講師：侯捷，30+ 年經驗於計算機技術之寫作/翻譯/授課，曾執教於元智大學、南京大學，同濟大學。著有《深入淺出 MFC》《STL 源碼剖析》《多型與虛擬》《無責任書評》等書，譯有《Inside the C++ Object Model》《C++ Primer》《Effective C++》《More Effective C++》等書。

■1,C++ Under the Hood (C++底層揭密)
◎說明：高階 C++程序員更進一步理解 C++幕後運作機制
◎特色：特別強調"內核揭密 源碼剖析"，徹底理解 C++之啟動與結束之種種背景運作、CRT (C Runtime Library) 扮演的重要角色，以及 C++ virtual functions / polymorphism / dynamic binding 的幕後機制。本課程談的是基礎內核知識，對於高階技術養成及通貫極有幫助。全部內容建立在源碼級別。
◎適合：C++中高階學員
◎時數：12 小時
◎大綱：
* Startup Code 是什麼？出現在哪裡？可觸摸嗎？可修改嗎？可利用嗎？
* C Runtime (CRT) 在 C++ 程式中扮演的重要角色
* 徹底理解 C++ programs 生前死後的每一個細節
* C++ programs 執行前 (before main)和退出後 (after exit) 的所有行為，包括 mainCRTStartup, heapinit, ioinit, environment variables, argc & argv & envp, cinit, core of C Initialization, core of C++ Initialization, main(), details of exit.
* 如何讓代碼在 main() 之前執行
* 為什麼代碼能夠在 main() 之前執行
* 如何讓代碼在 exit() 之後執行
* 為什麼代碼能夠在 exit() 之後執行
* main()執行前的小內存塊(small memory blocks)分配和釋放，含實例觀測。
* global/static objects 的 ctors & dtors 的特殊性
* CRT malloc/free 的行為與實例觀察 (涵蓋 Visual C++和 GNU C++兩大體系)
* Debug Heap 的形成和實例觀察
* CRT Reporting Functions
* virtual functions 能夠神奇實現 OCP (Open-Closed Principle) 的原因
* virtual functions 幕後運作原理 (關於 vptrs 和 vtbls)
* 何謂動態繫結 (dynamic binding)？編譯形式為何？
* virtual functions 的兩大應用形式：(1) Polymorphism (2) Template Method

■2, Memory Management (內存管理)
◎說明：從最重要且最普及的內存管理庫 (memory management libraries) 中獲得啟發，並從源碼分析中獲得實戰能力。

◎特色：特別強調 "源碼剖析 實例驗證"，徹底剖析 C/C++ 各種內存管理策略。全部內容建立在源碼級別。了解內存管理的來龍去脈，對於操作系統、標準庫、應用程序脈脈相承的運行形成一種「胸中自有丘壑」的通貫感。
◎適合：C++中高階學員
◎時數：12 小時
◎大綱：
* 萬法歸宗：C++ Applications => C++ Standard Library (containers & allocators) => CRT (malloc / free) => O.S. (memory APIs)
* 內存管理訴求：空間效能和速度效能
* 內存管理之 C++語言構件：new/delete, array new/delete, placement new/delete.
* 兩個易混淆的語言構件：new expression vs. operator new
* 如何重載 (overloading) 和內存管理相關的語言構件
* 一個最簡化的 Memory Pool
* 從最簡化之 Memory Pool 進化到 std::allocator
* std::allocator 之最佳範例 (GNU 版本; 源碼剖析和詳細圖示)
* std::allocator 的優缺點和改善之道
* GNU C++ allocators 之近期演變
* CRT malloc/free 針對小內存塊的管理(源碼剖析、詳細圖示、實例觀察)
* VC++ allocators 之近期演變
* 一個貌似更佳的小內存塊分配器：loki::SmallObjAllocator
* 總整理：應用程序➔運用 std::containers➔std::allocator 運行➔malloc()運行➔O.S. API 運行

