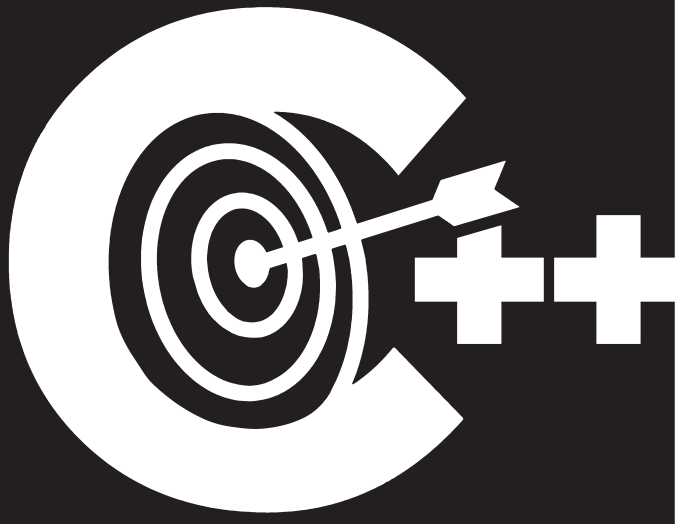
쉽게 설명한
C++ 핵심 가이드라인

C++ Core Guidelines Explained
by Rainer Grimm

쉽게 설명한 C++ 핵심 가이드라인

전자책 1쇄 발행 2023년 5월 18일 **지은이** 라이너 그림 **옮긴이** 류광 **펴낸이** 한기성 **펴낸곳** (주)도서출판인사이트 **편집** 정수진
등록번호 제2002-000049호 **등록일자** 2002년 2월 19일 **주소** 서울시 마포구 연남로5길 19-5 **전화** 02-322-5143 **팩스** 02-3143-5579 **블로그** https://blog.insightbook.co.kr **이메일** insight@insightbook.co.kr **ISBN** 978-89-6626-411-7

프로그래밍 인사이트



쉽게 설명한
C++ 핵심 가이드라인

라이너 그림 지음 | 류광 옮김

인사이트

차례

엄선한 C++ 핵심 가이드라인 규칙들

옮긴이의 글

한때 시대를 주도하던 C++은 언제부터인가 변화에 적응하지 못한 공룡 취급을 받기도 했습니다. 그러나 다행히도 시기를 크게 놓치지 않고 "C++ 현대화" 작업이 시작되어서 C++11과 C++14, C++17을 거쳐 C++20까지 나왔고, C++23도 정식 발표를 눈앞에 두고 있습니다. 이러한 C++ 현대화의 흐름 속에서 저는 도서출판 인사이트의 후의 덕분에 2015년의 《이펙티브 모던 C++》에서 2022의 《필요한 것만 골라 배우는 모던 C++》까지 현대적 C++(모던 C++)에 관한 여러 좋은 책을 번역하는 행운을 누렸습니다. 그리고 마치 그 책들에 담긴 지혜와 통찰을 한 권의 책으로 정리할 때가 왔다는 듯이 "C++ 핵심 가이드라인(C++ Core Guidelines)"을 해설한 *C++ Core Guidelines Explained: Best Practices for Modern C++*(Addison-Wesley Professional, 2021)가 세상에 등장했고, 그 책을 부족하나마 제가 번역해서 본서 《쉽게 설명한 C++ 핵심 가이드라인》을 독자 여러분께 선보이게 되었습니다.

C++은 방대하고 복잡한 언어로 알려져 있고 실제로 방대하고 복잡하지만, 그러한 방대함과 복잡함은 본질적으로 현실의 반영일 뿐 그 자체를 단점으로 보는 것은 바람직하지 않겠습니다. 예를 들어 어떤 여객용 제트 비행기의 설계도가 세발자전거의 설계도만큼이나 단순하다면 오히려 걱정스러울 것입니다. 문제는 그런 복잡한 언어를 어떻게 나의 요구에 맞게 다스리고 활용할 것인가인데, 여기에는 지식뿐만 아니라 지혜와 통찰이 크게 작용합니다. 오랜 경험 없이도 그런 지혜와 통찰을 더 많은 C++ 프로그래머가 갖출 수 있게 하려는 노력의 하나가 바로 이 책이 다루는 C++ 핵심 가이드라인입니다.

