

# C프로그래밍

## 2 프로그램 기본 구성

01

# 프로그램의 기본 구성

# C언어의 기본단위인 '함수'의 이해 (1/2)

- **C언어의 기본단위는 함수**

- ✓ C언어에서 프로그램 작성 : 함수를 만들고, 만들어진 함수의 실행순서를 결정하는 것

- **함수의 기본특성**

- ✓ 수학적으로 함수에는 입력과 출력이 존재

- **C언어의 함수**

- ✓ C언어의 함수에도 입력과 출력이 존재

# C언어의 기본단위인 '함수' 의 이해 (2/2)

- **C언어의 함수와 관련된 용어의 정리**

- ✓ 함수의 정의 : 만들어진 함수, 실행이 가능한 함수를 일컬음
- ✓ 함수의 호출 : 함수의 실행을 명령하는 행위
- ✓ 인자의 전달 : 함수의 실행을 명령할 때 전달하는 입력 값

# 예제에서 함수는 어디에? (1/2)

- **프로그램의 시작**

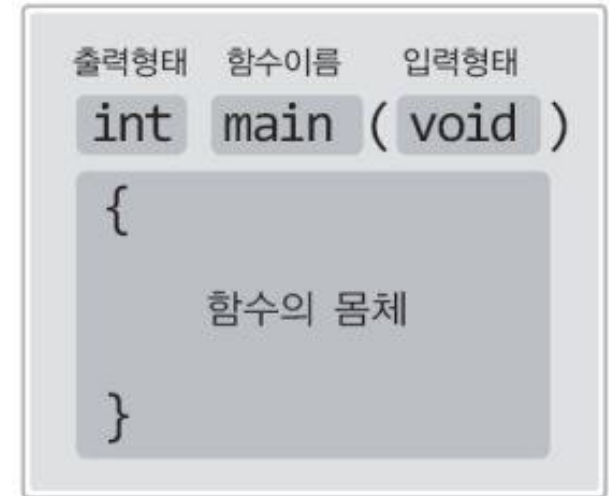
- ✓ 첫 번째 함수가 호출이 되면서 프로그램은 시작

- **제일 먼저 호출되는 함수는?**

- ✓ main이라는 이름의 함수!
- ✓ C언어로 구현된 모든 프로그램은

main이라는 이름의 함수를 반드시 정의

- ✓ main 함수가 자동으로 호출이 되면서 프로그램은 실행



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

↓  
순차적으로 실행

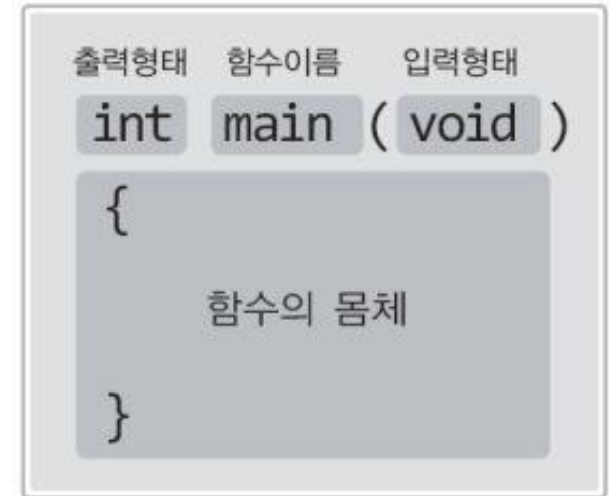
# 예제에서 함수는 어디에? (2/2)

## • 함수의 기능

- ✓ 함수의 기능은 중괄호 안에 표현
- ✓ 함수의 기능 : 함수의 몸체

## • C언어의 함수에 표시가 되는 세 가지

- ✓ 함수의 이름
- ✓ 출력형태 : 실행의 결과!
  - 일반적으로 반환형(return type)이라 함
- ✓ 입력형태 : 함수를 호출할 때 전달하는 입력 값의 형태
  - 일반적으로 parameter라 함



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

순차적으로 실행

# 세미콜론

- **함수 내 문장의 끝 : 세미콜론 문자 ;**

- ✓ 세미콜론은 문장의 끝을 표현

- **열 줄에 표현된 코드는 열 개의 문장인가?**

- ✓ 하나의 문장이 둘 이상의 줄에 표시될 수도 있고, 한 줄에 둘 이상의 문장이 표시될 수도 있음
- ✓ 다음 세 가지 경우는 동일한 프로그램

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

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

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



# 소스코드 더 살펴보기

```
1 #include <stdio.h> 헤더파일 선언문
  int main(void)
  {
    처음 보는 함수의 호출문
2   printf("Hello world! \n");
3   return 0;
  }
```

## • 표준함수

- ✓ 기본적으로 제공이 되는 함수!
- ✓ printf 함수는 표준함수

## • 표준 라이브러리

- ✓ 표준함수들의 모임
- ✓ printf 함수는 표준 라이브러리의 일부

### 1 #Include <stdio.h>

- ✓ stdio.h 파일의 내용을 이 위치에 가져다 놓아라!
- ✓ printf 함수의 호출을 위해서 선언해야 하는 문장
- ✓ stdio.h 파일에는 printf 함수호출에 필요한 정보 존재

### 2 printf("Hello world! \n");

- ✓ printf라는 이름의 함수를 호출하는 문장
- ✓ 인자는 문자열 "Hello world! \n"
- ✓ 인자는 소괄호를 통해서 해당 함수에 전달

### 3 return 0;

- ✓ 함수를 호출한 영역으로 값을 전달(반환)
- ✓ 현재 실행중인 함수의 종료



# 주석 (1/3)

## • 주석의 이해

- ✓ 주석은 소스코드에 삽입된 메모
- ✓ 컴파일의 대상에서 제외

## • 주석의 필요성

- ✓ 자신과 다른 사람을 위해서 필수!

