1. What is polymorphism in C++ and why is it important?

2. Explain the concept of compile-time (static) polymorphism with examples.

3. Describe the concept of runtime (dynamic) polymorphism with examples.

4. What is the difference between static and dynamic polymorphism?

5. How is polymorphism implemented in C++?

6. What are pointers in C++ and how do they work?

7. Explain the syntax for declaring and initializing pointers.

8. How do you access the value pointed to by a pointer?

9. Describe the concept of pointer arithmetic.

10. What are the common pitfalls when using pointers?

11. How are pointers used with objects in C++?

12. Explain the process of dynamically allocating objects using pointers.

13. Provide an example of accessing object members using pointers.

14. What is the difference between a pointer to an object and a reference to an object?

15. How do you release dynamically allocated objects in C++?

16. What is the this pointer in C++ and what is its significance?

17. How is the this pointer used in member functions?

18. Explain how the this pointer can be used to return the current object.

19. What is a virtual function in C++ and why is it used?

20. Describe the syntax for declaring a virtual function.

21. Explain the concept of a vtable (virtual table) and its role in virtual functions.

22. What is a pure virtual function and how is it declared?

23. Provide an example of a class with pure virtual functions.

24. What are the implications of having pure virtual functions in a class?

25. How is polymorphism implemented using inheritance and virtual functions?

26. Provide an example of implementing polymorphism with base and derived classes.

27. Explain the concept of late binding in the context of polymorphism.

28. How does the compiler manage polymorphism in C++?

29. What is an abstract class in C++?

30. How do abstract classes differ from regular classes?

31. Explain the role of abstract methods in abstract classes.

32. Provide an example of defining and using an abstract class.

33. What are the benefits of using abstract classes in C++?

34. What is exception handling in C++ and why is it important?

35. Describe the syntax for throwing and catching exceptions in C++.

36. Explain the concept of try, catch, and throw blocks.

37. What is the role of the catch block in exception handling?

38. Provide an example of handling multiple exceptions in C++.

39. How does the throw keyword work in exception handling?

40. What is the purpose of the finally block in exception handling?

41. How do you create custom exception classes in C++?

42. What are templates in C++ and why are they useful?

43. Describe the syntax for defining a function template.

44. Provide an example of a function template that performs a generic operation.

45. What is a class template and how is it different from a function template?

46. Explain the syntax for defining a class template.

47. Provide an example of a class template that implements a generic data structure.

48. How do you instantiate a template class in C++?

49. What are the advantages of using templates over traditional class inheritance?

50. How do templates promote code reusability in C++?