■3, Design Patterns (設計模式)
◎說明：此課程將令學員對於 Design Patterns 有充足、具體、深刻的認識，並獲得 "他山之石" 的實例體驗。
◎特色：特別強調 "從實例中檢驗和學習"，實例多取自知名 libraries 如 C++標準庫, Java 標準庫, Loki, MFC, Boost…，避免玩具示例 (toy samples)。您將能夠從若干 patterns 的設計原理深層理解大型框架選用它們的原因，以及龐大體系所考量的實現手法。
◎適合：C++/Java/C# 中高階學員 (本課程之實例主要以 C++ and/or Java 呈現，技術概念則適用於所有面向對象語言)
◎時數：12-24 小時 (課程實際份量取決於邀課時數&現場情況。根本原則是不趕進度，務必讓學員對於實際講授的每個 patterns 都獲得深刻的理解)。
◎大綱：
* Overview & Concepts
* OO Principles

* Design Patterns : (以下，根據邀課時數由老師挑選，或由邀課客戶挑選)
- Abstract Factory,
- Adapter,
- Bridge,
- Builder,
- Chain of Responsibility,
- Command,
- Composite,
- Decorator,
- Factory Method,
- Façade,
- Flyweight,
- Iterator,
- Mediator,
- Memento,
- Observer,
- Prototype,
- Singleton,
- State,
- Strategy,
- Template Method,
- Visitor

■4, Modern C++新特性

◎說明：C++自 2011 起有了大變革，此後的 2014, 2017, 2020 又有大大小小的增添。這些新版本統稱為 Modern C++。本課程挑選變革之大者，為學員墊定面對變局的最重要根基。

◎特色：只談新特性，區分"語言"及"標準庫"兩大方向。給予學員 C++新特性之大局觀及最重要成份之深入探討 (特別是影響層面最廣的 Rvalue references, Move Semantics)，全課程含 sample code 測試及解說，及相關標準庫組件之關鍵源碼解說。

◎適合：C++中高階學員

◎時數：12-18 小時 (時數將影響授課內容之多寡)

◎大綱：

* Modern C++ Overview.
* Part I : Language :
- auto, type deduction

・Uniform Initialization, std::initializer_list
・Ranged-Based for loop,
・Lambdas,
・Move Semantics and Rvalue References,
・Perfect Forwarding,
・Variadic Templates,
・constexpr, decltype,...

* Part II : Standard Library:
・Pairs, (std ::pair)
・Tuples, (std ::tuple)
・Type Traits,
・Unordered Containers, (std::unordered_set, std::unordered_map)
・A good enough hash function provided by BOOST,
・move-aware Containers,
・Data Structures of all Containers and Iterators,
・Smart Pointers (especially std::shared_ptr)

* (optional) Introductions of Clocks & Times, Concurrency & Multithreading,
* (optional) Introductions of Concepts, Filesystem,

■5, Generic Programming and STL Architectures (泛型編程與 C++標準庫體系結構)
◎說明：泛型編程 (GP) 和面向對象編程 (OOP) 並列 C++ 最重要的兩大編程思維，前者更是 C++ 標準庫賴以架構的技術，重要性不言可喻。至於 STL / C++ 標準庫 更是每位 C++程序員不可須臾離之的日常工具，其架構博大精深，非常值得梳理，而了解其源碼中的精要亦有助於學習到最高階的編程手段。
◎特色：首先探討 C++ templates 的三種形式，然後引導學員認識 C++標準庫 (主要是 STL) 體系結構。以眾多圖示表現繁複的 containers, algorithms, allocators, adapters, iterators, functors.
◎適合：C++中高階學員
◎時數：12-15 小時
◎大綱：
* 泛型編程 (GP) 大局觀.
* C++ Templates : class templates, function templates, member templates.
* C++ templates 之泛化 vs. 特化 (specialization)
* C++標準庫體系結構 / STL 六大組件 之大局觀

* 分述六大組件，及彼此關係 (這是了解整個體系結構的最重點關鍵)
* 所有容器/Containers 之特性介紹、最佳運用場合(含例)、精要圖示、關鍵源碼引介
* 難以想像而又技術精妙之適配器/Adapters
* 分配器/Allocators 在 VC, BC, GCC 中的實現 (optional)

