CERTIFIED QUIZ

History and Features

C++ was developed by Bjarne Stroustrup at Bell Labs in the late 1970s as an enhancement to the C programming language.
Its main features include:
- Object-oriented programming (OOP)
- Generic programming
- Low-level memory manipulation
- Compatibility with C
- Standard Template Library (STL)

Setting Up the Development Environment

To start programming in C++, you need:
- A text editor or an Integrated Development Environment (IDE) such as Visual Studio, Code::Blocks, or CLion.
- A C++ compiler like GCC, Clang, or MSVC.
- Proper configuration of the environment variables, if necessary.

Basic Syntax

- Every C++ program starts with a `main` function.
- Statements end with a semicolon (`;`).
- C++ is case-sensitive.

Data Types

Basic Data Types:
- int: Integer type (e.g., `int age = 25;`).
- float: Floating-point type (e.g., `float pi = 3.14;`).
- double: Double-precision floating-point type.
- char: Character type (e.g., `char letter = 'A';`).
- bool: Boolean type (true or false).

Variables and Constants

- Variables are declared using data types and can be modified.
- Constants are declared using the `const` keyword and cannot be modified after initialization.

Operators

Arithmetic Operators:
- Used for performing mathematical calculations.
- Examples: `+`, `-`, ``, `/`, `%`.

Relational Operators:
- Used for comparing values.
- Examples: `==`, `!=`, `>`, `<`, `>=`, `<=`.

Logical Operators:
- Used to combine or modify boolean expressions.
- Examples: `&&`, `||`, `!`.

Bitwise Operators:
- Used for manipulating bits.
- Examples: `&`, `|`, `^`, `~`, `<<`, `>>`.

Control Structures

Conditional Statements:
- if statement: Executes code if the condition is true.
- else statement: Executes code if the condition is false.
- switch statement: Selects one of many code blocks to execute.

Loops:
- for loop: Used for iterating over a range of values.
- while loop: Continues executing as long as the condition is true.
- do-while loop: Executes code at least once before checking the condition.

Functions

Function Declaration and Definition:
- Functions are defined with a return type, name, and parameters.
- Example:
cpp
int add(int a, int b) {
return a + b;
}

Function Parameters and Return Values:
- Parameters are specified in the function declaration.
- The return type indicates what type of value the function will return.

Function Overloading:
- Multiple functions can have the same name with different parameters.

Inline Functions:
- Inline functions are defined with the `inline` keyword to suggest to the compiler to insert the function's body at the point of call to reduce function call overhead.

Arrays and Strings

Declaring and Initializing Arrays:
- Arrays are a collection of items of the same type.
- Example: `int numbers[5] = {1, 2, 3, 4, 5};`

Multidimensional Arrays:
- Arrays that contain arrays. Example: `int matrix[3][3];`

C-style Strings vs. C++ Strings:
- C-style strings: Null-terminated character arrays.
- C++ strings: Use the `std::string` class for more functionality.

Intermediate Topics

Pointers and References:

Pointer Basics and Syntax:

- Pointers store memory addresses. Example: `int ptr;`

Pointer Arithmetic:
- You can perform arithmetic operations on pointers.

References vs. Pointers:
- References are an alias for a variable, while pointers store memory addresses.

Dynamic Memory Management

New and Delete Operators:
- `new` is used to allocate memory, and `delete` is used to deallocate it.

Memory Leaks and Smart Pointers:
- Memory leaks occur when memory is allocated but not freed.
- Smart pointers (`std::unique_ptr`, `std::shared_ptr`) help manage memory automatically.

Structures and Enums

Defining and Using Structures:
- Structures are user-defined data types. Example:
cpp
struct Person {
string name;
int age;
};

Enumerated Types:
- Enumerations define a variable that can hold a set of predefined constants. Example:
cpp
enum Day {Sun, Mon, Tue, Wed, Thu, Fri, Sat};

File Handling

Reading from and Writing to Files:
- Use `ifstream` for input and `ofstream` for output.

File Streams:
- Stream classes in C++ for handling file operations.

Classes and Objects

Class Declaration and Definition:
- Classes encapsulate data and functions. Example:
cpp
class Car {
public:
void drive();
};

Constructors and Destructors:
- Constructors initialize objects, while destructors clean up when an object goes out of scope.

Member Functions and Access Specifiers:

- Member functions are defined within classes. Access specifiers control visibility: `public`, `protected`, and `private`.

Static Members:
- Static members belong to the class rather than any instance.

Advanced Topics

Inheritance:

Single and Multiple Inheritance:
- Single inheritance involves one base class; multiple inheritance involves more than one.

Base and Derived Classes:
- Derived classes inherit properties and methods from base classes.

Access Specifiers in Inheritance:
- Control the accessibility of base class members in derived classes.

Polymorphism:

Function Overriding:
- Derived classes can provide specific implementations of base class functions.

Virtual Functions and Pure Virtual Functions:
- Virtual functions enable dynamic binding; pure virtual functions make a class abstract.

Abstract Classes:
- Classes with at least one pure virtual function.

Templates

Function Templates:
- Allow functions to operate with generic types.

Class Templates:
- Define a class that can operate with any data type.

Template Specialization:
- Customizes the behavior of templates for specific types.

Standard Template Library (STL)

Containers:
- Sequence containers (e.g., `vector`, `list`) and associative containers (e.g., `map`, `set`).

Iterators:
- Objects that allow traversal of container elements.

Algorithms:
- Predefined functions for common tasks (e.g., `sort`, `search`).

Exception Handling

Try, Catch, and Throw:
- Mechanism for handling runtime errors.

Custom Exceptions:
- User-defined exception classes for specific error types.

Operator Overloading

Overloading Operators:
- Customizing the behavior of operators for user-defined types.

Friend Functions:
- Allow non-member functions to access private and protected members of a class.

Multithreading (C++11 and Above)

Threads and Thread Management:
- Create and manage threads for concurrent execution.

Mutexes and Condition Variables:
- Synchronization tools to prevent data races.

Thread Safety and Synchronization:
- Techniques to ensure data integrity in multithreaded environments.

Additional Topics

- Preprocessors and Macros.
- Typecasting.
- Namespaces.
- Lambda Expressions.
- C++11 and newer features.