Awesome — let's go!
Here's the **full 100-question C++ questionnaire** (level 5–10) — all in one big post, as you requested.
Answer key is at the end. 🧠

📄 **C++ Expertise Questionnaire (100 Questions)**

---

1. Which C++ feature enables a function to behave differently based on whether an argument is an lvalue or rvalue?

A) Function pointers
B) Perfect forwarding
C) Virtual functions
D) Operator overloading

---

2. What does std::move actually do?

A) Moves the memory content of an object
B) Marks an object as an rvalue reference
C) Frees the original object
D) Copies the object using move semantics

---

3. Which of the following statements about constexpr is FALSE?

A) constexpr functions can be evaluated at compile-time
B) constexpr variables must be initialized with constant expressions
C) All constexpr functions must be trivial
D) constexpr functions can have loops and conditions

---

4. In C++20, what does the concept feature primarily provide?

A) Static polymorphism
B) Dynamic type checking
C) Compile-time constraint checking
D) Better error messages for runtime bugs

---

5. What happens in this code?

```
struct A { A(); ~A(); };

void f() {
    A a;
    throw 42;
```

}

A) Compiler error
B) Undefined behavior
C) Destructor of a is called before propagating exception
D) Program terminates without cleanup

---

6. What happens when an exception is thrown inside a constructor?

A) Memory leak
B) Destructors of already constructed members are called
C) Undefined behavior
D) The object becomes invalid but remains allocated

---

7. What is SFINAE?

A) Runtime type checking
B) Compile-time technique to remove invalid template instantiations
C) Thread synchronization primitive
D) A type of memory allocation strategy

---

8. Difference between static inside a function and inside a class?

A) No difference
B) In a function, it persists; in a class, it's shared across instances
C) In a function, it's shared; in a class, it persists
D) Both persist but only in different scopes

---

9. How is a deleted function declared?

A) void foo() delete;
B) void foo() = delete;
C) delete void foo();
D) void foo() delete();

---

10. State of v after:

```
std::vector<int> v = {1,2,3};

for (auto& i : v) { i++; }
```

A) {1,2,3}
B) {2,3,4}
C) {1,2,3,4}
D) Compilation error

---

11. Which C++ cast is safest for polymorphic classes?

A) reinterpret_cast
B) static_cast
C) dynamic_cast
D) const_cast

---

12. Which of the following guarantees no memory overhead for empty classes?

A) Always true
B) True only if [[no_unique_address]] is used
C) Depends on the compiler optimization
D) Never true

---

13. volatile keyword in C++ indicates:

A) Variable will change unpredictably outside the program
B) Variable cannot be optimized
C) Variable is thread-safe
D) Both A and B

---

14. What is RVO (Return Value Optimization)?

A) A way to prevent dangling references
B) A compiler optimization to eliminate copy/move construction
C) A way to optimize heap allocations
D) A type-erasure technique

---

15. In C++20, std::span is:

A) A container
B) A view over contiguous memory
C) A dynamic array
D) A pointer wrapper

---

16. Which option best describes the rule of 5?

A) Copy constructor, move constructor, destructor
B) Copy constructor, move constructor, copy assignment, move assignment, destructor
C) Constructor, copy assignment, destructor
D) Constructor, destructor, operator=

---

17. What is alignment in C++?

A) Number of bytes between object instances
B) Number of bytes an object must start at in memory
C) Size of the object
D) Cache line size

---

18. Which method allows overriding a final function?

A) override
B) virtual
C) None
D) final

---

19. What happens if a pure virtual function is called in a constructor?

A) Program crashes
B) Undefined behavior
C) Calls base version if provided
D) Compile-time error

---

20. Which library provides std::shared_mutex?

A) <mutex>
B) <shared_mutex>
C) <thread>
D) <atomic>

---

21. When is the destructor of a thread_local variable called?

A) End of scope
B) Program exit
C) Thread exit
D) Undefined behavior

---

22. How many parameters can a variadic template accept?

A) Exactly 1
B) Any number
C) Only integers
D) None

---

23. Which of the following enables conditional compilation based on a type property?

A) static_assert
B) enable_if
C) constexpr if
D) All of the above

---

24. unique_ptr is NOT copyable because:

A) It uses deep copies
B) It deletes the copy constructor
C) It is designed to own a unique resource
D) B and C

