

# 1장 C++ 프로그래밍 언어

1. C++ 프로그램 개발
2. C++ 프로그램의 디버깅
3. C++ 프로그램 언어의 특징
4. 프로그래밍 기법
5. C++ 언어로 프로그램 하기
6. C++ 배우기

# 학습목표

- Visual Studio에서 C++ 프로그램을 개발하는 과정에 대하여 설명할 수 있다.
- Visual Studio에서 C++ 프로그램을 사용할 수 있다.
- C++ 언어의 특징을 설명할 수 있다.
- 프로그래밍 기법에 대하여 설명할 수 있다.

# C++ 프로그램의 역사

- C 언어

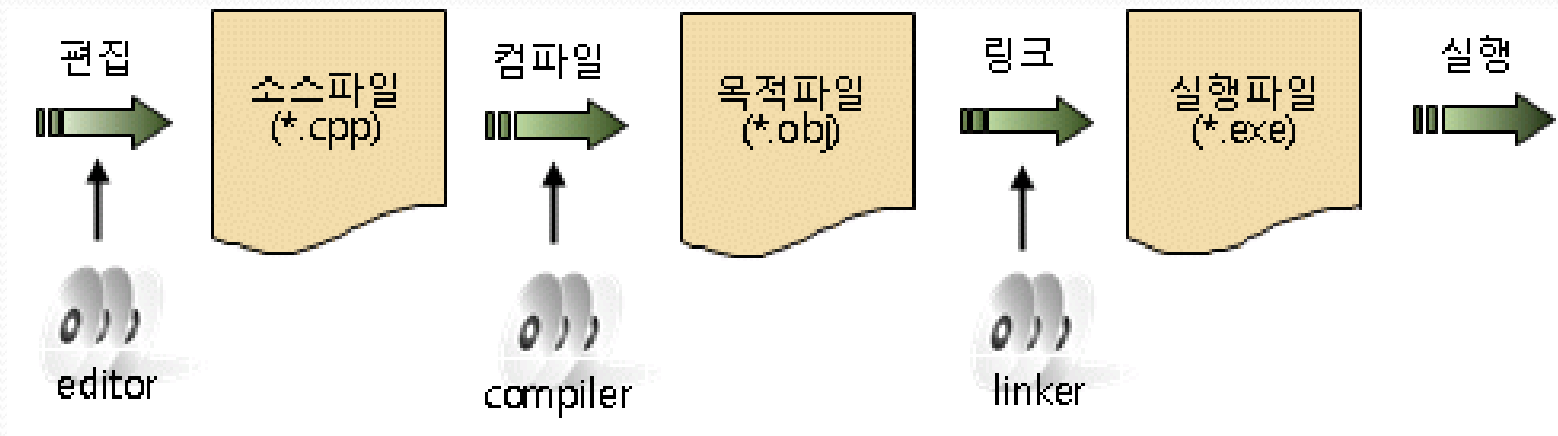
- 1970년대, Bell 연구소의 데니스 리치(Dennis Ritchie)
- 절차적 언어(Procedural Language) : 함수들의 집합으로 프로그래밍
- UNIX 운영체제(OS)를 작성하기 위해 개발

- C++ 언어

- 1980년, Bell 연구소의 비얀 스트로스트럽(Bjarne Stroustrup)
- 객체지향 언어(Object-Oriented Language) : 클래스의 집합으로 프로그래밍
- C with Classes → C++
- 표준
  - C++98
  - C++03
  - C++11
  - C++14

# C++ 프로그래밍 과정

- 프로그램 작성
  - 소스파일(source file)
  - 목적파일(object file)
  - 실행파일(execution file)



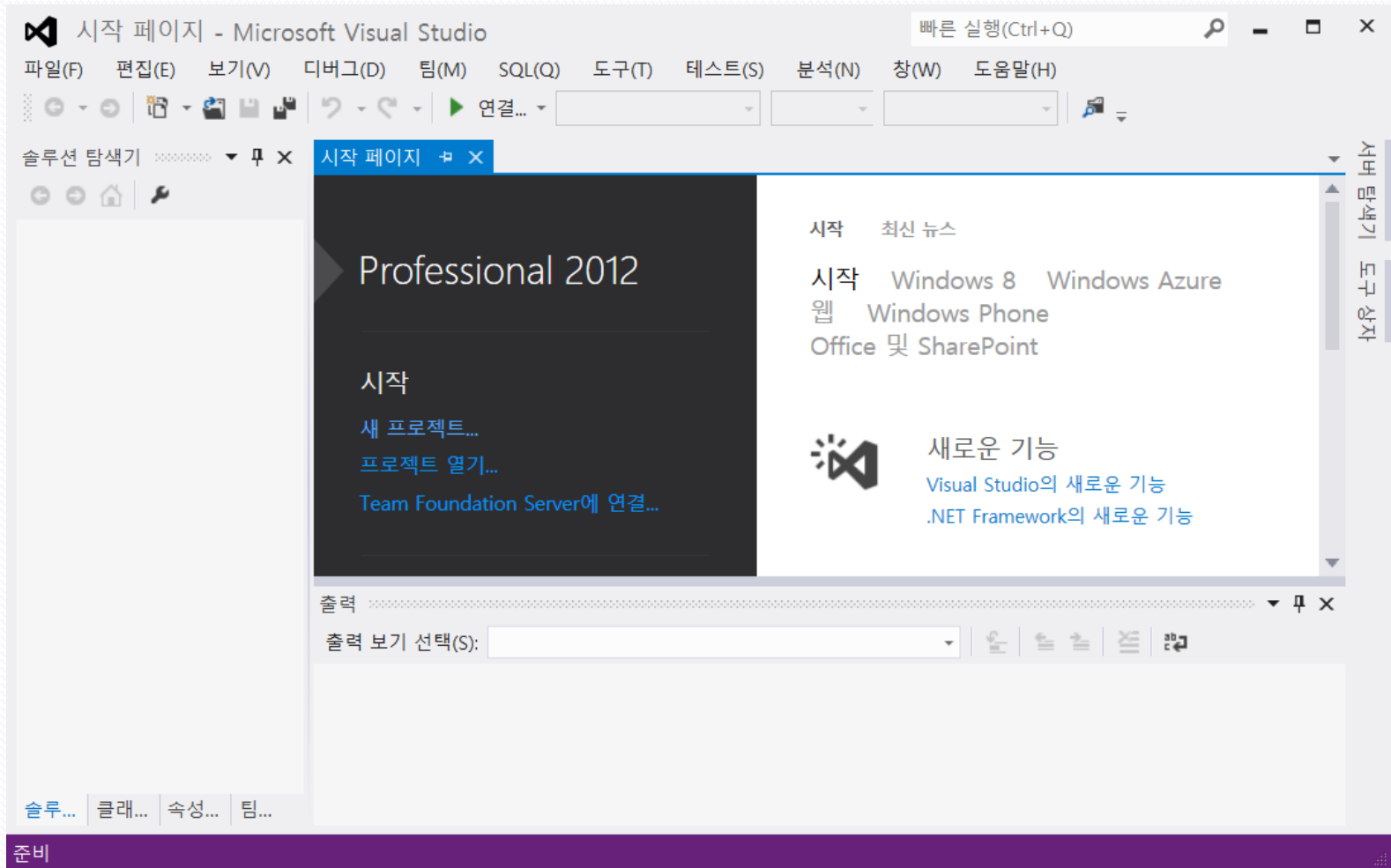
- 오류(error) 수정: 디버깅(debugging)
  - 실행 추적,
  - 디버깅 툴 사용

# C++ 개발 환경

- 통합개발환경(IDE, Integrated Development Environment)
  - 편집, 컴파일, 링크, 그리고 디버깅 툴 지원
  - 판매 제품
  - 혹은 무료로 인터넷에서 다운로드하여 사용 :
    - Dev-C++
    - Microsoft Visual Studio Express version
- Visual Studio 2012
  - 이 책에서 사용하는 통합개발 환경

