<"Enter two numbers";
cout<<sum;



The operator >> is known as extraction or get from operator. It extracts or takes the value from the keyboard and assigns it to the variable on its right. >> operator can also be overloaded.
e.g.: cin >> x ;



Cascading of I/O operators: The multiple use of << in one statement is called cascading. This is known as cascading of output operator.

e.g.: cout<<"Their sum is"<<sum<<"\n";

This statement sends the string "Their sum is" to cout and then sends the value of sum, then the newline.

Similarly, >> operator can be cascaded. This is known as cascading of input operator.

```
e.g.:   cin>>x>>y;
        sum = x+y;
        cout<<
```

This is the computational statement. The values of x and y are summed up with + operator and the value is stored in the variable sum.

```
        cout<<"Their sum is "<<sum<<"\n";
```

"\n" contained at the end of the output statement tells the computer to start a newline after writing the text.

1.4.2 Compiling and Running

C++ program is typed in using a text editor. There are different text editors. Turbo C++ provides a built-in editor and a menu bar including the options such as File, Edit, Compile and Run. The source file is created and saved under the File Option and can be edited under Edit option. The program is compiled under Compile Option and Run using Run option.

Compilation of the program will produce a machine-language translation of the source code, called the object code. The object code must be linked (combined) with the object code for routines (input and output routines) that are already written. Run option executes the program. If there are no errors in the program, then compiling, linking and running will go smoothly. However, errors may occur which has to be rectified and executed.

1.4.3 Testing and Debugging

A mistake in the program is usually called a bug, and the process of eliminating bugs is called debugging. There are three kinds of programming errors. They are

- Syntax errors
- Runtime errors and
- Logical errors

The errors that are caused due to the violation of syntax (grammar rules) of the programming language are called syntax errors. E.g.: Omitting semicolon at the end of the statement. These errors can be found during compilation.

There are certain kinds of errors that the computer system can detect only when the program is run. These are run-time errors. Eg:If a computer attempts to divide a number by zero.

There are certain kinds of errors, which cannot be identified during compilation. The program is run successfully but the output is wrong. This is due to a mistake in the logic of the program. These are known as logical errors.

E.g.: By mistake, using + instead of * during addition of two numbers.

1.4.4 Applications Of C++

- C++ is a versatile language for handling very large programs.
- C++ is suitable for virtually any programming task including development of editors, compilers, databases, communication systems and any complex real life application systems.
- Since C++ allows us to create hierarchy-related objects. We can build special object oriented libraries which can be used later by many programmers.
- C++ programs are easily maintainable and understandable.

1.5 Summary

Procedure oriented programming follows a Top Down approach where the problem is viewed as sequence of tasks. Functions are used to implement it.

To overcome the drawbacks, such as free movement of data around the program and as it is difficult to model real world problems, object oriented programming is introduced.

Object oriented programming follows a Bottom Up programming approach and it does not allow data to move freely.

The different concepts of OOPS like class, object, encapsulation, abstraction, inheritance, polymorphism, dynamic binding, message passing are briefly discussed.

Advantages and applications of OOPS are discussed.

The structure of a C++ program , writing a sample program, compiling, debugging and running of the programs are discussed.

The applications of C++ are covered.

1.6 Technical Terms

Object : An entity that can store data and, send and receive messages. An instance of class

Class: A group of objects that share common properties and relationships. A class is a new data type that contains member variables and member functions that operate on the variables.

Data Abstraction: The insulation of data from direct access by the programs.

Encapsulation: The mechanism by which the data and functions (manipulating this data) are bound together within an object definition.

Inheritance: Mechanism of deriving a new class from an old class.

Polymorphism: A property by which objects belonging to different classes are able to respond to the same message, but in different forms.

Dynamic Binding: The addresses of the functions are determined at runtime rather than compile time. This is also known as late binding.

1.7 Model Questions

1. Define Procedure oriented programming?
2. Define object oriented programming?
3. What is the difference between object oriented programming and procedure oriented programming?
4. Write the concepts of object oriented programming?

1.8 References

Object-oriented programming with C++
by E. Bala Gurusamy.

Problem solving with C++
by Walter Savitch

Mastering C++
by K.R.Venugopal,
Rajkumar Buyya, T.Ravi Shankar

AUTHOR:

M. NIRUPAMA BHAT, MCA., M.Phil.,
Lecturer,
Dept. Of Computer Science,
JKC College,
Guntur.