✓ 블록 단위 주석

한 행의 주석처리

```
/* 주석처리 된 문장 */
```

```
/*  
    주석처리 된 문장1  
    주석처리 된 문장2  
    주석처리 된 문장3  
*/
```

여러 행의 주석처리

✓ 행 단위 주석

```
// 주석처리 된 문장1  
// 주석처리 된 문장2  
// 주석처리 된 문장3
```

한 행 단위로의 주석처리

주석을 다는 방식은

프로젝트 별로 팀원과 상의하여 결정하게 된다.

## 주석 (2/3)

```
/*
제 목: Hello world 출력하기
기 능: 문자열의 출력
파일이름: HelloComment.c
수정날짜: 2014. 07. 15
작성자: 윤성우
*/
#include <stdio.h>    // 헤더파일 선언

int main(void)    // main 함수의 시작
{
    /*
    이 함수 내에서는 하나의 문자열을 출력한다.
    문자열은 모니터로 출력된다.
    */
    printf("Hello world! \n");    // 문자열의 출력
    return 0;    // 0의 반환
}    // main 함수의 끝
```

과도하게 처리된 주석(주석도 과하면 좋지 않다)!

주석을 다는 방법을 소개하기 위한 예제일 뿐이다.

## 주석 (3/3)

```
1.  /*
2.     주석처리 된 문장1
3.     /* 단일 행 주석처리 */
4.     주석처리 된 문장2
5.  */
```

잘못 달린 주석(컴파일 시 오류 발생)

```
1.  /*
2.     주석처리 된 문장1
3.     // 단일 행 주석처리
4.     주석처리 된 문장2
5.  */
```

잘 달린 주석(컴파일 시 오류 발생하지 않음)

주석을 달다 보면 주석이 겹치는(중첩되는) 경우가 발생하기도 한다. 그런데 블록 단위 주석은 겹치는 형태로 달 수 없다.

02

# 프로그램의 작성 순서 복습

# 1. 두 번째 프로젝트 만들기 (1/4)

## 01 Visual Studio를 실행

✓ 프로젝트 이름은 'Second'



그림 2-1 프로그램 작성 순서

# 1. 두 번째 프로젝트 만들기 (2/4)

- 02 [시작 화면] 오른쪽 아래 [새 프로젝트 만들기]를 클릭 →
- [모든 언어]를 [C++] 선택 → [Windows 데스크톱 마법사] 선택
- <다음> 클릭 → 프로젝트 이름을 'Second'로 입력
- 위치는 'C:\₩CookC'를 입력하거나 선택
- '솔루션 및 프로젝트를 같은 디렉터리에 배치'에 체크 → <만들기> 클릭

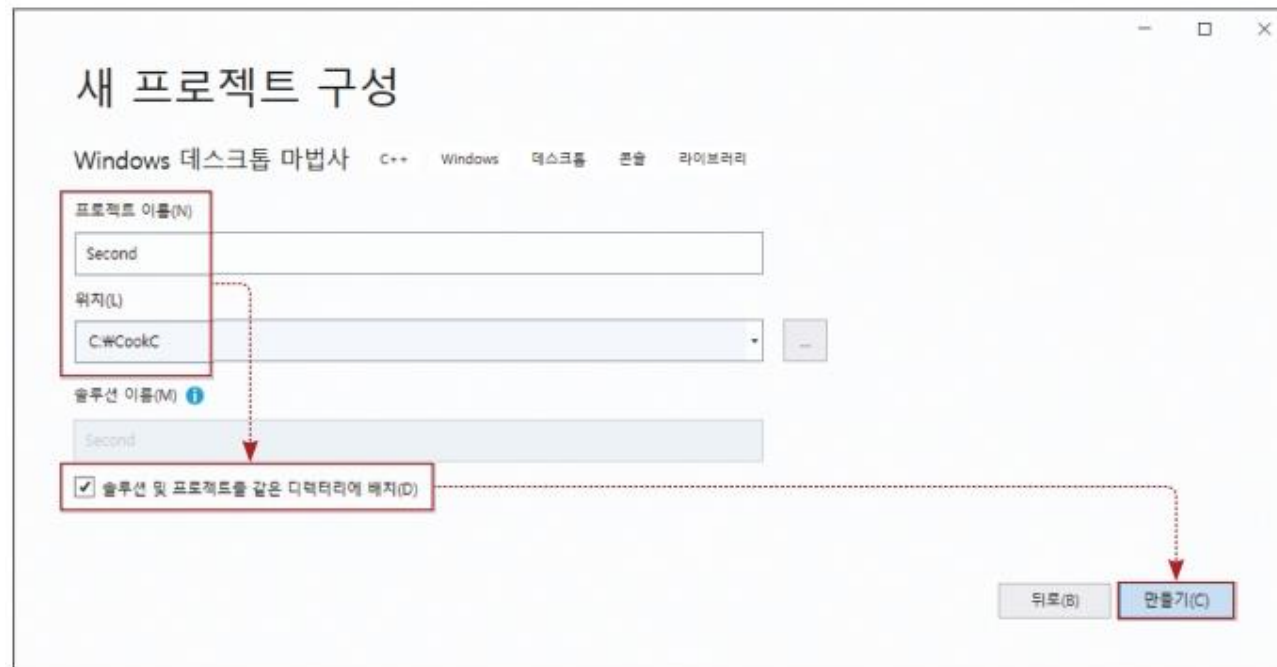


그림 2-2 새 프로젝트인 Second 생성



# 1. 두 번째 프로젝트 만들기 (3/4)

03 [Windows 데스크톱 프로젝트] 창에서 애플리케이션 종류는 '콘솔 애플리케이션(.exe)'을 선택하며, 추가 옵션으로 '빈 프로젝트'에 체크하고 <확인>을 클릭

