

프로그래밍 언어 (실습)

**실습 1. (보충 설명)**  
**Visual Studio의 기본기능 안내,**  
**C 프로그램 작성 기본 및 Debugging 방법**



**김영탁, 황현동**

**영남대학교 기계IT 대학**  
**정보통신공학과**

(Tel : +82-53-810-3940; <http://antl.yu.ac.kr/>; E-mail : [ytkim@yu.ac.kr](mailto:ytkim@yu.ac.kr), [mch2d@ynu.ac.kr](mailto:mch2d@ynu.ac.kr) )

# Outline

- ◆ Visual Studio 소개
- ◆ Visual Studio 설치하기
- ◆ 솔루션과 프로젝트
- ◆ 프로젝트 생성/속성 설정
- ◆ 프로젝트 시작하기
- ◆ 기본적인 코드 구조
- ◆ 응용 프로그램 예
- ◆ 보편적인 에러상황
- ◆ Visual Studio Debugging tool 사용법
- ◆ 종합정리
- ◆ Homework

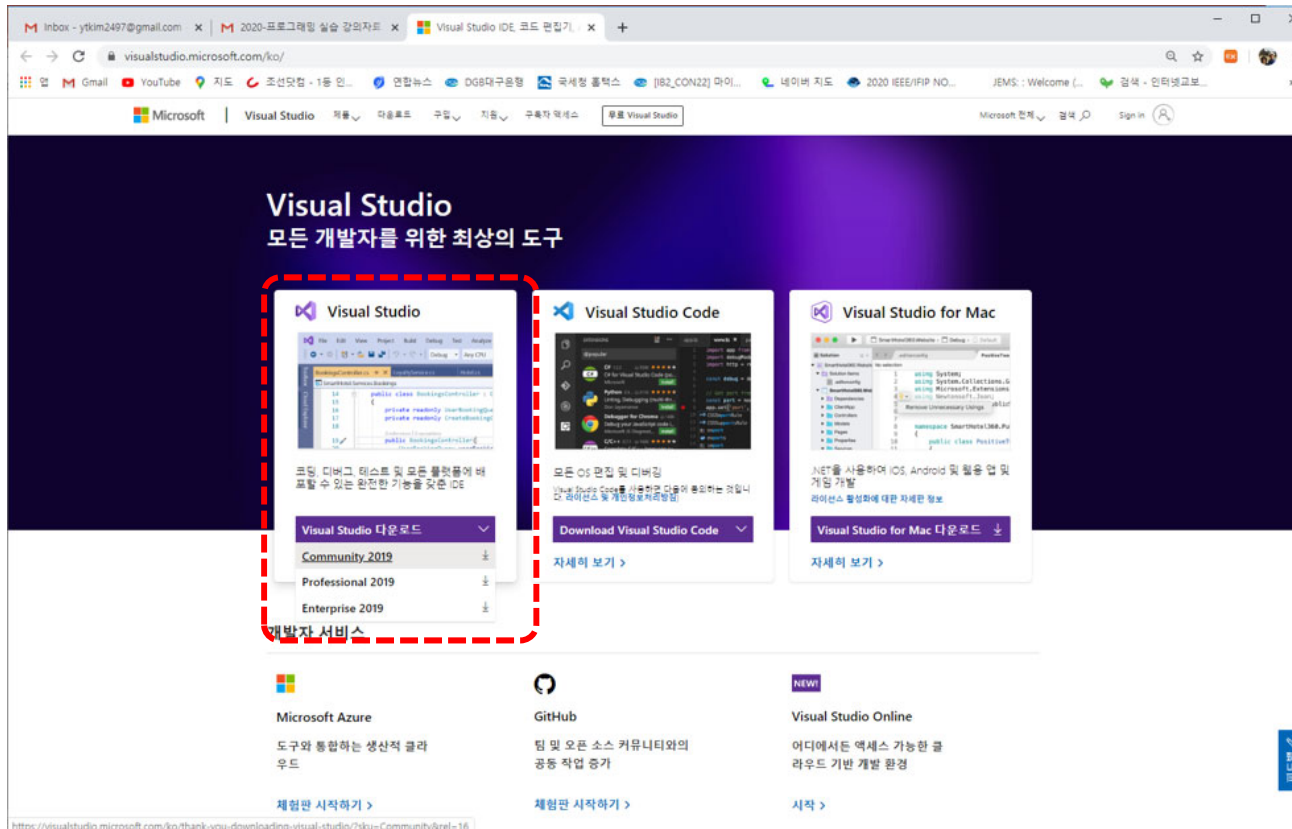


# **Visual Studio Community 2019**

# Visual Studio 소개

## ◆ Visual Studio란?

- Microsoft window 환경에서 다양한 언어를 프로그래밍할 수 있는 통합 개발환경
- C, C++, MFC GUI 등 다양한 프로그래밍 언어 지원
- <https://visualstudio.microsoft.com/ko/>



# Visual Studio 설치하기(1)

## ◆ Visual Studio Community 2019 다운로드

- <https://visualstudio.microsoft.com/ko/>
- Visual Studio 다운로드 -> Community 2019 선택

Microsoft | Visual Studio 제품 다운로드 구입 지원 구독자 액세스 무료 Visual Studio Microsoft 전체 검색 Sign in

## Visual Studio

모든 개발자를 위한 최상의 도구

### Visual Studio

코딩, 디버그, 테스트 및 모든 플랫폼에 배포할 수 있는 완전한 기능을 갖춘 IDE

Visual Studio 다운로드

- Community 2019
- Professional 2019
- Enterprise 2019

### Visual Studio Code

모든 OS 편집 및 디버깅

Visual Studio Code를 사용하면 다음에 동의하는 것입니다. (라이선스 및 개인정보처리방침)

Download Visual Studio Code

자세히 보기 >

### Visual Studio for Mac

.NET을 사용하여 iOS, Android 및 웹 앱 및 게임 개발

라이선스 활성화에 대한 자세한 정보

Visual Studio for Mac 다운로드

자세히 보기 >

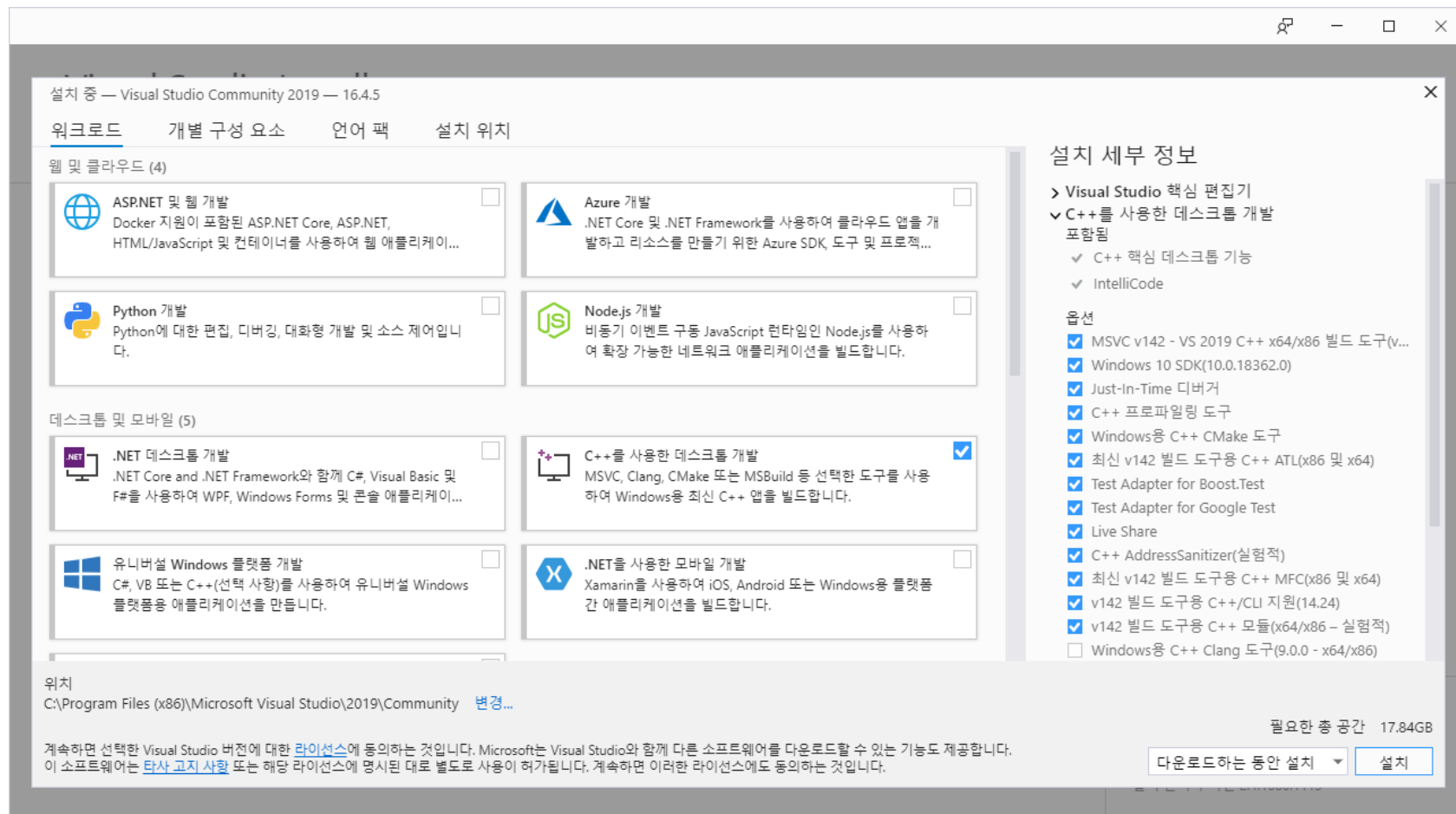
개발자 서비스



# Visual Studio 설치하기(2)

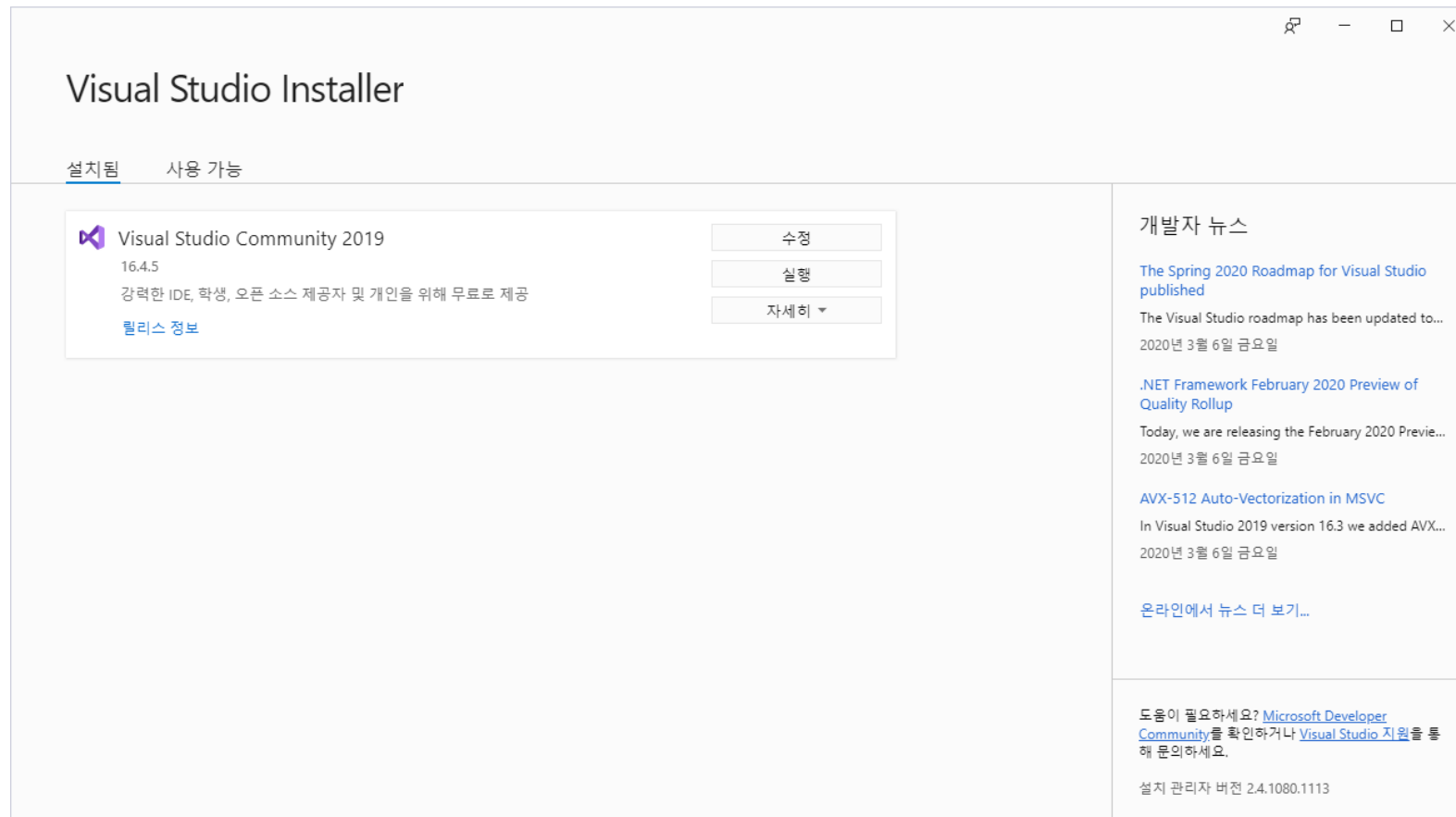
## ◆ Visual Studio Installer 설치 및 실행

- C++를 사용한 데스크톱 개발 워크로드 체크
  - Hard Disk 용량 20GB 확보 후 설치



# Visual Studio 설치하기(3)

## ◆ 설치완료



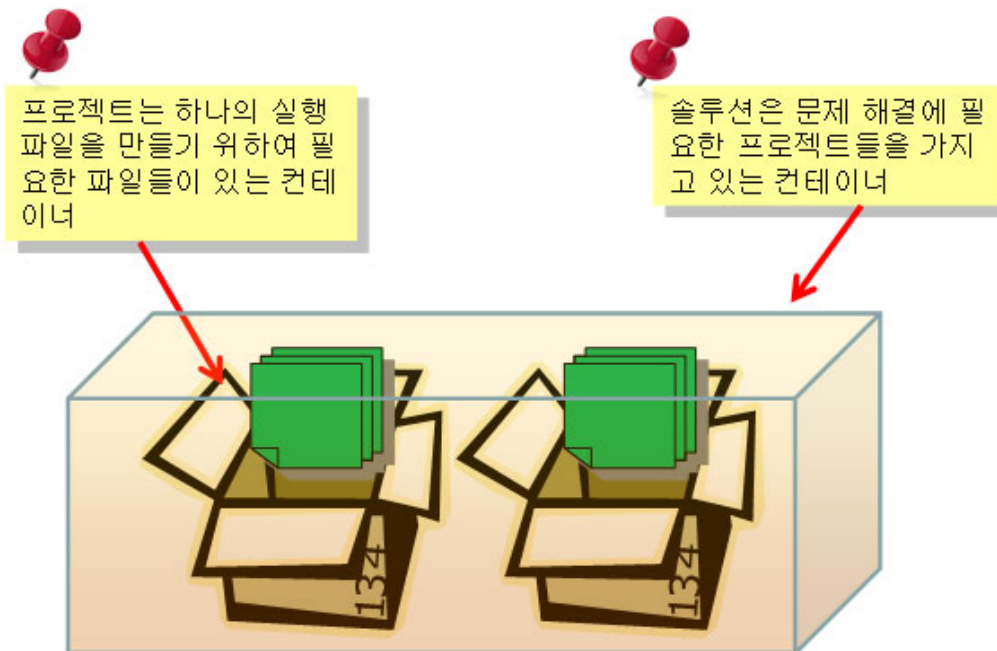
# 솔루션과 프로젝트

## ◆ 솔루션 (Solution)

- 문제 해결에 필요한 프로젝트가 들어있는 컨테이너

## ◆ 프로젝트 (Project)

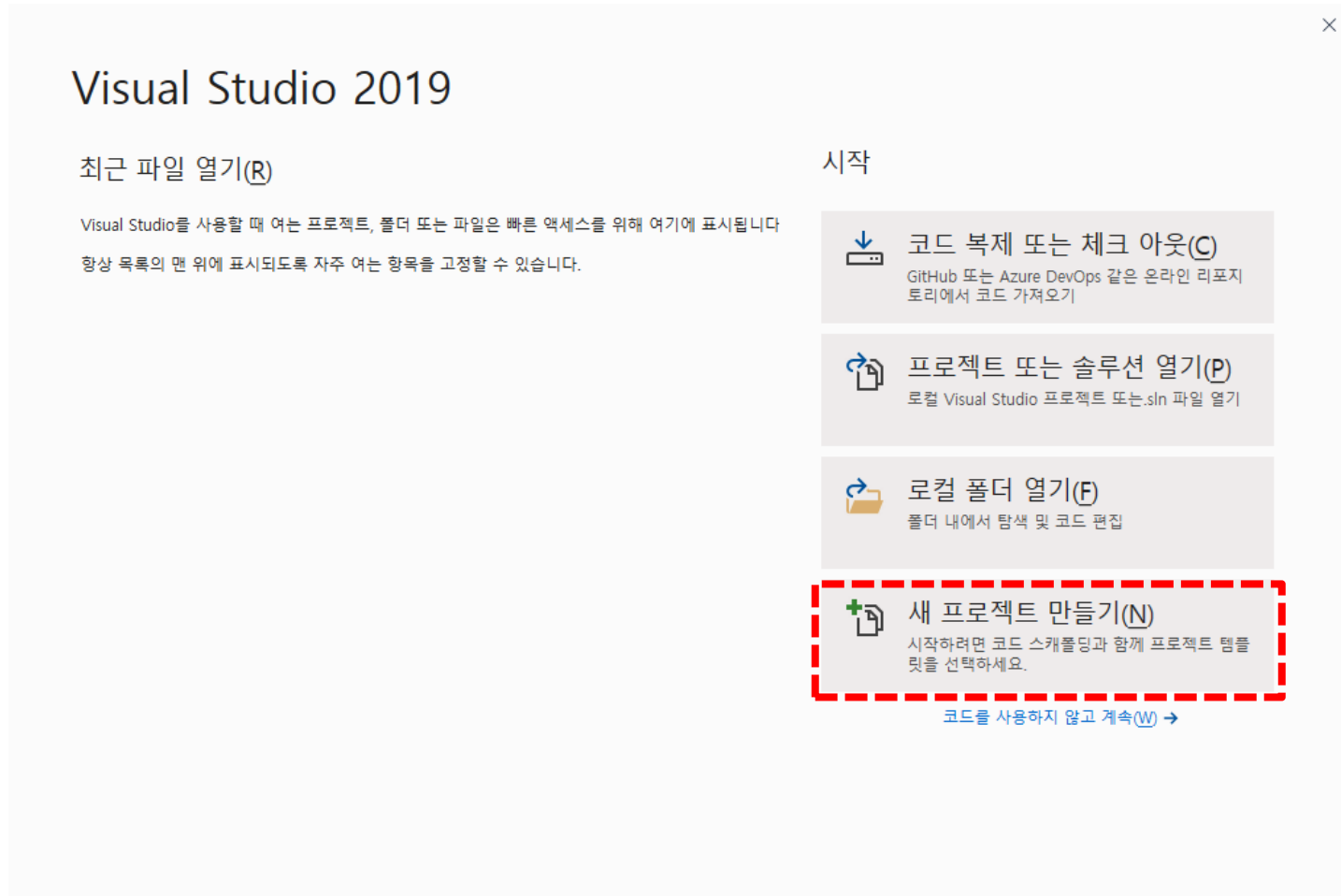
- 하나의 실행 파일을 만드는데 필요한 여러 가지 항목들이 들어 있는 컨테이너
- 프로젝트에 필요한 소스파일, 헤더파일 및 리소스 파일들을 포함





