# ARIZONA STATE UNIVERSITY (ASU) CSE100 PRINCIPLES OF PROGRAMMING WITH C++ MIDTERM 1 PRACTICE EXAM (SAMPLE) STUDY GUIDE


## SAMPLE STUDY GUIDE WITH LIMITED QUESTIONS

**VISIT US ONLINE FOR THE FULL VERSION STUDY GUIDE**

https://asu-cse100midterm1.examzify.com

# Table of Contents

# Introduction

Preparing for a certification exam can feel overwhelming, but with the right tools, it becomes an opportunity to build confidence, sharpen your skills, and move one step closer to your goals. At Examzify, we believe that effective exam preparation isn't just about memorization, it's about understanding the material, identifying knowledge gaps, and building the test-taking strategies that lead to success.

This guide was designed to help you do exactly that.

Whether you're preparing for a licensing exam, professional certification, or entry-level qualification, this book offers structured practice to reinforce key concepts. You'll find a wide range of multiple-choice questions, each followed by clear explanations to help you understand not just the right answer, but why it's correct.

The content in this guide is based on real-world exam objectives and aligned with the types of questions and topics commonly found on official tests. It's ideal for learners who want to:

- Practice answering questions under realistic conditions,
- Improve accuracy and speed,
- Review explanations to strengthen weak areas, and
- Approach the exam with greater confidence.

We recommend using this book not as a stand-alone study tool, but alongside other resources like flashcards, textbooks, or hands-on training. For best results, we recommend working through each question, reflecting on the explanation provided, and revisiting the topics that challenge you most.

Remember: successful test preparation isn't about getting every question right the first time, it's about learning from your mistakes and improving over time. Stay focused, trust the process, and know that every page you turn brings you closer to success.

Let's begin.

# How to Use This Guide

This guide is designed to help you study more effectively and approach your exam with confidence. Whether you're reviewing for the first time or doing a final refresh, here's how to get the most out of your Examzify study guide:

### 1. Start with a Diagnostic Review

Skim through the questions to get a sense of what you know and what you need to focus on. Your goal is to identify knowledge gaps early.

### 2. Study in Short, Focused Sessions

Break your study time into manageable blocks (e.g. 30 – 45 minutes). Review a handful of questions, reflect on the explanations.

### 3. Learn from the Explanations

After answering a question, always read the explanation, even if you got it right. It reinforces key points, corrects misunderstandings, and teaches subtle distinctions between similar answers.

### 4. Track Your Progress

Use bookmarks or notes (if reading digitally) to mark difficult questions. Revisit these regularly and track improvements over time.

### 5. Simulate the Real Exam

Once you're comfortable, try taking a full set of questions without pausing. Set a timer and simulate test-day conditions to build confidence and time management skills.

### 6. Repeat and Review

Don't just study once, repetition builds retention. Re-attempt questions after a few days and revisit explanations to reinforce learning. Pair this guide with other Examzify tools like flashcards, and digital practice tests to strengthen your preparation across formats.

There's no single right way to study, but consistent, thoughtful effort always wins. Use this guide flexibly, adapt the tips above to fit your pace and learning style. You've got this!

