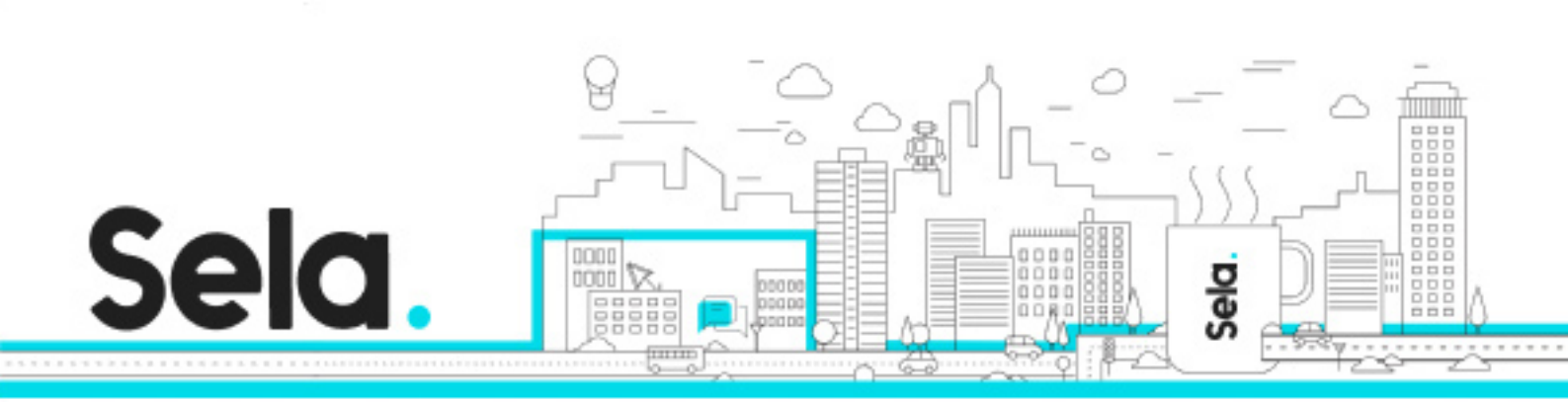
Sela.

CPPDbg

Advanced C++ WNA Debugging

college@sela.co.il 03-6176666



Advanced C++ WNA Debugging

CPPDbg - Version: 3

3 days Course

Description:

In this three-day instructor-led course, you will learn how to debug Windows Native Applications written in C/C++ or any other language compiled into Windows binary format in development, QA, and production environments. Extensive hands-on labs and demonstrations ensure you can apply the skills learned to your applications and systems.

Intended audience:

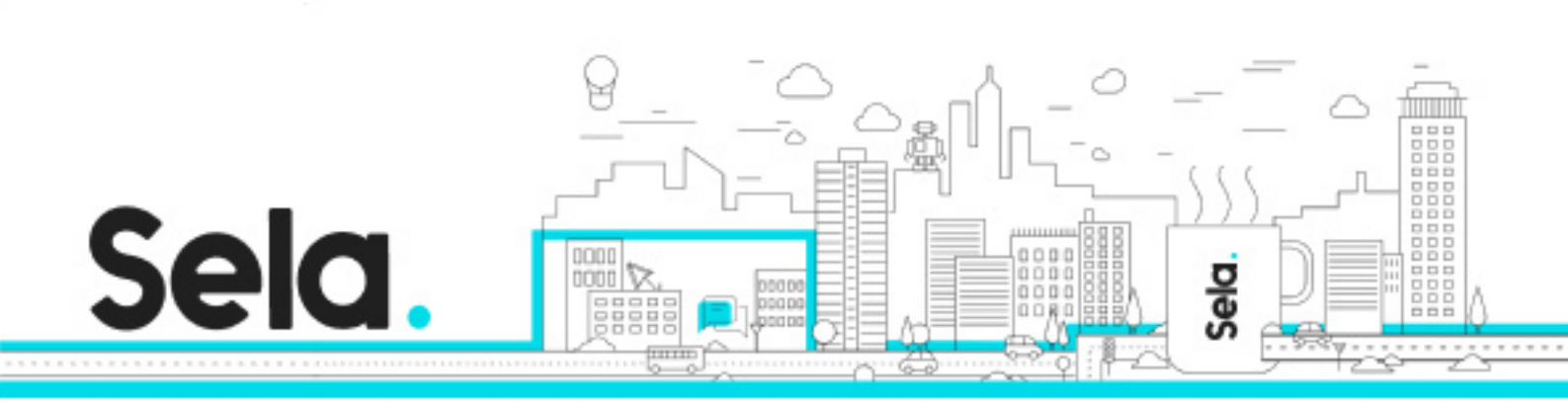
Experienced Windows developers that want to learn advanced debugging tools and techniques to debug native Windows applications.

