세상의 속도를
따라잡고 싶다면

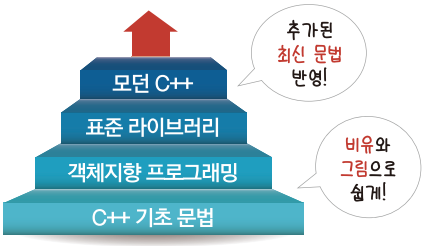
Do it!

기초부터 실전까지 제대로 배운다!

C++ 완전 정복

게임 속 몬스터를 구현하며 25년 차 개발자의 프로그래밍 노하우를 익힌다!

조규남, 문종채 지음



이지스퍼블리싱

좋은 코드는
그 자체로 최고의 문서이다.

"Good code is its own best documentation."

개발자 스티브 매코널
Steve McConnell

“이제 C를 벗어나 C++답게 프로그래밍하자!”

C++를 접하기 전 C 언어를 처음 배울 때에 포인터의 장벽을 넘게 해준 고마운 책이 있었습니다. 그 책으로 공부하면서 언젠가 나도 누군가에게 도움이 되는 책을 써보자고 다짐했습니다. 그리고 27년이 훌쩍 지나 이 책을 내놓습니다.

이 책은 케이트 그레고리(Kate Gregory)의 〈Stop Teaching C〉라는 강연에서 영감을 얻었습니다. 강연 내용을 원고에 녹여 낼 수 있도록 허락해 준 그녀에게 감사의 마음을 전합니다. 원고를 집필하면서 객체지향 원칙을 잘 적용할 수 있을 뿐 아니라 C에서 벗어나 C++만의 고유한 장점을 살릴 수 있는 내용을 담고자 노력했습니다. 또한 복잡하고 잘 사용하지 않는 새로운 문법이나 라이브러리보다는 쉽고 자주 사용하는 C++의 문법과 라이브러리를 중심으로 소개했습니다.

수많은 프로젝트는 여전히 C++ 언어로 개발하고 있습니다. C++ 언어는 특히 실행 성능과 메모리 효율이 절실한 대규모 과제에 적합합니다. 그리고 코드를 더 간결하고 강력하게 작성할 수 있도록 돕는 모던 C++가 계속 발전하고 있습니다. 이 책을 통해 C++의 문법뿐만 아니라 실무에서 활용할 수 있는 코드, 유지·보수하기 쉬운 코드를 어떻게 작성하는지 터득할 수 있기를 바랍니다.

이 책이 나오기까지 많은 분께서 도움을 주셨습니다. 함께 집필해 준 문종채 님, 편집에 힘써 준 이인호 편집자와 이지스퍼블리싱 편집 팀에 감사를 드립니다. 회사의 상사, 동료분들과 학교 지도교수님께도 감사와 사랑을 전합니다. 사랑하는 아내 지예, 든든한 맏이 현준, 귀여운 막내 현민이의 응원과 사랑은 커다란 힘이 되었습니다. 부모님과 가족 모든 분께 감사의 마음을 표합니다.

C++ 프로그래밍에 도전하는 독자분들께 무한한 응원을 보냅니다.

조규남 드림

“누구나 읽기 쉬운 C++ 입문서!”

학창 시절 C++ 언어로 객체지향이라는 개념을 처음 잡을 때 무려 천 쪽이 넘는 두꺼운 원서로 공부했던 아찔한 기억이 있습니다. 그로부터 20여 년이 지난 어느 날 초등학생 아들이 마인크래프트라는 게임을 하다가 C++가 무엇이냐고 묻더군요. 저는 게임 화면에서 “Made in C++!”라는 노란색 문구를 발견할 수 있었습니다. 그 문구와 아들을 보면서 ‘누구나 읽기 쉬운 C++ 입문서를 써보자’라고 생각했습니다.

이 책은 초보자도 쉽게 이해할 수 있도록 난해한 전문 용어를 최대한 자제하고, 쉬운 비유와 그림으로 C++ 프로그래밍의 기본 개념을 설명하려고 노력했습니다. 부담스럽지 않은 두께에 핵심만 담아 C++ 프로그래밍 입문용으로 구성했지만 경험자도 복습하면서 실력을 다져서 실무에서 활용할 수 있도록 고려했습니다.

먼저 집필을 제안해 주신 존경하는 조규남 선배님, 그리고 응원해 준 회사 상사, 동료들과 이인호 편집자님께 깊이 감사드립니다. 인생의 동반자이자 버팀목인 아내 혜란, 그리고 아빠가 하는 모든 일에 관심이 많은 귀여운 아들 성민이에게도 사랑한다고 남기고 싶습니다.

문종채 드림

“객체지향 프로그래밍에 날개를 달아 주는 책!”

C++ 언어는 1985년에 등장한 이후 꾸준히 사용되면서 가치를 증명해 왔습니다. 이 책은 **기본 문법을 시작으로 현업에서 객체지향 언어의 활용 방법을 알려 줍니다.** 이와 더불어 여러 사람이 함께 개발할 때 코딩 규칙 위반 같은 문제를 컴파일 단계에서 발견하는 방법 등 몇 가지 실무 노하우도 소개합니다. 이러한 구성으로 C++를 처음 접하는 입문자부터 다른 프로그래밍 언어에 익숙한 경험자까지 모두에게 유용한 책이 될 것입니다.

황규별 님 • LG유플러스 CDO 전무

이 책은 실용적인 예와 출력 결과를 보여 줌으로써 C++ 언어의 문법을 제대로 이해하고 스스로 응용할 수 있도록 안내합니다. 체계적인 구성과 설명으로 초보자도 쉽게 접근할 수 있어 **객체지향 프로그래밍을 주제로 하는 C++ 교재로 추천**합니다.

유헌창 님 • 고려대학교 정보대학 컴퓨터학과 교수

C++ 같은 객체지향 프로그래밍 언어는 추상적인 개념을 다루기 때문에 어렵게 느껴질 수 있습니다. 그러나 이 책은 C++ 객체지향 프로그래밍을 ‘눈에 보이게’ 설명해 줍니다. **명쾌한 설명과 이를 그림으로 쉽게 이해할 수 있게 해주고, 실용적인 소스 코드**까지 담았으니 궁극의 C++ 책이라고 소개하고 싶습니다. 1997년부터 C++를 다뤄 온 전문가가 C++를 이해하고 쓰는 길로 인도합니다.

임지순 님 • 3PM Inc. 대표, 리얼월드 암호학 등 다수 기술 번역

이 책은 모던한 C++의 아름다움을 탁월하게 담아 냈다고 생각합니다. 특히 **객체지향의 원리와 SOLID 패턴, 유지·보수를 하기 쉽게 해주는 코드 작성, 실무에 적용하는 방법** 등을 균형 있게 다루며 양질의 실습 코드를 제공합니다. 많은 개발자가 C++를 C 스타일로 쓸 때가 많은데, 이 책은 C++를 본연의 방식대로 사용하고 싶은 개발자에게 추천합니다.

맹윤호 님 • 이화여대 신산업융합대학 겸임교수, 카논그룹 CTO

C++의 기초부터 실전까지 한 권에!

첫째마당 | C++ 프로그래밍 기초

쉽고 간결한 언어와 직관적인 그림으로 C++의 기본 개념을 공부합니다.

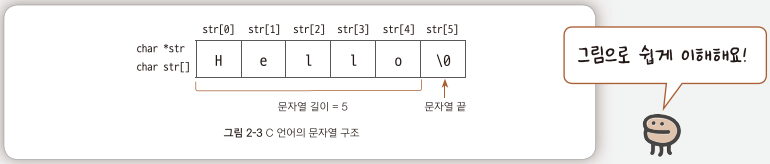
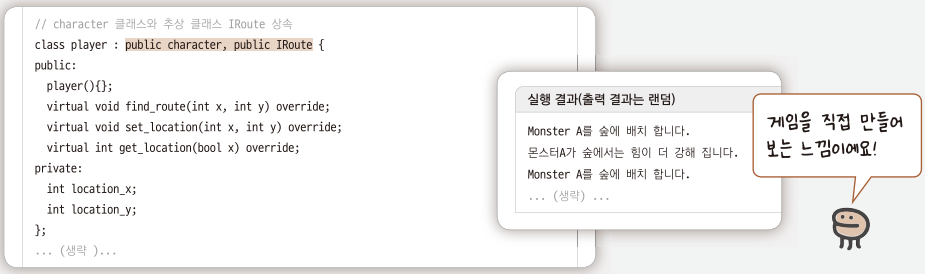


그림 2-3 C 언어의 문자열 구조

둘째마당 | 객체지향 프로그래밍

수백 줄이 넘는 게임 속 몬스터 코드를 구현해 보며 C++를 C++답게 사용해 봅니다. 이 과정에서 객체지향 설계 원칙, 디자인 패턴을 적용하는 법을 배울 수 있습니다.



