Defect
Detect

DUmps Binaries Logs INternals
School of Security

Memory Thinking C & C++

Windows Diagnostics

Dmitry Vostokov
Software Diagnostics Services

Memory Thinking for C & C++ Windows Diagnostics

Slides with Descriptions Only

Dmitry Vostokov
Software Diagnostics Services

OpenTask

Memory Thinking for C & C++ Windows Diagnostics: Slides with Descriptions Only

Published by OpenTask, Republic of Ireland

Copyright © 2023 by OpenTask

Copyright © 2023 by Dmitry Vostokov

Copyright © 2023 by Software Diagnostics Services

Copyright © 2023 by Dublin School of Security

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means 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: 978-1912636730 (Paperback)

Revision 1.00 (November 2023)

Table of Contents

Table of Contents 3

Preface 15

About the Author 16

Introduction 17

Original Training Course Name 17

Prerequisites 18

Training Goals 19

Training Principles 20

Schedule 21

Training Idea 22

General C & C++ Aspects 23

What We Do Not Cover 25

Windows C & C++ Aspects 26

Why C & C++? 27

Which C & C++? 29

My History of C & C++ 30

C and C++ Mastery Process 32

Thought Process 33

Philosophy of Pointers **34**

Pointer *35*

Pointer Dereference *36*

Many to One *37*

Many to One Dereference *38*

Invalid Pointer *39*

Invalid Pointer Dereference *40*

Wild (Dangling) Pointer *41*

Pointer to Pointer *42*

Pointer to Pointer Dereference *43*

Naming Pointers and Entities *44*

Names as Pointer Content *45*

Pointers as Entities *46*

Memory and Pointers **47**

Mental Exercise *48*

Debugger Memory Layout *49*

Memory Dereference Layout *50*

Names as Addresses *51*

Addresses and Entities *52*

Addresses and Structures 53

Pointers to Structures 54

Arrays 55

Arrays and Pointers to Arrays 56

Strings and Pointers to Strings 57

Basic Types 58

ASCII Characters and Pointers 59

Bytes and Pointers 60

Wide Characters and Pointers 61

Integers 62

Long Integers 63

Little-Endian System 64

Short Integers 65

Long Long Integers 66

Signed and Unsigned Integers 67

Fixed Size Integers 68

Booleans 69

Bytes 70

Size 71

Alignment *72*

LLP64 *73*

Nothing and Anything *74*

Automatic Type Inference *75*

Entity Conversion **76**

Pointer Conversion (C-Style) *77*

Numeric Promotion/Conversion *78*

Numeric Conversion *79*

Incompatible Types *80*

Forcing *81*

Structures, Classes, and Objects **83**

Structures *84*

Access Level *85*

Classes and Objects *86*

Structures and Classes *87*

Pointer to Structure *88*

Pointer to Structure Dereference *89*

Many Pointers to One Structure *90*

Many to One Dereference *91*

Invalid Pointer to Structure *92*

Invalid Pointer Dereference *93*

Wild (Dangling) Pointer *94*

Pointer to Pointer to Structure *95*

Pointer to Pointer Dereference *96*

Memory and Structures **97**

Addresses and Structures *98*

Structure Field Access *99*

Pointers to Structures *100*

Pointers to Structure Fields *101*

Structure Inheritance *102*

Structure Slicing *103*

Inheritance Access Level *105*

Structures and Classes II *106*

Internal Structure Alignment *107*

Static Structure Fields *108*

Uniform Initialization **109**

Old Initialization Ways *110*

New Way {} *111*

Uniform Structure Initialization *112*

Static Field Initialization *113*

Macros, Types, and Synonyms **114**

Macros *115*

Old Way *116*

New Way *117*

Memory Storage **118**

Overview *119*

Thread Stack Frames *120*

Local Variable Value Lifecycle *121*

Stack Allocation Pitfalls *123*

Explicit Local Allocation *125*

Dynamic Allocation (C-style) *126*

Dynamic Allocation (C++) *127*

Memory Operators *128*

Memory Expressions *129*

Local Pointers (Manual) *130*

In-place Allocation *132*

Source Code Organisation **133**

Logical Layer (Translation Units) *134*

Physical Layer (Source Files) *135*

Inter-TU Sharing *136*

Classic Static TU Isolation *137*

Namespace TU Isolation *138*

Declaration and Definition *139*

TU Definition Conflicts *140*

Fine-grained TU Scope Isolation *141*

Conceptual Layer (Design) *142*

Incomplete Types *143*

References **144**

*Type& vs. Type** *145*

Values **146**

Value Categories *147*

Constant Values *148*

Constant Expressions *149*

Functions **150**

Pointers to Functions *151*

Function Pointer Types *153*

Reading Declarations 154

Structure Function Fields 155

Structure Methods 156

Structure Methods (Inlined) 157

Structure Methods (Inheritance) 158

Structure Virtual Methods 160

Structure Pure Virtual Methods 162

Structure as Interface 163

Function Structure 164

Structure Constructors 165

Structure Copy Constructor 166

Structure Copy Assignment 167

Structure Destructor 168

Structure Destructor Hierarchy 169

Structure Virtual Destructor 170

Destructor as a Method 171

Conversion Operators 172

Parameters by Value 174

Parameters by Pointer/Reference 175

Parameters by Ptr/Ref to Const *176*

Possible Mistake *177*

Function Overloading *178*

Immutable Objects *179*

Static Structure Functions *180*

Lambdas *181*

x64 CPU Registers *182*

Instructions and Registers *183*

Memory and Stack Addressing *184*

Memory Cell Sizes *185*

Memory Load Instructions *186*

Memory Store Instructions *187*

Flow Instructions *188*

Function Parameters *189*

Struct Function Parameters *190*

this *191*

Function Objects vs. Lambdas *192*

Captures and Closures *193*

Lambdas as Parameters *195*

Lambda Parameter Optimization *197*

Lambdas as Unnamed Functions *199*

std::function Lambda Parameters *201*

auto Lambda Parameters *203*

Lambdas as Return Values *205*

Virtual Function Call **207**

VTBL Memory Layout *208*

VPTR and Struct Memory Layout *209*

Templates: A Planck-length Introduction **210**

Why Templates? *211*

Reusability *212*

Types of Templates *214*

Types of Template Parameters *215*

Type Safety *216*

Flexibility *218*

Metafunctions *219*

Iterators as Pointers **220**

Containers *221*

Iterators *222*

Constant Iterators *223*

Pointers as Iterators *224*

Algorithms *225*

Memory Ownership **226**

Pointers as Owners *227*

Problems with Pointer Owners *228*

Smart Pointers **229**

Basic Design *230*

Unique Pointers *231*

Handles as Unique Pointers *232*

Shared Pointers *233*

RAII **234**

RAII Definition *235*

RAII Advantages *236*

Handle RAII *237*

Threads and Synchronization **238**

Threads in C/C++ *239*

Threads in C++ Proper *240*

Synchronization Problems *241*

Synchronization Solution *242*

Resources **243**

C and C++ *244*

Reading Windows-based Code *245*

Windows (C and C++) *247*

Training (Windows C and C++) *249*