Modern C++ 17 OOP and Windows Reverse Engineering Essentials

By Milad Kahsari Alhadi

Last Update: Wednesday - 2019 24 April

- **Object Oritented Programming with C++ – 960 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?
    iii. Native C++ Programming
    iv. Managed C++ Programming
    v. DotNet Framework and C++/CLI
    vi. Graphical Programs – Win32 API
    vii. Console Programs
  - Fundamental and User Data Type
    i. Fundamental Data Types
      1. Int
      2. Float
      3. Double
    ii. User Defined Data Types
      1. Classes
      2. Structure
  - Cast and Converting
    i. Roundoff Problem
    ii. Losing Precisions

- 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. Intermediate File
    2. Output File
    3. Compile As
    4. Language Standard
- Memory Addressing
  i. What is a Pointer?
  ii. Pointers Declaration
  iii. Pointers Initialization
  iv. Pointers to Pointers
  v. Pointers Dereferencing
  vi. C++ References
  vii. Pass by References
  viii. 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. Ninja Binary
    4. 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?
  - iii. Why is it Important?
  - iv. Function Overloading
  - v. Class Member Overloading
  - vi. Operator Member Overloading

- Templates
    i. What are Templates?
    ii. Why is it Important?
    iii. Standard Template Library
    iv. Template in Action
        1. Free Function Templates
        2. Member Function Templates
        3. Class Templates
        4. Specialization Templates
- Constants and const keyword
    i. What are Cons Qualifier?
    ii. Why is it Important?
    iii. Const Keyword
    iv. Const in Action
        1. Constant Variables
        2. Constant Pointers
        3. Constant Pointers and Constant Locations
        4. 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

– Collection and Smart Arrays

- 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::Map
  1. Keys and Values
  2. Reverse Iterator
  3. Iterator
  4. Insert
- iii. Std::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:Pair
  1. Pair Concept
  2. Make Pair
  3. Pair Compare
- v. Std:Stack
  1. Stack Structure
  2. Stack Push Back
  3. Stack Pop Back
  4. Stack Empty
- vi. Std:Queue
  1. Queue
  2. Priority Queues
  3. Double Ended Queue

– Static and Mutable Storage Class

    i. Storage Class
    ii. Static Storage Class
        1. Static Global Variable
        2. Static Global Function
        3. Static Local Variable
    iii. Mutable Storage Class
        1. Const Member Function
        2. 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. 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
   v. XML File Processing
      1. XML File
      2. XML Processing
      3. Pugixml Library
         a. Installation pugixml
         b. Accessing document data
         c. Modifying document data