Defect

Detect

DUmps Binaries Logs INternals

School of Security

Memory Thinking C & C++

Windows Diagnostics

Version 2.0

Dmitry Vostokov
Software Diagnostics Services

Memory Thinking for C & C++ Windows Diagnostics

Slides with Descriptions and Source Code Illustrations

Second Edition

Dmitry Vostokov
Software Diagnostics Services

OpenTask

Memory Thinking for C & C++ Windows Diagnostics: Slides with Descriptions and Source Code Illustrations, Second Edition

Published by OpenTask, Republic of Ireland

Copyright © 2024 by OpenTask

Copyright © 2024 by Dmitry Vostokov

Copyright © 2024 by Software Diagnostics Services

Copyright © 2024 by Dublin School of Security

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, transmitted in any form or by any means, or used for training artificial intelligence systems without the prior written permission of the publisher.

OpenTask books are available through booksellers and distributors worldwide. For further information or comments, send requests to press@opentask.com.

Product and company names mentioned in this book may be trademarks of their owners.

A CIP catalog record for this book is available from the British Library.

ISBN-13: 978-1912636617 (Paperback)

Revision 2.01 (July 2024)

Table of Contents

Preface **10**

About the Author **11**

Introduction **12**

Original Training Course Name *12*

Prerequisites *12*

Training Goals *13*

Training Principles *13*

Schedule *14*

Training Idea *14*

Version 2.0 Idea *15*

General C & C++ Aspects *15*

What We Do Not Cover *17*

Windows C & C++ Aspects *17*

Why C & C++? *18*

Which C & C++? *19*

My History of C & C++ *19*

C and C++ Mastery Process *20*

Thought Process *21*

Philosophy of Pointers **21**

Pointer *22*

Pointer Dereference *22*

One to Many *23*

Many to One *23*

Many to One Dereference *24*

Invalid Pointer *24*

Invalid Pointer Dereference *25*

Wild (Dangling) Pointer *25*

Pointer to Pointer *26*

Pointer to Pointer Dereference *26*

Naming Pointers and Entities *27*

Names as Pointer Content *27*

Pointers as Entities *28*

Pointer Code Examples **29**

Warning *30*

Pointer *30*

** Placement Style* *32*

Pointer Dereference *32*

One to Many *34*

Memory Leak *35*

Many to One 37

Many to One Dereference 39

Invalid Pointer 40

Invalid Pointer Dereference 42

Wild (Dangling) Pointer 43

Pointer to Pointer 45

Pointer to Pointer Dereference 47

Undefined Behavior 48

Memory and Pointers **49**

Mental Exercise 50

Debugger Memory Layout 50

Memory Dereference Layout 51

Names as Addresses 51

Addresses and Entities 52

Addresses and Structures 52

Pointers to Structures 53

Arrays 53

Arrays and Pointers to Arrays 54

Strings and Pointers to Strings 55

Basic Types **57**

ASCII Characters and Pointers 58

Bytes and Pointers 59

Wide Characters and Pointers 60

Integers 61

Long Integers 62

Little-Endian System 63

Short Integers 64

Long Long Integers 65

Signed and Unsigned Integers 66

Fixed Size Integers 67

Booleans 68

Bytes 69

Size 70

Alignment (C11) 71

Alignment (C++11) 72

LLP64 73

Nothing and Anything 74

Automatic Type Inference 75

Entity Conversion **76**

Pointer Conversion (C-Style) 77

Pointer Conversion (C++) 78

Numeric Promotion/Conversion 79

Numeric Conversion 80

Incompatible Types 81

Forcing *82*

Uniting *83*

Structures, Classes, and Objects **85**

Structures *86*

Access Level *87*

Classes and Objects *88*

Structures and Classes *89*

Pointer to Structure *90*

Pointer to Structure Dereference *91*

One to Many *92*

Memory Leak *94*

Many Pointers to One Structure *96*

Many to One Dereference *97*

Invalid Pointer to Structure *99*

Invalid Pointer Dereference *101*

Wild (Dangling) Pointer *102*

Pointer to Pointer to Structure *105*

Pointer to Pointer Dereference *106*

Memory and Structures **108**

Addresses and Structures *109*

Structure Field Access *110*

Pointers to Structures *112*

Pointers to Structure Fields *113*

Structure Inheritance *116*

Structure Slicing *117*

Inheritance Access Level *118*