# Questions

**1. How are comments written in C++ code?**

A. Using `#` for all comments

B. Using `//` for single-line and `/* comment */` for multi-line

C. Using `--` for single-line and `/* comment */` for multi-line

D. Using `//` for multi-line and `#` for single-line

**2. What are operators used for in programming languages?**

A. To perform operations on memory

B. To define data types

C. To perform operations on data

D. To manage user input

**3. What is the function of the 'main' function in a C++ program?**

A. It defines a user interface for the program

B. It stores global variables

C. It is the starting point of program execution

D. It is used for error handling

**4. How many bits are there in a byte?**

A. 4

B. 8

C. 16

D. 32

**5. What does the following code do: `int x = 5; x += 3;`?**

A. It sets x to 3

B. It adds 5 and 3, setting x to 8

C. It adds 3 to x, so x becomes 8

D. It subtracts 3 from x, so x becomes 2

**6. What does the `throw` keyword do in C++?**

A. Signals the occurrence of an exception

B. Defines a new variable type

C. Indicates the end of a function

D. Initiates a loop structure

**7. How is the scope of a variable defined in C++?**

A. By its datatype

B. By the function it is declared in

C. By where it can be accessed within the program

D. By its name

**8. What does the keyword `static` indicate in C++?**

A. A variable's value is reset after each function call

B. A class member is shared among all instances

C. A function cannot return a value

D. A variable is visible only within its function

**9. What keyword is used to define a new function in C++?**

A. function

B. method

C. def

D. void

**10. How do you correctly define a member function outside of a class in C++?**

A. FunctionName ClassName::ReturnType(parameters) { // function body }

B. ClassName::FunctionName(parameters) { // function body }

C. ReturnType ClassName.FunctionName(parameters) { // function body }

D. ReturnType ::ClassName FunctionName(parameters) { // function body }

# Answers

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

# Explanations

## 1. How are comments written in C++ code?

A. Using `#` for all comments

B. Using `//` for single-line and `/* comment */` for multi-line

C. Using `--` for single-line and `/* comment */` for multi-line

D. Using `//` for multi-line and `#` for single-line

*In C++, comments are written using two primary formats: single-line comments and multi-line comments. Single-line comments begin with `//` and continue until the end of that line, making it easy to add brief notes without disrupting the flow of the code. Multi-line comments are enclosed between `/*` and `*/`, allowing for longer descriptions or notes that may span multiple lines. This method of commenting is essential for code documentation, as it provides clarity for anyone reading the code later, whether that be the original author or someone else. By utilizing both single-line and multi-line comments effectively, programmers can improve the readability and maintainability of their code. Understanding the correct syntax to avoid syntax errors is crucial; for example, failing to close a multi-line comment with `*/` could lead to compilation errors. Other options presented have incorrect comment syntax as per the C++ standards, such as using `#` or `--`, which do not serve the purpose of commenting in C++.*

## 2. What are operators used for in programming languages?

A. To perform operations on memory

B. To define data types

C. To perform operations on data

D. To manage user input

*Operators in programming languages are essential tools that allow developers to perform various operations on data. They are used to manipulate values and variables by carrying out specific tasks, such as arithmetic calculations, logical comparisons, and bit manipulations. When you think of common operations like addition, subtraction, multiplication, or division, these are all carried out using operators. For example, in C++, using the `+` operator can add two numbers together, while the `==` operator can compare two values for equality. This functionality is fundamental to programming, as it enables the creation of expressions that determine the flow of the program and the processing of data. By performing operations on data, operators facilitate the implementation of algorithms, the evaluation of conditions, and the execution of various computational tasks, making them integral to any programming language's functionality. The ability to manipulate data effectively through operators is crucial in developing logic and functionality within programs.*

## 3. What is the function of the 'main' function in a C++ program?

A. It defines a user interface for the program

B. It stores global variables

C. It is the starting point of program execution

D. It is used for error handling

*The 'main' function in a C++ program serves as the entry point for execution. When a C++ program is run, the operating system looks for the 'main' function to begin executing the code. This function is where the program starts, and it is essential for establishing the program's flow. The 'main' function can return an integer value, typically indicating the success or failure of the program to the operating system. In this sense, it acts as a point of initiation and termination for program execution. The other options do not accurately describe the primary function of 'main'. While user interfaces can be created within a C++ program, the 'main' function itself does not define them. Global variables can be declared outside of 'main,' but 'main' is not specifically responsible for storing them. Additionally, while error handling can occur within 'main' or in functions called by it, this is not its primary purpose.*

## 4. How many bits are there in a byte?

A. 4

B. 8

C. 16

D. 32

*A byte consists of 8 bits. This is a fundamental concept in computer science and programming. Bits, the smallest unit of data in computing, can represent a binary value of either 0 or 1. When combined, 8 bits form a byte, which can represent 256 different values (ranging from 0 to 255). This grouping is essential for encoding characters and data formats in computer systems, influencing how information is stored and processed. The other options represent different sizes of data or number of bits: 4 bits is a nibble, 16 bits form a word in many systems, and 32 bits typically correspond to a double word. However, none of these represent a byte, as the definition of a byte is specifically established as 8 bits.*

## 5. What does the following code do: `int x = 5; x += 3;`?

A. It sets x to 3

B. It adds 5 and 3, setting x to 8

C. It adds 3 to x, so x becomes 8

D. It subtracts 3 from x, so x becomes 2

*The code `int x = 5; x += 3;` performs an operation where the initial value of `x` is set to 5. The statement `x += 3;` is shorthand for adding 3 to the current value of `x` and then assigning the result back to `x`. Initially, `x` is 5. When you execute `x += 3;`, it computes 5 + 3, resulting in 8, and then updates `x` to hold this new value. Thus, after the execution of the code, the value of `x` becomes 8, which is why the correct answer is that it adds 3 to `x`, so `x` becomes 8. The other options imply incorrect operations: setting `x` to 3 or subtracting values, neither of which accurately represents the operation being performed in the code snippet.*

