Modern C++ 17 OOP and Windows Reverse Engineering Essentials

By Milad Kahsari Alhadi

Last Update: **Friday - 2020 10 January**

- **Object Oritented Programming with C++ – 1671 Min**
  - Introduction to Microsoft C++
    i. Microsoft C++ Compiler
    ii. Microsoft C++ Linker
    iii. Visual Studio IDE
    iv. Visual Studio Debugger
  - Introduction to C++ and OOP
    i. What is C++?
    ii. What is a Multiparadigm Language?
      1. Inline
      2. Structural
      3. Object Oriented
      4. Functional
    iii. C++ Programming Approaches
      1. Native C++ Programming
      2. Managed C++ Programming
      3. DotNet Framework and C++/CLI
    iv. C++ Programs Type
      1. Graphical Programs – Win32 API
      2. Console Programs
  - Fundamental and User Data Type
    i. Fundamental Data Types
      1. Int
      2. Char
      3. Bool
      4. Float
      5. Double

- ii. Data Type Casting
    1. Roundoff Problem
    2. Losing Precisions
- iii. User Defined Data Types
    1. Classes
    2. Structure
    3. Enumerations

— Classes and Objects

- i. Classes and Objects
- ii. Inheritance and Access Modifiers
    1. Public
    2. Private
    3. Protected
    4. Friend Classes and Functions
- iii. Namespaces and Enumerations

— Conditions and Repeations

- i. If and Else
- ii. Switch Cases
- iii. For and While loop
- iv. Range based for loop
- v. Visual Studio Arguments Settings:
    1. Language Standard
    2. Intermediate File
    3. Output File
    4. Compile As

— Memory Addressing

- i. What is a Pointer?
    1. Pointers Declaration
    2. Pointers Initialization
    3. Pointer Size in 32 Bit Systems
    4. Pointer Size in 64 Bit Systems
- ii. Pointers to Pointers
    1. Pointers Dereferencing
- iii. C++ References
    1. References vs Pointers
    2. Pass by References

    iv. Memory Analysis for References
- Translation Phases
    i. Preprocessing – Microsoft Preprocessor
    ii. Compiling – Microsoft C++ Compiler and Optimizer
    iii. Assembling – Microsoft Assembler / MASM
    iv. Linking – Microsoft Linker
    v. Visual Studio Project Settings
        1. Preproccessing Output – .i Files
        2. Compiling Output – .Asm Files
        3. Assembling Output – .Obj Files
        4. Linking Output – .Exe Files
- Preprocessoring and Preprocessor
    i. What is Preprocessing?
    ii. Why Preprocessing is important?
    iii. Introduction to Translation Phase
    iv. Preprocessing Directives
        1. Include
        2. Pragma
        3. Define
        4. Undef
        5. Ifdef and ifndef
        6. Else
- Disassembler and Disassembling
    i. Reverse of Compilation Process
    ii. Disassemblers Tasks
    iii. Disassemblers Types
        1. Capstone Engine
        2. IDA Disassembler
        3. Radare2 Cutter
- Debugger and Debugging
    i. Visual Studio Builtin Debugger
    ii. Standalone Debuggers
        1. OllyDBG
        2. ImmDBG
        3. x64DBG

- Overloading
    i. What is Overloading?
    ii. What is an Operator?
        1. Why is it Important?
        2. Function Overloading
        3. Class Member Overloading
        4. Operator Member Overloading
- Templates
    i. What are Templates?
    ii. Why is it Important?
    iii. Standard Template Library
        1. Template in Action
            a. Free Function Templates
            b. Member Function Templates
            c. Class Templates
            d. Specialization Templates
- Constants and const keyword
    i. What are Cons Qualifier?
    ii. Why is it Important?
        1. Const Keyword
        2. Const in Action
            a. Constant Variables
            b. Constant Pointers
            c. Constant Pointers and Constant Locations
            d. Pass Constant Arguments to Functions
- Free Store or Heap Memory
    i. Free Store / Heap Memory
    ii. Dynamic Memory Allocation
        1. Struct Memory Management
        2. Class Memory Managment
        3. Constructor and Destructor
        4. Free Store Keywords
            a. New
            b. Delete
            c. Malloc
            d. Free
    iii. Smart Pointers and Automatic Memory Managment

1. Raw Pointers
2. Raw Pointers Memory Management Issues
    a. Never Free
    b. Double Free
    c. Danling Pointers
    d. Other Memory Leakage Issues
3. What are Smart Pointers?
    a. Auto Deductions
    b. Unique Pointers
    c. Shared Pointers
    d. Weak Pointers

