Object-Oriented Programming Using

Rishabh Anand

KHANNA PUBLISHERS®

Investing in Learning®

Object-Oriented Programming Using C++

Rishabh Anand

PostDoc (AI & ML), Ph.D. (Computer Science), MBA, LMCEGR, ASR & IAENG
Global Service Delivery Manager (ITES), PfMP®,PMP®, PRINCE2®, DevOps-PM™,
TIL®, CSM®, CSSGB™ & Kanban-ASC™ Certified Professional

KHANNA PUBLISHERS®
Investing in Learning®
Operational Office :
4575/15, Onkar House, Opp. Happy School,
Ground Floor, Daryaganj, New Delhi 110 002
Phones : 011-45033819 • *Mob*. 09811541460
email : contactus@khannapublishers.in

Published by :

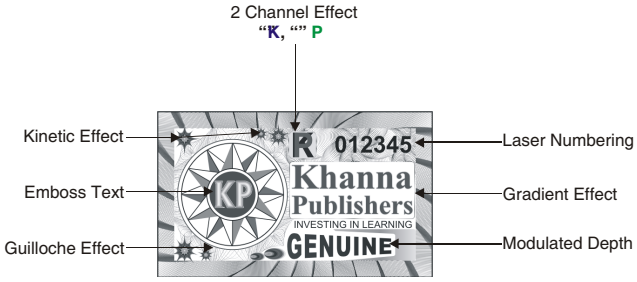
Romesh Chander Khanna & Vineet Khanna
for KHANNA PUBLISHERS
2-B, Nath Market, Nai Sarak
Delhi- 110 006 (India)

Website : www.khannapublishers.in

Hologram & Description

To all readers of our books, to prevent yourself from being defrauded by piraters, please make sure that there is an Hologram on the cover of our books with the below specifications. If you find any book without Hologram and Description, please mail us at **contactus@khannapublishers.in**

Thanking you



ISBN No. : 978-81-950287-3-3

First Edition : **2021**

Preface

Object-Oriented Programming (OOP) has become the preferred programming approach of the software industry, as it offers a powerful way to cope with the complexity of real-world problems. Considered one of the most powerful languages, C++ lays a strong foundation for object-oriented programming and has helped in enhancing Java and Python Programming.

Since its creation by Bjarne Stroustrup in early 1980s, C++ has undergone many changes and improvements. The language was standardized in 1998 by the American National Standards Institute (ANSI) and the International Standards Organization (ISO) by incorporating not only the new features but also the changes suggested by the user groups.

C++ is decorated (?) with a lot of bombastic jargon. One of my aims here was to keep this jargon at bay and concentrate on the underlying concepts instead. At most places, I have tried to show how things work and more importantly why do they work that way.

Object-Oriented Programming with C++ is for the programmers who wish to know all about C++ language and Object-Oriented Programming. It, however, assumes that the reader is familiar with C language and need not be an expert. In its simple and easy-to-understand style, the book explains the what, why and how of Object-Oriented Programming with C++.

The book provides numerous examples, illustrations and complete programs. The sample programs are meant to be both simple and educational. Wherever necessary, pictorial descriptions of concepts have been included to improve clarity and facilitate better understanding. The book also presents the concept of object-oriented approach and discusses important elements of object-oriented analysis and design of systems.

I realise my young readers always seek to be challenged with new and innovative things. With AICTE, UGC and other academic bodies in India and abroad stressing more on Outcome-Based Education, I felt there was need to reframe and classify the questions under learning domains as per Bloom's Taxonomy.

Hope you enjoy reading this first edition as much as I did in writing it !

New Delhi — Rishabh Anand

Acknowledgments

No one walks alone and when one is walking on the journey of life just where I start to thank those who joined me, walked besides me, and helped me along the way.

Over the years, those that I have met and worked with have continuously motivated me to write this book. So at last, here it is. So, perhaps this book and its pages will be seen as "thanks" to the tens of thousands of you who have helped bringing out this book in the form what is today.

There was, in and always will be the love of my students whom I taught, for what is written in this book has been the fruit of experience that I taught them. Finally, thanks to my mother, my wife and to rest of my family for their patience and support during the long hours of writing this book and above all there is the one almighty whose humble children we are. It is his blessings we cherish and pray for. It is the blessing I wish for you.

