# Chater-3

# Object Oriented Programming

**Program and Programing**

We know that the group of instructions written in order to complete a certain task is called a program and an operation in a computer is called an instruction. Generally, a program can be seen as a logical process in which data is given (input), that data is processed and then, as a result, we get data. To solve a problem by computer, it has to be changed in such a way that computer can understand, this process is called programming. To prepare this, we use some languages,such as Fortran, Pascal, Cobol etc. These languages are procedural languages. By the year 1960-70s, these languages were used very effectively, but by this time the programs were becoming big and complex.

## 3.1 Unstructured programming

While learning programming, we prepare small and simple programs at the initial level, they have one main program. Main program means - a set of instructions in which the data is available throughout the program and those data can be changed. See Figure 3.1

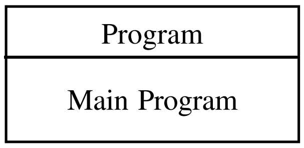

Figure 3.1: Unstructured Programming - program processes data directly

When size of a program increases, this kind of programming leads to a lot of problems. For example, if a group of same instructions has to be re-used, we will have to re-write that group, this unnecessarily will increase the size of program.It is natural to have difficulties to improve such a program. The solution to this problem came out that as, this group of instructions should be taken out from the main program and given a new name, and reference of that name should be used in the main program. Thus, formed the basis of procedural programming,in which other programs can be used in a main program. These programs can be understood as helpful processes, which we call procedures.

## 3.2 Procedural Programming

With the help of procedural programming, the size of program can be reduced, chances of errors are less and, maintenance is also easier. When a procedure is called in the main program, the control of the program goes to that procedure, the computer operates that group of instructions (procedure). After the completion of that group, the control goes back to the instruction written next in the main program. See Figure 3.2

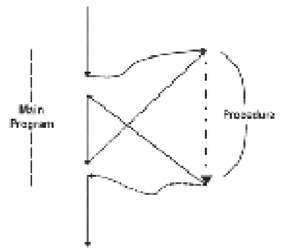

Figure 3.2: Execution of Procedure

In the main program, the control of program transfers to the procedure, instructions written in procedure are executed, control comes to the next statement in the main program.

With the help of procedures, we are able to reduce the size of program, and make it an error-free.

Now, we can assume that one program is a series of many procedures. Each procedure is called from the main program, the data is processed and the results are transmitted to the main program. Thus, when the whole series is completed, the main program gets the final result. The data is transferred from main program in the form of parameters. See figure 3.3

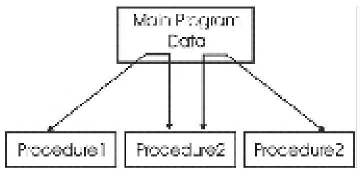

Figure 3.3: Procedural Programming - In the control of main programs - Procedure execution and data exchange.

So now we can say that the main program is divided into small segments which are called procedures. Common types of procedures that can be used in other programs, and likewise, the processors created by the user themselves can also work in another program, it is necessary that the procedures can be stored separately.

## 3.3 Modular Programming

In modular programming, the procedures are collectively stored in groups, which are called modules. In general, the procedures which are meant for similar kind of processing are kept in one module. The purpose is to get all of them at one place. These stored procedures can be used directly in the main program. The main program is divided into very small pieces and data processing is done through the procedures. See figure 3.4

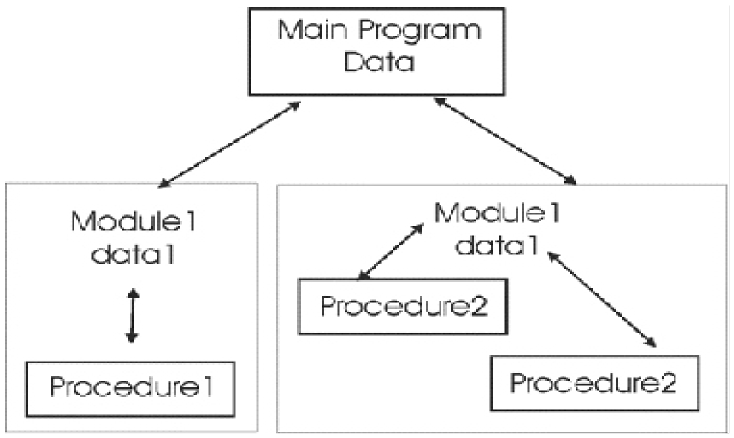

Figure. 3.4 : Modular Programming -Calls of procedures from different modules are coordinated in the main program, appropriate data is handled as parameters.

