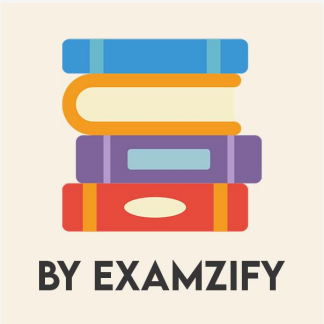
Arizona State University (ASU) CSE100 Principles of Programming with C++ Midterm 1 Practice Exam (Sample)

Study Guide



Everything you need from our exam experts!

Questions

1. What symbol is used for a single-line comment in C++?

   A. /* ... */

   B. //

   C. #

   D. **

2. What is the role of the control unit within the CPU?

   A. To perform arithmetic calculations

   B. To retrieve and decode instructions

   C. To execute logical operations

   D. To allocate memory resources

3. What will happen when you perform an operation between an int and a float in a mixed expression?

   A. The float will be ignored

   B. The int will be ignored

   C. Both will be converted to int

   D. The int will be converted to float

4. In C++, what control structure is typically used to execute code repeatedly?

   A. Condition statement

   B. Loop

   C. Switch case

   D. Function call

5. What defines a global variable in C++?

   A. It is only accessible within a single function

   B. It is declared inside a loop

   C. It can be accessed from any part of the program

   D. It can only be accessed in its compilation unit

6. 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

7. In mixed expressions, what occurs with different data types?

A. They cause runtime errors

B. They cannot be executed

C. They convert to a single data type

D. They are ignored in calculations

8. How do you declare a float variable in C++?

A. float variableName;

B. var float variableName;

C. float: variableName;

D. declare float variableName;

9. What is the purpose of a variable in C++?

A. To store data that cannot be changed

B. To store data that can be changed during program execution

C. To create functions that perform tasks

D. To define a fixed value

10. What does the 'return 0;' statement indicate in the main function of a C++ program?

A. The program has completed with errors

B. The program has not compiled

C. The program executed successfully

D. The program should restart

Answers

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

Explanations

1. What symbol is used for a single-line comment in C++?

A. /* ... */

B. //

C. #

D. **

In C++, a single-line comment is denoted by the use of two forward slashes, represented as "//". This tells the compiler to ignore everything on that line after the slashes. Using single-line comments is a practical way for programmers to annotate their code, making it easier to understand without affecting the execution of the program. The other options represent different forms of commenting or other concepts in programming. For instance, /* ... */ is used for multi-line comments, where anything between the /* and */ is commented out, allowing for several lines to be commented at once. The # symbol is often used for preprocessor directives in C and C++, not for comments. Lastly, ** is not a recognized comment syntax in C++. Thus, the correct symbol for a single-line comment in C++ is indeed "//", as it efficiently comments out a line or a portion of a line of code.

2. What is the role of the control unit within the CPU?

A. To perform arithmetic calculations

B. To retrieve and decode instructions

C. To execute logical operations

D. To allocate memory resources

The control unit within the CPU is primarily responsible for retrieving and decoding instructions from program memory. It orchestrates the overall operation of the processor by directing the flow of data between the CPU's other components, such as the arithmetic logic unit (ALU), memory, and input/output devices. As a result, it ensures that the instructions are executed in the proper sequence and that the correct operations are carried out by coordinating the activities of these components. While performing arithmetic calculations and executing logical operations are critical functions, these tasks are specifically handled by the arithmetic logic unit, not the control unit. Memory resource allocation typically falls under the responsibilities of the memory management unit or the operating system, rather than the control unit of the CPU. Thus, retrieving and decoding instructions is the primary and defining role of the control unit, making it the correct answer.

3. What will happen when you perform an operation between an int and a float in a mixed expression?

A. The float will be ignored

B. The int will be ignored

C. Both will be converted to int

D. The int will be converted to float

In a mixed expression where an operation involves both an integer and a floating-point number, implicit type conversion happens to ensure that the operation can be performed accurately. In this case, the integer will be converted to a floating-point value. This conversion allows the arithmetic operation to take place without losing any decimal precision that might be associated with the float. When an integer is converted to a float, its value remains the same, but it is represented in a way that can accommodate decimal points. This is crucial because if the integer were to be ignored or converted to an int during the operation, it could lead to loss of precision, especially in cases where results may yield a decimal value. The conversion is a part of the rules of type promotion in C++, where the data type that can hold a larger range or more precision (in this case, float) takes precedence over the other. Thus, by converting the integer to a float, both values can be treated consistently in the operation, resulting in a float output.

