Introduction to Procedure Oriented Programming vs Object Oriented Programming

Procedure Oriented Programming

As we know, prior to object-oriented programming (OOP), programs were written using procedural languages. Procedural languages stress functions. The bigger problems are broken down into smaller sub-problems and written as functions. Procedural languages did not pay attention to data. As a result, the possibility of not addressing the problem in an effective manner was high. Also, as data was almost neglected, data security was easily compromised.

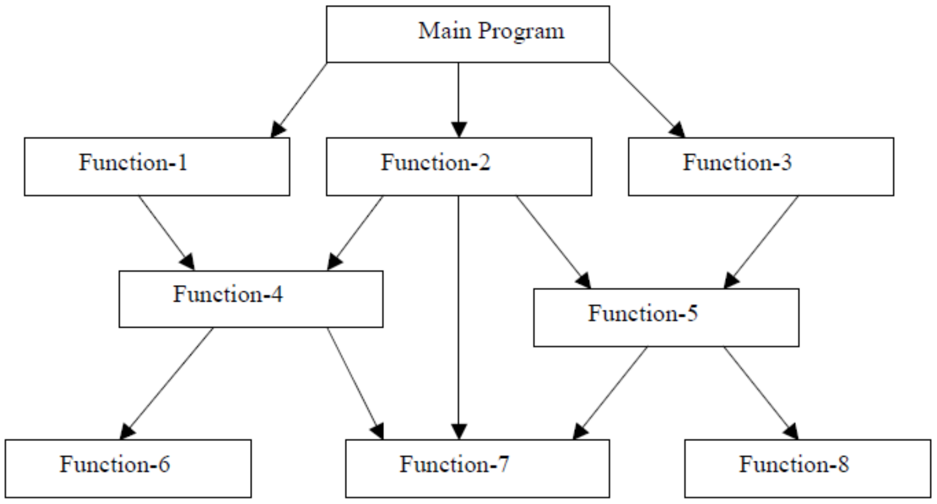


Fig. 1.2 Typical structure of procedural oriented programs

Advantages of POP

1. It emphasis on algorithm (doing this).
2. Large programs are divided into smaller programs known as functions.
3. Function can communicate by global variable.
4. Data move freely from one function to another function.
5. Functions change the value of data at any time from any place. (Functions transform data from one form to another.)
6. It uses top-down programming approach.

Disadvantages of POP

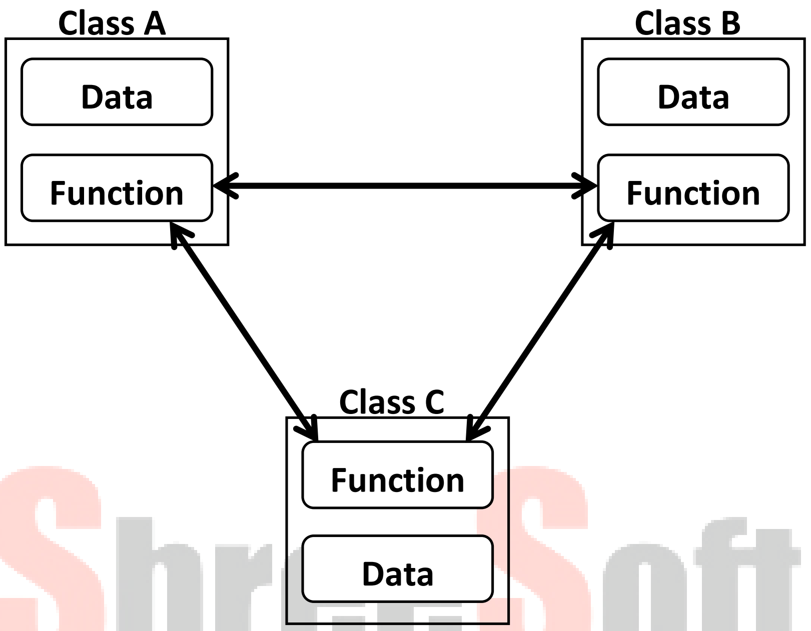
1. Procedural languages are difficult to relate with the real world objects.
2. Procedural codes are very difficult to maintain, if the code grows larger.
3. Procedural languages does not have automatic memory management as like in Java. Hence, it makes the programmer to concern more about the memory management of the program.
4. The data, which is used in procedural languages are exposed to the whole program. So, there is no security for the data.