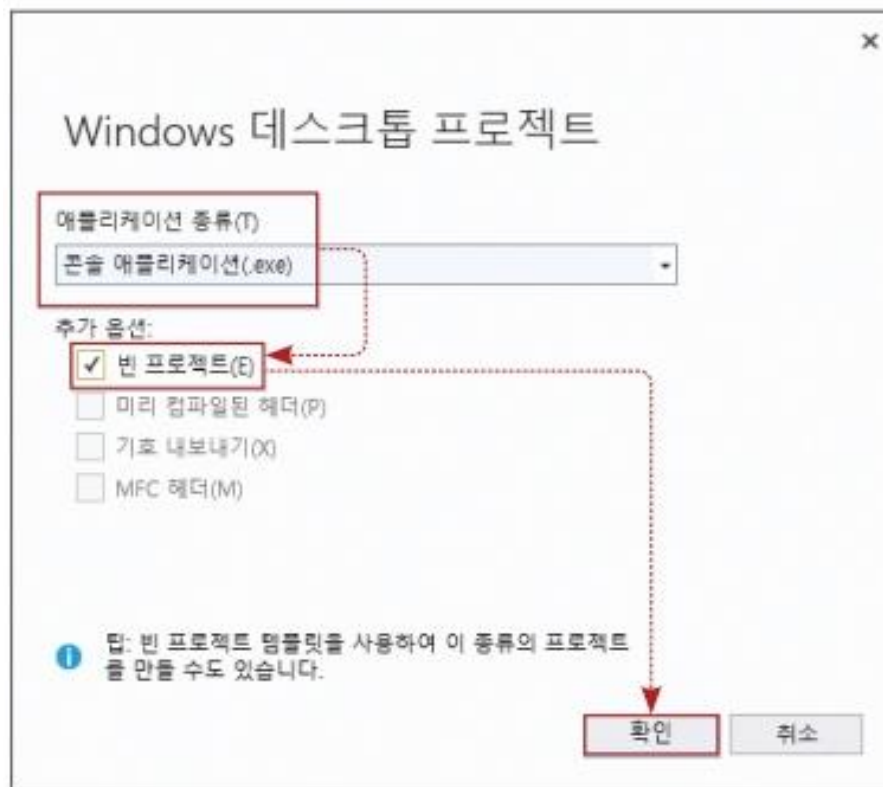


그림 2-3 응용 프로그램 설정



# 1. 두 번째 프로젝트 만들기 (4/4)

## 04 다음과 같이 빈 프로젝트가 완성, 이 프로젝트의 이름은 'Second'

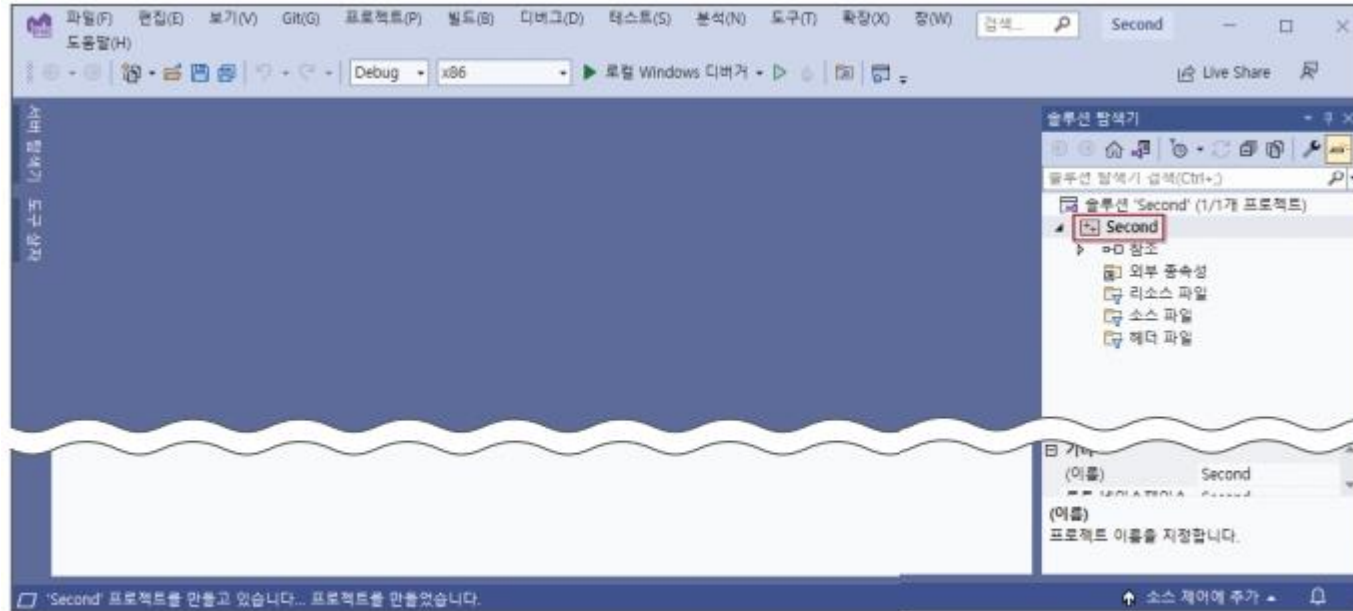


그림 2-4 생성된 Second 프로젝트

## 05 메뉴의 [파일] - [끝내기]를 선택하여 Visual Studio를 종료

## 2. 프로그램 코딩 (1/3)

01 Visual Studio를 실행

02 Visual Studio의 오른쪽 [시작] 부분의 [프로젝트 또는 솔루션 열기]를 선택한 후 앞서 작업했던 'C:\₩CookC₩Second' 폴더의 'Second.sln' 파일을 선택

03 오른쪽 [솔루션 탐색기]의 프로젝트 이름(현재 Second) 아래 '소스 파일' 폴더에서 마우스 오른쪽 버튼을 클릭하여 [추가] - [새 항목]을 선택

04 [새 항목 추가] 창의 왼쪽에서 [Visual C++]를 선택한 후, 오른쪽의 'C++ 파일(.cpp)'을 선택한 상태에서 이름에 'Second.c'를 입력하고 <추가>를 클릭

