

Module-1:

LECTURE-1

Introduction:

Programmers write instructions in various programming languages to perform their computation tasks such as:

- (i) Machine level Language
- (ii) Assembly level Language
- (iii) High level Language

Machine level Language :

Machine code or machine language is a set of instructions executed directly by a computer's central processing unit (CPU). Each instruction performs a very specific task, such as a load, a jump, or an ALU operation on a unit of data in a CPU register or memory. Every program directly executed by a CPU is made up of a series of such instructions.

Assembly level Language :

An assembly language (or assembler language) is a low-level programming language for a computer, or other programmable device, in which there is a very strong (generally one-to-one) correspondence between the language and the architecture's machine code instructions. Assembly language is converted into executable machine code by a utility program referred to as an assembler; the conversion process is referred to as assembly, or assembling the code.

High level Language :

High-level language is any programming language that enables development of a program in much simpler programming context and is generally independent of the computer's hardware architecture. High-level language has a higher level of abstraction from the computer, and focuses more on the programming logic rather than the underlying hardware components such as memory addressing and register utilization.

The first high-level programming languages were designed in the 1950s. Now there are dozens of different languages, including Ada , Algol, BASIC, COBOL, C, C++, JAVA, FORTRAN, LISP, Pascal, and Prolog. Such languages are considered high-level because they are closer to human languages and farther from machine languages. In contrast, assembly languages are considered low-level because they are very close to machine languages.

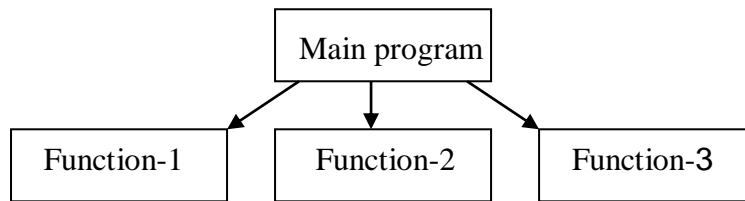
The high-level programming languages are broadly categorized in to two categories:

- (iv) Procedure oriented programming (POP) language.
- (v) Object oriented programming (OOP) language.

Procedure Oriented Programming Language

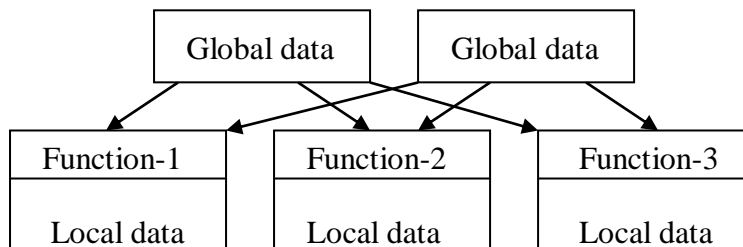
In the procedure oriented approach, the problem is viewed as sequence of things to be done such as reading , calculation and printing.

Procedure oriented programming basically consist of writing a list of instruction or actions for the computer to follow and organizing these instruction into groups known as functions.



The disadvantage of the procedure oriented programming languages is:

1. Global data access
2. It does not model real word problem very well
3. No data hiding



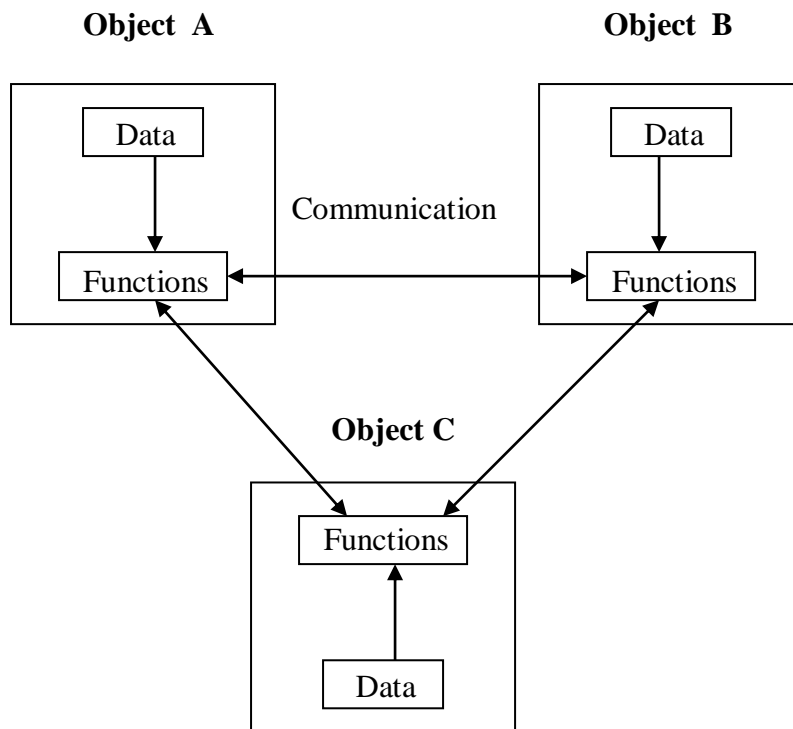
Characteristics of procedure oriented programming:

1. Emphasis is on doing things(algorithm)
2. Large programs are divided into smaller programs known as functions.
3. Most of the functions share global data
4. Data move openly around the system from function to function
5. Function transforms data from one form to another.
6. Employs top-down approach in program design

LECTURE-2

Object Oriented Programing

“Object oriented programming as 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”.



Features of the Object Oriented programming

1. Emphasis is on doing rather than procedure.
2. programs are divided into what are known as objects.
3. Data structures are designed such that they characterize the objects.
4. Functions that operate on the data of an object are tied together in the data structure.
5. Data is hidden and can't be accessed by external functions.
6. Objects may communicate with each other through functions.
7. New data and functions can be easily added.
8. Follows bottom-up approach in program design.

LECTURE-3

BASIC CONCEPTS OF OBJECTS ORIENTED PROGRAMMING

1. Objects
2. Classes
3. Data abstraction and encapsulation
4. Inheritance
5. Polymorphism
6. Dynamic binding
7. Message passing

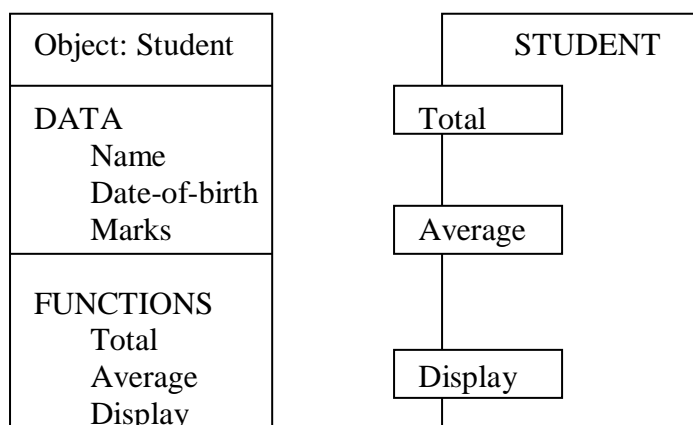
OBJECTS

Objects are the basic run-time entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program must handle.

The fundamental idea behind object oriented approach is to combine both data and function into a single unit and these units are called objects.

The term objects means a combination of data and program that represent some real word entity. For example: consider an example named Amit; Amit is 25 years old and his salary is 2500. The Amit may be represented in a computer program as an object. The data part of the object would be (name: Amit, age: 25, salary: 2500)

The program part of the object may be collection of programs (retrive of data, change age, change of salary). In general even any user –defined type-such as employee may be used. In the Amit object the name, age and salary are called attributes of the object.



CLASS:

A group of objects that share common properties for data part and some program part are collectively called as class.

In C ++ a class is a new data type that contains member variables and member functions that operate on the variables.

DATA ABSTRACTION :

Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as size, width and cost and functions to operate on the attributes.

DATA ENCAPSALATION :

The wrapping up of data and function into a single unit (called class) is known 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 provide the interface between the objects data and the program.

INHERITENCE :

Inheritance is the process by which objects of one class acquire the properties of another class. In 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 possible by designing a new class which will have the combined features of both the classes.

POLYMORPHISM:

Polymorphism means the ability to take more than one form. An operation may exhibit different instances. The behaviour depends upon the type of data used in the operation.

A language feature that allows a function or operator to be given more than one definition. The types of the arguments with which the function or operator is called determines which definition will be used.

Overloading may be operator overloading or function overloading.

It is able to express the operation of addition by a single operator say '+'. When this is possible you use the expression $x + y$ to denote the sum of x and y , for many different types of x and y ; integers, float and complex no. You can even define the $+$ operation for two strings to mean the concatenation of the strings.

DYNAMIC BINDING :

Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding means the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with a polymorphic reference which depends upon the dynamic type of that reference.