C++ MASTER Series 소개

및 수강 안내

❑ C++ MASTER SERIES 소개

C++ 은 현존하는 프로그램 언어 중 가장 복잡하고 어려운 언어 중 하나입니다.

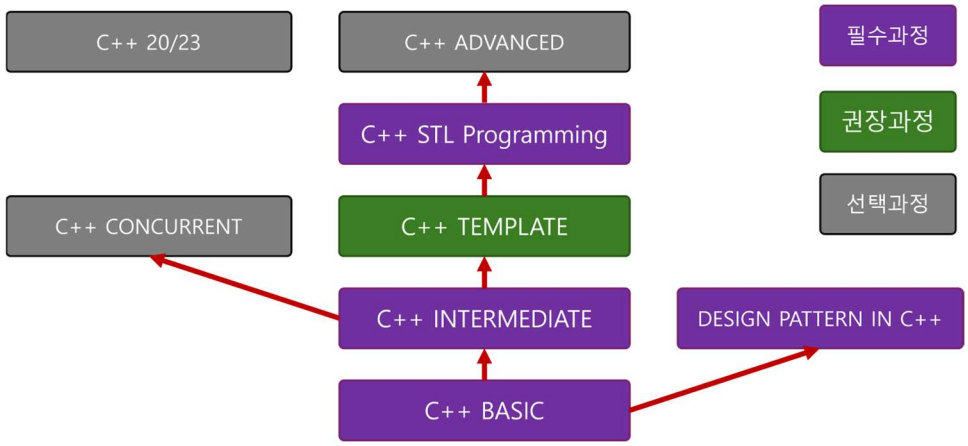
"C++ MASTER SERIES" 는 복잡하고 어려운 C++ 언어를 빠르고 정확하기 이해하기 위해 설계된 과정입니다. C++98 의 legacy 기술부터 C++20/23 의 최신 기술까지를 모두 배울 수 있게 설계 되었습니다.

현재, 아래의 총 8개의 과정으로 구성되어 있습니다.

필수과정 4개	① C++BASIC ② C++INTERMEDIATE ③ C++ STL PROGRAMMING ④ DESIGN PATTERN IN C++
권장과정 1개	⑤ C++ TEMPLATE PROGRAMMING
선택과정 3개	⑥ C++ CONCURRENT PROGRAMMING ⑦ C++ ADVANCED ⑧ C++20/23

❑ 권장 수강 순서

개인별 상황에 따라 다를 수 있지만, 아래의 수강순서를 권장합니다.



❑ 필수 과정 4개

필수 과정은 C++을 정확히 이해 하기 위해 되도록이면 반드시 수강을 권하는 과정입니다

아래의 4개 과정입니다.

C++ BASIC	C++ 언어에서 반드시 알아야 하는 것을 배우는 과정입니다. 타입, 변수, 함수, 제어문, 반복문 등 프로그램의 기본 요소 중에서 C 언어에는 없는 요소. 객체지향 개념과 클래스 상속 문법 연산자 재정의, 예외 등의 문법 C++ 표준 라이브러리인 STL 에 대한 간단한 사용법 등을 배우게 됩니다.
C++ INTERMEDIATE	C++ 기본에서 배우지 못한 고급 문법 과 기법을 배우는 과정입니다. 생성자 고급 기술, CONVERSION, TRIVIAL, PLACEMENT NEW RVALUE REFERENCE, FORWARDING REFERENCE, PERFECT FORWARINDG, MOVE SEMANTICS THISCALL, DEDUCING THIS, FUNCTION OBJECT, LAMBDA EXPRESSION, COROUTINE 등.. C++98 ~ C++23 까지의 기술 중 C++ BASIC 에서 다루지 못한 핵심 기술을 다루는 과정입니다.
C++ STL PROGRAMMING	C++표준 라이브러리인 STL 라이브러리를 배우는 과정입니다. 단순한 사용법 뿐 아니라, 설계 철학, 구현 원리, 성능 등 다양한 관점에서 STL 을 이야기 합니다. C++98 의 CONTAINER, ITERATOR, ALGORITHM 뿐 아니라 C++20 의 CONSTRAINT ALGORITHM 의 특징, RANGES, VIEW 의 새로운 개념과 도구를 배우게 됩니다 SMART POINTER, CHRONO, std::bind, std::function 등의 다양한 utility 도 배우게 됩니다.
DESING PATTERN IN C++	객체지향 디자인 패턴을 배우는 과정입니다. C++ 언어의 문법적인 내용이 아닌, 프로그램을 객체지향으로 설계할 때 사용하는 다양한 코딩 관례(패턴)을 배우게 됩니다. GOF's 디자인 패턴을 기반으로 해서, C++ 오픈소스가 사용하는 다양한 기술을 배우게 됩니다. C++ BASIC 정도의 문법만 알고 있으면 수강할 수 있습니다.

❑ 수강 권장 과정 1개

C++ TEMPLATE PROGRAMMING

C++ 의 template 기술은 일반 개발자 보다는 라이브러리 설계자들의 주로 사용하는 기술 입니다. 따라서, 일반 개발자가 반드시 알아야 하는 것은 아닙니다.

하지만, 갈수록 오픈소스의 C++ 코드가 복잡해지고 있고, C++ 표준 라이브러리 인 STL 같은 라이브러리의 설계 원리를 이해 하려면 반드시 template 문법을 알아야 합니다. Template 이라는 문법 자체가 쉽지는 않습니다.

본 과정에서는 template 문법 뿐 아니라 다양한 코딩 관례(Idioms) 도 배울 수 있습니다.

