Introduction: The role of a raw pointer in legacy C++ is often ambiguous. It could represent a single object on the heap that must be deleted. It could also represent an array of objects, a shared object, or one not on the heap. In modern C++, smart pointers can help to alleviate that problem. However, raw pointers are still common in code, most notably within legacy systems or when programmers intend to stay resource-efficient. This project wants to use the new ideas introduced with the C++ Core Guidelines to help clear up this confusion of raw pointer roles in modern C++14 by giving developers easy-to-use refactoring tools.

Approach/Technologies: The C++ Core Guidelines that were introduced at CppCon15 aim to provide a set of guidelines on how to write better modern C++. They include several topics on how to handle memory and raw pointers. Following the guidelines allows for C++ code to be checked by static analysis tools and catch many semantic programming errors, such as out-of-bounds accesses. Many of the guidelines make use of the Guidelines Support Library (GSL). The GSL provides several types that should be used in context of raw pointers in code. The GSL types ease static analysis with respect to raw pointers and often contain additional functionalities that help prevent errors with minimal additional overhead.

Result: The result of this project is a plug-in for the Cevelop IDE that helps developers to adhere to the C++ Core Guidelines with pointers. The plug-in contains code checkers that mark problematic code using raw pointers and provides refactorings towards the GSL types marking their role. We have tested our plug-in on a real-world code base to ensure a better user experience and more reliable functionality.