Arizona State University (ASU) CSE100 Principles of Programming with C++ Midterm 1 Practice Exam (Sample)

Study Guide

Everything you need from our exam experts!

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. What does exception handling in C++ aim to address?

A. Compile time errors

B. Syntax errors

C. Runtime errors

D. Logical errors

2. What does the "++i" notation signify in C++?

A. Post-increment

B. Pre-increment

C. Decrement

D. None of the above

3. What is method overriding in C++?

A. Defining two methods with the same name but different parameters

B. Providing a new implementation of a method in a derived class

C. Disallowing method definitions in a derived class

D. Creating a method that prevents base class methods from being called

4. Which of the following is NOT one of the properties of an algorithm?

A. Ambiguous

B. Independent from the programming language

C. Has a finite number of steps

D. Terminating

5. What is the outcome of uninitialized variables in C++?

A. They automatically receive a default value

B. They can cause compile errors

C. They may contain garbage values

D. They are set to zero

6. What is the role of the `return` type in a function declaration?

- **A. To define how the function can be called**
- **B. To specify the value type returned after execution**
- **C. To indicate the function's parameters**
- **D. To declare global variables**

7. What does the 'return 0;' statement signify in a C++ program?

- **A. It indicates an error occurred during execution**
- **B. It signifies successful execution of the program**
- **C. It is a command to print output to the console**
- **D. It initializes the program's return value**

8. What is the purpose of the 'throw' keyword in C++?

- **A. To terminate the program**
- **B. To create an object**
- **C. To raise an exception**
- **D. To declare a variable**

9. What is the first level of operator precedence in C++?

- **A. +, -**
- **B. ( )**
- **C. *, /, %**
- **D. None**

10. What is the function of an array index in C++?

- **A. To modify the content of an array**
- **B. To access specific elements within an array**
- **C. To declare an array**
- **D. To store multiple data types**

Answers

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

Explanations

1. What does exception handling in C++ aim to address?

A. Compile time errors

B. Syntax errors

C. Runtime errors

D. Logical errors

Exception handling in C++ is primarily designed to address runtime errors. These are errors that occur during the execution of a program, which can disrupt the normal flow of operations. When such errors arise, they can lead to system crashes or unexpected behavior, making it essential to have a way to manage them gracefully. By using exception handling, programmers can anticipate potential problems, such as invalid input, memory allocation failures, or out-of-bounds access, and implement a strategy to catch and respond to these issues at runtime. This helps in maintaining program stability and allows developers to provide meaningful error messages or recovery options. In contrast, compile time errors, syntax errors, and logical errors are issues that are typically identified through the compiling process or during the code review stage. Compile time errors are related to code that does not conform to the language rules, syntax errors involve mistakes in the program structure, and logical errors pertain to flaws in the algorithm that lead to incorrect results without crashing the program. Exception handling does not directly address these types of issues, which is why runtime errors are the focus of this feature in C++.

2. What does the "++i" notation signify in C++?

A. Post-increment

B. Pre-increment

C. Decrement

D. None of the above

The "++i" notation in C++ signifies pre-increment. When you use this notation, the variable "i" is incremented by 1 before its value is used in an expression. This means that if "i" is initially, for example, 5, using "++i" will first increase its value to 6 and then return that new value. In contrast, if the notation were "i++", it would imply post-increment. In that case, the current value of "i" would be used in an expression first, and then "i" would be incremented afterwards. Understanding pre-increment is important as it can affect the order of operations and the final outcome of expressions when variables are involved. Therefore, recognizing "++i" as pre-increment is essential for correctly predicting the behavior of your code in execution contexts.

3. What is method overriding in C++?

A. Defining two methods with the same name but different parameters

B. Providing a new implementation of a method in a derived class

C. Disallowing method definitions in a derived class

D. Creating a method that prevents base class methods from being called

Method overriding in C++ occurs when a derived class provides a new implementation of a method that is already defined in its base class. This is a core concept in object-oriented programming that allows the derived class to have a behavior that differs from the base class, while still maintaining the same method signature. When a method in a base class is declared as `virtual`, the derived class can override this method with its own implementation. This allows for polymorphism, enabling objects of the derived class to be treated as objects of the base class while still executing their specific overridden behavior when the method is called. This is particularly useful when you want to have a common interface but vary the functionality in subclasses. For instance, if you have a base class `Animal` with a method `speak()`, the derived class `Dog` can override `speak()` to provide a sound appropriate to a dog, while other animals like `Cat` might implement their own version of `speak()` that sounds like a meow. In contrast, defining two methods with the same name but different parameters pertains to method overloading, which is a different concept where the method signature varies. Disallowing method definitions or creating methods that prevent base class methods from being called

4. Which of the following is NOT one of the properties of an algorithm?

A. Ambiguous

B. Independent from the programming language

C. Has a finite number of steps

D. Terminating

An algorithm is a well-defined set of instructions that provides a solution to a particular problem. For it to be effective, several properties are essential, and one of those is clarity. A good algorithm should not be ambiguous; it needs to be clear and unambiguous in its instructions to ensure that it can be followed or implemented correctly. The other properties outlined are critical for the definition of an algorithm. Firstly, being independent from the programming language means that an algorithm can be expressed in various programming languages without altering its fundamental logic. This universality allows algorithms to be versatile and widely applicable in different contexts. Having a finite number of steps is also essential because an algorithm must conclude after a specific number of operations, leading to a definitive answer rather than continuing indefinitely. This finiteness ensures that the algorithm ultimately produces a result. Lastly, for an algorithm to be practical, it must be terminating, meaning that it should reach a conclusion after executing its instructions. In summary, the correct answer indicates an improper characteristic of an algorithm, while the others reflect important properties that define effective algorithms.

