Competitive Coding for Learners in C++

From basics to brilliance in simple steps

Dr. Ankush Mittal

www.bpbonline.com

First Edition 2025

ISBN: 978-93-55516-565

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

About the Author

Dr. Ankush Mittal holds a B.Tech in Computer Science and Engineering from the Indian Institute of Technology (IIT), Delhi. He got his PhD from the National University of Singapore (NUS). He has worked in premier institutes such as NUS, Singapore, IIT Roorkee and BITS Pilani. He has received several national and international awards, such as the IIT Roorkee Outstanding teacher award, the Young Scientist award from The National Academy of Sciences India, the IIT Roorkee Star performer and the IBM faculty award. Dr. Mittal has a keen interest in philosophy and teaching all levels of students, from school kids to college students. He is presently serving as the Vice Chancellor of COER Univ., Roorkee and as Adjunct Professor at IIT Mandi.

Preface

Embarking on the journey of learning programming can be both exhilarating and challenging. This book is not just a compilation of code snippets; it is a carefully crafted resource aimed at making the fundamentals of C++ programming readily accessible. Whether you are a beginner taking your first steps or an experienced coder seeking to refine your skills, this book is designed to cater to your needs.

Our approach is rooted in simplicity yet powerful in its impact. Each chapter is dedicated to a fundamental programming topic, breaking down complex concepts into digestible segments. The goal is not just to teach syntax but to foster a deep understanding of programming principles and logic building.

Structured meticulously, each chapter follows a logical progression. It begins with a revisit of essential basics, providing a solid foundation for the topic at hand. Following this, you will encounter a variety of challenges – output-based questions, error identification tasks, and multiple-choice questions – designed to reinforce your understanding.

However, the true essence of learning programming lies in practical application. Hence, each chapter is accompanied by an array of programming problems. These problems are carefully curated to cover a spectrum of scenarios related to the chapter's topic. They are more than just exercises; they are opportunities to hone your problem-solving skills, sharpen your logic, and apply your coding prowess.

To complement your learning, video lectures are available. These lectures serve as visual guides, walking you through programming nuances and providing additional insights. Combining text and video ensures a comprehensive learning experience catering to various learning preferences.

Take the time to absorb the concepts, grapple with the challenges, and relish the victories. The joy of coding lies not only in the end result but in the process of crafting elegant solutions to complex problems.

Whether you are here to lay the foundation for a promising career or to expand your programming repertoire, we invite you to immerse yourself in the world of C++ programming. Let the journey begin – happy coding!

Chapter 1: Introduction to General Concepts - explains everything needed by the programmer to develop basic coding requirements. Starting from declaring a variable

and learning to take input from user, to control flow, data-types, handling variables and working with various types of operators. Furthermore, the chapter also gives the reader a set of practice questions to try and learn the concepts from and finally gives some coding questions with tweaked cases to practice on own.

Chapter 2: Single Loop - presents a detailed overview of various types of loops. It covers aspects of for, while and do-while loop and also shows a comparative study between all three. The explanations are expanded by covering loop control statements, such as break and continue. There are essential practice questions as this chapter that covers fundamental aspects of loops. Thereafter the coding questions leads to actual programming skills.

Chapter 3: Single Loop: Advanced - It is a continuation of the previous chapter, and builds on the loops concept described previously.

Chapter 4: 1D Arrays - builds the ability to understand the basic implementation of single dimensional arrays. The concepts discussed around this chapter are ranging from as basic as declaration and initialization of arrays to performing operations on the arrays. The chapter also slightly touches dynamic memory allocation which is further covered in the next chapter.

Chapter 5: Advanced Arrays - allows the reader to learn fundamental concepts related to arrays with first layer already done in chapter 4. In this chapter, some advance topics other than those discussed in chapter 4 are highlighted. These are finding the sizes of array and dynamic memory allocation. The concept of dynamic memory allocation allows the readers to learn the efficient memory management especially when dealing with arrays whose sizes are fixed at runtime. The concepts are enhanced via practice questions and coding problems discussed thereafter.

Chapter 6: Nested Loops - gives special attention to nested loops, demonstrating how to implement applications like multidimensional arrays and repetitive tasks with multiple levels of iteration and explaining concepts that help the reader solve real complex problems in terms of program flow through practice questions.

Chapter 7: Series and Patterns - helps the readers understand the role of loops in sequence control and their automating repetitive tasks. Thus allowing the readers to learn to implement the previously studied concepts. In this chapter creating patterns and sequences will be taught using different types of loops and control mechanisms of loops.

Chapter 8: Advanced Patterns and Sequences - is a continuation of chapter 6, and chapter 7 where more light has been thrown over implementing advanced patterns such as fractal pattern leading to revision of essential mathematical series. Hence, building the aptitude for more real-life problems.

Chapter 9: Strings - covers the basics of strings in C/C++, including their declaration, initialization, and the importance of the null character ('\0'). It also explains how to traverse strings using loops, work with ASCII values, and handle implicit and explicit data type conversions related to strings.

Chapter 10: Recursion - is dedicated to understanding this powerful technique of recursion. It is very essential for any programmer. This chapter helps in building this concept from understanding the style of recursion to how it actually unfolds in the memory. Also, the chapter discusses in detail the process image helping the reader to understand the program and its background functioning better. Thereby, building the concepts more strongly through practice questions and coding question over the same.