Prerequisites:

Experience with C/C++ programming
Experience programming windows native applications
Familiarity with the WIN32 APIs
Familiarity with windows operating systems concepts such as virtual memory, multithreading and synchronizations
Familiarity with computer architecture concepts such as CPU registers, cache, main memory

Objectives:

Learn how to use standard and advanced Visual Studio debugging features to debug native applications written in C/C++.
Learn how to debug native applications in development, QA, and production environments using WinDbg.



Learn how to trace application behavior in development, QA, and production environments using ETW.
Learn about other helpful tools that do not require installation you can use to gather information in development, QA, and mainly production environments to help you diagnose problems.
Learn how to analyze and solve performance and deadlock issues.

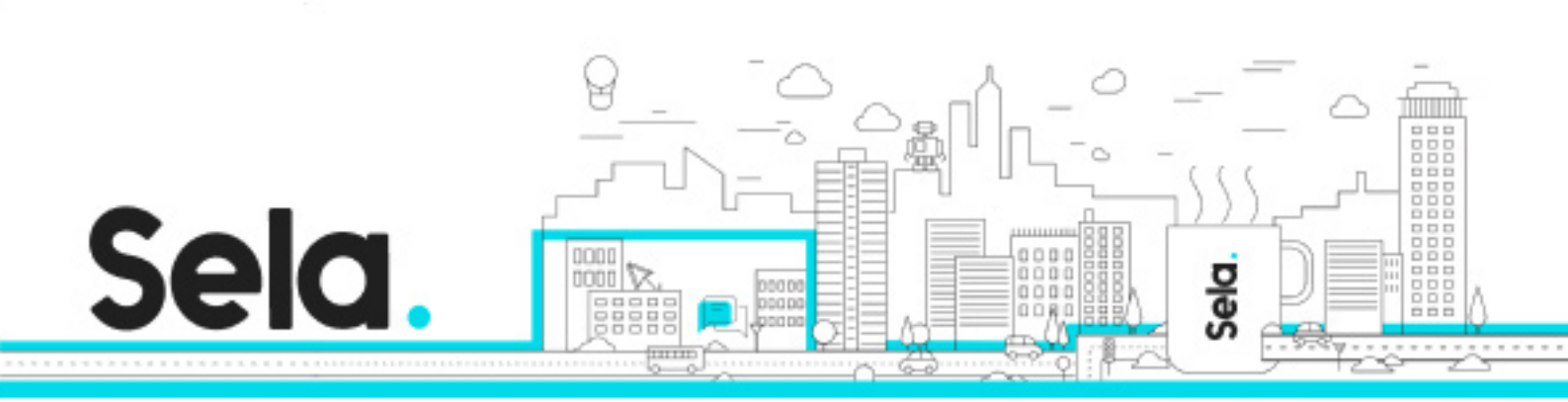
Topics:

Module 01 - Methodology of debugging

- Prerequisites
- Categorization of bugs
- Assumptions and Scenarios
- Environments
- Tools
- Demo(s),Lab(s)

Module 02 - Standard and advanced Visual Studio native features

- Visual studio tools to avoid and eliminate bugs
- Static code analysis
- Configuring Visual Studio for native debugging
- Debugging symbols and symbols server configuration
- Visual studio Build in tools for debugging, tracing and performance
- Short review on the well-known commands and tools
- Deep dive into the less known commands and tools
- Remote Debugging
- Demo(s), Lab(s)



Module 03 - Debugging tools for windows package

- Getting and installing debugging tools for windows
- Debugging tools and other tools included in the package
- Using the tools without installation
- Demo(s), Lab(s)

