RogueWave
SOFTWARE
Accelerating Great Code

CODEDYNAMICS

Faster fault isolation, advanced thread debugging, and powerful reverse debugging for your enterprise apps

Debug more threads and processes than ever before

Within a single window, you get complete control over multiple individual threads and processes covering all aspects of program execution and data — running, stepping, and halting line-by-line through code within a single thread or within arbitrary groups of processes or threads.

Reduce fault isolation time with reverse debugging

Work backwards from the crash or failure using reverse debugging, isolating the root cause faster by eliminating the need to repeatedly breakpoint and restart the application.

Understand data quickly

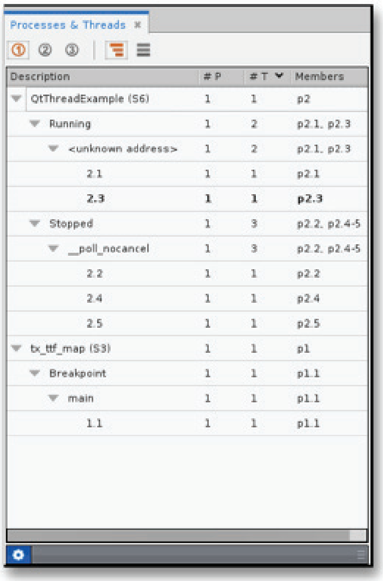
With advanced data debugging features, CodeDynamics lets you inspect, modify, and test complex data types (such as STL collection classes and multidimensional arrays) at runtime.

CodeDynamics is an easy to use and intuitive dynamic analysis tool for inspecting, debugging, and fixing your C and C++ applications. Built for enterprise customers that demand performance, scalability, and high-availability, CodeDynamics offers deep thread control, unique reverse debugging, and advanced data analysis features to simplify and shorten the process of debugging and optimizing complex applications.

Manage the challenges presented by concurrency, parallelism, and threads

Today's data-driven applications and workloads demand multithreaded and multicore architectures, presenting significant development challenges. CodeDynamics is a source code debugger for troubleshooting these complex multithreaded or multiprocess programs.

- Operate with equal ease on single thread/process or with groups of thread/process
- Set breakpoints on processes and even down to the individual thread level
- Control the execution of threads or processes individually or in groups
- View program data and threads/processes in a simple tree view



Description	# P	# T	Members
QtThreadExample (S6)	1	1	p2
Running	1	2	p2.1, p2.3
<unknown address>	1	2	p2.1, p2.3
2.1	1	1	p2.1
2.3	**1**	**1**	**p2.3**
Stopped	1	3	p2.2, p2.4-5
__poll_nocancel	1	3	p2.2, p2.4-5
2.2	1	1	p2.2
2.4	1	1	p2.4
2.5	1	1	p2.5
tx_ttf_map (S3)	1	1	p1
Breakpoint	1	1	p1.1
main	1	1	p1.1
1.1	1	1	p1.1

Concise information on the status of threads

Record, save, and replay execution history

The ReplayEngine feature within CodeDynamics simplifies the troubleshooting process. By recording and saving program execution, ReplayEngine allows you to work back from a failure, error, or crash to find the root cause without repetitive restarts and stops, and allows you to store the recording to investigate the error at any time. ReplayEngine drastically reduces the amount of time invested in troubleshooting your code.

- Freedom to explore application execution either backwards or forwards
- Step back through execution history and review all variables and program behavior at any point
- Go back and look at functions and variables in the context of the crash or error

- Easily follow the logic of unfamiliar routines
- Set a breakpoint and run backwards until the breakpoint is hit
- Enable recording in the middle of a debugging session without needing to restart the process