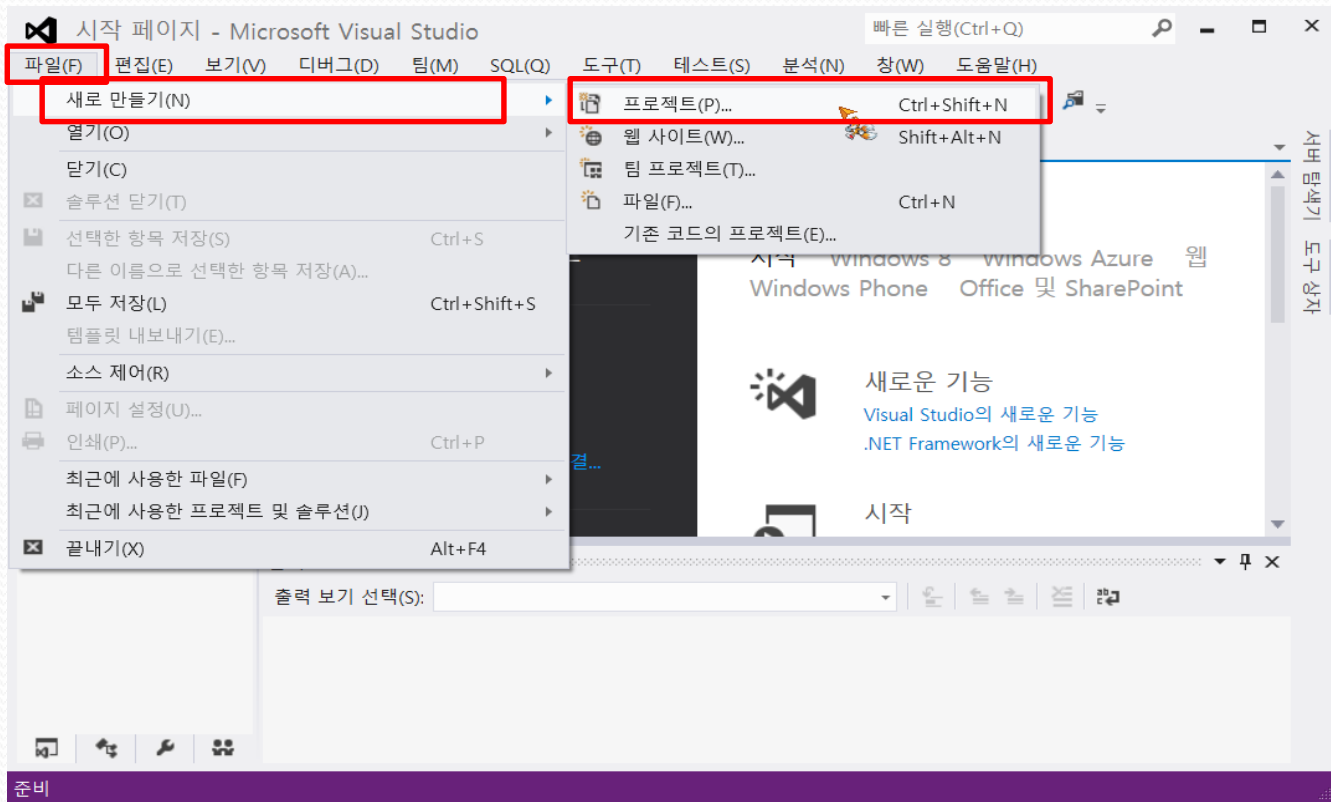
# Visual C++ 실행하기

- Visual Studio를 실행 (



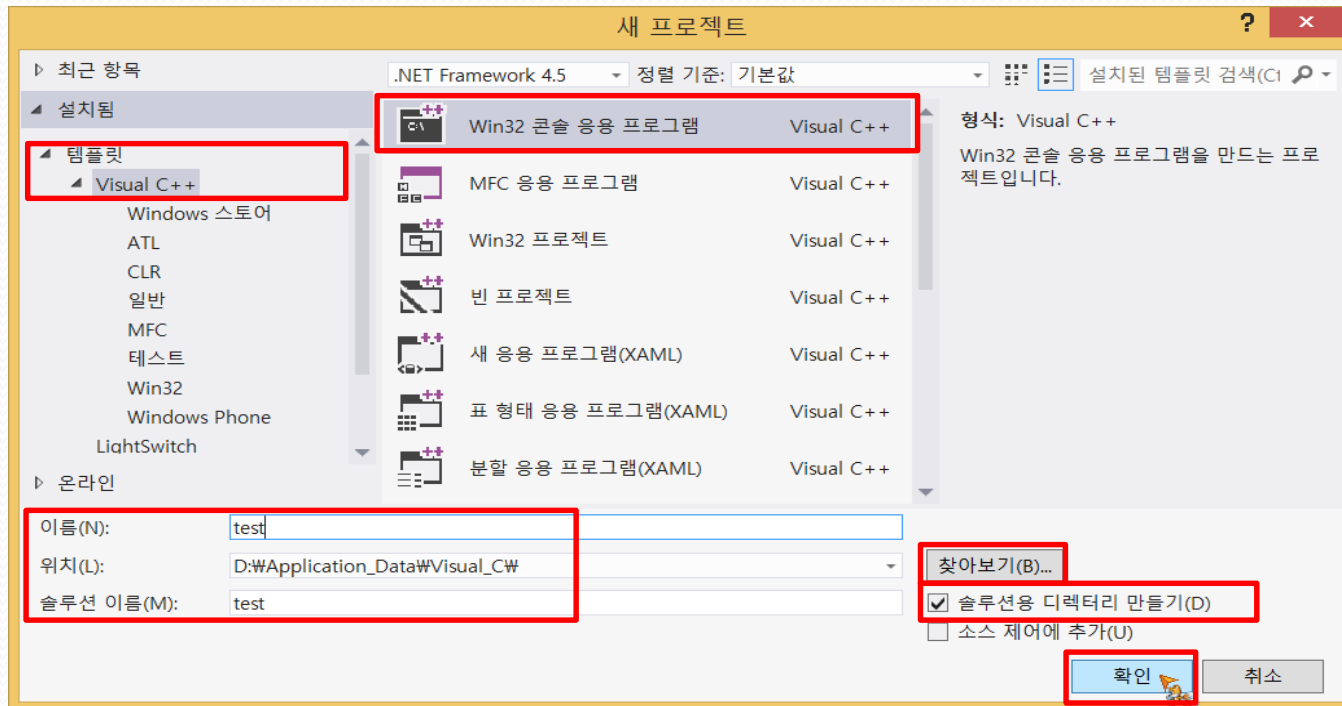
# 프로젝트 생성하기

- C++ 프로그램
  - 소스 파일, 헤더 파일, 리소스 파일 → 프로젝트로 관리
  - test 프로젝트 만들기
- [파일] / [새로 만들기] / [프로젝트] → [새 프로젝트] 대화상자



# 프로젝트 생성하기

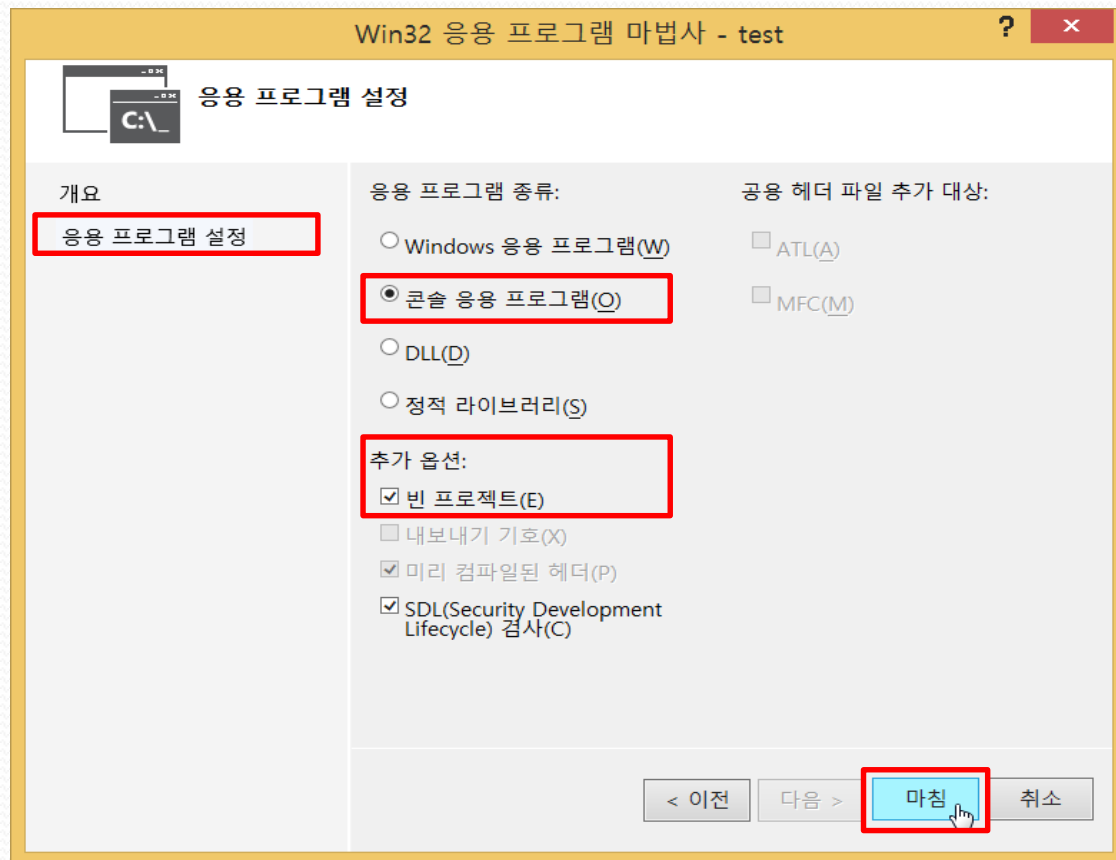
- [새 프로젝트] 대화상자
  - [Win32 콘솔 응용 프로그램] 선택
  - [찾아보기]를 눌러 위치를 정하고, 프로젝트 이름을 입력한다.
  - [솔루션용 디렉터리 만들기]를 체크하고 [확인]을 누른다.
  - 여기서는 프로젝트 이름을 test로 하자.





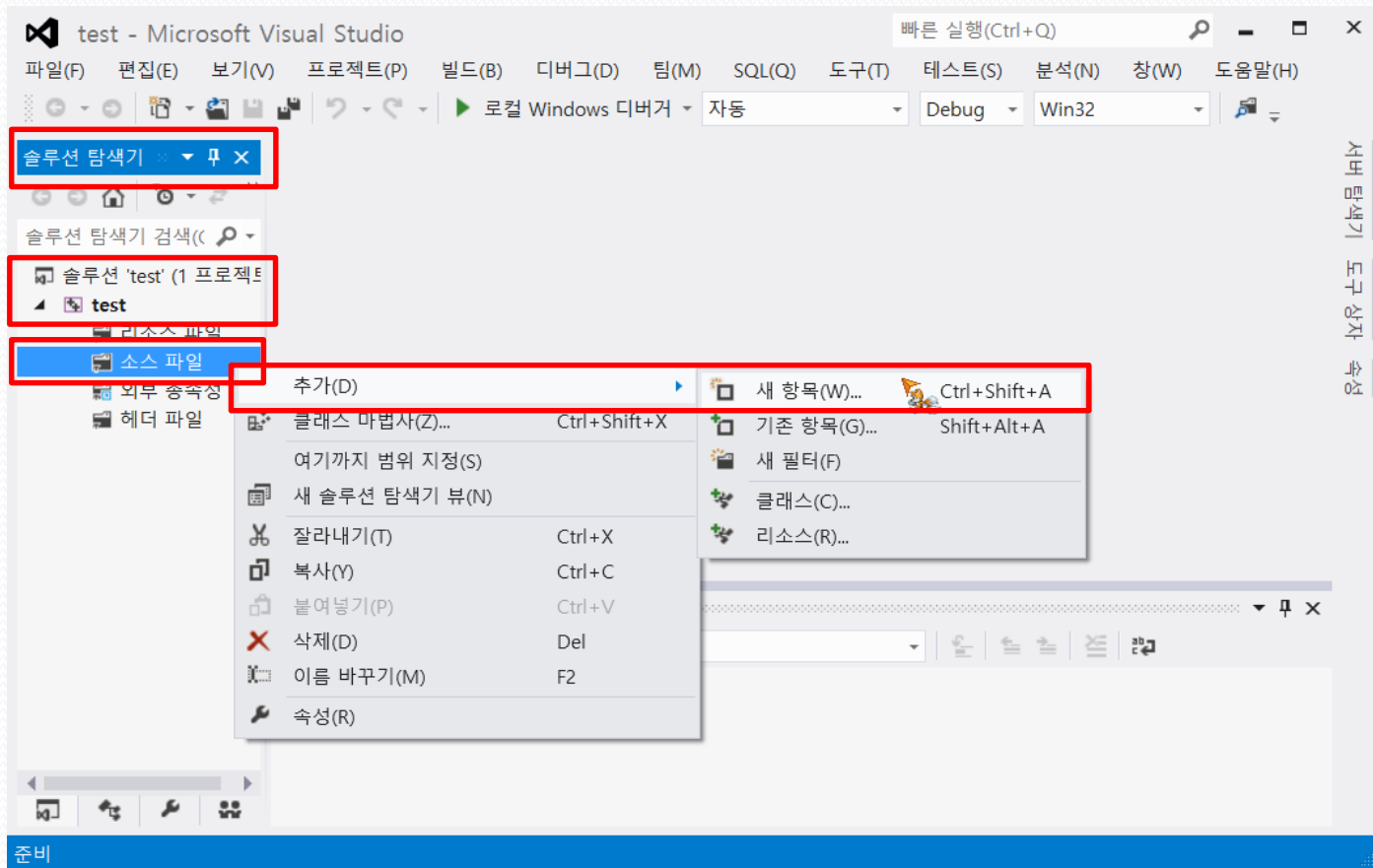
# 프로젝트 생성하기

- [Win32 응용 프로그램 마법사]의 [응용 프로그램 설정] 탭
  - [콘솔 응용 프로그램]을 확인하고, [추가 옵션]에서 [빈 프로젝트]를 체크하고 [마침] 버튼을 누른다.



