Modern C++

**1. Smart Pointers (C++11)**

Instead of manually managing memory with new and delete, smart pointers like std::unique\_ptr and std::shared\_ptr help prevent memory leaks and ensure automatic cleanup.

**2. Move Semantics & Rvalue References (C++11)**

This optimizes resource management by allowing objects to "move" instead of deep-copying, making programs faster and reducing unnecessary allocations.

**3. Lambda Expressions (C++11)**

They allow concise anonymous functions and are especially handy for functional programming and callback mechanisms.

**4. Range-Based For Loops (C++11)**

Easier iteration over containers without dealing with iterators directly.

**5. constexpr (C11, C14)**

Enables compile-time evaluation of functions and expressions for performance boosts.

**6. Type Deduction (auto and decltype) (C++11)**

Reduces verbosity in type declarations, making code more readable.

**7. Threads & Concurrency (std::thread) (C11, C17)**

Standardized threading makes it easier to write parallel code.

**8. std::optional (C++17)**

A safer alternative to using pointers when a value might be absent.

**9. std::variant (C++17)**

A modernized replacement for union, helping with type-safe alternatives.

**10. Concepts (C++20)**

Allows constraints on template parameters, making generic programming clearer and more robust.

**11. Ranges Library (C++20)**

Makes working with containers and sequences more declarative and intuitive.

**12. Coroutines (C++20)**

Simplifies asynchronous programming by eliminating callback hell.

**13. Modules (C++20)**

Improves compilation speed and prevents header file issues.

Would you like me to focus on any of these concepts in more detail or maybe write some sample code? 🚀