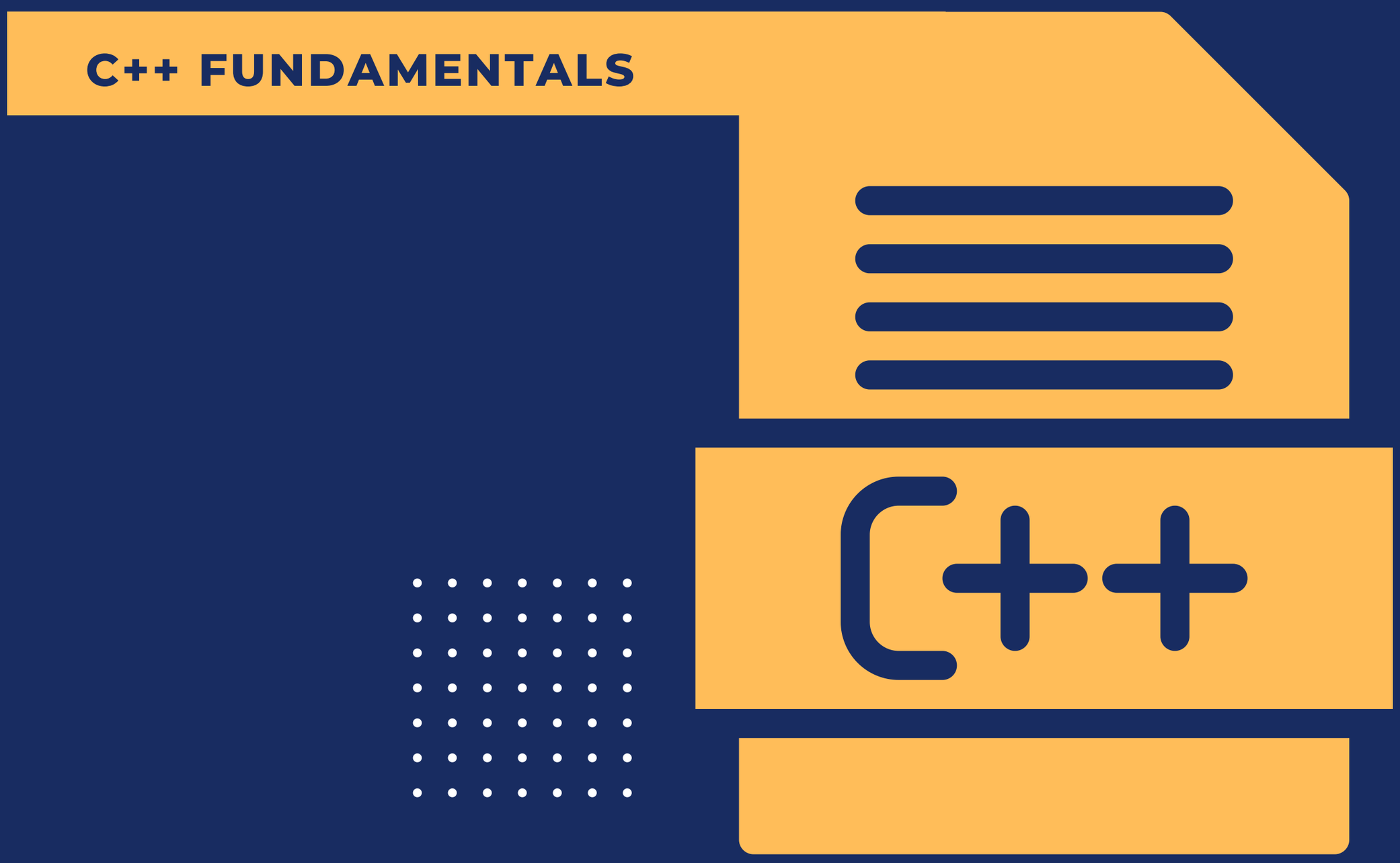
INTRODUCTION TO C++

Introduction to C++ is a clear, beginner-friendly textbook that guides you from your first program through core programming fundamentals, including variables, control structures, loops, functions, arrays, structures, and an introduction to object-oriented programming.