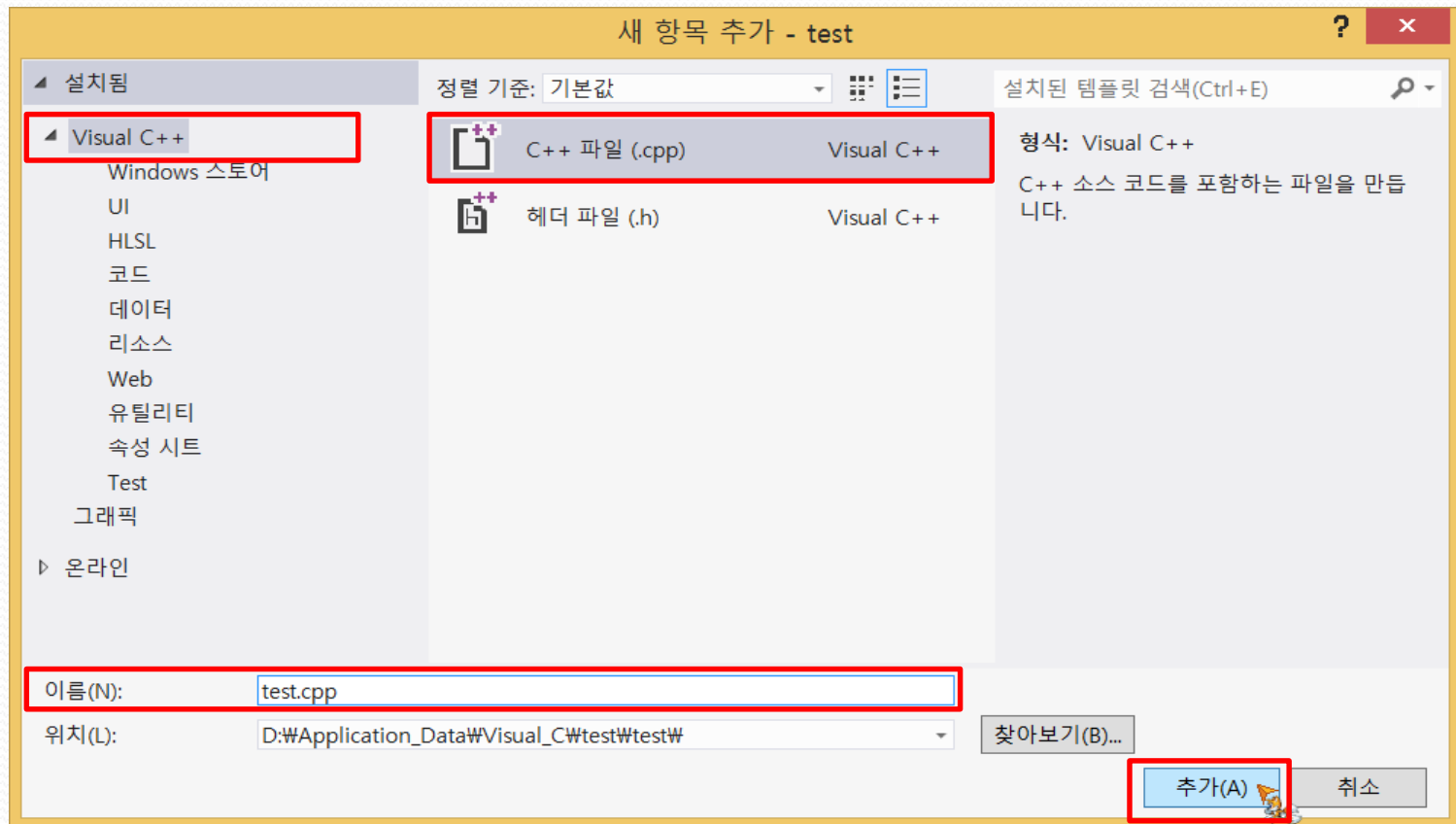
# 소스 파일 추가하기

- test 솔루션(프로젝트)에 소스 파일 test.cpp 만들기
  - [솔루션 탐색기]의 [test] 프로젝트에서 마우스를 [소스 파일]에 두고
  - 마우스 오른쪽 버튼 → 팝업 메뉴에서 [추가] / [새 항목...]을 클릭



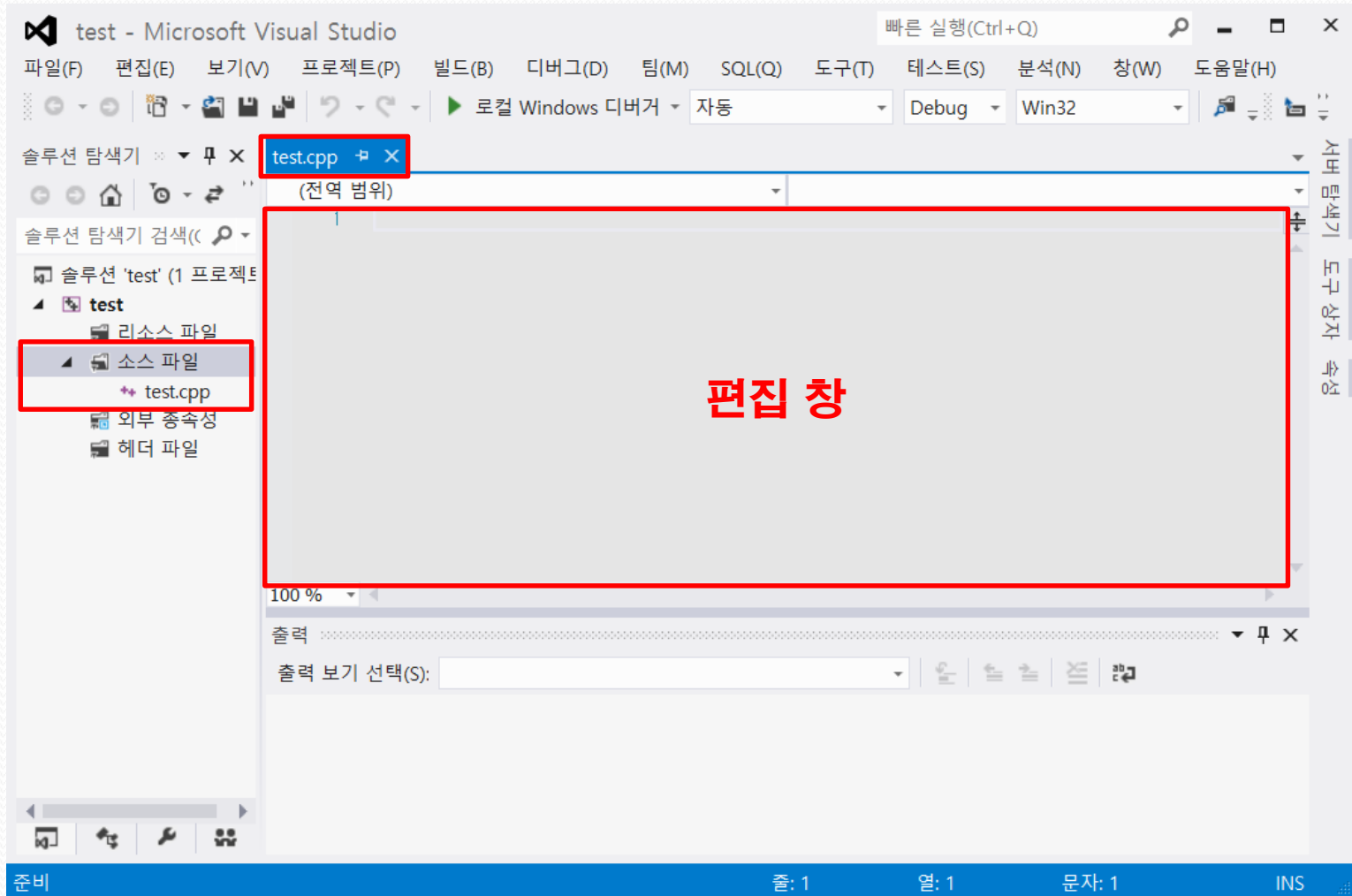
# 소스 파일 추가하기

- [새 항목 추가] 대화상자
  - [C++ 파일]과 [헤더 파일] 중에서 선택하고 파일의 이름을 입력하고
  - 여기서는 파일 이름을 test.cpp로 하자.



# 소스 파일 추가하기

- [추가] 버튼을 클릭하면 → test.cpp 소스 파일 생성



# 소스 코드 편집하기

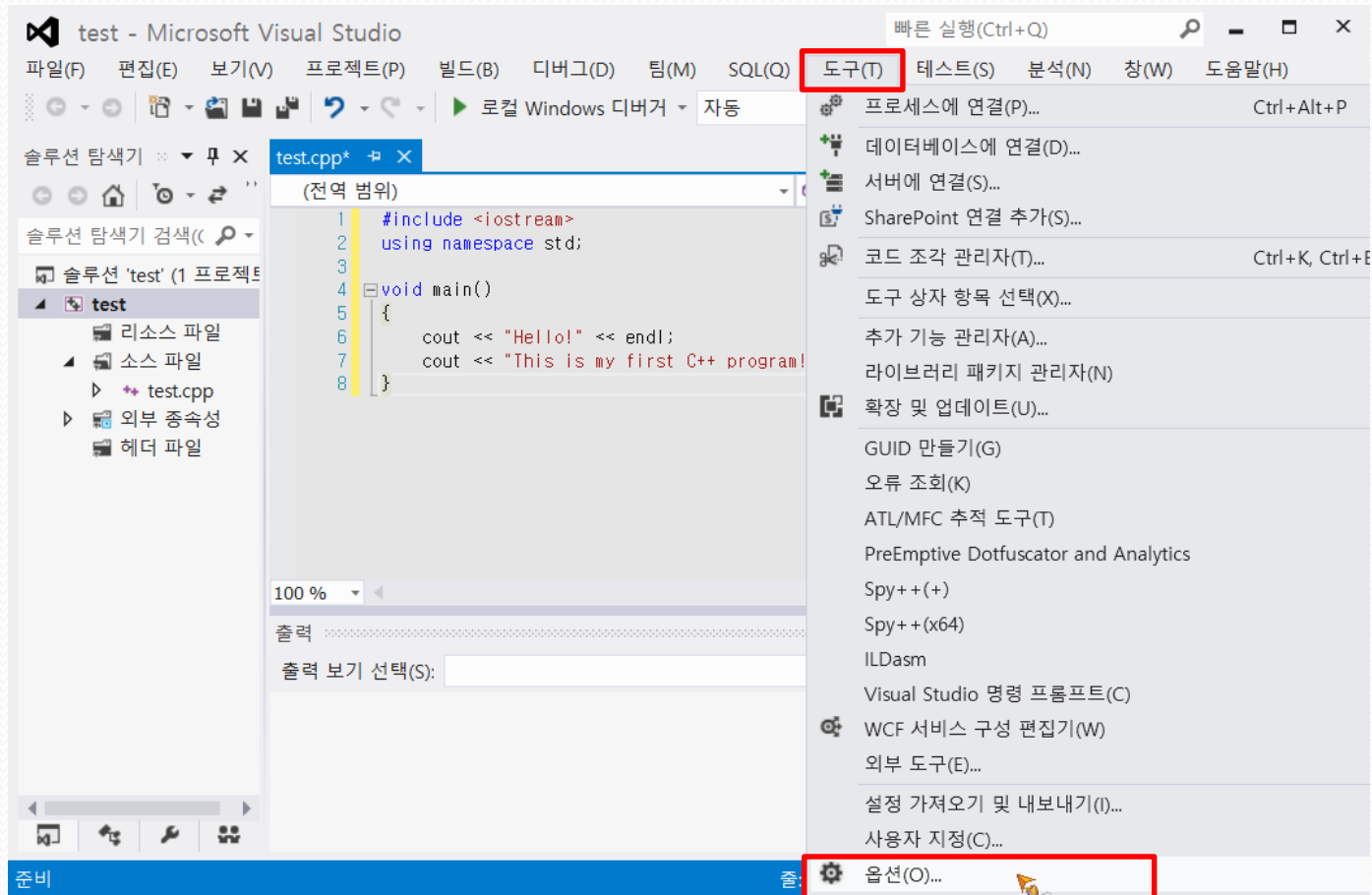
- 예제 소스 코드
  - 편집 창에 다음과 같이 코딩하자.

```
#include <iostream>
using namespace std;
```

```
void main()
{
    cout << "Hello!" << endl;
    cout << "This is my first C++ program!" << endl;
}
```

