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

Copyright © 2026 by Examzify - A Kaluba Technologies Inc. product.

ALL RIGHTS RESERVED.

No part of this book may be reproduced or transferred in any form or by any means, graphic, electronic, or mechanical, including photocopying, recording, web distribution, taping, or by any information storage retrieval system, without the written permission of the author.

Notice: Examzify makes every reasonable effort to obtain accurate, complete, and timely information about this product from reliable sources.

SAMPLE

Table of Contents

Copyright 1

Table of Contents 2

Introduction 3

How to Use This Guide 4

Questions 5

Answers 8

Explanations 10

Next Steps 16

SAMPLE

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.

SAMPLE

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!

SAMPLE

Questions

SAMPLE

1. What is the difference between public and private access specifiers in a class?

A. Public members cannot be accessed outside the class

B. Private members are visible to derived classes only

C. Public members can be accessed from outside the class

D. Private members can be accessed publically

2. What keyword is used to create a new object in C++?

A. create

B. new

C. init

D. allocate

3. What happens to variables defined as `static` within a function when the function is called multiple times?

A. Their value resets each time

B. They are destroyed and recreated

C. They retain their value between calls

D. They become global variables

4. 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

5. What are the three main cycles that the CPU goes through?

A. Fetch, decode, display

B. Fetch, execute, save

C. Fetch, decode, execute

D. Retrieve, process, store

SAMPLE

6. What is the difference between '==' and '=' in C++?

A. '==' is for assignment and '=' is for comparison

B. '==' checks value equality and '=' is for variable assignment

C. '==' has higher precedence than '='

D. '==' is used only in conditional statements

7. What does passing by value mean in C++?

A. Passing a reference to the original variable

B. Passing a copy of the variable to a function

C. Passing the original variable unchanged

D. Passing multiple values at once

8. What is 'compilation' in the context of C++ programming?

A. The process of executing the program

B. The process of translating C++ source code to machine code

C. The process of debugging the source code

D. The process of writing source code in C++

9. In C++, what is the purpose of the 'using namespace std;' declaration?

A. It allows access to standard library functions

B. It restricts the use of standard library functions

C. It defines a new namespace

D. It includes a library

10. What will the following code: int x = 10; x++; return x; return 5; yield?

A. 5

B. 10

C. 11

D. Compilation error

SAMPLE

Answers

SAMPLE

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

SAMPLE

Explanations

SAMPLE

1. What is the difference between public and private access specifiers in a class?

A. Public members cannot be accessed outside the class

B. Private members are visible to derived classes only

C. Public members can be accessed from outside the class

D. Private members can be accessed publically

In C++, access specifiers are essential for encapsulating class members and controlling their visibility and accessibility. Public access specifiers allow members of a class to be accessible from outside the class itself. This means that any function or code that has visibility of a class object can access and manipulate its public members directly. For instance, if you have a class with a public variable or method, any instance of that class can utilize those members freely, making it easier to interact with the class's data and functionalities from external code. In contrast, private members are not accessible from outside the class, which means they can only be accessed from within the class itself or by friend functions. This encapsulation protects the internal state of the object and enforces controlled access, which is a fundamental aspect of object-oriented programming. The incorrect options address misunderstandings regarding how access specifiers work. Members marked as private are not intended for access outside the class. Similarly, private members are not exclusively visible to derived classes; they are typically inaccessible to anything that is not part of the defining class itself, unless specific methods or friend declarations are used. Lastly, private members cannot be accessed publicly; doing so contradicts the purpose of having private access in the first place.

2. What keyword is used to create a new object in C++?

A. create

B. new

C. init

D. allocate

In C++, the keyword used to create a new object is "new." When you use "new," it dynamically allocates memory for an object on the heap and returns a pointer to that memory. This is essential in scenarios where you need objects that outlive the scope in which they were created or when you don't know the number of objects you will need at compile time. Using "new" allows for the creation of complex data structures and the management of memory, which is crucial in many programming situations. For example, writing `MyClass* obj = new MyClass();` creates an instance of `MyClass` and returns a pointer to it, enabling you to use that object throughout its lifetime until it is explicitly deallocated with "delete." Other options like "create," "init," and "allocate" are not keywords in C++. They may represent concepts related to object creation or memory management, but they do not function as reserved words in the C++ language used for creating objects. While you may encounter similar terms in other contexts or programming languages, only "new" has the specific functionality required in C++ for dynamic object creation.

SAMPLE

3. What happens to variables defined as `static` within a function when the function is called multiple times?

A. Their value resets each time

B. They are destroyed and recreated

C. They retain their value between calls

D. They become global variables

When a variable is defined as `static` within a function, it is initialized only once, and its value persists for the lifetime of the program, even between multiple calls to that function. This means that each time the function is called, the `static` variable does not reset to its initial state; rather, it retains the value from the previous call. The storage duration of a `static` variable is static, meaning that it exists for the duration of the program and is not recreated or destroyed with each function call. This behavior allows for the accumulation of values or state retention across function invocations, which can be particularly useful for scenarios where you want to keep track of data without exposing it outside the function's scope. Thus, when the function is invoked again, the `static` variable continues with its last stored value, reinforcing its purpose of maintaining state information over time.

4. 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++.