C++ 생태계의 두 주요 인사인 비야네 스트롭스트룹(C++의 창시자)과 허브 서터(C++ 표준 위원회 의장)가 주도하는 오픈소스 프로젝트(*https://github.com/isocpp/CppCoreGuidelines*)인 C++ 핵심 가이드라인은 경험, 지혜, 통찰이 가득한 보물상자와도 같지만, 라이너 그림은 C++ 핵심 가이드라인이 "좀 더 많은 사람이 접하기에 이상적인 형태는 아니"라고 생각했습니다. 그래서 저자는 C++ 핵심

가이드라인의 규칙들을 해설한 글을 써서 자신의 블로그를 비롯한 여러 매체에 발표했고, 급기야는 한 권의 책으로 내게 되었습니다.

이상의 소개에서 짐작하겠지만, 그리고 저자 서문을 읽으면 더욱 확실해지겠지만, 이 책은 C++ 언어를 처음 배우려는 사람이 아니라 현대적 C++을 좀 더 효과적이고 안전하게 활용하고자 하는 사람을 위한 책입니다. 따라서 C++11에서 시작하는 현대적 C++에 관한 기본적인 지식이 필요합니다. C++에 익숙하다고 생각하는 프로그래머라도, 예를 들어 `auto mult = [&val](auto&& x) -> decltype(val) { return x * val; };` 같은 문장(현대적 C++의 특징적인 요소들을 보여주기 위해 작위적으로 만든 것입니다)을 보고 "이게 뭐야? 내가 아는 C++이 아닌데?"라는 생각이 든다면 C++을 새로이 공부할 때가 된 것입니다. 미리 현대적 C++을 어느 정도 체계적으로 공부한 후 이 책으로 돌아올 수도 있고 이 책을 읽다가 모르는 부분이 나오면 그때그때 공부할 수도 있겠는데, 어떤 경우이든 2015년부터 제가 번역한 도서출판 인사이트의 현대적 C++ 관련 번역서들이 도움이 될 것입니다. 제 번역서들은 홈페이지 **류광의 번역서 이야기**(*https://occamsrazr.net*)의 "번역서 정보" 섹션에서 확인할 수 있습니다.

저자 서문의 '판타 레이' 절에 나오듯이 C++ 핵심 가이드라인은 계속해서 변합니다. 번역하면서 원서에 나온 규칙과 C++ 핵심 가이드라인에 실제로 나온 규칙이 조금 다른 경우를 몇 개 발견했는데, 독자가 저자 서문의 판타 레이 부분을 읽었으리라고 가정하고 별다른 역주 없이(이런 사실 자체를 언급하는 역주를 빼고는) 번역문에 새로운 내용을 반영했음을 알려 드립니다. 그밖에 원서의 문장을 그대로 옮기는 대신 "저자가 쓰려고 했을" 또는 "썼어야 했을" 문장을 짐작해서 옮긴 경우도 있는데, 어쩌면 제가 오해한 것일 수도 있겠습니다. 내용상의 오류와 오역, 오타를 발견하면 다른 독자들을 위해서라도 제게 꼭 알려주시기 바랍니다. 앞서 언급한 제 홈페이지의 번역서 정보 섹션에 이 책을 위한 페이지로 가는 링크가 있습니다.

감사 인사로 옮긴이의 글을 마무리하고자 합니다. 먼저, 현대적 C++ 책들을 꾸준히 내주시는 도서출판 인사이트 한기성 사장님께 감사합니다. C++의 현대화에는 C++ 언어 자체뿐만 아니라 C++ 사용자의 현대화도 포함된다는 점에서, 좋은 책의 출간은 아주 중요한 문제입니다. 그리고 이번에도 번역과 교정 과정을 매끄럽게 진행해 주신 정수진 편집자님과 다소 산만한 원서의 조판과는 달리 차분하면서도 지루하지 않은 조판으로 번역서의 품격을 높여 주신 조판 디자

이너 최우정님께 감사드립니다. 그 밖에도 이 책의 출간에 많은 분이 기여하셨지만, 일일이 거론하지 못해 죄송할 따름입니다. 마지막으로, 여러 가지 악조건 속에서도 세밀한 교정·교열로 미숙한 원고를 출판에 적합한 원고로 탈바꿈해 준 아내 오현숙에게 감사와 사랑의 마음을 전합니다.