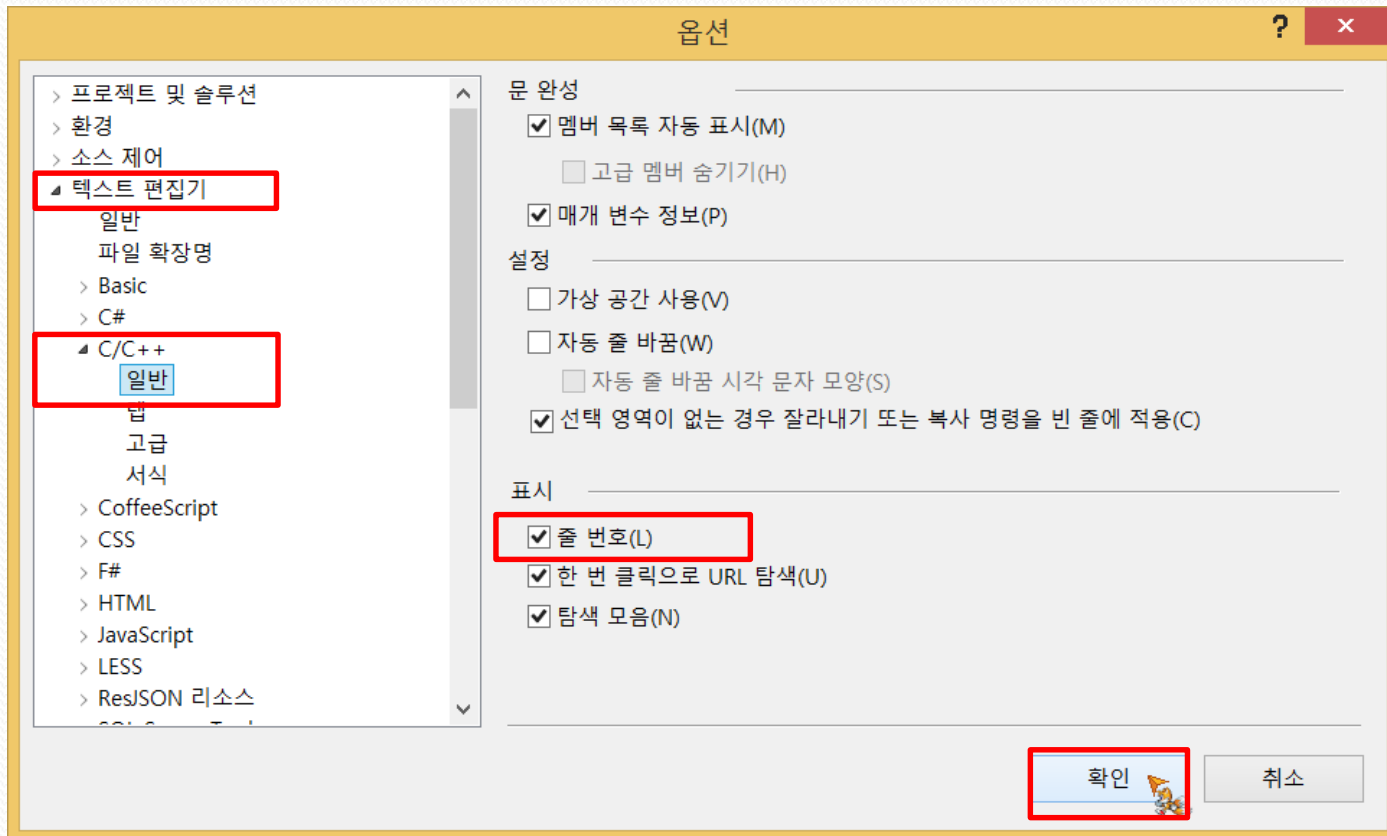
# 소스 코드 편집하기

- 편집 창에 줄 번호 보이기 (1)
  - [도구] / [옵션...] 메뉴 클릭



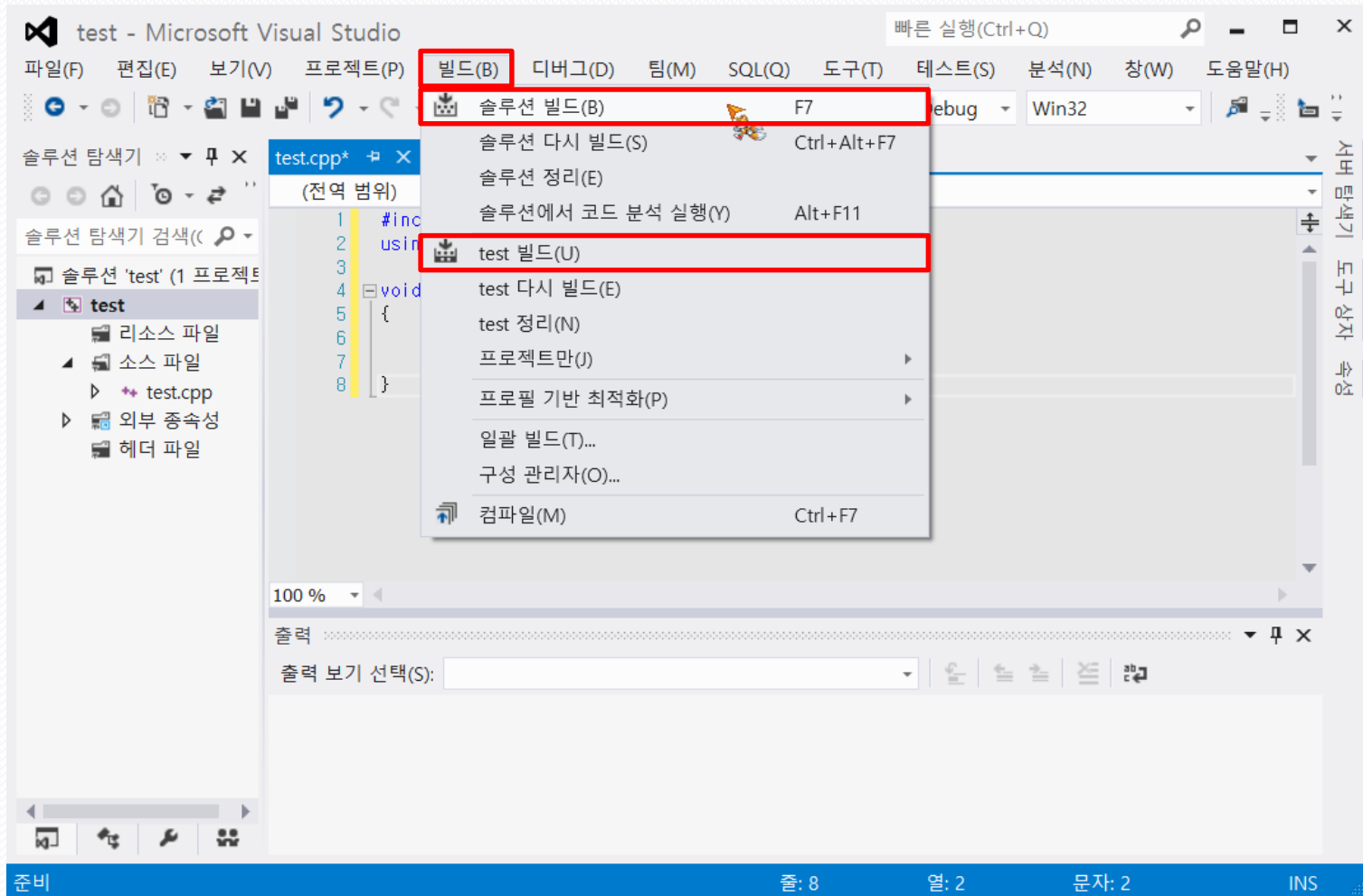
# 소스 코드 편집하기

- 편집 창에 줄 번호 보이기 (2)
  - [옵션] 대화상자, [텍스트 편집기] / [C/C++ 일반]
  - [줄 번호]를 체크하고 [확인] 버튼을 누른다.