4. In C++, what control structure is typically used to execute code repeatedly?

A. Condition statement

B. Loop

C. Switch case

D. Function call

In C++, the primary control structure that facilitates the repeated execution of code is the loop. Loops, such as `for`, `while`, and `do-while`, enable a programmer to run a block of code multiple times until a specified condition evaluates to false. This is essential for tasks that require repetition, such as iterating through arrays, processing user input until a sentinel value is entered, or performing an operation a set number of times. Each type of loop serves different scenarios: a `for` loop is often used when the number of iterations is known beforehand; a `while` loop is suitable when the number of iterations is not predetermined and depends on a condition; and a `do-while` loop ensures that the block of code runs at least once before checking the condition. Other structures, such as condition statements (like `if` and `else`), handle branching logic and executing code based on certain conditions but do not inherently repeat execution. The switch case directs the flow of execution to different sections of code based on the value of a variable without establishing repetition. A function call invokes a separate piece of code that can be executed, but it doesn't create repetition on its own unless combined with loops. Thus, the loop is the

5. What defines a global variable in C++?

A. It is only accessible within a single function

B. It is declared inside a loop

C. It can be accessed from any part of the program

D. It can only be accessed in its compilation unit

A global variable in C++ is a variable that is declared outside of all functions, typically at the top of the source file. This positioning gives it a scope that extends throughout the entire program, allowing it to be accessed and modified from any function within that program. This characteristic is what fundamentally defines a global variable, and it plays a significant role in managing data that needs to be shared across different parts of a program. In contrast, a variable declared inside a function (which would be local to that function) can only be accessed within that function, illustrating the limitations of local scope. Variables declared inside loops are also local and suffer from the same restriction. Furthermore, a compilation unit restricts visibility to only that specific unit, which may include a single file or module. This limitation further distinguishes it from global variables that are accessible program-wide. Therefore, the correct understanding of a global variable's definition is that it can indeed be accessed from any part of the program.

6. 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++.

7. In mixed expressions, what occurs with different data types?

A. They cause runtime errors

B. They cannot be executed

C. They convert to a single data type

D. They are ignored in calculations

In mixed expressions involving different data types, the values are converted to a single data type to provide a consistent type for the operation being performed. This process is known as "type promotion" or "implicit conversion." For example, if you combine an integer and a floating-point number in an expression, the integer will be converted to a float so that both operands match in type, allowing the operation to be executed without error. This conversion ensures that calculations can proceed smoothly without the programmer needing to manually adjust the data types. The other options are not applicable in this context. For instance, while runtime errors can occur in some cases, they are not a direct outcome of mixed data types since C++ manages type conversions. The idea that mixed types cannot be executed is incorrect, as C++ allows for expressions involving multiple data types through this conversion mechanism. Lastly, saying that mixed types are ignored in calculations is inaccurate because C++ computes the result by converting the operands to a common type, rather than overlooking any component of the expression.

8. How do you declare a float variable in C++?

A. float variableName;

B. var float variableName;

C. float: variableName;

D. declare float variableName;

To declare a float variable in C++, the correct syntax is to use the keyword `float` followed by the variable name. This establishes that the variable will store floating-point numbers, which can represent decimal values. The declaration format `float variableName;` is essential because it clearly indicates the type of the variable (float) and assigns it a name that will be used to reference that variable throughout the program. This is a foundational concept in programming, where understanding the data types and their declarations is crucial for effective variable management in any code. The other options do not follow the correct syntax for declaring a float variable in C++. For example, 'var float variableName;' does not conform to C++ rules, as 'var' is not a recognized keyword for declaring variables in the language. Similarly, 'float: variableName;' and 'declare float variableName;' violate syntactical rules and would result in compilation errors. Understanding the correct way to declare variables ensures that your program can compile and run as expected.

9. What is the purpose of a variable in C++?

A. To store data that cannot be changed

B. To store data that can be changed during program execution

C. To create functions that perform tasks

D. To define a fixed value

The purpose of a variable in C++ is fundamentally about its ability to store data that can be changed during the execution of a program. Variables serve as named storage locations in memory, which allows programmers to hold different values at different points in time. This flexibility is crucial for a wide array of programming tasks, as it enables dynamic manipulation of data based on user inputs, calculations, or changes in program state. For instance, consider a simple program that asks a user for their age and then updates the variable to reflect any changes, such as a year passing. The variable can take on different values as the program runs, which aligns perfectly with the nature of many applications that require interaction and adaptation. In contrast, the other options describe characteristics that do not align with the primary function of variables. Storing data that cannot be changed pertains to constants rather than variables. Functions indeed perform tasks, but they are not the same concept as variables. Defining a fixed value corresponds to how constants are used within a program rather than variables, which are meant to vary. Understanding these definitions helps clarify the vital role that variables play in programming.

10. What does the 'return 0;' statement indicate in the main function of a C++ program?

A. The program has completed with errors

B. The program has not compiled

C. The program executed successfully

D. The program should restart

The statement 'return 0;' in the main function of a C++ program signifies that the program has executed successfully. In C++, the value returned from the main function serves as an exit status code back to the operating system. By convention, returning 0 indicates that the program has run without any issues or errors, while a non-zero return value typically indicates the presence of some error or exceptional condition. This convention provides a way for other programs or scripts that might call your program to determine whether it has finished its task successfully or if any problems arose during its execution. Therefore, recognizing 'return 0;' as an indication of successful completion is essential for structuring robust C++ applications.