My special acknowledgment to all the authors whose books, journals and paper are referred while writing this book.

I deeply express my heartful thanks to the Management Team and Editorial Staff of Khanna Publishers for publishing the book in such a beautiful get-up and well in time.

When you finish the journey through this book, having variety of experiences, ideas and thoughts. You are welcome to express your opinion at rishabhanand009@gmail.com.

Rishabh Anand

Contents

CHAPTER 1

Principles of Object-Oriented Programming

1.1. INTRODUCTION

When Thermos Flask was invented, it was labeled as one of the greatest inventions in the sense that how it remembers to keep hot things hot and cold things cold. The same analogy hold true for object-oriented programming (OOP). OOP is the most dramatic innovation in software development and some of its features draw parallels to that of thermos Flask.

The primary objective of OOP is to provide clearer, reliable, and easily maintainable approach to program design. OOP involves concepts that are new to programmers of traditional programming languages, also known as procedural languages, imperative languages or problem oriented languages, such as pascal, C, FORTAN, etc. These concepts, such as data abstraction, encapsulation, data hiding, inheritance overloading and polymorphism, lie at the heart of object-oriented programming. It is very important to understand that object-oriented programming is not primary concerned with the details of the program operation; instead it deals with the overall organization of the program.

Computers are designed to follow instructions. A computer program is a set of instructions that tells the computer how to solve a problem or perform a task.

Programming refers to the method of creating a sequence of instructions to enable the computer to perform a task. It is done by developing logic and then writing instructions in a programming language. In short, a programming language is a language used to write computer programs.

During the last few decades, the increasing complexity of software system is due to the increase in the uses requirement. To build today's complex software, it is just not enough to put together a sequence of programming statements and set of procedures and modules, we need to incorporate sound construction techniques and program structures that are easy and simple to comprehend, implement and modify.

Since the invention of the computer, many programming approaches have been tried and the primary motivation in each has been the concern to handle the increasing complexity of programs that are reliable and maintainable. These include following techniques have become popular among computer programmers over the last two decades. These techniques are:

1. **Modular Programming:** It emphasizes separating the functionality of a program into only one aspect of the desired functionality. Examples: Ada, Algol, C#, COBOL etc.

2. **Top-down programming:** It focuses on the use of modules. The program is broken up into small modules so that it is easy and simple to trace a particular segment of code in

the software program. The modules at the top level are (main module) those that perform general tasks, and those at the lower level (sub-ordinate, modules) perform specific tasks. Each module in this approach is based on the functionality of its functions and procedures and the programming begins from the top level of hierarchy and progresses towards the lower levels.

Examples: C++, Java etc.

3. **Bottom-up Programming:** It refers to the style of programming where a application is constructed with the description of modules. The description begins at the bottom of the hierarchy of modules and progresses through higher levels until it reaches the top. Example: 'C'.

4. **Structured Programming:** It is concerned with the structures used in a computer program. Generally, structures of computer program comprise decisions, sequences and loop.

5. Structured Programming: It is concerned with the structures used in a computer program. Generally, structures of computer program comprise decisions, sequences and loops. Examples: Pascal, PL/I etc.

With the advent of language such as C language, structured programming become very popular in the year 1980s. Structured programming was a powerful tool that enabled programmers to write moderately complex programs fairly easily and simply. However, as the programs size grew larger, even the structured approach failed to show the devised results in terms of easy-to-update, bug-free, easy-to-maintain, and reusable programs.

OOP (Object Oriented Programming) is an approach to program organization and development that attempts to eliminate some of the pitfalls of conventional programming (structured) methods by incorporating the best of structured programming features with several powerful new concepts.

OOP (Object Oriented Programming) is a new way of organizing and developing programs and has nothing to do with any particular language.

With the rapidly changing and highly competitive software industry, programming paradigms are also changing according to the requirements to cope up with the new challenges. Software engineer continuously look for the new approaches to software design and development to deal with the increased complexities. The traditional procedure-oriented (POP) failed to show the desired bug-free result for a complex program. The POP is based on the functions so there is no flexibility in programs that are implemented using these languages. The latest programming approach object-oriented programming (OOP) offers a new and powerful way to cope with these complexities. The objective of OOP is clear; reliable and easy to understand programs. OOP operates on abstract values called objects which can communicate to another object by using message passing. In this approach the data and functions are combined together to give us more flexibility in programming. C++ is a very popular OOP language. Some vital features of OOP ar data biding, encapsulation, inheritance and polymorphism.

