**KEY BENEFITS**

- Reduces cost while improving time-to-market
- Lowers the risk of program failure
- Identifies coding problems early in the development cycle
- Ensures quality code and coding standard compliance
- Accelerates and refocuses the code review process and improves teamwork
- Builds programmer awareness - fosters best practices
- Enhances reliability, portability, and maintainability
- Allows instant and repeatable code audits and reviews

**KEY FACTS**

- Identifies software defects and non-compliance issues
- Recognizes issues caused by dangerous, overly complex and non-portable language usage
- Warns on issues that are not reported by compilers or other tools
- Incorporates advanced message suppression and incremental analysis capability
- Combines diagnostics and code visualizations with metrics into easy-to-understand flexible reporting

**QA·C Summary**

Market leader QA·C is designed for engineers to use at the coding phase of a project to ensure the development of high integrity code. QA·C can be used to detect and prevent defects and ensure compliance to coding standards, on existing or new code at any stage of the development process. It is fast, non-disruptive, easy-to-use, and adds value to any size development environment.

QA·C detects and reports on dataflow problems, software defects, language implementation errors, inconsistencies, dangerous usage and coding standard violations quickly and efficiently. By adhering to the “early & often” philosophy, software defects are identified at creation resulting in simplified development lifecycle and reduced costs and cycle time. QA·C provides an efficient, robust, and fully automated environment to introduce and enforce coding standards. Various reports on detected defects as well as metric calculations can be created, and personalized. QA·C provides the ability to monitor and limit complexity, enabling the development of testable and maintainable code.

**Example QA·C Messaging Areas**

**Advanced Defect Prevention**

QA·C detects buffer overflows, division by zero, dead code, unreachable code and much more by linking the in-depth language analysis performed by QA·C with a state of the art Satisfiability Modulo Theories (SMT) solver. Wide scope of checking includes: inter-dependency between variables, pointer aliasing, bi-directional suspicious variable usage analysis and loop analysis (first, last and intermediate iteration analysis) and checking of arguments to Standard C Library functions.
Undefined Behavior
There are many constructs in the C language which are not explicitly classified as incorrect but which can result in unpredictable behavior. These issues range from the ones familiar to most developers to the obscure and less well known issues that are explicitly stated in the ISO C standards. QA-C is highly effective in identifying these problems (which compilers often ignore).

Constraint Violations
Misuse of the language – 163 issues described in the language standards where the syntax is correct but the code is semantically illegal. Compilers are often less than rigorous in identifying such errors.

Cross Module Analysis
Link time errors and anomalies such as multiply defined objects / functions and incompatible declarations – problems which a linker will seldom identify.

Portability
It is vitally important to be aware of implementation defined language features and language extensions if code is to behave consistently across different compilers and platforms.

Type Conversions
The ability to perform automatic conversion between arithmetic types is a freedom which can easily be abused in the C language. QA-C supports a comprehensive range of messages which identify questionable or dangerous implicit type conversions which can result in loss of data, sign information or precision.

Redundancy
QA-C detects many different types of redundant construct, ranging from code that is simply never used to code that does nothing. QA-C identifies unused variables, unused functions, unused parameters and code with no side-effects. Using its sophisticated dataflow engine, QA-C can detect dead code, unreachable code, redundant assignments and initializations as well as redundant operations (such as multiplying by one or adding zero).

Coding Standard Rules
QA-C generates many messages highlighting language usage which is legal but possibly unwise or dangerous. As with other messages, these can be enabled or disabled according to the requirements of the adopted coding standard.

Identifiers
Naming conventions for identifiers can be checked against a combination of attributes such as linkage, scope, storage duration and type. Rules are user configurable using regular expressions.
QA·C’s Analysis and Reporting Features

QA·C’s powerful GUI delivers a contextual drill-down environment linked to a deep knowledge base. QA·C explains why problems it discovers need to be corrected and then provides detailed examples of how to fix them.

Metrics:
- Project based
- File based
- Function based
- User definable metrics
- Warning on metric thresholds

Configurable HTML Reports
- Compliance
- Metrics
- Code review
- Suppression
Key Product Features

- Interactive GUI
- Contextual Messaging
- Online Help and Knowledge Base
- Coding Standard Specific Messaging
- Summary and Detailed Reports
- IDE Integrations
- More than 1,650 Selectable Messages
- C Language Specific Parsing Engine
- Cross Module Analysis (Link Time Checking)
- Semantic Error Detection
- Dataflow Error Detection
- Defect Prevention Configuration
- Close Name Analysis
- Comment Based Message Suppression
- Suppression Auditing
- Incremental Analysis
- No Limit on Project Size or SLOC
- Reprise License Management
- Extensive Compiler Support
- Naming Convention Checker
- Support for UTF-8 and Kanji Text

Supported Platforms
- Windows 7 (32 & 64-bit),
- Linux RHEL 5 (32 & 64-bit)
- Solaris 10 (64-bit)

Deployment Options
- New Cross Platform GUI
- Command Line
- Makefile Integration
- Tightly integrated with Eclipse™ based IDE’s (e.g. CodeWarrior and Windriver Workbench™)
- Model-Driven Development Environments (e.g. Rhapsody™)
- Continuous Integration Environments (e.g. Jenkins™)
- Cross Platform Analysis Project Management

Compliance Modules
- MISRA C:2004
- MISRA C:2012

PRQA

Established in 1985, PRQA, ISO 9001 and TickIT certified, is recognized throughout the industry as a pioneer in static analysis, championing automated coding standard inspection and defect detection, delivering its expertise through industry-leading software inspection and standards enforcement technology.

PRQA’s industry-leading tools, QA·C, QA·C++ and QA·Verify, offer the closest possible examination of C and C++ code. All contain powerful, proprietary parsing engines combined with deep accurate dataflow which deliver high fidelity language analysis and comprehension. They identify problems caused by language usage that is dangerous, overly complex, non-portable or difficult to maintain. Plus, they provide a mechanism for coding standard enforcement.

PRQA has corporate offices in UK, USA, India and Ireland, complemented by a worldwide distribution network.

Contact Us

For further information regarding PRQA products and consulting services, please contact PRQA via your local sales representative, or directly at: info@programmingresearch.com

© Programming Research Ltd 2015. V1.8/01.15

SGS-TÜV Saar has certified QA·C with MISRA C Extended as “usable in the development of safety related software” for the key safety critical standards, IEC 61508, ISO 26262, EN 50128, IEC 60880 and IEC 62304.