www.meerkat.pub

DR. KEVIN ROARK | 2026 EDITION

Book Overview

Introduction to C++ is an approachable textbook ideal for first-time programmers embarking on the field of computer science. Crafted with clarity, a thoughtful structure, and practical examples, this publication guides readers through the fundamental concepts of programming with C++, one of the most versatile and widely used programming languages in educational and industrial settings worldwide.

Suitable for introductory courses, this book systematically introduces the essential building blocks of programming: variables, control structures, loops, functions, arrays, structures, and an introductory overview of object-oriented programming. Each chapter is carefully sequenced to build on the previous one, facilitating the development of solid foundational knowledge and enabling students to progressively and confidently acquire their skills.

This digital textbook is complemented by more than 50 **professionally produced video tutorials** that explicitly demonstrate C++ syntax, clarify complex topics, and guide learners through comprehensive coding examples. These visual resources serve to enhance understanding and make programming more accessible to individuals from diverse backgrounds.

Throughout the course, learners can utilize practical exercises, including creative programming laboratories, thematic assignments, and engaging projects, designed to stimulate creativity and develop critical skills. Whether constructing a student information management system, creating a game-inspired application, or designing a menu-driven utility, learners not only refine their coding abilities but also cultivate confidence in problem-solving, approaching challenges with motivation and enthusiasm.

Key Features of *Introduction to C++*

Beginner-Friendly, Easy-to-Read Format. This textbook is thoughtfully written with clarity and purpose, making it perfect for students who are just starting out and have no previous programming experience. The book breaks down complex topics into simple, manageable sections to help you learn with confidence.

Includes Over 50 Professional Video Walkthroughs. Each major topic is paired with engaging video demonstrations that guide students through coding examples, syntax explanations, and best practices. This approach helps turn abstract ideas into a clear, tangible understanding.

Flexible Delivery Options. Designed to support multiple teaching formats:

- Self-paced learning
- Instructor-facilitated courses
- Face-to-face classroom instruction
- Synchronous or asynchronous online delivery

Hands-On Labs and Creative Projects. Each module features themed lab assignments designed to reinforce concepts through engaging practice and creativity. This approach helps students connect with what they've learned in a meaningful and personal way.

Designed specifically for college students and adult learners. The book is thoughtfully organized to align with college-level learning outcomes. It's a wonderful choice for academic programs and workforce training, providing support and relevance every step of the way.

Available online and as an iOS/Android app. Students can easily access the textbook on any device, whether via the web or via iOS or Android apps. The platform remembers where they paused and visually shows their progress through each module, making learning more seamless and encouraging.

Who This Book Is For

- **First-Time Programmers -** Whether you're just starting college or exploring programming for the first time, this book assumes no prior experience and builds from the ground up.
- **Career Switchers and Industry Trainees -** Ideal for workforce training programs or individuals transitioning into tech, the book offers clear explanations, hands-on practice, and real-world context.
- **High School Dual Credit and College-Level Courses -** Designed to meet the needs of dual-credit programs and introductory college courses, supporting both academic rigor and student accessibility.

Course/Textbook Overview

Module 1 Overview:

In this Module, you'll start your exciting journey into computer programming with the C++ language. We'll begin with the fundamentals - what programming is, why it's important, and how C++ fits into the tech world. One of your first fun milestones will be writing the classic "Hello, World!" program, giving you a friendly introduction to compiling and running code. From there, you'll discover how to create variables, understand data types, and start writing simple programs that do exactly what you want.