1.1.1. Object-oriented Paradigm

Object-oriented approach is a major solution to the drawbacks that programmers encountered in procedure-oriented approach. The main idea programmers encountered in procedure-oriented approach. The main idea behind objected-oriented programming languages is to encapsulate both data and functions that operate on these data into a meaningful unit known as an object. In an object-oriented programming environment, data is considered as a critical element and is tied to functions that operate on it. A set of data that is encapsulated in an object is known as the **member data** and the functions that operates on these data are

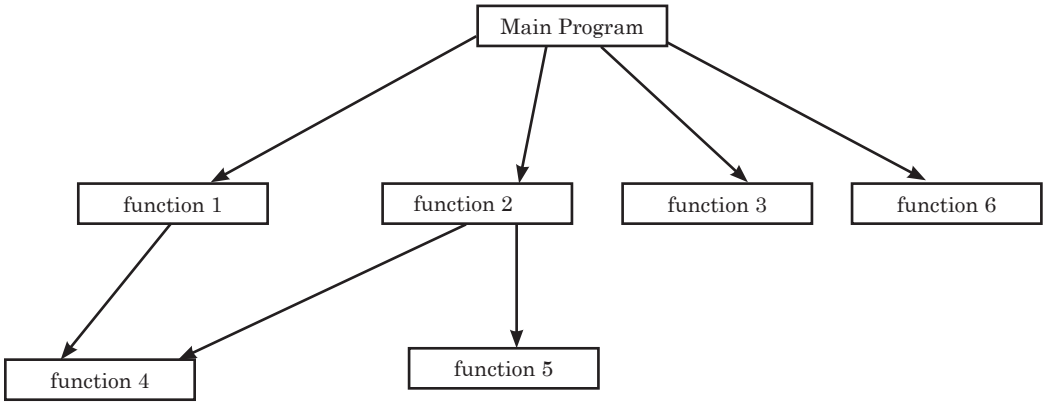
known as the *member functions*. In OOP, data cannot move freely around the system and data can be accessed and modified only by the *member function*. An external function cannot access the member data of an object and, hence, the data is protected from modification by external function. OOP treats data as an important entity compared to procedure-oriented approach where the emphasis is an functions rather than data. Data is hidden in an object-oriented programming and, hence, it is safe from the accidental modification. The member data of a particular object can be accessed only by the member function of the particular object, while the member functions of an object can be accessed by the member function of the of objects.

Object-oriented programming is a new approach to programming through modularization of program by creating separate area for data and functions that is used for creating copies of such modules on demand.

The process of software development involves the application of various tools and techniques for the design and development of software systems. During the last few decodes, many new tools and techniques have been invented for the development of efficient software systems. The main motivation for the continuous development of these tools and techniques has been to help the software developers handle the increasing complexity of the software systems and develop the efficient ones. The major reason for the failure of software projects is due to the inability of the software developers in handling the complexity efficiently. Inability of handle the complexity of the problem results in failure of the software in meeting the desired requirements and results in higher cost. One of the major goals of the software development team is to hide this complexity from the user and to create an illusion of simplicity. Complex system are hierarchic with multiple levels and each level represents different levels of abstraction with clear boundaries between the levels. The increasing complexity of software system is due to the increase in the user requirement. The complexity in a software system is arbitrary in nature which varies from application to application programming languages like C, FORTAN, Pascal and BASIC are based on POP approach. POP concept follows the top-down design approach. In this approach, we have a single program (normally a main function) which is divided into small pieces called procedures or functions. These functions consist of a sequence of instructions that performs some specific task with clearly defined interface with the other functions. Each statement in the function tells the function tells the computer to do something, that is, get some input, perform some operation display the output. In this approach, the focus is on the processing so the algorithms are needed to perform the desired computation. The main function calls other functions or procedures by passing the required data to the function or procedure. In procedure-oriented languages the emphasis is on doing the things, or we can say the emphasis is on the action. The data on which the action is taken or performed is of least importance. These languages have been an avid choice for the programmers because the larger program is divided into a number of small subprograms using functions so that handling of such programs is easy. They are also suitable for scientific applications development because there are many built-in scientific functions such as log, cos, sun etc. However, there are various short comings of the languages also. As the programs grow larger and more complex, this approach beings to shown signs of strain and fails to give the desired results in terms of bug-free and reusable programs. Another problem with this approach is that the data defined inside the function (local) or procedure cannot be accessed from outside the function. To make the data accessed by all functions, they are declared outside all functions (globally) so that any function can manipulate the data. As many functions access the same data, important data may be lost anytime. Several functions can access the data therefore the way data is arranged or organized is very crucial the arrangement of data cannot be changed without modifying all the functions that access it.