05 100과 50의 더하기, 빼기, 곱하기, 나누기를 수행하는 프로그램을 [기본 2-1]과 같이 코딩

## 2. 프로그램 코딩 (2/3)

### 기본 2-1 두 번째로 만드는 C 프로그램

2-1.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int a, b;
06     int result;
07
08     a=100;
09     b=50;
10
11     result = a + b;
12     printf(" %d + %d = %d \n", a, b, result);
13
14     result = a - b;
15     printf(" %d - %d = %d \n", a, b, result);
16
```

———— 계산할 두 수를 저장할 변수 a, b와 결과를 넣을 변수 result를 선언한다.

———— a에 100을, b에 50을 넣는다.

———— a와 b를 더한 결과를 result에 넣는다.

———— 모니터에 a, b, result를 출력한다.

———— a와 b를 뺀 결과를 result에 넣는다.

———— 모니터에 a, b, result를 출력한다.

## 2. 프로그램 코딩 (3/3)

```
17  result = a * b;      —— a와 b를 곱한 결과를 result에 넣는다.  
18  printf(" %d * %d = %d \n", a, b, result); —— 모니터에 a, b, result를 출력한다.  
19  
20  result = a / b;      —— a와 b를 나눈 결과를 result에 넣는다.  
21  printf(" %d / %d = %d \n", a, b, result); —— 모니터에 a, b, result를 출력한다.  
22 }
```

# 변수는 '값을 저장하는 그릇(또는 방)'

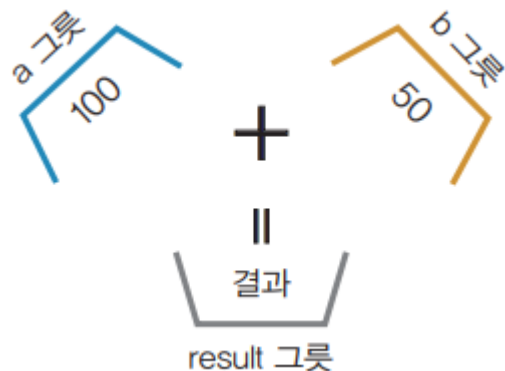
- 5행과 6행 : 변수(그릇)가 3개 등장



- 8행과 9행 : a 그릇에는 100을 넣고 b 그릇에는 50을 넣기



- 11행 : a 그릇 값과 b 그릇 값을 더한 결과를 result 그릇에 넣기



### 3. 빌드(컴파일/링크)

01 메뉴의 [빌드] - [솔루션 빌드]를 선택해서 프로젝트를 빌드

02 특별히 문제가 없다면 다음 그림과 같이 '성공 1, 실패 0, 최신 0, 생략 0'이 출력

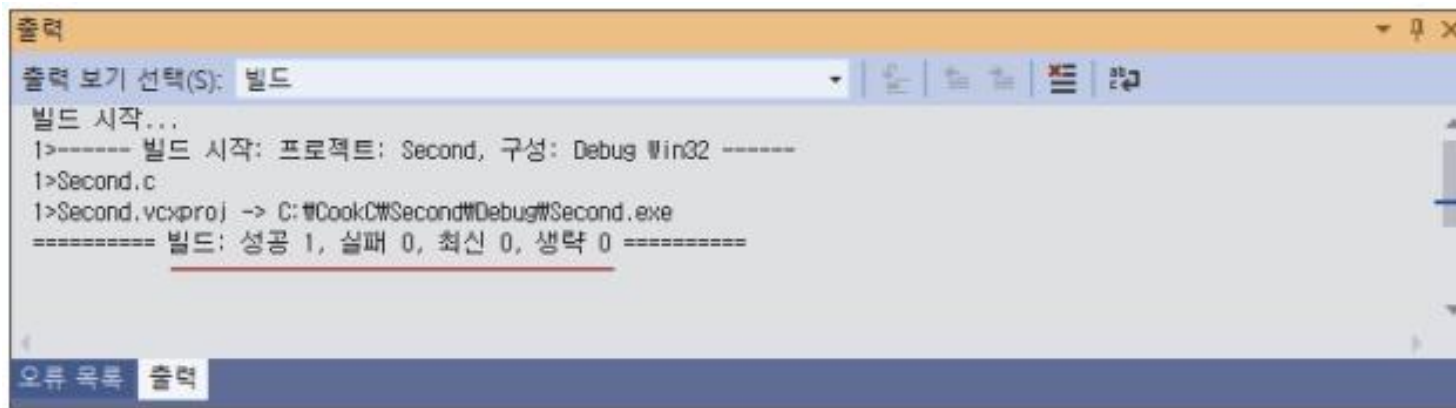


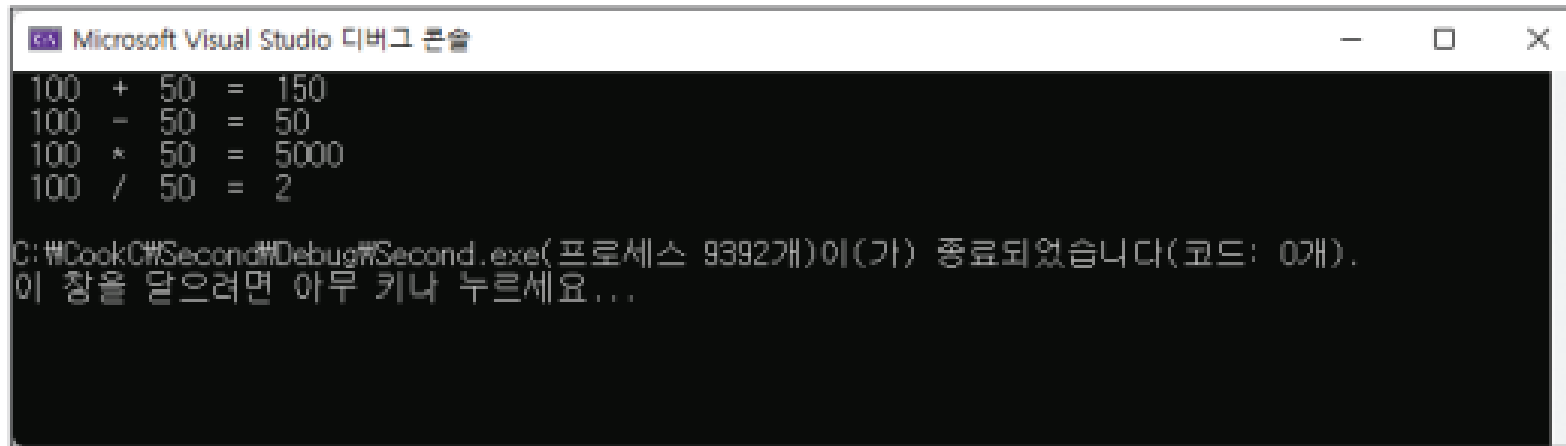
그림 2-6 프로젝트의 빌드 성공

03 만약, 실패가 나오면 소스에서 틀린 부분이 있다는 뜻이므로 소스에서 틀린 부분을 찾아 수정한 후 다시 빌드

## 4. 실행

### 01 [Ctrl] + [F5] 를 눌러서 실행

#### 실행 결과▼



