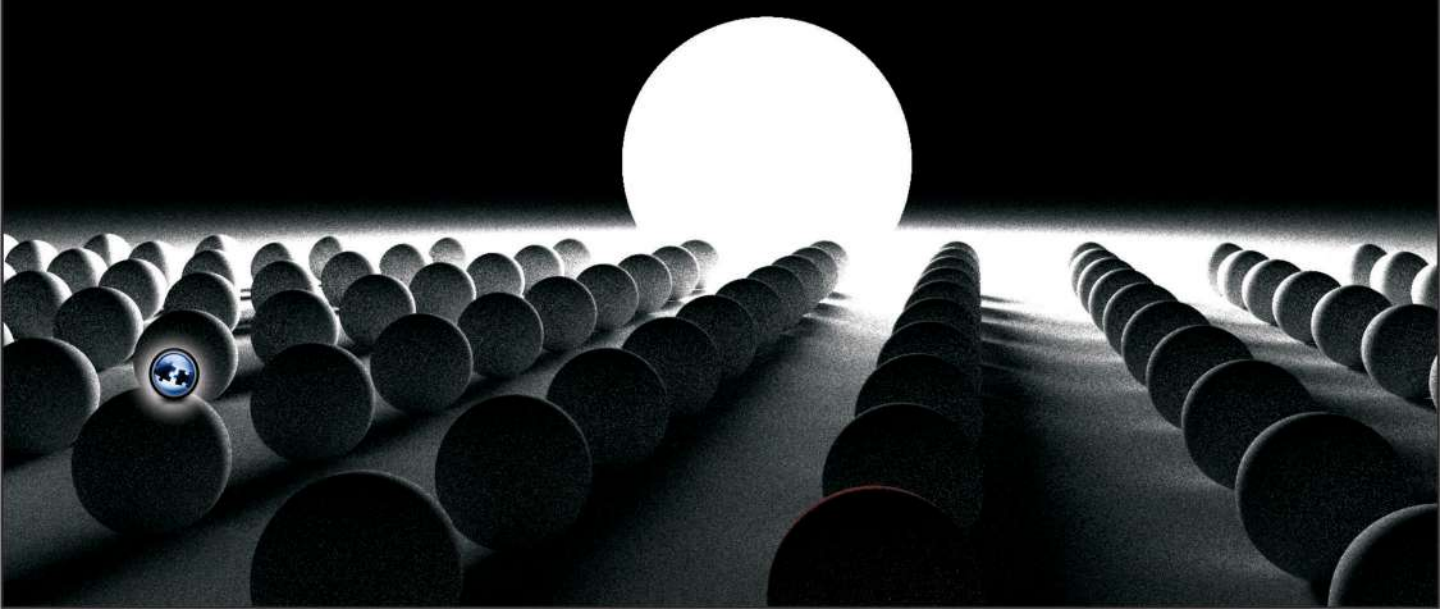
Object - Oriented Programming C++ Simplified

Hari Mohan Pandey



Preface

This book **"*OBJECT-ORIENTED PROGRAMMING C++ SIMPLIFIED*"** is a comprehensive, hands-on guide to C++ programming but one that doesn't assume you've programmed before. (People familiar with earlier programming or another structured programming language will, of course, have an easier time and can move through the early chapters quickly.)

Soon, you will write sophisticated programs that take full advantages of C++'s exciting and powerful object-oriented nature. You will start as a beginner and when you have finished this book, you will have moved far along the road to C++ mastery. I have tried hard to cover at the least the fundamentals of every technique that a C++ professional will need to master.

I have also made sure to stress the new ways of thinking needed to master C++ programming, so even experts in more traditional programming languages can benefit from this book. I have taken this approach because trying to force C++ into the framework of older programming languages is ultimately self-defeating, you can't take advantage of its power if you continue to think within an older paradigm.