# 프로젝트 생성(1)

## ◆ 프로젝트 생성(1)



# 프로젝트 생성(2)

## ◆ 프로젝트 생성(2)

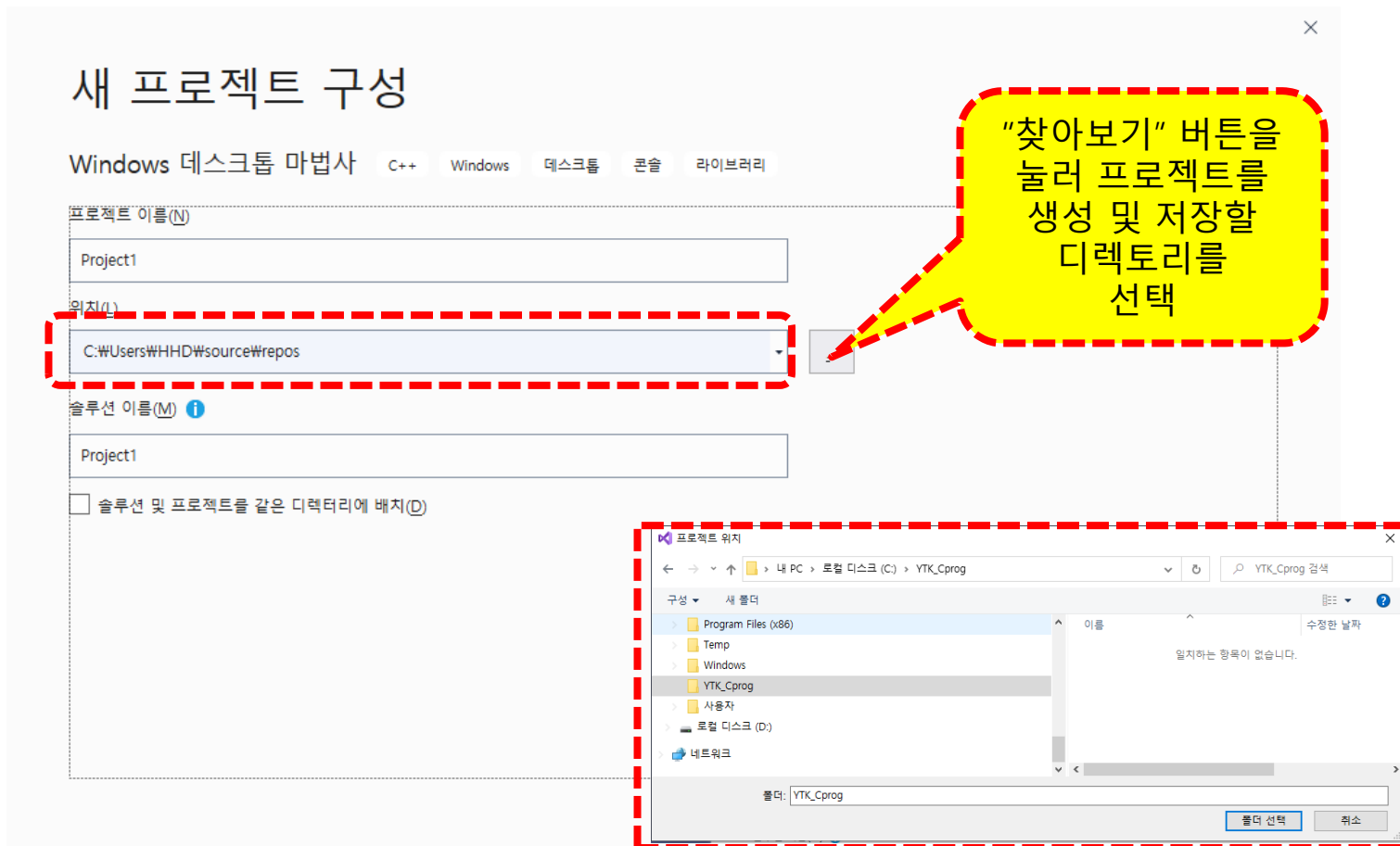
- 빈 프로젝트 선택 후 다음



# 프로젝트 생성(3)

## ◆ 프로젝트 생성(3)

- 저장위치(디렉토리) 생성/지정
- 예: 내 PC > 로컬디스크 (C:) > XYZ\_Cprog



# 프로젝트 생성(4)

## ◆ 프로젝트 생성(4)

- 프로젝트 이름 설정
- 예: Lab01 – Basic C Programming

새 프로젝트 구성

빈 프로젝트 C++ Windows 콘솔

프로젝트 이름(N)

Lab01 - Basic C programming

위치(L)

C:\YTK\_Cprog\

솔루션 이름(M) ⓘ

Lab01 - Basic C programming

☐ 솔루션 및 프로젝트를 같은 디렉터리에 배치(D)

뒤로(B) 만들기(C)

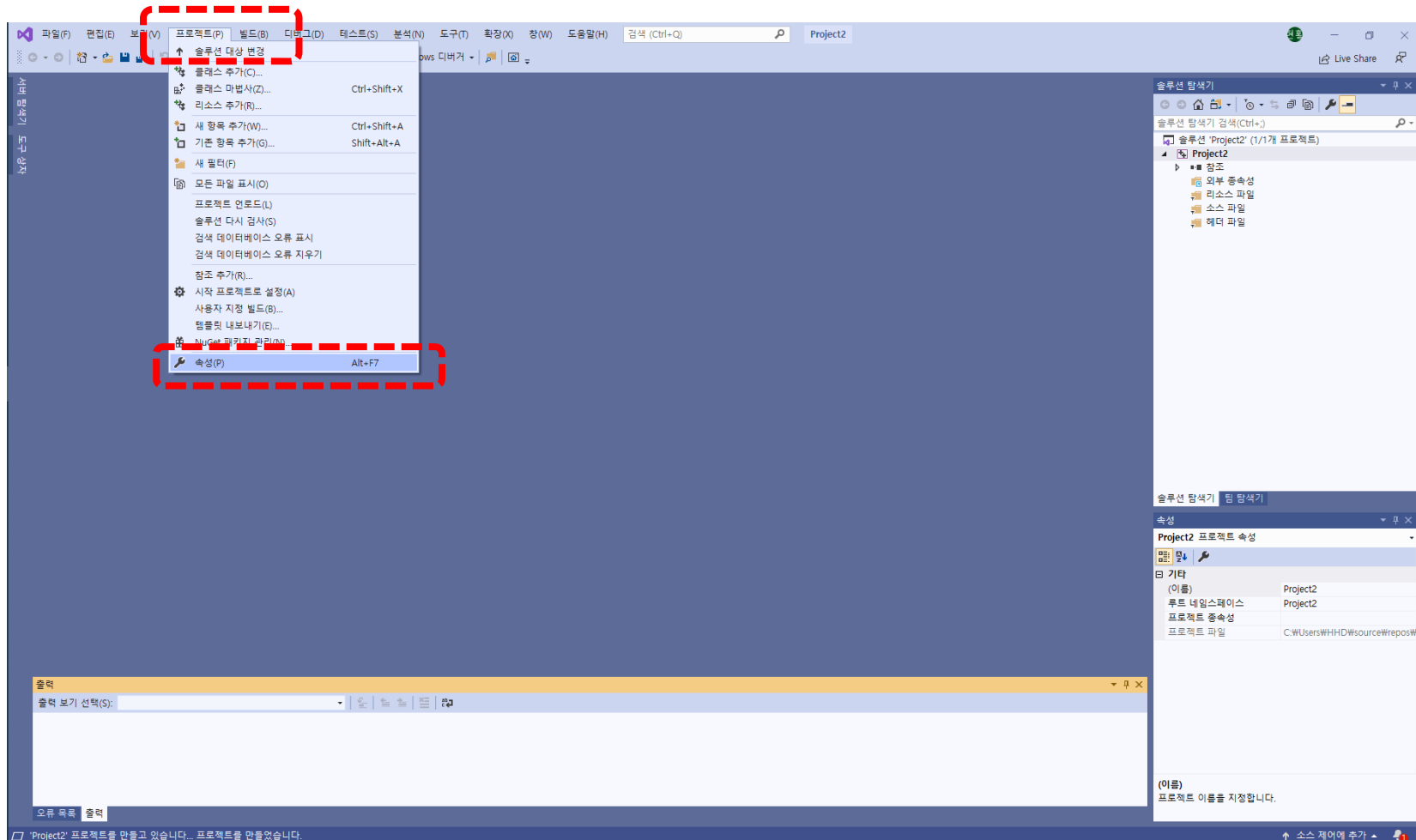
프로젝트 이름에  
"Lab01 – Basic C Programming"  
입력



# 프로젝트 속성 설정(1)

## ◆ 프로젝트 속성 설정(1)

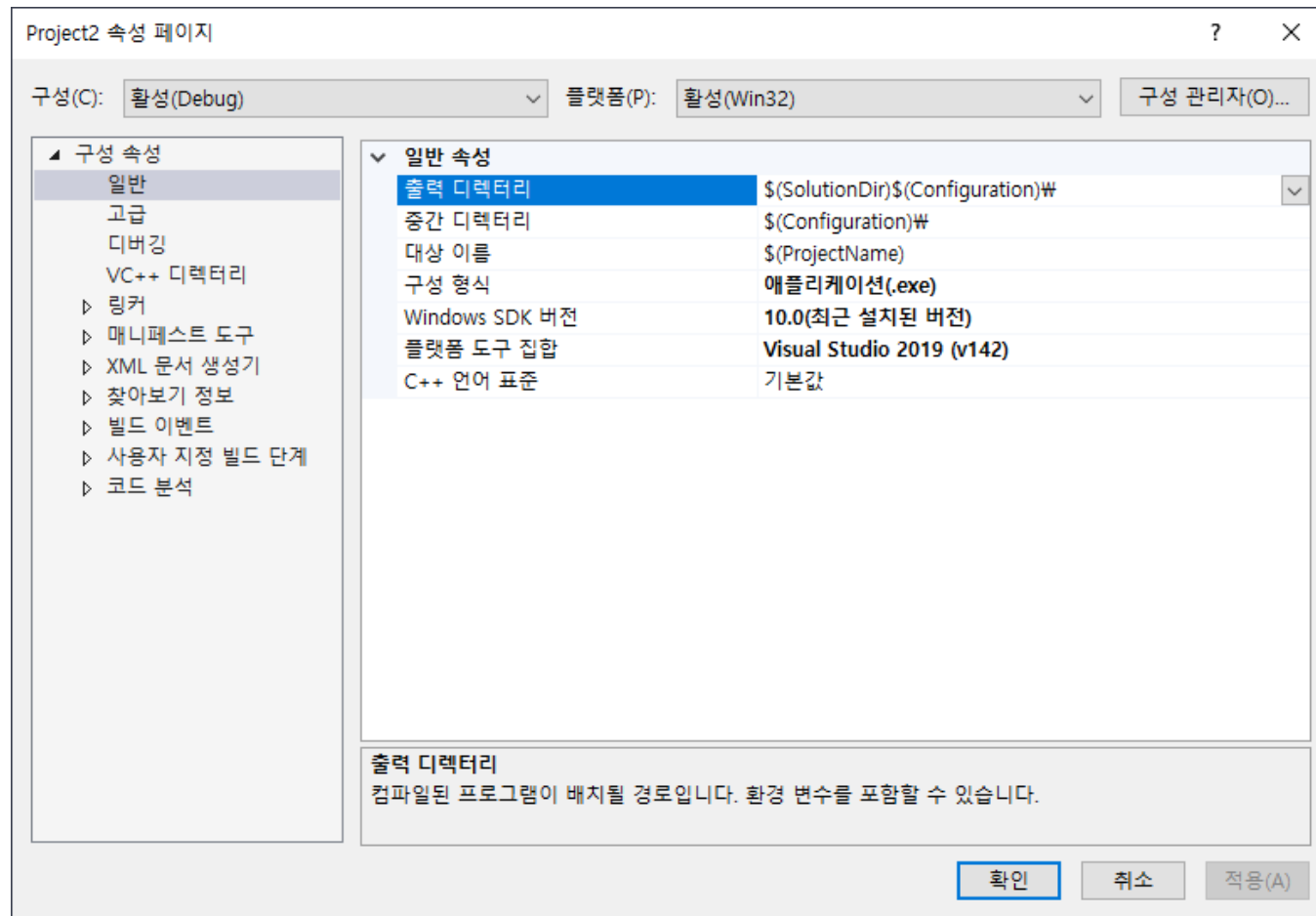
- 프로젝트 tab -> 속성 클릭(Alt+F7)



# 프로젝트 속성 설정(2)

## ◆ 프로젝트 속성 설정(2)

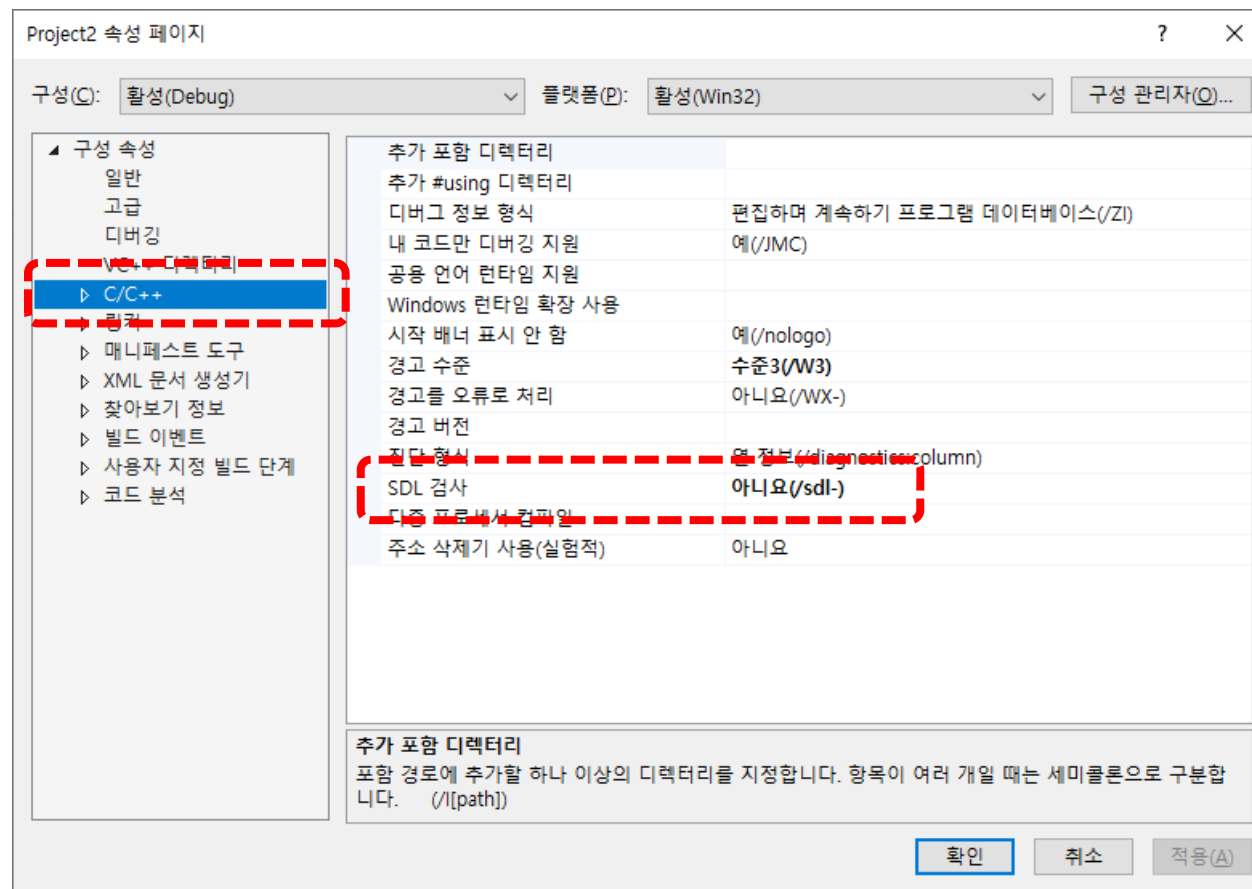
- 프로젝트 속성 페이지



# 프로젝트 속성 설정(3)

## ◆ 프로젝트 속성 설정(3)

- C/C++ 항목 클릭 후 SDL 검사를 아니요(/sdl-)로 변경 후 확인
  - scanf 함수 사용시 Visual Studio 2010 이상 컴파일러에서는 scanf\_s 함수 사용을 권장함에 따라 scanf 함수를 컴파일시 에러 발생.