Detect and analyze memory errors

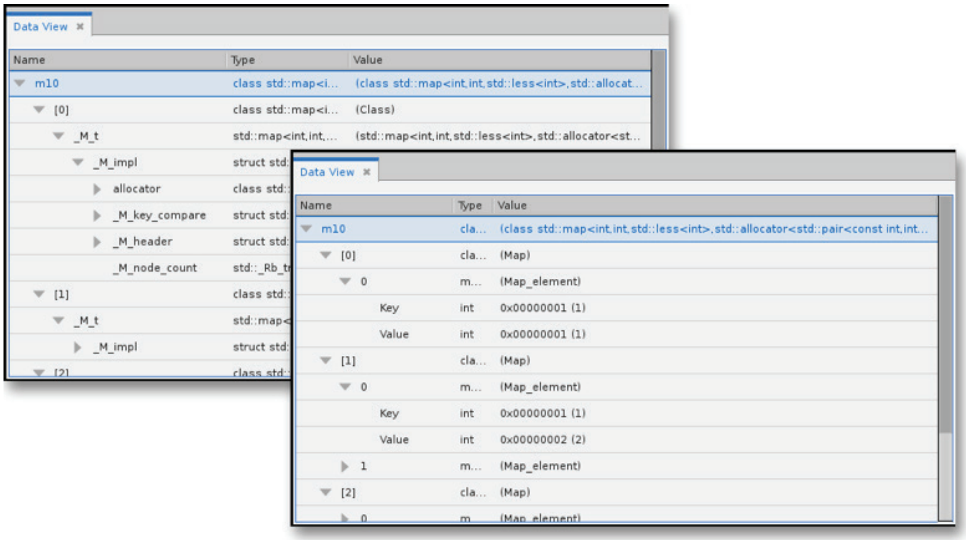
CodeDynamics includes MemoryScape, an interactive, dynamic memory analysis and debugging tool that simplifies and reduces time spent on memory debugging. The powerful memory error detection and analysis tool incurs minimal performance overhead and provides an interface for quick exploration and identification of heap memory within a program.

- No need to recompile applications
- Detect leaks and errors in vendor libraries, or filter them out of your analysis
- Track and interpret allocated, deallocated, and leaked memory blocks
- Detect memory leaks and events early
- Flag memory leaks and events before they crash your application
- Detect corrupted memory
- Analyze memory usage patterns, optimize applications, or scale them up to run on large grids or supercomputers
- Support for multiprocess and hybrid applications in clusters

Manipulate complex data types

An advanced type transformation facility takes your data debugging to new levels, visually simplifying complex data structures so you can understand them. This facility lets you inspect and modify data values for STL collection classes, very large single and multidimensional arrays, and Boost collection classes at runtime.

- No need to print or log individual data elements
- Examine data elements and their relationships through a simple view
- Easily cast, transform, and change the values of your data



Name	Type	Value
m10	class std::map<i...	(class std::map<int,int,std::less<int>,std::allocat...
[0]	class std::map<i...	(Class)
_M_t	std::map<int,int,...	(std::map<int,int,std::less<int>,std::allocator<st...
_M_impl	struct std:	
allocator	class std::	
_M_key_compare	struct std:	
_M_header	struct std:	
_M_node_count	std::_Rb_tr	
[1]	class std::	
_M_t	std::map<	
_M_impl	struct std:	
[2]	class std::	

Name	Type	Value
m10	cla...	(class std::map<int,int,std::less<int>,std::allocator<std::pair<const int,int...
[0]	cla...	(Map)
0	m...	(Map_element)
Key	int	0x00000001 (1)
Value	int	0x00000001 (1)
[1]	cla...	(Map)
0	m...	(Map_element)
Key	int	0x00000001 (1)
Value	int	0x00000002 (2)
1	m...	(Map_element)
[2]	cla...	(Map)
0	m...	(Map_element)

Easily examine data in STL collection types

USA	800-487-3217
FRANCE	+33 (0) 1 46 93 94 20
GERMANY	+49 (0) 6103 5934 0
UK	+44 (0) 8450 549950
JAPAN	+81 (0) 3 5211 7760

www.roguewave.com

Rogue Wave provides software development tools for mission-critical applications. Our trusted solutions address the growing complexity of building great software and accelerates the value gained from code across the enterprise. The Rogue Wave portfolio of complementary, cross-platform tools helps developers quickly build applications for strategic software initiatives. With Rogue Wave, customers improve software quality and ensure code integrity, while shortening development cycle times.