C & C++
Windows Diagnostics
Accelerated

Dmitry Vostokov
Software Diagnostics Services
Prerequisites

- Development experience

and (optional)

- Basic memory dump analysis
Training Goals

- Review common fundamentals of C and C++
- Review C++ specifics
- Use WinDbg for learning C and C++ internals
- See how C and C++ knowledge is used during diagnostics and debugging
Training Principles

- Talk only about what I can show
- Lots of pictures
- Lots of examples
- Original content and examples
std::vector<Session> sessions;
assert(sessions.size() == 12);
assert(sessions.capacity() > 12);
Training Idea

- Reading Windows-based Code training
- Memory dump analysis training
- Reversing training
- Windows API training
General C & C++ Aspects

- Philosophy of pointers
- Structures, classes, and objects
- Promotions and conversions
- Macros, types, and synonyms
- Source code organization, PImpl
- Pointer dereference walkthrough
- Functions and function pointers
- Inheritance
- Operators, function objects
- Destructors, virtual destructors
- Local stack variables and values
- Memory operators and expressions
- Alignment
- Slicing
- Iterators as pointers
- Lambdas and their internals
- Threads and synchronization

- Memory and pointers
- Basic types
- Memory and structures
- Uniform initialization
- Memory storage
- References
- Values, lvalues, rvalues
- Constant values and expressions
- Namespaces
- Constructors, copy, assignment
- Virtual functions, pure methods
- VTBL and VPTR
- Access levels
- Overloading, overriding
- Templates
- Memory ownership, RAII
- Smart pointers

© 2023 Software Diagnostics Services
What We Do Not Cover*

- Enumerations
- Move constructors and assignment operators
- Deleted and default members
- Universal references
- Concepts
- Coroutines
- Modules
- Tasks
- Ranges
- Container and algorithm semantics and pragmatics
- Container allocators
- Polymorphic allocators

* We promise to include these topics in the second edition
Windows C & C++ Aspects

- Windows-specific type aliases and macros
- Desktop application walkthrough
- Desktop application improvement
- Service walkthrough
- Command-line application walkthrough
- LLP64
- COM
- Necessary x64 disassembly
- Parameter passing
- Implicit parameter
Links

- Memory Dumps
  Included in Exercise CPP0

- Exercise Transcripts
  Included in this book
Exercise CPP0

- **Goal:** Install Visual Studio 2022 and WinDbg or Debugging Tools for Windows

- **Memory Analysis Patterns:** Stack Trace; Incorrect Stack Trace

- \ACPP-Dumps\Exercise-CPP0-WinDbg.pdf
Why C & C++?

- Interfacing
- Malware analysis
- Vulnerability analysis and exploitation
- Reversing
- Diagnostics
- Low-level debugging
- OS Monitoring
- Memory forensics
- Crash and hang analysis
- Secure coding
- Static code analysis
- Trace and log analysis
Which C & C++?

- C
- C++ as a better C
- Proper C++ (legacy and modern)
- Windows specifics
My History of C & C++

- C from 1987 and C++ from 1989 (*Old CV*)
- C++ as a better C from 1991
- Implicit design patterns in 1994-1995
- C++ as proper C++ from 2000
- Explicit design patterns in 2000
- C++98/03/STL from 2001
- Windows memory dump analysis from 2003
- [...] 
- C++11/14 from 2016
- C++17 from 2017
- Functional programming from 2020
- C++20 from 2023
C and C++ Mastery Process

Coding

Mental Compiling
Thought Process

- C and C++
- Scala/FP
- Python
Philosophy of Pointers
Pointer
Pointer Dereference
Many to One

Pointer

Pointer

Pointer
Many to One Dereference
Invalid Pointer
Invalid Pointer Dereference
Wild (Dangling) Pointer
Pointer to Pointer
Pointer to Pointer Dereference
Naming Pointers and Entities

A
1
fadb6810

B
2
86556810

C
3
a656ffbd
Names as Pointer Content
# Pointers as Entities

<table>
<thead>
<tr>
<th>fadb6810</th>
<th>86556810</th>
<th>a656ffbd</th>
</tr>
</thead>
<tbody>
<tr>
<td>86556810</td>
<td>a656ffbd</td>
<td>00000000</td>
</tr>
</tbody>
</table>
Memory and Pointers
Mental Exercise

How many pointers can you count?

