Technical Interview Questions for Entry-Level Software Engineer/Developer

This document contains 50 of the most probable technical interview questions for entry-level software engineering positions. Questions are organized by category to help you prepare systematically.

Table of Contents

Programming Fundamentals

1. What is the difference between a compiler and an interpreter?

Key Points: Compiler translates entire code before execution, interpreter executes line by line.

2. Explain the difference between stack and heap memory.

Key Points: Stack for local variables/function calls, heap for dynamic memory allocation.

3. What are the different data types in programming languages you know?

Key Points: Primitive types (int, float, bool, char) vs reference types (objects, arrays).

4. What is the difference between pass by value and pass by reference?

Key Points: Value copies data, reference passes memory address.

5. Explain variable scope and lifetime.

Key Points: Global, local, block scope; when variables are created and destroyed.

Data Structures

6. What is an array and what are its advantages/disadvantages?

Key Points: Contiguous memory, O(1) access, fixed size limitation.

7. Explain the difference between array and linked list.

Key Points: Memory allocation, access time, insertion/deletion complexity.

8. What is a stack? Explain LIFO principle.

Key Points: Last In First Out, push/pop operations, use cases.

9. What is a queue? Explain FIFO principle.

Key Points: First In First Out, enqueue/dequeue operations, use cases.

10. What is a hash table/hash map? How does it work?

Key Points: Key-value pairs, hash function, collision handling, O(1) average access.

11. Explain the concept of a binary tree.

Key Points: Hierarchical structure, root/leaf nodes, left/right children.

12. What is the difference between a binary tree and binary search tree?

Key Points: BST ordering property, search efficiency.

Algorithms

13. What is Big O notation? Explain time complexity.

Key Points: Asymptotic analysis, common complexities (O(1), O(n), O(log n), O(n^2)).

14. Explain different sorting algorithms you know.

Key Points: Bubble sort, selection sort, merge sort, quick sort, time complexities.

15. What is the difference between BFS and DFS?

Key Points: Breadth-first vs depth-first traversal, use cases, implementation.

16. Explain binary search algorithm.

Key Points: Divide and conquer, O(log n) complexity, sorted array requirement.

17. What is recursion? Give an example.

Key Points: Function calling itself, base case, recursive case, stack usage.

18. What is dynamic programming?

Key Points: Breaking problems into subproblems, memoization, optimization.

Object-Oriented Programming

19. What are the four pillars of OOP?

Key Points: Encapsulation, Inheritance, Polymorphism, Abstraction.

20. Explain encapsulation with an example.

Key Points: Data hiding, private/public access modifiers, getters/setters.

21. What is inheritance? Types of inheritance.

Key Points: Parent-child relationship, code reusability, single/multiple inheritance.

22. Explain polymorphism with examples.

Key Points: Method overloading, method overriding, runtime/compile-time polymorphism.

23. What is abstraction? Difference between abstract class and interface.

Key Points: Hiding implementation details, contracts, when to use each.

24. What is method overloading vs method overriding?

Key Points: Same method name different parameters vs redefining in subclass.

Web Development

25. What is the difference between HTTP and HTTPS?

Key Points: Security, SSL/TLS encryption, port numbers.

26. Explain the HTTP request/response cycle.

Key Points: Client-server communication, request methods, status codes.

27. What are HTTP status codes? Explain common ones.

Key Points: 200 (OK), 404 (Not Found), 500 (Server Error), categories.

28. What is the difference between GET and POST requests?

Key Points: Data transmission, security, idempotency, caching.

29. What is REST? Explain RESTful principles.

Key Points: Representational State Transfer, stateless, uniform interface.

30. What is the DOM? How do you manipulate it?

Key Points: Document Object Model, tree structure, JavaScript manipulation.

31. Explain the box model in CSS.

Key Points: Content, padding, border, margin layers.

32. What is responsive web design?

Key Points: Multiple device compatibility, media queries, flexible layouts.

Databases

33. What is the difference between SQL and NoSQL databases?

Key Points: Relational vs non-relational, ACID properties, scalability.

34. Explain ACID properties in databases.

Key Points: Atomicity, Consistency, Isolation, Durability.

35. What is a primary key and foreign key?

Key Points: Unique identifier, referential integrity, relationships.

36. What are different types of SQL JOINs?

Key Points: INNER, LEFT, RIGHT, FULL OUTER joins, use cases.

37. What is database normalization?

Key Points: Reducing redundancy, normal forms (1NF, 2NF, 3NF).

System Design Basics

38. What is client-server architecture?

Key Points: Separation of concerns, request-response model, scalability.

39. Explain the concept of load balancing.

Key Points: Distributing traffic, high availability, horizontal scaling.

40. What is caching? Why is it important?

Key Points: Temporary storage, performance improvement, different cache levels.

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

Key Points: Scale out vs scale up, cost, reliability considerations.

Problem Solving & Coding

42. Write a function to reverse a string.

Example: Implement in your preferred language, discuss time/space complexity.

43. How would you find the largest element in an array?

Example: Iterate through array, track maximum, O(n) complexity.

44. Write a function to check if a number is prime.

Example: Check divisibility, optimization techniques, edge cases.

45. How would you remove duplicates from an array?

Example: Hash set approach, two-pointer technique, discuss trade-offs.

Software Engineering Practices

46. What is version control? Why is Git important?

Key Points: Code versioning, collaboration, branching, merging.

47. Explain the software development lifecycle (SDLC).

Key Points: Planning, analysis, design, implementation, testing, deployment.

48. What is the difference between unit testing and integration testing?

Key Points: Testing scope, isolation, test pyramid concept.

49. What are code reviews? Why are they important?

Key Points: Quality assurance, knowledge sharing, bug prevention.

General Technical Knowledge

50. How do you stay updated with new technologies and programming trends?

Key Points: Online resources, communities, continuous learning, practical projects.

Interview Tips

Preparation Strategy:

1. **Practice coding problems** on platforms like LeetCode, HackerRank
2. **Understand fundamentals** deeply rather than memorizing
3. **Practice explaining concepts** out loud
4. **Review your projects** and be ready to discuss technical decisions
5. **Prepare questions** to ask the interviewer about the role/company

During the Interview:

- **Think out loud** while solving problems
- **Ask clarifying questions** before jumping into solutions
- **Start with a simple approach** then optimize
- **Test your code** with examples
- **Admit when you don't know** something but show willingness to learn

Common Mistake to Avoid:

- Don't memorize answers - understand the concepts
- Don't jump straight into coding - discuss the approach first
- Don't give up easily - show your problem-solving process
- Don't forget to consider edge cases
- Don't be afraid to ask for hints if stuck

Good luck with your interview! Remember, the goal is not just to know the answers but to demonstrate your problem-solving ability, communication skills, and eagerness to learn.