1.2. STRUCTURED VS OBJECT ORIENTED DEVELOPMENT (OOP)

In structured programming, the primary focus is on the function. In structured programming program has a fixed well defined structure. A typical program structure for structured development is shown below.



For example suppose we want to write a program for the following problem:

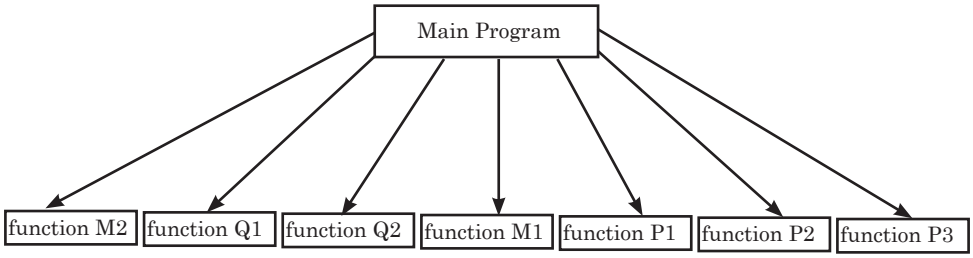
Write a program which implement a company, in which there are following departments.

1. Quality
2. Production
3. Marketing

If we solve the above program with the help of structured programming, we first divide the total department into different fucntions. Suppose we need two functions Q1 and Q2 for Quality, three functions P1, P2 and P3 for production, two functions for marketing M1 and M2 then the program structure look like.

In a multifunction program many data items are placed as global. i.e., these data items are accessed by all the functions.

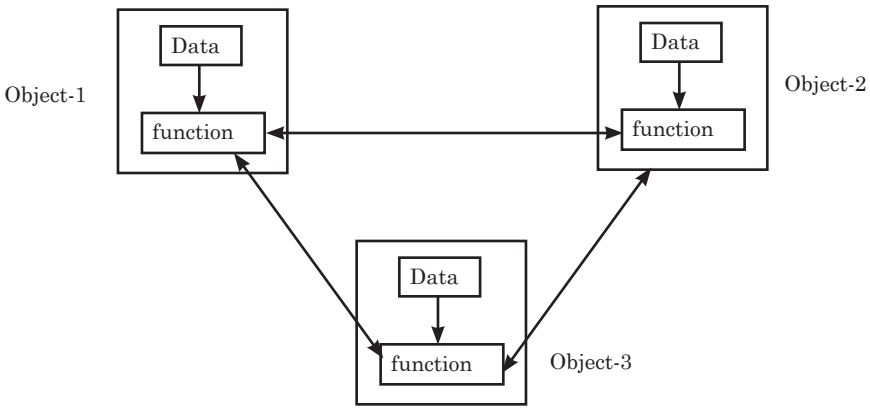
Some data items are local to a function, i.e., these data items are accessed by only the function in which they are defined.