재미있게 읽으시길!

옮긴이 류광

추천사

C++은 수많은 기능을 가진, 대단히 다채롭고 표현력이 큰 언어이다. C++이 그런 언어가 된 것은 당연하다면 당연하다. 범용 프로그래밍 언어가 성공하려면 한 사람의 개발자가 필요로 하는 것보다 더 많은 기능을 제공해야 하며, 언어가 계속 살아남고 진화하다 보면 어떠한 하나의 개념을 표현하는 다양한 관용구들이 축적되기 때문이다. 하지만 그러면 개발자가 선택할 것이 너무 많아질 수 있다. 개발자는 다양한 프로그래밍 스타일과 전문 지식을 적절하게 선택하는 능력을 갖추어야 한다. 또한, 낡고 효과적이지 않은 기법과 프로그래밍 스타일에 묶여서 더 나아가지 못하는 사태를 피할 수 있어야 한다.

C++ 핵심 가이드라인(C++ Core Guidelines)[1]은 그런 문제점들을 해결하기 위해 현대적 C++의 널리 알려진 모범 관행(best practice)들을 한곳에 모으려는 오픈소스 프로젝트로, 지금도 계속 진행 중이다. C++ 핵심 가이드라인은 수십 년간의 경험과 기존 코딩 규칙들에 기반한다. C++ 핵심 가이드라인의 지침들과 규칙들은 C++ 자체와 개념적 틀을 공유하며, 형식 안전성(type safety)과 자원 안전성, 그리고 피할 수 있는 복잡성과 비효율성의 제거에 초점을 둔다. C++ 핵심 가이드라인은 알려진 문제 영역들을 해결하기 위한 지침들과 규칙들을 제공한다. 그중 일부는 정적 코드 분석 도구를 이용해서 집행할 수 있도록 만들어졌다.

C++ 핵심 가이드라인은 특정한 하나의 주제를 손쉽게 참조하고 공유하기 쉽도록 참고자료의 형태로 구성되어 있다. 다른 말로 하면, C++ 핵심 가이드라인은 현대적 C++을 잘 활용하기 위해 처음부터 끝까지 차례대로 읽으면서 여러 주제를 익힐 수 있는 튜토리얼이 아니다. 그런 만큼, C++ 핵심 가이드라인의 규칙들을 좀 더 많은 사람이 수월하게 익히게 한다는 어렵고도 꼭 필요한 작업을 저자라이너 그림이 자신의 교육 기술과 업계 경험을 적용해서 완수했다는 것은 우리 C++ 핵심 가이드라인 편집자들에게 매우 기쁜 소식이다. 독자가 이 책으로 C++ 핵심 가이드라인을 익히면서 많은 영감을 얻기를, 그리고 C++ 핵심 가이드라인을

1 *https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md*

실제 업무에 적용함으로써 훨씬 더 효과적이고 즐겁게 일할 수 있게 되길 희망한다.

비야네 스트롭스트룹Bjarne Stroustrup

허브 서터Herb Sutter

C++ Core Guidelines Explained

서문

이 서문의 목적은 단 하나, 친애하는 독자가 이 책을 최대한 활용하는 데 필요한 배경지식을 전달하는 것이다. 이 배경지식에는 저자인 나에 관한 세부사항과 내 저술 스타일, 이 책을 쓰게 된 동기, 그리고 이런 책을 쓰면서 어려웠던 점이 포함된다. 원한다면 이 서문을 건너뛰어도 좋지만, 서문 다음에 나오는 감사의 글은 꼭 읽어주기 바란다.

관례

다음은 이 책 전반에 쓰이는 그리 많지 않은 관례들이다.

규칙 대 지침

C++ 핵심 가이드라인은 지침(guideline)을 '규칙(rule)'이라고 칭할 때가 많다. 그래서 이 책에서 규칙이라는 용어를 사용하기로 한다. 이 책 전반에서 지침과 규칙은 사실상 같은 말이다.

특별한 글꼴

- 중요한 용어나 문구는 **굵은 글자**나 고딕체로 표시한다.
- 소스 코드, 명령, 키워드, 형식, 변수, 함수, 클래스 이름은 `name`처럼 고정폭 글꼴로 표시한다.