# 프로젝트 빌드하기

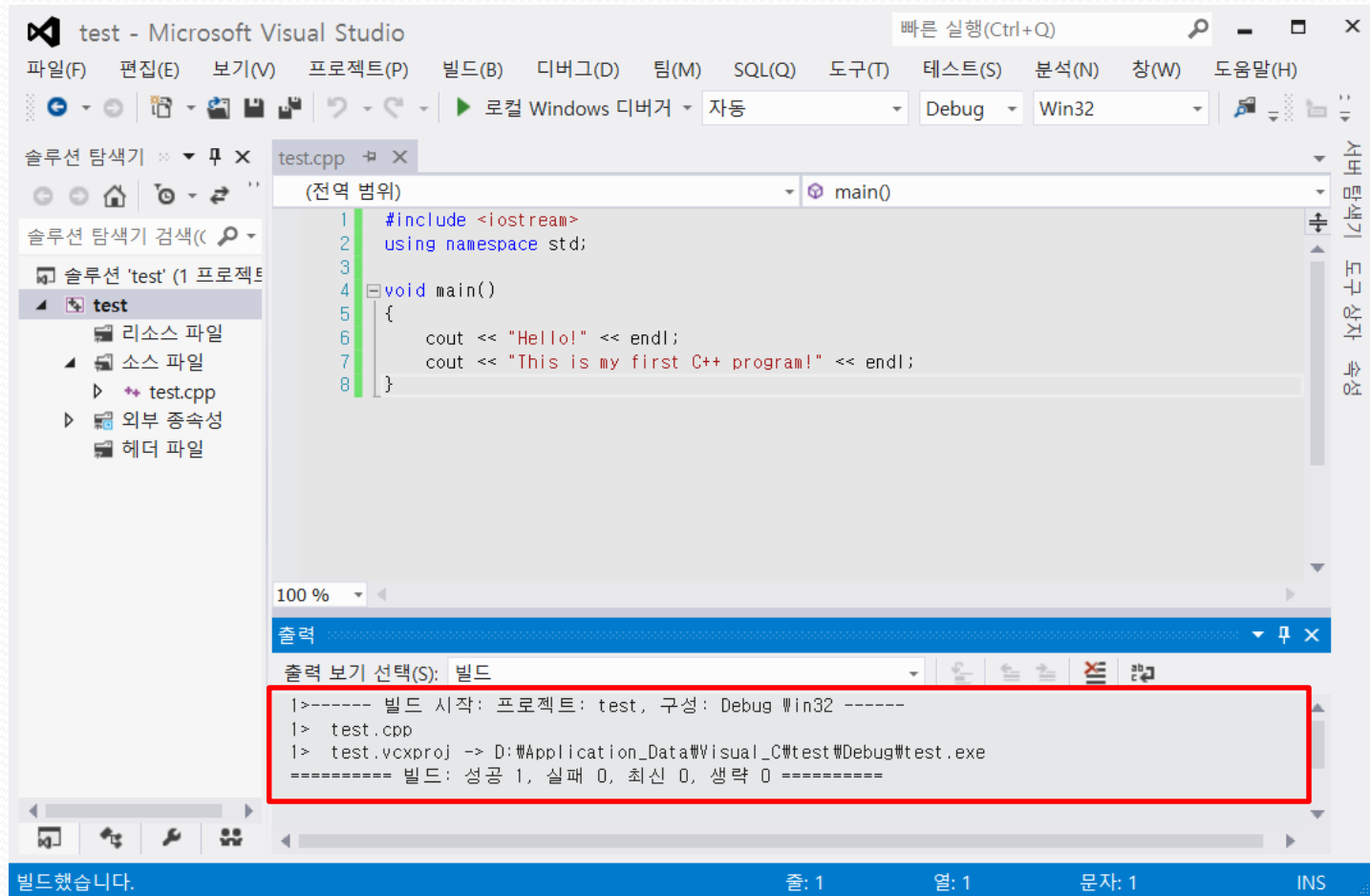
- [빌드] / [솔루션 빌드] (F7) 혹은 [Test 빌드]를 클릭





# 프로젝트 빌드하기

- 빌드 결과 → '빌드: 성공 1, 실패 0' 이면 프로그램 실행할 수 있다.

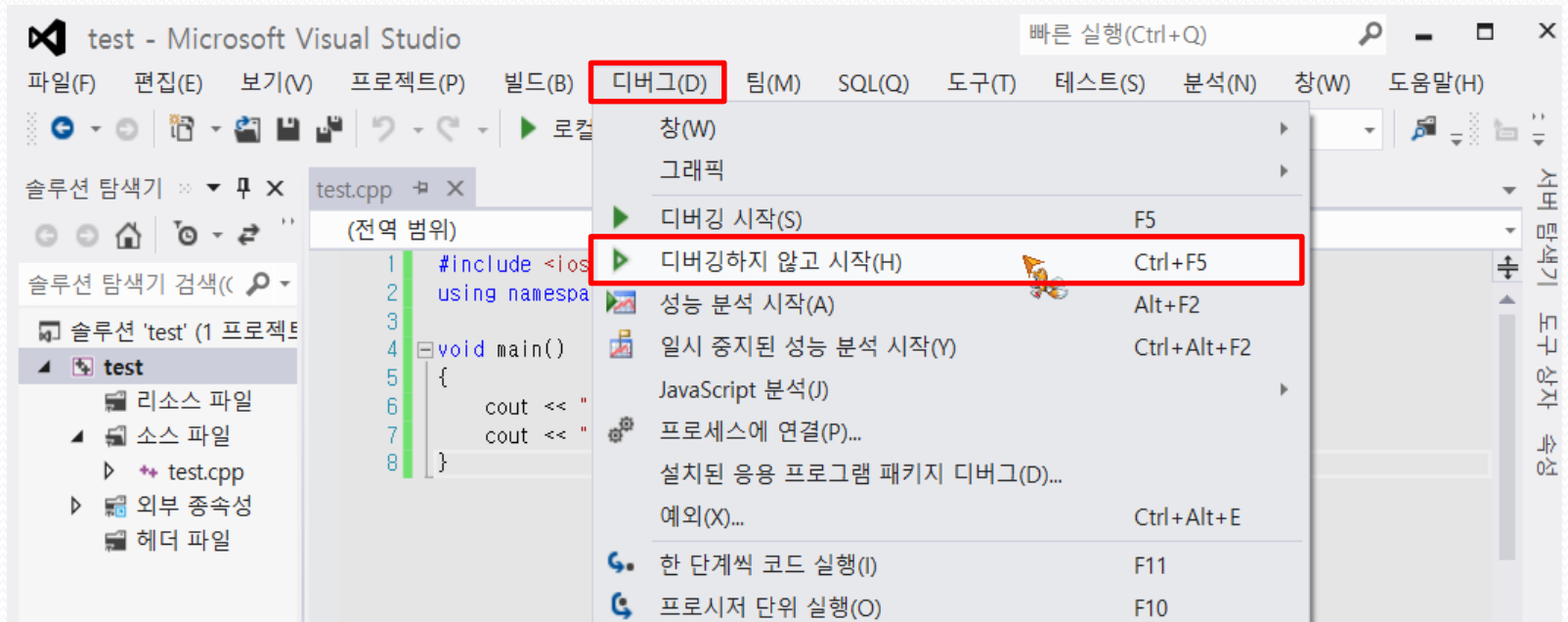


The screenshot shows the Microsoft Visual Studio IDE with a project named 'test'. The main window displays the source code for 'test.cpp', which includes `<iostream>` and uses the `std` namespace. The `main` function prints 'Hello!' and 'This is my first C++ program!'. The left sidebar shows the project structure with 'test' as the active project. The bottom status bar indicates '빌드했습니다.' (Built).

```
test - Microsoft Visual Studio
빠른 실행(Ctrl+Q)
파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버그(D) 팀(M) SQL(Q) 도구(T) 테스트(S) 분석(N) 창(W) 도움말(H)
로컬 Windows 디버거 자동 Debug Win32
솔루션 탐색기
솔루션 탐색기 검색
솔루션 'test' (1 프로젝트)
test
리소스 파일
소스 파일
test.cpp
외부 종속성
헤더 파일
test.cpp
(전역 범위) main()
1 #include <iostream>
2 using namespace std;
3
4 void main()
5 {
6     cout << "Hello!" << endl;
7     cout << "This is my first C++ program!" << endl;
8 }
100 %
출력
출력 보기 선택(S): 빌드
1>----- 빌드 시작: 프로젝트: test, 구성: Debug Win32 -----
1> test.cpp
1> test.vcxproj -> D:\Application_Data\Visual_C#\test\Debug\test.exe
===== 빌드: 성공 1, 실패 0, 최신 0, 생략 0 =====
빌드했습니다.
출: 1 열: 1 문자: 1 INS
```

# 프로그램 실행하기

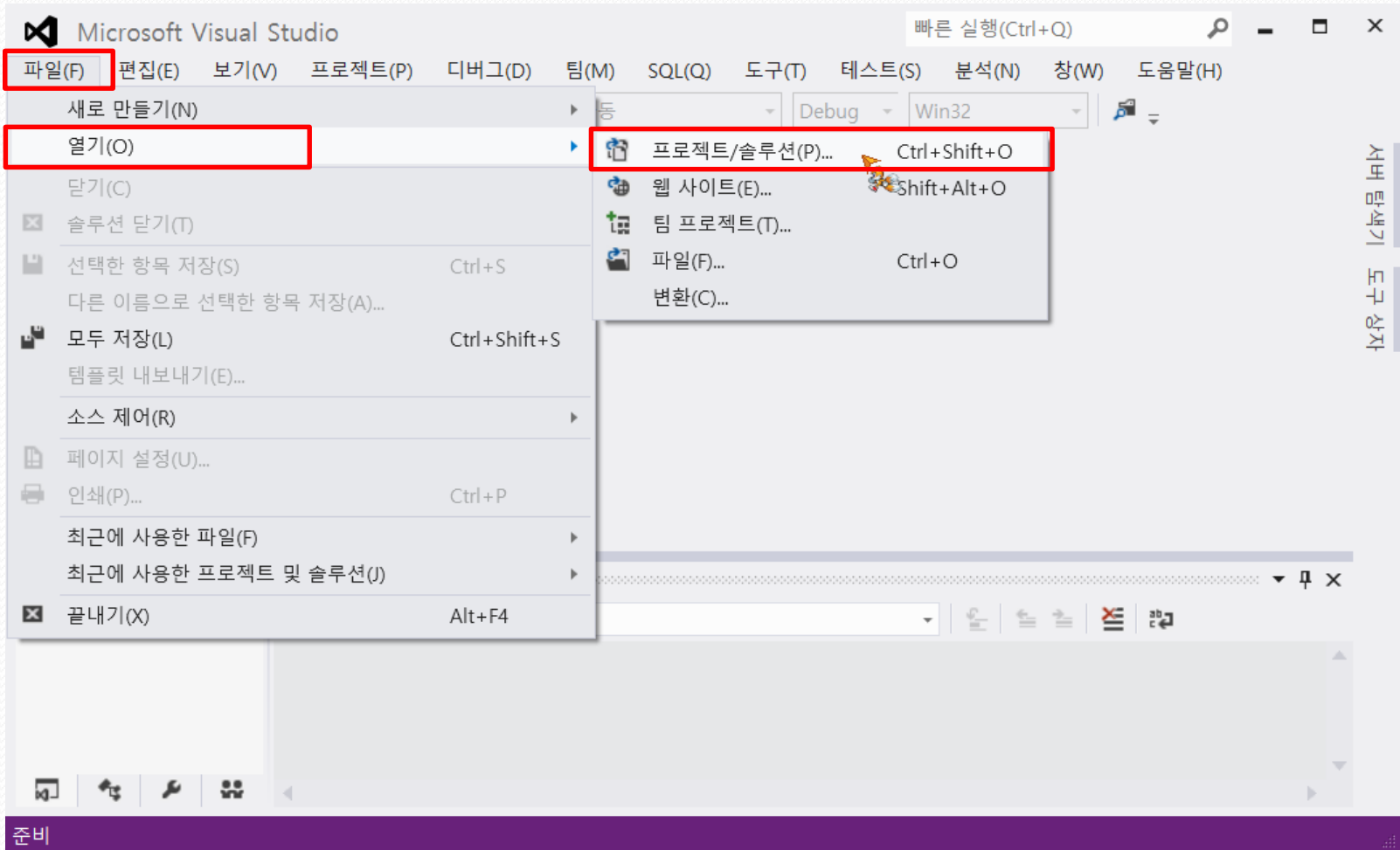
- [디버그] / [디버깅하지 않고 시작] ([Ctrl]+F5 )하면
  - 다음 그림과 같이 실행된다.



```
Hello!  
This is my first C++ program!  
계속하려면 아무 키나 누르십시오 . . .
```