Each module has its own data. When the procedures of a module are used, each module maintains its internal system of data. But a module can be present in a program only once at a time.

## 3.4 Benefits of Procedural and Modular Programming Languages

- o Very good for general programming
- o Solution code may be available for many problems, so it may not be necessary to repeat coding for the same.
- o As required in case of machine languages, knowledge of hardware is not necessary.
- o Programs written in such languages can be run on another CPU after translating with the help of different compilers.

**Limitations**

- o Availability of many languages, the programmer has to stick to one language, and different languages require different types of expertise.

- o In-depth knowledge of language-specific programming instructions is necessary.
- o For problems requiring artificial intelligence, where logic is fuzzy (logical), procedural languages are not suitable.

## 3.5 Concept of Object-Oriented Programming

Object-Oriented Programming is different from procedural programming.Programs are based on classes and objects in such languages. Objects-classes are used with the help for methods written in them. Before the development of OOP, procedural languages were used. Programming instructions and data are separate entities in such languages and for this reason, chances of errors are high in such programming. This problem becomes more serious when the programs become lengthy, i.e. the number of instructions is higher or the problem itself is a complex one.

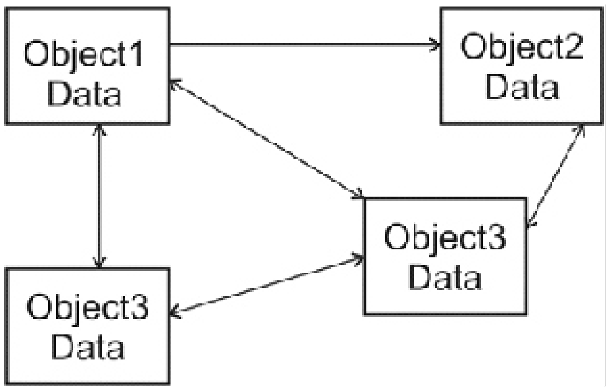

Figure 3.5: Object-Oriented Programming

### 3.5.1 Object

In OOP, an object is considered as an entity. Everything that is in existence and which can be seen, touched or felt logically is an object. It is quite possible that an object is a group of other small objects. One object can perform many activities. These activities define the behavior of the object. For example, a two-wheeler is an entityand this two-wheeler is made up of many other entities like wheels, seat, handle, pedals etc. We know that there are many two-wheelers which have some or all features, for example, bicycle, bike or scooter. At the same time pedals will not be present in all two wheelers. Another example, if we consider that "Student" is an object, to which we can relate information like student's name, address, class. Similarly, StudentStatus can be another object that which may contain details about presence, subject, exam_score, exam_results etc. Objects store data and instructions, and they are created for special processing. Object can be used through the available instructions in the language. An object can also call another object to perform an action through messages. The object to request the object to act through the data or the instruction is called Sender Object and the object that receives is called the Receiving Object.

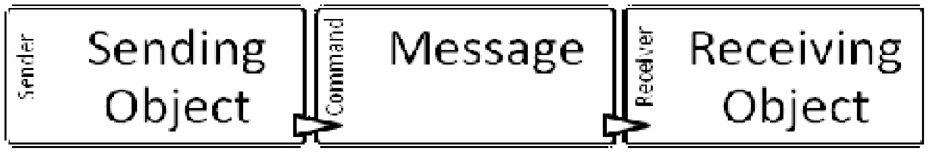

Figure. 3.6 Object

The control of the implementation now comes to the receiving object and it remains until the instruction is completed. After execution of the instruction, the control goes back to the sender object. For example, if we consider two objects -exam and student. Exam object, uses the student object to get the name of the student.

Sender object can also send information to the recipient object through message, it is called argument. The recipient object generally returns a value to the sender object. This value is used by the sender object in further processing. For example, the StudentStatus object wants to change the value of the student's attendance, for that he sends the new value into a message as a parameter. In this example the value returned by the receiving object will be ignored.

### 3.5.2 Method

How does the receiving object understand messages received from the sender object? How are those messages processed, and the parameters obtained with them are processed? When an object receives a message, the program code that accompanies that message is executed. In other words, these messages determine how the behavior of an object will be and the program code written in the object determines what the object will do. The code associated with the message is called method.

When the object receives a message, that message also contains the name of the method.It determines which program code will be executed and control is transferred to that code for execution.

We know, in procedural languages, a procedure is a group of instructions. method is also a group of instructions. The program code written is a method is similar to code in a procedure. Sending a message to an object is like invoking or calling a procedure.