관련 규칙

C++ 핵심 가이드라인에는 서로 연관된 규칙들이 있다. 일부 장(챕터)들은 그 장에서 설명한 규칙들과 관련된 규칙들을 장의 끝부분의 '관련 규칙' 절에서 소개한다.

요약 글상자

대부분의 장은 그 장의 주요 내용을 불릿 목록으로 요약한 글상자로 끝난다.

> **요약**
>
> **주요 사항**
> • 이번 장의 주요 내용

소스 코드

나는 `using` 선언과 `using namespace` 지시문을 좋아하지 않는다. 소스 코드에 쓰인 이름들의 출처(origin)가 무엇인지(이를테면 이 함수가 어떤 라이브러리에 속한 것인지) 알기 어렵기 때문이다. 그러나 지면의 제약 때문에 종종 `using` 지시문과 `using namespace` 선언을 사용해야 했다. `using namespace std;`나 `using std::cout;` 등 익숙한, 따라서 출처를 충분히 짐작할 수 있는 이름들에 대해서만 지시문과 선언을 사용하려고 노력했다. 일부 헤더는 `#include` 문을 생략하기도 했다. 부울 값은 `true` 또는 `false`로 출력된다고 가정했으며, 이에 필요한 입출력 스트림 조작자 `std::boolalpha`는 굳이 예제 코드에 포함시키지 않았다.

특별한 언급이 없는 한 예제 코드에서 마침표 세 개(`...`)는 코드 생략을 뜻한다(가변 인수가 아니라).

예제 코드가 하나의 완결적인 프로그램일 때는 첫 행에 소스 파일 이름을 주석으로 표시했다. 독자가 적어도 C++14를 지원하는 컴파일러를 사용한다고 가정하고 예제 코드를 작성했다. C++17이나 C++20이 필요한 경우에는 소스 파일 이름 다음에 해당 C++ 표준을 표시해 두었다.

소스 코드를 설명하기 편하도록 코드에 `// ❶` 같은 주석을 붙이기도 했다. 본문에서 설명할 코드가 있는 행(지면이 여의치 않으면 그 행 바로 위의 행)에 그런 숫자를 표시해 두었다. 이 책(원서)의 깃허브 저장소(*https://github.com/*

RainerGrimm/CppCoreGuidelines)에 있는 소스 코드에는 그런 표시가 없다. 또한, 지면의 제약으로 이 책에서는 소스 코드의 포매팅(줄 바꿈, 들여쓰기)을 저장소에 있는 것과 다르게 조절했다는 점도 밝혀 둔다.

C++ 핵심 가이드라인에 있는 예제 코드를 가져온 경우에는 가독성을 위해 `using namespace std`를 추가하거나 포매팅을 변경하기도 했다.

C++ 핵심 가이드라인에 주목한 이유

내가 C++ 핵심 가이드라인을 설명하는 책을 쓰기로 한 데에는 C++과 파이썬, 그리고 소프트웨어 전반에 대한 교육자로 15년을 보낸 경험이 크게 작용했다. 지난 몇 년 동안 나는 심장 제세동기(defilbrillator)에 내장되는 소프트웨어를 개발하는 팀을 이끌었다. 내 책임에는 제세동기에 관한 규제 조항들을 준수하는 것도 포함되었다. 제세동기는 환자와 사용자의 사망이나 치명상을 유발할 수도 있는 장치인 만큼, 그런 장치를 위한 소프트웨어를 작성하는 것은 대단히 어려운 일이다.

이 책은 C++ 공동체의 일원으로서 우리가 반드시 답해야 하는 질문 하나에 기초한 것이다. 그 질문이란, "우리에게 현대적 C++을 위한 지침(guideline)들이 필요한 이유는 무엇인가?"이다. 그리고 이 질문에 대한 답은 지난 경험에서 얻은, C++에 대한 나의 세 가지 의견에서 찾을 수 있다.

C++은 초보자에게 너무 복잡하다

C++은 원래부터 복잡한(complex) 언어이고, 초보자는 특히나 복잡하게 느낀다. 근본적인 이유는 C++로 풀고자 하는 문제들 자체가 원래부터 까다롭고(complicated) 복잡할 때가 많기 때문이다. C++을 누군가에게 가르칠 때는 모든 용례(use case)의 적어도 95%에서 유효한 규칙들을 학습자에게 제시해야 한다. 내가 생각하는 그런 규칙들은 다음과 같다.