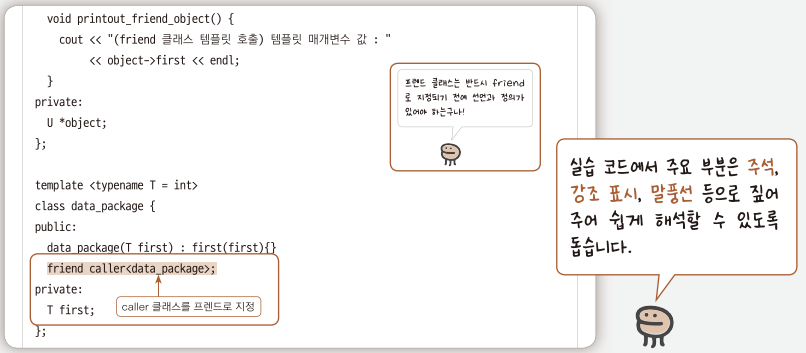
셋째마당 | 라이브러리 활용

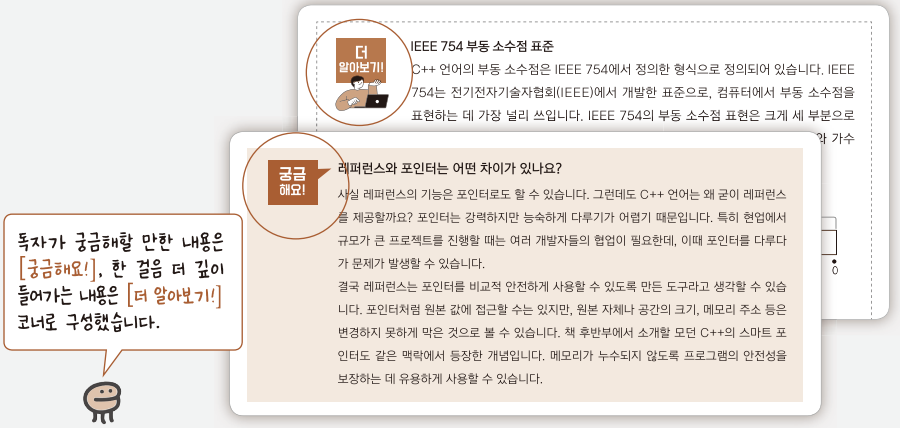
C++ 프로그래밍에 강력한 무기가 되어 줄 C++ 표준 라이브러리 활용법을 배웁니다.

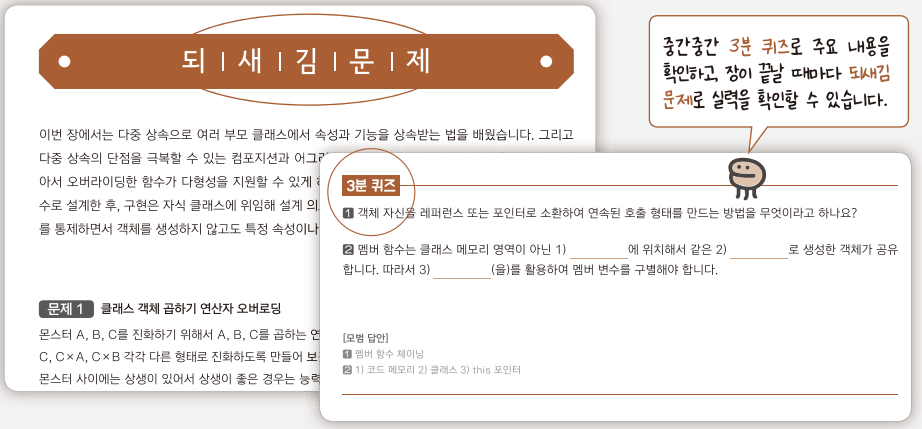
넷째마당 | 모던 C++ 프로그래밍

C++는 지금도 계속 진화하고 있습니다. 모던 C++에 추가된 내용 중에서 실무에서 많이 사용하는 것을 살펴봅니다.

C++이 어려워서 걱정인가요?
혼자서도 완독할 수 있게 친절히 구성했어요!







실습 환경은 이렇게 준비하세요

실습용 소스 파일은 저자의 깃허브 저장소에서 내려받을 수 있습니다. 이 책에서는 윈도우 환경에 비주얼 스튜디오를 설치해서 실습을 진행합니다. 자세한 내용은 「01-2」절에서 설명합니다. 다른 운영체제를 사용하거나 통합 개발 환경을 설치하기 어려운 상황일 때에 웹 브라우저에서 실행하는 방법도 소개합니다.

- **실습용 소스 파일 내려받기:** github.com/mystous/DoItCPP

궁금한 내용은 저자에게 질문해 보세요

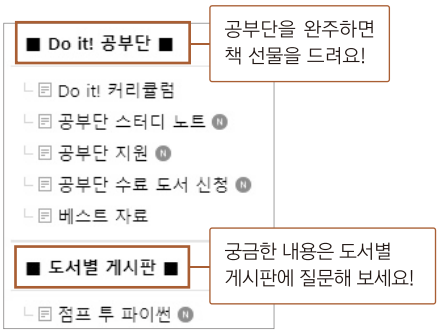
책을 읽다가 궁금한 내용이 있으면 저자의 깃허브 저장소에 질문해 보세요. 몇 쪽에서 어떤 점이 궁금한지 자세히 적어야 정확하고 빠른 답변을 받을 수 있습니다.

- **깃허브 저장소:** github.com/mystous/DoItCPP/issues

이지스 플랫폼 — 연결하면 더 큰 가치를 만들 수 있어요

‘Do it! 스터디룸’ 카페에서 친구들과 함께 공부!

cafe.naver.com/doitstudyroom



독자 설문 참여하면 6가지 혜택!

의견도 보내고 선물도 받고!

1. 추첨을 통해 소정의 선물 증정
2. 이 책의 업데이트 정보 및 개정 안내
3. 저자가 보내는 새로운 소식
4. 출간될 도서의 베타테스트 참여 기회
5. 출판사 이벤트 소식
6. 이지스 소식지 구독 기회

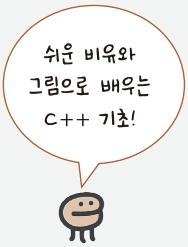
객체지향 프로그래밍 정복을 위한 쾌속 완성 코스!

16차시로 계획을 세우고 학습해 보세요.
독학이나 한 학기 강의용 교재로도 좋습니다.

16차시 완성!

차시	구분	장	주제	완료 날짜
1차시	첫째 마당	01 C++ 프로그래밍 시작하기	C++ 언어의 특징과 생태계, 개발 환경 구축	/
2차시		02 변수와 연산자	C++ 표준 입출력, 데이터 형식, 형식 변환, 키워드와 리터럴, 표현식과 연산식	/
3차시		03 포인터와 메모리 구조	포인터와 메모리, 함수와 구조체, 정적 변수와 상수 변수, 레퍼런스 변수	/
4차시		04 실행 흐름 제어	조건문, 반복문	/
5차시		05 예외 처리하기	예외 처리 구문, 생략과 실패 대응	/
6차시	둘째 마당	06 객체지향과 클래스	프로그래밍 패러다임, 객체지향 프로그래밍, 클래스와 인스턴스	/
7차시		07 객체지향 프로그래밍 특징	추상화, 캡슐화, 상속성, 다형성, this 포인터, 함수와 연산자 오버로딩, 접근 지정자와 프렌드	/
8차시		08 객체지향을 돕는 기능들	컴포지션과 어그리게이션, 가상 함수와 동적 바인딩, 추상 클래스와 정적 클래스 멤버	/
9차시		09 객체지향 설계 원칙	SOLID 원칙	/
10차시		10 템플릿	함수 템플릿과 클래스 템플릿	/
11차시	셋째 마당	11 C++ 표준 라이브러리	문자열 라이브러리, 파일 시스템, 기타 유용한 함수	/
12차시		12 STL의 컨테이너와 알고리즘	컨테이너와 반복자, 알고리즘	/
13차시	넷째 마당	13 모던 C++에 추가된 기능	C++ 버전별 주요 특징	/
14차시		14 새로운 데이터 형식과 라이브러리	형식 연역, 열거형, 수학 상수, 널 포인터, 2진수 표현, constexpr 지정자, function 객체, 스마트 포인터	/
15차시		15 새로운 구문 1	튜플과 구조적 바인딩, 범위 기반 for 문, 제어문의 초기화 구문, 람다 표현식	/
16차시		16 새로운 구문 2	폴드 표현식, 3방향 비교 연산자, using 키워드, 함수 키워드(default, delete, override, final)	/

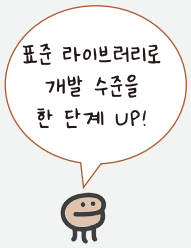
첫째마당
C++ 프로그래밍 기초



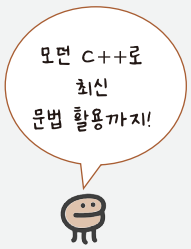
둘째마당

객체지향 프로그래밍

셋째마당 라이브러리 활용



넷째마당 모던 C++ 프로그래밍



첫째마당

C++ 프로그래밍 기초

첫째마당에서는 C++ 언어의 기본 사용법을 공부합니다. 먼저 개발 환경을 준비하고 프로그래밍에서 기본인 변수와 연산자를 배웁니다. 이어서 C++ 언어의 장점인 포인터와 메모리 구조에 관해 알아봅니다. 그리고 실행 흐름을 제어하고 예외를 처리하는 방법도 살펴보겠습니다.