■6, Effective C++ and More (C++ 編程之專家經驗) (本課程不涵蓋 Modern C++)
◎說明：C++ 範圍廣泛，可謂是個十分複雜 (也許最複雜) 的語言。落實各個細節，內化為良好的編程風格，十分有益。
◎特色：本課程以業界極富盛名之《Effective C++》書籍為藍本，從中取出最有價值的專家經驗 (expertise)，加上我個人的補充。
◎適合：C++各級程序員
◎時數：12 ~ 18 小時 (取決於客戶要求的份量和預算)
◎大綱：(以下是默認選項，可能彈性選擇和增減，實際取決於現場情況)
・Item 13: Use objects to manage resources
・Item 54: Familiarize yourself with the standard library, including TR1
・Item 55: Familiarize yourself with Boost
・Item 14: Think carefully about copying behavior in resource-managing classes
・Item 5: Know what functions C++ silently writes and calls
・Item 6: Explicitly disallow the use of compiler-generated functions you do not want
・Item 7: Declare destructors virtual in polymorphic base classes
・Item 29: Strive for exception-safe code
・Item 11: Handle assignment to self in operator=()
・Item 12: Copy all parts of an object
・Item 32: Make sure public inheritance models “is-a”
・Item 34: Differentiate between inheritance of interface and inheritance of implementation
・Item 36: Never redefine an inherited non-virtual function
・Item 38: Model “has-a” or “is-implemented-in-terms-of” through composition
・Item 39: Use private inheritance judiciously
・Item 2: Prefer consts, enums, and inlines to #defines
・Item 3: Use const whenever possible
・Item 4: Make sure that objects are initialized before they¹re used
・Item 18: Make interfaces easy to use correctly and hard to use incorrectly
・Item 20: Prefer pass-by-reference-to-const to pass-by-value
・Item 21: Don¹t try to return a reference when you must return an object
・Item 22: Declare data members private

・Item 26: Postpone variable definitions as long as possible
・Item 28: Avoid returning “handles” to object internals
・Item 50: Understand when it makes sense to replace new and delete
・Item 9: Never call virtual functions during construction or destruction
・Item 25: Consider support for a non-throwing swap
・Item 27: Minimize casting
・Item 31: Minimize compilation dependencies between files
・Item 44: Factor parameter-independent code out of templates

■7, C++ 面向對象深入解析
◎說明：C++很博大，面向對象(Object Oriented)很精深；為了以良好的面向對象觀念和手法來設計和編寫程序，每一位 C++程序員或許都需要自問：我對「面向對象」了解多少？我是不是正確運用了面向對象的觀念？我寫的程序有未來性嗎？
◎特色：深入淺出, 輔以經典實例
◎適合：C++各級程序員
◎時數：12 小時
◎大綱：
* Single class 的 BIG5 : copy constructor, copy assignment operator, move constructor, move assignment operator, destructor.
* 從低階角度看 Class Hierarchies (階層體系)：
・Construction(構造) 和 Destruction(析構) 深究
・’this’ pointer 深究
・’Virtual Functions’ 深究
・’Polymorphism’ 深究
・’Object Model’ 深究 (包含 memory model, virtual mechanism, dynamic binding)
・Abstraction 深究; Liskov Substitution principle(里氏替換原則)
* 從高階角度看 Classes 間的各種關係
・Composition (複合)
・Inheritance (繼承)
・Aggregation (聚合)
・Delegation (委託)
・UML Classes Diagrams 淺釋
・設計模式 : Template Method, Strategy, Observer, Composite, …(實際講解數量取決於現場情況)
・專家經驗 (取自《Effective C++》書籍; 實際講解數量取決於現場情況)
・Item 13: Use objects to manage resources (RAII)

・Item 32: Make sure public inheritance models “is-a”
・Item 34: Differentiate between inheritance of interface and inheritance of implementation
・Item 36: Never redefine an inherited non-virtual function
・Item 38: Model “has-a” or “is-implemented-in-terms-of” through composition
・Item 39: Use private inheritance judiciously
・Item 18: Make interfaces easy to use correctly and hard to use incorrectly
・Item 9: Never call virtual functions during construction or destruction
・Item 25: Consider support for a non-throwing swap
・Item 31: Minimize compilation dependencies between files

