원리를 알면 IT가 맛있다
IT COOKBOOK

IT CookBook, C++ 하이킹 객체지향과 만나는 여행

[강의교안 이용 안내]

C++ 하이킹

객체지향과 만나는 여행

Chapter 01. C++ 프로그래밍의 첫 걸음

한빛아카데미 Hanbit Academy, Inc.

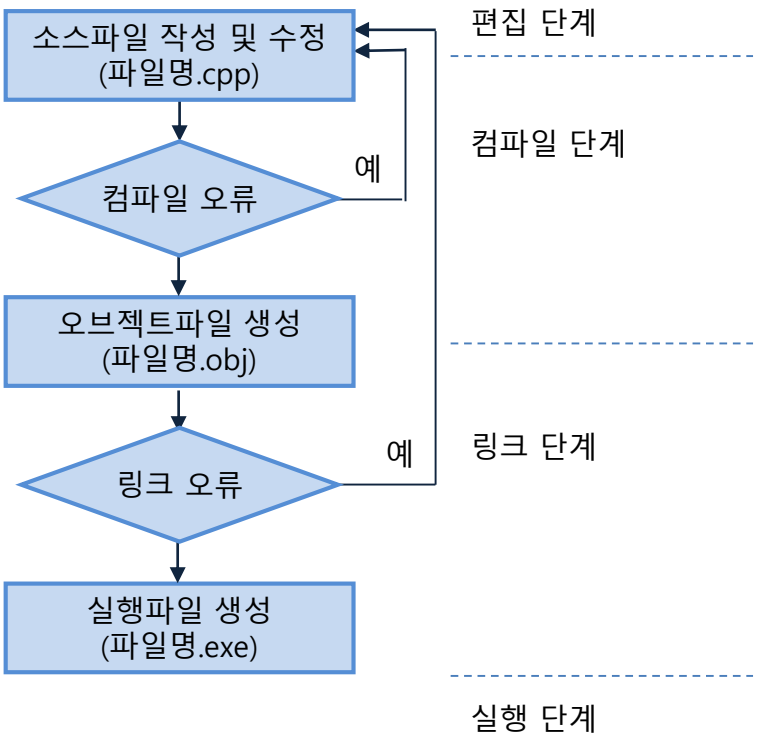
목차

학습목표

- C++로 프로그래밍한다는 것이 무엇인지에 대한 개념을 이해한다.
- C++를 이용해 간단한 프로그램의 작성 방법을 익힌다.
- C++ 프로그래밍의 기본 구조를 이해한다.
- 비주얼 스튜디오 2012의 기본 사용 방법을 익힌다.

C++로 프로그래밍한다는 것의 의미

- C++로 프로그래밍한다는 것은 컴퓨터로 원하는 작업을 할 수 있도록 C++를 도구로 사용해서 프로그램을 작성하는 것을 뜻한다.
- C++ 프로그램을 작성하는 과정은 크게 4단계인 편집(edit), 컴파일(compile), 링크(link), 실행(execute) 단계로 이루어진다.



[그림 1-1] C++ 프로그램 작성 단계

01. C++의 이해

편집단계

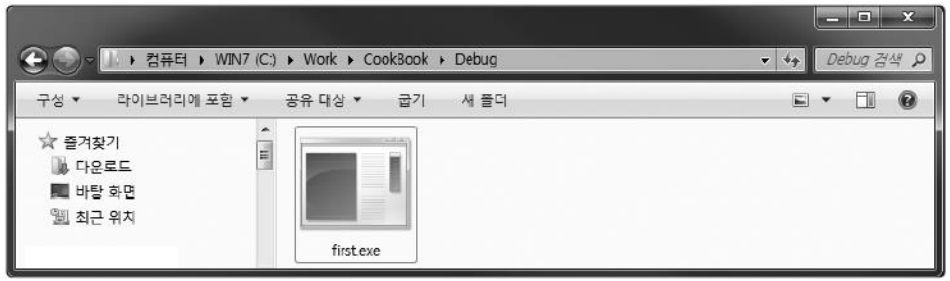
- 프로그래머가 C++의 문법에 맞게 프로그램을 작성한 파일을 소스파일이라고 하는데, 소스파일을 작성한 후에는 C++ 소스파일임을 의미하는 cpp 확장자를 꼭 붙여야 한다.

컴파일 단계

- 컴파일 단계에서는 편집 단계에서 작성한 소스파일이 C++ 컴파일러에 의해 문법에 맞는지를 검증한 후 문법적으로 오류가 없음을 확인하면 소스파일을 기계어 상태로 변경하여 오브젝트파일을 만든다.

링크 단계

- 프로그램에서 자주 사용되는 로직을 미리 정의해 제공하는 것을 라이브러리(library)라고 한다. 링크 단계에서는 라이브러리에서 제공하는 형식에 맞게 사용했는지를 검사한다. 올바르게 참조되었다면 컴파일 단계에서 만들어진 오브젝트파일을 실행 가능한 파일로 만들어 준다.



실행 단계

- 실행파일(확장자가 exe인 파일)은 컴퓨터에서 바로 실행할 수 있다. 이 실행파일을 콘솔창에서 입력하거나 비주얼 스튜디오에서 제공하는 툴을 이용해 실행시키는 단계가 바로 실행 단계다.

01. C++의 이해

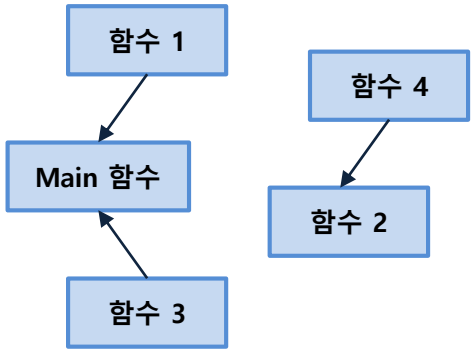
객체지향 언어인 C++

① 절차적 프로그래밍 : 프로그램이 함수의 집합으로 구성되므로 함수를 정의하면서 함수에 필요한 데이터를 선언하여 사용한다. 대표적인 언어로는 C가 있다.