To make this book even more useful, there are extensive discussions of important topics left out of most other introductory books. There are whole chapters on objects, including non-trivial examples of building your own objects with C++. When I teach you process of inheritance at that time I have introduced you to develop C++ code on behalf of inherited diagrams. There is a chapter for input-output manipulators which shows you number of functions used in C++ for input/output control. In the same chapter I teach you about manipulator and ways to develop your own manipulator. There is a whole chapter for miscellaneous new features of C++. There is also a chapter on exception handling. The book has also dealt about keeping file information through the chapter File Handling. It also has chapters for Standard Template Library and for the String Class and a whole chapter for showing the hidden secrets of C++. The book also includes lots of examples with step-by-step explanation and an extensive discussion of sorting and searching techniques and lots of tips and tricks. In sum, unlike many of the introductory books out there, I not only want to introduce you to a topic, but I go into it in enough depth that you can actually use the techniques for writing practical programs.

Now a confession: My original goal was to make this book a "**one-stop resource**", but, realistically, C++ has gotten far too big and far too powerful for any one book to do this. Nonetheless, if you finish this book, I truly believe that you will be in a position to begin writing commercial-quality C++ programs! True mastery will take longer. I have tried to give suggestions that can take you to next level.

• HOW THIS BOOK IS ORGANIZED

The subject matter of this book is divided into **15 chapters (including chapter 0)**. Each chapter has been written and developed with immensely simplified programs **(except chapter 1, which is foundation chapter for various programming methodology)** which will clear the core concepts of the C++ language. The book **"OBJECT ORIENTED PROGRAMMING C++ SIMPLIFIED"** has been written specially for those students who are tyro in the field of programming. Inside the book you will find numerous programs instead of just code snippet to illustrate even the basic concept. The book assumes no previous exposure to the C++ programming language. It also contains some good programming examples which might be useful for experienced programmers. All the programming examples given in the book have been tested on **VC++ compiler**, **Turbo C++ 3.0** and **Turbo C++ 4.5 compilers** under **windows** and **DOS**.

Each chapter contains a number of **examples** to explain the theoretical as well as practical concepts. **Every chapter is followed by questions to test the student performance and retentivity**.

Here are short descriptions of the chapters:

Chapter 0: Covers the topics of basic introduction of programming methodology and introduction of OOP like **structured programming** by which one can understand the elementary elements (**sequence structure, Loop or iteration and Decision structure**) of any programming language. The chapter also explains the basic approaches (**Bottom-up** and **Top-down**) and the basic concepts of object-oriented programming too. It gives an idea to programmer to categorise any programming language into **object-oriented or object based language.**

Chapter 1: This chapter gives the details of fundamental aspects of "object oriented design and analysis" and covers the details of **"Grady Booch Approach",** principles used for OOAD. It flashes the concepts where OOAD fits in software development life cycle.

Chapter 2: In this chapter I have explained the historical development of C++ language. The chapter also gives introductory idea of tokens, variables, data types and basic structure of C++ program. In the same chapter I have explained the method of **compiling** and **executing** the C++ program on **Turbo C++3.0**, **Turbo C++ 4.5 and VC++.**

Chapter 3: This chapter introduces programmers about the behaviour of operators used in C++. Here I have explained the most of the common features applied in C and C++ both, because as we say C++ is super set of C then **operators and expressions** used in C must be implemented with C++ too.

Chapter 4: Covers the **operators used only with C++ and not with C**. Here I have covered the operators like **scope resolution operator, reference variables, bool data type**. This chapter gives idea of **dynamic memory allocation** and operators **new and delete** for dynamic memory allocation in C++.

Chapter 5: Gives the idea of **declaring function (prototyping),** function of **main ()** function, introduction of **recursion**. It also gives the meaning of **call by reference** and **call by address** and difference between **call by reference** and **call by address**. Here I have explained the functionality of **inline function** and **function overloading** too.

Chapter 6: Gives the introduction of **class and objects** used in C++. Here I have put the comparison of **structure and class**, way of **accessing private data** and given an idea about

passing and returning objects. In this chapter I have given some very crucial elements of C++ like **array of objects, friend function, Static class members and constant member function**. All these concepts play a very important role in software development.

Chapter 7: This chapter covers the behaviours of a **constructor.** Here I show the role of different types of constructor like **default constructor, constructor with parameters, copy constructor.** This chapter also gives the ideas of dynamic constructor and destructor.

Chapter 8: Gives ideas to programmer to **overload different types of operators** used in C++ like, **binary operators, assignment operator, unary operators**. Overloading with the help of **friend function** and rules of overloading any operator and way for **type conversion** too.