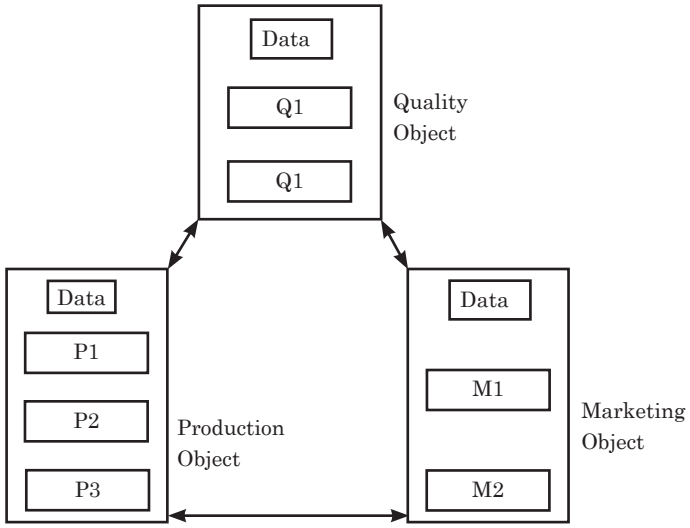
In structured programming most of the data are global, if program is too large it is very difficult to identify what data is used by which fucntion. Another serious draw back with structured programming is that it does not model real world problem.

1.2.1. Object-Oriented Programming Solve These Drawbacks

In object-Oriented programming data and function (which operate these data) are organised in one entity called class. The organisation of data and function in object-oriented program is shown below.



Object oriented programming has other feature also like inheritence, polymorphism, overloading, data abstraction, and encapsulation. We study these properties in element of object oriented programming section i.e., object is the variable of a class.



Note : *Object is the run time entities in Object-oriented programming (OOP).*

We study object and class in next section. If we solve the same problem which we have solved with the help of structured programming, then the program structure looks like.

1.3. ELEMENTS OF OBJECT ORIENTED PROGRAMMING

In object oriented programming data and functions are organised in one entity. Object-oriented programming has some other characteristic also like, inheritence, polymorphism etc.

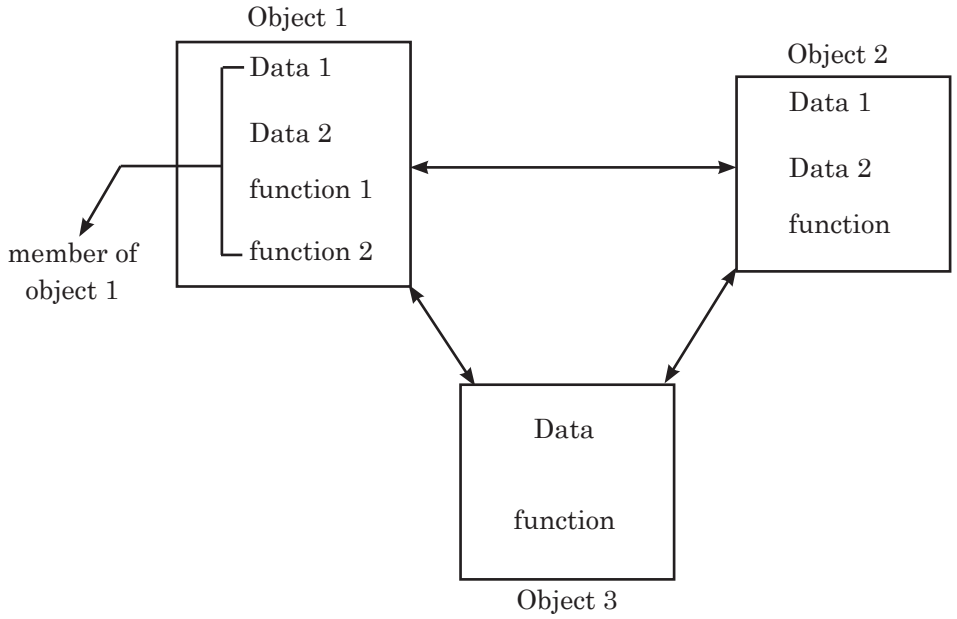
The elements of object-oriented programming are:

1. Object and class
2. Data abstraction
3. Encapsulation
4. Inheritence
5. Polymorphism

1.3.1. Object and Class

Class is an entity in which data and fuctions, (which operate these data) are organised.

Object is a class variable. When a program is executed, the objects interact by sending messages to one another.



If object 3 wants to access function 1 of object 1 then that access throws the object 1.

The data and function inside a class are known as member of that class.

1.3.2. Encapsulation

In OOP the data and function are in one single unit called class, this property is known as encapsulation. Data encapsulation is the most striking feature of class. The data is not accessible to the outside of class. Only the functions which are inside the class can access the data. The insulation of the data from direct access by program is called data hiding.