<table>
<thead>
<tr>
<th>2ab1000</th>
<th>2ab1004</th>
<th>2ab1008</th>
<th>2ab100c</th>
<th>2ab1010</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ab1008</td>
<td>ffffffff</td>
<td>2ab1010</td>
<td>2ab100c</td>
<td>00000000</td>
</tr>
</tbody>
</table>
Debugger Memory Layout

```
2ab1000:  2ab1008
2ab1004:  ffffffff
2ab1008:  2ab1010
2ab100c:  2ab100c
2ab1010:  00000000
2ab1014:  00002000
```

```
2ab1000:  2ab1008  ffffffff
2ab1008:  2ab1010  2ab100c
2ab1010:  00000000  00002000
```
# Memory Dereference Layout

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Address</th>
<th>Value</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ab1000</td>
<td>2ab1008</td>
<td>2ab1000</td>
<td>2ab1008</td>
<td>2ab1000</td>
<td>2ab1010</td>
</tr>
<tr>
<td>2ab1004</td>
<td>ffffffff</td>
<td>2ab1004</td>
<td>ffffffff</td>
<td>2ab1004</td>
<td>????????</td>
</tr>
<tr>
<td>2ab1008</td>
<td>2ab1010</td>
<td>2ab1008</td>
<td>2ab1010</td>
<td>2ab1008</td>
<td>00000000</td>
</tr>
<tr>
<td>2ab100c</td>
<td>2ab100c</td>
<td>2ab100c</td>
<td>2ab100c</td>
<td>2ab100c</td>
<td>2ab100c</td>
</tr>
<tr>
<td>2ab1010</td>
<td>00000000</td>
<td>2ab1010</td>
<td>00000000</td>
<td>2ab1010</td>
<td>????????</td>
</tr>
<tr>
<td>2ab1014</td>
<td>00002000</td>
<td>2ab1014</td>
<td>00002000</td>
<td>2ab1014</td>
<td>????????</td>
</tr>
</tbody>
</table>
Names as Addresses

2ab1000: 2ab1008
2ab1004: ffffffff
2ab1008: 2ab1010
2ab100c: 2ab100c
2ab1010: 00000000
2ab1014: 00002000
# Addresses and Entities

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ab1000:</td>
<td>2ab1008</td>
</tr>
<tr>
<td>2ab1004:</td>
<td>ffffffff</td>
</tr>
<tr>
<td>2ab1008:</td>
<td>2ab1010</td>
</tr>
<tr>
<td>2ab100c:</td>
<td>2ab100c</td>
</tr>
<tr>
<td>2ab1010:</td>
<td>0000000000</td>
</tr>
<tr>
<td>2ab1014:</td>
<td>00002000</td>
</tr>
</tbody>
</table>

© 2023 Software Diagnostics Services
Arrays

2ab1000: 2ab1008
2ab1004: ffffffff
2ab1008: 2ab1010
2ab100c: 2ab100c
2ab1010: 00000000
2ab1014: 00002000

2ab1000: 2ab1008
2ab1008: 2ab1010
2ab100c: 2ab100c
2ab1010: 00000000
2ab1014: 00002000
Arrays and Pointers to Arrays

2ab1000:
- 2ab1008
- ffffffff

2ab1008:
- 2ab1010
- 2ab100c

2ab1010:
- 00000000
- 000002000

2ab1216: 2ab1000
Strings and Pointers to Strings

2ab1000: 'H'
2ab1001: 'e'
2ab1002: 'l'
2ab1003: 'l'
2ab1004: 'o'
2ab1005: 00

2ab1216: 2ab1000