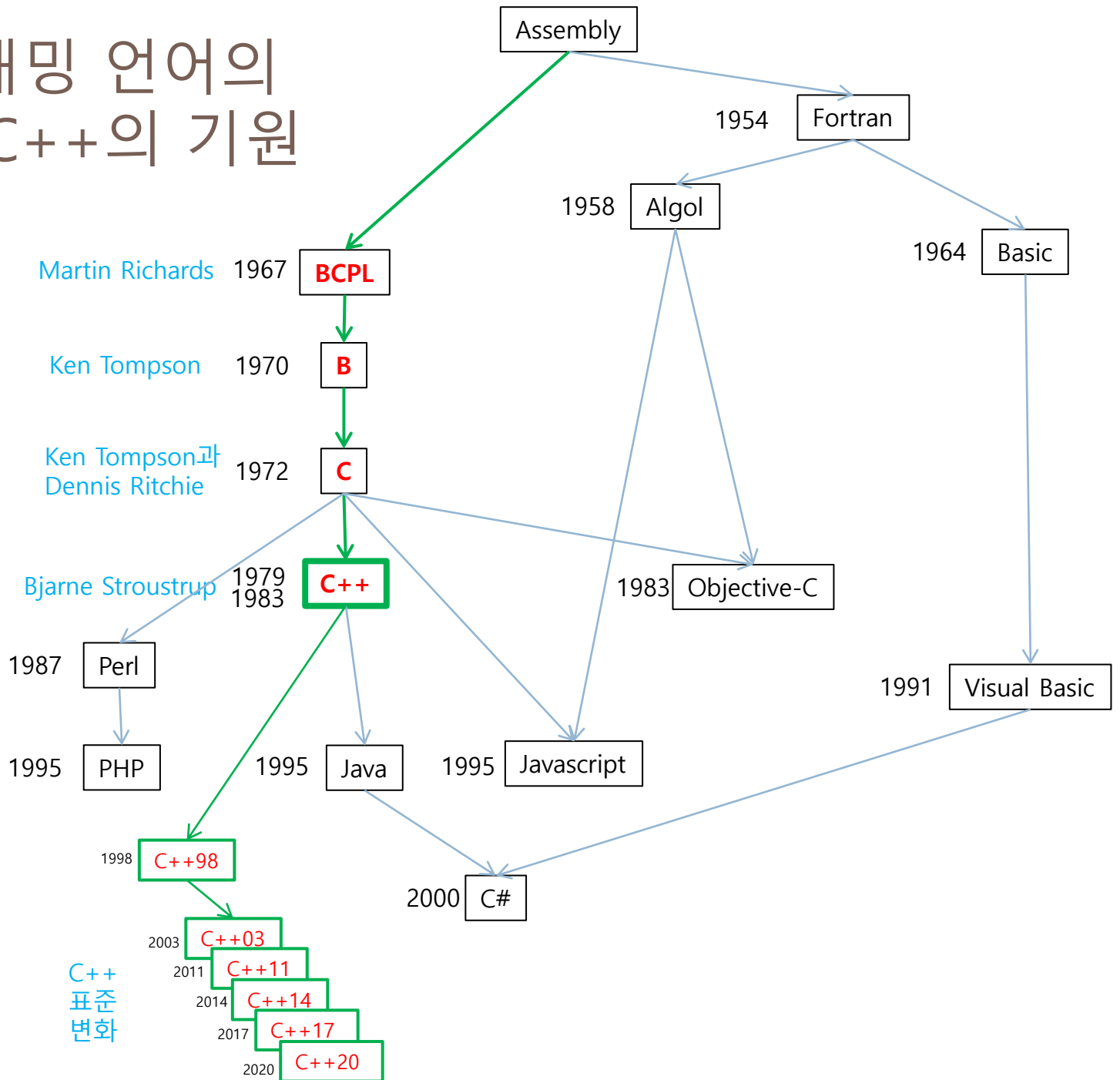


프로그래밍 언어의 진화와 C++의 기원



표준/비표준 C++ 프로그램의 비교

2

표준 C++ 규칙에 따라
작성된 C++ 프로그램

```
#include <iostream>
int main() {
    std::cout << "Hello";
    return 0;
}
```

모든 C++
컴파일러
에 의해
컴파일

볼랜드 C++
컴파일러

비주얼 C++
컴파일러

GNU C++
컴파일러

실행
파일

실행
파일

실행
파일

표준 C++ 규칙에 따라
작성되지 않는 비주얼 C++ 프로그램

```
#include <iostream>
int _cdecl main() {
    std::cout << "Hello";
    return 0;
}
```

비주얼
C++ 전용
키워드

볼랜드 C++
컴파일러

비주얼 C++
컴파일러

GNU C++
컴파일러

~~실행
파일~~

실행
파일

~~실행
파일~~

`_cdecl` : MS 전용으로 이전 버전과의 호환성을 위해 제공(참고)

C++ 언어의 주요한 설계 목적

3

- C 언어와의 호환성
 - ▣ C 언어의 문법 체계 계승
 - 소스 레벨 호환성 - 기존에 작성된 C 프로그램을 그대로 가져다 사용
 - 링크 레벨 호환성 - C 목적 파일과 라이브러리를 C++ 프로그램에서 링크
- 객체 지향 개념 도입
 - ▣ 캡슐화, 상속, 다형성
 - ▣ 소프트웨어의 재사용을 통해 생산성 향상
 - ▣ 복잡하고 큰 규모의 소프트웨어의 작성, 관리, 유지보수 용이
- 엄격한 타입 체크
 - ▣ 실행 시간 오류의 가능성을 줄임
 - ▣ 디버깅 편리
- 실행 시간의 효율성 저하 최소화
 - ▣ 실행 시간을 저하시키는 요소와 해결
 - 작은 크기의 멤버 함수 잦은 호출 가능성 -> 인라인 함수로 실행 시간 저하 해소

C 언어에 추가한 기능

4

- ▣ 함수 중복(function overloading)
 - 매개 변수의 개수나 타입이 다른 동일한 이름의 함수들 선언
- ▣ 디폴트 매개 변수(default parameter)
 - 매개 변수에 디폴트 값이 전달되도록 함수 선언
- ▣ 참조와 참조 변수(reference)
 - 하나의 변수에 별명을 사용하는 참조 변수 도입
- ▣ 참조에 의한 호출(call-by-reference)
 - 함수 호출 시 참조 전달
- ▣ new/delete 연산자
 - 동적 메모리 할당/해제를 위해 new와 delete 연산자 도입
- ▣ 연산자 재정의
 - 기존 C++ 연산자에 새로운 연산 정의
- ▣ 제네릭 함수와 클래스
 - 데이터 타입에 의존하지 않고 일반화시킨 함수나 클래스 작성 가능