② 객체지향 프로그래밍 : 객체를 지향하는 프로그램 방식이므로 객체를 생산하기 위한 클래스를 설계한 후에 이를 다룰 함수(사용자 인터페이스)를 정의하여 함수로 객체를 다루도록 한다. 대표적인 언어로는 C++, 자바, C# 등이 있다.

절차적 프로그래밍

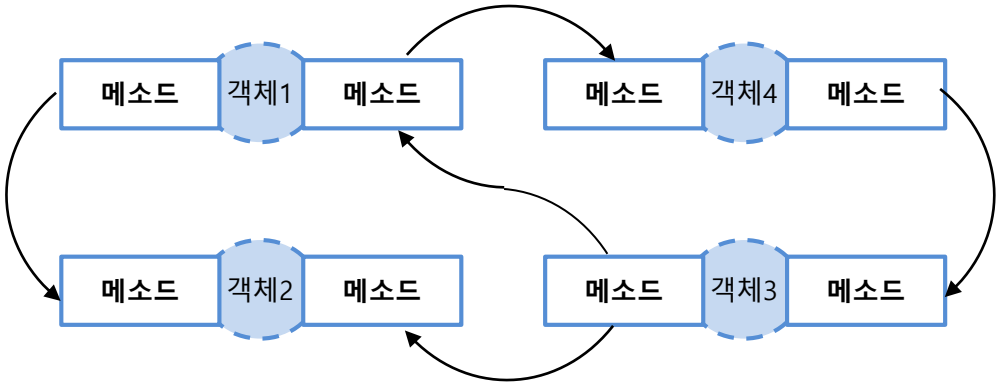
- 함수를 중심으로 프로그램을 설계한 후 함수에 필요한 데이터를 정의한다.



[그림 1-2] 절차적 프로그래밍

객체지향 프로그래밍

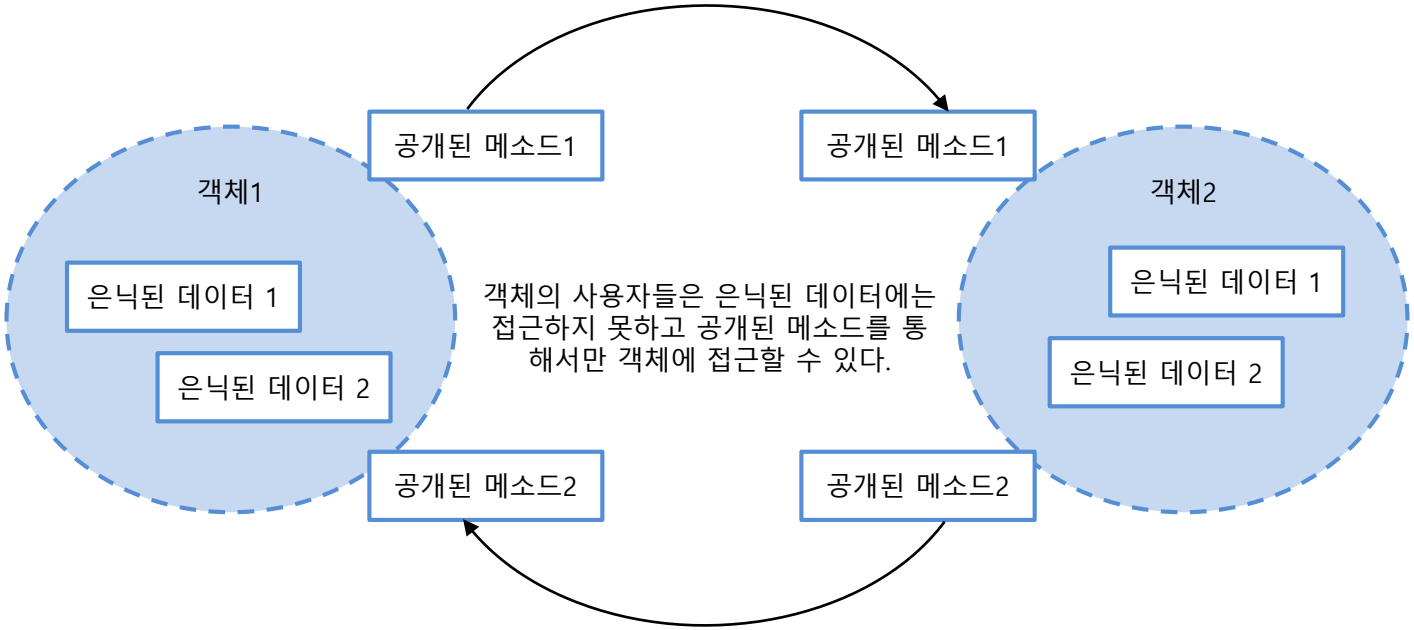
- 객체들로 이루어진 프로그램에서 사건이 일어날 때마다 그에 따른 처리를 하는 식으로 프로그램이 진행된다.



