

# Top 50 CPP Interview Questions

Topics : [CPP Interview Questions](#)

Written on [November 22, 2023](#)

## Basics and Fundamentals:

### 1. What is C++?

- C++ is a general-purpose programming language developed as an extension of the C programming language. It supports both procedural and object-oriented programming paradigms.

### 2. Differentiate between C and C++.

- C++ is an extension of C and includes features like classes, objects, and polymorphism, which are not present in C. C is procedural, while C++ supports both procedural and object-oriented programming.

### 3. Explain the importance of the volatile keyword in C++.

- The `volatile` keyword is used to indicate that a variable may be changed by multiple threads or external factors, preventing compiler optimizations that might assume the variable remains unchanged.

### 4. What is the difference between struct and class in C++?

- In a `struct`, members are public by default, while in a `class`, members are private by default. Additionally, `class` supports encapsulation and inheritance, while `struct` is primarily used for plain data structures.

### 5. Discuss the concept of function overloading and provide an example.

- Function overloading allows multiple functions with the same name but different parameter lists. For example:

```
int add(int a, int b);  
float add(float a, float b);
```

### 6. What is the difference between new and malloc() in C++?

- `new` is an operator in C++ that allocates memory for an object and calls its constructor. `malloc()` is a function in C that allocates raw memory without calling any constructors.

**7. Explain the purpose of the `explicit` keyword in C++.**

- The `explicit` keyword is used to prevent implicit type conversions, making constructors with a single parameter explicit in their usage.

**8. How does the `sizeof` operator work in C++?**

- The `sizeof` operator returns the size, in bytes, of a variable or data type. For example, `sizeof(int)` returns the size of an integer.

**9. Describe the purpose of the `inline` keyword.**

- The `inline` keyword suggests the compiler to insert the code of a function directly into the calling code, potentially improving performance by avoiding function call overhead.

**10. What is a lambda expression in C++? Provide an example.**

- A lambda expression is an anonymous function. Example:

```
auto sum = [](int a, int b) { return a + b; };  
int result = sum(3, 4);
```

**Object-Oriented Programming (OOP):**

**11. Explain the concept of inheritance in C++.**

- Inheritance allows a class to inherit properties and behaviors from another class. Example:

```
class Base {  
public:  
    int getData() { return data; }  
private:  
    int data;  
};  
  
class Derived : public Base {  
    // Derived has access to getData()  
};
```

**12. What is polymorphism? How is it achieved in C++?**

- Polymorphism allows objects of different types to be treated as objects of a common type. It is achieved in C++ through function overloading and virtual functions.

**13. Describe encapsulation and its advantages in C++.**

- Encapsulation is the bundling of data and the methods that operate on the data into a single unit (class). It helps in data hiding and protects the integrity of the data.

**14. What is the difference between early binding and late binding?**

- Early binding (static binding) occurs at compile-time, while late binding (dynamic binding) occurs at runtime. Virtual functions enable late binding in C++.

**15. Discuss the importance of the `virtual` keyword in C++.**

- The `virtual` keyword is used to declare virtual functions in base classes, allowing them to be overridden by derived classes. It enables polymorphic behavior.

**16. How does the `friend` keyword work in C++?**

- The `friend` keyword allows a function or class to access private members of another class. It is often used for implementing non-member functions that need access to private members.

**17. Explain the concept of an abstract class.**

- An abstract class is a class that cannot be instantiated and may have one or more pure virtual functions. It serves as a base class for other classes, providing a common interface.

**18. What is multiple inheritance, and how is it implemented in C++?**

- Multiple inheritance allows a class to inherit from more than one base class. It is implemented in C++ by separating the base class names with commas in the class declaration.

**19. Discuss the role of a destructor in C++.**

- A destructor is a special member function that is called when an object goes out of scope or is explicitly deleted. It is used to release resources and perform cleanup.

**20. What is an interface in C++?**

- C++ does not have a specific `interface` keyword like some other languages. Interfaces are typically represented by abstract classes with pure virtual functions.

## **Pointers and Memory Management:**

**21. What is a pointer? How is it different from a reference?**

- A pointer is a variable that holds the memory address of another variable. A reference is an alias for a variable. Unlike pointers, references cannot be NULL and cannot be reseated to refer to another variable.

**22. Explain the purpose of the `nullptr` keyword in C++.**

- `nullptr` is a keyword introduced in C++11 to represent a null pointer. It is recommended to use `nullptr` instead of `NULL` or `0` for better type safety.

**23. Discuss the differences between `delete` and `delete[]` in C++.**

- `delete` is used to deallocate memory for a single object created using `new`, while `delete[]` is used to deallocate memory for an array of objects created using `new[]`.

**24. What is a smart pointer, and why is it used?**

- A smart pointer is a C++ object that acts like a pointer but provides additional features such as automatic memory management. Examples include `std::shared_ptr` and `std::weak_ptr`.

**25. How does memory leak occur in C++? How can it be avoided?**

- Memory leaks occur when dynamically allocated memory is not deallocated. They can be avoided by properly using `delete` for each `new` and by using smart pointers.