- Standard CPP Containers
    i. Std::Vectors
        1. Push and Pop back
        2. Begin and RBeign
        3. End and REnd
        4. Capacity and Size
        5. At and []
    ii. Std:Deque
        1. Push Back
        2. Push Front
        3. Emplace Back
        4. Emplace Front
        5. Advance and Erase
    iii. Std::List and std::forward_list
        1. Doubly Linked List
        2. Push Back and Front
        3. Emplace Back and Front
        4. Advance and Erase
        5. Merge and Sort
        6. Unique and Remove
    iv. Std::Map and Std::Multimap
        1. Keys and Values
        2. Reverse Iterator
        3. Iterator
        4. Insert
    v. Std::Set and Std::Multiset
        1. Keys and Values
        2. Reverse Iterator

3. Iterator
4. Insert

vi. Std:Pair
1. Pair Concept
2. Make Pair
3. Pair Compare

vii. Std:Stack
1. Stack Structure
2. Push Back and Pop Back
3. Stack Empty

viii. Std:Queue
1. Queue
2. Priority Queues
3. Double Ended Queue

- Static and Mutable Storage Class
  i. Storage Classes
     1. Static Storage Class
        a. Static Global Variable
        b. Static Global Function
        c. Static Local Variable
     2. Mutable Storage Class
        a. Const Member Function
        b. Mutable Field
- Polymorphism and Its Types
  i. Compile-time Polymorphism
  ii. Run-time Polymorphism
     1. Virtual Functions
     2. Overrided Functions
     3. Pure Virtual Functions
     4. Template-based Functions
  iii. Coercion Polymorphism
  iv. Ad-hoc Polymorphism
- Lambda Expression
  i. Lambda Calculus
  ii. Lambda Expression
     1. Capture Clause
     2. Parameter List

3. Return Type
4. Algorithm Header
    a. for_each
    b. find_if
5. Functional Header
    a. std::function

- Exception Handling
    i. Different Model of Handling
        1. C-Style
        2. C++-Style
        3. COM Model
        4. Posix Model
    ii. C++ Exception Handling
        1. Try
        2. Catch
        3. Multiple Catch
    iii. Runtime Exceptions
        1. invalid_argument
        2. out_of_range
        3. exception
    iv. Stack Unwinding
    v. Structured Exception Handler
    vi. Assert and Static Assert
- Modern CPP Standard Coding
    i. Linux Environment
        1. Clang++
        2. Github Commands
        3. Makefile
    ii. CPP Code Refactoring
    iii. Template Deduction
- Input and Output File Stream
    i. File Systems
    ii. C++ Streams
    iii. Output File Stream
        1. File Creation
        2. File Opening
        3. Ofstream Flags

4. Writing Data to File

iv. Input File Stream

1. Reading Data
2. Parsing Data
3. Showing Data in Terminal

— C++ Technical Concepts

i. Binding

1. Static Binding
2. Dynamic Binding
3. Virtual Pointer Table

ii. Callbacks

1. Why Callbacks?
2. Overview of Callbacks
3. Function Pointers
4. Function Objects / Functors

iii. Problem Solving with Polymorphism

1. Compile-time polymorphism with Templates
2. Runtime polymorphism with Inheritence
3. Overloading

iv. Translation Unit Testing

1. Integration Test
2. Acceptance Test
3. Performance Test
    a. Instrumentation
    b. Test Case Analysis
4. Unit Test Frameworks
    a. GoogleTest Framework
        - Configuring in VS17
        - Test a Project
    b. BoostTest Framework
        - Installation with by VCPKG
        - Configuring in VS17
        - Test a Project
    c. Microsoft Native Unit Testing
        - Configuring in VS17
        - Test a Project
    d. Runtime Speed Execution Performance Test
        - Using Chrono for Measurement
        - Using GoogleBenchmark for Measurement

v. Introduction to Iterators
    1. STL Components
        a. Algorithms
            - Sort
            - Search
            - Merge
        b. Containers
            - Sequence containers
            - Associative containers
            - Container adaptors
        c. Iterators
            - Why we need Iterators?
            - Iterators Categories
                1. Input Iterators
                2. Output Iterators
                3. Forward Iterators
                4. Bidirectional Iterators
                5. Random-Access Iterators
            - Auxiliary Iterator Functions
                1. std::advance
                2. std::distance
                3. std::next
                4. std::prev

vi. Introduction to Algorithms
    1. What is an Algorithm?
    2. Algorithm Characteristics
        a. Complexity
            - Constant
            - Logarithmic
            - Linear
            - Quaslinear
            - Polynomial
        b. Execution Policy
            - Sequential
            - Parallelism
        c. Vectorized Algorithms
        d. Non-Vectorized Algorithms
    3. Algorithm Standard Members
        a. All of
        b. Any of
        c. For each
        d. Find if
        e. …

vii. Concurrency and Parallelism
    1. What is an Concurrency?