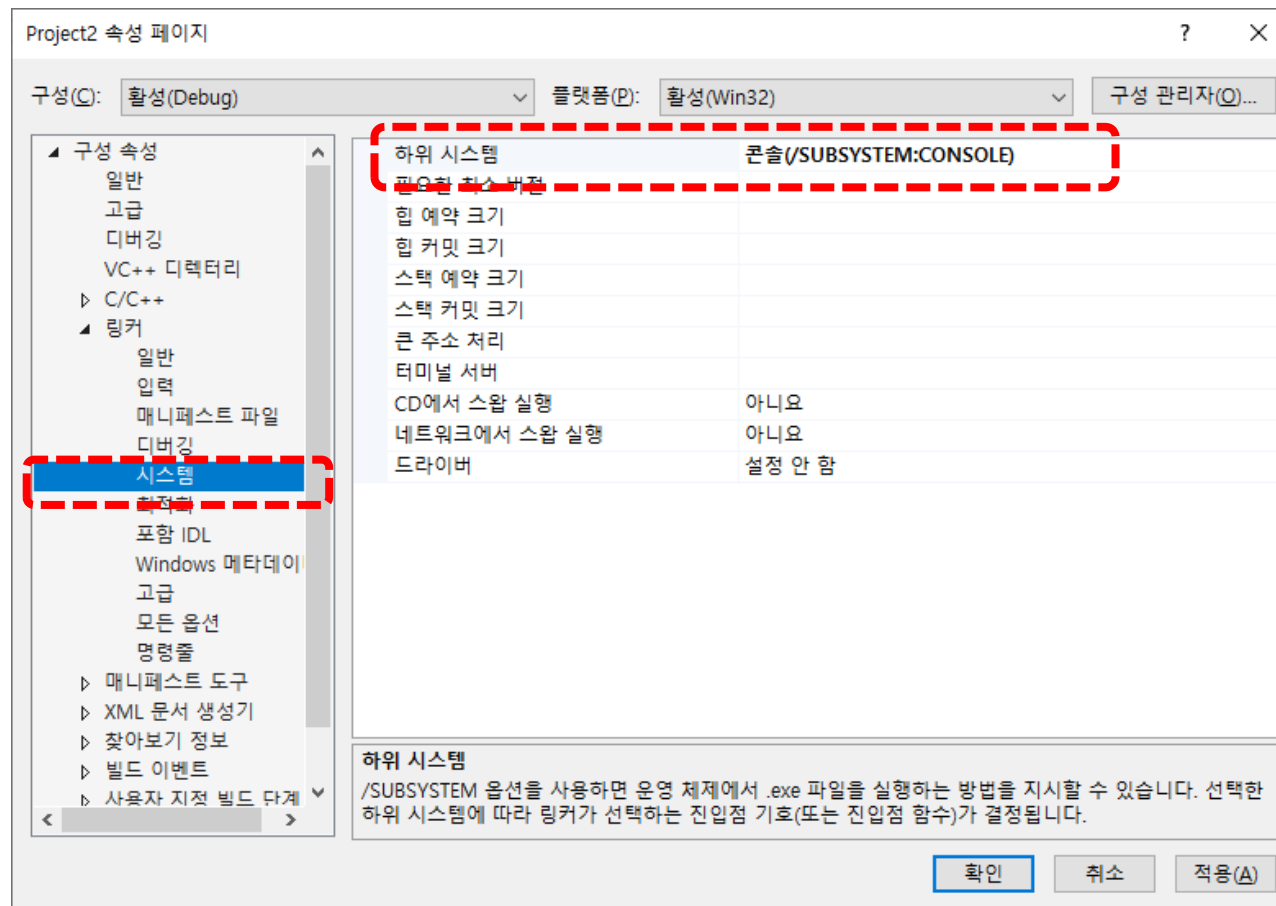


# 프로젝트 속성 설정(4)

## ◆ 프로젝트 속성 설정(4)

### ● 링커->시스템

- 하위 시스템의 목록이 콘솔(/SUBSYSTEM:CONSOLE)로 체크 되어 있는지 확인
- 컴파일후 실행 창이 자동으로 종료되는 현상 발생시 다음 속성을 확인 후 수정한다.

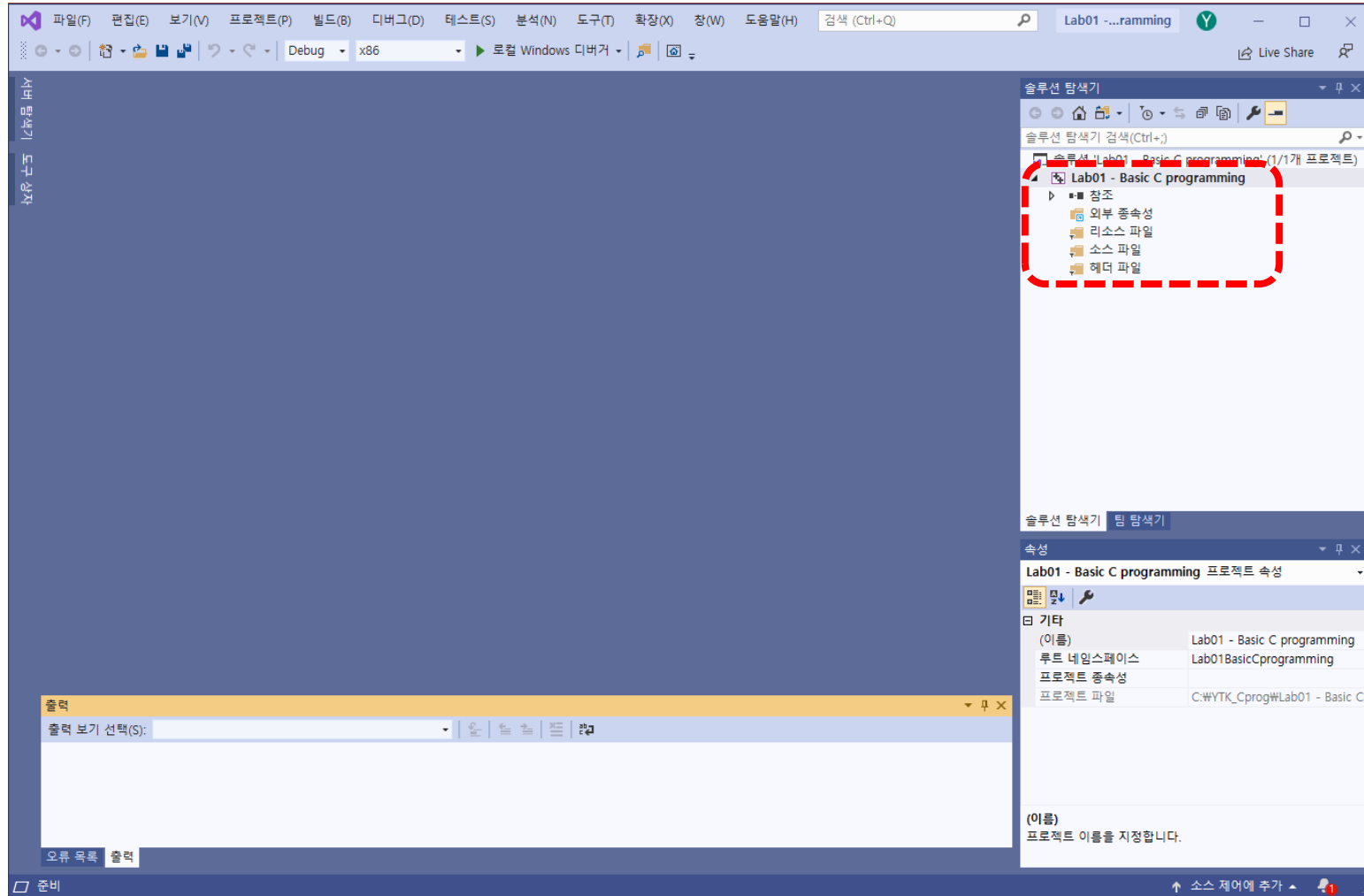




**Visual Studio Community 2019**  
**C 프로그램 프로젝트 생성 및**  
**C 프로그램 소스 코드 작성**

# 프로젝트 시작하기(1)

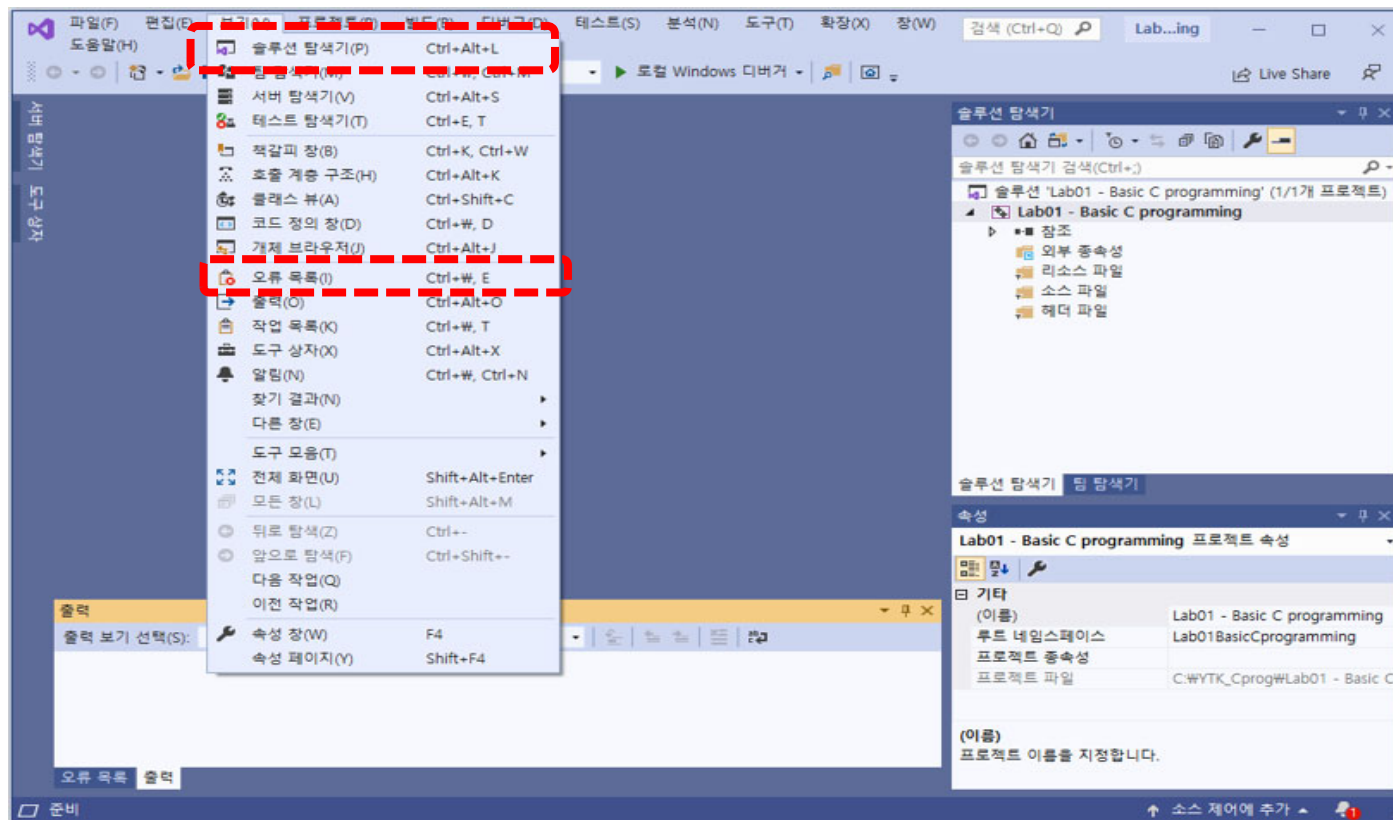
## ◆ 기본 구성 화면



# 프로젝트 시작하기(2)

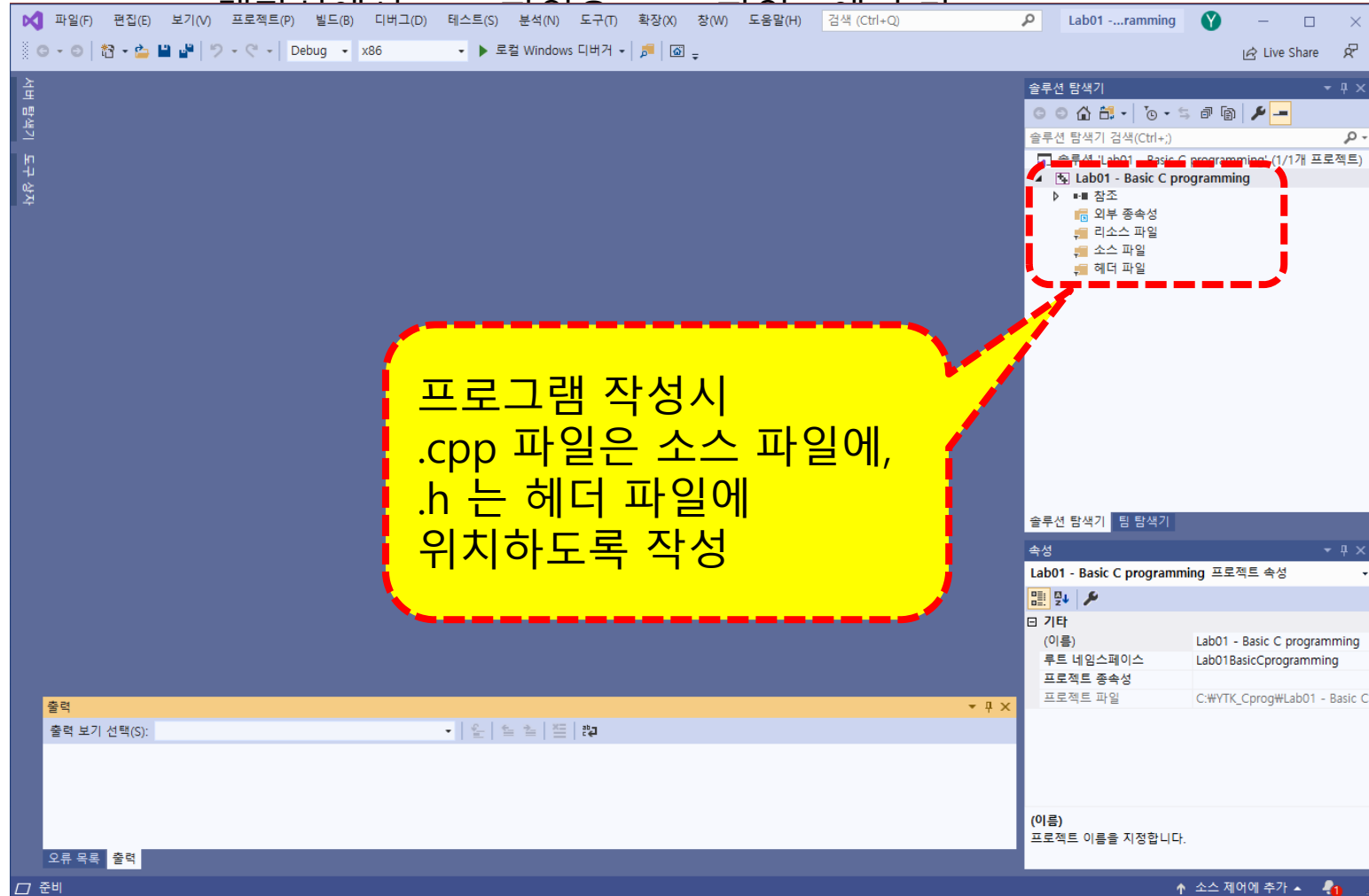
## ◆ Visual Studio 창(window)의 종류

- 메뉴->보기 클릭 후 원하는 탐색기 on/off 가능
  - C 언어 프로그램시 필요 창: 솔루션 탐색기, 출력 창, 오류목록
  - C++ 프로그램시 클래스 뷰를 이용시 빠른 소스 트리 구성 가능