```
Microsoft Visual Studio 디버그 콘솔
100 + 50 = 150
100 - 50 = 50
100 * 50 = 5000
100 / 50 = 2

C:\#CookC#\Second\#Debug\Second.exe( 프로세스 9392개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

### 02 결과를 확인한 후 아무 키나 누르면 결과 창이 닫힘



03

scanf( ) 함수 맛보기

# 값을 입력받는 scanf( ) 함수 (1/6)

## 여기서 잠깐 scanf( )와 scanf\_s( ) 함수

- C에서는 전통적으로 값을 입력받기 위해 scanf( ) 함수를 사용
- 마이크로소프트에서는 scanf( ) 함수 대신 scanf\_s( ) 함수의 사용을 적극 권장
- scanf( )와 scanf\_s( ) 함수의 용도는 동일하지만 scanf( ) 함수가 사용되는 소스의 1행에는 #define \_CRT\_SECURE\_NO\_WARNINGS라는 코드를 추가해야 함
- scanf\_s( ) 함수의 단점은 Visual C++에서만 사용할 수 있으며 다른 C 컴파일러는 인식 하지 못하는 것
- scanf\_s( )와 같이 기존의 함수 이름에 '\_s'가 붙은 새로운 함수는 보안이 한층 강화 된 개선된 함수

## 값을 입력받는 scanf( ) 함수 (2/6)

- <option #1> 1행에 #define  
\_CRT\_SECURE\_NO\_WARNINGS라는 코드를 추가

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4
```

# 값을 입력받는 scanf( ) 함수 (3/6)

## • <option #2> Visual Studio 관련 보안 옵션 끄기

✓ 프로젝트 혹은 소스파일 우클릭 -> [속성] -> [C/C++] -> [SDL 검사] -> [아니요]

The image shows the steps to disable SDL checks in Visual Studio. On the left, the 'Solution Explorer' (솔루션 탐색기) is open, and the 'Properties' (속성) button is highlighted at the bottom. A large blue arrow points from the Solution Explorer to the 'Properties' window on the right. The 'Properties' window shows the 'C/C++' settings, and the 'SDL 검사' (SDL Check) option is selected and highlighted with a red box. The 'SDL 검사' dropdown menu is open, showing '아니요 (/sdl-)' as the selected option.

설정	값
추가 BMI 디렉터리	
추가 모듈 종속성	
추가 헤더 단위 종속성	
소스의 모듈 종속성 검사	아니요
include 지시문을 import 지시문으로 변환	아니요
디버그 정보 형식	편집하며 계속하기 프로그램 데이터베이스(/ZI)
내 코드만 디버깅 지원	예(/JMC)
공용 언어 런타임 지원	
Windows 런타임 확장 사용	
시작 배너 표시 안 함	예(/nologo)
경고 수준	수준3(/W3)
경고를 오류로 처리	아니요(/WX-)
경고 버전	
진단 형식	열 정보(/diagnostics:column)
SDL 검사	아니요 (/sdl-)
다중 프로그래밍 컴파일	
주소 삭제기 사용	아니요
유사 항목 지원 사용(실험적)	아니요

**SDL 검사**  
추가 SDL(Security Development Lifecycle) 권장 검사입니다. 추가 보안 코드 생성 기능을 사용하고 추가 보안 관련 경로를 오류로 처리하도록 설정합니다. (/sdl, /sdl-)

확인 취소 적용(A)

# 값을 입력받는 scanf( ) 함수 (4/6)

## 01 값을 입력받는 scanf( ) 함수를 사용

✓ [기본 2-1]의 8행과 9행을 아래 표시된 부분과 같이 수정

응용 2-2 소스 수정하기(키보드로 값을 입력받음)

2-2.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     int a, b;
06     int result;
07
08     scanf("%d", &a);      ----- 키보드로 a에 들어갈 값을 입력받는다.
09     scanf("%d", &b);      ----- 키보드로 b에 들어갈 값을 입력받는다.
10
11     result = a + b;
12     printf(" %d + %d = %d \n", a, b, result);
13
14     ~~~ 이하는 [기본 2-1]의 14~21행과 동일함 ~~~
15     :
22 }
```

## 값을 입력받는 scanf( ) 함수 (5/6)

### 02 [Ctrl]+[F5]를 눌러서 빌드와 실행을 동시에 진행

실행 결과 ▼



### 03 숫자를 하나 입력하고 [enter]를 누름. 다시 숫자 하나를 입력하고 [enter]

실행 결과 ▼



# 값을 입력받는 scanf( ) 함수 (6/6)

## 04 프로그램을 좀 더 편하게 사용하기 위해 다음과 같이 수정

응용 2-3 소스 수정하기(도움말 출력)