C++ 언어에서 객체 지향을 도입한 목적

5

- 소프트웨어 생산성 향상
 - ▣ 소프트웨어의 생명 주기 단축 문제 해결 필요
 - ▣ 기 작성된 코드의 재사용 필요
 - ▣ C++ 클래스 상속 및 객체 재사용으로 해결

- 실세계에 대한 쉬운 모델링
 - ▣ 과거의 소프트웨어
 - 수학 계산이나 통계 처리에 편리한 절차 지향 언어가 적합
 - ▣ 현대의 소프트웨어
 - 물체 혹은 객체의 상호 작용에 대한 묘사가 필요
 - 실세계는 객체로 구성된 세계
 - 객체를 중심으로 하는 객체 지향 언어 적합

C++ 프로그램 개발 과정

6



C++ 소스 프로그램 작성

```
#include <iostream>
int main() {
    std::cout << "Hello";
    return 0;
}
```

소스 파일
(hello.cpp)

컴파일

```
_main,12#
$<<01010
00000111
_Hello001
```

목적 파일
(hello.obj)

C++ 라이브러리

cout

<<

.....

링킹

```
01010000
01000101
01001111
01011010
10100101
11010101
```

실행 파일
(hello.exe)

실행

오류 발생

디버깅

오류 수정



C++ 프로그램 작성 및 컴파일

7

□ 편집

- ▣ C++ 소스 프로그램은 텍스트 파일
 - 아무 텍스트 편집기로 편집 가능
- ▣ C++ 소스 프로그램의 표준 확장자는 .cpp
- ▣ C++ 통합 개발 소프트웨어 이용 추천
 - C++ 소스 편집, 컴파일, 링킹, 실행, 디버깅 등 모든 단계 통합 지원
 - 대표적인 소프트웨어 - Visual Studio

□ 컴파일

- ▣ C++ 소스 프로그램을 기계어를 가진 목적 파일로 변환
 - cpp 파일을 obj 파일로 변환

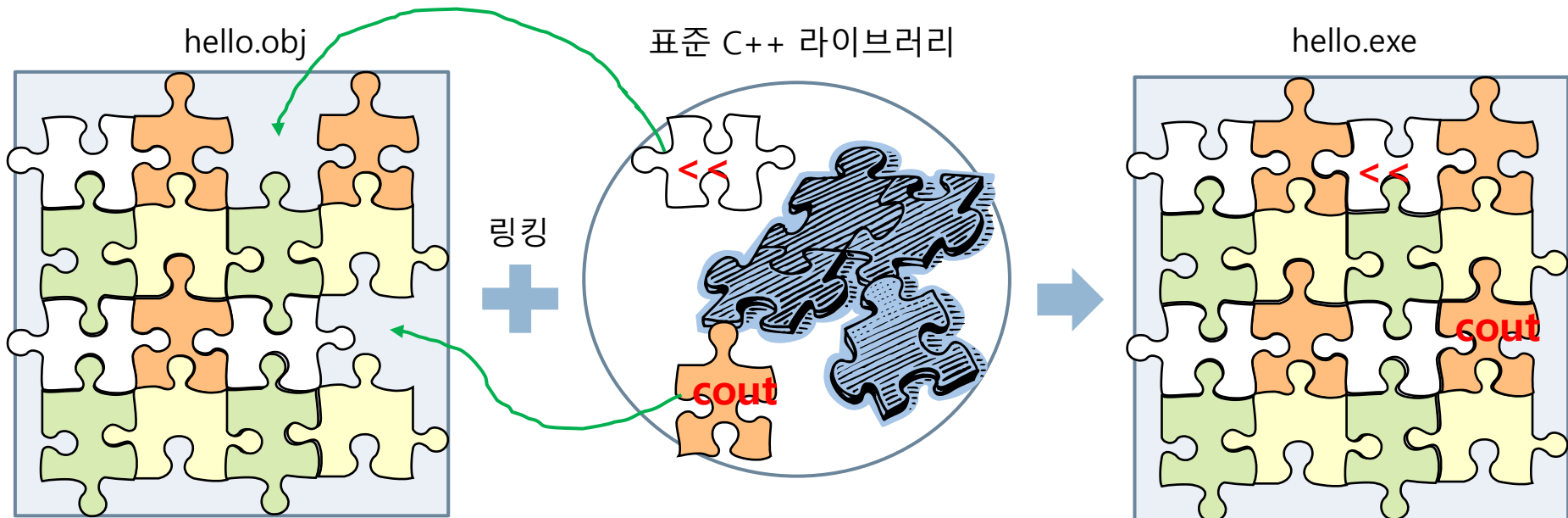
링킹

8

□ 링킹

- ▣ 목적 파일끼리 합쳐 실행 파일을 만드는 과정
 - 목적 파일은 바로 실행할 수 없음
- ▣ 목적 파일과 C++ 표준 라이브러리의 함수 연결, 실행 파일을 만드는 과정

hello.obj + cout 객체 + << 연산자 함수 => hello.exe를 만듦



C++ 표준 라이브러리

9

□ C++ 표준 라이브러리는 3 개의 그룹으로 구분

▣ C 라이브러리

- 기존 C 표준 라이브러리를 수용, C++에서 사용할 수 있게 한 함수들
- 이름이 c로 시작하는 헤더 파일에 선언됨

▣ C++ 입출력 라이브러리

- 콘솔 및 파일 입출력을 위한 라이브러리

▣ C++ STL 라이브러리

- 제네릭 프로그래밍을 지원하기 위해 템플릿 라이브러리

| | | | | |
|-----------|---------|------------|---------|-----------|
| algorithm | complex | exception | list | stack |
| bitset | csetjmp | fstream | locale | stdexcept |
| cassert | csignal | functional | map | strstream |
| cctype | cstdarg | iomanip | memory | streambuf |
| cerrno | cstddef | ios | new | string |
| cfloat | cstdio | iosfwd | numeric | typeinfo |
| ciso646 | cstdlib | iostream | ostream | utility |
| climits | cstring | istream | queue | valarray |
| clocale | ctime | iterator | set | vector |
| cmath | deque | limits | sstream | |