# 프로젝트 시작하기(3)

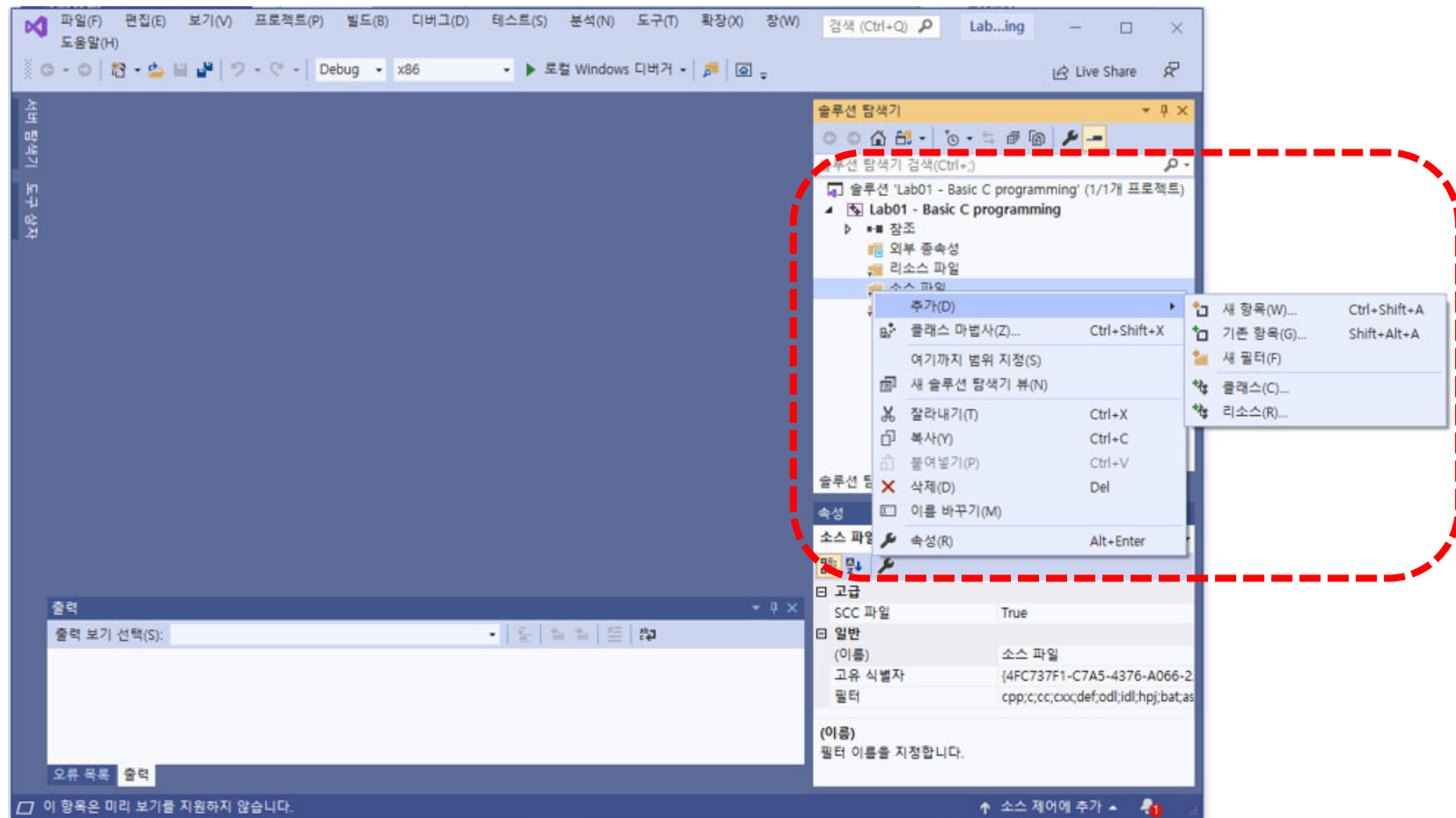
## ◆ 솔루션 탐색기



# 프로젝트 시작하기(4)

## ◆ 소스 파일 생성하기(1)

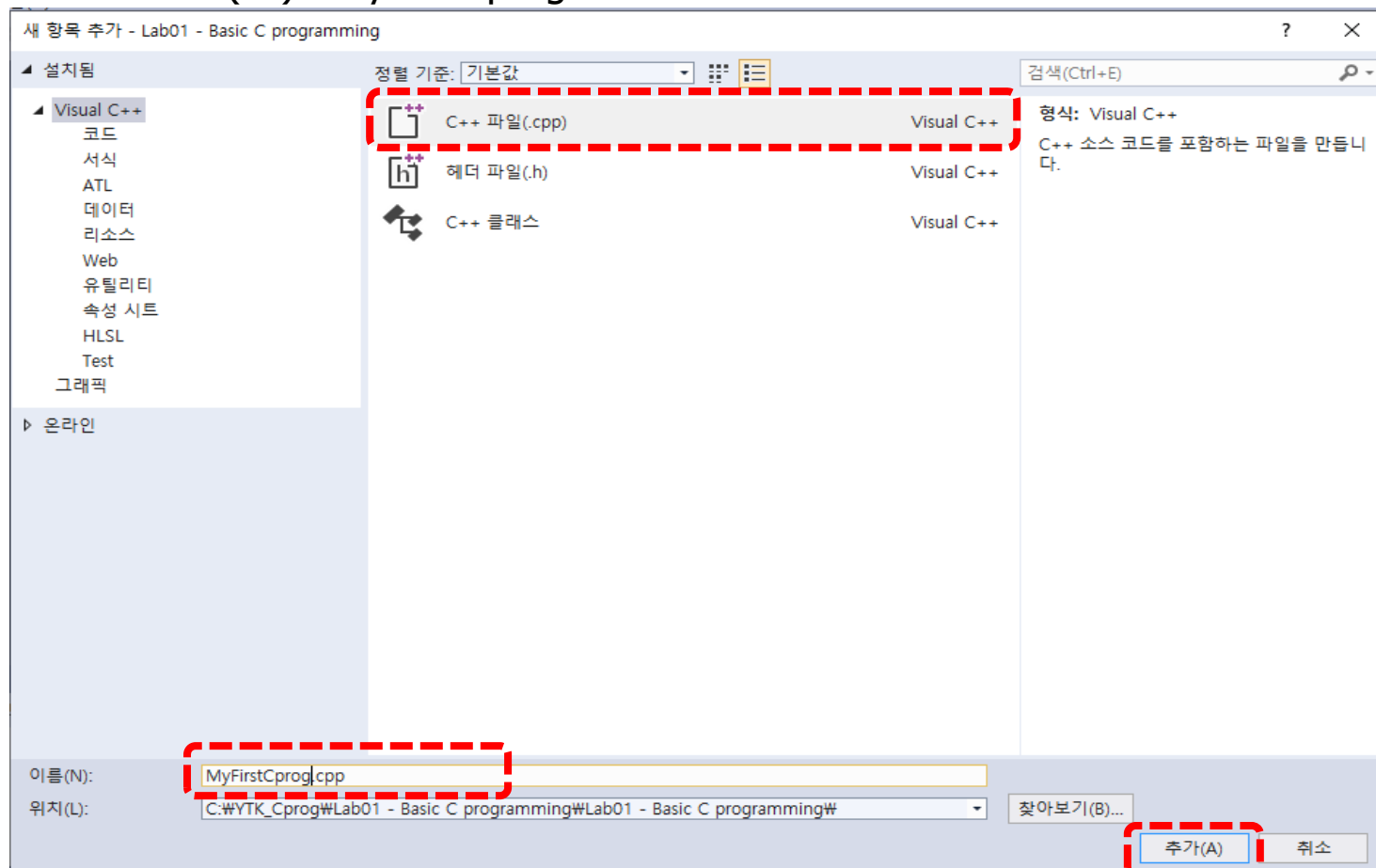
- 솔루션 탐색기 --> 소스파일 --> 추가 --> 새항목



# 프로젝트 시작하기(5)

## ◆ 소스 파일 생성하기(2)

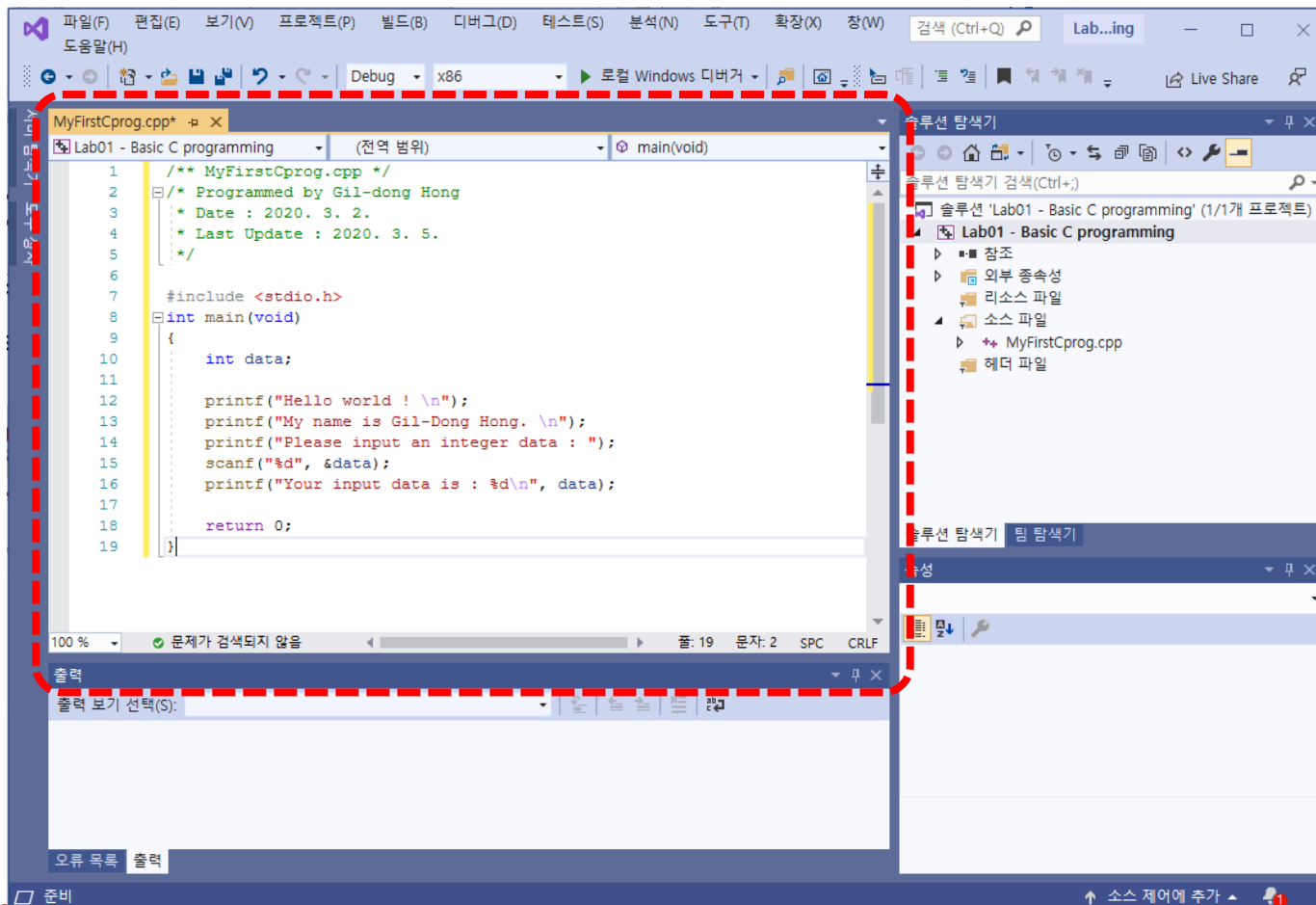
- C++파일(.cpp) 유형 선택 및 파일 이름 작성
- 파일 이름 (예) : MyFirstCprog



# 프로젝트 시작하기(6)

## ◆ 소스 파일 편집하기

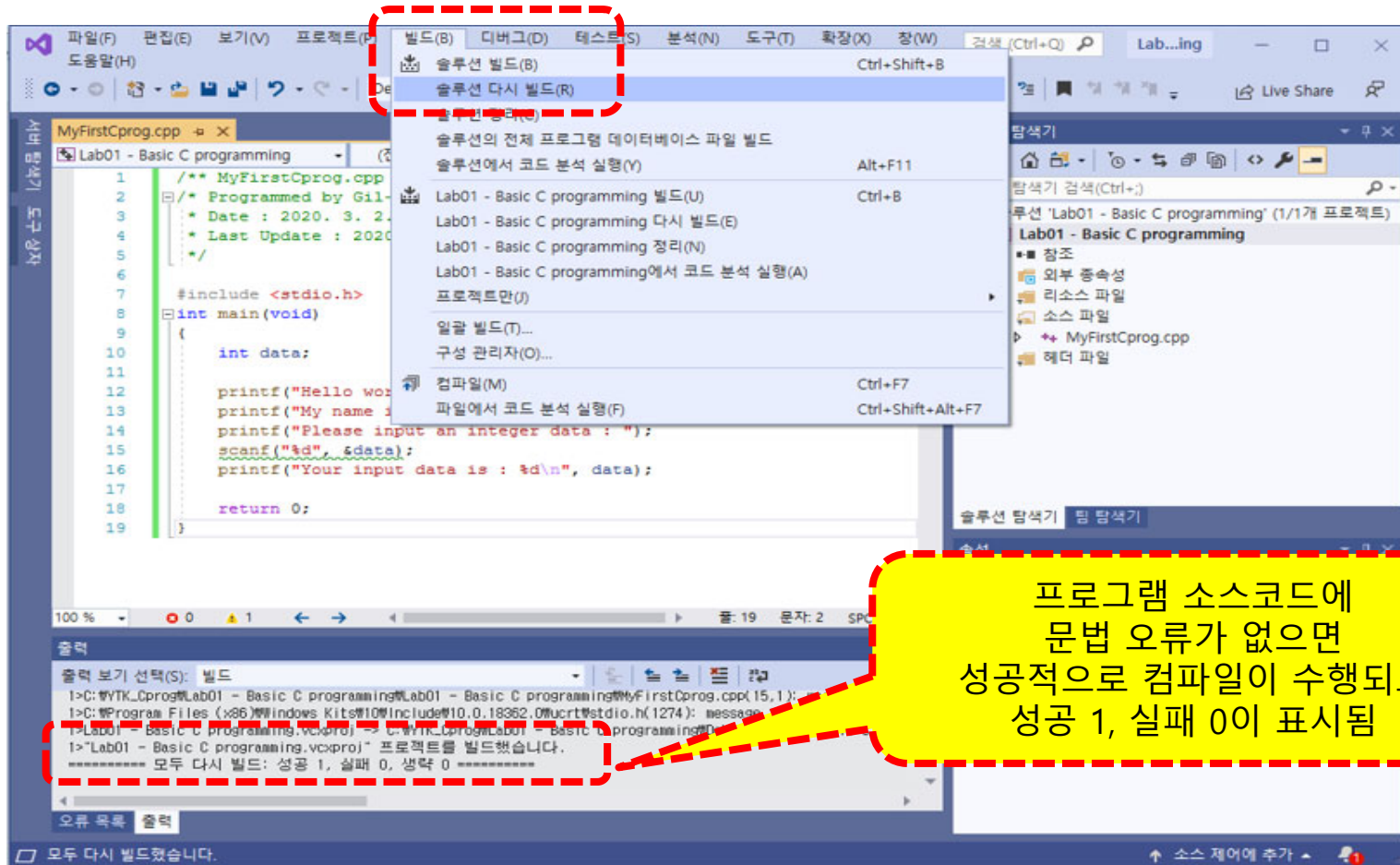
- 편집창에 소스코드 입력 및 편집
- 파일->파일이름\_저장 또는 Ctrl+S 키를 이용해 작성/편집한 코드 파일로 저장



# 프로젝트 시작하기(7)

## ◆ 컴파일 하기

- 빌드 탭-->솔루션 다시 빌드(R) 명령 수행(단축키 : Ctrl+Alt+F7)

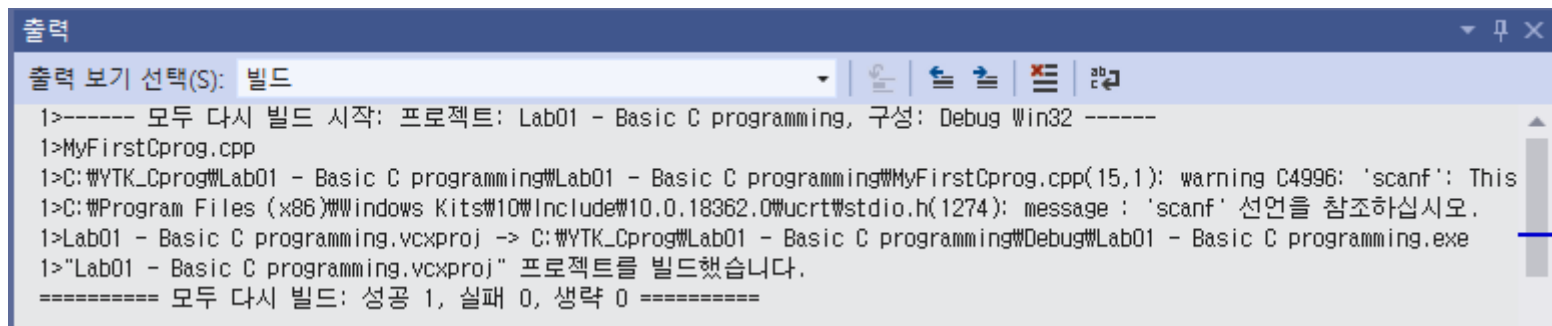




# 프로젝트 시작하기(8)

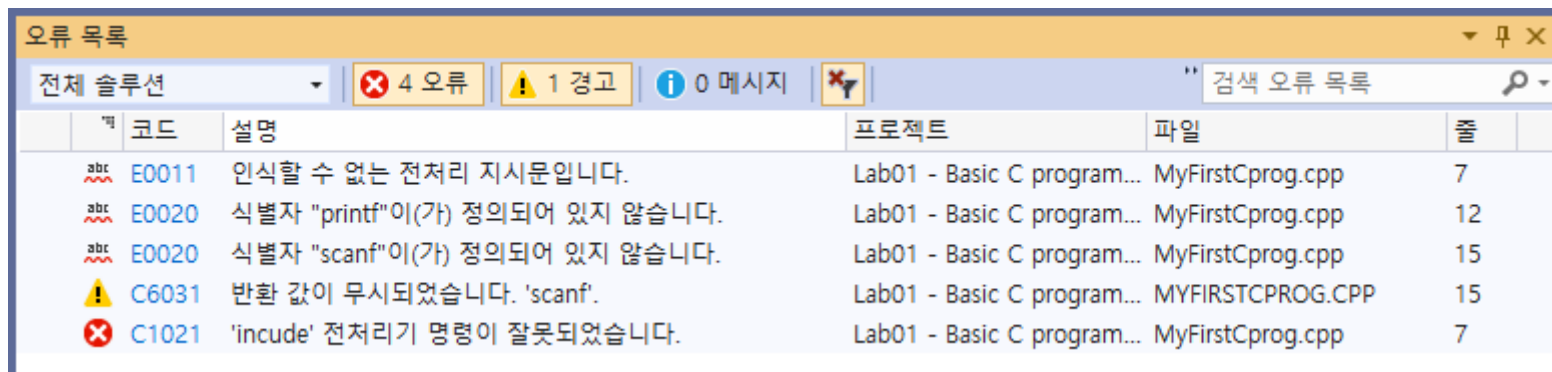
## ◆ 출력 창

- 컴파일시 발생하는 각종 상태들을 출력해주는 창



```
출력
출력 보기 선택(S): 빌드
1>----- 모두 다시 빌드 시작: 프로젝트: Lab01 - Basic C programming, 구성: Debug Win32 -----
1>MyFirstCprog.cpp
1>C:\YTK_Cprog\Lab01 - Basic C programming\MyFirstCprog.cpp(15,1): warning C4996: 'scanf': This
1>C:\Program Files (x86)\Windows Kits\10\Include\10.0.18362.0\ucrt\stdio.h(1274): message : 'scanf' 선언을 참조하십시오.
1>Lab01 - Basic C programming.vcxproj -> C:\YTK_Cprog\Lab01 - Basic C programming\Debug\Lab01 - Basic C programming.exe
1>"Lab01 - Basic C programming.vcxproj" 프로젝트를 빌드했습니다.
===== 모두 다시 빌드: 성공 1, 실패 0, 생략 0 =====
```