[그림 1-3] 객체지향 프로그래밍

객체지향 프로그래밍의 주요 특징

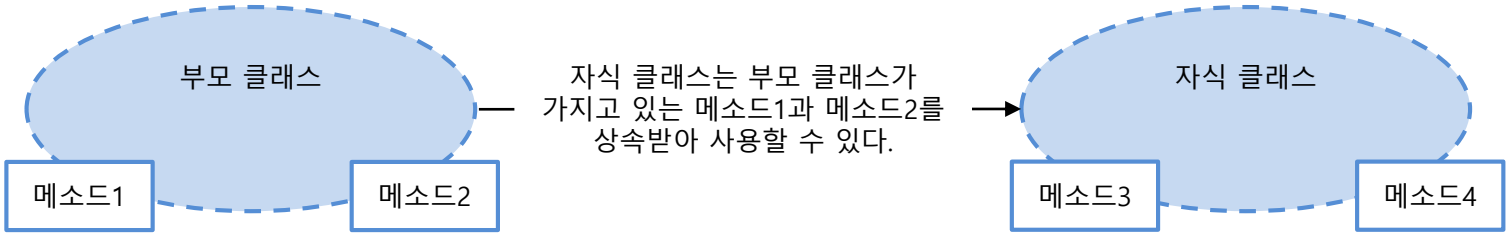
- 캡슐화와 데이터 은닉
  - 데이터를 직접 다루면 데이터가 손상될 수 있으므로 이를 방지하기 위해 제공되는 것이 캡슐화(encapsulation)와 데이터 은닉(data hiding)이다.



[그림 1-4] 객체의 은닉된 데이터와 공개된 함수

01. C++의 이해

- 다형성과 함수의 오버로딩, 연산자 오버로딩
  - 다형성(polymorphism)은 동일한 함수나 연산자를 자료에 따라 다르게 동작하도록 적용할 수 있음을 의미한다. 함수명은 동일하지만 매개변수의 자료형이나 개수를 서로 다르게 주어 함수명을 여러 번 정의할 수 있는데, 이를 함수의 오버로딩이라 한다. 이미 사용중인 연산자를 다른 용도로 다시 정의해서 사용하는 것을 연산자의 오버로딩이라고 한다.
- 상속성
  - 상속성(inheritance)은 객체지향의 가장 대표적인 특징으로, 특정 객체의 성격을 다른 객체가 상속받아 사용할 수 있도록 하는 것이다.

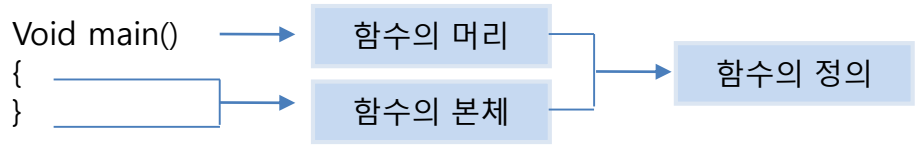


예제 1-1. 세상에서 가장 간단한 C++ 프로그램(first.cpp)

```
01 void main()
02 {
03 }
```

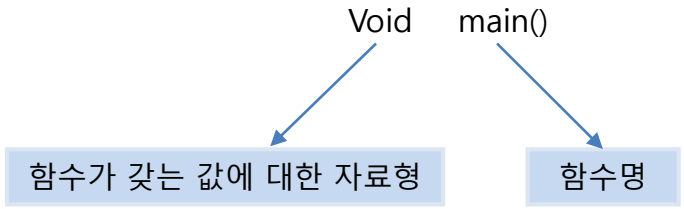
세상에서 가장 간단한 C++ 프로그램

- 프로그램에서 반드시 하나는 필요한 함수 main( )