As you get more comfortable with the basics, you'll learn how to communicate with the user by displaying output using `cout`. You'll discover how to make your output neat and professional with formatting tools like `setw`, `setfill`, and `setprecision`. Then, you'll flip the script and learn how to take input from the user using tools like `cin`, `getline()`, and `cin.get()`. Along the way, you'll tackle common input issues and pick up essential tricks like using `cin.ignore()` and `cin.clear()` to manage the input buffer and keep your programs running smoothly.

Each concept builds on the last, helping you develop the problem-solving and logical thinking skills you'll need as we progress to more advanced topics, such as decision-making, loops, functions, and eventually, working with objects.

By the end of this module, you will be able to:

1. Explain the basic structure of a C++ program and write simple programs using appropriate syntax and comments.
2. Declare and use variables of different data types to store and manipulate values within a program.
3. Display output using `cout` and apply formatting tools like `setw`, `setfill`, and `setprecision` to improve program readability.
4. Receive input from users using `cin`, `getline()`, and `cin.get()`, and understand how to manage the input buffer with `cin.ignore()` and `cin.clear()`.
5. To build more robust programs, identify and resolve common input-related issues, including input mismatches and skipped lines.
6. Design and implement interactive console applications that collect and process user input to generate meaningful output.

Module 2 Overview

In this module, you will elevate your programming skills by building more innovative, more interactive programs. So far, you've written simple code that collects input and displays output. Now, you'll learn how to create programs that can make decisions, handle different data types, and even store and retrieve information for later use.

You will explore type casting, a technique that enables you to convert between data types, ensuring your calculations remain accurate and flexible. You'll also make use of predefined C++ functions to perform tasks like calculating powers, manipulating characters, and formatting output - all with just a single line of code.

As you progress, you'll learn how to work with files, enabling your programs to save results and load data. This opens the door to creating more useful and real-world applications. You'll also discover how to use if, else if, and else statements to guide your program's behavior based on user input or conditions.

In addition, you'll practice handling input errors so your programs can respond to unexpected input without crashing. Finally, you'll learn to build menu-driven programs using switch statements, giving users clear and organized choices.

By the end of this module, you will be able to write programs that are more dynamic, flexible, and ready to solve real-world problems. You'll gain the tools to guide users, manage data, and make your code come alive.

By the end of this module, you will be able to:

- Convert between data types using both implicit and explicit type casting to support accurate calculations and data handling.
- Use predefined C++ functions such as `pow()`, `sqrt()`, `abs()`, `toupper()`, `setprecision()`, and `fixed` to perform mathematical operations, manipulate characters, and control output formatting.
- Create and write data to external files using `ofstream`, and format output for clear documentation and reporting.
- Read input from external files using `ifstream`, and process that data within a program to generate meaningful results.
- Use conditional structures such as `if`, `else if`, and `else` to guide program flow based on user input or data conditions.
- Implement **`switch`** statements to handle multi-option decisions in a clean and structured way.
- Handle common input errors using tools like `cin.fail()`, `cin.clear()`, and `cin.ignore()` to make your programs more robust and user-friendly.
- Apply formatting tools to control numeric output display using `fixed`, `setprecision()`, `showpoint`, and `setw()`for cleaner and more professional program output.

Module 3 Overview

In this module, you will enter the world of loops - a powerful tool that allows your programs to repeat actions automatically. Instead of writing the same code over and over, you'll learn how to use loops to process repeated input, perform automated calculations, draw patterns, build simple game-like systems, and create interactive menus.

You will explore the three main types of loops in C++: while, for, and do-while. Each has its purpose, and you'll learn when and why to use each one. Along the way, you'll strengthen

your understanding of important programming concepts like input validation, counters, random number generation, decision-making, and output formatting.

By the end of this module, you will be writing programs that are smarter, more efficient, and far more interactive. Loops are one of the most essential tools in programming, and once you learn to control repetition with logic and precision, you'll unlock a whole new level of problem-solving power.