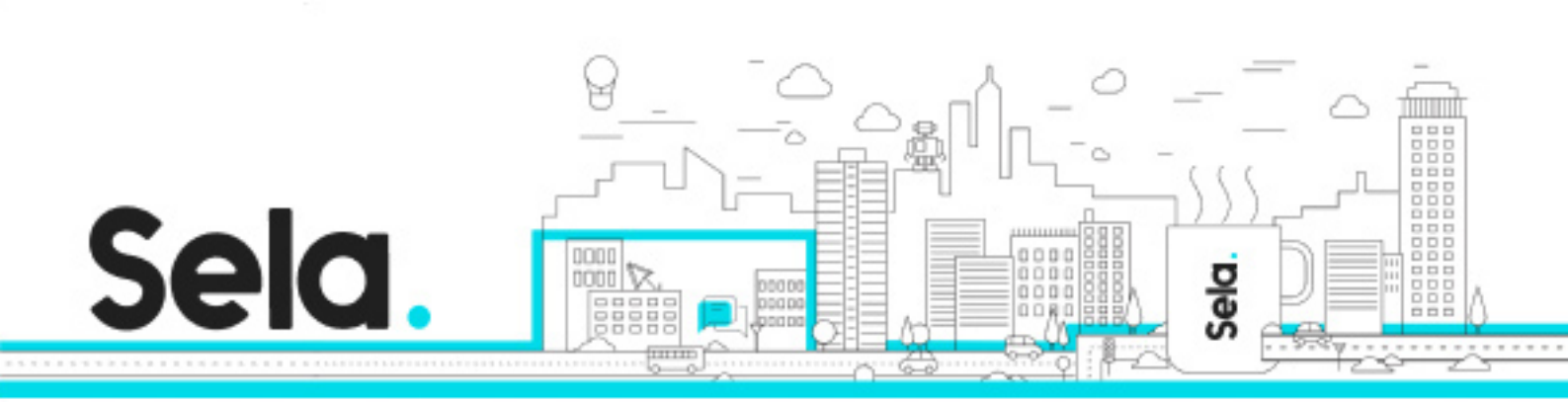
Module 04 - Windows handling of process and exceptions

- Windows Architecture
- Kernel, X-86, and X-64 address spaces
- SHE
- WOW
- Windows executables and DLLs loading and terminating process
- Processes, Threads, Fibers and Jobs.
- Windows handling of exceptions
- Win32 APIs
- Demo(s), Lab(s)

Module 05 - Using WinDbg to debug live processes

- Introduction to WinDbg
- Configuring WinDbg for live debugging
- Using WinDbg to debug live processes
- WinDbg basic commands
- Calling conventions and parameter passing
- Demo(s), Lab(s)

Module 06 - Assembly Language Fundamentals (Appendix)



- When to use dumps (Use cases)
- Dump types
- Techniques to manually and automatically generate dumps
- Kernel dumps Vs. process dumps
- Tools to analyze dumps
- Demo(s), Lab(s)

Module 07 - Windows Programming Concepts (Appendix)

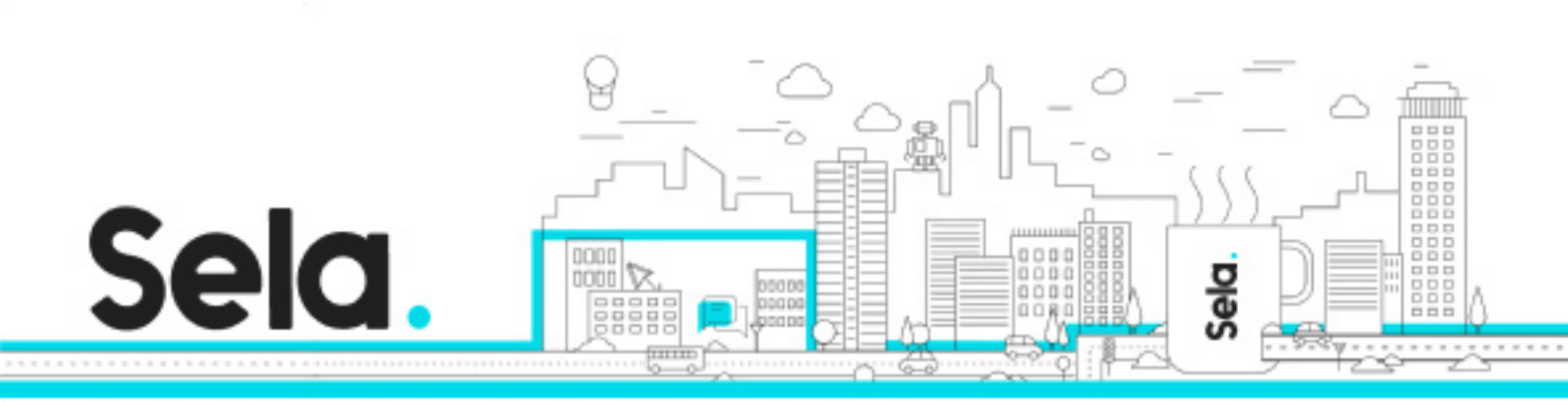
- Configuring WinDbg for dump analyzing
- Using WinDbg to debug dumps
- WinDbg commands used in dump analysis
- WinDbg Extensions
- WinDbg Scripts
- Demo(s), Lab(s)