Chapter 9: In this chapter we will deal the concept of **inheritance,** different types of inheritance *i.e.,* **single level, multilevel, multiple, hierarchical and hybrid**. Here I have also defined the different visibility modifier with respect to inheritance. Application of **constructor and destructor** in inheritance and concept of **containership** is also defined in the same chapter.

Chapter 10: This chapter gives the way of implementing the concepts like **pointer to objects**, **this pointer**, and way of **binding**, what is **virtual function** and how to work with **virtual function**, rules for **virtual function.** This chapter also gives the comparison of **virtual function** and **pure virtual** function. Also included are the fundamental concepts of **object slicing** and **virtual destructor.**

Chapter 11: Here I have explained the concepts of C++ **stream classes** and f**ormatted** and **unformatted** input and output operation applied in C++ as well as the concept of **manipulator.**

Chapter 12: This chapter gives the idea of how to handle **file** in C++ programming language and introduces the programmer about the fundamental concepts of **file streams, way of opening and closing file, different modes of opening a text file** in C++. This chapter also gives the approaches to **check end of any file**, **Random access in file**. In the same chapter I have put the introductory ideas of **command line argument** and ways of working with **binary mode** and **error handling mechanism** with file handling.

Chapter 13: Covers the introductory idea of **template programming** *i.e.,* **function template and class template.**

Chapter 14: This chapter deals with basic concepts of **exception handling mechanism and how to handle exception with the help of class,** how to **re-throw** an exception, how to **catch** all exceptions.

Chapter 15: Covers all the experiments given in the syllabus.

Appendix–1 Presents some language-technical elements.

Appendix–2 Discusses the technical questions which are generally asked in technical interviews.

• IMPLEMENTATION NOTE

The language used in this book is "Pure C++" as defined in the C++ standard. Therefore, the examples ought to run on every C++ implementation. The major program fragments in this book were tried using several C++ implementations. Examples using features only recently adopted into C++ didn't compile on every implementation. However, I see no point in mentioning which implementations failed to compile which examples. Such information would soon be out

of date because implementers are working hard to ensure that their implementations correctly accept every C++ feature.

• SUGGESTIONS FOR C PROGRAMMERS

The better one knows C, the harder it seems to be to avoid writing C++ in C style, thereby losing some of the potential benefits of C++. Please take a look at Appendix B, which describes the differences between C and C++. Here are a few pointers to the areas in which C++ has better ways of doing something than C has:

1. Macros are almost never necessary in C++. Use `const` or *`enum`* to define manifest constants, `inline` to avoid function-calling overhead, *`template`*s to specify families of functions and types, and *`namespace`*s to avoid name clashes.
2. Don't declare a variable before you need it so that you can initialize it immediately. A declaration can occur anywhere a statement can, in *for-statement* initializers, and in conditions.
3. Don't use *`malloc`*`()`. The *`new`* operator does the same job better, and instead of *`realloc`*`()`, try a *`vector`*.
4. Try to avoid *`void*`*, pointer arithmetic, unions, and casts, except deep within the implementation of some function or class. In most cases, a cast is an indication of a design error. If you must use an explicit type conversion, try using one of the "new casts" for a more precise statement of what you are trying to do.
5. Minimize the use of arrays and C-style strings. The C++ standard library *`string`* and *`vector`* classes can often be used to simplify programming compared to traditional C style. In general, try not to build yourself what has already been provided by the standard library.

To obey C linkage conventions, a C++ function must be declared to have C linkage. Most important, try thinking of a program as a set of interacting concepts represented as classes and objects, instead of as a bunch of data structures with functions twiddling their bits.

• SUGGESTIONS FOR C++ PROGRAMMERS

By now, many people have been using C++ for a decade. Many more are using C++ in a single environment and have learned to live with the restrictions imposed by early compilers and first generation libraries. Often, what an experienced C++ programmer has failed to notice over the years is not the introduction of new features as such, but rather the changes in relationships between features that make fundamentally new programming techniques feasible. In other words, what you didn't think of when first learning C++ or found impractical just might be a superior approach today. You find out only by re-examining the basics.

Read through the chapters in order. If you already know the contents of a chapter, you can be through in minutes. If you don't already know the contents, you'll have learned something unexpected. I learned a fair bit writing this book and I suspect that hardly any C++ programmer knows every feature and technique presented. Furthermore, to use the language well, you need a perspective that brings order to the set of features and techniques. Through its organization and examples, this book offers such a perspective.