By the end of this module, you will be able to:

- Explain the purpose of loops and when repetitive processing is needed in a program.
- Use `while` loops to repeat actions based on a condition.
- Use `for` loops to repeat actions a specific number of times with a counter.
- Use `do-while` loops to perform actions at least once before checking a condition.
- Use nested loops to solve multidimensional problems, such as constructing tables and patterns.
- Design menu-driven programs that allow users to make repeated choices until they exit.
- Validate user input inside loops to ensure correct and safe data entry.
- Apply logical operators (`&&`, `||`, `!`) within loops and conditions to build flexible and powerful logic.
- Control loop behavior using `break` and `continue` statements appropriately.
- Use formatting manipulators (`setw`, `setprecision`, `setfill`) to create clean, readable program output.
- Solve real-world problems by combining loops, conditional structures, and arithmetic operations.

Module 4 Overview

In this module, you'll make exciting progress as a programmer by learning how to craft and utilize your own functions in C++. So far, most of your code has been within the main() function, which is appropriate for small projects. But as your projects grow, managing everything in one place can become somewhat overwhelming. That's where functions come in - they let you split your code into smaller, clear sections, each focusing on one specific task. This not only makes your code easier to read and debug but also more reusable, helping you become even more confident in your coding journey!

Throughout this module, you'll learn how to define functions, pass information into them using parameters, and return values to the part of the program that needs them. You'll also explore more advanced features like default arguments and function overloading, and learn how to organize your projects into separate header and implementation files, just like professional developers do. By the end of this module, you'll be writing programs that are better structured, easier to maintain, and ready to scale as your skills grow.

By the end of this module, you will be able to:

- Explain the purpose of functions and how they improve code organization and readability.
- Define and call functions that perform specific tasks using proper syntax.
- Use parameters to pass information into functions and return values to pass results back.
- Apply function overloading and default arguments to make functions more flexible and reusable.
- Distinguish between pass-by-value and pass-by-reference and know when to use each.
- Demonstrate modular program design by separating code into header and implementation files.
- Write programs that use multiple functions to manage logic in a clean, maintainable structure.

Module 5 Overview

In this module, you will learn how to store and manage multiple pieces of related data using one of the most essential tools in programming: the array. Up until now, you've used variables to hold single values like a name or a score. However, many real-world problems involve handling *lists* of information, and arrays make this possible by allowing you to store multiple values under a single variable name.

You will begin by working with one-dimensional arrays, which are well-suited for storing a series of numbers or strings. Then, you'll explore parallel arrays, which allow you to keep track of connected data, like a student's name and their exam score, side by side. You'll use loops to efficiently input, display, and process these arrays, and practice everyday tasks such as calculating totals, averages, and identifying the highest or lowest value.

As you build your skills, you'll also learn how to pass arrays to functions, making your code cleaner and easier to reuse. You will implement basic search algorithms to locate specific values in a list and use sorting algorithms, such as bubble sort and selection sort, to organize the data in a meaningful way.

By the end of this module, you will be able to create programs that work with groups of data, process and analyze that information, and present useful results to users. Arrays are a key building block in programming, and this module will prepare you to use them with confidence.

By the end of this module, you will be able to:

- Define what an array is and explain when and why arrays are used in programming.
- Declare, initialize, and populate one-dimensional arrays using both direct assignment and user input.
- Use loops to access, display, and process elements in an array.
- Create and manage parallel arrays to store and process related data, such as names and scores or products and prices.
- Write functions that accept arrays as arguments, thereby enabling modularization and code reuse.

- Implement search algorithms, such as linear and binary search, to locate values in an array.
- Implement sorting algorithms, such as bubble sort and selection sort, to sort array data.
- Track totals, averages, maximums, and minimums using arrays and loop-based logic.
- Avoid common pitfalls such as accessing out-of-bounds indices.