## 6. What does the `throw` keyword do in C++?

A. Signals the occurrence of an exception

B. Defines a new variable type

C. Indicates the end of a function

D. Initiates a loop structure

*The `throw` keyword in C++ is used to signal that an exception has occurred. When an error or unexpected behavior is detected in the program, the `throw` statement allows you to generate an exception object that represents the error condition. This exception can then be caught and handled by surrounding code that is set up with a `try` block and a corresponding `catch` block. This mechanism is part of C++'s exception handling framework, providing a way to separate error handling logic from regular code. By using `throw`, you can effectively manage error conditions and allow a program to continue running or to terminate gracefully by managing exceptional situations in a controlled manner. In contrast, the other options refer to different functionalities in C++. For instance, defining a new variable type is accomplished through type declaration, not the `throw` keyword. Similarly, indicating the end of a function is done using a return statement, while initiating a loop structure requires keywords like `for`, `while`, or `do`. Understanding the specific role of `throw` helps clarify how C++ manages exceptions and why it is a critical part of robust programming practices.*

## 7. How is the scope of a variable defined in C++?

A. By its datatype

B. By the function it is declared in

C. By where it can be accessed within the program

D. By its name

*In C++, the scope of a variable refers to the region of the program where the variable is valid and can be accessed. This means that the determining factor for a variable's scope is based on where it can be referenced and used throughout the code. For example, a variable declared within a function has a local scope and can only be accessed within that function. Conversely, a variable declared outside of any function or class has a global scope and can be accessed from any part of the program that follows its declaration. This concept is crucial for understanding how to manage and organize code effectively, as it helps prevent naming conflicts and ensures that variables are used in the intended context. The other options do not accurately capture the definition of a variable's scope in C++. While the datatype determines the type of data the variable can hold and the function declaration might imply a local scope, these aspects do not define how accessibility is controlled. The name of the variable is simply an identifier and does not relate to the scope itself. Thus, focusing on where the variable can be accessed is the key aspect of understanding variable scope in C++.*

## 8. What does the keyword `static` indicate in C++?

A. A variable's value is reset after each function call

B. A class member is shared among all instances

C. A function cannot return a value

D. A variable is visible only within its function

*The keyword `static` in C++ serves several purposes, one of which is to indicate that a class member is shared among all instances of that class. When a class member is declared as static, it means that there is only one copy of that member, regardless of how many objects of the class are created. Thus, all instances of the class can access and modify the same static member, leading to shared state across objects. This is particularly useful for maintaining data that should be consistent or shared across all instances, such as a counter tracking the number of objects created or a configuration setting applicable to all instances. The other choices refer to different concepts in C++. A variable's value being reset after each function call relates to automatic storage duration, which is not what static indicates. The concept of a function not being able to return a value pertains to a specific declaration of a void function, unrelated to the static keyword. Lastly, a variable being visible only within its function pertains to scope rules, specifically for local variables, which again does not correlate with the use of the static keyword.*

## 9. What keyword is used to define a new function in C++?

A. function

B. method

C. def

D. void

*The keyword used to define a new function in C++ is "void" when you are indicating that the function does not return a value. In C++, when defining a function, you need to specify the return type of the function, which indicates what kind of data the function is expected to return to the caller. Using "void" as the return type signifies that the function will perform its tasks but will not return any value. For example: ```cpp void myFunction() { // Code for the function } ``` In this case, "myFunction" is a function that performs some operations but does not give any value back at the end. If you wanted to define a function that does return a value, you would use an appropriate data type (like int, double, float, etc.) instead of "void." The other terms, like "function" and "method," are not keywords in C++. "def" is used in Python for defining functions, and those terms may refer to concepts in different programming languages or be informal terms used to describe functions. However, in the C++ context, "void" and other data types are what you use to define functions correctly.*

## 10. How do you correctly define a member function outside of a class in C++?

A. FunctionName ClassName::ReturnType(parameters) { // function body }

B. ClassName::FunctionName(parameters) { // function body }

C. ReturnType ClassName.FunctionName(parameters) { // function body }

D. ReturnType ::ClassName FunctionName(parameters) { // function body }

*To correctly define a member function outside of a class in C++, the proper syntax requires using the scope resolution operator `::` to associate the function definition with the specific class it belongs to. This means that the function definition starts with the return type, followed by the class name and the scope resolution operator, leading into the function name and its parameters within parentheses. This method enables the compiler to understand that the function being defined is not a standalone function, but rather a member of the specified class. This is essential for properly organizing code in object-oriented programming and ensuring that the function has access to the members and methods of the class. The other options lack correct structure as follows: one incorrectly places the function name before the class name, another uses a dot notation instead of the scope resolution operator, and one misplaced the return type and class name, which does not conform to C++ syntax.*

# Next Steps

Congratulations on reaching the final section of this guide. You've taken a meaningful step toward passing your certification exam and advancing your career.

As you continue preparing, remember that consistent practice, review, and self-reflection are key to success. Make time to revisit difficult topics, simulate exam conditions, and track your progress along the way.

If you need help, have suggestions, or want to share feedback, we'd love to hear from you. Reach out to our team at hello@examzify.com.

Or visit your dedicated course page for more study tools and resources:

https://asu-cse100midterm1.examzify.com

We wish you the very best on your exam journey. You've got this!