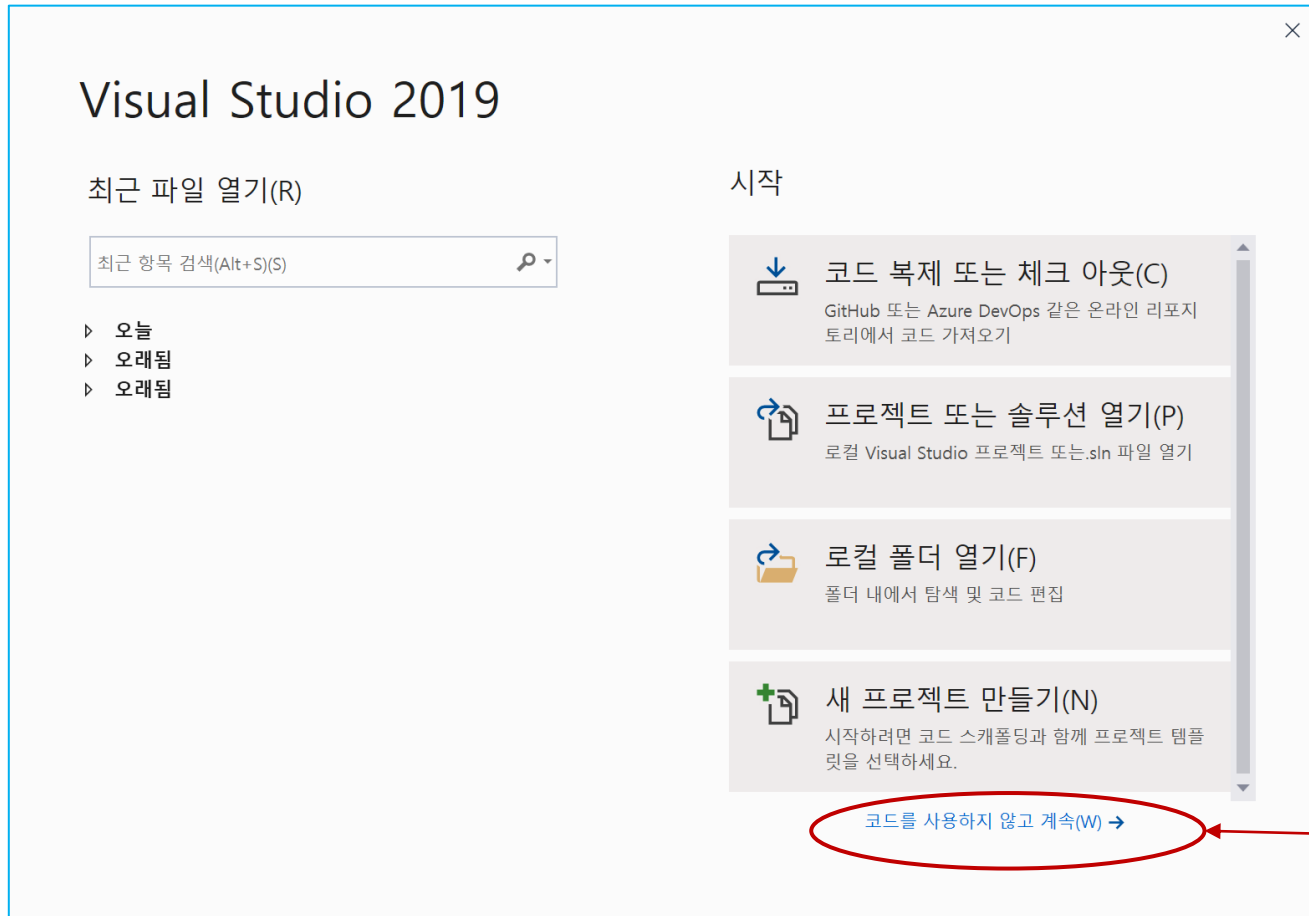
STL 라이브러리

C 라이브러리

C++ 입출력 라이브러리

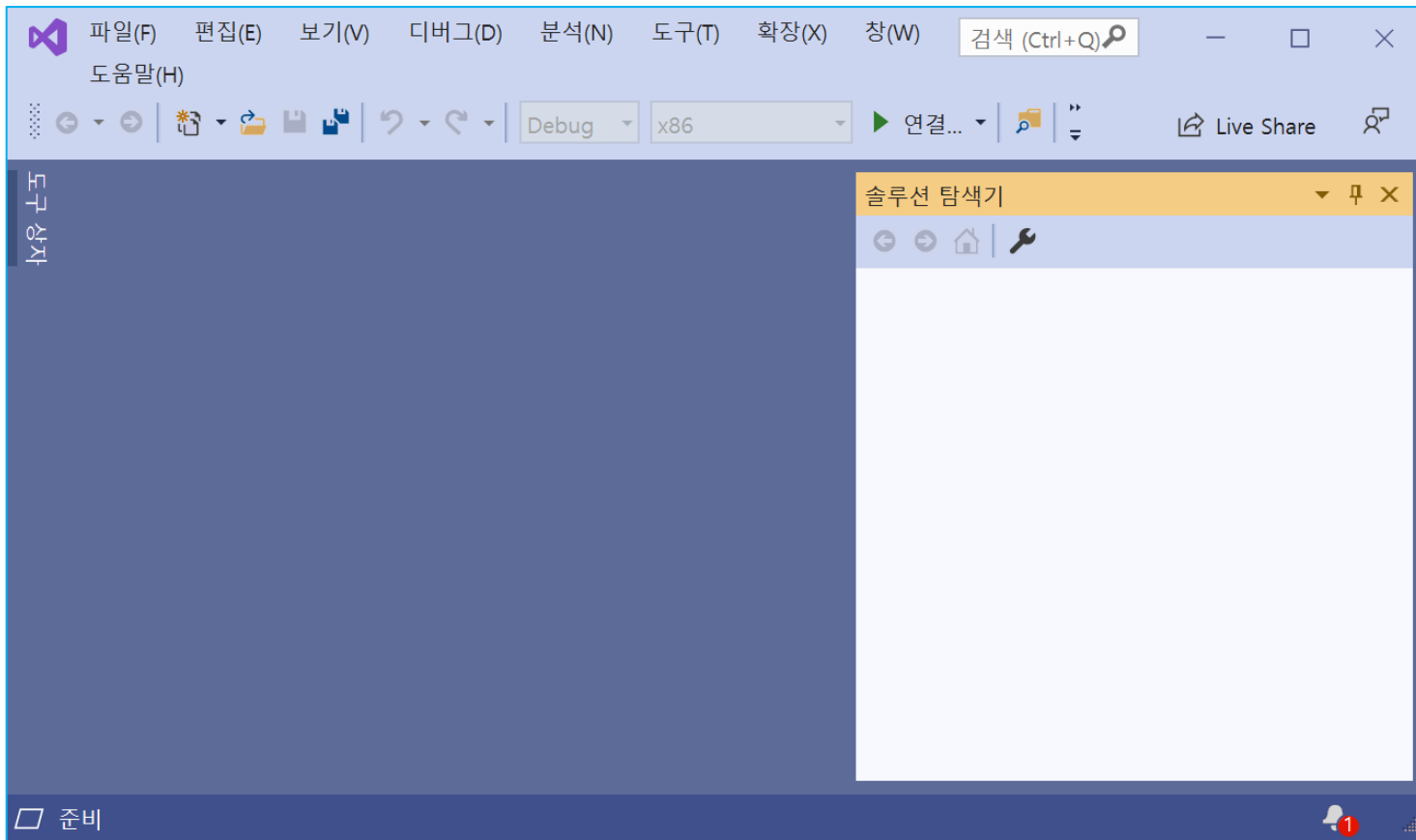
*`<new>` 헤더 파일은 STL에 포함되지 않는 기타 기능을 구현함

Visual Studio 시작



클릭하면
다음 슬라이드로


Visual Studio 스크린




프로젝트 만들기

새 프로젝트 만들기

최근 프로젝트 템플릿(R)

 빈 프로젝트 C++

 Windows 데스크톱 마법사 C++

템플릿 검색(Alt+S)(S)



모두 지우기(C)

C++

모든 플랫폼(P)

모든 프로젝트 형식(T)



빈 프로젝트

Windows용 C++를 사용하여 처음부터 시작합니다. 시작 파일을 제공하지 않습니다.

C++

Windows

콘솔



콘솔 앱