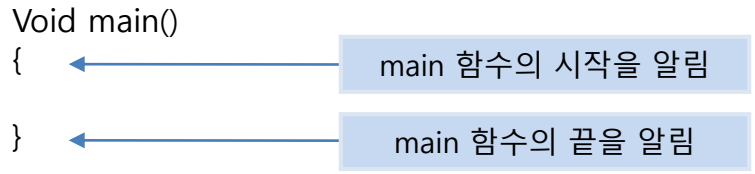
[그림 1-6] **함수의 구조**

- 함수가 값을 갖지 않도록 하는 자료형 void



[그림 1-7] **함수의 머리 구조**

- 함수의 시작과 끝을 위한 기호 { }



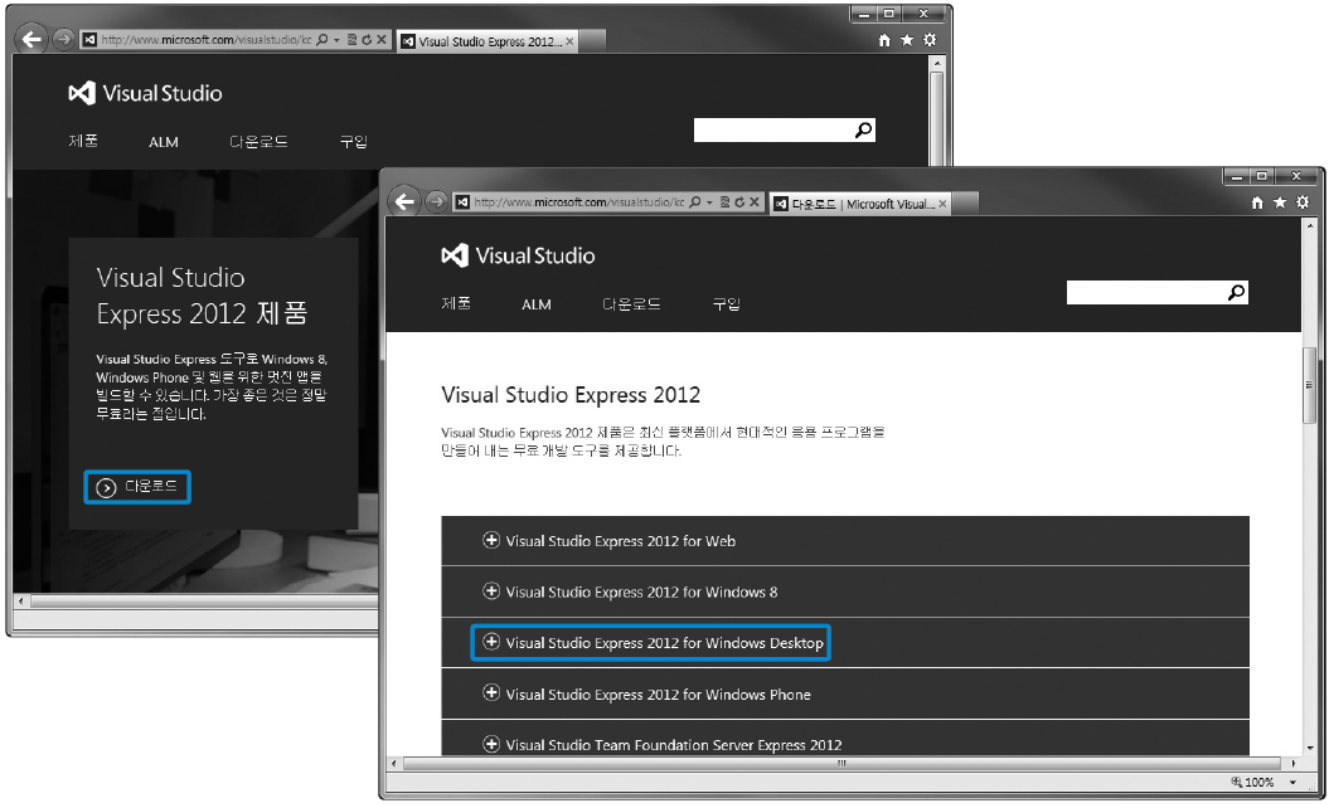
[그림 1-8] **함수의 시작과 끝**

02 비주얼 스튜디오 2012 맛보기

비주얼 스튜디오 2012 설치하기

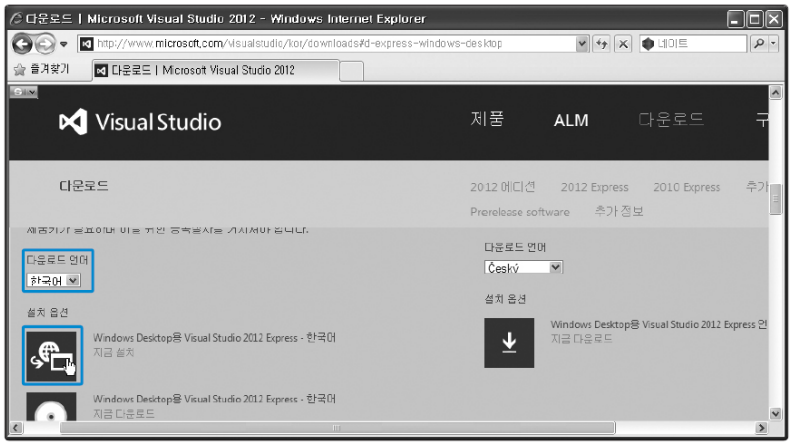
① 비주얼 스튜디오 2012 다운로드

(http://www.microsoft.com/visualstudio/kor/products/visual-studio-expressproducts)

