Runtime analysis tool C-RUN

Ensuring code quality through runtime analysis

C-RUN, our easy-to-use add-on tool, performs runtime analysis directly in the development environment. It checks for arithmetic, bounds, and heap integrity issues, pinpointing what went wrong and where, based on customer feedback.

Key Highlight Features

- Analysis of C and C++ code
- Intuitive and easy-to-use settings
- Unique optimizations of test instrumentation minimizes code size overhead
- Comprehensive and detailed runtime error information
- Call stack information provided for each error found
- Code correlation and graphical feedback in editor
- Flexible error filter management
- Bounds checking to ensure accesses to arrays and other objects are within boundaries
- Buffer overflow detection
- Detection of value changes when casting between types
- Checks for overflow and wraparound in computations
- Discovery of bit losses in shift operations
- Heap and memory leaks checking
- Available as an add-on product for: IAR Embedded Workbench for Arm, version 7.20 and forward IAR Embedded Workbench for RX, version 3.10 and forward

Arithmetic issues, Bounds issues and Heap checking

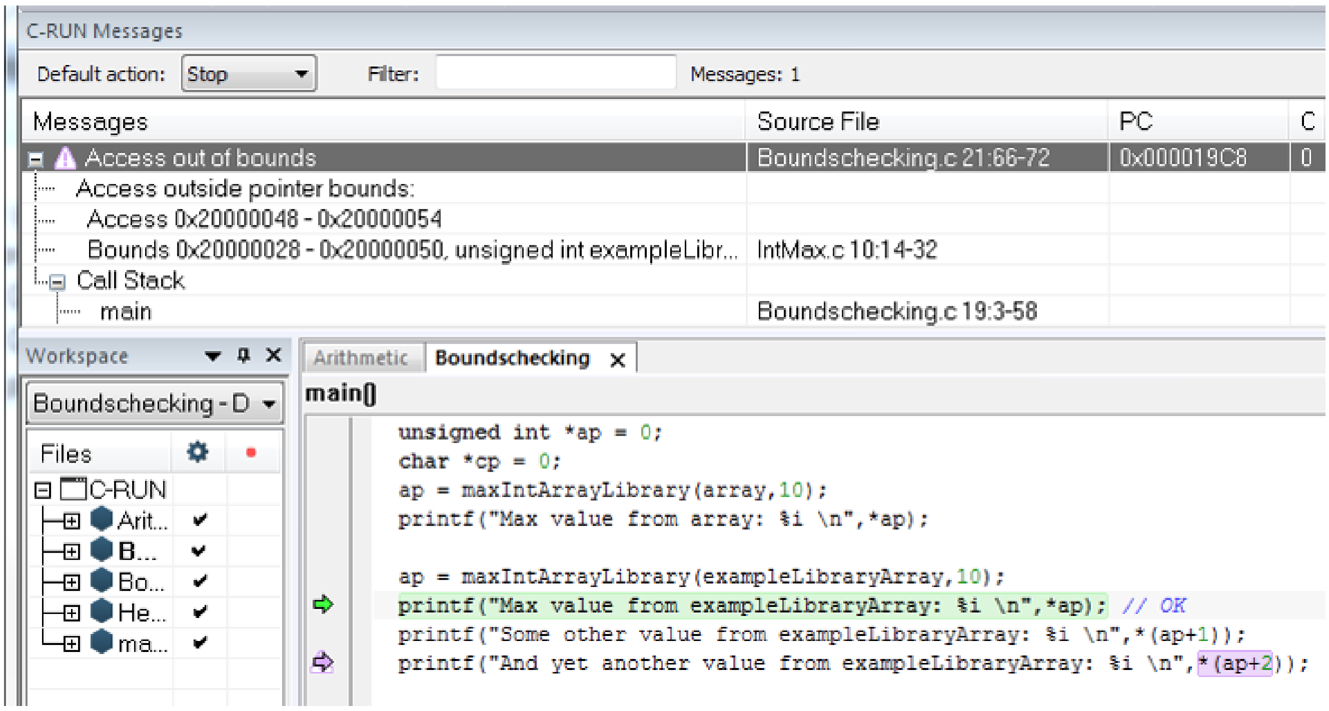
Arithmetic issues include overflow, wraparound, conversion errors, division by zero, and missing default labels in switch statements. These errors can be detected by inserting instrumentation code where errors might occur. At runtime, a compiler can insert checks to report issues, though this increases code size proportionally to the number of operations being checked.

Bounds issues occur when accessing memory outside defined limits, such as arrays or pointers to objects. Advanced bounds checkers can detect if a pointer has been modified to reference an invalid memory location, ensuring safe memory access.

Heap checking ensures the heap's integrity and prevents memory leaks. Efficient heap checks rely on library implementations, and leveraging compiler internals can help optimize this process. Integrity checks are usually performed during malloc, free, and similar calls in both C and C++. For large heaps, performance can be impacted, so controlling check frequency is essential for certain applications

Convenient runtime analysis and error checking with C-RUN

C-RUN is an extension to IAR Embedded Workbench and is available for Arm and RX. C-RUN is designed to be a natural part of your development workflow, when working in a traditional edit/build/debug cycle, running unit tests or doing integration tests. C-RUN provides you with extremely valuable feedback already as soon as the first iteration of code is about to be taken for a test drive. Thanks its tight integration into IAR Embedded Workbench, C-RUN can be part of the daily work for any developer.



Messages	Source File	PC	C
Access out of bounds	Boundschecking.c 21:66-72	0x000019C8	0
Access outside pointer bounds:			
Access 0x20000048 - 0x20000054			
Bounds 0x20000028 - 0x20000050, unsigned int exampleLibr...	IntMax.c 10:14-32		
Call Stack			
main	Boundschecking.c 19:3-58		

```
unsigned int *ap = 0;
char *cp = 0;
ap = maxIntArrayLibrary(array,10);
printf("Max value from array: %i \n",*ap);

ap = maxIntArrayLibrary(exampleLibraryArray,10);
printf("Max value from exampleLibraryArray: %i \n",*ap); // OK
printf("Some other value from exampleLibraryArray: %i \n",*(ap+1));
printf("And yet another value from exampleLibraryArray: %i \n",*(ap+2));
```

IAR Embedded Workbench

IAR Embedded Workbench is a complete C/C++ development toolchain for embedded applications, offering top-tier code quality, optimized performance, and extensive debugging tools. C-RUN is fully integrated with the IAR Embedded Workbench IDE, which helps developers to ensure their code is safe and of high quality at an early stage, which also aids companies to shorten their time to market as impact of errors further down the line might be very time consuming and expensive.

Do need to ensure your code is safe, secure and high-quality?

We're here to support you every step of the way, from project start to product life-cycle completion.

Contact your local IAR team to get started: iar.com/contact!