All these drawbacks of procedural programming were overcome by object-oriented programming.

Object-Oriented Programming In C++

Object-oriented programming revolves around data. The main programming unit of OOP is the object. An object is a representation of a real-time entity and consists of data and methods or functions that operate on data. This way, data, and functions are closely bound and data security is ensured.

In OOP, everything is represented as an object and when programs are executed, the objects interact with each other by passing messages. An object need not know the implementation details of another object for communicating.



Features of OOPs

1. Classes and Objects

A class, on the other hand, is a blueprint of the object. Conversely, an object can be defined as an instance of a class. A class contains a skeleton of the object and does not take any space in the memory.

An object is a basic unit in object-oriented programing. An object contains data and methods or functions that operate on that data. Objects take up space in memory.

2. Abstraction

Abstraction is the process of hiding irrelevant information from the user. For Example, when we are driving the car, first we start the engine by inserting a key. We are not aware of the process that goes on in the background for starting the engine.

Using abstraction in programming, we can hide unnecessary details from the user. By using abstraction in our application, the end user is not affected even if we change the internal implementation.

3. Encapsulation

Encapsulation is the process using which data and the methods or functions operating on them are bundled together. By doing this, data is not easily accessible to the outside world. In OOP we achieve encapsulation by making data members as private and having public functions to access these data members.

4. Inheritance

Using inheritance object of one class can inherit or acquire the properties of the object of another class. Inheritance provides reusability of code.

As such we can design a new class by acquiring the properties and functionality of another class and in this process, we need not modify the functionality of the parent class. We only add new functionality to the class.

5. Polymorphism

Polymorphism means one things in multiple form. Polymorphism is an important feature of OOP and is usually implemented as operator overloading or function overloading. Operator overloading is a process in which an operator behaves differently in different situations. Similarly, in function overloading, the same function behaves differently in different situations.

6. Dynamic Binding

OOP supports dynamic binding in which function call is resolved at runtime. This means that the code to be executed as a result of a function call is decided at runtime. Virtual functions are an example of dynamic binding.

7. Message Passing

In OOP, objects communicate with each other using messages. When objects communicate, information is passed back and forth between the objects. A message generally consists of the object name, method name and actual data that is to be sent to another object.

Why C++ is Partial OOP language?

C++ language was designed with the main intention of using object-oriented features to C language. Although C++ language supports the features of OOP like Classes, objects, inheritance, encapsulation, abstraction, and polymorphism, there are few reasons because of which C++ is classified as a partial object-oriented programming language.

We present a few of these reasons below:

1) Creation of class/objects is Optional

In C++, the main function is mandatory and is always outside the class. Hence, we can have only one main function in the program and can do without classes and objects.

This is the first violation of Pure OOP language where everything is represented as an object.

2) Use of Global Variables

C++ has a concept of global variables that are declared outside the program and can be accessed by any other entity of the program. This violates encapsulation. Though C++ supports encapsulation with respect to classes and objects, it doesn’t take care of it in case of global variables.

3) Presence of a Friend Function

C++ supports a friend class or function that can be used to access private and protected members of other classes by making them a friend. This is yet another feature of C++ that violates OOP paradigm.

To conclude, although C++ supports all the OOP features mentioned above, it also provides features that can act as a workaround for these features, so that we can do without them. This makes C++ a partial Object-oriented programming language.

Advantages of OOPs:

1. Reusability
2. Modularity
3. Flexibility
4. Maintainability
5. Data and Information Hiding