1.3.3. Abstraction

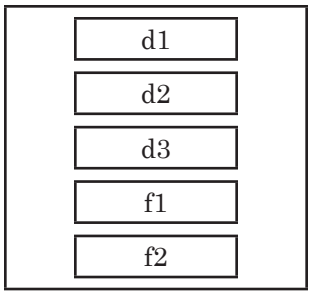
Abstraction means representing essential features without including the background details or explanation. Class use the concept of abstraction and are defined as a list of abstract data and functions to operate on these data. Classes encapsulate all the essential properties of the object that are to be created.

Since the classes use the concept of data abstraction, they are known as Abstract Data Type (ADT).

1.3.4. Inheritance

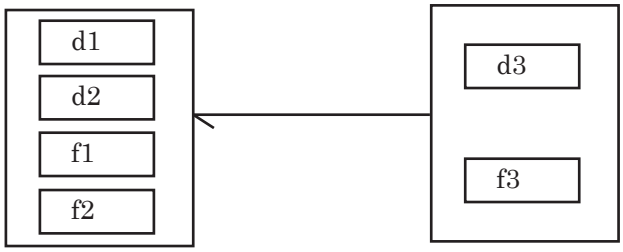
Inheritance is a method by using which we can create a new class by extending and enhancing existing class. The existing class is called the base class and the new class is called derived class. Inheritence is very powerfull feature of OOP.

For example we have a class X which has following member: two functions F_1 and F_2 and three data d1, d2 and d3.



Class X

Suppose after sometime we want to add one function, f3 and data d3 in that class X. We can do this by designing the class again but the main drawback of this is we have again to check the class X (test & debug). But in oop with the help of inheritance we can do this. Drive a new class Y from the base class X. In class Y add only function f3 and data d3, Because the class Y is inherited from the class X. Therefore the member of class X are automatically copied into class Y.



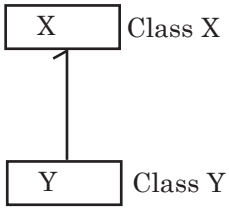
Class X
(Base class)

Class Y
(Derived class)

If we create an object of class Y then in that object the member are d1, d2, d3, f1 , f2 and f3. But if we create an object of class X, in that object the member are d1, d2, f1 and f2.

That means the base member are transformed into derived class but not in reverse order.

If class Y derived from class X, then that is written like is :



1.3.5. Polymorphism

Using functions in different ways, depending on what they are operating on is called the polymorphism. i.e., one thing with several distinct forms.

For example suppose we are writing a program, which calculates the area of circle, area of triangle, area of rectangle. With the help of polymorphism we can give same name area to all the area functions. In other words function overloading is called polymorphism.

We can overload the operator also that is known as operator overloading for ***example + operator*** is used to add, numeric data (int or float). If we use ***same + operator*** for adding two objects, then this is known as operator overloading.

The Object-Based and Object-Oriented Languages

The language which supports the class and objects but does not support the inheritance and polymorphism is called object-based language, for example, Module-2 and 83-Ada. The object-oriented language supports all features such as class, object, inheritance, polymorphism, etc. as discussed earlier in this chapter. Ada-95, Modula-3 C++, Smalltalk, object pascal and Java are some examples of OOP languages.

Depending on the object features supported by the languages are classified into two categories namely. Object-based programming language and object-oriented programming language.

1. **Object-Based Programming Language.** These languages support encapsulation and object identity without supportive important features of OOP languages such as Polymorphism, inheritance and message based communications.

Ada and Modula-2 are examples of object-based programming languages. In short, ***object-Base language = Encapsulation + Object Identity***

2. **Object-Oriented Programming Languages.** The high-level language that implements the object oriented (OO) concepts like encapsulations, data hiding, operator overloading, etc. is known as an object-oriented language (also called as OO language).

Object-oriented languages incorporate all features of object based programming languages along with inheritance and polymorphism. Therefore, an object-oriented programming language is defined by the following statement ***Object-oriented language = Object-based Features + Polymorphism + Inheritance***

Depending on the extent to which they support object oriented (OO) concepts, the object Oriented (OO) languages are classified into following several categories:

(*i*) ***Pure Languages:*** Languages that not only support but also enforce all object-oriented concepts are called pure object oriented languages. In pure object-oriented languages, everything from character and punctuations to modules is treated as an object. Small Talk, Eiffel and Ruby are examples of pure object-oriented languages.