- 형식(type)을 컴파일러가 연역하게 한다.
- 초기화에는 중괄호({}) 구문을 사용한다.
- 스레드보다 작업(task)을 우선시한다.
- 원시(raw) 포인터 대신 스마트 포인터를 사용한다.

나는 내 세미나에서 이런 규칙들을 가르친다. 그렇지만 C++ 공동체 전체 차원에서는 좀 더 공식적이고 규범적인 규칙 또는 모범 관행(best practice)들을 만들고 합의할 필요가 있다. 그런 규칙들은 부정적인 방식이 아니라 긍정적인 방식으로 형식화해야 할 것이다. 즉, 코드를 이러저러하게 작성하지 말라는 규칙들이 아니라 이러저러하게 작성하라는 규칙들이어야 한다.

C++은 전문가에게도 쉽지 않다

3년마다 나오는 새 C++ 표준에 포함되는 다수의 새 기능들을 익히는 것은 그리 큰 문제가 아닐 수 있지만, 경력 있는 전문 C++ 프로그래머라도 현대적 C++이 지원하는 새로운 개념들과 착안들에 잘 적응하지 못하는 경우가 있다. 코루틴과 느긋한 평가를 이용한 이벤트 주도적 프로그래밍, 무한 데이터 스트림, 구간 라이브러리(Ranges library)를 이용한 함수 합성을 생각해 보라. 템플릿 매개변수에 의미론적 범주를 도입하는 콘셉츠를 생각해 보라. C 프로그래머가 객체 지향적 프로그래밍의 개념들을 익히기란 꽤 어려운 일일 수 있다. 구식 C++에 익숙한 프로그래머들도 마찬가지이다. 여러분이 새로운 패러다임으로 전환하려면 지금까지 자신이 프로그래밍 문제를 푸는 데 사용해 온 방식을 다시 생각해 보아야 하고, 필요하다면 뜯어고쳐야 한다. 현대적 C++에는 새로운 개념이 많이 도입되었다. 나는 초보자보다는 전문 프로그래머가 오히려 그런 개념들에 적응하기가 더 어려울 것이라고 가정한다. 전문 프로그래머들은 오랫동안 성공적으로 사용해온 문제 해결 기법들을 고집하기 쉬우며, 그러다 보니 소위 **망치-못 함정**(hammer-nail trap)†에 빠지기 쉽다.

C++은 안전성이 중요한 소프트웨어에 쓰인다

내가 가장 신경 쓰는 사항이 이것이다. 안전성이 중요한(safety-critical) 소프트웨어를 개발할 때는 특정한 코딩 지침을 준수해야 할 때가 많다. 안전성을 강조하는 코딩 지침으로 가장 두드러진 예는 MISRA C++이다. MISRA C++은 영국의 *Motor Industry Software Reliability Association*(자동차 산업 소프트웨어 신뢰성 협회)[1]가 발행하는 코딩 지침으로, 현재 버전은 MISRA C++:2008이다. MISRA C++

† [옮긴이] "망치를 들고 있으면 모든 것이 못으로 보인다"라는 속담을 비유한 것으로, 문제에 맞게 도구나 기법을 선택하는 대신 익숙한 도구나 기법에 맞게 문제를 재단하는 오류를 뜻한다.

1 *https://www.misra.org.uk/*

은 1998년 발표된 *MISRA C guidelines*[2]에 기초한다. 원래는 자동차 산업을 위해 만들어진 지침이지만 이제는 항공, 군사, 의료 분야의 안전성 중요 소프트웨어의 구현을 위한 사실상의 표준이 되었다. MISRA C처럼 MISRA C++도 C++의 안전한 부분집합을 위한 지침들을 제공한다. 그런데 개념적인 문제점이 하나 있다. MISRA C++은 현대적인 C++ 소프트웨어 개발을 위한 지침이라고 말하기에는 너무 낡았다. MISRA C++ 이후로 새 C++ 표준이 네 개나 나왔다! 극명한 예로, MISRA C++은 연산자 중복적재(operator overloading)를 허용하지 않는다. 내 세미나에서는 `auto constexpr dist = 4 * 5_m + 10_cm - 3_dm;`처럼 사용자 정의 리터럴을 이용해서 형식에 안전한 산술을 구현해야 한다고 가르친다. 산술 연산자들과 적절한 접미사들을 위한 리터럴 연산자들을 중복적재하지 않고는 이런 형식에 안전한 산술을 구현할 수 없다. 솔직히 나는 MISRA C++이 빠르게 진화해서 현재의 C++ 표준을 따라잡을 것이라고는 믿지 않는다. C++ 핵심 가이드라인처럼 공동체가 주도하는 지침만이 계속 발전하는 C++ 표준과 보조를 맞출 수 있다.