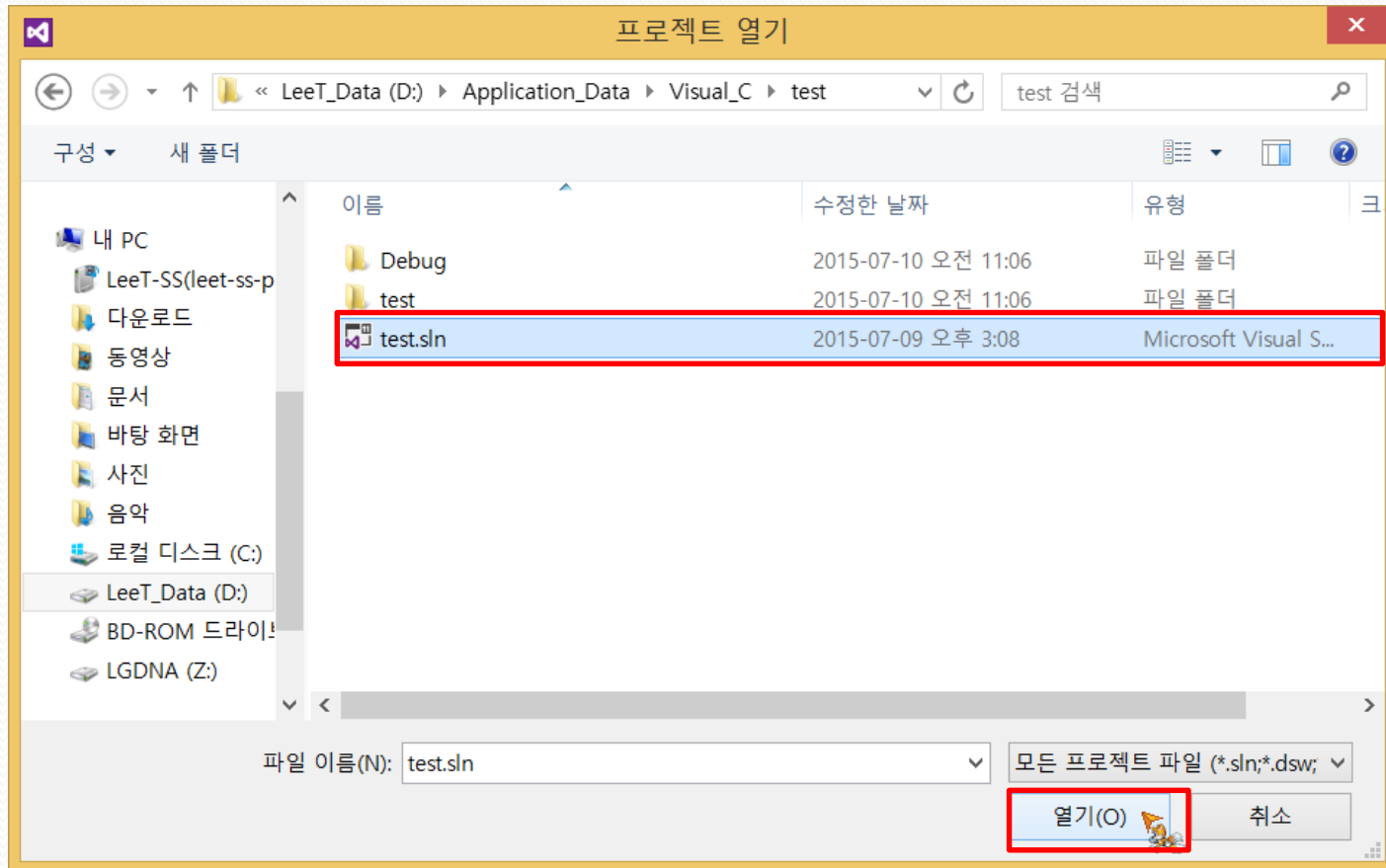
# 기존의 프로젝트 불러오기

- [파일] / [열기] / [프로젝트/솔루션...] 클릭



# 기존의 프로젝트 불러오기

- [프로젝트 열기] 대화상자에서 \*.sin 파일을 선택하고 [열기] 클릭



# C++ 프로그램 예제

- ex1\_1.cpp

```
01  #include <iostream>           // 헤더 파일을 포함한다.
02  using namespace std;          // std 이름 공간을 사용한다.
03
04  int main()                     // main() 함수
05  {
06      cout << "Hello!" << endl;
07      cout << "This is my first C++ program!\n";
08          // cout은 삽입 연산자(<<)를 사용하여 바이트 스트림을
09          // 표준 출력(모니터)으로 출력한다.
10
11      return 0;                  // 0을 반환한다.
12  }
```

```
Hello!
This is my first C++ program!
계속하려면 아무 키나 누르십시오 . . .
```

# C++ 프로그램의 디버깅

- 프로그램 오류
  - 경고(warning)
    - 무시해도 되지만
    - 잘못된 부분이 있어 발생
      - 알고 있는 경고가 아니면 코드를 고치는 것이 좋다
  - 오류(error)
    - 컴파일 오류 - 문법 오류
    - 링크 오류
      - 오류 메시지 출력, 쉽게 고칠 수 있다.
    - 실행시간 오류
    - 논리적인 오류
      - 오류 메시지 없음, 특정한 경우에만 나타나는 경우도 있음.
      - 프로그램을 자세히 검토해야 함.

# 디버거 사용하기

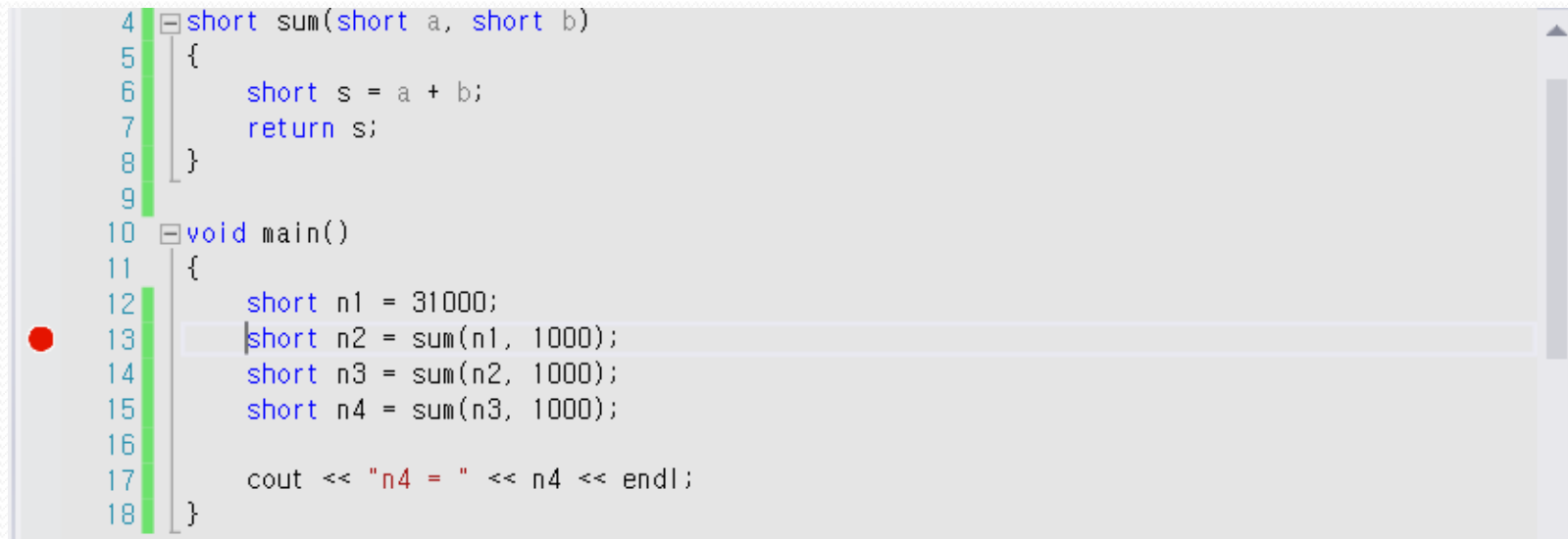
- ex1\_2.cpp (1)

```
01  #include <iostream>
02  using namespace std;
03  short sum(short a, short b)
04  {
05      short s = a + b;
06      return s;
07  }
08  void main()
09  {
10      short n1 = 31000;
11      short n2 = sum(n1, 1000);
12      short n3 = sum(n2, 1000);
13      short n4 = sum(n3, 1000);
14      cout << "n4 = " << n4 << endl;
15  }
16
17
18
```

```
n4 = -31536
계속하려면 아무 키나 누르십시오 . . .
```

# 디버거 사용하기

- ex1\_2.cpp (2)
  - 실행 결과
    - 예상 : n4 = 34000
    - 실제 : n4 = -31536
  - 13번째 줄부터 디버깅하기 위해 중단점을 13번 줄에 위치한다.
    - 마우스 커서를 13번 줄에 위치
    - [디버그] / [중단점 설정/해제 (F9)] 메뉴 선택



```
4 short sum(short a, short b)
5 {
6     short s = a + b;
7     return s;
8 }
9
10 void main()
11 {
12     short n1 = 31000;
13     short n2 = sum(n1, 1000);
14     short n3 = sum(n2, 1000);
15     short n4 = sum(n3, 1000);
16
17     cout << "n4 = " << n4 << endl;
18 }
```

The screenshot shows a C++ code editor with a debugger. A red dot, representing a breakpoint, is placed on the left margin next to line 13. The code defines a `sum` function and a `main` function. In `main`, variables `n1`, `n2`, `n3`, and `n4` are declared as `short` and their values are calculated using the `sum` function. The final value of `n4` is printed to the console.