(*ii*) ***Multi-paradigm Languages:*** Languages that support many programming paradigms like procedural programming, generic programming, etc. one of which is object-oriented paradigm are called multi-paradiagm object-oriented languages. C++ is the example of multi-paradigm object oriented language.

(*iii*) ***Hybrid Languages:*** Languages that support some (not all) of the object-oriented concepts are called hybrid languages. Python and C# are examples of hybrid object oriented languages.

In this book we study the OOP concept with the help of C++.

1.4. OBJECT MODELLING

For designing a software by object oriented programming technique we have to first understand about the elements of object oriented programming. We have already studied about elements of object oriented programming in section 1.2.

For design a software by OOP we have to identity objects, design the model of object and we have to create link between object. We have already covered some topic under section 1.2. In this section we will mainly cover the topic related to link between objects.

1.4.1. Association

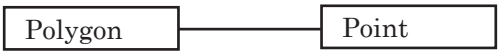
Association represent the relation between objects for examples suppose the object are company and person then the association is like follow:

1.4.2. Relationship

The company is employer of any person, and the person can be employee of any company.

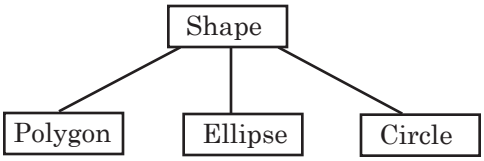
1.4.3. Aggregation or "A-Part-of" Relationship

If two objects have whole part relationship then that is called aggregation. For example suppose polygon is one object and point is another object, then the relationship between point and polygon is aggregation type, because all point can be used to design a polygon.



1.4.4. Specialization and Generalization (Is-A or Is-A-Kind-of Relationship)

If one object can be used to design some other objects then the relationship is generalization type. For example shape is one object polygon, ellipse, circle are other object and the shape is used to design polygon, ellipse, circle then the relationship between shape and polygon, shape and ellipse, circle are other objects and the shape is used to design polygon, ellipse, circle then the relationship between shape and polygon, shape and ellipse, shape and circle is generalization type.



1.5. FUNCTIONAL MODELLING

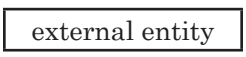
Information is transformed as it flows through a computer used system. We can create a flow model for any computer based system, regardless of size and complexity. It should be noted that the model may be applied to the entire system or to the software element only.

1.5.1 Data Flow Diagrams

As information moves through software it is modified by a series of transformations. A data flow diagram (DFD) is graphical technique that depicts information flow and the transforms that are applied as data move from input to output. The DFD is also known as a data flow graph or a bubble chart. The data flow diagram may be used to represent a system or software at any lever of abstraction. In fact DFD may be partitioned into levels that represent increasing information flow and functional details. Therefore the DFD provides a mechanism for functional modelling as well as information flow modelling. A level 0 DFD, also called a fundamental system model or a context model, represents the entire software element as a single bubble with input and output data indicated by incoming and outgoing arrows, respectively. Additional processes (bubbles) and information flow paths are represented as the level 0 DFD is partitional to reveal more detail. For example, a level 1 DFD might contain five or six bubbles with interconnecting arrows, Each of the processes represented at level 1 are subjunctions of the overall system depicted in the context model.

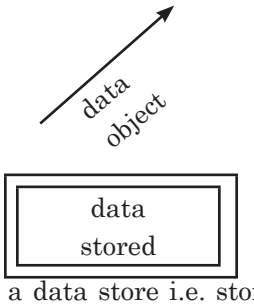
The basic notation used to create a DFD are as follow:

(1) A rectangle is used to represent an external element (example hardware, a person, another program).