- 오류 발생시 오류 목록



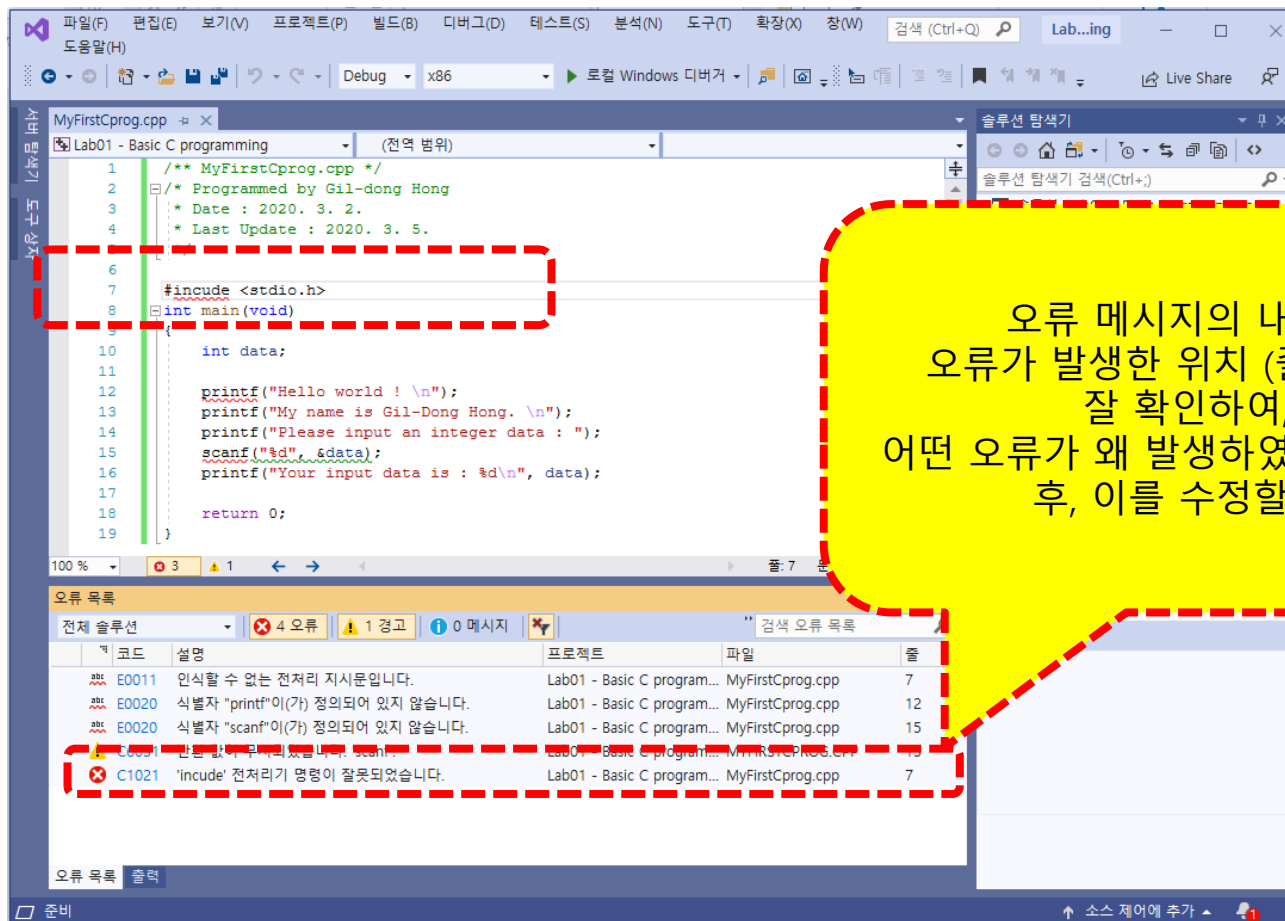
오류 목록						
전체 솔루션			4 오류	1 경고	0 메시지	검색 오류 목록
	코드	설명	프로젝트	파일	줄	
abc	E0011	인식할 수 없는 전처리 지시문입니다.	Lab01 - Basic C program...	MyFirstCprog.cpp	7	
abc	E0020	식별자 "printf"이(가) 정의되어 있지 않습니다.	Lab01 - Basic C program...	MyFirstCprog.cpp	12	
abc	E0020	식별자 "scanf"이(가) 정의되어 있지 않습니다.	Lab01 - Basic C program...	MyFirstCprog.cpp	15	
!	C6031	반환 값이 무시되었습니다. 'scanf'.	Lab01 - Basic C program...	MYFIRSTCPCROG.CPP	15	
✖	C1021	'incude' 전처리기 명령이 잘못되었습니다.	Lab01 - Basic C program...	MyFirstCprog.cpp	7	



# 프로젝트 시작하기(5)

## ◆ 오류목록

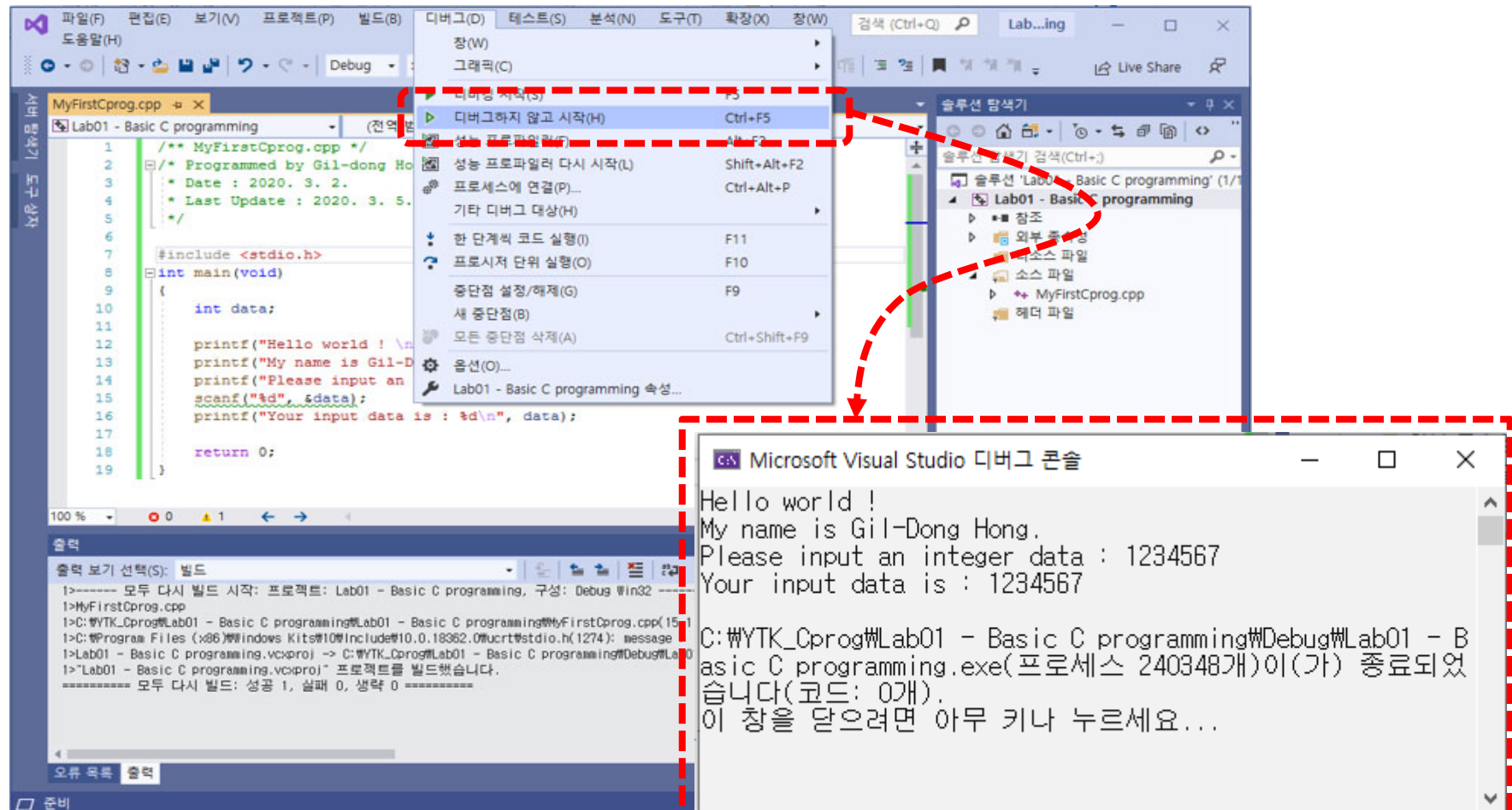
- 컴파일 오류 발생시 내용 표시창.
  - 오류 코드와 설명 및 오류가 발생한 파일과 줄을 표시한다.



# 프로젝트 시작하기(10)

## ◆ 프로그램 실행 하기

- 디버그 탭 -> "디버깅 하지 않고 시작(H)" 메뉴선택(단축키 : Ctrl+F5)



# 기본적인 코드 구조

Comment

```
/** MyFirstCprog.cpp */  
/* Programmed by Gil-dong Hong  
 * Date : 2020. 3. 2.  
 * Last Update : 2020. 3. 5.  
 */
```

```
#include <stdio.h>
```

**include system library for  
standard input/output functions**

```
int main(void)  
{  
    int data;  
  
    printf("Hello world ! \n");  
    printf("My name is Gil-Dong Hong. \n");  
    printf("Please input an integer data : ");  
    scanf("%d", &data);  
    printf("Your input data is : %d\n", data);  
  
    return 0;  
}
```



# 주석 (Comment)

## ◆ 프로그램 소스 코드 작성에서의 기본 주석 표기내용

```
/**
 * 파일명: "automaticTempControl.c" or "xxx.h", or "yyy.cpp"
 * 프로그램의 목적 및 기본 기능:
 *   - 이 프로그램은 사물인터넷의 온도 센서를 읽고, 사전에 설정된 온도가 유지될 수 있도록 히터/에어컨을
 *     동작시키는 . . . . .
 *
 * 프로그램 작성자: 홍길0 (2021년 3월 1일),
 * 최종 수정 및 보완 : Version 2.0, 2021년 3월 4일 (김영0).
 *
 * =====
 * 프로그램 수정/보완 이력
 * =====
 * 프로그램 수정/보완작업자   일자   수정/보완 내용
 * 홍길0       2020/03/01   v1.0   온도센서 읽기, 히터 동작, 에어컨 동작 모듈 완성
 * 정동0       2020/03/03   v1.1   GUI 기능 보완
 * 김영0       2020/03/04   v2.0   온도 센서 읽기의 데이터 오류 발생 여부 확인 기능 추가
 * =====
 */
```



## 중간 점검

- ◆ 에디터, 컴파일러, 링커, 실행, 디버깅 등의 기능이 하나의 프로그램 안에 들어 있는 것을 무엇이라고 하는가?

=> 통합개발 환경

### IDE (Integrated Development Environment)

- ◆ Visual C++에서 새로운 프로젝트를 생성하는 메뉴는 무엇인가?
- ◆ Visual C++에서 프로젝트에 속하는 소스 파일을 컴파일하여 실행 파일을 생성하는 메뉴는?
- ◆ C 언어에서는 대문자와 소문자를 구별하는가?
- ◆ Visual C++를 이용하여서 MyFirst\_C\_Prog.cpp라는 소스 파일을 컴파일하였을 때 생성되는 파일들은 무엇인가?
- ◆ Visual C++를 사용하여 소스 프로그램을 편집하는 경우, 메모장같은 다른 텍스트 에디터를 사용하여도 되는가?



# C 프로그래밍 언어의 소스 코드 구조

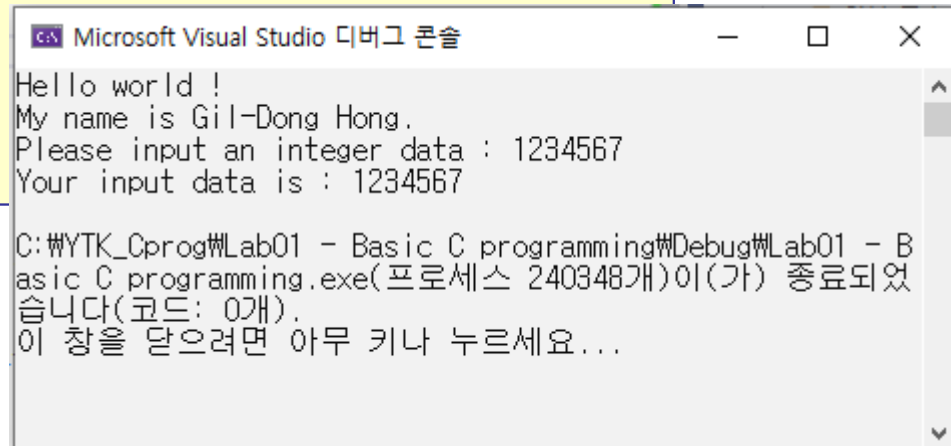
# 첫번째 프로그램의 설명

```
/** MyFirstCprog.cpp */
/* Programmed by Gil-dong Hong
 * Date : 2020. 3. 2.
 * Last Update : 2020. 3. 5.
 */

#include <stdio.h>
int main(void)
{
    int data;

    printf("Hello world ! \n");
    printf("My name is Gil-Dong Hong. \n");
    printf("Please input an integer data : ");
    scanf("%d", &data);
    printf("Your input data is : %d\n", data);

    return 0;
}
```



```
Microsoft Visual Studio 디버그 콘솔

Hello world !
My name is Gil-Dong Hong.
Please input an integer data : 1234567
Your input data is : 1234567

C:\YTK_Cprog\Lab01 - Basic C programming\Debug\Lab01 - Basic C programming.exe(프로세스 240348개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```





# 프로그램 == 작업 지시서

```
/** MyFirstCprog.cpp */  
/* Programmed by Gil-dong Hong  
* Date : 2020. 3. 2.  
* Last Update : 2020. 3. 5.  
*/
```

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int data;
```

```
    printf("Hello world ! \n");
```

```
    printf("My name is Gil-Dong Hong. \n");
```

```
    printf("Please input an integer data : ");
```

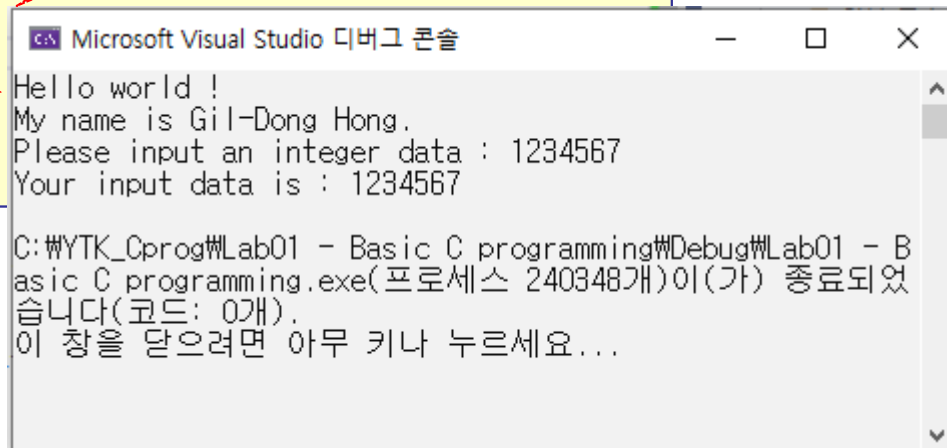
```
    scanf("%d", &data);
```

```
    printf("Your input data is : %d\n", data);
```

```
    return 0;
```

```
}
```

프로그램의 작업 지시에 따라  
화면으로 출력 기능 및  
키보드로 부터의 입력을 수행



Microsoft Visual Studio 디버그 콘솔

```
Hello world !  
My name is Gil-Dong Hong.  
Please input an integer data : 1234567  
Your input data is : 1234567
```

C:\YTK\_Cprog\Lab01 - Basic C programming\Debug\Lab01 - Basic C programming.exe(프로세스 240348개)이(가) 종료되었습니다(코드: 0개).  
이 창을 닫으려면 아무 키나 누르세요...



# 프로그램에서 수행하여야 할 작업들을 순서에 따라 작성

```
/** MyFirstCprog.cpp */  
/* Programmed by Gil-dong Hong  
* Date : 2020. 3. 2.  
* Last Update : 2020. 3. 5.  
*/
```

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int data;
```

```
    printf("Hello world ! \n");
```

```
    printf("My name is Gil-Dong Hong. \n");
```

```
    printf("Please input an integer data : ");
```

```
    scanf("%d", &data);
```

```
    printf("Your input data is : %d\n", data);
```

```
    return 0;
```

```
}
```

프로그램에서  
수행하여야 할  
작업 들을  
순서에 따라  
이곳에 작성



# 프로그램 소스코드에 대한 설명

```
/** MyFirstCprog.cpp */  
/* Programmed by Gil-dong Hong  
 * Date : 2021. 3. 2.  
 * Last Update : 2021. 3. 5.  
 */
```

프로그램 소스코드에 대한 간략한  
설명을 제공하는  
주석문 (comments)

```
#include <stdio.h>
```

프로그램 소스코드에서 사용할  
라이브러리 함수들의 함수원형  
(function prototype)이 포함된  
헤더파일을 추가하도록 지시

```
int main(void)
```

main() 함수 선언 : 함수 실행에서  
전달 받는 인수 (argument/  
parameter) 목록과  
반환 자료형 명시

```
{
```

```
int data;
```

함수에서 사용하는 지역 변수  
(local variable) 선언

```
printf("Hello world ! \n");  
printf("My name is Gil-Dong Hong. \n");  
printf("Please input an integer data : ");  
scanf("%d", &data);  
printf("Your input data is : %d\n", data);
```

함수의 내용