Windows 터미널에서 코드를 실행합니다. 기본적으로 "Hello World"를 출력합니다.

C++

Windows

콘솔



Windows 데스크톱 마법사

마법사를 사용하여 고유한 Windows 앱을 만드세요.

C++

Windows

데스크톱

콘솔

라이브러리



Windows 데스크톱 애플리케이션

Windows에서 실행되는 그래픽 사용자 인터페이스를 사용하는 애플리케이션용 프로젝트입니다.

C++

Windows

데스크톱

다음(N)

새 프로젝트 구성

×

새 프로젝트 구성

빈 프로젝트 C++ Windows 콘솔

프로젝트 이름(N)

Hello

C:\WC++\Wchap1\Hello
폴더가 생긴다.

위치(L)

C:\WC++

솔루션 위치

...

솔루션 이름(M) ⓘ

chap1

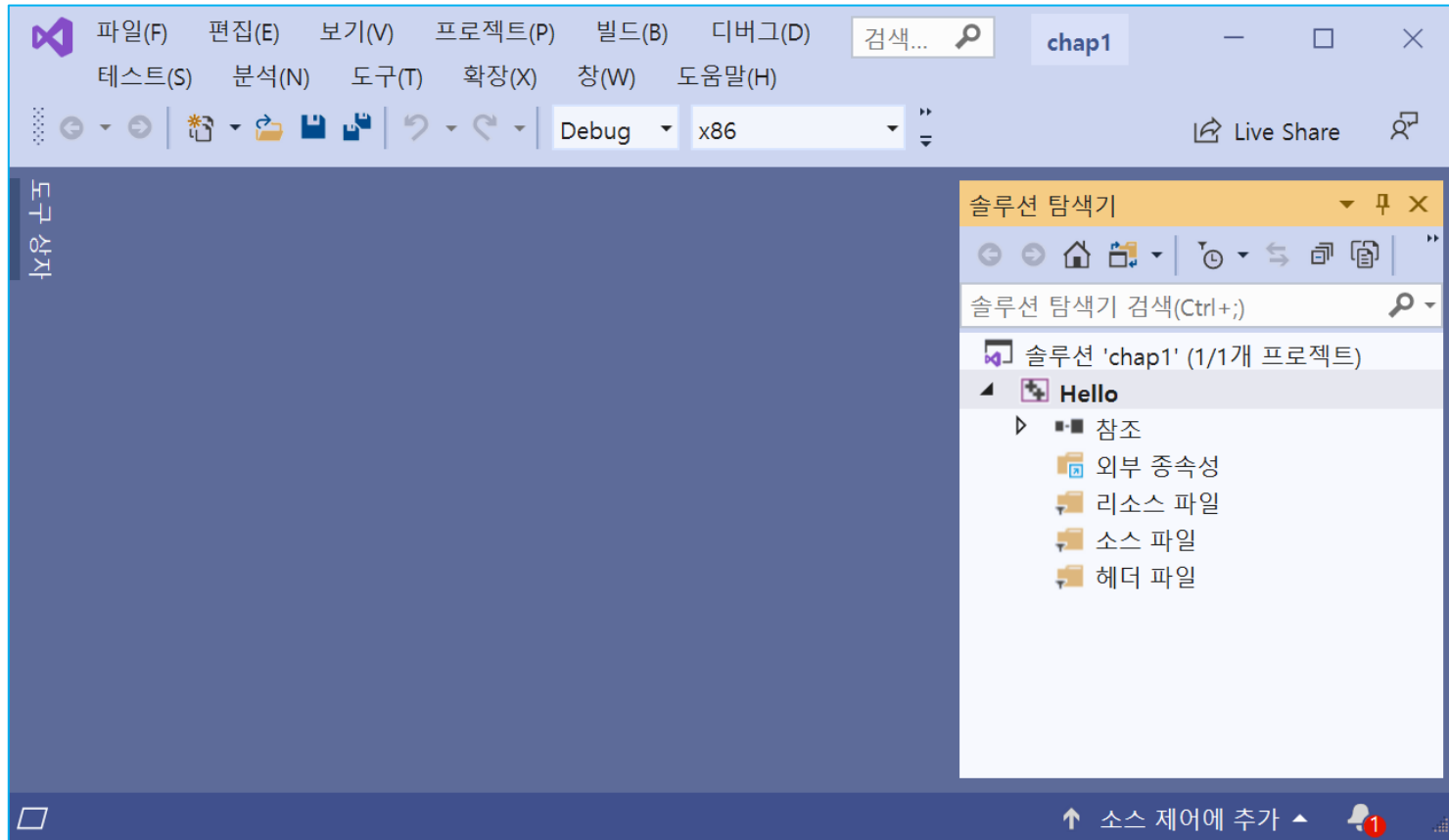
C:\WC++\Wchap1
폴더를 생성한다.