---

25. What is the output?

```
std::string s1 = "abc";

std::string s2 = std::move(s1);
```

A) Both s1 and s2 are "abc"
B) s1 is "abc", s2 is "abc"
C) s1 is empty, s2 is "abc"
D) Compilation error

---

26. Which is a correct way to define a custom allocator?

A) Inherit from std::allocator
B) Provide allocate and deallocate methods
C) Provide new and delete operators
D) Inherit from std::memory_resource

---

27. In multithreading, what is a data race?

A) Multiple threads accessing same memory, at least one is a write, unsynchronized
B) Two threads running at the same speed
C) Priority inversion between threads
D) Cache invalidation between processors

---

28. Which of the following ensures memory visibility across threads?

A) std::atomic
B) volatile
C) thread_local
D) shared_ptr

---

29. What is type erasure?

A) Forgetting a type at runtime
B) Using templates to hide type
C) Removing type information at compile-time
D) Hiding implementation details of a type

---

30. What happens when you dereference an invalid iterator?

A) Undefined behavior
B) Throws std::out_of_range
C) Returns null
D) Segmentation fault always

Alright, let's continue and finish all 100 questions, same style, no gaps — let's roll:

C++ Expertise Questionnaire (continued)

---

31. What does emplace_back in std::vector do?

A) Inserts a copy at the end
B) Constructs an object in-place at the end
C) Moves an object to the end
D) Allocates new memory at the end

---

32. If a class has a move constructor but no copy constructor, what happens when you try to copy it?

A) Compiles fine, uses move constructor
B) Compile-time error
C) Runtime error
D) Undefined behavior

---

33. Which C++ feature most reduces the need for typedef?

A) auto
B) decltype
C) using
D) typedef struct

---

34. Which STL container guarantees constant-time insertion/removal at both ends?

A) std::list
B) std::vector

C) std::deque
D) std::array

---

35. If you call std::terminate(), what happens?

A) Graceful shutdown
B) No destructors are run
C) Stack unwinding occurs
D) catch(...) handler is invoked

---

36. Which best defines *alignment padding*?

A) Additional memory between variables for cache optimization
B) CPU cycles wasted on cache misses
C) Bytes added to satisfy type alignment requirements
D) None of the above

---

37. What is the default access specifier for a struct?

A) private
B) public
C) protected
D) depends on compiler

---

38. What is a lambda capture-by-move introduced in C++14?

A) Capturing variables by copying
B) Capturing references
C) Moving variables into the lambda
D) Disabling captures altogether

---

39. Which design pattern fits smart pointers?

A) Proxy
B) Decorator
C) Factory
D) RAII

---

40. When is it better to use std::vector<bool>?

A) Never
B) If you want bit-packing optimization
C) When working with strings
D) When implementing a priority queue

---

41. Which of the following is undefined behavior?

A) Reading uninitialized memory
B) Dereferencing a dangling pointer
C) Modifying const data
D) All of the above

---

42. std::async returns:

A) A std::thread
B) A std::future
C) A std::promise
D) A std::shared_future

---

43. What happens when you std::move a const object?

A) Moves normally
B) Copies instead
C) Compilation error
D) Move constructor called, but effectively a copy

---

44. Which feature replaces CRTP in certain cases in C++20?

A) Concepts
B) Ranges
C) Coroutines
D) Modules

---

45. In modern C++, raw new/delete is:

A) Recommended for control
B) Best replaced by smart pointers
C) Faster than smart pointers
D) Only valid inside templates

---

46. What are allocator-aware containers?

A) Containers that optimize allocation based on type
B) Containers that can accept a custom allocator
C) Containers that avoid memory allocation
D) Only vectors and strings

---

47. In template metaprogramming, std::integral_constant is used to:

A) Delay computation to runtime
B) Provide a compile-time constant
C) Replace constexpr
D) Calculate Fibonacci numbers

---

48. Which of these expressions is NOT a valid lvalue?

A) A variable
B) A dereferenced pointer
C) A literal like 5
D) An array element

---

49. In exception handling, stack unwinding means:

A) Stack grows rapidly
B) Stack is popped until a matching catch is found
C) Stack frames are copied
D) Stack frames are dumped to disk

---

50. In C++, a function template specialization is selected by:

A) Name matching only
B) SFINAE rules
C) Exact match on template parameters
D) Argument-dependent lookup