2-3.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     int a, b;
06     int result;
07
08     printf("첫 번째 계산할 값을 입력하세요 ==> "); ----- 도움말을 화면에 출력한다.
09     __1__("%d", &a);
10
11     printf("두 번째 계산할 값을 입력하세요 ==> "); ----- 도움말을 화면에 출력한다.
12     __2__("%d", &b);
13
14     result = a + b ;
15     __3__(" %d + %d = %d \n",a ,b ,result);
16
17     ~~~ 이하는 [기본 2-1]의 14~21행과 동일함 ~~~
18     :
25 }
```

scanf 1 scanf 2 scanf 3 printf



04

printf( ) 함수의 기본 형태

# printf( ) 함수의 기본 사용법(정수) (1/4)

- printf( ) 함수는 화면(모니터)에 무언가를 보여주는 역할
- 모니터에 큰따옴표(“ ”) 안의 내용을 출력하라는 의미

<pre>printf("안녕하세요?");</pre>	실행 결과 ▶	안녕하세요?
<pre>printf("100"); printf("%d", 100);</pre>	실행 결과 ▶	100 100

- ✓ 실행 결과 100이 두 번 출력되었지만 둘은 완전히 다른 결과
- ✓ 첫 번째 printf ("100")의 100은 '숫자 100(백)'이 아닌 **‘글자 100(일영영)’**
  - printf( )에서 큰따옴표 안에 있는 값은 무조건 ‘글자’ 취급
- ✓ 두 번째 printf("%d", 100)의 결과로 나온 100은 '숫자 100(백)'을 의미
  - 서식(%d)이 지정 된 숫자는 숫자의 의미를 그대로 지니기 때문

# printf( ) 함수의 기본 사용법(정수) (2/4)

## 기본 3-1 printf( ) 함수 사용 예 1

3-1.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     printf("100+100"); ----- 모두 글자로 취급한다.
06     printf("\n");
07     printf("%d", 100+100); ----- 숫자를 계산해서 결과를 출력한다.
08     printf("\n");
09 }
```

### 실행 결과

```
100+100
200
```

# printf( ) 함수의 기본 사용법(정수) (3/4)

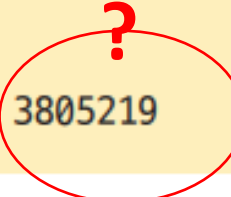
## 기본 3-2 printf( ) 함수 사용 예 2

3-2.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     printf("%d", 100, 200); —— %d는 1개, 숫자는 2개이다.
06     printf("\n");
07     printf("%d %d", 100); —— %d는 2개, 숫자는 1개이다.
08     printf("\n");
09 }
```

### 실행 결과

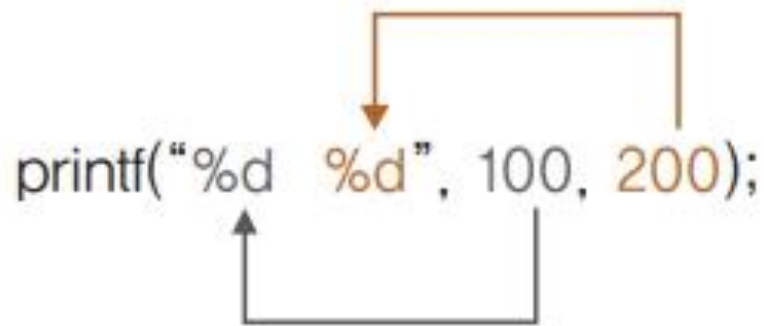
```
100
100 3805219
```



# printf( ) 함수의 기본 사용법(정수) (4/4)

## • 기본 3-2 수정 방법

- ✓ 숫자 2개를 출력하려면 %d 2개를 넣은 뒤 숫자 2개가 나와야 함  
→ 따라서 5행은 다음과 같이 고쳐야 함



The diagram illustrates the correspondence between format specifiers and arguments in the `printf` function. It shows the code `printf("%d %d", 100, 200);`. An arrow points from the first `%d` to the value `100`, and another arrow points from the second `%d` to the value `200`. The format specifiers and the arguments are highlighted in orange.

그림 3-1 서식과 숫자의 대응

### SELF STUDY

100과 200을 더한 결과가 나오도록 %d를 3개 사용해서 printf( )문을 만들어보자. 또한 나눗셈의 결과도 나오게 해보자. 즉 다음과 같이 출력되게 해야 한다.

결과\_ 100 + 200 = 300, 100 / 200 = 0.5

# 정수 외에 자주 사용되는 서식 (1/3)

## 기본 3-3 서식을 사용한 출력 예 1

3-3.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     printf("%d / %d = %d", 100, 200, 0.5);
06     printf("\n");
07 }
```

—— %d가 3개, 숫자도 3개이다.

### 실행 결과

100 / 200 = 0



그림 3-2 서식과 숫자의 불일치

## 정수 외에 자주 사용되는 서식 (2/3)

### • 기본 3-3 수정 방법

- ✓ 마지막 0.5에 대응되는 서식이 정수를 표현하는 %d이므로 0.5에서 소수 부분이 떨어져서 0만 출력

➔ 실수를 표현하는 서식을 사용해야 함

표 3-1 printf() 함수의 대표적인 서식

서식	서식값의 예	설명
%d, %x, %o	10, 100, 1234	정수(10진수), 정수(16진수), 정수(8진수)
%f 또는 %lf	0.5, 1.0, 3.14	실수(소수점이 있는 수)
%c	'a', 'b', 'F'	문자(한 글자이며 '로 둘러싸야 한다)
%s	"안녕", "abcdefg", "a"	문자열(한 글자 이상이며 "로 둘러싸야 한다)

- ✓ 이때 문자와 문자열은 구분할 필요가 있음
  - 문자는 꼭 작은따옴표(' ') 안에 한 글자만 들어 있어야 하고 문자열은 큰따옴표(" ") 안에 한 글자 이상이 들어 있어야 함



