C++ FUNCTION POINTERS — COMPLETE REVISION SHEET (SDE-1 → SDE-2+)

1. Function Pointer
Stores address of a free/static function. No object, no this pointer.
Syntax: return_type (*fp)(params);

2. Calling
fp(args); or (*fp)(args);

3. Use Cases
Callbacks, runtime behavior, plugins, embedded systems, performance paths.

4. Passing Function Pointers
Used for callbacks and strategy pattern.

5. Arrays of Function Pointers
Used in jump tables, state machines, command dispatch.

6. Type Safety
Signature must match exactly or UB occurs.

7. Lambdas
Non-capturing lambdas convert to function pointers. Capturing lambdas do not.

8. Function Pointer vs std::function
FP: zero overhead, limited. std::function: flexible, slight overhead.

9. Member Functions
Require implicit this pointer, cannot be stored in normal FP.

10. Pointer to Member Function
Syntax: void (A::*fp)(int); Called via object.

11. Static Member Functions
Behave like free functions; can use FP.

12. Virtual Functions
PMF still respects virtual dispatch; does not bypass vtable.

13. Memory & ABI
FP: direct code address. PMF: compiler-dependent (may include offsets).

14. Common Pitfalls
Missing parentheses, signature mismatch, null calls, FP vs PMF confusion.

Final Interview One-Liner
A function pointer stores a callable code address without object context, while member functions require an implicit this pointer and different invocation semantics.