Module 08 – Basic assembler

- Why do you need basic knowledge of assembler
- CPU registers and their use in running programs
- Basic assembler commands
- Using WinDbg to disassemble code
- What does the optimizer do to your code
- How optimization affect debugging
- Making you code more debugger friendly
- Demo(s), Lab(s)

Module 09 – Sysinternals tools package

- Getting and installing Sysinternals package



- tools included in the package
- Using the tools without installation
- Tools from the package that are helpful in debugging
- Demo(s), Lab(s)

Module 10 – ETW the tracing framework of windows

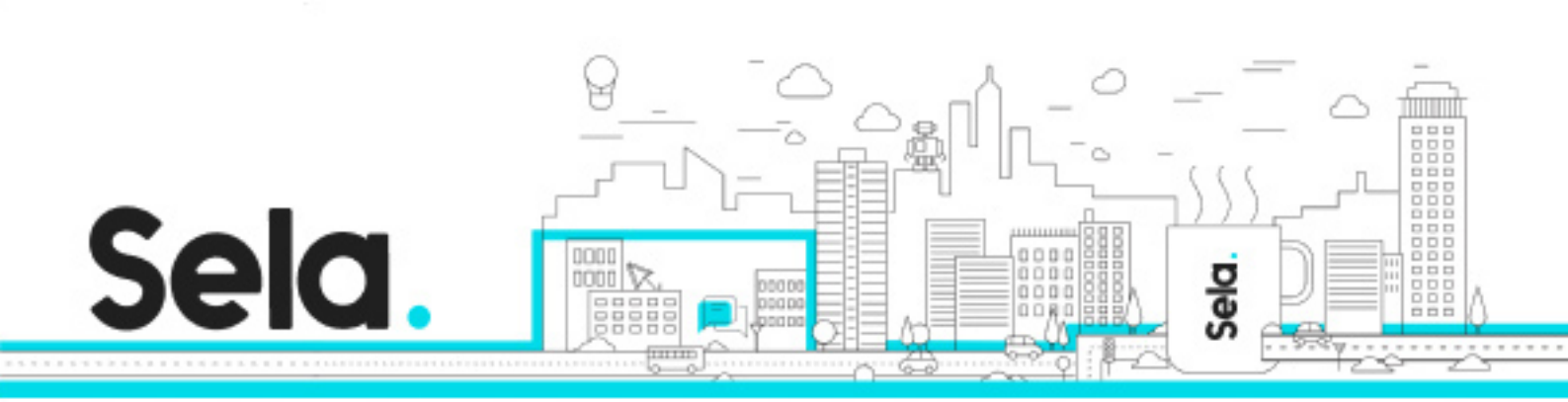
- ETW features
- Tools that use ETW
- Incorporating ETW in your code
- Creating and controlling ETW traces.
- Analyzing ETW traces.
- Demo(s), Lab(s)

Module 11 – Memory management in windows

- Memory manager APIs
- Heal allocations
- The CRT memory heap
- Analyzing heaps
- Application Verifier, UMDH, VMMap
- Demo(s), Lab(s)

Module 12 – Windows Performance monitor

- Detecting anomalies
- Creating a baseline
- Counters to watch for specific debug scenarios
- Counters to watch for performance problems
- Demo(s), Lab(s)



Module 13 – Windows performance toolkit (WPT)

- Getting and installing Windows performance toolkit
- tools included in the package
- Using the tools without installation
- Collecting WPT trace data
- Analyzing WPT trace data
- Demo(s), Lab(s)

Module 14 – Debugging techniques for specific Scenarios

- CPU Hang
- Deadlock Hang
- Exception crash
- Memory Leaks
- Resource Leaks
- Demo(s), Lab(s)