### 3.5.3 Class

An object comes in to picture when execution of program takes place. While writing the code, we define it through class. Class is a data type and object is a variable. After defining the class, we can make its object variable as per the requirement. The data and program code that we write in definition, are consistent with the need for programming. They are called member data and member functions.

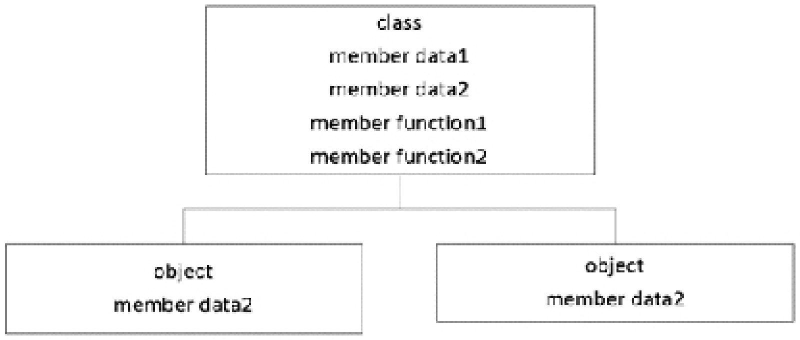

Figure. 3.7 Class

## 3.6 Characteristics of OOP

### Reusability

Assume that we write a program to solve one problem and we have some similar more problems. There are all the chances that we need to write programs again to solve such problems because one program written to solve a problem may not be useful in solving similar other problems.This happens because in traditional programming, data and programs are kept separate. For example, the new problem for which we need to write program is exactly same except the data types. We need different data types in our new problem. Similarly, there may be slight changes required in the logic, most of the code remains the same. Still, we need to write the program again as the old program cannot work in such situations. Object-Oriented programming is a system in which solutions can be written for new problems while changing the available solutions.It means we do not have to try again from the beginning to program for the new problem. We have already developed programs. What we need to do is make some changes according to our need. Reusability is possibly one of the most important features of Object Oriented Programming.

### Encapsulation

In procedural languages, the data and the processes which will use that data are defined separately. In object-oriented programming, as we go about defining the class, we write the data and the processes together. These processes are called methods. This is called encapsulation. The abstraction implies that whatever data is defined in a class, it can be used without further explanation and details. The advantage of encapsulation and abstraction is that the use of data structures and symbols is the same as the programmer has thought of while defining them. In this way, the information given in the class is hidden (information hiding) by which the programming becomes more secure.

**Polymorphism**

The behaviour and implementation of an object are different. Many objects may work for the same message or the same object can be used in different forms. The form of an object may vary according to the requirement at the time of execution. For example, we define a class as 'addition' for mathematical addition operation. In this class we define a method called 'add', in which we write the code to add two integer variables. In the same class there is another method called add, in which we write the code to add two float variables. Now,at the time of execution of the program, if two integer variables are used while the add method, the first method will come in action, and if two float variables are used, the second method will come in action. Here the point to note is that, there are two methods with the same name but the desired method will be used depending on the data sent at the time of execution. The same name is being used in different forms. Due to arrangement of polymorphism, the exchange of messages between the sender object and the receiving object becomes possible.

**Inheritance**

Inheritance is another important concept in object-oriented programming. After defining a class, if we have to define another class - a class whose properties are the same as the first class and we want to add some other properties - we can do it by use of inheritance. In such a system, we are also using reusability, due to this reason there is no need to do all the work again which we did while defining the first class. This also saves time and facilitates maintenance of the program.The class taking the property of a class is called a Sub Class or Derived Class, and the class from which properties are forwarded is called Super Class or Base Class.

## 3.7 Object-Oriented Problem Solving Approach

Object-oriented programming approach is similar to the approach we follow in our daily life to find solutions to problems. We identify the real-life objects that are helpful in solving the problem and use them in a certain order. We usually do it to solve our problems. Think about the objects in problem and use them to solve the problem. While programming, we create objects that can solve the problem. Based on the messages sent to the object, there are different operations are performed which solve the problem.

The process of resolving an object-oriented problem can be divided into four phases.

1. Identify the problem
2. Identify the objects needed for the solution
3. Identify messages to be sent to the objects
4. Create a sequence of messages to the objects that solve the problem.

## 3.8 Benefits of Object-Oriented Programming

- o Programming is easy. After examining a class thoroughly, the convenience of using in the work decreases the likelihood of errors.

- o Class can be considered as a "black box".We don't need to have the internal details of a class to use it. Methods available in a class can be used without the knowledge of details.