■8 Effective Modern C++
◎說明：本課程內容全部以《*Effective Modern C++*》一書為藍本。這本書很難啃，對讀者的要求很高，然而書中內容對於 "真真正正深入" Modern C++ (since 2011) 有極大幫助，其中有實用性高的指導綱領，例如強烈推薦使用 using, const_iterator, override, decltype, noexcept, constexpr, lambda，也有深入原理的堅實技術，例如 type deduction(型別推導), R-value reference(右值引用), auto 背後原理, std::move, std::forward, std::ref 等等。
◎特色：本課程將《*Effective Modern C++*》書中的代碼片段全部實現為完整可運行的程序，並輔以我的更多實驗和延伸觀察(例如追蹤標準庫源碼)。
◎適合：已熟悉 C++(98)或已開始使用 Modern C++而對以下條款感興趣者。
◎時數：12-18 小時。 (客戶自決。授課內容之多寡取決於時數)
◎大綱：以下列出《*Effective Modern C++*》書中所有 42 個條款 (items)，星號是必講內容(最重要)，其他自由選擇或由老師根據時數安排。
Item 1: Understand template type deduction.
Item 2: Understand auto type deduction.
Item 3: Understand decltype. ★
Item 4: Know how to view deduced types.
Item 5: Prefer auto to explicit type declarations. ★
Item 6: Use the explicitly typed initializer idiom when auto deduces undesired types.
Item 7: Distinguish between () and {} when creating objects. ★
Item 8: Prefer nullptr to 0 and NULL.
Item 9: Prefer alias declarations to typedefs. ★
Item 10: Prefer scoped enums to unscoped enums.
Item 11: Prefer deleted functions to private undefined ones. ★
Item 12: Declare overriding functions override. ★
Item 13: Prefer const_iterators to iterators.

Item 14: Declare functions noexcept if they won’t emit exceptions. ★
Item 15: Use constexpr whenever possible.
Item 16: Make const member functions thread safe. ★
Item 17: Understand special member function generation. ★
Item 18: Use std::unique_ptr for exclusive-ownership resource management. ★
Item 19: Use std::shared_ptr for shared-ownership resource management. ★
Item 20: Use std::weak_ptr for std::shared_ptr-like pointers that can dangle.
Item 21: Prefer std::make_unique and std::make_shared to direct use of new. ★
Item 22: When using the Pimpl Idiom, define special member functions in the implementation file. ★
Item 23: Understand std::move and std::forward. ★
Item 24: Distinguish universal references from rvalue references. ★
Item 25: Use std::move on rvalue references, std::forward on universal references. ★
Item 26: Avoid overloading on universal references.
Item 27: Familiarize yourself with alternatives to overloading on universal references.
Item 28: Understand reference collapsing.
Item 29: Assume that move operations are not present, not cheap, and not used.
Item 30: Familiarize yourself with perfect forwarding failure cases.
Item 31: Avoid default capture modes. ★
Item 32: Use init capture to move objects into closures.
Item 33: Use decltype on auto&& parameters to std::forward them.
Item 34: Prefer lambdas to std::bind. ★
Item 35: Prefer task-based programming to thread-based. ★
Item 36: Specify std::launch::async if asynchronicity is essential.
Item 37: Make std::threads unjoinable on all paths. ★
Item 38: Be aware of varying thread handle destructor behavior.
Item 39: Consider void futures for one-shot event communication.
Item 40: Use std::atomic for concurrency, volatile for special memory.
Item 41: Consider pass by value for copyable parameters that are cheap to move and always copied.
Item 42: Consider emplacement instead of insertion. ★

關於網課(直播)：
1, 請尊重智財權；客戶請勿以任何方式錄影錄音。
2, 對客戶僅提供紙本講義；電子文件不開放。
3, 網課(直播)所用之軟件，由客戶決定 (例如 ZOOM, Microsoft Teams, Webex, 騰訊會議…)
4, 根據我個人網課直播上百小時的經驗，連續六小時網課的學習效果不佳，因此強烈建議每次最多實施 3 小時，分次進行。