🔥 **51–100 continues immediately**

---

51. Which one is NOT a property of std::map?

A) Unique keys
B) Logarithmic lookup
C) Ordered elements
D) O(1) access

---

52. std::tuple is:

A) A replacement for std::pair
B) A fixed-size collection of heterogeneous elements

C) A dynamic array
D) A replacement for std::variant

---

53. How does std::variant store its alternatives internally?

A) As union
B) As array
C) As linked list
D) As std::tuple

---

54. What is a consequence of violating the one-definition rule (ODR)?

A) Compilation error
B) Link-time error
C) Runtime crash
D) Undefined behavior

---

55. Which feature allows multiple return types elegantly in C++17?

A) std::tuple
B) std::optional
C) Structured bindings
D) std::any

---

56. Which of these is NOT a C++ core guideline?

A) Use RAII
B) Prefer exceptions over error codes
C) Always use macros for constants
D) Prefer nullptr over NULL

---

57. How is "type deduction" in templates formally performed?

A) Pattern matching
B) Substitution
C) Type checking
D) Instantiation

---

58. std::optional internally uses:

A) A union
B) A pointer
C) A dynamic allocation
D) Undefined storage

---

59. Which is safer: pointer or std::optional<std::reference_wrapper<T>>?

A) Pointer
B) Optional reference
C) They are equivalent
D) Depends on use case

---

60. Which container has O(1) insertions, deletions, and access?

A) std::vector
B) std::deque
C) std::unordered_map
D) std::list

---

61. Purpose of std::allocator_traits?

A) Replace new
B) Abstract differences between allocators
C) Speed up memory allocation
D) Force inlining

---

62. C++ concept of *move-only type* means:

A) No copy allowed, only move
B) Only heap allocation allowed
C) No destructor
D) Only stack allocation allowed

---

63. Which C++ feature allows static reflection (C++23)?

A) constexpr
B) static_assert
C) reflexpr
D) co_await

---

64. Which is the most appropriate smart pointer for polymorphic types?

A) std::unique_ptr with custom deleter
B) std::shared_ptr always
C) std::weak_ptr
D) std::scoped_ptr

---

65. What does final keyword prevent?

A) Copying of object
B) Further inheritance
C) Move operations
D) Overloading operators

---

66. What guarantees resource release even during exceptions?

A) try/catch
B) RAII
C) goto
D) Virtual destructors

---

67. Which algorithm modifies a container in-place?

A) std::transform
B) std::accumulate
C) std::find
D) std::count_if

---

68. A noexcept function:

A) Can never throw
B) Can throw under conditions
C) May terminate if exception is thrown
D) B and C

---

69. Which syntax is correct to delete a move constructor?

A) A(A&&) delete;
B) A(A&&) = delete;
C) delete A(A&&);
D) delete(A(A&&));

---

70. *Allocator propagation* matters during:

A) Construction
B) Copy/move operations
C) Assignment
D) All of the above

🔥 **Home stretch! 71–100 below**

---

71. What is a "monostate" used for in std::variant?

A) Represent a "null" type
B) Force heap allocation
C) Distinguish among multiple voids
D) Always causes compilation error

---

72. A lambda can be mutable if:

A) It captures by value and needs to modify captured variables
B) It captures by reference only
C) It doesn’t capture anything
D) It's constexpr

---

73. C++'s two-phase name lookup mainly impacts:

A) Templates
B) Virtual functions
C) Const functions
D) Function overloading

---

74. In coroutines, co_yield is used for:

A) Return values
B) Pause/resume
C) Exception throwing
D) Generator-like behavior

---

75. When is a virtual destructor necessary?

A) Always
B) For classes intended for inheritance
C) Never, C++ handles it
D) Only in pure abstract classes

---

76. What is *aggregate initialization*?

A) Initialization of a non-copyable type
B) Initialization using braces {} without constructors
C) Using new operator
D) Using placement new

---

77. Which principle is violated if you publicly inherit from a concrete class?

A) Liskov Substitution
B) DRY
C) SOLID
D) Interface Segregation

---

78. What allows implicit narrowing conversions?

A) Uniform initialization {}
B) C-style casting
C) static_cast
D) dynamic_cast

---

79. How does std::weak_ptr avoid cycles?

A) By sharing ownership
B) By breaking ownership without deallocating
C) By locking
D) By reference counting independently

