90 C++ Interview Questions to Hire Top Engineers

Questions

1. What's the difference between `struct` and `class` in C++? It's like asking, 'Are they twins or just good friends?'

2. Can you explain what a pointer is in C++? Imagine it as a treasure map that tells you where something valuable is hidden.

3. What is the purpose of the `new` and `delete` keywords in C++? Think of them as tools to borrow and return toys from a big toy library.

4. What is a constructor in C++? Why is it useful? Imagine it's like setting up a new toy when you get it, so it's ready to play with.

5. What's the difference between pass by value and pass by reference? Consider sending a copy of your drawing versus letting your friend draw directly on your original.

6. Explain what is meant by function overloading in C++. It's like having your name, but people call you by different nicknames.

7. What is the purpose of the `const` keyword? Think of it as a rule that says, 'You can look, but don't touch!' about something.

8. What are header files and source files in C++? Imagine organizing toys where the box label is the header, and the actual toys are the source.

9. Describe what inheritance is in C++. It's when a toy robot gets new features from a toy car. What are the benefits?

10. What is polymorphism in C++? Consider a shape that can be drawn as a square or a circle. What are the different ways polymorphism can be achieved in C++?

11. What are namespaces in C++? How do they help? Think of them as different rooms in your house to prevent toy collisions.

12. Explain the difference between `==` and `=` in C++. One checks if things are the same, and the other assigns a value. Can you give an example for when it matters?

13. What is the Standard Template Library (STL) in C++? Consider it as a box of pre-built toy parts that you can use in many ways.

14. How does exception handling work in C++? Imagine if a toy breaks, how do you handle the problem gracefully without breaking all your other toys?

15. What are virtual functions in C++? Can you give an example of how virtual functions are useful with inheritance?

16. Explain the concept of memory leaks in C++. What could cause them and how can you prevent them?

17. What is a friend function in C++? Imagine a function that's allowed to play with the private parts of a toy, with the toy's permission.

18. Describe the purpose of the `static` keyword in C++. How does the meaning of `static` change when applied to a variable versus a function?

19. What are smart pointers in C++? Why and when should you use them? Consider them as toy handlers that automatically clean up toys when you're done.

20. Explain what is meant by RAII (Resource Acquisition Is Initialization). Why is RAII a good practice to follow in C++?

21. What is the difference between `class` and `struct` in C++?

22. Explain what a pointer is, and why is it useful?

23. What is the meaning of `const` keyword in C++?

24. What are the differences between pass by value, pass by reference, and pass by pointer?

25. What is the difference between `malloc` and `new`?

26. What does the term 'memory leak' mean in C++?

27. Explain the concept of inheritance.

28. What is function overloading?

29. What is the purpose of a constructor?

30. What is a destructor and when is it called?

31. What are virtual functions?

32. What is polymorphism, and how is it achieved in C++?

33. What are the access modifiers in C++ (public, private, protected) and how do they work?

34. Explain the Standard Template Library (STL). What are some common containers?

35. What is an iterator in C++?

36. What are templates in C++?

37. What is exception handling, and why is it important?

38. What is the difference between pre-increment and post-increment operators?

39. Describe the concept of namespaces in C++.

40. Explain what RAII (Resource Acquisition Is Initialization) is.

41. What are smart pointers and why are they used?

42. What are lambda expressions?

43. What is the purpose of the `static` keyword in C++?

44. What is operator overloading?

45. What are friend functions and friend classes?

46. Explain the concept of dynamic memory allocation.

47. What are some common debugging techniques in C++?

48. What is the difference between shallow copy and deep copy?

49. What is a header file, and why do we use them?

50. What are some of the new features introduced in C++11 or later standards that you find most useful?

51. What are virtual functions and why are they important in C++?

52. Explain the difference between shallow copy and deep copy.

53. What is the purpose of the `friend` keyword in C++?

54. How does exception handling work in C++? What are try, catch, and throw?

55. Describe the difference between `new` and `malloc` when allocating memory in C++.

56. What are namespaces in C++, and how do they help prevent naming collisions?

57. Explain the concept of operator overloading in C++ with an example.

58. What are smart pointers in C++, and why are they preferred over raw pointers?

59. Describe the difference between compile-time polymorphism and runtime polymorphism.

60. What is the Standard Template Library (STL) in C++? Give some examples of containers it provides.

61. Explain the concept of move semantics in C++11 and how it improves performance.

62. What is RTTI (Runtime Type Information) in C++ and when might you use it?

63. Describe the use of lambda expressions in C++11 and later.

64. Explain the concept of RAII (Resource Acquisition Is Initialization) and how it's used in C++.

65. What are templates in C++ and how do they enable generic programming?

66. How can you achieve thread safety in a multithreaded C++ application?

67. What are the different types of casting operators in C++ (static_cast, dynamic_cast, const_cast, reinterpret_cast) and when should each be used?

68. Explain what a virtual destructor is and why it's important in inheritance hierarchies.

69. Describe how you would implement a simple singleton pattern in C++.

70. What are variadic templates in C++ and how are they useful?

71. Explain the concept of RAII and how it prevents memory leaks in C++.

72. How does the C++ memory model support multithreading, and what are the key challenges?

73. Describe the difference between `std::move` and `std::forward`, and when should each be used?

74. What are the advantages and disadvantages of using exceptions for error handling in C++?

75. Explain the purpose of the `noexcept` specifier and its impact on code generation and exception safety.

76. How does the compiler resolve virtual function calls at runtime, and what are the performance implications?

77. Describe the SFINAE (Substitution Failure Is Not An Error) principle and provide an example of its use.

78. What are the differences between shared pointers, unique pointers and weak pointers? When do you use each?

79. Discuss the use cases for constexpr functions and variables in modern C++.

80. Explain the role of allocators in C++ and how they can be customized for specific memory management requirements.

81. How does the concept of perfect forwarding work, and why is it important for generic programming?

82. Describe the various ways to achieve compile-time polymorphism in C++, and compare their trade-offs.

83. What is the purpose of the `std::optional` type, and how does it improve code safety and readability?

84. Explain the concept of move semantics and its benefits for performance and resource management.

85. How can you prevent name collisions in large C++ projects?

86. Explain the difference between a function object (functor) and a lambda expression in C++.

87. Describe how you would implement a thread pool in C++ using standard library features.

88. Discuss the performance implications of using virtual inheritance compared to non-virtual inheritance.