```
return 0;
```

함수의 실행 결과를 반환

```
}
```



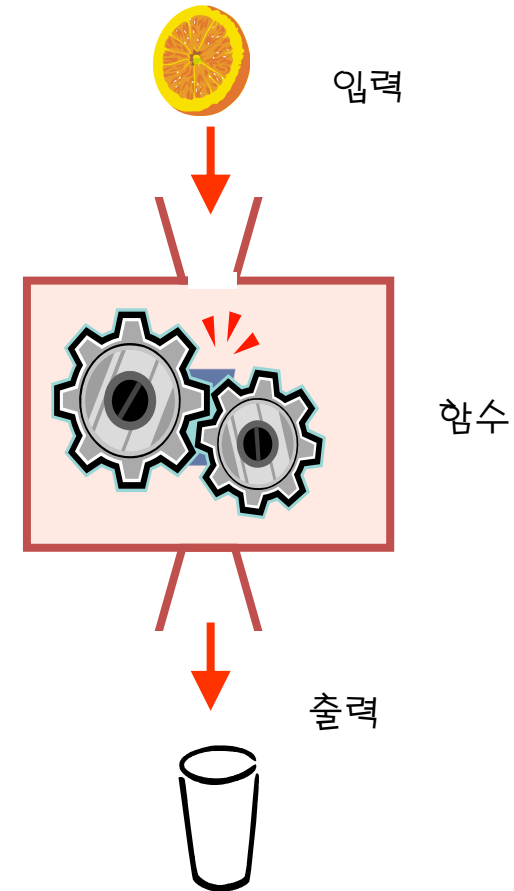
# #include <stdio.h>

- ◆ #include는 소스 코드 안에 특정 파일을 현재의 위치에 포함
- ◆ 헤더 파일(header file): 컴파일러가 필요로 하는 정보를 가지고 있는 파일
- ◆ **stdio.h: standard input output header file**
  - 입력을 위한 scanf() 함수, 출력을 위한 printf() 함수를 사용하기 위해서는 반드시 <stdio.h> 헤더파일을 포함 해야 된다.  
(화살 괄호 "<>"는 시스템 라이브러리를 의미)
- ◆ 주의!: 전처리기 지시자 문장 끝에는 세미콜론(;)을 붙이면 안 된다.

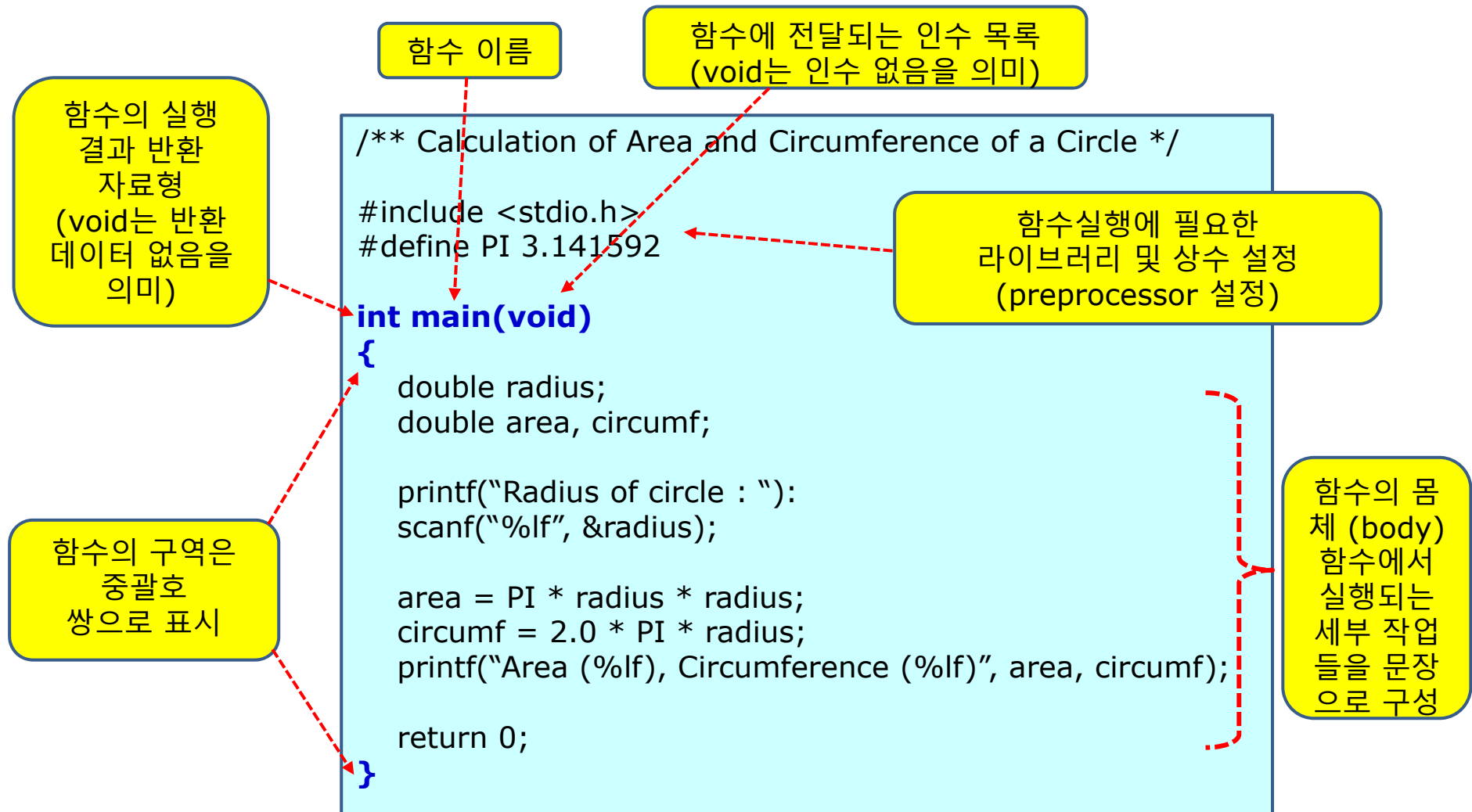


# 함수 (function)

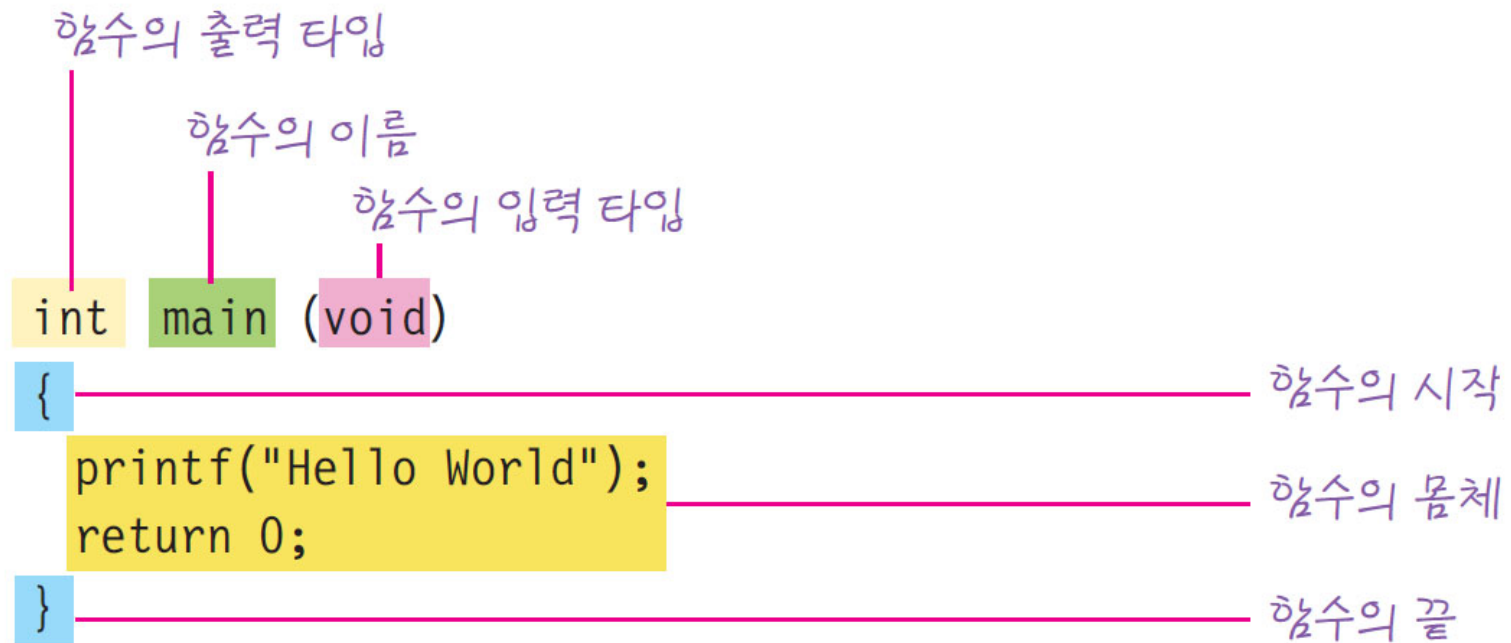
- ◆ **함수(function)**: 특정 기능을 수행하는 처리 단계들을 중괄호({})로 묶어서 이름을 붙인 것
- ◆ 함수는 프로그램을 구성하는 기본적인 단위(부품)
- ◆ 각 함수에는 전달되는 인수(argument)들의 자료형이 명시됨
- ◆ 각 함수의 반환 자료형도 명시됨
- ◆ 전달되는 인수가 없거나, 반환되는 데이터가 없는 경우 **void**로 표시
- ◆ **main()** 도 함수 중 하나



# 함수의 구조



# 함수의 간략한 설명



# C 프로그램 문장 (statement)

## ◆C/C++ 프로그램의 문장

- 함수는 여러 개의 문장으로 이루어진다.
- 문장들은 순차적으로 실행된다.
- 각 문장은 ; (세미콜론)으로 끝나야 한다.

```
/** Calculation of Area and Circumference of a Circle */  
#include <stdio.h>  
#define PI 3.141592  
  
int main(void)  
{  
    double radius, area, circumf;  
  
    printf("Radius of circle : "):  
    scanf("%lf", &radius);  
  
    area = PI * radius * radius;  
    circumf = 2.0 * PI * radius;  
    printf("Area (%lf), Circumference (%lf)", area, circumf);  
  
    return 0;  
}
```

프로그램 소스코드의 문장들은 차례대로 실행된다. 함수 호출이 있는 경우, 그 호출된 함수를 실행하고 난 후, 호출한 함수로 돌아와서 나머지 문장을 계속 실행한다.





# printf("Hello World!");

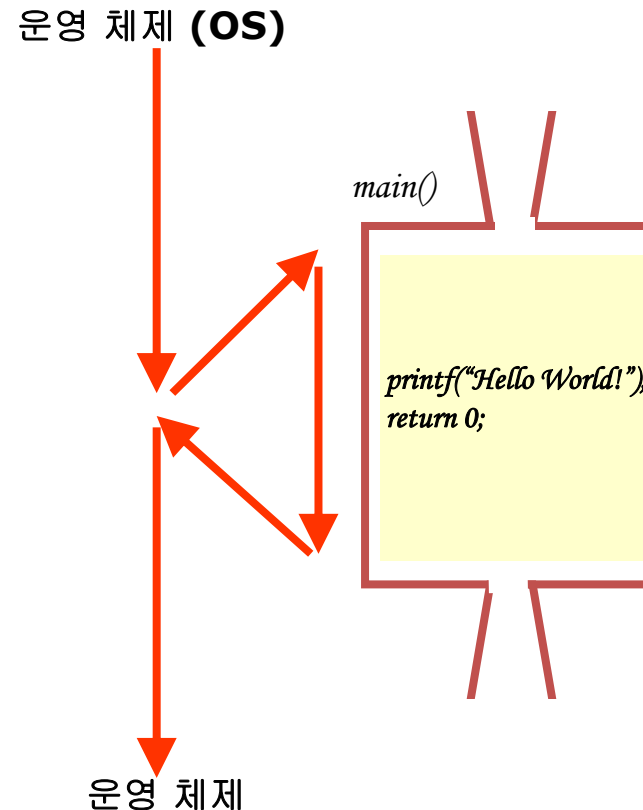
- ◆ **printf()**는 컴파일러가 제공하는 함수로서 출력을 담당
- ◆ 큰따옴표 안의 문자열을 화면에 출력



# return 0;

## ◆ return은 함수의 결과값을 외부 (함수를 호출한 프로그램)로 반환

- 프로그램 실행에서 문제가 없는 경우 0또는 1을 반환
- 프로그램 실행에서 문제가 발생하는 경우 -1 또는 음수 값을 반환



# 중간 점검

- ◆ 문장의 끝에 추가하여야 하는 기호는?
- ◆ `printf()`가 하는 기능은 무엇인가?
- ◆ `Scanf()`가 하는 기능은 무엇인가?
- ◆ 변수(**Variable**)이란 무엇인가?
- ◆ 자료형(**Data type**)이란 무엇인가?



응용 프로그램

# 응용 프로그램 #1

- ◆ 다음과 같은 형식으로 본인의 이름과 학번을 출력하는 프로그램을 제작하여 보자.



# 첫번째 버전

## ◆ 문장들은 순차적으로 실행된다는 사실 이용

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int st_id = 21312014;
```

```
    printf("Simple C-style program on Visual C++\n");
```

```
    printf("Hello world !\n");
```

```
    printf("My name is 홍길동\n");
```

```
    printf("My student ID is %10d\n", st_id);
```

```
    return 0;
```

```
}
```

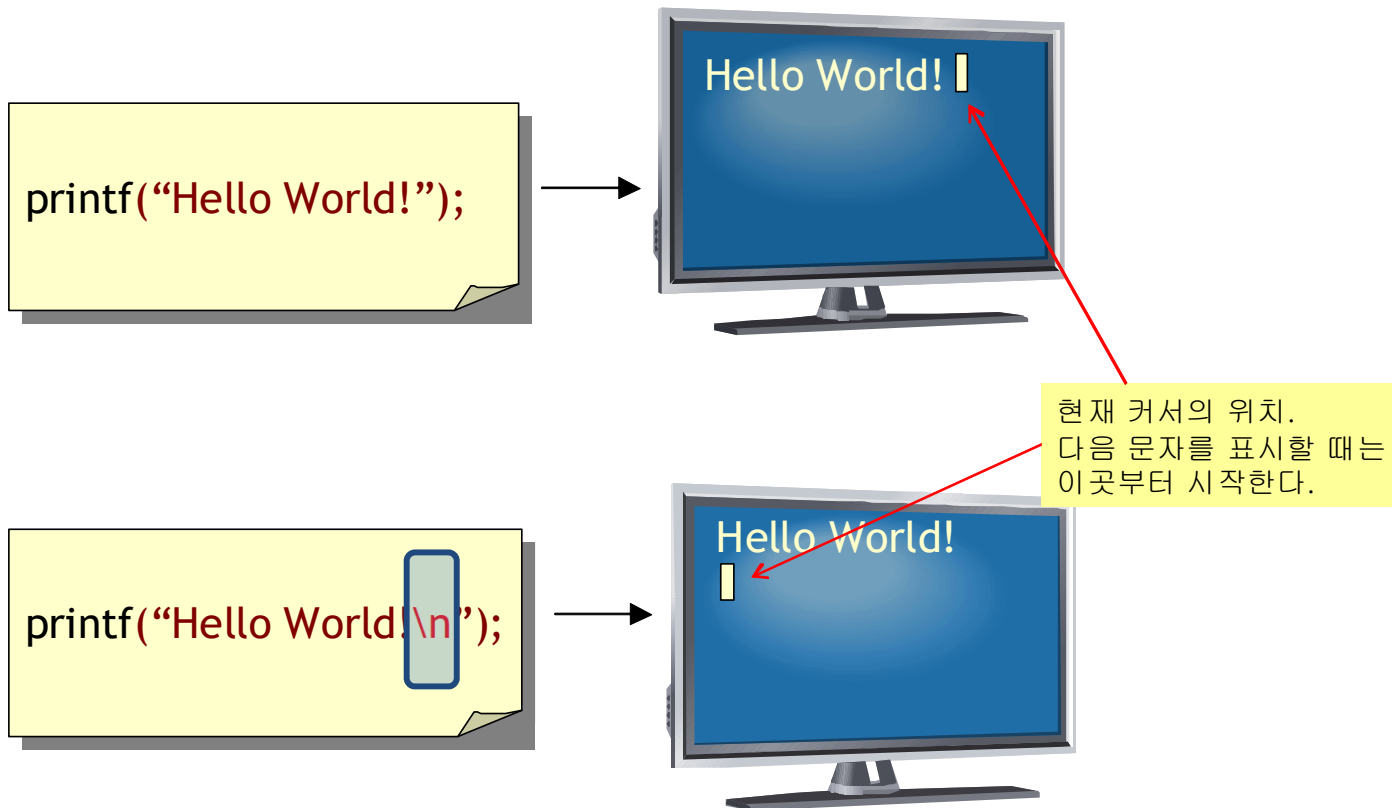
4개의 문장은 순차적으로 실행된다.

```
Simple C-style program on Visual C++
Hello world !
My name is 홍길동
My student ID is 21312014
계속하려면 아무 키나 누르십시오 . . .
```



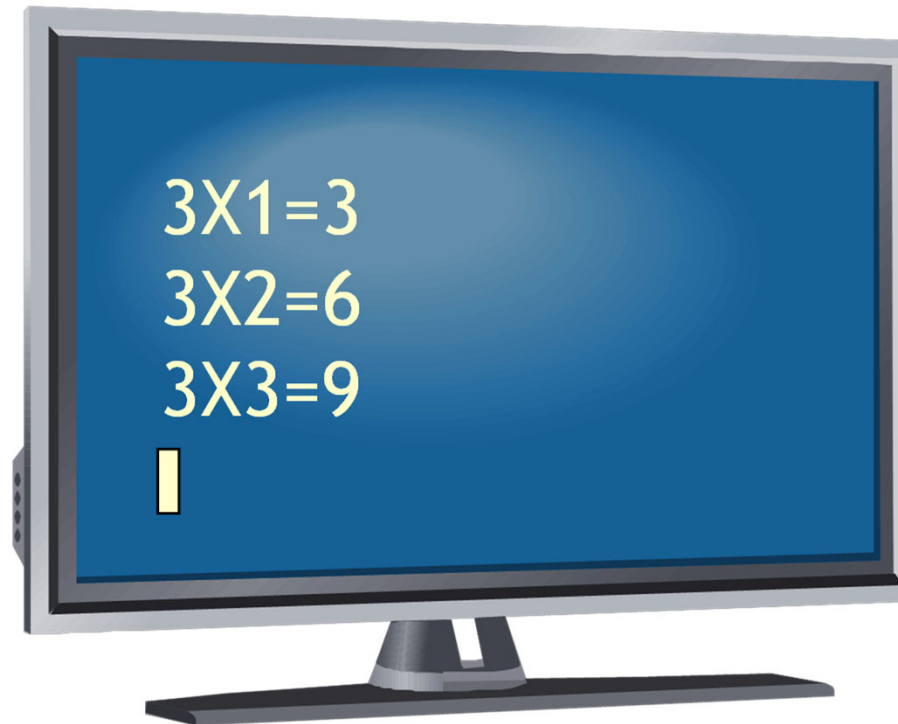
# 줄바꿈 문자 \n

- ◆ 줄바꿈 문자인 \n은 화면에서 커서는 다음줄로 이동하게 한다.



## 응용 프로그램 #2

◆ 다음과 같은 출력을 가지는 프로그램을 제작하여 보자.





# 응용 프로그램

- ◆ 역시 문장들은 순차적으로 수행된다는 점을 이용한다.