- o Unnecessary effort of writing the same type of programs is avoided. A class can be used directly as per our requirement. In other words, program code can be reused.

- o With the help of different compilers, code can be made in conformance with another CPU, which means code-portability is available.

## 3.9 C++ Basic Elements

C ++ uses the class to use the concept of objects. Keyword 'class' is used to create a class. Class is a user-defined data type that a programmer can use to define a real-time object for its programming, and then programming through a resource based on it. While defining we do not create objects, but make a model which is called a class. We make objects using class as per the requirement. For example, we define a class named student, and, in the program, can create different objects for different courses according to our need.

Each class has some features defined in it, these are the characteristics which we want in an object.They are called attributes of an object. It is not necessary that all the attributes are required in each object.We can create another class which has all the attributes the first class andsome new attributes too. This feature is available to us through inheritance.

Example

```
class class_name

{

private:

 Data_Members;

 Member_Functions;

public:

 Data_Members;
 Member_Functions;

 };
```

We create object to use class, which we call instance of a class. Class is a definition and an object a variable. A class contains variables known as data members, and procedures to handle these variables known as ember functions.

Each programming language is a set of symbols, specific words and rules. A program is written using all of these. There are some elements that are found in every language. We now discuss some of the elements available in C++.

**Character Set**

Group of characters which is recognized in a language is known as Character Set.

| | |
|---|---|
| Letters | A-Z, a-z |
| digits | 0-9 |
| Special Characters (विशेष अक्षर) | Space + - * / ^ \ () [] {} = ! = >< ' " $ , ; : % ! & ? # < = > = @ |
| Formatting characters | backspace, horizontal tab, vertical tab, form feed and carriage return |

## 3.10 Tokens

A token is a group of characters. Programmer writes program using these tokens. Tokens available in C++language are: Keywords, Identifies, Literal, Integer Constants.

**Keywords**

C++ language has certain words which are already defined in the language. They are reserved words. Meaning of these words is already known to the compiler and programmer cannot change that meaning.

**Identifiers**

C++ provides a feature of using symbolic names (identifiers) for various data elements (variable, function, class, module, or any other user-defined item) in C ++ is available to the programmer. These names are created by taking the letters from the C ++ character set. The rules for making the name are as follows:

o An identifier may have alphabets (A-Z, a-z), digits (0-9), and / or under score (_) in the name.

- o The initial character cannot be a number.
- o It should not be a reserved word.

**Literals**

Those data elements whose value cannot be changed in the program are called literals.

- ♦ Integer constant : Integer constants mean whole numbers which do not have any fractional part. C++provides three types of integer constants.
- ♦ Decimal Integer Constant: Numbers, first digit cannot be zero. Example: 78, -168, +4
- ♦ Octal Integer Constant: Numbers,first digit is zero. Example: 014.
- ♦ Hexadecimal Constant:Numbers, first two characters are ox or OX. Example: OX24C

**Character Constant:** One or more character which are written within single quotation marks, for example 'A', '4', '\t'. The characters which cannot be printed directly using keyboard, like tab, backspace, are printed using escape sequence.

**Floating Constant:** Fractionalnumbers.These can be written in fractional form or exponential form, example -0.342, 314159E-5

String Constant: Sequence of characters written in double quotation marks is known as string constant. '\0' is added at the end of string, which denotes the end of string. Example, "TECHNOLOGY" will be stored in memory as "TECHNOLOGY\0"and its size will be 12 characters.

**Data Types**

Basic Data Types

| **Type** | **Keyword** |
|---|---|
| Boolean | bool |
| Character | char |
| Integer | int |
| Floating point | float |
| Double floating point | double |
| Valueless | void |
| Wide character | wchar_t |

Following data type modifiers can be used to change the basic data types.

- signed
- unsigned
- short
- long

Various variable types, how much memory would be needed to store the value in memory, and what is maximum and minimum value which can be stored in those variables, is shown in the following table.

| Type | Typical Bit Width | Typical Range |
|---|---|---|
| char | 1byte | -127 to 127 or 0 to 255 |
| unsigned char | 1byte | 0 to 255 |
| signed char | 1byte | -127 to 127 |
| int | 4bytes | -2147483648 to 2147483647 |
| unsigned int | 4bytes | 0 to 4294967295 |
| signed int | 4bytes | -2147483648 to 2147483647 |
| short int | 2bytes | -32768 to 32767 |
| unsigned short int | Range | 0 to 65,535 |
| signed short int | Range | -32768 to 32767 |
| long int | 4bytes | -2,147,483,648 to 2,147,483,647 |
| signed long int | 4bytes | same as long int |
| unsigned long int | 4bytes | 0 to 4,294,967,295 |
| float | 4bytes | +/- 3.4e +/- 38 (~7 digits) |
| double | 8bytes | +/- 1.7e +/- 308 (~15 digits) |
| long double | 8bytes | +/- 1.7e +/- 308 (~15 digits) |
| wchar_t | 2 or 4 bytes | 1 wide character |

Depending upon the compiler and the computer you are using, the size of data types in memory may differ from what is shown here.