**26. Explain shallow copy and deep copy.**

- Shallow copy copies the values of the members, including pointers, but does not duplicate the dynamically allocated memory. Deep copy creates a new copy of the dynamically allocated memory.

**27. What is the purpose of the `const` keyword when used with pointers?**

- When used with pointers, `const` can be used to indicate that the pointed-to data is constant (`const int*`) or that the pointer itself is constant (`int* const`).

**28. Discuss the concept of a dangling pointer.**

- A dangling pointer is a pointer that points to memory that has been deallocated or is otherwise invalid. Dereferencing a dangling pointer can lead to undefined behavior.

**29. How is dynamic memory allocation handled in C++?**

- Dynamic memory allocation is done using operators `new` and `delete` or `malloc()` and `free()`. It allows memory to be allocated at runtime and must be manually deallocated.

**Templates and STL:**

**30. What are templates in C++? Provide an example.**

- Templates allow the creation of generic classes and functions. Example:

```
template <typename T>
T add(T a, T b) {
    return a + b;
}
```

**31. Explain the concept of template specialization.**

- Template specialization allows defining a specialized implementation for a specific data type. Example:

```
template <>
int add<int>(int a, int b) {
    return a * b; // Specialized implementation for int
}
```

**32. Discuss the purpose of the Standard Template Library (STL).**

- The STL is a collection of template classes and functions in C++, providing generic algorithms, containers, and iterators. It simplifies complex data structures and algorithms.

**33. What is the difference between a vector and a list in STL?**

- A vector is a dynamic array that allows fast random access, while a list is a doubly-linked list that allows efficient insertion and deletion at both ends.

**34. How does the map container work in STL?**

- `std::map` is an associative container that stores key-value pairs in a sorted order based on the key. It uses a binary search tree (usually a red-black tree) for efficient lookups.

**35. What is an iterator in C++?**

- An iterator is an object that points to an element in a container, allowing traversal and manipulation of container elements. Iterators provide a uniform way to access elements in different containers.

**36. Explain the purpose of the algorithm header in STL.**

- The `algorithm` header provides a collection of template functions for common algorithms such as sorting, searching, and manipulating elements in containers.

**37. Discuss the difference between `std::vector` and `std::array`.**

- `std::vector` is a dynamic array that can grow or shrink in size, while `std::array` is a fixed-size array with a fixed capacity determined at compile-time.

**Exception Handling:**

**38. What is exception handling in C++?**

- Exception handling is a mechanism to handle runtime errors and unexpected situations. It involves `try`, `catch`, and `throw` keywords.

**39. Explain the `try`, `catch`, and `throw` keywords.**

- `try` is used to enclose a block of code that might throw an exception. `catch` is used to handle exceptions, and `throw` is used to throw an exception.

**40. Discuss the difference between runtime and compile-time errors.**

- Compile-time errors are detected by the compiler during the compilation process, while runtime errors occur during program execution. Exception handling is used to address runtime errors.

**41. How is exception handling implemented in C++?**

- C++ uses a stack-based mechanism to propagate and handle exceptions. When an exception is thrown, the program unwinds the call stack until a matching catch block is found.

## **Advanced Topics:**

### **42. What is the role of the `volatile` keyword in multithreading?**

- The `volatile` keyword indicates that a variable may be changed by multiple threads. It prevents the compiler from optimizing away reads and writes to the variable.

### **43. Explain the concept of move semantics in C++.**

- Move semantics involve efficiently transferring ownership of resources (such as memory) from one object to another without unnecessary copying. It is implemented using move constructors and move assignment operators.

### **44. Discuss the purpose of the `override` keyword.**

- The `override` keyword is used to indicate that a member function in a derived class is intended to override a virtual function in the base class. It helps catch errors when the intended override is missing.

### **45. What is the Rule of Three in C++?**

- The Rule of Three states that if a class defines any of the following three, it should define all three: destructor, copy constructor, and copy assignment operator. This ensures proper resource management.

### **46. What is the Rule of Five in C++?**

- The Rule of Five extends the Rule of Three to include move constructor and move assignment operator. If a class manages resources, it should define all five special member functions.

### **47. Explain the concept of CRTP (Curiously Recurring Template Pattern).**

- CRTP is a C++ programming pattern where a class template derives from a class that is a template specialization of itself. It is often used to implement static polymorphism.

### **48. Discuss the use of the `auto` keyword in C++.**

- The `auto` keyword allows the compiler to automatically deduce the type of a variable during compilation, reducing the need for explicit type declarations.

### **49. What is RAII (Resource Acquisition Is Initialization)?**

- RAII is a programming idiom in C++ where resource management is tied to object lifetime. Resources are acquired in the object's constructor and released in its destructor.

### **50. Explain the concept of type erasure in C++.**

- Type erasure is a technique to store objects of different types in a container while exposing a uniform interface. It is often used in scenarios where a single interface is needed for diverse types.

© Copyright **Aryatechno**. All Rights Reserved. Written tutorials and materials by [Aryatechno](#)

ARYATECHNO