2. What is an Parallelism?
3. Concurrent Programming
    a. Asynchronous Tasks
        - std::launch::async
        - std::launch::deferred
        - std::future
        - std::future_status
    b. Sharing and Coordinating
        - std::mutex
        - std::shared_mutex
        - std::lock_guard
        - std::shared_lock
        - Atomics
        - Condition Variables
    c. Low-Level Concurrency Facilities
        - std::this_thread::yield
        - std::this_thread::get_id
        - std::this_thread::sleep_for
        - td::this_thread::sleep_until
4. Parallel Algorithms
    a. std::execution::seq
    b. std::execution::par
    c. std::execution::par_unseq

viii. C++17 Language Changes

1. Type Inference
    a. auto
    b. decltype
2. Fold Expressions
    a. Parameters Packs
    b. Foldable Operators
    c. Functions Overloads
3. Expression Type
    a. Left Value
    b. Right Value
    c. eXpiring Value
4. Advanced Semantics
    a. Value-Based References
        - Left Value Reference
        - Right Value Reference
    b. Traditional Constructors
        - Deep Copy
        - Shallow Copy
        - Special Methods
        - Copy Constructors

        - Assignment Constructors
        - Delegated-based Constructors
    c. Move Semantics
        - std::move
        - Move Assignment
        - Move Constructors
        - std::remove_reference
    d. Template Specialization
    e. Tag Dispatching
    f. Subsituarion Failure is Not An Error
        - Domain Name Lookup
        - Type Argument Deduction
        - Type Substitution
        - Type SubstitutionFailure
        - Function Overload Sets
    g. Scope Resolution
        - Type Aliasing
        - Using Declarations
        - Namespace Resolution
5. Conditional Return Value
6. Compile-Time Rational Arithmetic
7. Pseudo-Random Number Generator
    a. Deterministic
    b. Non-Deterministic
    c. Sources of Entropy
        - Low Entropy Sources
        - High Entropy Sources
    d. Random Number Engine
        - Mersenne Twister
        - Random Device
        - C Randomness
    e. Randomness Application
        - Game Engine
        - VS ASLR Mechanism
            1. Image Base Randomization
            2. Address of Entry Point
            3. Code Injection
            4. PE Modifying
        - Cryptographic-Key Generation
    f. Random Number Distributions
        - Uniform
        - Normal
        - Poisson
    g. Entropy and Randomness Measurement
8. Filesystem Support

a. Introduction to File System
b. Operating System File Managers
    - Windows File Explorer
    - Linux Gnome Project
    - Linux KDE Project
    - Macintosh Finder
c. File System Header
    - filesystem::path
        1. Windows-Based Path
        2. Linux-Based Path
        3. Raw String Path
    - filesystem::path.empty()
    - filesystem::path.filename()
    - filesystem::path.extension()
    - filesystem::path.make_pereferd()
    - filesystem::directory_iterator
    - filesystem::recursive_directory_iterator
    - …
d. PE Signture Checker Software

ix. Boost Library Features

1. Boost Library Utilities
    a. Any
    b. Variant
    c. Optional
    d. PropertyTree
        - Tree Structure
        - Key and Value Pair
        - XML-JSON Parsing with PT
    e. Tribool Logic
        - Boolean Logic
        - Triple Value Logic
        - Tribool Decision Making
2. Boost Networking
    a. Introdunction to Networking
        - IP Addresses
        - Port Numbers
        - Port Numbers Relation with Services
        - Analogy of IP and Logical Ports
        - Packet Inspection with Wireshark
        - Communication Architecture
            1. Client/Server Architecture
            2. One-way (C&C) Architecture
    b. ASIO Network Programming Model
        - The Internet Protocol Suite
        - Hostname Resolution
        - Connecting

- Buffers
- Reading and Writing Data with Buffers

c. The Hypertext Transfer Protocol (HTTP)
   - Implementing a Simple HTTP Client
   - Asynchronous Reading and Writing
   - Serving

d. Multithreading Boost Asio