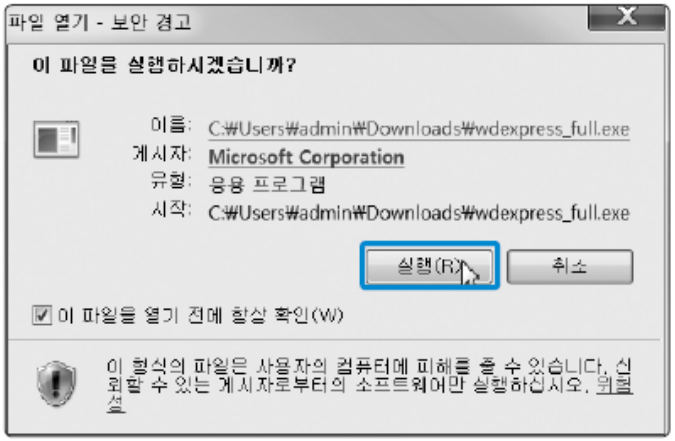


02 비주얼 스튜디오 2012 맛보기

② 실행 언어 선택

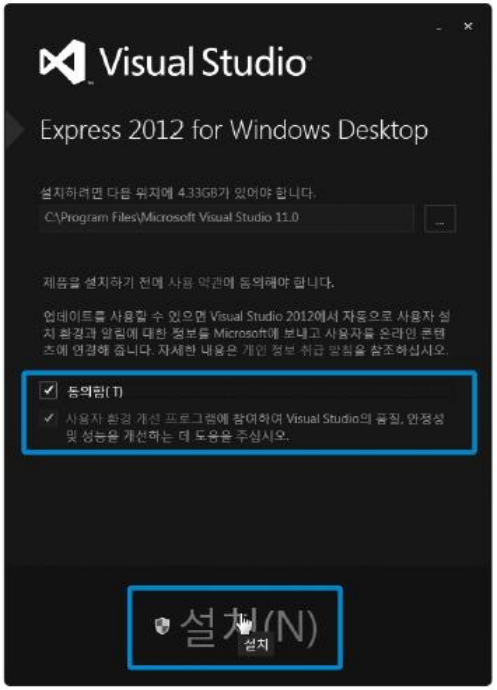


③ 실행



02 비주얼 스튜디오 2012 맛보기

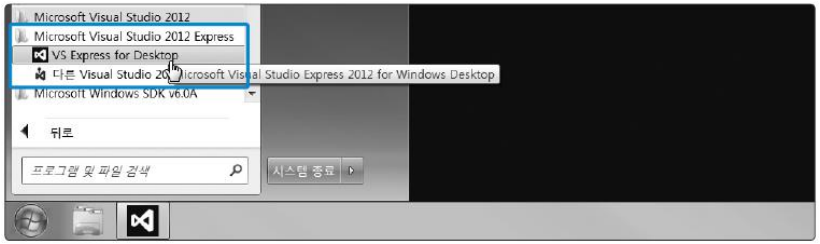
④ 설치와 시작



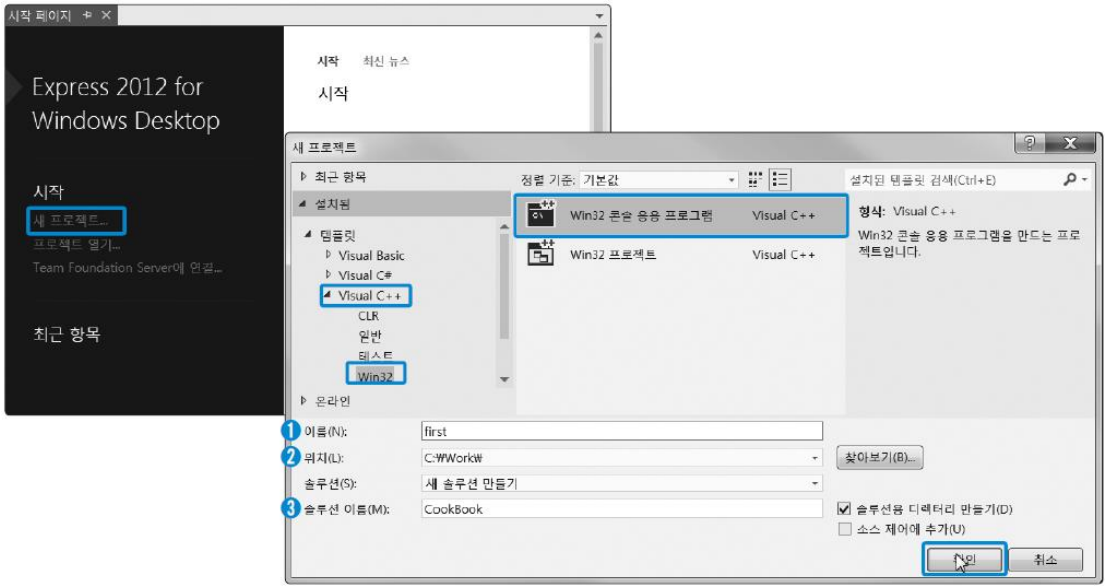
02 비주얼 스튜디오 2012 맛보기

비주얼 스튜디오로 C++ 프로그래밍하기

① 비주얼 스튜디오 실행

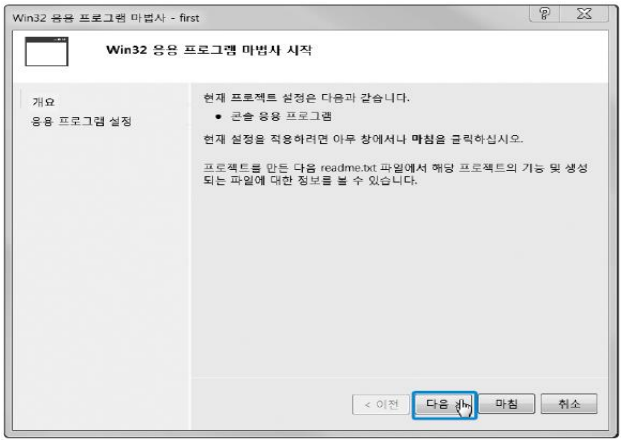


② 새 프로젝트 항목 설정

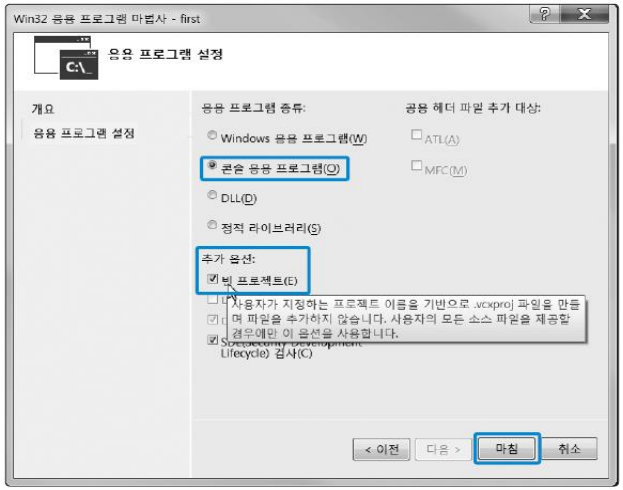


02 비주얼 스튜디오 2012 맛보기

③ Win32 응용 프로그램 마법사 시작

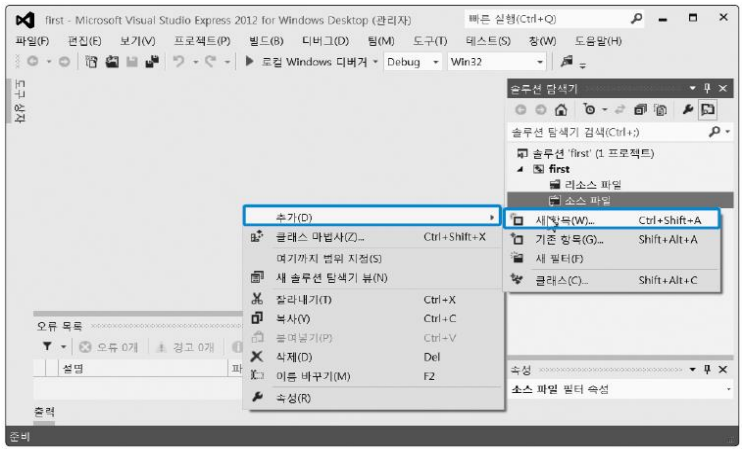


④ 응용 프로그램 설정

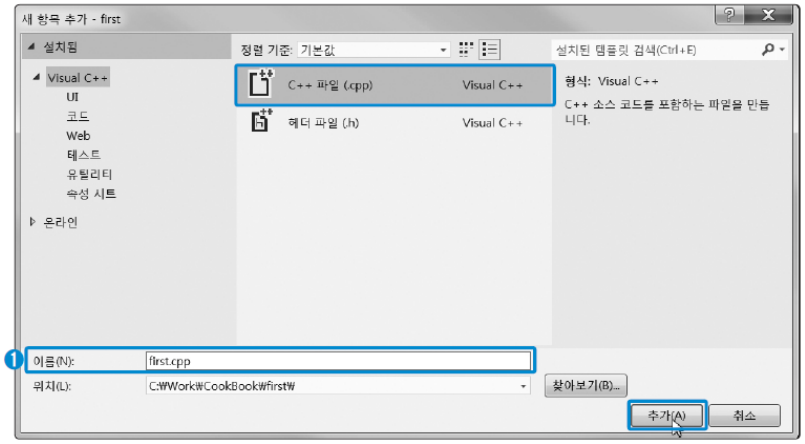


02 비주얼 스튜디오 2012 맛보기

⑤ 새 항목 추가



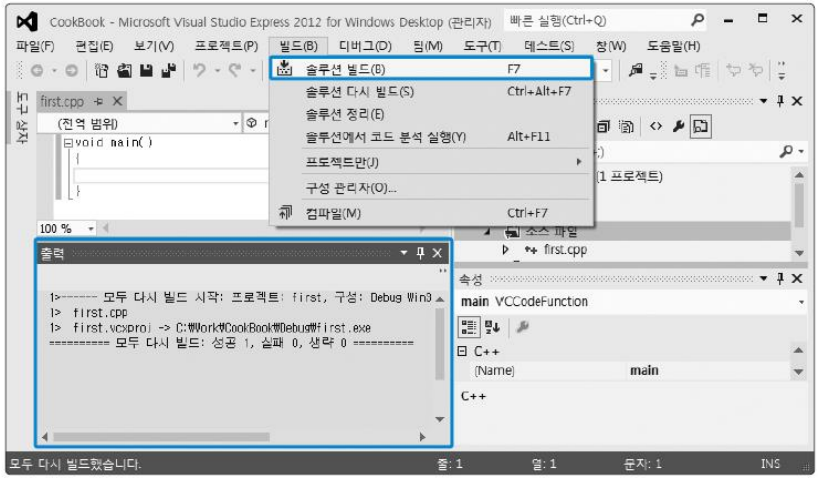
⑥ 프로젝트 C++ 소스파일 생성



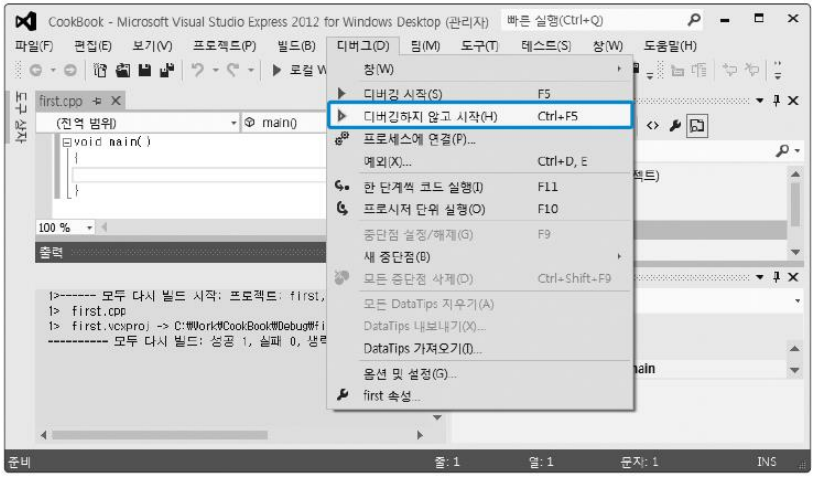

```
void main()
{
}
```

02 비주얼 스튜디오 2012 맛보기