MISRA C++과 AUTOSAR C++14의 병합 계획

하지만 희망이 있다. MISRA C++이 AUTOSAR C++14를 포함할 것이라고 한다. *AUTOSAR C++14*[3]는 C++14에 기초한 지침으로, MISRA C++을 확장하기에 아주 적합하다. 하지만 업계의 단체가 주도하는 지침이 현대적 C++의 역동적인 변화를 계속해서 따라잡을 수 있을지는 심히 의심스럽다.

나의 도전 과제

나는 2019년 5월에 비야네 스트롭스트룹과 허브 서터에게 C++ 핵심 가이드라인에 관한 책을 쓰고 싶다는 메일을 보냈다. 그 메일에서 핵심 부분을 인용하자면 다음과 같다: "저는 C++ 핵심 가이드라인의 가치를 절대적으로 지지합니다. 왜냐하면 저는 우리 C++ 프로그래머들에게 현대적 C++의 올바른/안전한 사용법에 관한 지침들이 꼭 필요하다고 믿기 때문입니다. 제 C++ 강좌에서 C++ 핵심 가이드라인의 예제들과 착안들을 자주 사용하고 있습니다. C++ 핵심 가이드라인의 포맷은 대체로 MISRA C++이나 AUTOSAR C++14의 규칙들과 비슷한데, 아마 의도적

2 *https://en.wikipedia.org/wiki/MISRA_C*

3 *https://www.autosar.org/fileadmin/standards/adaptive/18-03/AUTOSAR_RS_CPP14Guidelines.pdf*

으로 그렇게 하셨겠지만 좀 더 많은 사람이 접하기에 이상적인 형태는 아니라고 생각합니다. 만일 C++ 핵심 가이드라인의 일반적인 개념들을 설명하는 또 다른 문서가 있다면 더 많은 사람이 C++ 핵심 가이드라인의 지침들을 읽고 고민하게 되리라 생각합니다."

이와 관련해서 몇 가지 더 언급하고자 한다. 지난 몇 년 동안 나는 C++ 핵심 가이드라인에 관한 백여 건의 글을 써서 내 독일어 및 영어 블로그에 올렸다. 또한 독일어 잡지 *Linux-Magazin*[4]에서 C++ 핵심 가이드라인에 관한 글을 연재했다. 이런 글들을 쓴 이유는 두 가지이다. 첫째로, 나는 C++ 핵심 가이드라인을 더 많은 사람이 알아야 한다고 생각했다. 둘째로, 나는 C++ 핵심 가이드라인의 규칙들을 좀 더 읽기 쉬운 형태로 설명하고 필요하다면 배경 정보도 함께 제공하고자 했다.

이런 글들을 쓰는 것이 쉽지는 않았다. C++ 핵심 가이드라인에는 지침이 500개가 넘는다. 대부분의 경우 C++ 핵심 가이드라인은 지침들을 그냥 '규칙'이라고 부른다. 이 규칙들은 대부분 정적 분석을 염두에 두고 작성된 것들이다. 전문 C++ 소프트웨어 개발자에게 생명줄과도 같은 규칙들이 많지만, 너무 전문적인 규칙도 많고, 불완전하거나 다른 규칙들과 겹치는 규칙들도 드물지 않다. 심지어는 다른 규칙과 모순되는 규칙들도 있다. 이 책을 쓰면서 내가 설정한 도전 과제는, C++ 핵심 가이드라인의 규칙 중에서 C++을 사용하는 전문적인 소프트웨어 개발자에게 꼭 필요한 주요 규칙들을 선별하고, 거기서 너무 난해한 내용을 제거하고 누락된 배경 정보를 채워서 읽기 좋은(심지어 재미있는) 이야기를 만들어 내는 것이었다.