# 디버거 사용하기

- ex1\_2.cpp (3)
  - [디버그] / [디버깅 시작 (F5)] 메뉴를 실행
    - 노란색 화살표가 중단점(13번 줄) 위에 나타난다.  
→ 12번 줄까지 실행하였고 현재(13번) 줄을 실행할 차례라는 의미
  - 자동 창
    - n1 : 31000 → 올바른 값
    - n2 : -13108 → 아직 실행하지 않은 상태이므로 쓰레기 값

```
10 void main()
11 {
12     short n1 = 31000;
13     short n2 = sum(n1, 1000);
14     short n3 = sum(n2, 1000);
15     short n4 = sum(n3, 1000);
16 }
```

100 %

이름	값	형식
n1	31000	short
n2	-13108	short

이름	번호
Ex01_01.exe!main() 줄 13	C++
Ex01_01.exe!_tmainCRTStartup() 줄 536	C

# 디버거 사용하기

- ex1\_2.cpp (4)
  - [디버거] / [한 단계씩 코드 실행 (F11)] 을 누르면 노란 화살표가 있는 현재 줄을 실행하고 다음 문장(sum() 함수 내로 진입)으로 넘어간다.
  - [디버거] / [프로시저 단위 실행 (F10)] 을 누르면 sum() 함수를 실행한 후 다음 줄(14번 줄)로 넘어간다.
- [F10]을 두 번 눌러 15번 줄로 가면
  - n1이 31000
  - n2가 32000,
  - n3이 -32536 으로 된다.
  - 14번 줄에서 문제가 있다는 것을 알 수 있다.

# 디버거 사용하기

- ex1\_2.cpp (5)

```
4 short sum(short a, short b)
5 {
6     short s = a + b;
7     return s;
8 }
9
10 void main()
11 {
12     short n1 = 31000;
13     short n2 = sum(n1, 1000);
14     short n3 = sum(n2, 1000);
15     short n4 = sum(n3, 1000);
16
17     cout << "n4 = " << n4 << endl;
18 }
```

100 %

이름	값	형식
sum이(가) 받:	-32536	short
n1	31000	short
n2	32000	short
n3	-32536	short
n4	-13108	short

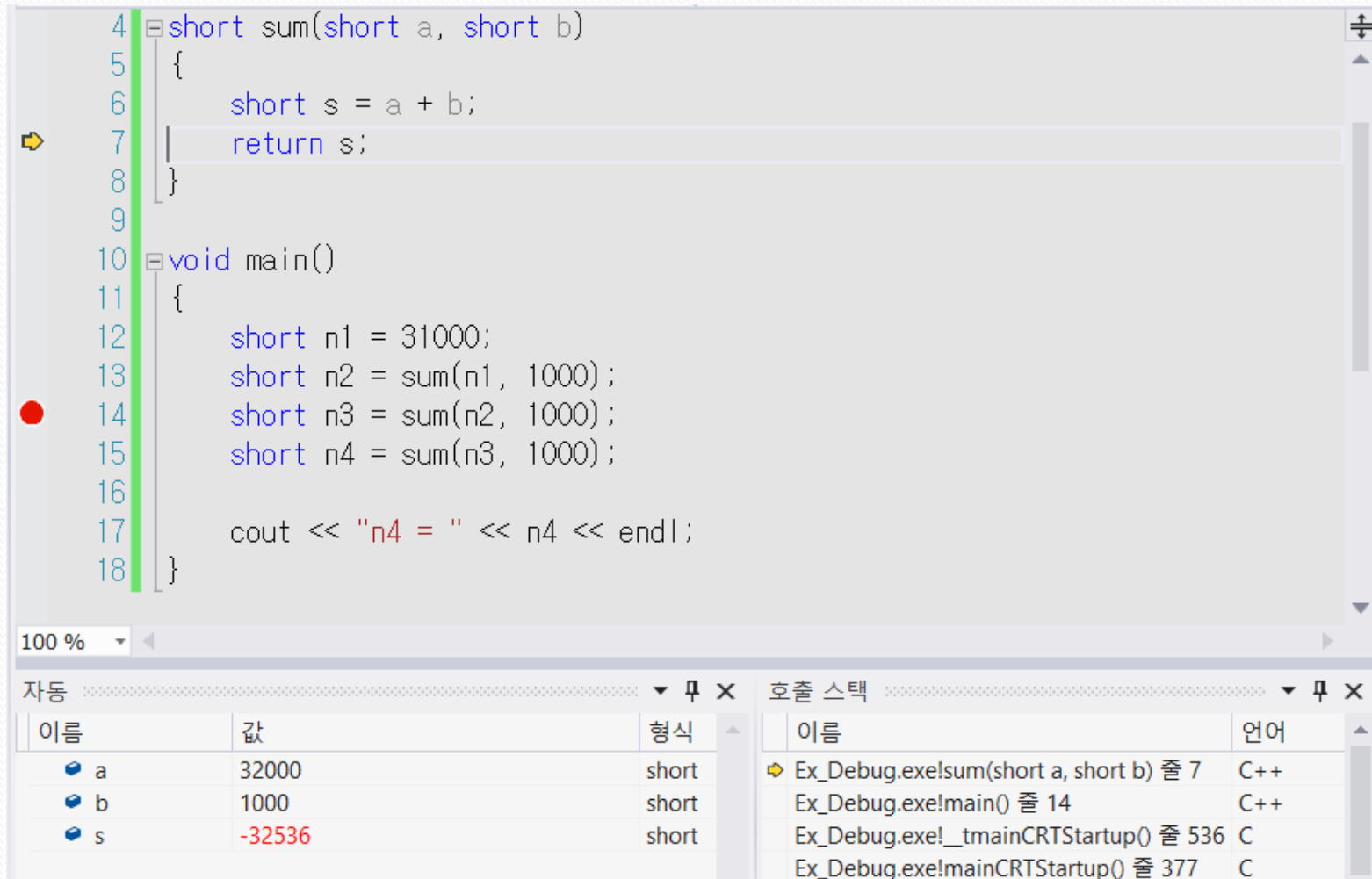
이름	언어
Ex_Debug.exe!main() 줄 15	C++
Ex_Debug.exe!_tmainCRTStartup() 줄 536	C
Ex_Debug.exe!mainCRTStartup() 줄 377	C
kernel32.dll!75097c04()	알 수 없음
[아래 프레임은 올바르지 않거나 누락되었]	
ntdll.dll!778bad1f()	알 수 없음

# 디버거 사용하기

- ex1\_2.cpp (6)
  - 이번에는 중단점을 14번 줄에 두고 디버그를 시작한 후,
  - [F11]을 눌러 sum() 함수 내로 들어간 후 [F10]을 두 번 눌러 7번 줄로 가면
    - a가 32000
    - b가 1000
    - s(두 값을 합한 값)는 -32536으로 된다.

# 디버거 사용하기

- ex1\_2.cpp (7)



```
4 short sum(short a, short b)
5 {
6     short s = a + b;
7     return s;
8 }
9
10 void main()
11 {
12     short n1 = 31000;
13     short n2 = sum(n1, 1000);
14     short n3 = sum(n2, 1000);
15     short n4 = sum(n3, 1000);
16
17     cout << "n4 = " << n4 << endl;
18 }
```

100 %

자동			호출 스택	
이름	값	형식	이름	언어
a	32000	short	Ex_Debug.exe!sum(short a, short b) 줄 7	C++
b	1000	short	Ex_Debug.exe!main() 줄 14	C++
s	-32536	short	Ex_Debug.exe!_tmainCRTStartup() 줄 536	C
			Ex_Debug.exe!mainCRTStartup() 줄 377	C

# 디버거 사용하기

- ex1\_2.cpp의 수정
  - short int의 기억 범위 : -32769 ~ 32767
    - 32000 는 정상적으로 나타낼 수 있다.
    - 33000 은 나타낼 수 있는 값의 범위를 넘었으므로 오류가 발생
  - 프로그램 수정
    - 변수 n1, n2, n3, n4, a, b, s와 함수 sum()의 반환 자료형을 int 자료형으로 수정

# 디버거 사용하기

- 수정한 ex1\_2.cpp 프로그램

```
01 #include <iostream>
02 using namespace std;
03
04 short sum(int a, int b)
05 {
06     int s = a + b;
07     return s;
08 }
09
10 void main()
11 {
12     int n1 = 31000;
13     int n2 = sum(n1, 1000);
14     int n3 = sum(n2, 1000);
15     int n4 = sum(n3, 1000);
16
17     cout << "n4 = " << n4 << endl;
18 }
```

n4 = 34000

계속하려면 아무 키나 누르십시오 . . .

# C++ 프로그램 언어의 특징

- C 언어
  - C 언어는 절차적 언어(procedural language)로
  - 함수들의 집합으로 프로그래밍 한다.
- C++ 언어
  - 객체지향 언어(object-oriented language)로
  - 데이터와 함수들로 이루어진 클래스(class)의 집합으로 프로그래밍.
- C++의 특징
  - C 만큼 효율적이고 실행 속도가 빠르며 이식성도 좋은 프로그래밍 언어이다.
  - C++는 C와의 호환성을 기본으로 제공하고
  - C보다 더 나은 여러 가지 기능들도 제공한다.
  - 절차적 프로그래밍(procedural programming)을 할 수 있으면서,
  - 객체 지향 프로그래밍(object-oriented programming) 기능
  - 일반화 프로그래밍(generic programming) 기능
  - 예외 처리(exception handling) 프로그래밍 기능



# C++ 프로그램 언어의 특징

- C 언어와의 호환성
  - C를 기초로 하여 C++를 만들었다.
    - C 언어의 확장이라는 의미로 C++라고 하였다.
    - 표준 C에 맞게 작성한 대부분의 C 프로그램은 C++ 프로그램에서도 유효하다.
- C++는 새로운 키워드가 추가되어 있다.
  - C++에 추가로 제공된 키워드는 C++ 프로그램에서는 식별자로 사용할 수 없다.
- C++에서는 엄격하게 형 검사를 한다.
  - C에서는 유효한 문장이라도 C++에서는 컴파일 오류가 발생할 수도 있다.

# C++ 프로그램 언어의 특징

- C 언어보다 개선된 C++ 언어의 기능 (1)
  - 엄격한 형 검사를 한다.
  - 참(true)과 거짓(false)을 나타내는 논리(bool)형을 제공한다.
  - 필요한 위치에서 변수 선언을 하여 사용할 수 있다.
  - 변수의 종류로 값 변수와 포인터 변수 이외에 참조(reference) 변수가 새로 도입되었다.
  - 동적 메모리 할당과 해제를 위하여 new와 delete 연산자를 제공한다.
  - 함수 호출 시에 발생하는 오버헤드를 줄이기 위해서 인라인 함수 (inline function)를 사용할 수 있다.
  - 함수의 인자를 디폴트(default) 값으로 지정할 수 있으며, 함수를 호출할 때 인자를 생략하면 자동적으로 디폴트 값이 사용된다.