⑦ 빌드(컴파일, 링크) 단계

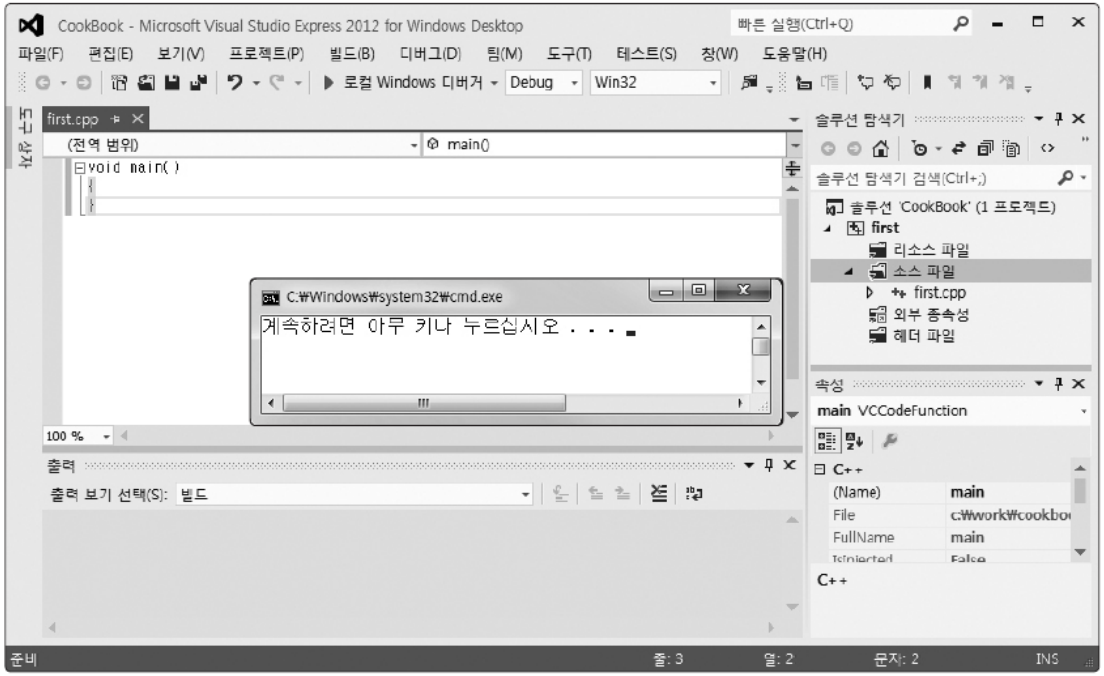


⑧ 실행 단계



02 비주얼 스튜디오 2012 맛보기

⑨ 실행 결과



예제 1-2. 간단한 문자열을 출력하는 프로그램(01_02.cpp)


```
01 #include <iostream> // 헤더파일을 포함시키는 문장
02 void main()
03 {
04 /* cout은 출력을 담당하는 객체로써,
05 스트림 삽입 연산자(stream insertion operator)인
06 <<를 이용해서 ""안에 있는 문자열을 출력한다. */
07
08 std::cout<<"C++ 세계에 오신 것을 환영합니다. \n";
09 }
```

03 간단한 출력을 하는 프로그램

외부 파일을 포함하는 #include문과 iostream 파일

- #include문 : 바로 뒤에 나오는 < > 기호 사이의 파일을 포함시키기 위해 사용한다.
- Iostream : io는 input(입력) 및 output(출력)을 말하며, 프로그램 소스코드 앞부분에 들어간다고 해서 헤더 파일이라고도 부른다.

std 네임 스페이스

- 이름이 속해 있는 공간이라는 의미다.

```
08    std::cout<<"C++ 세계에 오신 것을 환영합니다.\n";
```

- cout 객체가 속한 네임 스페이스
- 네임스페이스 std 소속 출력 객체

cout 객체와 출력 연산자 <<