**Input-Output**

In the standard library of C++, there is one header file called - iostream.h, which may be used to read data from keyboard and display it on the screen.

Following C++stream objects may be used for input-out

cout console output

cin console input

**cout object**

cout is used to display message on screen using << insertion operator.

cout<< "Hello World";displays Hello world on screen

cout<< 250; // displays number 250 on screen

cout<< sum; // displays the value of variable sum on screen

If variable and constant are to be displayed in combination, << can be used more than once. Example

cout<< "Area of Crop Field is "<< area<< " square meter" ;

**cin object**

cin is used to input value using keyboard by the user. To store the value in memory >> extraction operator is a must.

cin >> marks; // Data will be received from keyboard and will be stored in variable marks.

**Important Points**

1. With the help of procedural programming, the size of the program can be reduced, thereby reducing the chances of errors.

2. In modular programming, the procedures are collectively stored in modules.

3. Procedural languages are suitable for general programming.

2. For problems requiring artificial intelligence, where logic is fuzzy, procedural languages are not compatible.

3. Class is a data type and object a variable.

4. To program for a new problem, we do not need to try again from the beginning, by making changes in the program already made, we can get the same solution.

5. When defining a class, we write data and code together. This is called encapsulation.

6. Abstraction means that whatever data or function is defined in a class, it can be used without further explanation and explanation.

7. At the time of execution of the program, the object may be as per our requirements.

8. A method in object-oriented programming is similar to what is a procedure in procedural programming.

9. In the context of procecudural Programming methods are group of those instructions. Method and programe code written is code of procedure.

## Exercises

**Objective Type Questions**

1. The ability to have different forms

| | | | |
|---|---|---|---|
| A. | Inheritance | B. | Polymorphism |
| C. | Member function | D. | Encapsulation |

2. The process of taking out an object's properties is called

| | | | |
|---|---|---|---|
| A. | Polymorphism | B. | Inheritance |
| C. | Reusablisty | D. | Data hiding |

3. What facilities in C++ makes it a strong language?

| | | | |
|---|---|---|---|
| A. | Easy implementation | B. | Reusing old code |

C. Writing new code D. All of the above

4. Which facility is not available in C++?

   A. Encapsulation B. Abstraction

   C. Polymorphism D. Exceptions

5. To be called 'object-oriented', a programming language must have facility for

   A. Encapsulation B. Abstraction

   C. Polymorphism D. All of these

6. Which statement is not true?

   A. To perform a specific task, part of the code is called a function

   B. With the help of a function, large and complex programs can be divided into small and simple.

   C. Functions are helpful to carry out common tasks using the available code.

   D. A function can be used only once in a program.

7. A function defined in a class is called

   A. Member Variable B. Member function

   C. Class function D. Classic function

8. In OOP which concept means "Only the information required is communicated outside the object"

   A. Encapsulation B. Abstraction

   C. Data hiding D. Data binding

9. Which approach is better for saving information and understanding real-time ex amples, like information about vehicle or employee?

   A. Procedural Approach B. Object Oriented Approach

   C. Modula Approach D. None of the above

10. Advantages of object-oriented programming

    A. Re-use of code B. Better use of code

    C. Error free D. All of the above

**Very Short Type Questions**

1. Likelihood of errors in procedural programming are less because the size of the program can be .......... (reduced)

2. First O in OOP is ........... (Object)

3. Data and ........ may be taken in a new class, in hierarchy.

4. In C++, class is a ................ data type.

5. An advantage of Reusabilityis........................ (Ability to use already written code)

**Short Type Questions**

1. Differentiate between class and object.

2. Differentiate between procedure and method.

3. What are the tokens in C ++?

4. What are the character constants in C ++?

5. What are the data type modifier?

**Essay Type Questions**

9. Explain the characteristics of OOP.

10. Describe basic data types of C++.

11. Explain the use of 'cin'and'cout' with the help of examples.

12. Differentiate between Encapsulation and Abstractation.

**Answer Key**

1. B 2. D 3. D 4. D 5. D 6. D 7. B 8. C 9. B 10.D