Chapter 11: 2D Array - will allow the readers to build an advance layer over the previous concept discussed in chapters 4 and 5 of a two dimensional array. After understanding and practicing over 1D array, 2D array becomes simpler to build over. Thereby, in this chapter all the basic necessities are covered required to understand and implement 2D arrays. There are hands on coding questions to practice the coding problems as well as some basic practice questions to build the concept clearly.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/23b87j9

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Competitive-Coding-for-Learners-in-CPlusPlus.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Introduction to General Concepts

Introduction

In this chapter, we will discuss the fundamental concepts of C++ programming, focusing on the essential aspects of declaring variables and handling user input. Variables serve as crucial components in programming, enabling the storage and manipulation of data. By understanding how to declare variables with specific data types and assign values to them, programmers can effectively manage information within their programs. Furthermore, the chapter explores the process of taking user input for different data types, providing insights into interacting with users and creating dynamic and interactive applications. Through practical examples and detailed explanations, readers will gain a solid understanding of these foundational concepts in C++ programming, setting the stage for further exploration and application in their coding endeavors.

Structure

In this section, we will discuss the following topics:

- Declaring a variable
- Taking user input
- Control flow
- Datatypes
- Variables
- Operators

1.1 Declaring a variable

Variables play a pivotal role in programming, serving as containers for storing and manipulating data. In C++, declaring a variable involves specifying its data type and name.

Syntax and variable declaration:

```
dataType variableName;
```

Example:

```
#include <iostream>
int main()
{
    int age;  // Declaration
    age = 25; // Assignment
    // Alternatively, combine declaration and assignment
    double temperature = 98.6;
    std::cout << "Age: " << age << std::endl;
    std::cout << "Temperature: " << temperature << std::endl;
    return 0;
}
```

Output:

```
Age: 25
Temperature: 98.6
```

1.2 Taking user input

Taking user input for different data types involves using appropriate methods to handle various types of input. Let us explore examples for different data types and include relevant explanations:

Example 1

In this example, we will learn to take input from users for different data types:

```
#include <iostream>
int main() {
    int number; double value; char letter; bool flag;
    std::cout << "Enter an integer: ";
    std::cin >> number;
    std::cout << "Enter a floating-point number: ";
```

```
    std::cin >> value;
    std::cout << "Enter a character: ";
    std::cin >> letter;
    std::cout << "Enter a boolean value (0 or 1): ";
    std::cin >> flag;
    return 0;
}
```

Explanation:

- Four variables (number, value, letter, and flag) are declared, each representing a different data type: integer, double (floating-point), character, and Boolean.
- The program prompts the user to enter an integer, and the input is read using **`std::cin`** into the respective variable. Similarly, this is does for all the data types.
- The main function returns 0, indicating successful program execution.
- This example demonstrates the usage of **`std::cin`** to receive input from the user and store it in the appropriate variables.

Example 2

In this example, we will learn how to consider string as an input and output:

```
#include <iostream>
#include <string>
int main() {
    std::string text;
    std::cout << "Enter a string: ";
    std::getline(std::cin, text);
    std::cout << "You entered: " << text << std::endl;
    return 0;
}
```

Explanation: The **`std::getline`** function is used for string input. It reads an entire line, including spaces, until the user presses Enter and stores it in the text variable.

1.3 Control flow

To enable the implementation of intricate logic in a computer program, it becomes necessary to go beyond sequential execution. This involves executing statements out of order, particularly when the decision to execute one set of statements is made during program execution. This is achieved through selection structures like if, if-else, and switch. Additionally, to iterate over a set of statements based on a specific condition, repetition structures such as while, do-while, and for are utilized. The following sections will provide detailed explanations of each of these control statements.

1.3.1 The if statement

The **if** statement is one of the most important, and frequently used. It guides the flow of control in a computer program. It evaluates a conditional expression, generally involving relational operators, and enables the programmer to choose a particular set of statements to execute. The if statement is used in the following ways:

- if (<expression>) { / / {}'s are used to group statements:

```
<statement>
<statement>
}
```

- if (<expression>) { / / if - else form:

```
<statement>
}
else {
<statement>
}
```

- if (<expression>)

```
{ // if – elseif - else form
<statement>
}
else if
{
<statement>
}
else
{
<statement>
}
```

- Ternary operator:

  The ternary operator in C++ is a concise way to express conditional statements. It is a shorthand form of an if-else statement and is often used for simple, one-line decisions. The syntax of the ternary operator is as follows:

```
condition ? expression_if_true : expression_if_false;
```

Here is a breakdown of the components:

- **condition**: A Boolean expression that is evaluated. If it is true, the expression before the colon is executed; otherwise, the expression after the colon is executed.
- **expression_if_true**: The value or expression to be returned if the condition is true.
- **expression_if_false**: The value or expression to be returned if the condition is false.

Example 1: Determining if a number is even or odd

Use the following code to determine if a number is even or odd:

```
#include <iostream>
int main() {
    int number;
    std::cout << "Enter an integer: ";
    std::cin >> number;
    if (number % 2 == 0) {
        std::cout << number << " is even." << std::endl;
    } else {
        std::cout << number << " is odd." << std::endl;
    }
    return 0;
}
```

Explanation: The provided C++ code prompts the user to input an integer, and then check whether the entered number is even or odd using the modulo operator (%). If the remainder after dividing the number by 2 is zero, it prints that the number is even; otherwise, it declares the number as odd. The program demonstrates a simple example of using an if-else statement for conditional branching based on the parity of the entered integer.

Example 2: Using ternary operator vs. if-else statement

In the following example, we will see the equivalence of ternary with its respective if-else statement:

```
#include <iostream>

int main() {
    int number = 10;

    // Ternary Operator
    std::cout << (number > 0 ? "Positive" : "Non-positive") << std::endl;

    // Equivalent If-Else Statement
    if (number > 0) {
        std::cout << "Positive" << std::endl;
    } else {
        std::cout << "Non-positive" << std::endl;
    }

    return 0;
}
```