① Function template, class template 관련 문법
② using template, variable template 관련 문법
③ C++20 의 concept 개념, 문법, 표준 concept 라이브러리 소개
④ Specialization 과 Type traits 기술
⑤ Variadic template 문법
⑥ tuple/get 구현원리
⑦ SFINAE, std::enable_if, CRTP, Policy Base Design 등의 C++ Idioms

등을 배울 수 있습니다.

❑ 선택 과정 3개

아래의 과정들은 필요하신 경우만 수강하시면 됩니다.

C++ CONCURRENT	multi thread 와 concurrent programming 을 위해서 C++ 라이브러리의 제공하는 클래스를 배우는 과정입니다. std::thread, std::jthread 의 기본 클래스 promise & future, packed_task, async mutex, semaphore, latch, barrier 등의 동기화 도구 lock_guard, unique_lock, shared_lock, scoped_lock 등의 lock management 기술 std::atomic, memory_order, thread fence parallel stl 지원 등에 대해서 배우게 됩니다.
C++20/23	C++98 ~ C++17 까지의 기술을 알고 계신 개발자를 대상으로 C++20/23 에서 추가된 내용만 빠르게 습득할수 있도록 설계된 과정입니다. C++20 Big4 인 concept, ranges, coroutine, module C++23 의 핵심인 deducing this, auto rvalue, static operator() 등을 배우게 됩니다. **본 과정에서 다루는 내용은 주제별로 다른 과정과 겹치는 내용도 있습니다.**
C++ Advanced	STL 내부 코드나 오픈소스에서 발췌한 코드를 직접 구현하면서 설계원리와 다양한 기법을 배우는 과정입니다. std::unique_ptr, std::vector, std::advance 등의 주요 루틴을 직접 구현하면서 설계원리를 배우는 과정입니다. 또한, C++20 의 ranges 라이브러리를 직접 구현하면서 다양한 고급 기법도 배울수 있습니다. Android, webkit 등의 오픈소스에서 사용하는 몇가지 고급 기법도 배울 수 있습니다.

APPEDIX 1-2. C++INTERMEDIATE 강의 세부 항목

SECTION	UNIT	TITLE
SECTION #0. OT	OT	**C++ 실습환경안내**
		using compiler explorer
		install mingw
		install visual studio
		using cl compiler
SECTION #1	CONSTRUCTOR	생성자 호출원리 1
		생성자 호출원리 2
		std::initializer_list & CONSTRUCTOR
	CONVERSION	conversion #1
		conversion #2
		converion 예제
	NEW	new, delete, placement new
		왜 메모리 할당과 생성자 호출을 분리 하는가 ?
	TRIVIAL, STANDARD LAYOUT	trivial default constructor
		trivial copy constructor
	TYPE DEDUCTION	TYPE DEDUCTION#1
		TYPE DEDUCTION#2
		TEMPLATE_AUTO_TYPE_DEDUCTION
		DECAY
SECTION #2. LVALUE, RVALUE & FORWARDING REFERENCE	EXPRESSION	EXPRESSION
		UNEVALUATED EXPRESSION
		VALUE CATEGORY
		DECLTYPE(EXPRESSION)
		INSPECT VALUE_CATEGORY
		DECLTYPE(AUTO)
	TEMPORARY	TEMPORARY OBJECT
		REFERENCE RETURN
		COPY ELISION
		TEMPORARY MATERIALIZATION
		TEMPORARY LIFETIME
	LVALUE, RVALUE REFERENCE	RVALUE REFERENCE & REFERENCE RULE
		REFERENCE & FUNCTION OVERLOADING
	FORWARDING REFERENCE	REFERENCE_COLLAPSING
		FORWARDING_REFERENCE#1
		FORWARDING_REFERENCE#2
		FORWARDING_REFERENCE#3
SECTION #3. PERFECT FORWARDING & MOVE SEMANTICS	PERFECT FOWARDING	perfect forwarding #1
		perfect forwarding #2
		perfect forwarding #3
		chronometry 주의 사항
	MOVE SEMANTICS #1	MOVE SEMANTICS
		MOVE CONSTRUCTOR
		COPY ELISION & MOVE CONSTRUCTOR
		STD::MOVE
		IMPLEMENT STD::MOVE
		IMPLEMENT MOVE OPERATION

	MOVE SEMANTICS #2	MOVE & NOEXCEPT
		DEFAULT COPY & MOVE
		ARRAY & MOVE
	MOVE SEMANTICS #3	SETTER & MOVE #1
		SETTER & MOVE #2
		SETTER & MOVE #3
SECTION #4 THISCALL & DEDUCING THIS	THISCALL	this call
		멤버 함수 포인터
	POINTER TO MEMBER	멤버 함수 포인터
		멤버 데이타 포인터
		max 만들기 #1
		max 만들기 #2
		멤버 함수 포인터의 크기
SECTION #5. FUNCTION OBJECT & LAMBDA EXPRESSION	FUNCTION OBJECT	Function Object 개념
		상태를 가지는 함수
		Function Object & Closure
		인라인 치환성 #1
		인라인 치환성 #2
		ADL 과 함수 객체
		std::less 구현하기
		transparent function object 개념
		static operator()
	LAMBDA EXPRESSION #1	lambda expression 기본
		lambda expression 의 원리 #1
		lambda expression 의 원리 #2
	LAMBDA EXPRESSION #2	람다 표현식과 타입
		함수 포인터로의 변환
		람다표현식을 컨테이너에 담는 방법
		람다 표현식과 단위 전략
		generic lambda & template lambda