01 C++ 프로그래밍 시작하기

C++는 1983년 발표된 이후 지금까지 많은 개발자가 사용하는 프로그래밍 언어로 발전했습니다. 첫 장에서는 C++가 발전한 과정과 언어의 특징을 보면서 C++가 오랫동안 사용될 수 있었던 비결을 생각해 보겠습니다. 그리고 C++ 프로그래밍을 실습할 개발 도구도 설치하겠습니다.

01-1 C++ 언어 알아보기

학습 목표
- C++ 언어의 발전 과정 훑어보기
- C++ 언어의 특징 알아보기

C++ 언어를 간단히 소개할 때는 'C 언어에 객체와 관련된 기능을 추가해 발전시킨 언어'라고 이야기합니다. 그래서 'C++'*라는 이름에는 C를 기초로 발전한 언어라는 의미로 C의 증가 연산자 '++'가 붙었습니다.

* 보통 '씨 플러스 플러스'로 읽습니다.

C++ 발전 과정

C++ 언어의 역사는 AT&T 벨 연구소의 비야네 스트롭스트룹Bjarne Stroustrup이 1979년에 착수한 'C with Classes'라는 연구에서 출발했습니다. 당시 스트롭스트룹은 Simula라는 시뮬레이션 언어와 유사하면서 성능이 좋은 언어를 구상했습니다. 그리고 Cpre라는 전처리기preprocessor를 활용해서 C with Classes라고 이름을 붙인 클래스를 포함한 C 언어를 시뮬레이션했습니다.

C with Classes는 프로그램 개발 언어로서는 많이 부족했지만, 이때 다음과 같은 C++ 언어 규약specification의 초안이 완성되었습니다.

- 클래스
- 멤버 함수와 변수 접근 제어(public, private)
- 프렌드(friend) 클래스
- 인라인 함수
- 상속 클래스
- 생성자와 소멸자
- 형식 검사와 함수 인자 변환
- 기본 인잣값, 연산자 오버로딩

초기 언어 규약에서 가장 큰 특징은 객체지향의 핵심인 **클래스**class를 도입했다는 점입니다. C 언어에 없던 클래스가 도입되면서 클래스를 상속해서 사용하는 방법, 멤버 변수와 함수로 불필요한 자료 노출을 줄이는 방법 등이 고안되었습니다.

C++ 언어의 탄생

스트롭스트룹은 C++ 언어 규약을 더 정교하게 다듬었습니다. 그러면서 C++는 C 언어의 확장이 아닌 하나의 언어로서 제모습을 갖추었습니다. 그리고 그 내용을 《The C++ Programming Language》(1985)라는 제목으로 출판했습니다. 이때부터 비로소 'C++'라고 하기 시작했습니다.

이때 추가된 언어 규약에서 주목할 만한 특징은 메모리 관리가 C 언어보다 수월해졌다는 점입니다. C 언어는 메모리 크기, 위치 등을 개발자가 직접 관리해야 했습니다. 하지만 C++는 자동으로 메모리를 할당하거나 해제할 수 있는 `new`, `delete` 키워드를 지원했습니다. 또한 레퍼런스라는 개념을 통해 포인터를 더 편리하게 다룰 수 있게 되었으며, 가상 함수와 연산자 오버로딩을 지원하기 시작하면서 객체지향 언어의 특징인 다형성을 완전하게 지원할 수 있게 되었습니다.

C++ 표준 제정

1989년에는 Cfront 2.0을 개발하면서 C++ 언어 규약도 함께 개편되었습니다. 이때는 컴파일러의 규약을 만드는 데 많은 노력을 기울였습니다. Cfront 2.0 배포 이후에는 사용자가 급증하고 다양한 C++ 컴파일러가 등장했습니다.*

* Cfront는 컴파일러가 아니라 전처리기입니다.

그런데 컴파일러마다 문법이 다르게 해석되는 문제가 있었습니다. 이에 따라 문법이나 패턴을 통일하고자 노력했고, 1998년에 비로소 C++의 첫 번째 국제 표준 언어 규격인 'ISO/IEC 14882:1998 Programming languages — C++(www.iso.org/standard/25845.html)'이 제정되었습니다. 이를 줄여서 'C++98'이라고 합니다.

C++98은 처음으로 제정된 국제 표준이라는 점에서 큰 의미가 있습니다. 템플릿template을 활용한 메타 프로그래밍meta-programming과 이를 활용한 STLstandard template library의 도입이 가장 큰 특징입니다.

C with Classes부터 C++98까지 발전 과정을 간단하게 알아봤습니다. C++ 언어는 오랜 시간 동안 만들어진 만큼 지금까지도 다양한 분야에서 활발히 사용되고 있습니다.

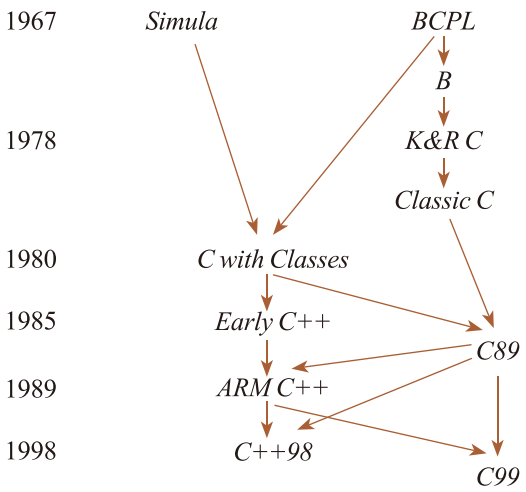


그림 1-1 C++98까지 발전과 C 언어 표준과의 관계
(출처: <Evolving a language in and for the real world: C++ 1991-2006>, https://dl.acm.org/doi/10.1145/1238844.1238848)

C++ 프로그램 빌드 과정

컴파일compile이란 C++ 소스 코드를 컴퓨터가 이해할 수 있는 코드로 변경하는 과정을 말합니다. 컴퓨터의 프로세서는 사람이 작성한 코드를 당장 해석할 수 없으므로 프로세서가 이해할 수 있는 오브젝트 코드object code로 변경해야 하는데, 이 과정을 컴파일이라고 합니다.

그런데 때로는 프로그램의 소스 파일이 여러 개일 수 있습니다. 이때 각각의 소스 파일을 컴파일하여 만든 오브젝트 파일들을 하나의 실행 파일로 묶는 과정을 **링크**link라고 합니다. 정리하면 컴파일과 링크를 거쳐 비로소 프로세서가 실행할 수 있는 파일이 만들어집니다.

C++는 컴파일하기 전에 전처리 과정을 거칩니다. 전처리pre-processing는 소스 파일이 컴파일되기 전에 소스 코드를 변경하거나 확장하는 등의 작업을 의미합니다. 전처리는 컴파일러가 전처리기에 지시하는 형태로 수행됩니다. 전처리기는 `#include`, `#if`, `#define`처럼 # 기호로 시작하는 지시문을 해석하고 그에 따라 소스 코드를 변경하여 컴파일러에 전달합니다.

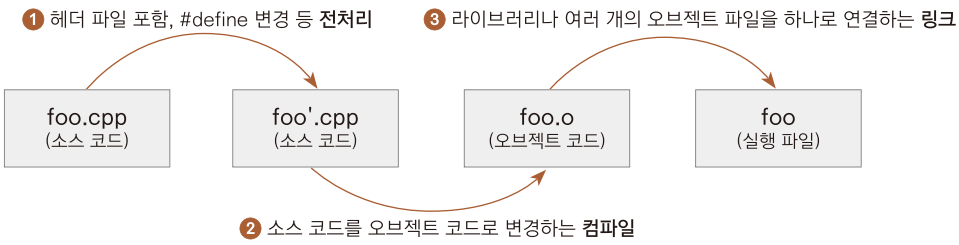
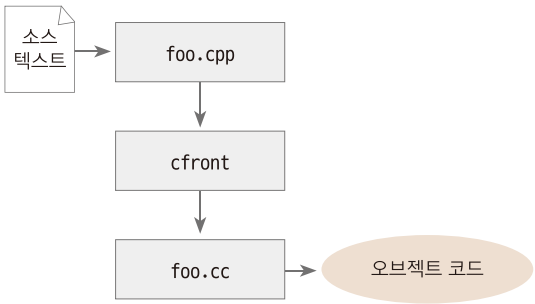


그림 1-2 C++ 프로그램 빌드 과정

요약하자면 C++로 작성한 코드는 #include와 같은 지시문을 전처리하고 이렇게 준비된 소스 코드를 오브젝트 코드로 컴파일합니다. 그리고 소스에 포함한 각종 라이브러리와 오브젝트 코드를 연결하는 링크를 거쳐 최종 실행 파일을 만듭니다. 이 과정을 통틀어 **빌드**build라고 합니다.

