C++ Pointers and Smart Pointers - Simple Definitions & Scenarios

Pointer

A pointer is a variable that stores the memory address of another variable. It allows direct access and manipulation of memory.

Common Scenario: Storing the address of an integer and accessing its value through the pointer.

Dangling Pointer

A dangling pointer is a pointer that refers to memory that has already been freed or is invalid.

Common Scenario: Returning the address of a local variable from a function; once the function ends, the memory is invalid.

Null Pointer

A null pointer is a pointer that does not point to any valid memory address. It is often used as a sentinel value.

Common Scenario: Setting a pointer to nullptr to indicate it is not currently assigned to any object.

Smart Pointers

Smart Pointer

A smart pointer is an object that acts like a pointer but also manages the lifetime of the object it points to. It automatically deletes the object when it is no longer needed.

Common Scenario: Using a smart pointer to manage a dynamically allocated object so you don’t have to manually delete it.

unique_ptr

A unique_ptr owns an object exclusively. Only one unique_ptr can point to an object at a time. When the unique_ptr goes out of scope, the object is automatically destroyed.

Common Scenario: Managing a file handle or socket that should not be shared with others.

shared_ptr

A shared_ptr allows multiple smart pointers to share ownership of the same object. The object is destroyed when the last shared_ptr is gone.

Common Scenario: Multiple threads sharing access to a configuration object.

weak_ptr

A weak_ptr is a non-owning reference to an object managed by shared_ptr. It does not affect the lifetime of the object and is used to break reference cycles.

Common Scenario: A cache holding weak references to objects so they can be destroyed when no longer used elsewhere.

Comparison Table

Type	Ownership	Auto Delete	Common Use
Pointer	Manual	No	General memory access
unique_ptr	Exclusive	Yes	Exclusive resource management
shared_ptr	Shared	Yes	Shared resource management
weak_ptr	None	No	Break circular references