판타 레이

판타 레이Panta rhei는 그리스 철학자 헤라클레이토스(Heraclitus of Ephesus)[5]의 유명한 말로, "모든 것은 흐른다"라는 뜻이다.‡ 판타 레이는 이 책을 쓰면서 내가 겪은 어려움을 잘 표현한다. C++ 핵심 가이드라인은 깃허브 저장소로 관리되는, 250여 명이 참여한 공개 프로젝트[6]이다. 이 책을 쓰면서 근거로 삼은 규칙들과 문장들이 계속해서 바뀌었고 앞으로도 바뀔 것이다.

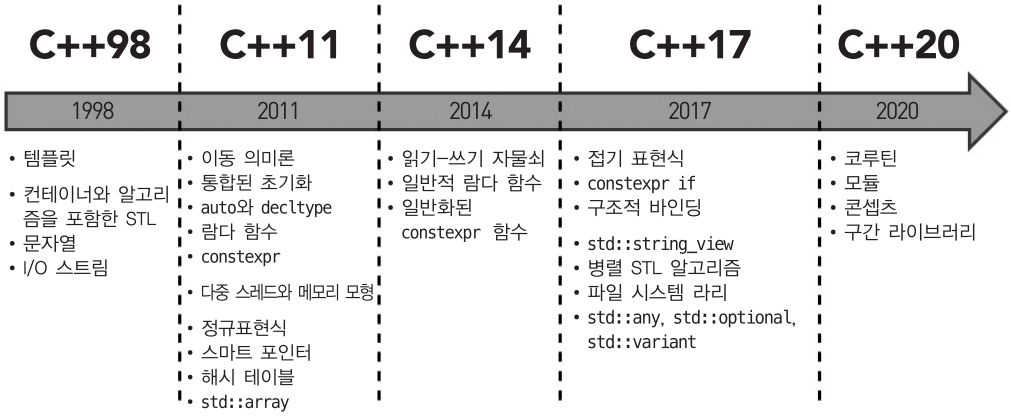
---

4 *https://www.linux-magazin.de/*

5 *https://en.wikipedia.org/wiki/Heraclitus*

‡ [옮긴이] 같은 뜻의 사자성어로 만물유전(萬物流轉)이 있다.

6 *https://github.com/isocpp/CppCoreGuidelines*



게다가 C++ 표준 자체도 계속 진화한다. 실제로 C++ 핵심 가이드라인에는 향후 표준(C++23 등)에 채용될 기능들에 관한 내용도 포함되어 있다. 계약(contract) 관련 규칙들이 대표적인 예이다. 이런 점들을 고려해서 나는 몇 가지 사항을 결정했다.

1. 언급한 C++ 핵심 가이드라인 규칙의 URL을 각주로 제시한다.
2. C++17 표준에 초점을 두되, 필요하다면 C++20 표준과 관련한 규칙들도 제시한다(콘셉츠 등).
3. C++ 핵심 가이드라인은 새 C++ 표준이 발표되면 더욱 크게 변할 것이다. 그에 따라 이 책의 개정판을 낼 계획이다.

이 책을 읽는 방법

이 책의 구성은 C++ 핵심 가이드라인의 구조를 반영한다. 제1부의 장들은 C++ 핵심 가이드라인의 주요 섹션들과 대응되고, 제2부의 장들은 지원 섹션들에 대응된다. C++ 핵심 가이드라인의 섹션들과는 대응되지 않는 부록들도 있다. 부록들에서는 C++ 핵심 가이드라인의 규칙들을 실제로 집행하는 방법과 C++20의 콘셉츠, 그리고 이후 표준에 포함될 계약 기능을 소개한다.

마지막으로, 이번 절의 제목인 "이 책을 읽는 방법"을 이야기하겠다. C++ 핵심 가이드라인의 주요 섹션들에 대응되는 제2부의 장들은 모두 읽기 바란다. 가능하면 각 장을 처음부터 끝까지 읽어 주었으면 한다. 지원 섹션들을 다루는 제2부는 추가적인 정보를 제공하는데, 특히 GSL(Guidelines Support Library)을 소개한다. 제3부의 부록들은 주요 섹션들을 이해하는 데 도움이 되는 필수 배경지식에 대한 참고자료로 의도한 것이다. 이런 추가 정보는 이 책의 완성도에 꼭 필요하다.