초기 C++는 전용 컴파일러가 없었다!

C++는 초기에 전용 컴파일러가 없었습니다. C++는 Cfront라는 전처리기로 C++ 소스 파일을 C 소스 파일로 변경한 다음, C 컴파일러로 컴파일했습니다. 하지만 1987년 GNU에서 G++ 컴파일러를 출시하여 전용 컴파일러가 생겼습니다.



초기 C++ 컴파일 과정(출처: <A history of C++: 1979-1991>, https://www.stroustrup.com/hopl2.pdf)

C++ 언어의 특징

프로그래밍 언어는 저마다 고유한 특징이 있습니다. 같은 객체지향 언어라도 설계 철학이 다르고 사용처가 다릅니다. 여기서는 C++ 언어의 성공 요인을 바탕으로 C++의 특징을 알아보겠습니다.

C++ 언어의 성공 요인

C++가 다방면에서 활용될 수 있었던 것은 언어가 가진 개발 철학 때문입니다. 스트롭스트룹은 자신의 논문 〈Evolving a language in and for the real world: C++ 1991-2006〉에서 C++가 성공할 수 있었던 언어적인 특징을 다음처럼 이야기하고 있습니다.

낮은 수준 액세스와 추상화 C++는 C 언어처럼 시스템에 직접 접근할 수 있고, Simula처럼 데이터를 추상화하여 접근할 수 있도록 했습니다.

유용한 도구 C++는 범용 언어로 애플리케이션 개발은 물론, 시스템에 접근하여 하드웨어를 직접 다룰 수도 있습니다.

시점 C++는 객체지향 프로그래밍을 지원하는 첫 번째 언어는 아니었지만, 언어 특유의 범용성 덕분에 출시부터 실제 문제를 해결하는 유용한 도구로 사용되었습니다.

비독점 AT&T 벨 연구소는 C++ 개발 이후 소유권을 독점하지 않았습니다. C++가 외부에서 개발되는 것을 장려하고 1989년 이후에는 모든 권리를 표준 기구로 이양했습니다.

안정성 초기 배포부터 C 언어와 호환성, 안정성을 확보했으며 이후에도 높은 호환성과 안정성을 유지하기 위해 표준화 과정을 충실하게 수행했습니다.

발전 예외 처리, 템플릿, STL 같은 새로운 기능이 C++ 전반에 걸쳐 계속 추가되었습니다.

C++는 끊임없이 발전하고 있으며 지금도 새로운 버전을 위한 표준화 위원회가 지속해서 열리고 있습니다. 초기부터 확보한 높은 성능과 안정성을 유지하면서 끊임없이 발전한 덕분에 C++가 계속 사용될 수 있었습니다.

객체지향 프로그래밍

C++는 객체지향 프로그래밍 언어입니다. 프로그래밍 세계에서 객체object는 '처리할 데이터와 처리 방법을 객체라는 논리 단위로 모두 포함하는 것'으로 간단하게 설명할 수 있습니다. 예를 들어 다음 그림에서 왼쪽은 게임 속 몬스터의 속성(데이터)과 동작(함수)을 추상화하여 나타낸 것입니다. 그리고 오른쪽은 이를 바탕으로 `monster`라는 객체를 UML 클래스 다이어그램으로 표현한 것입니다.

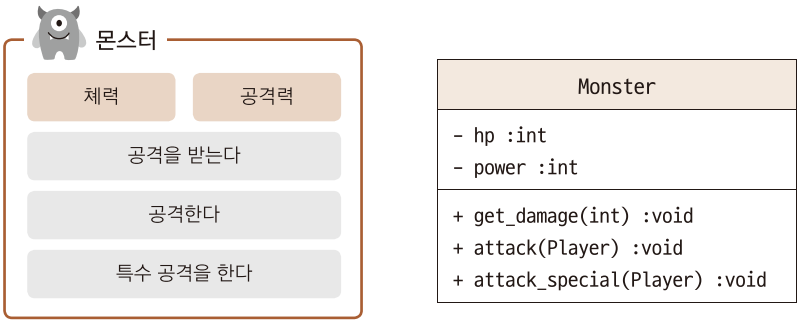


그림 1-3 게임 속 몬스터 객체 도식화

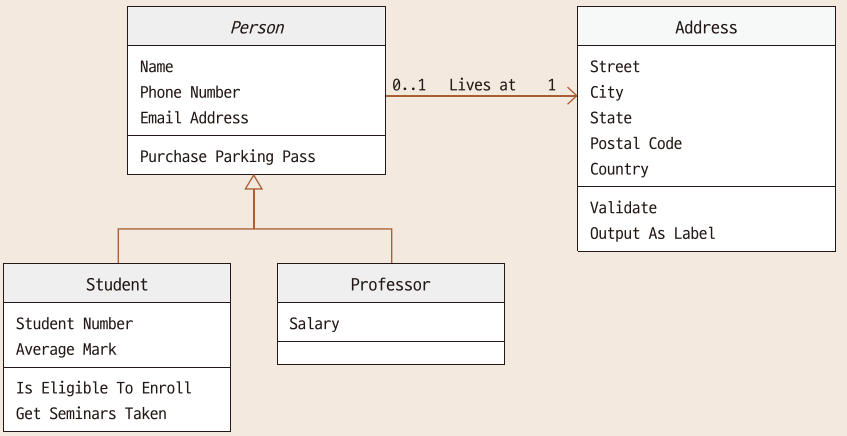
객체지향 언어인 C++는 다양한 실제 문제를 프로그램으로 모델링하여 해결하는 데 유용했습니다. C++ 이전에도 스몰토크smalltalk와 같은 객체지향 언어가 있었지만, 더 높은 범용성과 안정성을 갖춘 C++ 언어가 더 많은 문제를 모델링하고 해결할 수 있게 되면서 인기를 얻었습니다.

궁금해요!

UML이 뭔가요?

UML이란 통합 모델링 언어(unified modeling language)로, 1997년 OMG(object management group)에서 표준으로 채택한 언어입니다. 모델링이란 어떤 것을 만들기 전에 설계도를 만드는 작업을 말하는 용어입니다. 즉, UML은 프로그램 설계 구조와 의도를 쉽게 전달하기 위한 언어입니다. 다만 UML은 그림으로 표현되는 언어라서 '그래픽 언어'라고도 합니다.

UML을 그리는 다양한 도구가 있지만, 온라인에서 무료로 이용할 수 있는 app.diagrams.net이 있습니다. 다음은 app.diagrams.net에서 Person이라는 클래스 다이어그램을 UML로 그려 본 것입니다. 클래스 다이어그램은 클래스 내부의 특성이나 클래스 사이의 관계를 UML로 표현한 것입니다. 객체지향 언어를 배울 때는 클래스 다이어그램에 익숙해져야 합니다.



클래스 다이어그램 예시

저수준 접근과 메모리 직접 관리

C++가 가진 또 다른 특징은 저수준 접근이 가능하다는 것입니다. 저수준 접근이란, 프로그램이 동작하는 시스템의 하드웨어에 직접 접근하여 제어하는 것을 말합니다. 또한 메모리도 개발자가 프로그래밍으로 직접 관리할 수 있습니다. 메모리에 직접 접근하면 원하는 데이터를

즉시 처리할 수 있습니다. 저수준 접근과 메모리 직접 관리는 불필요한 과정을 줄여 매우 효율적인 프로그램을 개발할 수 있게 합니다.

데이터 추상화와 범용성

프로그램으로 다양한 문제를 해결한다는 것은 다양한 형태의 데이터와 구조를 다뤄야 한다는 의미입니다. 이러한 방법에는 여러 가지가 있지만, 다음처럼 두 가지를 생각해 볼 수 있습니다.

- 다양한 데이터와 구조를 처리할 수 있는 모든 방법을 제시한다.
- 한 가지 방법으로 여러 가지 데이터와 구조를 처리한다.

첫 번째 방법은 데이터나 구조별로 접근 방법을 일일이 알아야 하고 새로운 형태의 데이터나 구조가 추가되면 프로그래밍 언어 자체를 변경해야 합니다. 반면에 두 번째 방법은 데이터나 구조와 상관없이 한 가지 방법만 알면 되므로 사용이 쉽습니다. 다만, 컴파일러 구조가 복잡할 수 있습니다.

C++ 언어는 두 번째 방식을 채택해 다양한 데이터와 구조를 한 가지 방식으로 쉽게 처리할 수 있게 했습니다. 이를 범용성이라고 하며 C++ 언어는 객체를 추상화하는 방법으로 범용성을 구현합니다. 추상화에 관해서는 06장과 07장에서 자세히 알아보겠습니다.

C++ 활용 분야

C++ 언어로 만들 수 있는 프로그램이나 활용 분야는 독자 여러분의 학습 목표와 유사할 것입니다. 대략 다음처럼 정리할 수 있습니다.

