100+ Computer Science Fundamentals Interview Questions

Questions

1. What happens when you type a URL in your browser and press Enter?

2. Explain what a variable is, like you're explaining it to a toy robot.

3. If a computer program is like a recipe, what are the ingredients?

4. What's the difference between RAM and a hard drive? Imagine RAM is your desk and the hard drive is a filing cabinet.

5. Can you describe what an operating system does for a computer? Pretend the OS is a playground supervisor.

6. What does it mean to 'debug' a program? Like finding a boo-boo on a stuffed animal.

7. Why are algorithms important in computer science? Think of an algorithm as a set of instructions to build a tower.

8. What is binary, and why do computers use it? It's like computers only know 'yes' and 'no'.

9. Explain what a loop is in programming, like going around in a circle.

10. What's the internet? Imagine it's a giant web connecting all the computers.

11. If you had a big pile of unsorted toys, how would you organize them? How does that relate to sorting algorithms?

12. What is a database? Think of it like a super organized address book.

13. Can you explain the difference between hardware and software? One you can touch, one you can't.

14. What is a function in programming? Think of it like a mini-program inside a bigger program.

15. What is source control, and why is it useful when working on a project with multiple people?

16. What's the difference between compiling code and interpreting code?

17. Explain what a data structure is and give an example.

18. What is the importance of testing in software development?

19. Describe the difference between the 'front-end' and the 'back-end' of a website or application.

20. What is an API, and why is it useful?

21. What is cloud computing, and what are some of its benefits?

22. What is a virtual machine?

23. How does encryption keep our data safe on the internet?

24. If a computer is like a sandwich, what are the different layers made of?

25. Can you explain the difference between RAM and a hard drive like I'm trying to remember where I put my toys?

26. What is an algorithm, and can you give me an example of one you use every day, like when you're getting ready in the morning?

27. If a computer program is like a recipe, what are the ingredients and how do they mix together?

28. What does it mean for a program to 'debug', and is it like finding a mistake in a drawing?

29. Explain what a variable is in programming, using the concept of a labeled box that holds a toy.

30. What are some basic data types, and can you relate them to different kinds of building blocks?

31. What is a loop in programming, and can you explain it like going around in a circle?

32. What is an if/else statement, and can you explain it like making a choice between two toys?

33. Can you explain what an array is in simple terms, using the analogy of a train with multiple cars?

34. What does 'function' mean in programming, relating to how different Lego blocks snap together?

35. Explain what a computer network is, similar to friends communicating with walkie-talkies.

36. What is the internet, and can you explain it like a giant playground where everyone can connect?

37. What does it mean for data to be 'encrypted,' similar to writing a secret message in code?

38. What are the key differences between hardware and software, as if they were parts of a toy robot?

39. How would you explain the concept of a database if it were a well-organized toy closet?

40. What is an operating system, and how does it act like the boss of the computer, managing everything?

41. What is a compiler, and how does it act like a translator between different languages?

42. Explain how a computer represents numbers, using the concept of building towers with blocks.

43. What is the role of binary code, and how does it act like a secret language between computers using only 0s and 1s?

44. Can you explain what source code is, comparing it to the instructions needed to build a LEGO set?

45. What's the difference between front-end and back-end development, as if they're the face and brain of a robot?

46. What is version control and why is it important, like keeping track of different versions of your drawings?

47. If a program isn't working as expected, what are some ways you can figure out what's wrong? (Think detective work!)

48. Imagine you're teaching a robot how to make a peanut butter and jelly sandwich. How would you break down the steps into simple instructions?

49. What are APIs (Application Programming Interfaces), and how are they like ordering food at a restaurant?

50. Can you give a simple explanation of cloud computing, likening it to sharing toys in a big virtual toy box?

51. Explain the concept of Object-Oriented Programming (OOP) using the idea of building different types of toy cars with specific properties and behaviors.

52. How does caching improve system performance, and what are some common caching strategies?

53. Explain the difference between TCP and UDP, and when you might choose one over the other.

54. What is the purpose of a hash function in data structures, and what are some considerations for choosing a good one?

55. Describe the concept of deadlock in concurrent programming, and how can it be prevented?

56. How does normalization improve database design, and what are the different normalization forms?

57. Explain the difference between authentication and authorization.

58. What is the difference between a stack and a queue, and how might you implement them?

59. How does garbage collection work in languages like Java or C#, and what are its advantages and disadvantages?

60. What is the purpose of indexes in databases, and how do they affect query performance?

61. Describe the concept of recursion, and provide an example of a problem that can be solved recursively.

62. Explain the difference between symmetric and asymmetric encryption, and when you might use each.

63. What is the importance of time complexity analysis, and how do you determine the Big O notation of an algorithm?

64. How can you ensure that your program handles errors gracefully?

65. What are design patterns and why are they helpful?

66. Can you explain the difference between lossy and lossless compression?

67. What is the idea behind using version control systems?

68. Explain the concept of inheritance and polymorphism in object-oriented programming.

69. What are some common software testing techniques, and why is testing important?

70. Explain what a distributed system is and what challenges arise when building one.

71. What is the CAP theorem, and how does it influence database design?

72. Describe the difference between a process and a thread.

73. What are the advantages and disadvantages of using microservices architecture?

74. Explain the concept of dependency injection.

75. What are the key principles of SOLID?

76. Describe the different types of joins in SQL (INNER, LEFT, RIGHT, FULL).

77. How does a content delivery network (CDN) work, and why is it useful?

78. Explain the concept of idempotence in API design.

79. What is the purpose of message queues in software architecture?

80. What is the difference between horizontal and vertical scaling?

81. How does the concept of 'time complexity' influence your decisions when choosing data structures and algorithms for a large-scale application?

82. Explain the CAP theorem. How have you navigated trade-offs between consistency, availability, and partition tolerance in past projects?

83. Describe a situation where you had to optimize a slow-performing query. What steps did you take to diagnose the problem and improve its performance?

84. What are some of the key differences between TCP and UDP, and when would you choose one over the other?

85. Explain the concept of 'deadlock' in concurrent programming. How can deadlocks be prevented or resolved?

86. What is the significance of normalization in database design? Describe different normalization forms and their benefits.

87. How does caching work? Can you describe different caching strategies (e.g., LRU, LFU) and their tradeoffs?

88. Explain how object-oriented programming principles like polymorphism and inheritance contribute to code maintainability and scalability.

89. What are design patterns, and how have you used them in your software development projects? Give specific examples.

90. Describe your experience with different types of data structures (e.g., trees, graphs, hash tables). How do you decide which data structure is most appropriate for a given problem?

91. What is the difference between authentication and authorization? How are these concepts implemented in web applications?

92. Explain the concept of 'virtualization'. What are some of the benefits and drawbacks of using virtual machines?

93. How do you approach debugging complex software systems? Describe some of the tools and techniques you use.

94. What is the purpose of using indexes in a database? How do indexes affect query performance?

95. Explain the different types of joins used in SQL (e.g., inner join, outer join). When would you use each type?

96. What are microservices, and what are the benefits and challenges of using a microservices architecture?

97. Describe the difference between symmetric and asymmetric encryption. When would you use each type?