Combining Static Analysis with Dynamic Testing to Verify Code
VectorCAST/Lint is a static analysis tool for software developers that will look across your entire program that may consist of many C and/or C++ source modules to find bugs, glitches, inconsistencies, non-portable constructs, and much more, so you can find and fix bugs more quickly, and more economically, prior to compiling and executing.

Overview of Advantages
> Out-of-the-box MISRA support for C and C++
> Static Variable Tracking
> Thread Analysis
> Dimensional Analysis
> Stack Usage
> Source Code Browser with Message Annotations
> Message Rationale and Concept Browser
> Message Filtering
> Graphical Configuration of Analysis
> Customizable Reporting
> Macro Scavenging

Combining Static Analysis and Dynamic Testing
VectorCAST/Lint utilizes the powerful PC-Lint and FlexeLint analysis engine from Gimpel Software and has been extended to support the extensive list of embedded compiler environments currently integrated with the VectorCAST dynamic testing product line. VectorCAST/Lint static analysis provides a perfect complement to the current VectorCAST tools for dynamic testing. The dynamic testing component of VectorCAST provides users with automated unit and integration testing, including code coverage and regression test management capability. It is important to note that the static analysis and dynamic testing are completely exclusive testing activities. The dynamic tools used for unit/integration test and code coverage, have no dependence on the static analysis for information to construct executable test harnesses. This alleviates the need to run the static analysis every time a change is made to a file when doing unit, integration, and regression testing.
Syntax Checking
Compilers do an excellent job of finding syntax errors and most will produce warnings. But PC-lint and FlexeLint begin where compilers leave off. VectorCAST/Lint contains numerous features such as precision tracking, initialization checking, value tracking, strong type checking and macro analysis that compilers do not have. Also VectorCAST/Lint looks across a set of modules to find intermodule inconsistencies and redundancies. This, compilers do not do.

Reduces False Positive Rates
One of the issues with static source code analysis tools is the amount of “noise” or “potential” problems that might be present in a piece of source code. Steps have been taken to ensure that our users are not inundated by a mass of insignificant messages in which they may have no particular interest. All messages are carefully crafted to make use of all available information so that they have an inherently high wheat-to-chaff ratio.
We have four levels of message (Error, Warning, Informational and Elective Note) and distinguish between two kinds of header (library headers vs. project headers). Messages can be suppressed by number, by symbol, by macro, by line and by header type. Through a selection of options, you in effect tailor your own subset of the C/C++ language.

Supports Coding Standards
VectorCAST/Lint is preconfigured to provide out-of-the box checking for the MISRA C:2004, MISRA C++:2008, and MISRA C:2012 standards. These standards recommend the use of a restricted subset of constructs for the C and C++ languages, with the goal being a safer and more maintainable use of the language. Areas of code that are non-conforming to the standard's built-in-rules are highlighted in code analysis reports. VectorCAST/Lint features enhanced MISRA checking that includes the detection of recursion, support for the MISRA 2 'underlying type' concept, determination of side effects for functions and MISRA C++ support.

More information: www.vector.com/vectorcast