

# Welcome to Insure++

Parasoft Insure++ is an automated runtime application testing tool for C and C++ applications that detects elusive errors, such as memory corruption, memory leaks, memory allocation errors, variable initialization errors, variable definition conflicts, pointer errors, library errors, I/O errors, and logic errors.

## How Does Insure++ Work?

Insure++ performs static analysis at compile-time, followed by sophisticated dynamic analysis at runtime. Using a unique set of patented technologies, Insure++ develops a comprehensive knowledge of the software and all of its elements under test. During compilation, Insure++ reads and analyzes the source code, then inserts test and analysis functions around every line of source code. Insure++ builds a database of all program elements and checks each data value and memory reference against its database at runtime to verify consistency and correctness. For a quick test, re-link your application against Insure++'s runtime library and run the program as you normally would. For a deeper look, instrument your code to zoom in on errors and perform the most thorough testing possible.

Insure++ checks all types of memory references, including those to static (global), stack, and shared memory. It can find memory corruption and memory leaks, as well as errors allocating and freeing dynamic memory. Insure++ also checks third-party libraries and functions. Testing with Insure++, you can automatically find such errors as string manipulation errors, operations on uninitialized pointers, operations on pointers to unrelated data blocks, invalid pointer operations, incompatible variable declarations, and mismatched variable types.

To help you optimize dynamic memory usage and maximize test suite coverage, Insure++ includes two complementary tools: Parasoft Inuse and Parasoft TCA. Inuse is a graphical utility that lets you watch a program allocate and free dynamic memory blocks, helping you understand the memory usage patterns of algorithms and optimize their behavior. TCA shows test coverage analysis for an application by determining how many files, functions, and statements have been executed, giving you an idea of the overall quality of testing.

## Pinpointing Programming Errors

Subtle memory corruption errors and dynamic memory problems often don't crash the program or cause it to give incorrect answers until the program is delivered to customers and they run it on their systems. Even if Insure++ doesn't find any problems in your programs, running it gives you the confidence that your program doesn't contain any errors. Of course, Insure++ can't possibly check everything that your program does. However, its checking is extensive and covers every class of programming error, including:

- Memory corruption due to reading or writing beyond the valid areas of global, local, shared, and dynamically allocated objects.
- Operations on uninitialized, NULL, or "wild" pointers.
- Memory leaks.
- Errors allocating and freeing dynamic memory.
- String manipulation errors.
- Operations on pointers to unrelated data blocks.
- Invalid pointer operations.
- Incompatible variable declarations.
- Mismatched variable types in printf and scanf argument lists.

When Insure++ finds a problem, it reports the name of related variables, the line of source code containing the error, a description of the error, and a stack trace.

## Maximizing Test Suite Coverage with TCA

The Total Coverage Analysis (TCA) tool works hand-in-hand with Insure++ to show you which parts of code you've tested and which you've missed. With TCA, you can stop wasting time testing the same parts of code over and over again and start exercising untested code instead. TCA shows test coverage analysis for an application by determining how many files, functions, and statements have been executed, giving you an idea of the overall quality of testing. TCA provides the following coverage information:

- Overall Summary: Shows the percentage covered at the application level (i.e., summed over all program entities).
- Function Summary: Displays the coverage of each individual function in the application.
- Block Summary: Displays the coverage broken down by individual program statement blocks.

Unlike some other coverage analysis tools that only work on a line-by-line basis, TCA is able to group your code into logical blocks. A block is a group of statements that must always be executed as a group. Advantages of using blocks instead of lines include:


- Lines of code which have several blocks are treated separately.
- Grouping equivalent statements into a single block reduces the amount of data you need to analyze.

By treating labels as a separate group, you can actually detect which paths have been executed, in addition to which statements.

## Understanding and Optimizing Dynamic Memory Usage with Inuse

Inuse is a graphical "memory visualization" tool that helps you determine where unseen leaks and other memory abuses may be hurting your program. Inuse allows you to watch how your program allocates and frees dynamic memory blocks, in real-time. Inuse will help you to better understand the memory usage patterns of algorithms and how to optimize their behavior. With Inuse, you'll have a clear understanding of how your program actually uses (and abuses) memory. You can use Inuse to:

- See how much memory an application uses in response to particular user events.
- Compare an application's overall memory usage to its expected memory usage.
- Detect the most subtle memory leaks, which can cause problems over time.
- Look for memory fragmentation to see if different allocation strategies might improve performance.
- Identify other common memory problems, including memory blowout, memory overuse, and memory bottlenecks.
- Analyze memory usage by function, call stack, and block size.

 **Insure++ does not support managed C++.**