- **객체지향 프로그래밍 학습:** C++는 고수준 언어이면서 객체지향의 다양한 특징을 학습할 수 있으며, 디자인 패턴, 클린 코드 등 객체지향 언어의 특징을 활용한 프로그래밍 기법을 구현하기도 쉽습니다.
- **네이티브 애플리케이션 개발:** C++ 컴파일러는 플랫폼에 최적화된 기계어 코드를 만들어 내므로 엑셀, 포토샵 같은 애플리케이션을 개발할 수 있습니다.
- **과학 기술 계산:** C++는 계산과 통신 성능을 특정 문제에 맞게 최적화할 수 있습니다. 또한 과학 기술 계산에 필요한 컴파일러, 분산·병렬, 수치 해석 라이브러리 등을 비롯해 풍부한 개발 생태계가 구축되어 있습니다.
- **임베디드 프로그래밍:** C++는 메모리와 하드웨어를 직접 다룰 수 있어 메모리 사용이 제한적인 임베디드 프로그래밍에 적합합니다.

- **다중 플랫폼 지원 프로그래밍:** C++는 다중 플랫폼에 호환되는 문법과 표준 라이브러리를 제공하므로 플랫폼별로 최적화된 기계어 코드를 만들 수 있습니다.

이처럼 C++는 고수준 언어이면서 저수준으로 접근할 수도 있어서 고성능 프로그램을 쉽게 개발할 수 있습니다. 하지만 저수준 접근과 메모리 직접 관리는 프로그래밍을 어렵게 만드는 원인이기도 합니다. 성능과 개발 편의성은 여러 면에서 상충하므로 무조건 어떤 언어를 사용해야 한다고 잘라서 말할 수는 없습니다.

C++ 개발 생태계

개발 생태계는 프로그래밍 언어가 발전할 수 있는 밑바탕이라고 할 수 있습니다. 여기서는 언어의 '표준 명세', '개발자 커뮤니티', '개발 환경과 라이브러리'로 분류하여 C++의 개발 생태계를 알아보겠습니다.

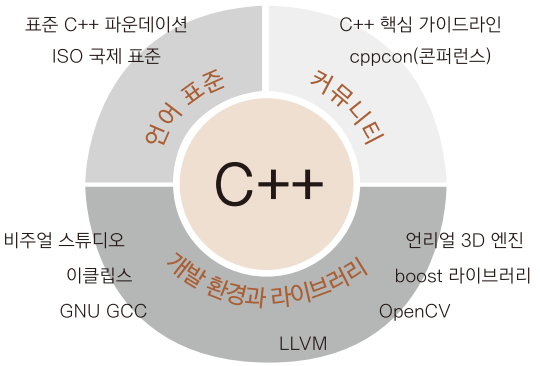


그림 1-4 C++ 개발 생태계

표준 명세

C++는 하나의 회사나 단체에 종속되지 않는 열린 생태계를 만들고자 국제 표준으로 재정되었습니다. 따라서 다양한 회사나 단체, 개발자들이 컴파일러를 만들어서 배포할 수 있습니다.

C++ 규약은 1998년 ISO 표준으로 채택된 이후 새로운 규약이 지속해서 추가되고 있습니다. 2024년 현재 가장 최근 표준은 C++23입니다. C++ 언어는 C++11부터 3년마다 새로운 표준안을 내놓으며 새로운 요구 사항과 시대 흐름을 반영하고 있습니다.

1998년 처음 ISO 표준이 제정된 이후 2003년, 2011년, 2014년, 2017년, 2020년, 2023년까지 총 여섯 차례 표준안이 개정되었습니다. 각 표준은 연도의 끝 두 자리를 사용해 C++03, C++11, C++14, C++17, C++20, C++23이라고 합니다. C++98~23까지 주요 변경 사항은 다음 그림에서 확인할 수 있습니다.

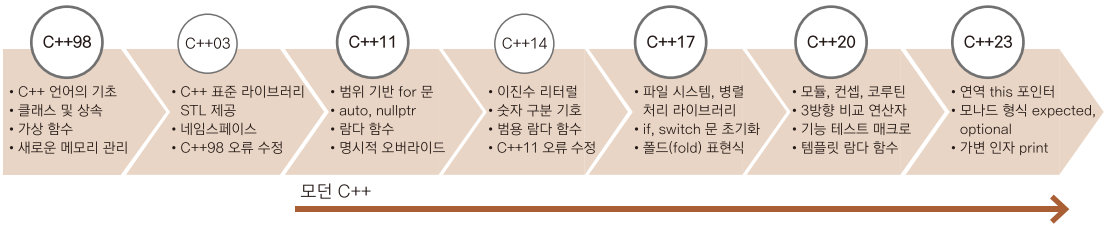


그림 1-5 C++ 버전별 주요 특징

C++11부터를 흔히 '모던 C++'이라고 하며 이 책의 넷째마당(13~16장)에서는 모던 C++에서 새로 추가된 데이터 형식, 구문, 표준 라이브러리 가운데 주요 내용을 소개합니다.

커뮤니티

프로그래밍 언어가 발전하고 유지되려면 표준 활동 외에도 대중에게 알리고 사용자끼리 유대를 장려하는 커뮤니티 활동 역시 필요합니다.

C++ 언어를 주제로 한 다양한 커뮤니티가 있지만, 가장 중심이 되는 활동은 CppCon입니다. CppCon은 ISO 표준을 이끄는 비영리 단체인 표준 C++ 파운데이션에서 주관하는 C++ 개발자 콘퍼런스입니다. CppCon에서는 매년 다양한 주제의 C++ 관련 주제가 발표되고 있습니다. C++ 언어 규약에 대한 발표가 많으므로 시간이 지나도 참고할 수 있습니다.

그리고 주목할 만한 커뮤니티로는 비야네 스트롭스트룹과 다른 개발자들이 함께 주관하는 'C++ 핵심 가이드라인'이 있습니다. 새로운 C++ 언어 표준과 개발에 참고할 수 있는 다양한 내용을 제공하고 있습니다. 그리고 'C++ 코리아'에서는 국내 C++ 개발자들과 만날 수 있습니다.

- **CppCon**: cppcon.org
- **Cpp 핵심 가이드라인**: github.com/isocpp/CppCoreGuidelines
- **C++ 코리아**: github.com/CppKorea

그림 1-6 CppCon 홈페이지

개발 환경과 라이브러리

개발 생태계 중에서 개발자들이 가장 빈번하게 접하는 것이 개발 환경과 라이브러리입니다. 개발 환경에는 소스 코드를 작성하는 데 도움을 주는 통합 개발 환경과 컴파일러, 디버거 등이 있습니다.

통합 개발 환경

C++ 통합 개발 환경integrated development environment, IDE으로는 윈도우에서 사용할 수 있는 비주얼 스튜디오Visual Studio와 macOS에서 사용할 수 있는 엑스코드Xcode, 다양한 운영체제에서 사용할 수 있는 비주얼 스튜디오 코드Visual Studio Code, 이클립스Eclipse 등이 있습니다.

- **비주얼 스튜디오**: visualstudio.microsoft.com
- **비주얼 스튜디오 코드**: code.visualstudio.com
- **엑스코드**: developer.apple.com/kr/xcode
- **이클립스**: eclipse.org

컴파일러와 디버거

C++ 언어는 국제 표준이므로 이를 준수한 다양한 컴파일러가 있습니다. 대부분의 IDE는 컴파일러를 함께 제공합니다. 디버거 역시 IDE에 통합되어 있습니다. 전 세계에서 많이 사용하는 컴파일러와 디버거는 다음과 같습니다.

- **GCC와 GDB**: gcc.gnu.org
- **Clang과 LLDB**: clang.llvm.org
- **DPC++/C++**: intel.com/content/www/us/en/developer/tools/oneapi/dpc-compiler.html

GCC[gnu compiler collection]는 GNU의 컴파일러 모음 프로젝트입니다. 프로젝트 중에서 C++ 언어를 위한 컴파일러는 G++이며, 디버거는 GNU의 다른 프로젝트인 GDB 디버거를 사용할 수 있습니다. GCC로 개발하면 이식성이 높은 애플리케이션을 개발할 수 있습니다.

Clang은 LLVM[low level virtual machine]을 기반에 둔 C 계열 언어들의 컴파일러입니다. Clang은 C와 C++뿐 아니라 오브젝티브-C[Objective-C]와 같은 C 계열 프로그래밍 언어를 모두 컴파일할 수 있습니다. 비주얼 스튜디오 같은 IDE에 기본 컴파일러 대신 사용할 수 있습니다. Clang으로 컴파일된 프로그램을 디버깅하려면 LLVM에서 제공하는 LLDB[low-level debugger]를 사용해야 합니다.

인텔은 자사의 CPU에 최적화되어 최고의 성능을 보장하는 C++ 컴파일러를 유료로 제공했습니다. 그리고 2020년 12월에 다양한 하드웨어 가속기와 플랫폼을 지원하는 oneAPI를 출시하면서 컴파일러를 포함해 다양한 소프트웨어를 oneAPI로 통합하여 무료로 배포하고 있습니다. oneAPI에서 C++ 컴파일러는 DPC++/C++입니다. 이 컴파일러는 인텔 CPU에서 병렬 처리에 최적화된 코드를 생성합니다. 인텔 컴파일러 역시 IDE와 통합하여 사용할 수 있습니다.