5. What is the outcome of uninitialized variables in C++?

A. They automatically receive a default value

B. They can cause compile errors

C. They may contain garbage values

D. They are set to zero

In C++, uninitialized variables do not automatically receive a value and can contain what is known as "garbage values." Garbage values are essentially random data that happen to be stored in the memory location allocated for the variable. Since the variable has not been explicitly initialized by the programmer, the contents of that memory location can reflect any residual data leftover from previous operations or functions. This behavior underscores the importance of initializing variables before use, as working with a variable containing garbage values can lead to unpredictable program behavior, including incorrect calculations, logical errors, or even program crashes. In contrast to what some might think, uninitialized variables do not get assigned default values automatically. They also don't cause compile errors, and they are not inherently set to zero; instead, their state is indeterminate, hence the critical need to ensure variables are initialized properly to maintain predictable program execution.

6. What is the role of the `return` type in a function declaration?

A. To define how the function can be called

B. To specify the value type returned after execution

C. To indicate the function's parameters

D. To declare global variables

The role of the `return` type in a function declaration is to specify the value type that the function will return after execution. This is a crucial element of function declarations in C++ because it informs the compiler (and the programmer) what kind of data to expect when the function is called. For example, if a function has a return type of `int`, it indicates that upon completion, the function will provide an integer value. This helps in ensuring type safety in programming, as it enables proper handling of the returned data and allows for better readability and maintenance of the code. Functions can also have a return type of `void`, which means they do not return any value. However, when a specific data type is declared, it helps to define the behavior and usage of the function clearly. Understanding the return type also aids in debugging and understanding the logic flow of a program, especially in larger and more complex codebases.

7. What does the 'return 0;' statement signify in a C++ program?

A. It indicates an error occurred during execution

B. It signifies successful execution of the program

C. It is a command to print output to the console

D. It initializes the program's return value

The 'return 0;' statement in a C++ program indicates successful execution of the program. In C++, the main function returns an integer value to the operating system. By convention, a return value of 0 signifies that the program has completed its operation without encountering any errors. This is an important aspect of programming that provides a standardized way for a program to communicate its execution status back to the environment that called it. In contrast, if the program were to return a non-zero value, it would typically indicate that an error or an exceptional condition occurred during execution. The actual values and their meanings can vary across different programs, but returning 0 is universally recognized as a signal of success. Other options mention aspects such as error indication, output commands, or initialization of return values, but they do not align with the standard interpretations of 'return 0;' in the context of C++. Therefore, 'return 0;' solely serves the purpose of signaling a successful exit from the program.

8. What is the purpose of the 'throw' keyword in C++?

A. To terminate the program

B. To create an object

C. To raise an exception

D. To declare a variable

The 'throw' keyword in C++ is used to raise an exception when an error or an unexpected condition occurs during program execution. When a 'throw' statement is encountered, control is transferred to the nearest exception handler that can manage the type of exception being thrown. This mechanism allows developers to implement robust error handling in their programs, enabling them to manage runtime errors gracefully instead of letting the program crash. In essence, using 'throw' provides a way to signal that an exceptional condition has occurred, prompting the program to take corrective measures, such as cleaning up resources or providing user feedback. This structured way of handling errors aligns with C++'s emphasis on object-oriented programming, promoting code that is both safer and more maintainable. The other options are not relevant to the function of the 'throw' keyword in C++. Terminating the program, creating an object, or declaring a variable do not involve signaling or managing exceptions, which is the primary role of 'throw' in C++.

9. What is the first level of operator precedence in C++?

A. +, -

B. ( )

C. *, /, %

D. None

In C++, operator precedence determines the order in which different operators are evaluated in an expression. The parentheses, represented by ( ), are at the highest level of precedence. This allows for the grouping of expressions, ensuring that the operations contained within the parentheses are carried out first, regardless of the precedence of other operators outside the parentheses. For example, in the expression \(3 + 5 * 2\), multiplication has a higher precedence than addition. However, if you write it as \((3 + 5) * 2 \), the addition inside the parentheses is evaluated first, followed by the multiplication. This demonstrates how parentheses can override the default precedence rules. While other operators like addition (+, -), multiplication (*, /, %) have their own levels of precedence, none of them can change the order of evaluation as effectively as parentheses can. Hence, acknowledging parentheses as the first and highest level of operator precedence is fundamental in understanding how expressions are evaluated in C++.

10. What is the function of an array index in C++?

A. To modify the content of an array

B. To access specific elements within an array

C. To declare an array

D. To store multiple data types

An array index in C++ serves the primary function of accessing specific elements within an array. In C++, arrays are collections of elements that are accessed using an index, which represents the position of an element within the array. The first element of the array is at index 0, the second at index 1, and so on. This numbering system allows programs to retrieve or manipulate data stored in the array efficiently. For example, if you have an array of integers named `arr`, you can access the first element using `arr[0]`, the second element using `arr[1]`, and so forth. This indexing mechanism is crucial for reading and writing data to specific locations in the array, enabling effective data management in programming. The other choices reflect different concepts related to arrays in C++. Modifying the content of an array is a broader operation that can be achieved using an index, but the index's specific role is not to modify but to access. Declaring an array involves specifying its type and size, separate from how elements of the array are accessed. Storing multiple data types is not applicable to standard arrays in C++ since arrays are intended to hold elements of the same data type only.

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!