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    printf("3 X 1 = 3\n");
```

```
    printf("3 X 2 = 6\n");
```

```
    printf("3 X 3 = 9\n");
```

```
    return 0;
```

```
}
```

3개의 문장은  
순차적으로  
실행된다.



# 반복문을 사용하는 예제

## ◆ 구구단 출력

```
#include <stdio.h>

int main(void)
{
    int level;
    printf("구구단의 레벨:");
    scanf("%d", &level);
    printf("구구단%d단:\n", level);

    for (int i=1; i<=10; i++)
    {
        printf("%2d x %2d = %2d\n", level, level*i);
    }

    return 0;
}
```

```
구구단의 레벨 : 3
구구단3단:
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
```



# 다른 함수를 호출하는 예제

## ◆ 구구단표

```
#include <stdio.h>
```

```
void multiplicationTable(int x); // proto-type 선언
```

```
int main(void)
```

```
{
```

```
    printf("구구단표:\n");
```

```
    for (int i=1; i<=10; i++)
```

```
    {
```

```
        printf("\n구구단 (%d단) :\n", i);
```

```
        multiplicationTable(i);
```

```
    }
```

```
    return 0;
```

```
}
```

```
void multiplicationTable(int x)
```

```
{
```

```
    for (int i=1; i<=10; i++)
```

```
    {
```

```
        printf("%2d x %2d = %2d\n", x, i, x*i);
```

```
    }
```

```
}
```

구구단표 :

구구단 (1단) :

```
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
```

구구단 (2단) :

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
```

...

```
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
```

구구단 (10단) :

```
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

계속하려면 아무 키나 누르십시오 . . .



# C 프로그래밍과 C++ 프로그래밍

## ◆ C 프로그래밍과 C++ 프로그래밍의 차이점

- Visual Studio의 C++ 프로그래밍 통합 개발 환경에서는 C와 C++를 함께 지원
- C 프로그래밍
  - `#include <stdio.h>` // include standard input/output header file
  - `printf()`와 `scanf_s()`를 사용
- C++ 프로그래밍
  - `#include <iostream>` // include input / output stream library
  - `using namespace std;` // standard name space를 사용
  - `cin`과 `cout`을 주로 사용
- `if-else`, `for`, `while`, `do-while` 등은 동일하게 사용



# C Programming 예제 (1)

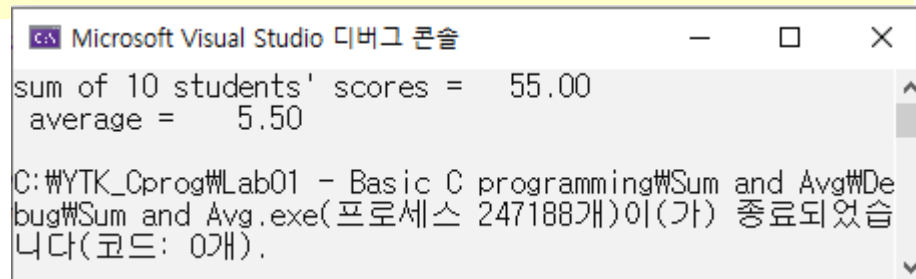
## ◆ 10개의 정수 데이터의 합산과 평균

```
#include <stdio.h>

#define NUM_STUDENTS 10

void main()
{
    int score[NUM_STUDENTS] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    double sum, avg; // local variable sum and average

    sum = 0.0;
    for (int i=0; i< NUM_STUDENTS; i++)
    {
        sum = sum + score[i];
    }
    avg = sum / (double) NUM_STUDENTS;
    printf(" sum of %d students' scores = %7.2lf\n", NUM_STUDENTS, sum);
    printf(" average = %7.2lf\n", avg);
}
```



```
Microsoft Visual Studio 디버그 콘솔
sum of 10 students' scores = 55.00
average = 5.50

C:\YTK_Cprog\Lab01 - Basic C programming\Sum and Avg\De
bug\Sum and Avg.exe (프로세스 247188개)이(가) 종료되었습
니다(코드: 0개).
```



# C Programming 예제 (2)

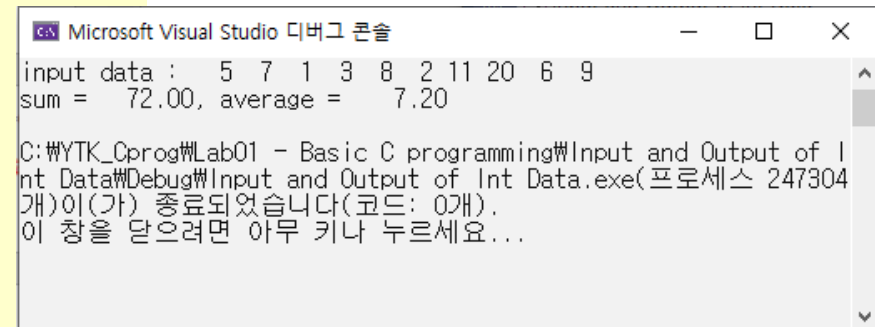
## ◆ 정수데이터의 입력, 출력 및 평균 계산

```
/** data processing with input and output */
#include <stdio.h>
#define NUM_STUDENTS 10

void main(void)
{
    int score[NUM_STUDENTS] = { 0 };

    int data;
    double sum = 0.0, avg;

    printf("input %d students' score : ", NUM_STUDENTS);
    for (int i = 0; i < NUM_STUDENTS; i++)
    {
        scanf("%d", &data);
        score[i] = data;
        sum = sum + data;
    }
    avg = sum / (double) NUM_STUDENTS;
    printf("input data : ");
    for (int i = 0; i < NUM_STUDENTS; i++)
    {
        printf("%3d", score[i]);
    }
    printf("\n");
    printf("sum = %7.2f, average = %7.2f\n", sum, avg);
}
```



Microsoft Visual Studio 디버그 콘솔

```
input data : 5 7 1 3 8 2 11 20 6 9
sum = 72.00, average = 7.20

C:\WYTK_Cprog\Lab01 - Basic C programming\Input and Output of I
nt Data\Debug\Input and Output of Int Data.exe(프로세스 247304
개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```



## 오류 수정 및 디버깅

# 오류 수정 및 디버깅

◆ 컴파일이나 실행 시에 오류가 발생할 수 있다.

◆ 에러와 경고

- 에러(error): 심각한 오류
- 경고(warning): 경미한 오류



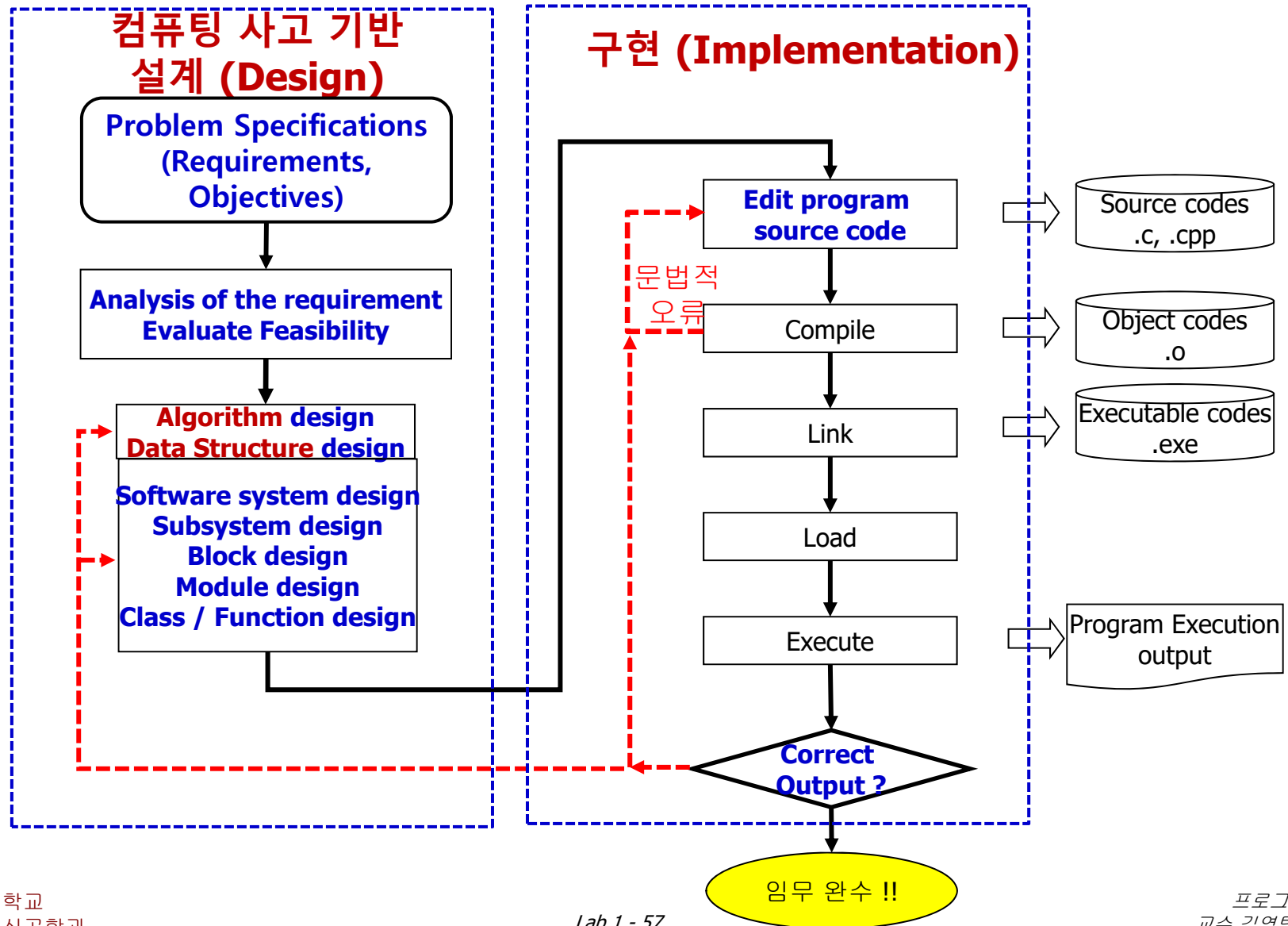
◆ 오류의 종류

- 컴파일 시간 오류: 대부분 문법적인 오류
- 실행 시간 오류: 실행 중에 0으로 나누는 연산 같은 오류
- 논리 오류: 논리적으로 잘못되어서 결과가 의도했던 대로 나오지 않는 오류





# 프로그램 개발 및 오류 수정 과정



# 오류 #1

```
#include <stdio.h>

int main(void)
{
    printf("Hello World!\n")
    return 0;
}
```

문장의 끝에 ;  
(세미콜론)이  
없음!!

오류가 발견된  
소스파일

오류가 발견된 줄 번  
호

return 앞에 ;을 빠뜨  
렸다는 의미이다.

```
1>----- 모두 다시 빌드 시작: 프로젝트: hello, 구성: Debug Win32 -----
1> hello.c
1>c:\users\chun\documents\visual studio 2010\projects\hello\hello\hello.c(7): error C2143: 구문 오류 :
';'이(가) 'return' 앞에 없습니다.
===== 모두 다시 빌드: 성공 0, 실패 1, 생략 0 =====
```



## 오류 #2

```
/* 에러가 발생하는 프로그램 */  
#include <stdio.h>  
  
int main(void)  
{  
    printf("Hello World!\n")  
    return 0;  
}
```

\*과 /이 떨어져  
있음 ->  
나머지 프로그램  
전체가  
주석처리 됨

```
1>----- 빌드 시작: 프로젝트: hello, 구성: Debug Win32 -----  
1> hello.c  
1>c:\Users\Wchun\documents\visual studio  
2010\projects\hello\hello\hello.c(9): fatal error C1071: 주석에서 예기치 않은  
파일의 끝이 나타났습니다.  
===== 빌드: 성공 0, 실패 1, 최신 0, 생략 0 =====
```

주석은 프로그램에 대한 설명글로서 /\* \*/ 안에 표시한다.



## 오류 #3

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    print("Hello World!");
```

```
    return 0;
```

```
}
```

함수이름을  
잘못 입력  
print() : X  
printf() : O

```
1>----- 빌드 시작: 프로젝트: hello, 구성: Debug Win32 -----
```

```
1> hello.c
```

```
1>c:\Users\Wchun\documents\visual studio 2010\projects\hello\hello\hello.c(6): warning C4013: 'print'  
이(가) 정의되지 않았습니다. extern은 int형을 반환하는 것으로 간주합니다.
```

```
1>hello.obj : error LNK2019: _print 외부 기호(참조 위치: _main 함수)에서 확인하지 못했습니다.
```

```
1>c:\Users\Wchun\Documents\Visual Studio 2010\Projects\hello\Debug\hello.exe : fatal error LNK1120:
```

```
1개의 확인할 수 없는 외부 참조입니다.
```

```
===== 빌드: 성공 0, 실패 1, 최신 0, 생략 0 =====
```



# 논리 오류

◆ 다음과 같은 출력을 가지는 프로그램을 작성하여 보자.



# 논리 오류가 존재하는 프로그램

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    printf("Hey!");
```

```
    printf("Good Morning");
```

```
    return 0;
```

```
}
```

줄이 바뀌지  
않았음!



# 논리 오류가 수정된 프로그램

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

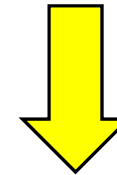
```
    printf("Hey! \n");
```

```
    printf("Good Morning \n");
```

```
    return 0;
```

```
}
```

논리 오류 수정!!



# 많이 발생하는 문법적 오류 (1)

## ◆ #include <stdio.h>를 하지 않았을 경우

- 잘못된 코드 작성 내용

```
main.cpp ×
(전역 범위)
void main()
{
    printf("Hello World\n");
}
```

- 올바른 코드작성 내용

```
main.cpp ×
(전역 범위)
#include <stdio.h>
void main()
{
    printf("Hello World\n");
}
```

- 에러문 출력 내용

```
오류 목록
▼ 오류 2개 | 경고 0개 | 메시지 0개
설명
1 error C3861: 'printf': 식별자를 찾을 수 없습니다.
2 IntelliSense: 식별자 "printf"이(가) 정의되어 있지 않습니다.
```





## 많이 발생하는 문법적 오류 (2)

### ◆ ; (semicolon)을 붙이지 않은 경우

- 잘못된 코드 작성 내용

```
main.cpp ㅅ X
(전역 범위)
#include <stdio.h>

void main()
{
    printf("Hello WorldWr")
}
```

- 올바른 코드 작성 내용

```
main.cpp ㅅ X
(전역 범위)
#include <stdio.h>

void main()
{
    printf("Hello Worldn");
}
```

- 에러문 출력 내용

```
오류 목록
▼ 2 오류 2개 | 경고 0개 | 메시지 0개
설명
1 error C2143: 구문 오류 : ';'이(가) ')' 앞에 없습니다.
2 IntelliSense: ';'가 필요합니다.
```



## 많이 발생하는 문법적 오류 (3)

### ◆ printf()나 scanf()에서 형식 지정자에 ""를 붙이지 않은 경우

#### ● 잘못된 코드 작성 내용

```
main.cpp [X]
(전역 범위)
#include <stdio.h>

void main()
{
    printf(Hello World\n);
}
```

#### 올바른 코드 작성 내용

```
main.cpp [X]
(전역 범위)
#include <stdio.h>

void main()
{
    printf("Hello World\n");
}
```

#### ● 에러문 출력 내용

오류 목록

오류 7개 | 경고 0개 | 메시지 0개

번호	설명
1	error C2065: 'Hello' : 선언되지 않은 식별자입니다.
2	error C2017: 이스케이프 시퀀스가 잘못되었습니다.
3	error C2146: 구문 오류 : ')'이(가) 'World' 식별자 앞에 없습니다.
4	error C2059: 구문 오류 : ' '
5	IntelliSense: 식별자 "Hello"이(가) 정의되어 있지 않습니다.
6	IntelliSense: ')'가 필요합니다.
7	IntelliSense: 인식할 수 없는 토큰입니다.



# 많이 발생하는 문법적 오류 (4)

## ◆ 함수 prototype 선언 시 발생할 수 있는 에러

### ● 잘못된 코드 작성 내용

```
main.cpp -> X
(전역 범위)
#include <stdio.h>

int Intsum(int operand1, int operand2);

void main()
{
    int a = 5;
    int b = 7;
    int result;

    result=Intsum(a, b);
}

int Intsum(int operand1, int operand2);
{
    return operand1 + operand2;
}
```

### 올바른 코드 작성 내용

```
main.cpp -> X
(전역 범위)
#include <stdio.h>

int Intsum(int operand1, int operand2);

void main()
{
    int a = 5;
    int b = 7;
    int result;

    result=Intsum(a, b);
}

int Intsum(int operand1, int operand2)
{
    return operand1 + operand2;
}
```

### ● 에러문 출력 내용

```
오류 목록
▼ 오류 4개 | 경고 0개 | 메시지 0개
설명
1 error C2144: 구문 오류 : void'은(는) ';' 다음에 와야 합니다.
2 error C2447: '{' : 함수 헤더가 없습니다. 이전 스타일의 형식 목록입니까?
3 IntelliSense: ';'가 필요합니다.
4 IntelliSense: 선언이 필요합니다.
```



# 많이 발생하는 문법적 오류 (5)

## ◆ 함수 proto type 선언 시 발생할 수 있는 외부참조 에러

### ● 잘못된 코드 작성 내용

```
main.cpp [X]
(전역 범위)
#include <stdio.h>

int Intsum(int operand1, int operand2);

void main()
{
    int a = 5;
    int b = 7;
    int result;

    result=Intsum(a, b);
}

int sum(int operand1, int operand2)
{
    return operand1 + operand2;
}
```

### 올바른 코드 작성 내용

```
main.cpp [X]
(전역 범위)
#include <stdio.h>

int Intsum(int operand1, int operand2);

void main()
{
    int a = 5;
    int b = 7;
    int result;

    result=Intsum(a, b);
}

int Intsum(int operand1, int operand2)
{
    return operand1 + operand2;
}
```

### ● 에러문 출력 내용

```
오류 목록
[X] 오류 2개 | 경고 0개 | 메시지 0개
설명
1 error LNK2019: "int __cdecl Intsum(int,int)" (?Intsum@@YAHHH@Z) 외부 기호(참조 위치: _main 함수)에서 확인하지 못했습니다.
2 error LNK1120: 1개의 확인할 수 없는 외부 참조입니다.
```



# 많이 발생하는 문법적 오류와 오류 메시지

C프로그램 문법적 오류	오류 메시지	올바른 입력
#include <stdio.h>	C1021 'include'전처리 명령이 잘못되었습니다.	#include<stdio.h>
#include<stdjo.h>	E1696 파일 소스(를) 열 수 없습니다. "stdjo.h" E0020 식별자 "printf"이(가) 정의되어 있지 않습니다. E0020 식별자 "scanf"이(가) 정의되어 있지 않습니다. C1083 포함 파일을 열 수 없습니다. 'stdjo.h':No such file or directory	#include<stdio.h>
#include stdio.h	C2006 '#include': "FILENAME" 또는 <FILENAME>이 필요합니다. C1083 포함 파일을 열 수 없습니다. ":No such file or directory	#include<stdio.h>
int x double y;	C2144: 구문오류: double은 ';'다음에 와야 합니다.	int x; double y;
main() { ... }	C4430 형식 지정자가 없습니다. Int로 가정합니다. 참고: C++에서는 기본 int를 지원하지 않습니다.	int main() { ... }
void main() int x;	E0130 '{'가 필요합니다. C2144: int는 ';'다음에 와야 합니다.	void main() { int x; }



# Visual Studio 단축키

## ◆ 단축키

단축키	기능
Ctrl+F5	프로젝트 빌드 및 실행
Ctrl+F7	프로젝트 빌드
F5	디버깅 모드 진입
F9	커서가 있는 부분에 중단점 생성 (왼쪽 편 빨간색 점)
F10	디버깅 모드 진입 후 코드 라인 바이 라인 진행 (함수 내 진입 x)
F11	디버깅 모드 진입 후 코드 라인 바이 라인 진행 (함수 내 진입 o)