☐ 솔루션 및 프로젝트를 만든 후 Visual Studio에서 열기(R)

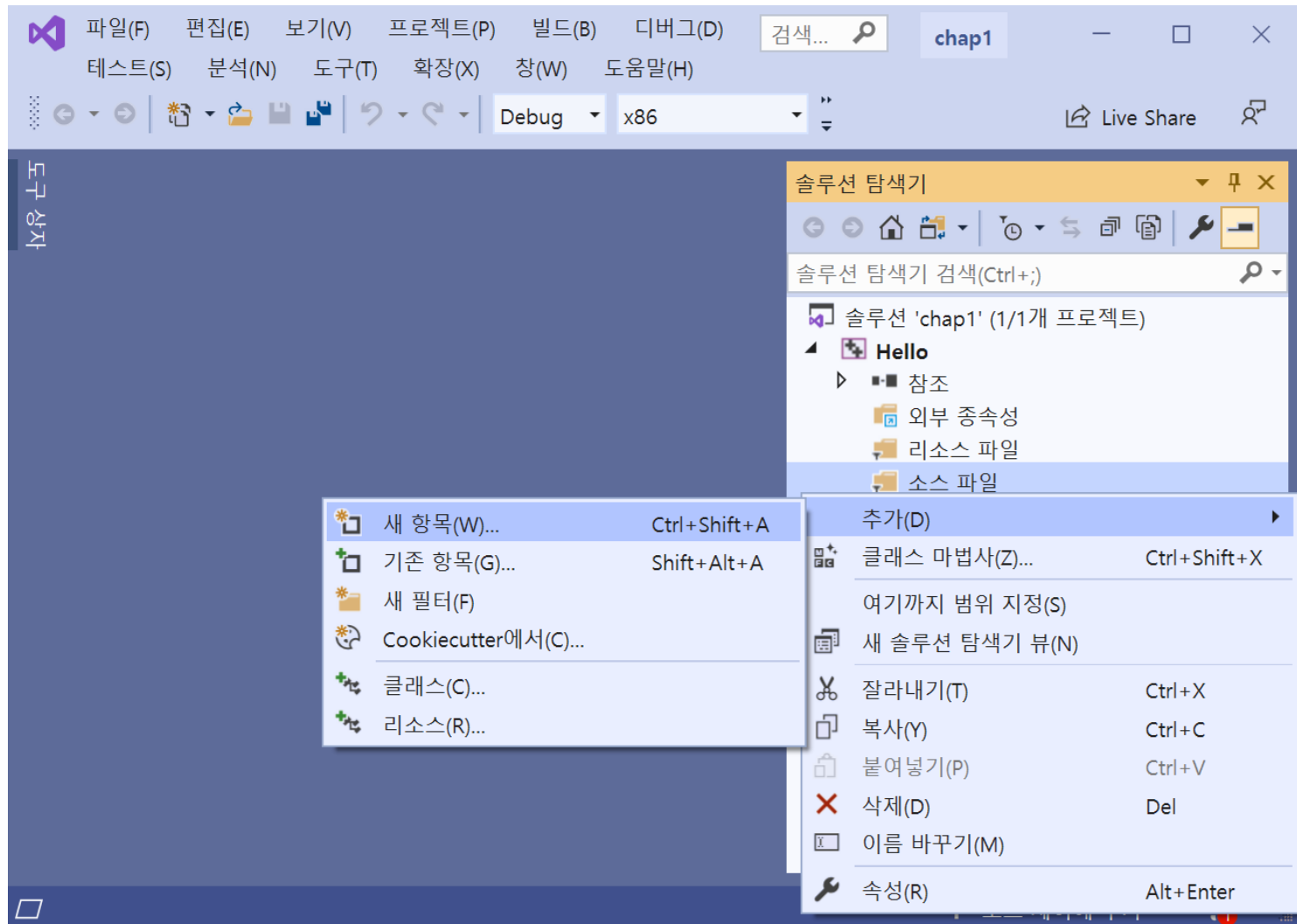
뒤로(B)

만들기(C)