라이브러리

현대의 소프트웨어는 규모도 크고 복잡한 기술을 바탕으로 설계되어 전체를 혼자 만들 수는 없습니다. 라이브러리나 프레임워크처럼 이미 개발된 공통의 기능을 가져와 사용해야 합니다. 비야네 스트롭스트룹은 라이브러리에 관해 다음처럼 이야기합니다.

> *"좋은 라이브러리 없이는 흥미로운 작업들을 C++로 수행하기 어렵다.*
> *하지만 좋은 라이브러리를 사용하면 대부분의 작업을 손쉽게 할 수 있다."*

C++ 언어의 오랜 역사만큼이나 다양한 라이브러리가 존재합니다. 다만, 파이썬이나 리눅스 운영체제들처럼 배포 시스템을 갖추고 있지 않아서 개발자가 스스로 필요한 라이브러리를 설치해야 합니다. 오픈소스 C++ 라이브러리 목록은 다음 주소에서 확인할 수 있습니다.

- **오픈소스 C++ 라이브러리 목록**: ko.cppreference.com/w/cpp/links/libs

많이 사용하는 C++ 라이브러리를 소개하면 다음과 같습니다. 이 가운데에 C++ 표준 라이브러리는 이 책의 11장과 12장에서 다룹니다.

- **C++ 표준 라이브러리(en.cppreference.com/w/cpp/standard_library)**: C++ 표준 라이브러리는 언어 표준을 지원하는 클래스, 라이브러리의 집합입니다.
- **Boost C++(boost.org)**: 문자열, 포인터, 컨테이너, 알고리즘 등 C++ 프로그램에 필요한 라이브러리가 통합되어 있습니다. C++ 언어 표준에 많이 포함되었지만 구현체 등이 달라서 구별하여 사용해야 합니다.
- **OpenCV(opencv.org)**: 이론적인 원리를 알지 못하더라도 영상 처리에 필요한 다양한 기능을 사용할 수 있도록 만들어졌습니다. C++ 언어 외에도 파이썬, C# 등 다양한 언어를 지원합니다. 최근에는 딥러닝 학습에도 많이 사용됩니다.
- **POCO(pocoproject.org)**: C++ 언어에서 사용되는 라이브러리를 다양한 운영체제에서 사용할 수 있도록 지원합니다.
- **텐서플로(tensorflow.org)**: 딥러닝을 위한 다양한 라이브러리와 알고리즘을 제공합니다. 텐서플로 자체는 C++로 구현되어 있고, C++와 파이썬을 위한 API를 제공합니다.
- **언리얼 게임 엔진(unrealengine.com)**: 3D 게임을 개발할 때 필요한 그래픽 엔진, 물리 엔진, 음향과 관련된 다양한 기능을 제공합니다.

지금까지 C++ 개발을 쉽게 해주고 C++ 언어가 계속해서 사용될 수 있는 환경을 제공해 준 개발 생태계에 대해서 알아봤습니다. "구슬이 서말이라도 꿰어야 보배"라는 말이 있습니다. 개발 생태계는 C++ 언어로 프로그램을 개발할 때 풍부한 구슬과 같습니다. 잘 꿰맬 수 있는 방법은 다음 장부터 자세히 살펴보겠습니다.

3분 퀴즈

1 C++ 언어가 ISO 국제 표준으로 추진된 배경을 설명해 보세요.

2 C++ 언어의 특징을 설명해 보세요.

3 C++ 개발 생태계를 구성하는 세 가지 요소는 무엇일까요?
1) _____ 2) _____ 3) _____

[모범 답안]
1 컴파일러에 따라 다르게 해석되는 현상을 막고 문법과 해석 패턴을 통일하려고
2 C++는 객체지향 언어입니다. 다른 언어들과 다르게 저수준 접근을 통해 하드웨어에 직접 접근할 수 있고 메모리를 직접 관리합니다. 데이터 추상화를 통해서 다양한 데이터를 일관되게 처리할 수 있으며 범용성을 확보하였습니다.
3 1)언어 표준 2) 커뮤니티 3) 개발 환경 및 라이브러리

01-2 개발 환경 준비하기

학습 목표
- 비주얼 스튜디오 설치하기
- 첫 번째 C++ 프로그램 만들기

이제 컴퓨터에 C++ 개발 환경을 준비해 보겠습니다. C++ 프로그램은 윈도우와 macOS를 비롯해 다양한 운영체제에서 개발할 수 있지만, 개발 도구를 설치하고 설정하는 과정은 운영체제마다 차이가 있습니다. 이 책의 실습 코드는 모두 윈도우 환경에서 실습한다는 가정으로 작성했습니다. 따라서 윈도우에서 개발하는 방법을 위주로 살펴보고, macOS와 웹 환경에서 개발하는 방법도 간략하게 소개하겠습니다.

비주얼 스튜디오 설치하기

비주얼 스튜디오Visual Studio는 전 세계에서 많은 개발자가 사용하는 통합 개발 환경integrated development environment, IDE입니다. 비주얼 스튜디오를 이용하면 C++ 언어로 작성된 코드를 컴파일하고 디버깅할 수 있으며 개발을 돕는 여러 가지 편리한 기능도 이용할 수 있습니다.

비주얼 스튜디오를 설치하려면 웹 브라우저에서 비주얼 스튜디오 홈페이지에 접속한 후 설치 파일을 내려받습니다. 비주얼 스튜디오는 누구나 무료로 내려받아 사용할 수 있는 커뮤니티 버전과 더 많은 기능이 포함된 전문가 버전, 기업용 유료 버전이 있습니다.

- **비주얼 스튜디오**: visualstudio.microsoft.com/ko/downloads

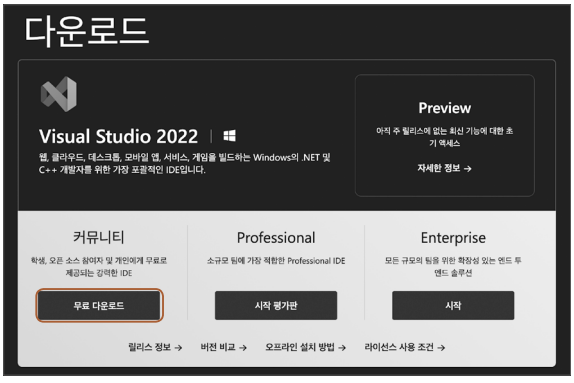


그림 1-7 비주얼 스튜디오 다운로드 화면

비주얼 스튜디오 설치 파일을 내려받았으면 실행해서 설치를 시작합니다.

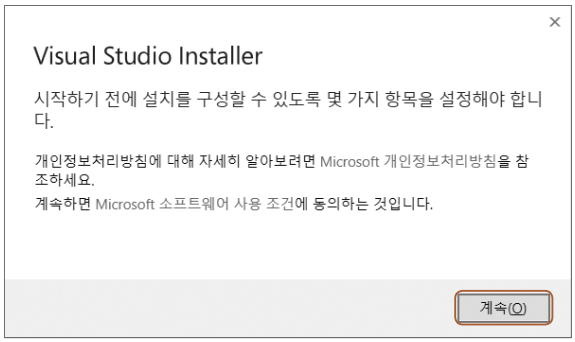


그림 1-8 비주얼 스튜디오 설치 시작

워크로드를 선택하는 화면이 나오면 **[C++를 사용한 데스크톱 개발]**을 선택하고, 오른쪽의 설치 세부 정보에서 **[v143 빌드 도구용 C++ 모듈(x64/x86 - 실험적)]**에 체크 표시합니다. 이는 C++20에서 추가된 모듈을 사용하기 위한 설정입니다.

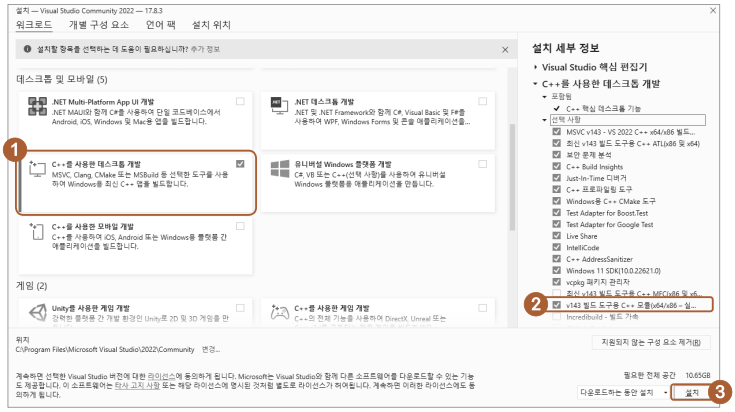


그림 1-9 워크로드 설정

설치를 마치면 〈시작〉을 클릭해 비주얼 스튜디오를 실행합니다.