SAMPLE

5. What are the three main cycles that the CPU goes through?

A. Fetch, decode, display

B. Fetch, execute, save

C. Fetch, decode, execute

D. Retrieve, process, store

The correct answer is based on the fundamental operations of a CPU during instruction execution, known as the fetch-decode-execute cycle. The fetch phase involves the CPU retrieving an instruction from memory, specifically from the address indicated by the program counter. This instruction is then loaded into the instruction register. In the decode phase, the CPU interprets the fetched instruction, determining what operation is to be performed and identifying the operands (the data or addresses involved in the operation). This step often involves translating the binary instructions into a format that the CPU can understand. Finally, in the execute phase, the CPU performs the operation as instructed, which could involve arithmetic or logical calculations, moving data from one place to another, or interacting with input/output devices. This cycle is crucial for the CPU's functionality, as it defines how instructions are processed sequentially. The other options present different combinations of actions that do not accurately represent the established cycle of instruction processing in a CPU. For example, while 'retrieve, process, store' might suggest general operations, it does not explicitly capture the decoding step and its importance in interpreting instructions.

6. What is the difference between '==' and '=' in C++?

A. '==' is for assignment and '=' is for comparison

B. '==' checks value equality and '=' is for variable assignment

C. '==' has higher precedence than '='

D. '==' is used only in conditional statements

In C++, the symbol '==' is used to check for value equality between two operands. When you use '==', you are asking whether the values on the left and right sides of the operator are the same. It's a comparison operator that returns a boolean result: true if the values are equal and false if they are not. For example, if you have two variables, x and y, and you use the expression x == y, it evaluates to true if both x and y hold the same value. On the other hand, the '=' symbol is the assignment operator. It is used to assign a value to a variable. For instance, if you write x = 5, you are stating that x should now hold the value 5. The assignment operator does not compare values; instead, it alters the state of the variable involved. Understanding this distinction is crucial for writing correct and logical C++ code, as mixing these two operators can lead to errors in program logic. Thus, the choice that states '==' checks value equality and '=' is for variable assignment accurately captures their roles in the C++ programming language.

SAMPLE

7. What does passing by value mean in C++?

A. Passing a reference to the original variable

B. Passing a copy of the variable to a function

C. Passing the original variable unchanged

D. Passing multiple values at once

Passing by value in C++ means that a copy of the variable is passed to a function. When a function takes parameters by value, it receives a duplicate of each argument's value. This means that any changes made to the parameter within the function do not affect the original variable used in the function call. The original variable remains unchanged because the function operates on the copy, not on the reference to the original data. This approach is useful when you want to ensure that the original data is protected from modifications within the function. For example, if you were computing a result and did not want to alter any input variables, passing by value guarantees that the original data remains intact. In contrast, passing a reference to the original variable would allow modifications to affect the original variable directly, which is not the case when using pass-by-value. As for passing multiple values at once, this typically involves passing other data structures like arrays or objects, which does not specifically address the concept of passing by value.

8. What is 'compilation' in the context of C++ programming?

A. The process of executing the program

B. The process of translating C++ source code to machine code

C. The process of debugging the source code

D. The process of writing source code in C++

Compilation in the context of C++ programming refers to the process of translating the human-readable C++ source code into machine code, which is a binary format that the computer's processor can execute. This process is crucial because computers do not understand high-level programming languages directly; they require instructions in a format that their hardware can process. During compilation, the C++ compiler performs several tasks, including lexical analysis, syntax analysis, semantic analysis, optimization, and code generation. The output of this compilation is typically an executable file or object file that can be run on the machine. The other options describe different aspects of software development but do not pertain to compilation. Executing the program refers to running the compiled code, debugging involves finding and resolving errors within the source code prior to compilation, and writing source code is the initial step in the programming process that occurs before any compilation takes place. Thus, the focus on translating source code into machine code clearly defines compilation and demonstrates why this is the correct choice.

SAMPLE

9. In C++, what is the purpose of the 'using namespace std;' declaration?

A. It allows access to standard library functions

B. It restricts the use of standard library functions

C. It defines a new namespace

D. It includes a library

The declaration 'using namespace std;' is pivotal in C++ as it allows the programmer to directly access all the identifiers (such as functions, variables, and classes) defined in the standard namespace without needing to prefix them with 'std::'. For example, if you want to use the 'cout' stream from the standard library, without this declaration, you would need to write 'std::cout'. By including 'using namespace std;', you can simply use 'cout' directly. This is particularly useful for simplifying code, especially in educational contexts or small projects where the potential for naming conflicts is minimal. It makes the code cleaner and easier to read, as it reduces the repetitiveness of writing 'std::' before every standard library call. The other options do not accurately reflect the purpose of this declaration. It does not restrict the use of functions, nor does it define a new namespace or include a library; rather, it allows for more accessible use of the existing standard library functions defined in the standard namespace.

10. What will the following code: int x = 10; x++; return x; return 5; yield?

A. 5

B. 10

C. 11

D. Compilation error

The code snippet provided demonstrates how the increment operator and the return statement operate in sequence. Initially, the variable `x` is assigned the value of 10. The subsequent statement `x++` uses the post-increment operator, which increases the value of `x` by 1 after its current value is evaluated. Since `x` is initially 10, after the `x++` operation, `x` becomes 11. The first return statement, `return x;`, will return the current value of `x`, which is now 11. After a return statement is reached, control exits from the function, so any code placed after the first return statement will not be executed. As a result, the second return statement, `return 5;`, would never be executed. Thus, the function effectively ends after returning the incremented value. Therefore, when the function with this code is invoked, it will yield the value 11, confirming that the correct answer is the third choice.

SAMPLE

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!

SAMPLE