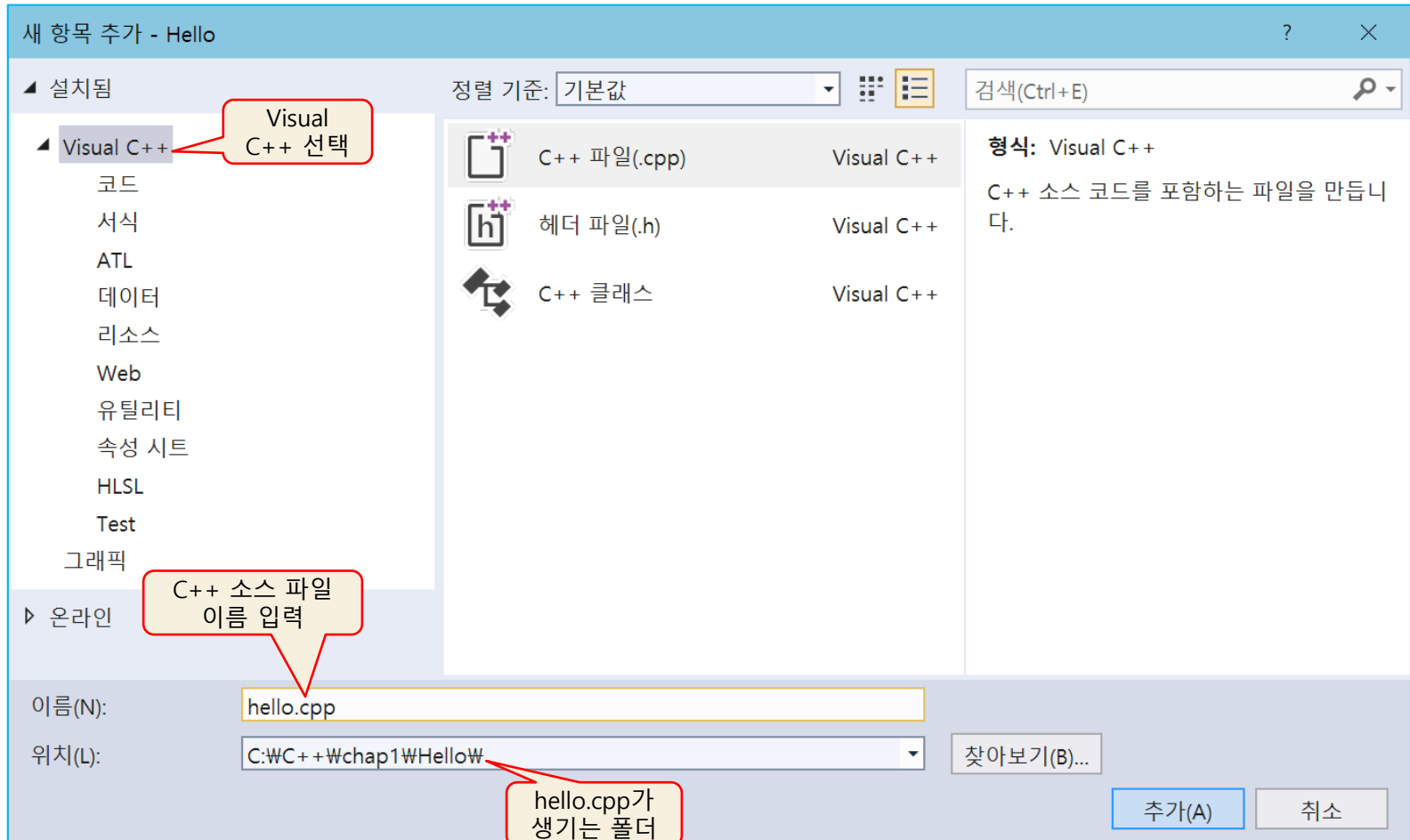
Hello 프로젝트 생성 후



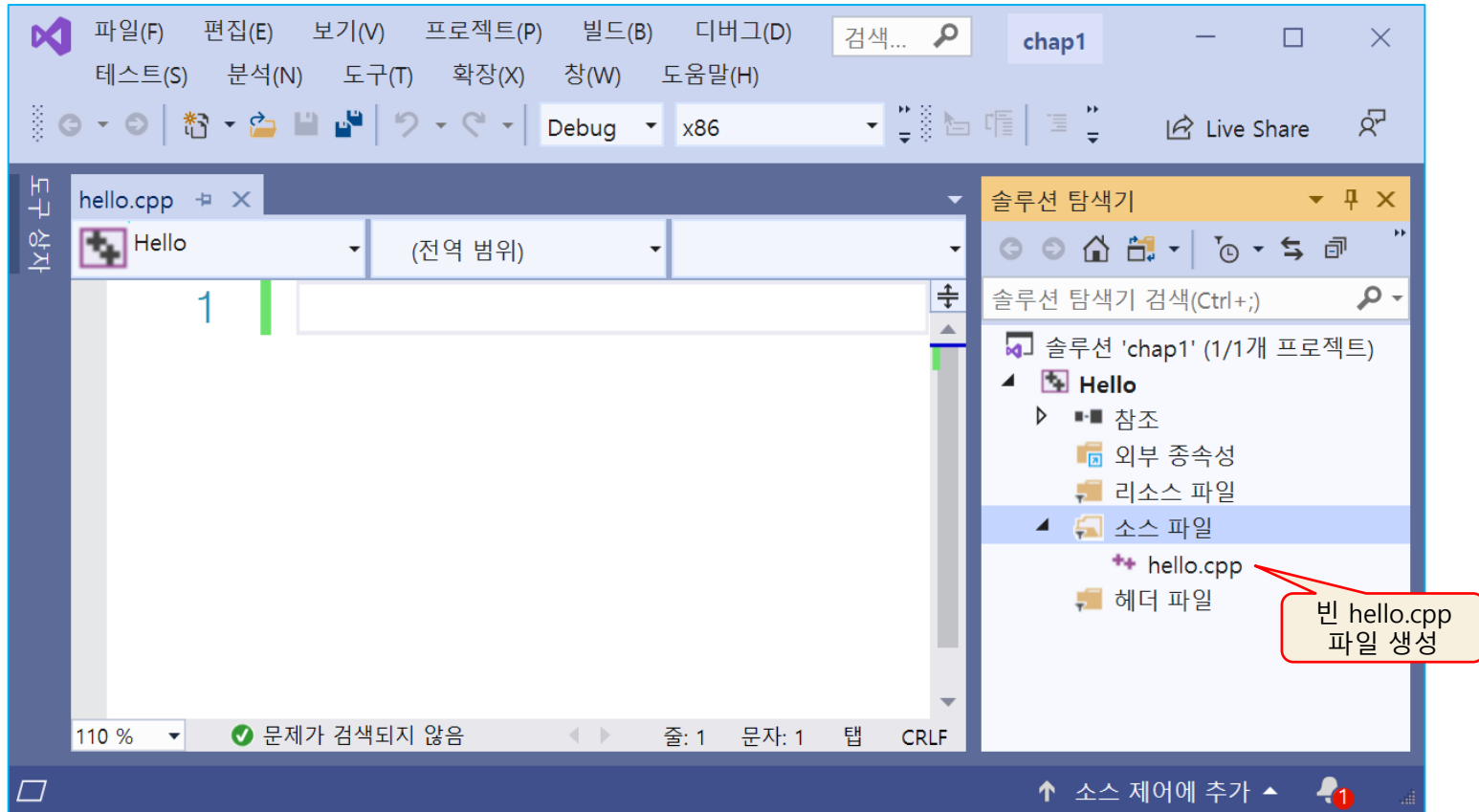
새 항목 만드는 메뉴 선택



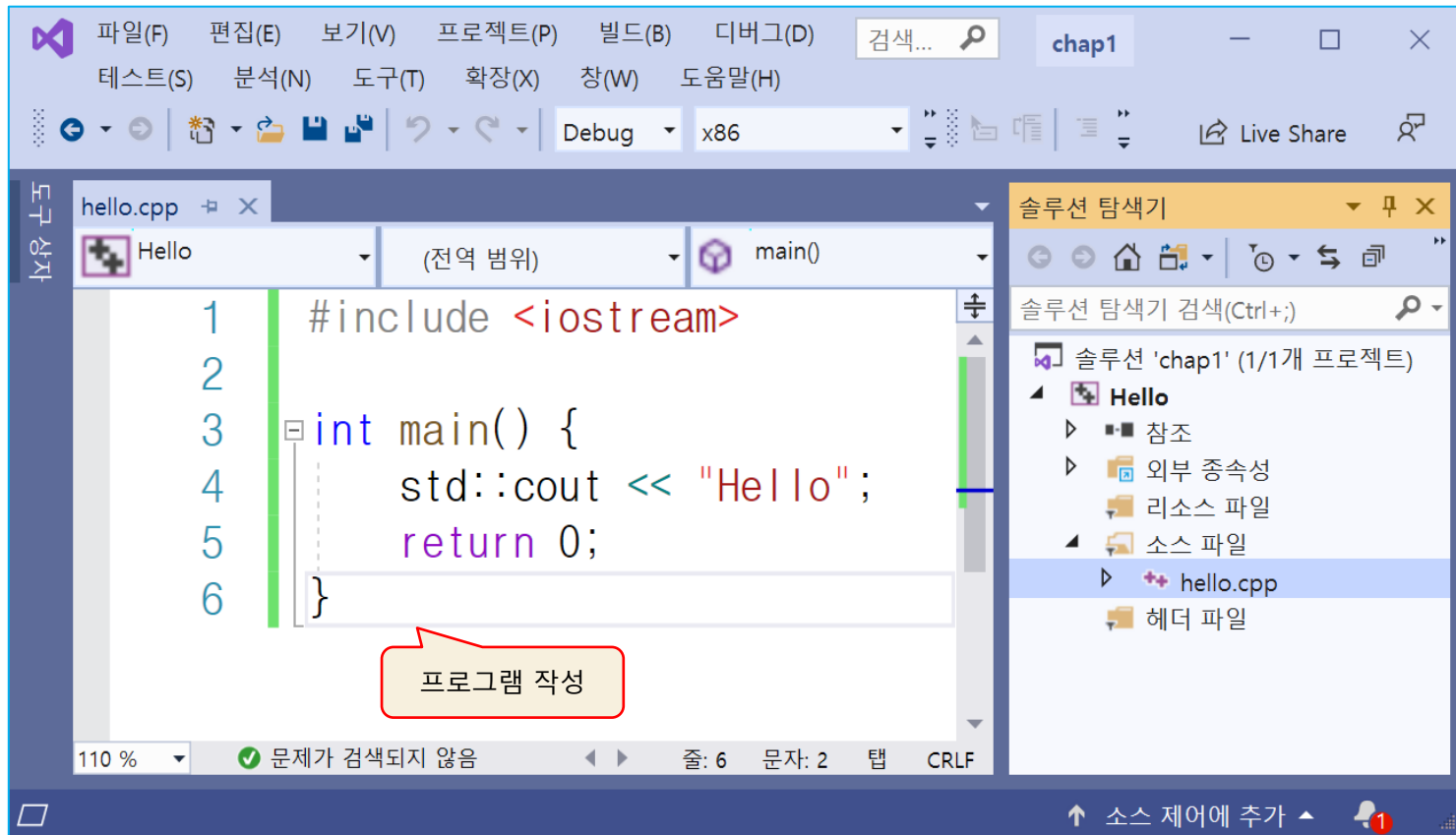
hello.cpp 소스 파일 생성



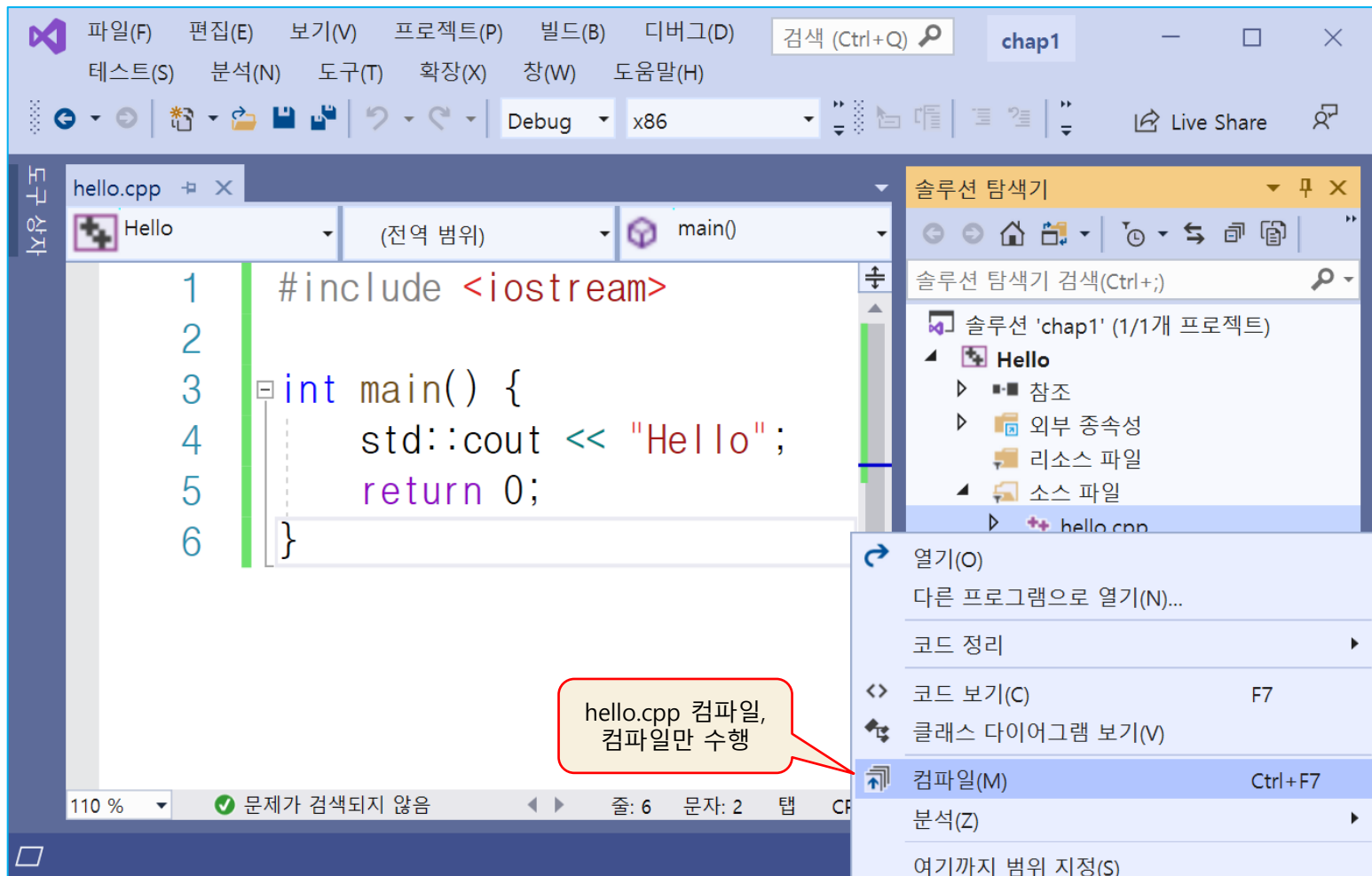
hello.cpp 파일이 생성된 초기 모습



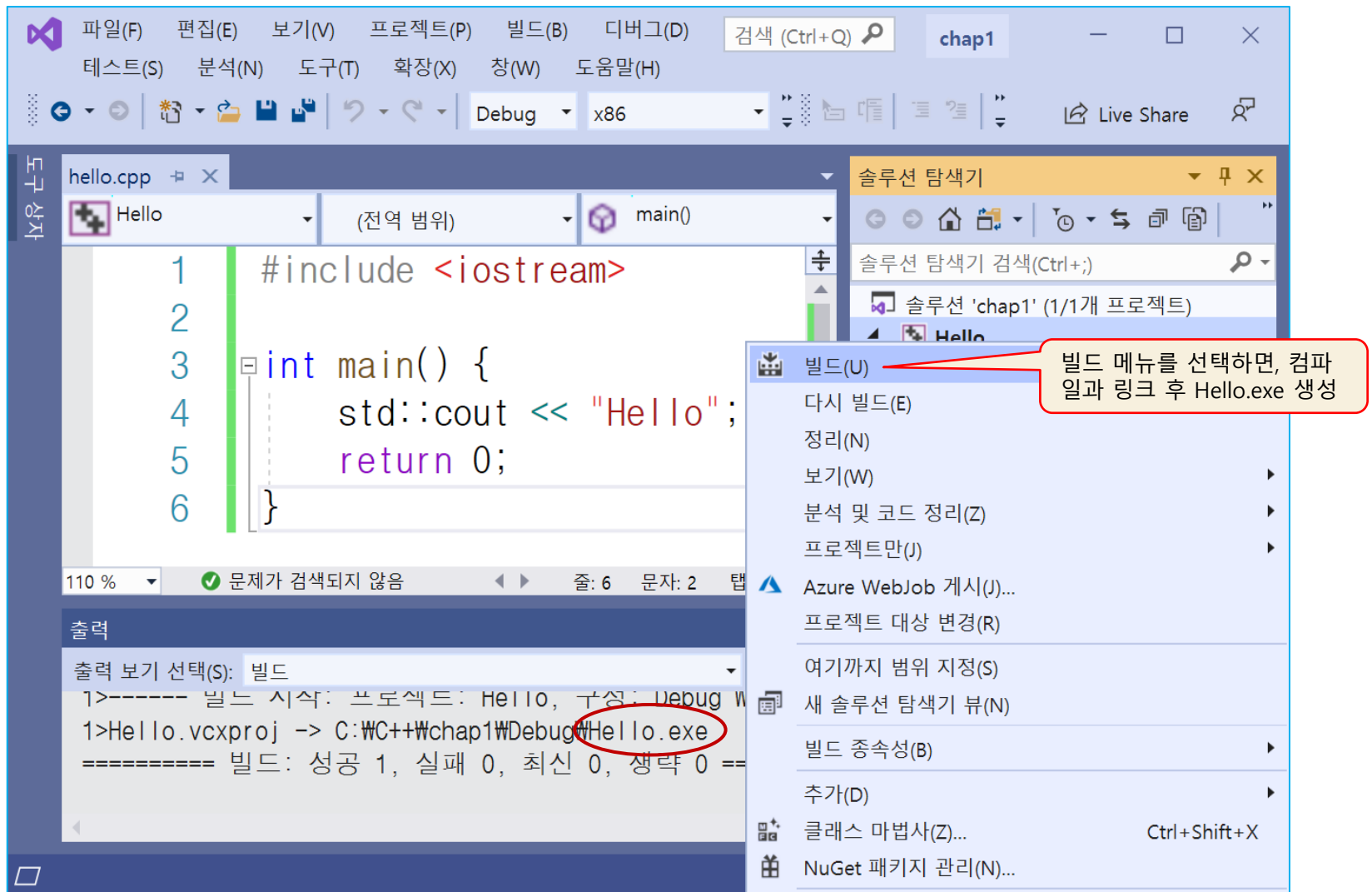
hello.cpp 작성



솔루션 탐색기에서 컴파일 메뉴 선택



Hello 프로젝트의 빌드로 Hello.exe 생성



Hello 프로젝트가 실행되는 화면

The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Menu Bar:** 파일(F), 편집(E), 보기(V), 프로젝트(P), 빌드(B), 디버그(D), 검색 (Ctrl+Q), 테스트(S), 분석(N), 도구(T), 확장(X), 창(W), 도움말(H).
- Toolbar:** Includes icons for file operations, build, and debug. The 'Debug' dropdown is set to 'x86'.
- Editor:** Shows the source file 'hello.cpp' with the following code:

```
1 #include <iostream>
2
3 int main() {
4     std::cout << "Hello";
5     return 0;
6 }
```
- Solution Explorer:** Displays the project structure for 'chap1' (1/1개 프로젝트), including 'Hello' and its sub-items: 참조, 외부 종속성, 리소스 파일, 소스 파일 (hello.cpp), and 헤더 파일.
- Output Window:** Shows the build output for 'Hello'. The output indicates a successful build:

```
1>----- 빌드 시작: 프로젝트: Hello, 구성: Debug Win32 -----
1>Hello.vcxproj -> C:\WC++Wchap1WDebugWHello.exe
===== 빌드: 성공 1, 실패 0, 최신 0, 생략 0 =====
```
- Microsoft Visual Studio 디버그 콘솔:** A separate window showing the program's output:

```
Hello
C:\WC++Wchap1WDebugWHello.exe(프로세스 23196개)이(가) 종료
되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

A red callout box points to the 'Hello 프로젝트 빌드 성공. Hello.exe 생성' message in the Output window.