# 정수 외에 자주 사용되는 서식 (3/3)

## 응용 3-4 서식을 사용한 출력 예 2

3-4.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     printf("%d / __1__ = __2__ \n", 100, 200, 0.5); —— 정수 2개와 실수 1개를 출력한다.
06     printf(" __3__ %c \n", 'a', 'K'); —— 문자 2개를 출력한다.
07     printf("%s %s \n", "안녕", "c 언어"); —— 문자열 2개를 출력한다.
08 }
```

3% 3 4% 2 5% 1 6% 1 7% 1 8% 1 9% 1 10% 1 11% 1 12% 1 13% 1 14% 1 15% 1 16% 1 17% 1 18% 1 19% 1 20% 1 21% 1 22% 1 23% 1 24% 1 25% 1 26% 1 27% 1 28% 1 29% 1 30% 1 31% 1 32% 1 33% 1 34% 1 35% 1 36% 1 37% 1 38% 1 39% 1 40% 1 41% 1 42% 1 43% 1 44% 1 45% 1 46% 1 47% 1 48% 1 49% 1 50% 1 51% 1 52% 1 53% 1 54% 1 55% 1 56% 1 57% 1 58% 1 59% 1 60% 1 61% 1 62% 1 63% 1 64% 1 65% 1 66% 1 67% 1 68% 1 69% 1 70% 1 71% 1 72% 1 73% 1 74% 1 75% 1 76% 1 77% 1 78% 1 79% 1 80% 1 81% 1 82% 1 83% 1 84% 1 85% 1 86% 1 87% 1 88% 1 89% 1 90% 1 91% 1 92% 1 93% 1 94% 1 95% 1 96% 1 97% 1 98% 1 99% 1 100% 1

### 실행 결과

100 / 200 = 0.500000

a K

안녕 c 언어

05

printf( ) 함수의 서식 지정

# 자릿수를 맞춘 출력 (1/4)

## 기본 3-5 다양한 서식 활용 예 1

3-5.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     printf("%d\n", 123);
06     printf("%5d\n", 123);
07     printf("%05d\n", 123);
08
09     printf("%f\n", 123.45);
10     printf("%7.1f\n", 123.45);
11     printf("%7.3f\n", 123.45);
12
13     printf("%s\n", "Basic-C");
14     printf("%10s\n", "Basic-C");
15 }
```

—— 정수형 서식을 활용한다.

—— 실수형 서식을 활용한다.

—— 문자열형 서식을 활용한다.

### 실행 결과

123

123

00123

123.450000

123.5

123.450

Basic-C

Basic-C

## 자릿수를 맞춘 출력 (2/4)

- 정수형 데이터를 출력하기 위한 5~7행의 서식은 [그림 3-3]과 같이 나타낼 수 있음



그림 3-3 정수형 데이터 서식 지정

## 자릿수를 맞춘 출력 (3/4)

- 9~11행 실수형 데이터의 서식 지정은 [그림 3-4]와 같음

- ✓ 두 번째의 %7.1f는 소수점을 포함 한 전체 자리인 일곱 자리를 확보한 후에 소수점 아래는 한 자리만 차지한다는 의미
- ✓ 그러므로 소수점 위 정수 부분은 다섯 자리