---

80. C++ module system helps with:

A) Better encapsulation
B) Faster compilation
C) Reduced header dependencies
D) All of the above

---

81. Which iterator category supports random access?

A) InputIterator
B) BidirectionalIterator
C) RandomAccessIterator
D) OutputIterator

---

82. In C++, an *incomplete type* is:

A) A type with a declaration but no definition
B) A type with private members
C) A forward-declared template
D) A dependent type

---

83. typeid operator requires:

A) RTTI enabled
B) Concepts
C) Polymorphism always
D) Overloaded operator==

---

84. What does [[nodiscard]] attribute mean?

A) Don't discard function return value
B) Function must throw
C) Function must be constexpr
D) Prevent recursion

---

85. Which one is a container adaptor?

A) std::list
B) std::vector
C) std::stack
D) std::map

---

86. Use std::byte instead of:

A) unsigned char
B) char*
C) void*
D) bool

---

87. A forwarding reference is:

A) T&
B) T&& in a non-template context
C) T&& where T is a template parameter
D) const T&

---

88. Which type of cast can *downgrade* constness?

A) static_cast
B) dynamic_cast
C) const_cast
D) reinterpret_cast

---

89. In multithreading, "false sharing" occurs when:

A) Threads share a cache line unintentionally
B) Mutexes are shared incorrectly
C) Memory is fragmented
D) Alignment is misused

---

90. std::variant throws std::bad_variant_access when:

A) Wrong alternative accessed
B) Variant is empty
C) Move operation fails
D) Copy operation fails

---

91. constexpr variable must:

A) Be runtime evaluatable
B) Be compile-time evaluatable
C) Be a pointer
D) Have a destructor

---

92. Which is not a guaranteed sequence point?

A) Function call
B) Assignment
C) Logical AND &&
D) Construction

---

93. std::scoped_lock is better than manual lock/unlock because:

A) Faster
B) Safer (exception-safe)
C) Supports deadlock avoidance
D) B and C

---

94. If two threads call operator++ on a shared int without locks:

A) Always safe
B) Causes a data race
C) Slower
D) Compiler error

---

95. An inline function is:

A) Always inlined
B) A hint to the compiler

C) A requirement for linker
D) Runtime optimization

---

96. If you violate a concept constraint:

A) Compile-time error
B) Runtime error
C) Linker error
D) Undefined behavior

---

97. Which guarantees strong exception safety?

A) Basic rollback
B) No resource leaks
C) Transactional behavior
D) noexcept

---

98. In C++, what is a dependent name?

A) Name depending on template parameter
B) Name that's overloaded
C) Name that's shadowed
D) Friend function name

---

99. What's true about function object vs lambda?

A) Lambda is syntactic sugar for function object
B) Lambdas can't capture
C) Functors are slower
D) Functors need inheritance

---

100. co_return in a coroutine:

A) Ends coroutine with a result
B) Throws an exception
C) Blocks coroutine
D) Suspends coroutine

✅ **ANSWER KEY (Short Version)**

1. B
2. B

3. C
4. C
5. C
6. B
7. B
8. B
9. B
10. B
11. C
12. B
13. D
14. B
15. B
16. B
17. B
18. C
19. B
20. B
21. C
22. B
23. D
24. D
25. C
26. B
27. A
28. A
29. D
30. A
31. B
32. B
33. C

34. C
35. B
36. C
37. B
38. C
39. D
40. B
41. D
42. B
43. D
44. A
45. B
46. B
47. B
48. C
49. B
50. C
51. D
52. B
53. A
54. B
55. C
56. C
57. B
58. A
59. B
60. C
61. B
62. A
63. C
64. A

65. B
66. B
67. A
68. D
69. B
70. D
71. A
72. A
73. A
74. D
75. B
76. B
77. A
78. B
79. B
80. D
81. C
82. A
83. A
84. A
85. C
86. A
87. C
88. C
89. A
90. A
91. B
92. D
93. D
94. B
95. B

96. A
97. C
98. A
99. A
100. A

✅ **Done!**

- If you want, I can also give you a scale (e.g., 90+ = expert) to map your score.
- Or help you analyze if you're 6/10, 7/10, 9/10 based on how you perform!

Would you like me to give you a grading scale next? 🎯
(Like: what score = 7/10, what score = 9/10, etc?)