- cout(console output)은 출력을 담당하는 객체이다. 출력을 위해 스트림 삽입 연산자(stream insertion operator)인 << 를 사용한다.

```
08    std::cout  <<  "C++ 세계에 오신 것을 환영합니다.\n";
```

- 출력을 담당하는 객체
- 출력을 위한 스트림 삽입 연산자
- 출력할 문자열을 << 연산자 다음에 기술함

행바꿈 기능 문자 \n

- <Enter> 키와 같은 역학을 한다.

03 간단한 출력을 하는 프로그램

문장의 종료 기호 ;

- 어디까지가 한 문장인지를 구분하기 위해서 세미콜론(;)을 사용한다.

```
08    std::cout<<"C++ 세계에 오신 것을 환영합니다.\n";
```

문장의 끝을 알리는 세미콜론

프로그래머의 이해를 돕는 주석문

- 주석(comment)은 소스코드의 설명을 보충하거나, 추가 자료를 기술한 문장이다.
  - 행 단위 주석 // : 한 행만 주석으로 처리할 때 사용한다.

```
01    #include <iostream> // 헤더파일을 포함시키는 문장
```

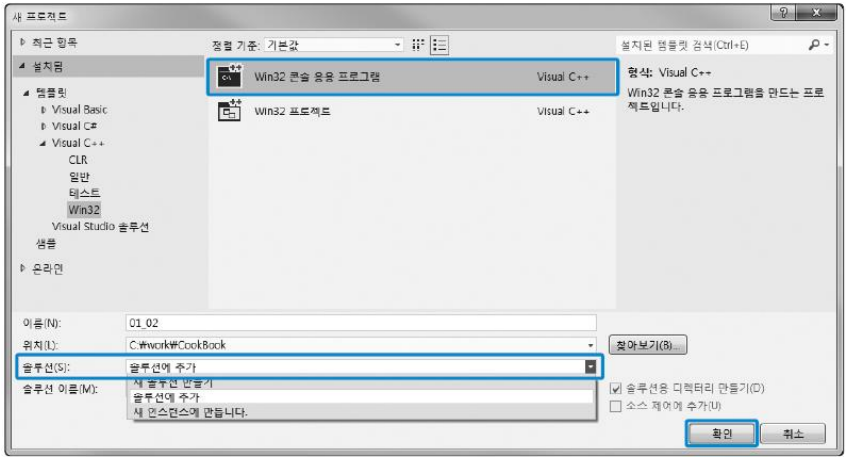
  - 블록 단위 주석 /* */ : 내부에 기술된 모든 문장이 주석으로 처리된다.

```
04    /* cout은 출력을 담당하는 객체로서
05    스트림 삽입 연산자(stream insertion operator)인
06    <<를 이용해서 ""내부의 문자열을 출력한다. */
```