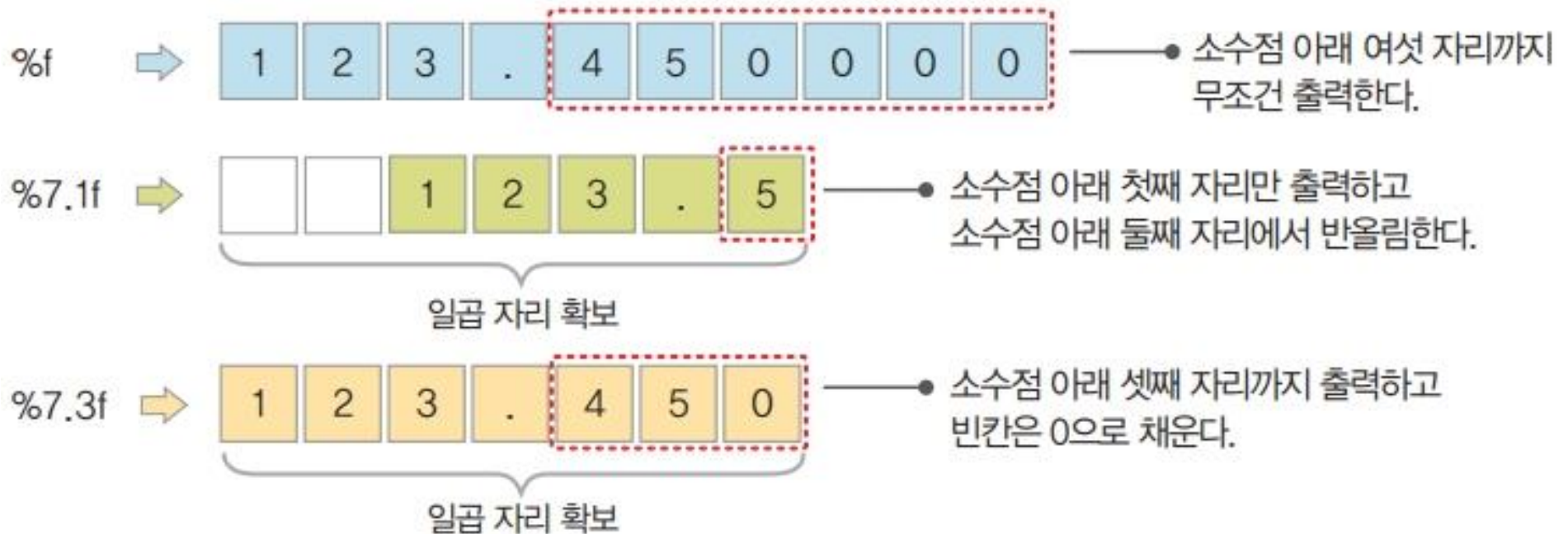


그림 3-4 실수형 데이터 서식 지정



## 자릿수를 맞춘 출력 (4/4)

- 13, 14행 문자열형 데이터의 서식도 오른쪽에 맞춰서 출력

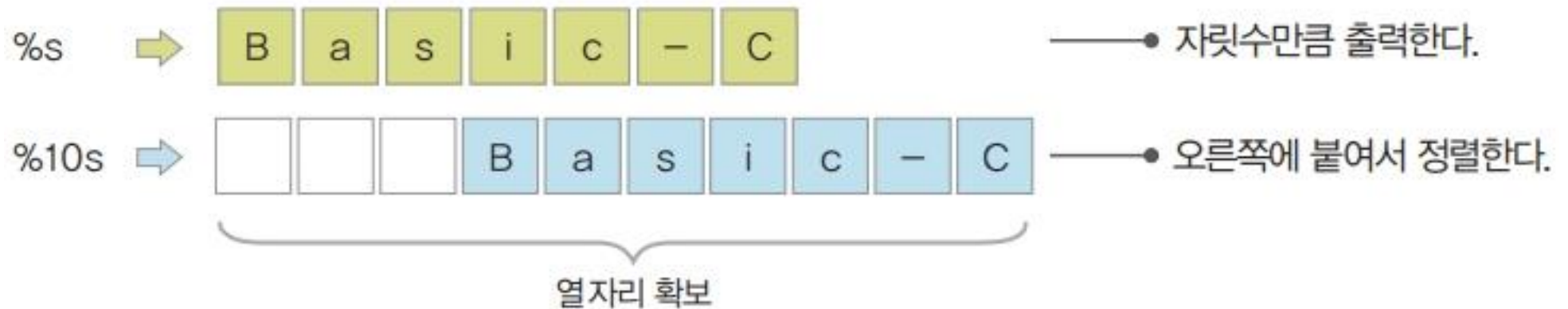






그림 3-5 문자열형 데이터 서식 지정

## 다양한 기능의 서식 문자 (1/2)

- 줄 바꿈의 역할을 하는 **\n**의 기능을 이미 사용해 보았음
- 서식 문자의 특징은 앞에 반드시 '**\**' 기호가 붙는다는 것

표 3-2 다양한 서식 문자

서식 문자	역할	비고
\n	새로운 줄로 이동한다.	 를 누른 효과와 동일하다.
\t	다음 탭으로 이동한다.	 을 누른 효과와 동일하다.
\b	뒤로 한 칸 이동한다.	 를 누른 효과와 동일하다.
\r	줄의 맨 앞으로 이동한다.	 을 누른 효과와 동일하다.
\a	'뽵' 소리를 낸다.	—
\\	\를 출력한다.	—
\'	'를 출력한다.	—
\"	"를 출력한다.	—



## 다양한 기능의 서식 문자 (2/2)

### 응용 3-6 다양한 서식 활용 예 2

3-6.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     printf("\n줄 바꿈\n연습 \n");
06     printf("\t탭키\t연습 \n");
07     printf("이것을\r덮어씹니다 \n");
08     printf("\a\a\a삐소리 3번 \n");
09     printf("글자가 \"강조\"되는 효과 \n");
10     printf("\\\\\\\\\\역슬래시 세 개 출력 \n");
11 }
```

#### 실행 결과

줄바꿈  
연습  
        탭키    연습  
덮어씹니다  
삐소리 3번  
글자가 "강조"되는 효과  
\\ \역슬래시 세 개 출력

----- 컴퓨터에 따라서 소리가 한 번만 날 수도 있다.

예 제

# 예제 제출

- **[기본 2-1], [예제-01], [예제-02] 제출**
  - ✓ 3개의 소스파일을 압축하여 제출
  - ✓ 파일명 : 학번
- **제출 : 강의 과제게시판 (2주차 수업예제 제출)**

# [예제 01] 숫자 4개를 더하는 프로그램

**예제 설명** 숫자 4개를 입력받아 그 합을 구하는 프로그램이다.

## 실행 결과

첫 번째 계산할 값을 입력하세요 => 100

두 번째 계산할 값을 입력하세요 => 200

세 번째 계산할 값을 입력하세요 => 300

네 번째 계산할 값을 입력하세요 => 400

$100 + 200 + 300 + 400 = 1000$

# [예제 01] 숫자 4개를 더하는 프로그램

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     int a, b, c, d;          —— 입력받을 변수 4개를 선언한다.
06     int result;
07
08     printf("첫 번째 계산할 값을 입력하세요 ==> ");
09     scanf("%d", &a);          —— 변수 a에 들어갈 값을 키보드로
10     printf("두 번째 계산할 값을 입력하세요 ==> ");          직접 입력한다.
11     scanf("%d", &b);          —— 변수 b에 들어갈 값을 키보드로
12     printf("세 번째 계산할 값을 입력하세요 ==> ");          직접 입력한다.
13     scanf("%d", &c);          —— 변수 c에 들어갈 값을 키보드로
14     printf("네 번째 계산할 값을 입력하세요 ==> ");          직접 입력한다.
15     scanf("%d", &d);          —— 변수 d에 들어갈 값을 키보드로
16                                     직접 입력한다.
17     result = a + b + c + d;    —— 변수 a, b, c, d의 값을 모두 더해
18                                     변수 result에 입력한다.
19     printf(" %d + %d + %d + %d = %d \n", a, b, c, d, result);
20 }
```

—— 변수 a, b, c, d와 result 값을 모니터에 출력한다.

## [예제 02] 간단한 사칙연산 계산기

- **사용자가 덧셈, 뺄셈, 곱셈, 나눗셈 중 하나를 선택해서 계산하는 프로그램**
  - ➔ scanf( ), printf( ) 사용
  - ➔ 나눗셈의 결과는 소수점 첫째 자리까지 출력
  - ➔ if문 사용 : 다음 페이지의 코드를 응용

### 실행 결과

```
첫 번째 계산할 값을 입력하세요 ==> 1000
<1>덧셈 <2>뺄셈 <3>곱셈 <4>나눗셈 ==> 2
두 번째 계산할 값을 입력하세요 ==> 122
1000 - 122 = 878
```



## [예제 02] 간단한 사칙연산 계산기

### - 참고 코드

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     int a, b;
06     int result;
07     int k;
08
09     // 계산할 수를 입력한다.
10
11     printf("<1>덧셈 <2>뺄셈 <3>곱셈 <4>나눗셈 ==> ");
12     scanf("%d", &k);
13
14     // 계산할 수를 입력한다.
15
16     if(k == 1) {
17         result = a + b;
18         printf(" %d + %d = %d \n", a, b, result);
19     }
```

—— 계산 방식을 선택할 변수를 선언한다.

—— 계산할 수를 입력한다.

—— 연산자를 선택한다(1: 덧셈, 2: 뺄셈, 3: 곱셈, 4: 나눗셈).

—— 계산할 수를 입력한다.

—— 입력한 k가 1이면 덧셈을 수행한다.



Q & A