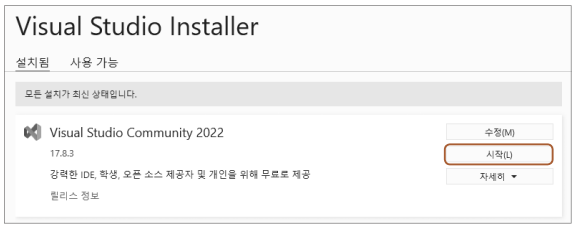


그림 1-10 설치 완료

비주얼 스튜디오 제품군은 마이크로소프트 계정으로 여러 컴퓨터나 버전의 설정을 공유할 수 있습니다. 마이크로소프트 계정에 로그인하는 화면이 나오면 로그인하고, 설정 공유를 원하지 않으면 계정 로그인을 건너뛰어도 됩니다.

그림 1-11 마이크로소프트 계정 로그인

첫 번째 프로그램 만들기

비주얼 스튜디오를 설치했으면 콘솔 화면에 'Hello World!'를 출력하는 간단한 프로그램을 만들어 보겠습니다. 비주얼 스튜디오를 실행하고 시작 화면에서 **[새 프로젝트 만들기]**를 선택합니다.

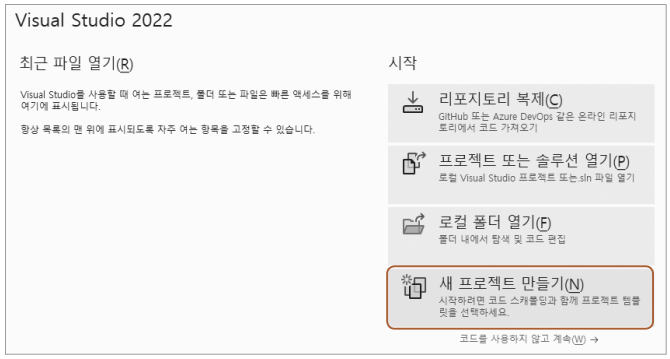


그림 1-12 새 프로젝트 만들기

그리고 프로젝트 템플릿 선택 화면에서 **[콘솔 앱]**을 선택합니다.

그림 1-13 템플릿 선택

프로젝트 이름, 위치, 솔루션 이름 등을 확인하고 〈**만들기**〉를 클릭합니다. 솔루션은 프로젝트보다 상위 개념입니다.

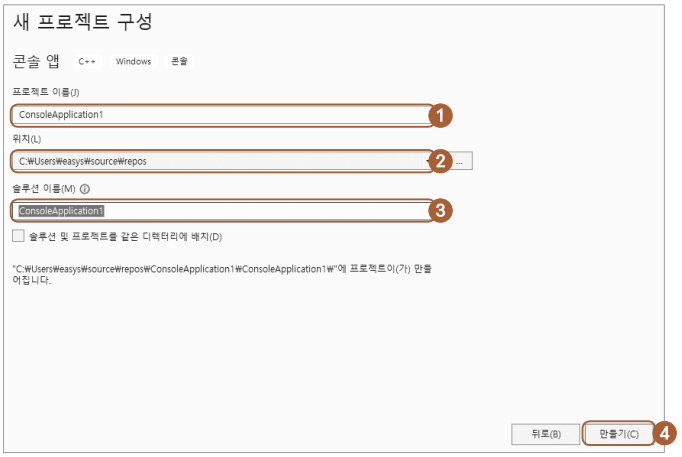


그림 1-14 새 프로젝트 구성

프로젝트가 만들어지면 다음 그림처럼 편집 창이 열립니다. 오른쪽에는 솔루션 탐색기가 보이고 왼쪽에는 기본 코드가 입력된 C++ 소스 파일이 열려 있습니다. 솔루션 탐색기에서 다른 파일을 더블클릭하면 해당 파일을 열 수 있습니다.

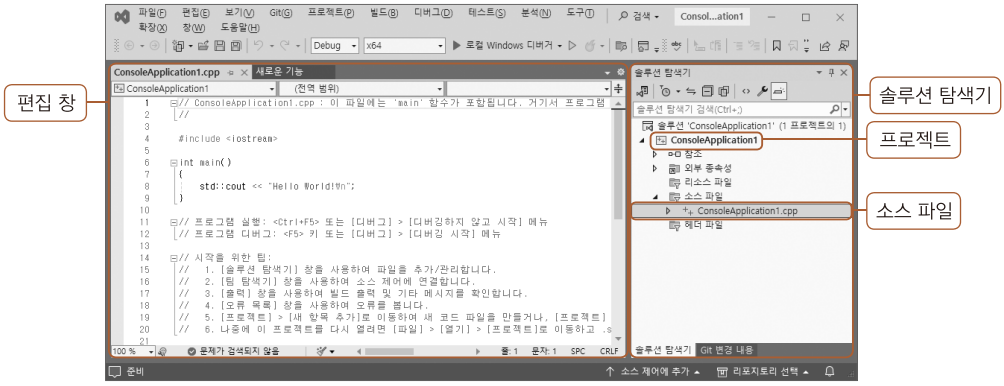


그림 1-15 소스 파일 편집 창

C++ 소스 파일은 확장자가 *.cpp입니다. 편집 창에 있는 모든 코드를 지우고 다음과 같은 코드를 직접 입력해 보세요. 이 코드는 자동으로 생성됐던 코드와 똑같이 동작합니다.

작성한 소스 파일을 실행하려면 키보드에서 단축 키 F5*를 누르거나 위쪽 도구 모음에서 녹색 버튼(▶ 로컬 Windows 디버거)을 클릭합니다. 그러면 아래쪽 상태 표시 줄에 빌드 진행 사항이 표시된 후 완료되면 자동으로 실행됩니다.*

* 환경에 따라서 디버거가 매우 느리게 실행되거나 멈출 수 있습니다. 오래 기다려도 동작하지 않거나 멈추었다면 Shift + F5로 디버그를 중지한 후 다시 실행하기 바랍니다.

Do it! 실습 ‘Hello World!’ 출력

```
#include <iostream>

int main()
{
  std::cout << "Hello World!\n";
}
```

실행 결과

```
Hello World!
```

* 원래 int main()처럼 반환값이 있는 함수는 return 문을 포함해야 하지만, main()은 특별한 함수이므로 생략해도 return 0을 반환합니다(참고: en.cppreference.com/w/cpp/language/main_function).

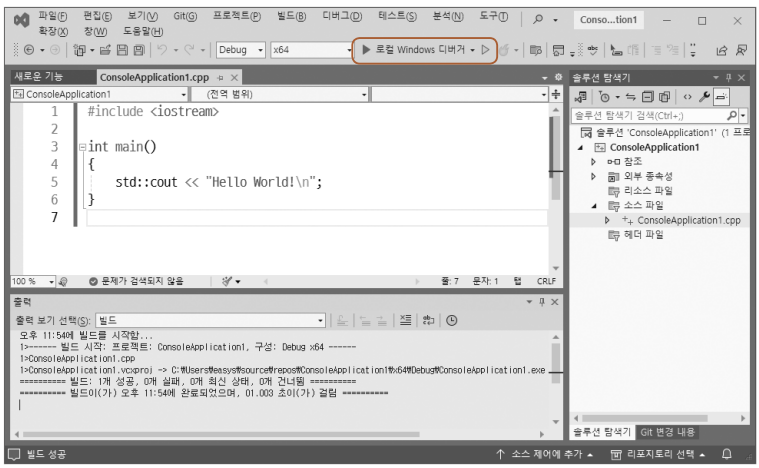


그림 1-16 빌드 완료 화면

프로그램이 실행되면 앞에서 프로젝트를 만들 때 [콘솔 앱]을 선택했으므로 콘솔 창이 열리면서 ‘Hello World!’가 출력되는 것을 볼 수 있습니다. 이 창은 아무 키나 누르면 닫힙니다.

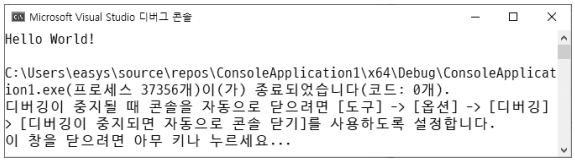


그림 1-17 콘솔 창에 'Hello World!' 출력 화면

궁금해요!

녹색 버튼 옆에 <디버그하지 않고 시작>은 무엇인가요?

녹색 버튼(▶ 로컬 Windows 디버거)은 <디버깅 시작> 메뉴이며, 그 옆의 빈 녹색 버튼(▷)은 <디버그하지 않고 시작> 메뉴입니다. 그리고 F5는 디버깅 시작 단축 키이며, 만약 디버그하지 않고 시작하려면 Ctrl 키를 누른 채로 F5를 누릅니다.

디버깅은 프로그램의 문제점을 찾기 위해 코드를 한 줄씩 실행하면서 변수 상태 등을 관찰하는 방법입니다. <디버그하지 않고 시작>은 디버깅 과정을 생략한 채 빌드된 프로그램을 직접 실행하는 방법입니다. 마치 윈도우 탐색기에서 실행 파일을 더블클릭한 것과 같습니다.