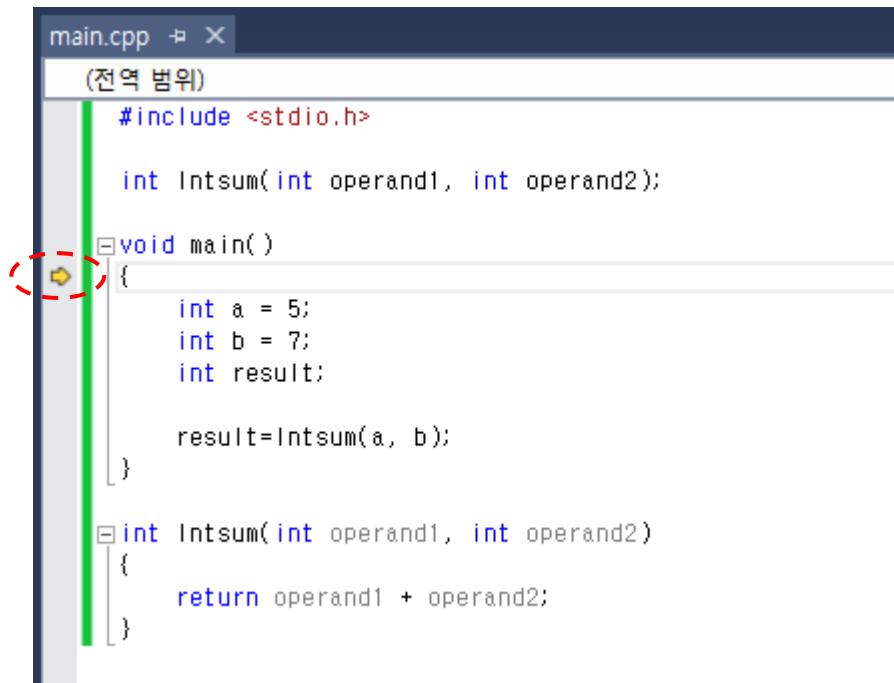




# Visual Studio Debugging tool 사용법(2)

## ◆ Debugging 모드 실행 방법(2)

- F5를 누른 후 F10(Step Over)을 누르게 되면 main문 상단에 노란색 커서가 생김.



```
main.cpp -> X
(전역 범위)
#include <stdio.h>

int Intsum(int operand1, int operand2);

void main()
{
    int a = 5;
    int b = 7;
    int result;

    result=Intsum(a, b);
}

int Intsum(int operand1, int operand2)
{
    return operand1 + operand2;
}
```

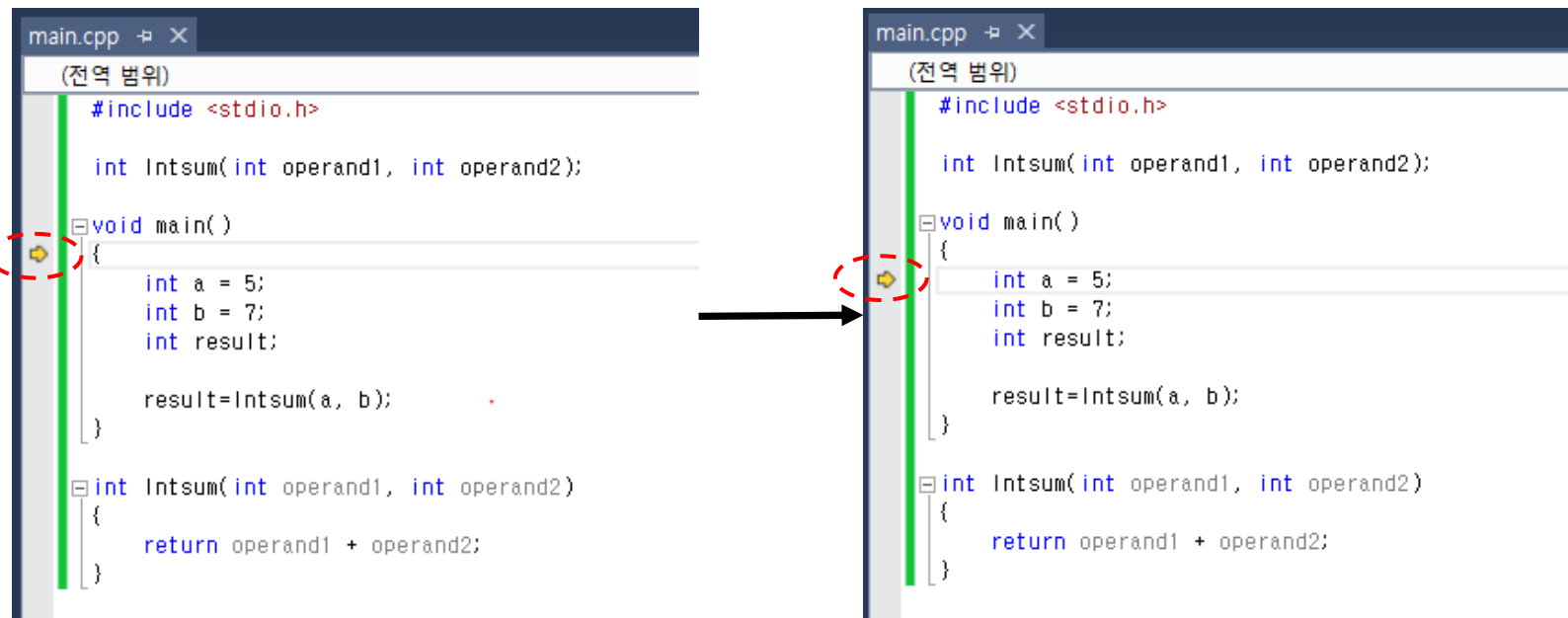




# Visual Studio Debugging tool 사용법(3)

## ◆ Debugging 모드 실행법(3)

- 이후 F10 키를 이용하여 한 문장씩 실행하여 순차적인 코드 진행을 확인 할 수 있음.



# Visual Studio Debugging tool 사용법(4)

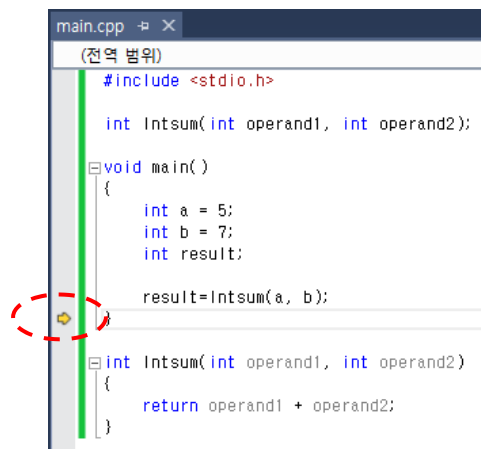
## ◆ 함수내부 Debugging (F11)

- F10 키를 이용하여 논리적 코드를 진행 하던 중 함수호출 라인을 만나게 되면 두 가지 선택지가 존재 하게 됨.

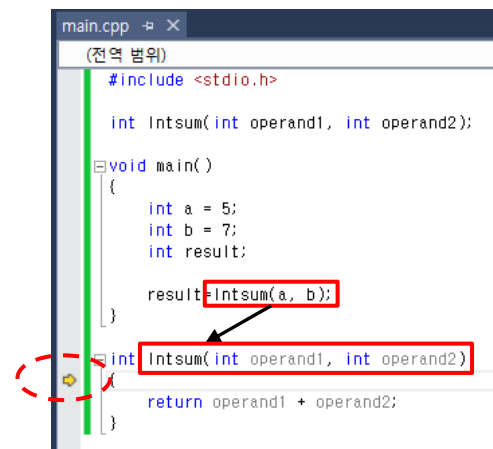
1. 함수호출을 생략하고 바로 다음 line 진행.
2. 함수호출을 생략하지 않고 함수 내부 코드를 진행.

1.의 경우 기존의 **F10(Step Over)**키를 다시 누르면 함수호출을 생략하고 다음 line으로 진행함

2.의 경우 **F11 (Step Into)**키를 누르면 바로 다음라인을 진행하지 않고 함수내부로 커서가 이동함



F10을 누른 경우



F11을 누른 경우



# Visual Studio Debugging tool 사용법(5)

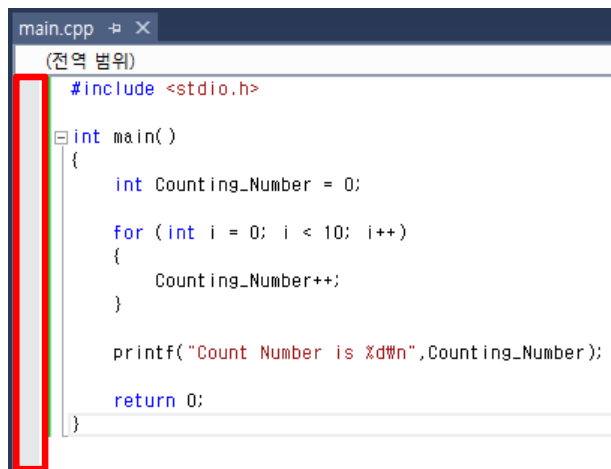
## ◆ Setting Break Points (F9)

### ● Break Point란?

순차적인 코드진행을 뛰어넘어서 사용자가 원하는 지점을 중단점으로 지정함으로 해서 Debugging을 도와주는 강력한 tool

### ● Break Point 설정하는 법

- 붉은 box로 표시된 부분을 마우스로 클릭하여 원하는 지점에 Break Point 설정
- 커서가 깜박이고 있는 지점에서 F9를 누르면 해당 라인에 Break Point 설정
- 2개 이상의 복수개의 Break Point도 설정가능



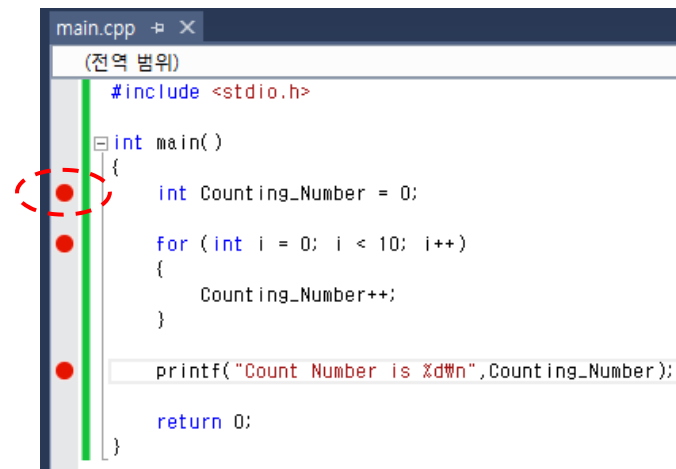
```
main.cpp [X]
(전역 범위)
#include <stdio.h>

int main()
{
    int Counting_Number = 0;

    for (int i = 0; i < 10; i++)
    {
        Counting_Number++;
    }

    printf("Count Number is %d\n", Counting_Number);

    return 0;
}
```



```
main.cpp [X]
(전역 범위)
#include <stdio.h>

int main()
{
    int Counting_Number = 0;

    for (int i = 0; i < 10; i++)
    {
        Counting_Number++;
    }

    printf("Count Number is %d\n", Counting_Number);

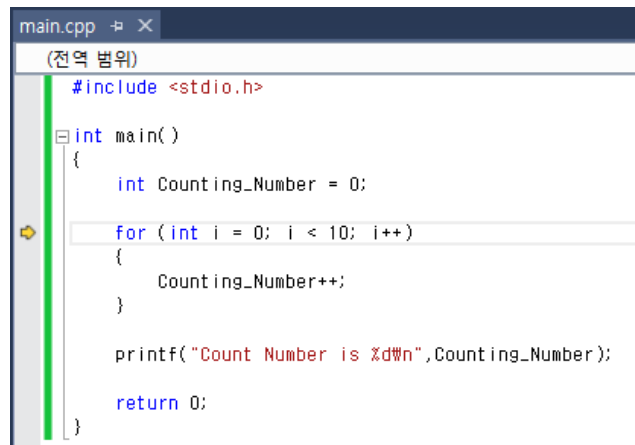
    return 0;
}
```

Break Point

# Visual Studio Debugging tool 사용법(6)

## ◆ Break Points 의 필요성

- 순차적 코드진행을 skip하기 때문에 더욱 빠른 Debugging 속도를 보장해줌.  
ex)

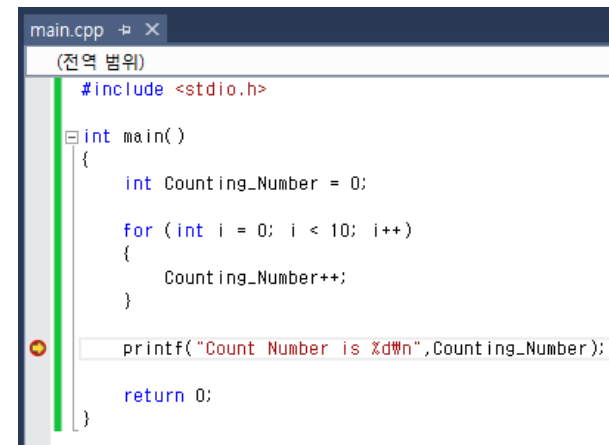


```
main.cpp [X]
(전역 범위)
#include <stdio.h>

int main()
{
    int Counting_Number = 0;
    for (int i = 0; i < 10; i++)
    {
        Counting_Number++;
    }

    printf("Count Number is %d\n", Counting_Number);

    return 0;
}
```



```
main.cpp [X]
(전역 범위)
#include <stdio.h>

int main()
{
    int Counting_Number = 0;
    for (int i = 0; i < 10; i++)
    {
        Counting_Number++;
    }

    printf("Count Number is %d\n", Counting_Number);

    return 0;
}
```

왼쪽 그림처럼 Break Point를 설정하지 않고 Debugging을 한다면 for loop이 종료될 때까지 F10을 눌러 Debugging을 진행해야 함.

하지만 오른쪽 그림처럼 for loop 이후에 break point를 설정해두면 for loop을 진행하지 않고 F5를 눌러 바로 break point로 이동 가능



# Visual Studio Debugging tool 사용법(7)

## ◆ 지역 변수 (Local Variables) 값 확인하기

- 지역 변수 출력창에서 지역 변수의 값 확인

Local Area

The screenshot shows the Visual Studio IDE with a C++ file named `main.cpp` open. The code is as follows:

```
#include <stdio.h>

int main()
{
    int Counting_Number = 0;

    for (int i = 0; i < 10; i++)
    {
        Counting_Number++;
    }

    printf("Count Number is %d\n", Counting_Number);

    return 0;
}
```

A red dashed box on the left side of the code editor, labeled "Local Area", highlights the scope of the `main()` function. The "지역" (Local) window is open, displaying the following table:

이름	값	형식
i	0	int
Counting_Number	0	int

At the bottom of the "지역" window, there are tabs for "자동" (Automatic), "지역" (Local), and "조사식 1" (Watch 1).



# Visual Studio Debugging tool 사용법(8)

## ◆ 조사식

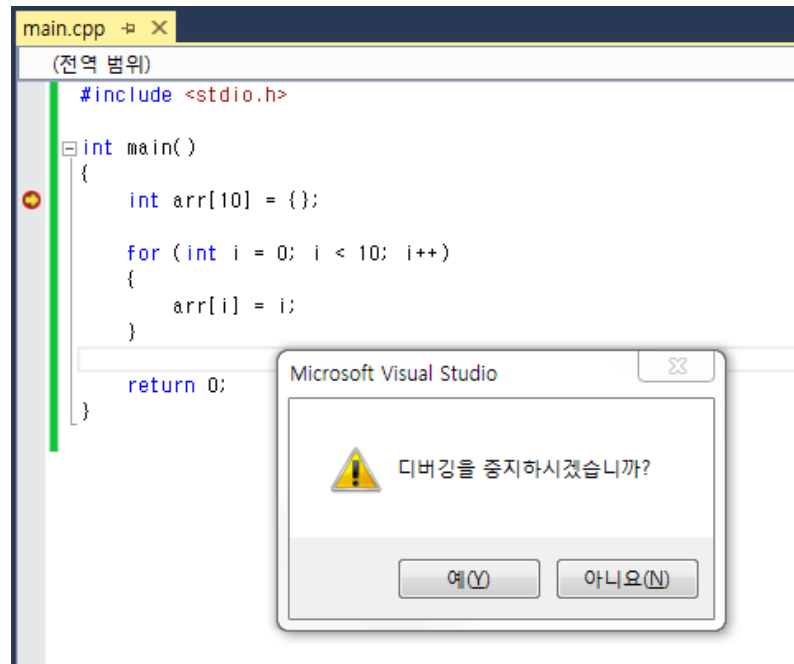
- 조사식 창에서 감시 대상 항목들을 추가하고, 지정된 변수들 값의 변화를 관찰 할 수 있음



# Visual Studio Debugging tool 사용법(8)

## ◆ Debugging 모드 종료

- Ctrl+F7(F7:키보드 셋팅 6.0 기준) 을 누르면 디버깅 모드 종료



## **실습 1 정리 및 Oral Test**



# 실습 1 종합정리

## ◆ Visual studio

- Visual studio community 2019

## ◆ C 프로그램의 기본적인 소스코드 구조

- Comments (주석)
- main() 함수
- 입력 및 출력
- Subroutine 함수

## ◆ Debugging

- break point 설정
- trace (1) : F10(Step Over)
- trace (2) : F11 (Step Into)



# Oral Test 1

**Q1.1** 소프트웨어 개발단계에서의 설계 및 구현 상세 절차에 대하여 설명하고, 유사코드 (pseudo code)가 무엇이며, 언제 작성하는 것인가에 대하여 설명하라.

**(Key points:** 설계 단계에서 알고리즘과 자료구조의 설계, 구현단계에서의 소스코드 작성, 논리적 오류 디버깅 절차를 상세하게 설명할 것.)

**Q1.2** C program의 컴파일 단계에서 발생할 수 있는 compile error message에는 어떤 종류들이 있는지에 대하여 20가지 이상 예를 들어 표로 만들고 원인과 오류 수정 방법에 대하여 설명하라.

**(Key points:** 소스코드 상에서 발생할 수 있는 다양한 오류를 미리 열거하고, 이 오류에서 어떤 오류 메시지가 발생하는가를 확인할 것.)



# Oral Test 1

## Q1.3 Visual studio의 debugger 기능

인 F5 (start debugging), F9 (break point), F10 (trace, step-over), F11 (trace, step-in) 기능에 대하여 설명하라.

**(Key points:** 실습 1의 프로그램을 디버깅 할 때 각 기능들을 어떻게 사용하게 되는지를 설명할 것.)

## Q1.4 Visual studio의 debugger 기능을 사용하여 프로그램의 실행 중간 단계에서 지역 변수들의 값이 어떻게 변화하는지를 확인할 수 있는 방법에 대하여 설명하라.

**(Key points:** 실습 1의 프로그램 실행에서 최소값, 최대값이 어떻게 변화되며, 최종적으로 어떤 값이 되는가를 확인할 수 있는 방법에 대하여 설명할 것.)