Structures and Classes II *119*

Internal Structure Alignment *120*

Static Structure Fields *121*

Uniform Initialization **123**

Old Initialization Ways *124*

New Way {} *125*

Uniform Structure Initialization *126*

Static Field Initialization *128*

Macros, Types, and Synonyms **129**

Macros *130*

Old Way *131*

New Way *132*

Enumerations **133**

Old Way *134*

New Way *135*

Memory Storage **137**

Overview *138*

Memory Regions *138*

Dynamic Virtual Memory 139
Static Memory 139
Stack Memory 140
Thread Stack Frames 140
Local Variable Value Lifecycle 141
Scope 142
Stack Allocation Pitfalls 143
Explicit Local Allocation 145
Heap Memory 146
Dynamic Allocation (C-style) 147
Dynamic Allocation (C++) 147
Memory Expressions 148
Memory Operators 148
Local Pointers (Manual) 151
In-place Allocation 152
Useful WinDbg Commands 153
Source Code Organisation` **154**
Logical Layer (Translation Units) 155
Physical Layer (Source Files) 155
Inter-TU Sharing 156
Classic Static TU Isolation 156
Namespace TU Isolation 157
Declaration and Definition 157
TU Definition Conflicts 158
Fine-grained TU Scope Isolation 159
Conceptual Layer (Design) 160
Incomplete Types 160
References **163**
*Type& vs. Type** 164
Values **165**
Value Categories 166
Constant Values 167
Constant Expressions 168
Functions **169**
Macro Functions 170
constexpr Functions 171
Pointers to Functions 172
Function Pointer Types 173
Reading Declarations 174
Structure Function Fields 175
Structure Methods 176
Structure Methods (Inlined) 177
Structure Methods (Inheritance) 178
Structure Virtual Methods 179

Structure Pure Virtual Methods 182

Structure as Interface 184

Function Structure 186

Structure Constructors 187

Structure Converting Constructors 188

Structure Delegating Constructors 189

Structure Member Initialization 191

Structure Copy Constructor 192

Copy vs. Move Semantics 193

Structure Move Constructors 193

Structure Copy Assignment 195

Structure Move Assignment 196

Structure Destructor 198

Structure Destructor Hierarchy 199

Structure Virtual Destructor 200

Structure Member Destruction 202

Destructor as a Method 203

Structure Default Operations 204

Structure Deleted Operations 205

Conversion Operators 207

Parameters by Value 208

Parameters by Pointer/Reference 210

Parameters by Ptr/Ref to Const 212

Parameters by Ref to Rvalue 214

Possible Mistake 215

Function Overloading 215

Default Arguments 216

Variadic Functions 217

Immutable Objects 218

Static Structure Functions 219

Lambdas 220

x64 CPU Registers 220

Instructions and Registers 221

Memory and Stack Addressing 221

Memory Cell Sizes 222

Memory Load Instructions 222

Memory Store Instructions 223

Flow Instructions 223

Function Parameters 224

Struct Function Parameters 224

this 225

Function Objects vs. Lambdas 226

Captures and Closures 227

Lambdas as Parameters 228

Lambda Parameter Optimization 229

Lambdas as Unnamed Functions 230

std::function Lambda Parameters 231

auto Lambda Parameters 232

Lambdas as Return Values 233

Virtual Function Call **234**

VTBL Memory Layout 235

VPTR and Struct Memory Layout 235

Templates: A Planck-length Introduction **237**

Why Templates? 238

Reusability 238

Types of Templates 239

Types of Template Parameters 240

Type Safety 241

Flexibility 242

Metafunctions 243

Variadic Templates 244

Iterators as Pointers **245**

Containers 246

Iterators 246

Constant Iterators 247

Pointers as Iterators 248

Algorithms 249

Memory Ownership **251**

Pointers as Owners 252

Problems with Pointer Owners 252

Smart Pointers **253**

Basic Design 254

Unique Pointers 254

Handles as Unique Pointers 255

Shared Pointers 256

RAII **258**

RAII Definition 259

RAII Advantages 259

Handle RAII 260

Threads and Synchronization **262**

Threads in C/C++ 263

Threads in C++ Proper 264

Synchronization Problems 265

Synchronization Solution 266

Memory-safe C++ Development **267**

URSS Principle 267

Resources **268**

C and C++ 268

Reading Windows-based Code 269

Windows (C and C++) 270

Training (Windows C and C++) 271