오류 메시지 확인하기

소스 파일을 컴파일하거나 실행할 때 오류가 발생하면 다음처럼 마지막으로 성공한 빌드를 실행할지 묻습니다. 이때 〈아니요〉를 클릭하면 아래쪽 [오류 목록] 창에 오류 메시지를 보여줍니다.

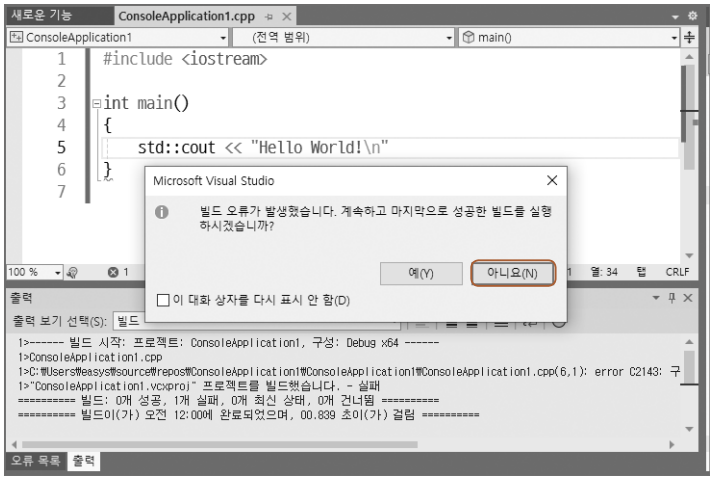


그림 1-18 빌드 오류 발생

다음 그림은 소스에서 5번째 줄 끝에 세미콜론을 생략하고 실행한 예입니다.

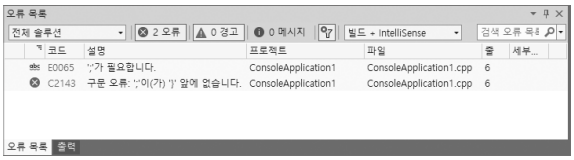


그림 1-19 오류 메시지 확인

지금까지 비주얼 스튜디오에서 프로젝트를 만들고 코드를 작성해 실행하는 방법을 살펴봤습니다. 이 책의 모든 실습 코드는 단일 cpp 파일로 구성했으므로 각 실습마다 이와 같은 방식으로 비주얼 스튜디오에 프로젝트를 만들고 소스 파일에 코드를 작성하여 실행하면 됩니다.

실습 파일 내려받기

프로그래밍 언어를 공부할 때는 코드를 키보드로 직접 작성해 보아야 빨리 익숙해집니다. 이 책의 실습 코드도 여러분이 직접 작성하여 실행해 보면 좋겠습니다. 다만 코드가 너무 길거나 내가 작성한 코드를 검증된 코드와 비교해 보고 싶다면 필자가 제공하는 소스 파일을 참고할 수 있습니다.

이 책의 실습용 소스 파일은 모두 필자의 깃허브 저장소에 올려 두었습니다. 다음 주소에 접속하면 소스 파일을 내려받아 실행해 볼 수 있습니다.

- **실습용 소스 파일 내려받기**: github.com/mystous/DoItCPP

깃허브 저장소에서 〈Code〉를 클릭하고 [Download ZIP]을 선택합니다. 그러면 이름이 'DoItCPP-master.zip'인 압축 파일을 내려받습니다.

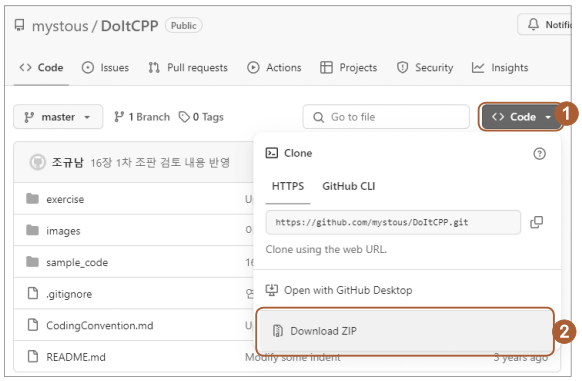


그림 1-20 깃허브에서 실습 파일 내려받기

내려받은 파일의 압축을 풀면 sample_code라는 폴더가 보입니다. 이 폴더에서 sample_code.sln 파일을 더블클릭하면 비주얼 스튜디오에서 이 책의 실습 파일이 모두 담긴 솔루션이 열립니다.

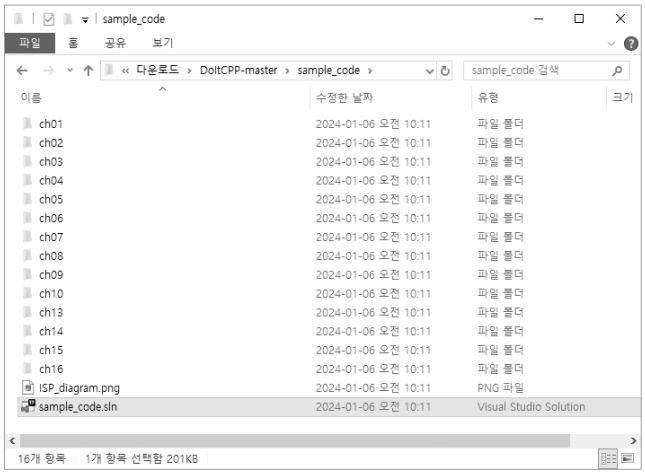


그림 1-21 솔루션 파일 실행

비주얼 스튜디오에서 솔루션 탐색기를 보면 장별 폴더가 있습니다. 폴더를 확장하면 해당 장에서 실습하는 프로젝트()가 나열되고, 프로젝트를 확장하면 *.cpp 확장자의 소스 파일()이 보입니다.

각 프로젝트의 소스 파일을 실행하려면 우선 해당 프로젝트를 시작 프로젝트로 설정해야 합니다. 프로젝트에 마우스 오른쪽을 클릭한 후 **[시작 프로젝트로 설정]**을 선택합니다. 시작 프로젝트로 설정되면 프로젝트 이름이 굵게 표시됩니다. 이를 확인하고 F5를 눌러 실행합니다.

* 만약 편집 창에 다른 프로젝트의 소스 파일이 열려 있더라도 시작 프로젝트로 설정된 소스 파일이 실행됩니다. 시작 프로젝트 설정을 잊지 마세요.

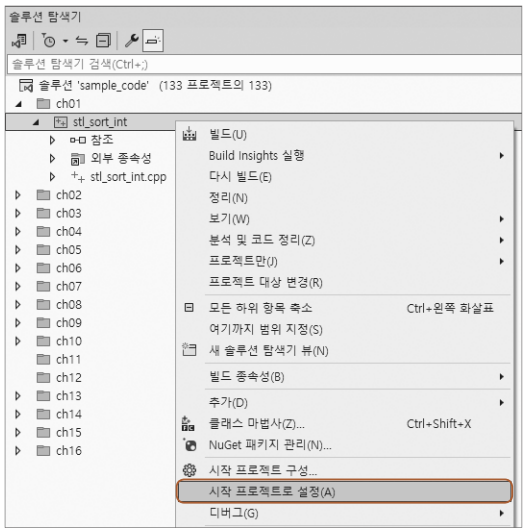


그림 1-22 시작 프로젝트로 설정하기

세상의 속도를
따라잡고 싶다면

Do it!

C++ 완전 정복

Do it!
C++ Complete Conquest

객체지향 프로그래밍 기초부터 실용적인 코드까지
'진짜' C++ 스타일로 코딩하는 방법을 배운다!

추천해요! "입문자도 쉽게 배울 수 있는 구성"

이 책은 C++ 기본 문법을 시작으로 최근에 추가된 새로운 문법과 필수 라이브러리를 입문자도 쉽게 배울 수 있도록 구성했습니다. 또한 여러 사람과 함께 개발할 때 발생할 수 있는 규칙 위반을 컴파일 단계에서 발견할 수 있는 몇 가지 방법도 소개합니다.

— **황규별** 님(LG유플러스 CDO 전무)

실용적인 실습과 실행 결과로 학습자가 문법을 이해하고 응용할 수 있도록 합니다. 객체지향 프로그래밍을 배우는 C++ 교재로 추천합니다.

— **유헌창** 님(고려대학교 정보대학 컴퓨터학과 교수)

추상적인 개념을 그림으로 쉽게 설명하고 실용적인 코드를 담았습니다. 1997년부터 C++를 다뤄 온 전문가가 코드를 정확하게 이해하고 작성할 수 있게 해주는 책입니다.

— **임지순** 님(3PM Inc. 대표, 리얼월드 암호학 등 다수 기술 번역)

객체지향의 원리와 SOLID 원칙, 다양한 디자인 패턴을 두루 담았으며, 실무에서 협업과 유지·보수가 쉬운 코드를 어떻게 작성하는지 엿볼 수 있습니다.

— **맹윤호** 님(이화여대 신산업융합대학 겸임교수, 카논그룹 CTO)

값 32,000원

9 791163 035695 93000

ISBN 979-11-6303-569-5