Module 6 Overview

In this module, you'll take a big step forward in building real-world C++ programs by learning how to organize and manage collections of data using vectors and structures. These two tools are really important for creating clean, scalable, and meaningful programs. You will begin with vectors, which are flexible, dynamic lists that allow you to store and manage multiple items, such as scores, names, or inventory data. You'll learn how to add, remove, and access elements in a vector, and how to pass vectors to functions. You will also examine how vector performance varies with memory usage.

Next, you will explore structures, which help you group related data into a single custom type. For example, instead of handling a name, ID, and GPA as three separate variables, you can define a Student structure and treat them as one cohesive unit. You'll learn how to define structures, store them in vectors, and pass them to functions for processing and display.

By the end of this module, you will be able to:

- Declare and initialize vectors to store dynamic lists of values.
- Use built-in vector functions such as `push_back()`, `pop_back()`, `clear()`, `size()`, and `resize()` to manipulate vector contents.
- Pass vectors to functions by value, reference, and `const` reference, and understand the implications of each.
- Access and modify vector elements using indexing and loops.
- Define custom `struct` types to group related data into meaningful records.
- Create and use vectors of structures to manage lists of structured data.
- Write functions that accept and return structures or vectors of structures.
- Display vector and structure data using formatted output with `iomanip`.
- Organize code into multiple files, using header (.h) and implementation (.cpp) files, to improve modularity.
- Develop interactive programs that utilize vectors and structures in real-world applications.

Module 7 Overview

In this module, you will take one of the most important steps in your journey as a programmer: learning about classes and objects. This is your first introduction into object-oriented programming (OOP), a powerful way to design programs that more closely reflect the real world.

So far, you've built programs using variables, functions, arrays, and structs to complete tasks step by step. While this procedural approach works well for small programs, it can become challenging to

manage as your code grows. That's where OOP comes in. With classes and objects, you'll learn how to bundle data and the actions that go with it into reusable building blocks.

In this module, you will walk through a real-world example step by step: creating a Car class. You'll see how objects can store their own data and know how to perform their own actions, like displaying details or updating information. This hands-on approach will help you connect the concept of classes to working code.

By the end of this module, you will understand not just the syntax of classes but also the mindset of object-oriented thinking. You'll begin writing cleaner, smarter, and more organized programs that prepare you for more advanced topics.

By the end of these sections, you will be able to:

- Explain the purpose of classes and how they are used in object-oriented programming
- Define a class that includes both data members (attributes) and member functions (methods)
- Understand and use access specifiers (`private`, `public`) to control access to class members
- Create objects from a class and access their functions and data using dot notation
- Write and use constructors to initialize objects, including both default and parameterized constructors
- Use setters and getters (mutators and accessors) to modify and retrieve private data safely
- Understand the principle of encapsulation and how it protects data and simplifies design
- Organize class code across header (.h) and implementation (.cpp) files using best practices.
- Begin thinking about software design in terms of objects and their responsibilities.

Module 8 Overview

In this module, you will bring together many of the programming skills you've learned throughout the course and apply them to more advanced, real-world techniques. You will focus on two key features in C++ that help make your code more organized, scalable, and professional: vectors of objects and namespaces.

You've already used vectors to store lists of basic data types like integers and strings. Now, you'll take that concept further by storing entire objects inside a vector. This will enable you to manage more complex data structures, such as groups of employees, game characters, or inventory items, using object-oriented programming principles.

You'll also be introduced to namespaces, a tool designed to help you avoid naming conflicts and keep your code well-structured. Namespaces are especially helpful when your projects grow larger or when you're working with code from multiple sources or teammates.

By the end of this module, you will be able to:

- Understand how to define and use classes to create objects in C++
- Store and manage objects using vectors

- Access and manipulate object data stored within a vector
- Write functions that process vectors of objects
- Organize code using header and implementation files for class definitions
- Explain the purpose of namespaces and how they help prevent naming conflicts
- Use the std namespace and create custom namespaces in your own programs
- Build modular, object-based programs that apply the concepts learned throughout the course