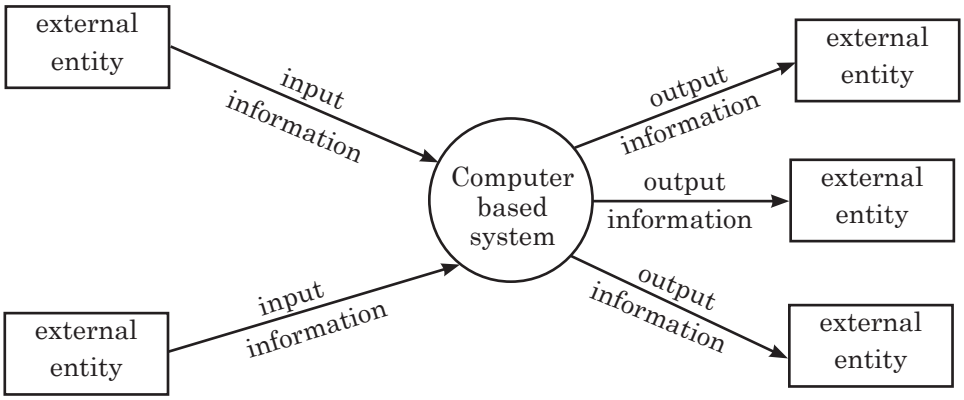
(2) A circle represents a process or transform that is applied to data (or control) and change it in some way

(3) An arrow represents one or morn data items or data objects. All arrows on a data flow diagram should be lebellel.



(4) The double line represents a data store i.e. stored information that is used by the software.

The following diagram represent an information flow model:



1.6. IMPORTANCE OF OOP

The concept of object oriented programming can directly be mapped with real life problem. Even we can say that the idea of OOP are taken from the real life. Now there are two questions.

Question 1: How we can say that OOP concepts are taken from real life?

Question 2: If concepts of OOP are taken from real life then what is the benefit of that i.e. how this increase the popularity of OOP?

Now we will analyse the answer of each question one by one. After analysing both answers we can easily understand why OOP is popular.

1.6.1. Answer of Question 1

In a real life the objects are: Pen, Pencil, Car, Book etc. These objects have two main things:

(1) Structure (or Attribute)

(2) Behaviour (or Nature)

Example 1: For example the structure of pen consists (1) color of pen (2) Length of pen (3) The material by using which body of pen is made etc.

Object-Oriented Programming Using

About the Book

This book offers solid, effective and easy to understand approach to the study of fundamental Object Oriented Programming. It renders expansive information about a wide array of topics like C++, arrays, structures, unions, bit fields, functions, pointers, template, exception handling, file handling and graphics with numerous examples. The text comprises nineteen chapters and each chapter is further divided into modules of major topics. Each module has a uniform structured presentation starting with learning objective, declaration, implementation, example programs, operations, and types, summary, multiple choice sections, programming assignments, review questions followed by the solution of the programming assignments.

The book is a boon for general readers, C++ Professionals, and students from both graduate and postgraduate courses in computer engineering, who are inquisitive to explore each and every aspect of OOPS and C++.

About the Author

Rishabh Anand is an eminent academician; plays versatile roles and responsibilities juggling between industry, research, publications and consultancy. He has completed his PostDoc (AI & ML) from Sao Paulo State University, Brazil, Ph.D. (Computer Science) from University of Bristol, United Kingdom, Degree of Master of Business Administration as MBA Management from International MBA Institute, Switzerland and Program Diploma in Innovation Management from International Business Management Institute, Germany. Also, he is the Reviewer/Editor for IJTESSS, IJISP, IJCAC, IJECME, IJICTE, JITR, IJMPA, IJTHI, IRJET, IJCRT and IJSDR. He is a prolific author with 34 Text and Reference books to his credit. He is also associated with various professional bodies like SCIEI, LMCEGR, IAENG, Internet Society, IAOP, and IAOIP. He is currently working in ITES MNC industry as a Global Service Delivery Manager with overall 15 years of experience. He is CDPT™, PMP®, PRINCE2®, DevOps-PM™, ITIL4®, CSM® & Kanban-ASC™ Certified Professional.

KHANNA PUBLISHERS®
ISO 9001:2015
4575/15, Onkar House, Opp. Happy School,
Ground Floor, Daryaganj, New Delhi-110002
Phones: 011-45033819, 9811541460
E-mail: contactus@khannapublishers.in

Website:
www.khannapublishers.in

ISBN 819502873-X
9 788195 028733