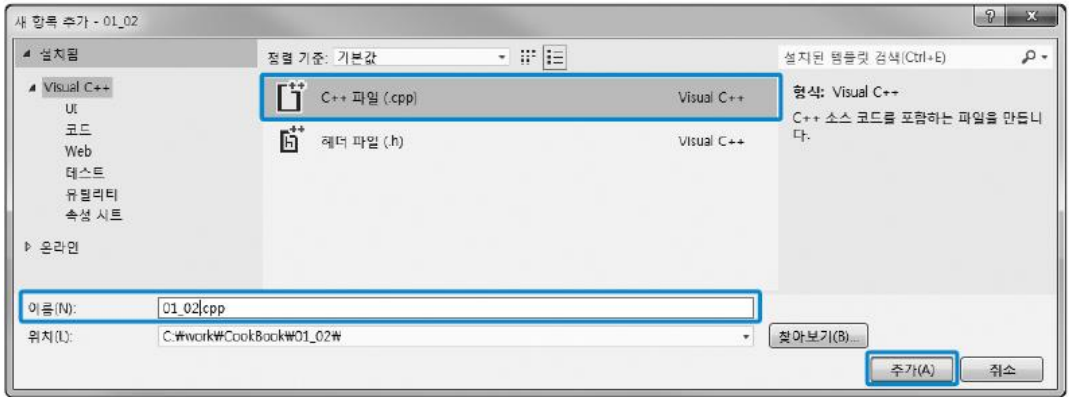
03 간단한 출력을 하는 프로그램

솔루션에 프로젝트 추가하기

① 프로젝트 생성



② 프로젝트에 C++ 소스 파일 생성

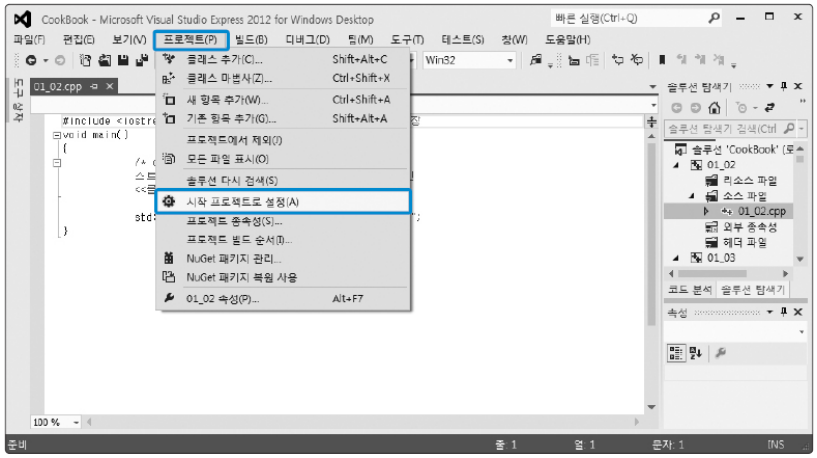


03 간단한 출력을 하는 프로그램

③ 소스코드 입력

```
01  #include <iostream> // 헤더파일을 포함시키는 문장
02  void main()
03  {
04  /* cout은 출력을 담당하는 객체로서
05  스트림 삽입 연산자(stream insertion operator)인
06  <<를 이용해서 ""내부의 문자열을 출력한다. */
07
08  std::cout<<"C++ 세계에 오신 것을 환영합니다. \ n";
09  }
```

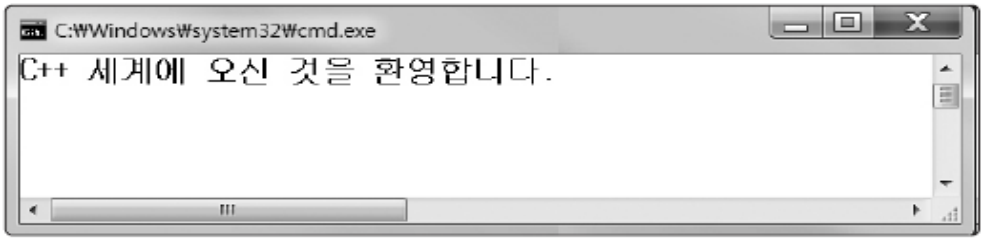
④ 시작 프로젝트로 설정



03 간단한 출력을 하는 프로그램

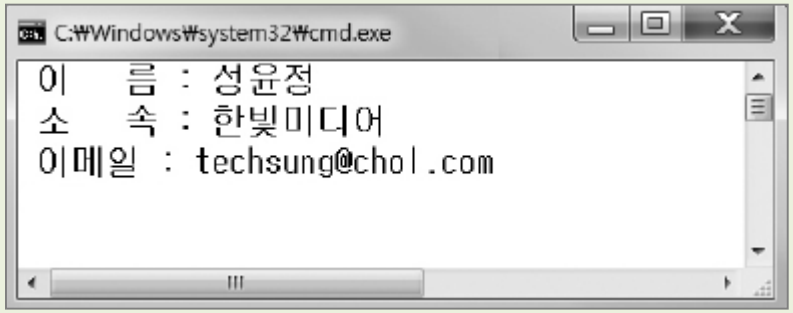
⑤ 실행 결과

[빌드] → [솔루션 빌드(F7) 메뉴를 실행 한 후 [디버그] → [디버깅하지 않고 시작(CTRL + F5) 메뉴를 선택해서 에러가 발생하지 않으면 실행 결과를 별도의 콘솔창에 출력한다.



예제 1-3. 여러 줄에 걸쳐서 여러 문장 출력하기(01_03.cpp)


```
01 #include <iostream> // 헤더파일을 포함시키는 문장
02 void main()
03 {
04 std::cout<<" 이 름 : 성윤정 "<<std::endl;
05 std::cout<<" 소 속 : 한빛미디어 "<<std::endl;
06 std::cout<<" 이메일 : techsung@chol.com "<<std::endl;
07 }
```



예제 1-4. using 구문으로 네임스페이스 사용하기(01_04.cpp)


```
01 #include <iostream> // 헤더파일을 포함시키는 문장
02 using namespace std; // 네임스페이스를 지정
03 void main()
04 {
05 cout<<" 이 름 : 성윤정 " <<endl;
06 cout<<" 소 속 : 한빛미디어 " <<endl;
07 cout<<" 이메일 : techsung@chol.com "<<endl;
08 }
```