• EXERCISES

Exercises are found at the ends of chapters. The exercises are mainly of the write a program variety. Always write enough code for a solution to be compiled and run with at least a few test cases. The exercises vary considerably in difficulty, so they are marked with an estimate of their difficulty. The scale is exponential so that if an exercise takes you ten minutes, it might take an hour and it might take a day. The time needed to write and test a program depends more on your experience than on the exercise itself. An exercise might take a day if you first have to get acquainted with a new computer system in order to run it. On the other hand, an exercise might be done in an hour by someone who happens to have the right collection of programs handy.

Any book on programming in C can be used as a source of extra exercises for some introductory chapters. Any book on data structures and algorithms can be used as a source of exercises for some middle chapters for the formation of algorithms.

Acknowledgement

First of all, I would like to say thanks to ***"BABA VISVNATH"*** for their constant bless in writing this book. As understanding of the study like this is never the outcome of the efforts of a single person, rather it bears the imprint of a number of people who directly or indirectly helped me in completing this book. I would be failing in my duty if I don't say a word of thanks to all those who have offered sincere advice to make this book educative, effective, and pleasurable. I would like to acknowledge **My Father Dr. V. N. Pandey My Mother Smt. Madhuri Pandey, My sisters Ms. Anjana Pandey, Ms. Ranjana Pandey and My brother Mr. Man Mohan Pandey.**

I have immense pleasure in expressing my whole hearted gratitude to **Prof. RR Sedamkar (Associate Dean, MPSTME),** for providing help and guidance during the writing of this book.

I also wish to thank my lots of student whose conceptual queries have always helped me in digging the subject matter deep.

I am thankful to all the employees of **MPSTME, NMIMS University** especially **Mr. Bharat Thodji** and **Mr. Mahendra Joshi** who have been supporting me in all types of Lab activities during the writing of this book.

How can I forget my close buddy **Lecturer Shreedhar Desmukh** who has supported me most of the time during writing of the book? I am also thankful to the librarian of MPSTME at Shirpur **Mr. Anand Gawdekar** who provided each and every good book timely.

—Author

Salient Features of the Book

(*a*) Lucid explanation of OOP concept.

(*b*) Covers C features in nutshell.

(*c*) Emphasis is on new features of C++.

(*d*) Overloaded almost all types of operators.

(*e*) 12 fully explained programs on type conversion.

(*f*) Virtual Functions explored in depth.

(*g*) Covers advance features.

(*h*) Detailed coverage of exception handling and templates.

(*i*) Over 600 thoroughly explained programs.

(*j*) Plenty of exercises to try.

(*k*) Detailed description of file handling with more than 40 thoroughly explained programs.

List of Experiments

OBJECT-ORIENTED PROGRAMMING LAB

1. Program illustrating function overloading feature.

2. Programs illustrating the overloading of various operators:

Ex: Binary operators, Unary operators, New and delete operators etc.

3. Programs illustrating the use of following functions:

(*a*) Friend functions (*b*) Inline functions

(*c*) Static member functions (*d*) Functions with default arguments.

4. Programs to create singly and doubly linked lists and perform insertion and deletion operations.

Using self referential classes, new and delete operators.

5. Programs illustrating the use of destructor and the various types of constructors (no arguments, constructor, constructor with arguments, copy constructor etc).

6. Programs illustrating the various forms of inheritance:

Ex. Single, multiple, multilevel, hierarchical inheritance etc.

7. Write a program having student as on abstract class and create many derived classes such as Engg, Science, Medical, etc., from student class. Create their objects and process them.

8. Write a program illustrating the use of virtual functions.

9. Write a program which illustrates the use of virtual base class.

10. Write a program which uses the following sorting methods for sorting elements in ascending order.

Note: Use function templates

(*a*) Bubble sort (*b*) Selection sort (*c*) Quick sort.

11. Write a program which illustrates the use of class templates:

Ex. (*a*) Stack class (*b*) Queue class.

12. Write programs illustrating file handling operations:

Ex. (*a*) Copying a text file (*b*) Displaying the contents of the file etc.

13. Write programs illustrating the console I/O operations.

14. Write programs illustrating how exceptions are handled (ex: division-by-zero, overflow and underflow in stacks etc.).