# C++ 프로그램 언어의 특징

- C 언어보다 개선된 C++ 언어의 기능 (2)
  - 함수 중복 정의(function overloading) 기능을 제공한다.
    - 함수 중복 정의란 인자의 개수나 데이터 형이 다르면 같은 이름의 함수를 중복하여 정의할 수 있는 것을 말한다.
    - C에서는 비슷한 기능을 하는 함수이더라도 모든 함수의 이름을 다르게 정의하여야 한다.
    - C++에서는 같은 이름을 사용하여 함수를 여러 번 정의하여 사용할 수 있다. 또한 연산자도 중복 정의할 수 있다.
  - 이름 공간(namespace)을 사용한다.
    - C++에서는 이름 공간(namespace)을 사용하여 같은 이름의 식별자(identifier)를 중복하여 정의하고 구분하여 사용할 수 있다.
    - C에서는 같은 이름의 식별자(변수, 함수, 사용자 정의형, typedef 등의 이름)를 중복해서 사용할 수 없다.

# 프로그래밍 기법

- 절차적 프로그래밍

- 절차적 프로그래밍(procedural programming)은 문제 해결을 위한 처리 방법과 절차를 중요시한다.
  - 문제 해결을 위한 데이터의 처리 방법과 절차를 알고리즘(algorithm)이라고 한다.
- 구조화 프로그래밍(structural programming)
  - 정형화 된 분기문만을 사용하도록 제한된다. 따라서 프로그램의 이해와 수정이 쉬운 프로그램을 작성할 수 있다.
- 하향식(top-down) 설계
  - 구조화 프로그래밍은 규모가 큰 프로그램을, 쉽게 해결할 수 있는 독립적인 작은 단위(모듈)로 쪼개어 나가는 방법을 사용한다.
- C 언어는 구조화 프로그래밍으로 정형화된 분기문만을 사용하고, 하향식 설계 방법으로 개별적인 작업을 수행하는 함수(function) 단위로 프로그램을 개발할 수 있는 절차적 프로그래밍 언어이다.

# 프로그래밍 기법

- 객체 지향 프로그래밍
  - 구조화 프로그래밍 방법
    - 프로그램 작성을 간결하게 그리고 유지보수를 용이하게 할 수 있다.
    - 프로그램의 규모가 커지면 여전히 프로그램을 작성하기가 어려운 문제가 있다.
  - 객체 지향 프로그래밍 방법은 데이터를 중요시한다.
    - 해결해야 할 문제에 맞게 데이터 형 자체를 설계한다. 데이터 형을 나타내는 데 사용하는 도구가 클래스(class)이다.
    - 클래스에는 문제 해결에 필요한 데이터와 이들 데이터가 처리되어야 할 처리방법 즉, 함수도 포함되어 있다.
    - 클래스는 문제에 맞게 설계된 데이터 형의 개념이고 객체(object)는 클래스에 의해 만들어지는 변수와 같은 개념이다.
  - 상향식(bottom-up) 프로그래밍
    - 저수준의 클래스를 먼저 설계한 후에 고수준의 클래스 설계로 진행하여 프로그램을 만든다.

# 프로그래밍 기법

- 객체 지향 프로그래밍의 특징 (1)
  - 캡슐화(encapsulation)
    - 데이터와 동작을 하나의 클래스로 묶는 것을 캡슐화(encapsulation)이라고 하고 캡슐화를 통해 데이터에 대한 불필요한 접근을 막아 데이터를 보호할 수가 있는데 이를 정보 은닉이라고 한다.
  - 상속성(inheritance)
    - 객체 지향 프로그래밍에서는 클래스간의 상속(inheritance)을 통해서 이미 존재하고 있는 클래스를 가져다가 수정하여 새로운 클래스를 손쉽게 만들 수 있다.
  - 다형성(polymorphism)
    - 함수나 연산자를 중복 정의할 수 있어 프로그램이 상황에 따라 적당한 함수나 연산자를 자동적으로 선택할 수 있으므로, 같은 함수나 연산자를 사용하더라도 서로 다른 동작을 수행할 수 있다.

# 프로그래밍 기법

- 객체 지향 프로그래밍의 특징 (2)
  - C++ 언어는 객체 지향 프로그래밍 언어이다.
    - 캡슐화, 상속성, 다형성의 세 가지 특징을 통하여 코드를 재사용할 수 있고 클래스를 부품으로 사용할 수 있다.
    - 이미 만들어져 있는 검증된 클래스를 가져와서 새로운 프로그램에 사용할 수 있으므로, 좋은 프로그램을 훨씬 쉽고 빠르게 개발할 수 있다.
    - 많은 소프트웨어 개발 회사들이 다양한 클래스 라이브러리를 개발하여 판매하고 있다.

# 프로그래밍 기법

- 일반화 프로그래밍
  - 객체 지향 프로그래밍은 클래스를 재사용하고 부품으로 사용하는 데이터 측면을 중요시하는 방법임에 반하여,
  - 일반화 프로그래밍 (generic programming)은 자료형(데이터 형)과 무관한 코드를 작성하고 컴파일 시에 특정한 자료형으로 지정되는 것으로 알고리즘 측면을 중요시하는 프로그래밍 방법이다.
- C++는 템플릿(template) 기능과 STL(standard template library)을 사용하여 일반화 프로그래밍 방법을 지원하고 있다.
  - 함수와 클래스를 중복 작성할 필요가 없다.



# C++ 언어로 프로그램 하기

- C++ 프로그램 (1)

- 함수

- 기능별로 함수를 만들어 두고 이 함수를 호출하여 사용한다.
    - 함수는 실행 후 반환하는 자료형을 지정하여야 한다. 함수의 반환 자료형이 없을 때는 void로 지정하여야 한다.
    - main() 함수는 프로그램이 시작하고 정상적인 경우 종료하는 함수로, 반드시 있어야 하고 또한 하나만 존재하여야 한다.

- 입출력

- cout 출력을 담당하는 객체로, 출력하기 위해 스트림 삽입 연산자 (stream insertion operator)인 <<을 사용한다. <<은 오른쪽에 있는 내용을 출력 스트림 객체인 cout에게로 전달한다.
    - cin은 스트림 추출 연산자(stream extraction operator)인 >>을 사용한다. >>은 키보드로부터 입력을 받아 >> 오른쪽에 있는 변수에게로 값을 전달한다.
    - cin, cout 객체는 iostream 라이브러리에 선언되어 있으므로 입출력을 사용하기 위해서는 iostream 라이브러리를 포함(include)하여야 한다.

# C++ 언어로 프로그램 하기

- C++ 프로그램 (2)

- 주석

- 주석(comment)은 프로그램의 유지보수에 사용하기 위하여 프로그램 소스에 보충 설명 자료를 기술하는 문장이다.
    - 컴파일러는 주석을 무시하므로 프로그램 실행과는 무관하지만 유지보수 시에는 매우 중요한 역할을 하므로 프로그램 작성 시에 주석을 기술하는 습관을 가지도록 하여야 한다.
    - 한 줄 주석 : // 다음부터 그 줄의 끝까지 주석으로 처리한다.
    - 블록 주석 : /\* ~ \*/ 사이의 블록은 주석으로 처리한다.
    - C와 C++ 모두에서 //와 /\* ~ \*/를 사용할 수 있다.
  - 한 문장의 끝을 나타내기 위한 구분자
    - 세미콜론(;)은 한 문장의 끝을 나타내기 위한 구분자로 사용한다.
    - #include 문장과 같은 전처리 문장은 끝에 세미콜론을 사용하지 않는다.

# C++ 언어로 프로그램 하기

- C++ 프로그램 예 (ex1\_3.cpp)

```
01 // ex1_3.cpp
02 #include <iostream>
03 #include <string>
04 using namespace std;
05
06 void main()
07 {
08     int num;
09     char ch = 'A';
10     cout << "정수를 입력하시오: ";
11     cin >> num;
12     cout << "num : " << num << ", ch : " << ch << endl;
13
14     string str;
15     str = "Hello";
16     str = str + "!";
17     cout << str << endl;
18 }
```

```
정수를 입력하시오: 25
num : 25, ch : A
Hello!
계속하려면 아무 키나 누르십시오 . . .
```

# C++ 언어로 프로그램 하기

- C 프로그램 예 (ex1\_4.c)

```
01  /* ex1_4.c */
02  #include <stdio.h>
03  #include <string.h>
04
05  void main()
06  {
07      int num;
08      char ch = 'A';
09      char str[10];
10
11      printf("정수를 입력하시오: ");
12      scanf("%d", &num);          // scanf_s("%d", &num);
13      printf("num : %d, chr : %c\n", num, ch);
14
15      strcpy(str, "Hello");        // strcpy_s(str, 10, "Hello");
16      strcat(str, "!");           // strcat_s(str, 10, "!");
17      printf("%s\n", str);
18  }
```

```
정수를 입력하시오: 25
num : 25, ch : A
Hello!
계속하려면 아무 키나 누르십시오 . . .
```

# C++ 언어로 프로그램 하기

- 알기 쉬운 프로그램 만들기
  - 프로그램을 이해하기 쉽게 작성하여야 한다.
  - 개발하여 사용하기 시작할 때까지의 비용에 비해 유지보수에 드는 비용이 몇 배가 된다
- 이해하기 쉬운 프로그램
  - 문장 끝에는 ;을 붙여주어야 한다.
  - 한 문장이 끝났을 때 줄을 바꿔주면 프로그램이 이해하기 쉽고 유지보수가 쉬워진다.
  - 빈 줄 넣기 : 연관성이 있는 문장들을 묶음으로 하여준다.
  - 단락 맞추기(indentation) :블록 내부를 들여쓰기 하여 블록의 범위를 명확하게 알 수 있게 한다.
  - 식별자에 의미 있는 이름 붙이기
  - 주석(comment) : 설명을 달아두는 습관을 가진다.

# C++ 배우기

- C 언어를 잘 알아야 한다.
  - 그러나 이미 C 언어를 알고 있다면 C++의 이점을 활용하기 위해서는 C 프로그래밍 습관을 바꾸어야 한다.
- C++는 배우는데 많은 노력과 시간이 필요하다.
  - C++ 언어에는 기능이 많다.
  - C++ 언어는 절차적 프로그래밍 방법, 클래스를 사용하는 객체 지향 프로그래밍 방법, 템플릿이 제공하는 일반화 프로그래밍 방법의 세 가지 프로그래밍 방법이 결합되어 있다.
  - C++ 프로그래밍에 익숙해지려면 많은 노력과 시간이 필요하지만,
  - 일단 익숙해지고 나면 좋은 프로그램을 쉽게 작성할 수 있다.

# C++ 배우기

- 이 책은 C 언어를 알고 있는 독자를 대상으로 C++ 언어를 공부한다.
  - 먼저 C 언어의 문법은 간단히 소개한다 : C 언어보다 개선된 C++의 기능을 C 언어와 비교하여 설명하면서 C 언어에 대한 기초를 다진다.
  - 다음으로 클래스, 상속, 다형성에 대해 설명하면서 객체 지향 프로그래밍을 배운다 → 많은 노력이 필요하다.
  - 그 다음으로 템플릿에 대해 설명하면서 일반화 프로그래밍을 배운다.
  - 마지막으로 예외 처리 프로그래밍에 대하여 공부한다.

# Question & Answer

Any Question?

Please.