C++ Overlooked Concepts

RAII (Resource Acquisition Is Initialization)

Concept: A programming idiom used to manage resources such as memory, file handles, and network connections. The idea is that resources are tied to the lifetime of objects, ensuring that they are properly released when the objects go out of scope.

Slicing Problem

Concept: Occurs when an object of a derived class is assigned to an object of a base class, causing the derived class's additional attributes to be "sliced" off.

Solution: Use pointers or references to base class objects to avoid slicing.

Rule of Three/Five/Zero

Rule of Three: If a class needs a custom destructor, copy constructor, or copy assignment operator, it likely needs all three.

Rule of Five: Extends Rule of Three to include the move constructor and move assignment operator.

Rule of Zero: If possible, avoid defining any of these special member functions and rely on the compiler-generated ones, using RAII and smart pointers.

Default Member Initialization

Concept: C++11 introduced the ability to initialize class members directly within the class definition.

Lambda Expressions and Closures

Concept: Introduced in C++11, lambdas are anonymous functions that can capture variables from their enclosing scope.

Uniform Initialization Syntax

Concept: C++11 introduced a uniform way to initialize variables and objects using curly braces {}.

Advantages: Avoids issues like narrowing conversions and provides a consistent syntax for all types of initialization.

nullptr

Concept: C++11 introduced nullptr as a type-safe null pointer constant to replace the old NULL macro.

std::chrono for Time Handling

Concept: Provides a set of types for representing time points, durations, and clocks.

Strongly Typed Enums (enum class)

Concept: C++11 introduced strongly typed enums, which provide better type safety and scoping than traditional enums.

Type Traits and std::enable_if

Concept: Part of the type traits library used to enable or disable template instantiations based on compile-time conditions.

Move Semantics

Concept: Introduced in C++11 to optimize resource management by transferring resources from temporary objects (rvalues) to permanent ones.

EBO (Empty Base Optimization)

Concept: Allows the compiler to optimize away the memory footprint of empty base classes.