



## CH02 변수와 자료형



# 이번 장에서 학습할 내용



- \* 변수와 상수의 개념 이해
- \* 자료형
- \* 정수형
- \* 실수형
- \* 문자형
- \* 기호 상수 사용
- \* 오버플로우와 언더플로우 이해

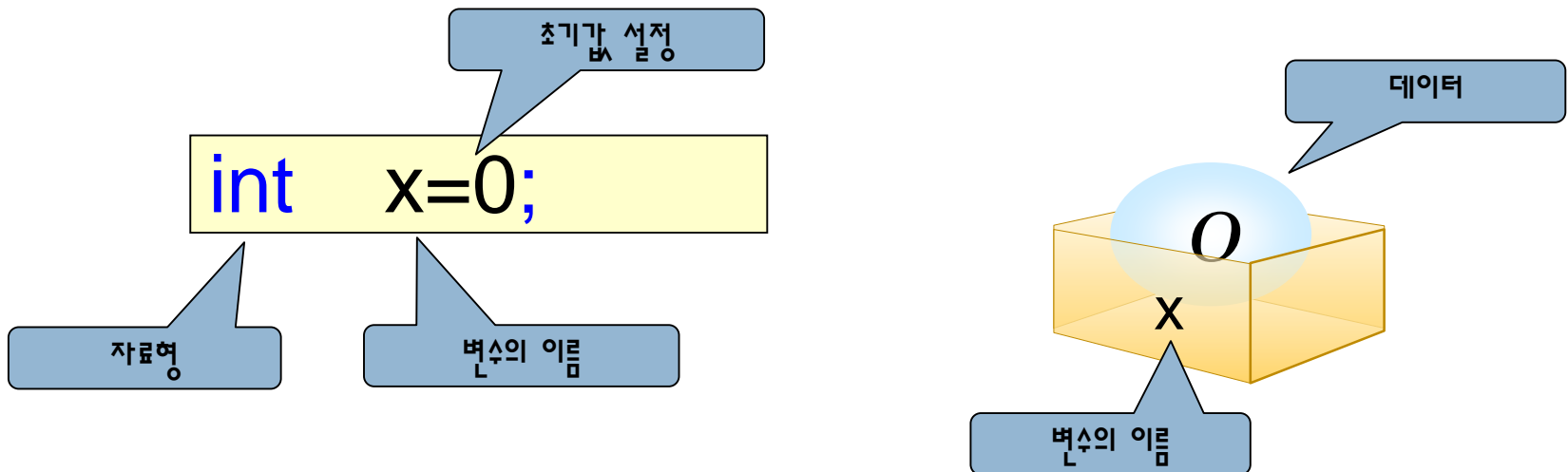
이번 장에서는 변수와  
각종 자료형을  
살펴봅니다.





# 변수의 종류

- 프로그램이 사용하는 데이터를 일시적으로 저장할 목적으로 사용하는 메모리 공간
- 변수는 데이터를 담는 상자로 생각할 수 있다.
- Visual studio 에서는 초기값을 주면서 변수를 선언해야 한다
  - 숫자는 보통 0의 값으로 초기화
- 변수는 필요한 만큼 생성해서 사용할 수 있다.





# 변수

- 컴퓨터 프로그램은 값을 저장하기 위하여 변수(variable)을 사용한다.
- 변수는 게임에서 점수를 저장하는데 사용될 수 있고, 대형 마트에서 우리가 구입한 물건들의 가격을 저장할 수도 있다.

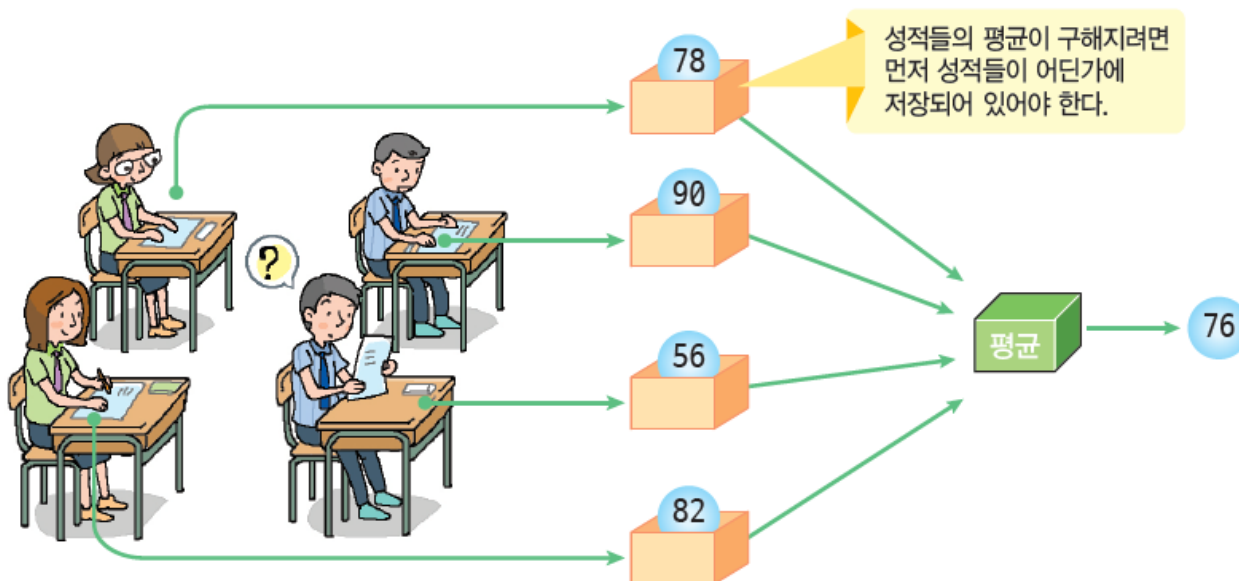


점수는 변수에  
저장된다.



# 변수가 왜 필요한가? #1

- 사용자에게서 받는 데이터를 저장하는 장소이다. – 변수가 없다면 사용자로부터 받은 데이터를 어디에 저장할 것인가?





# 변수가 왜 필요한가? #2

| 변수를 사용하지 않는 코드   | 변수를 사용하는 코드   |
|--|---|
| <pre>// 크기가 100x200인 사각형의 면적<br/>area = 100 * 200;</pre> | <pre>// 크기가 widthxheight인 사각형의 면적<br/>width = 100;<br/>height = 200;<br/>area = width * height;</pre> |

어떤 코드가 더  
유연한가요?  
변경에 더 잘 적응할 수  
있나요?





# 변수의 이름

## ● 식별자 만드는 규칙

- 식별자는 영문자와 숫자, 밑줄 문자 \_로 이루어진다.
- 식별자의 중간에 공백이 들어가면 안 된다.
- 식별자의 첫 글자는 반드시 영문자 또는 밑줄 기호 \_이어야 한다. 식별자는 숫자로 시작할 수 없다.
- 대문자와 소문자는 구별된다. 따라서 변수 `index`와 `Index`, `INDEX`은 모두 서로 다른 변수이다.
- C언어의 키워드와 똑같은 식별자는 허용되지 않는다.



# 키워드

- 키워드(keyword): C언어에서 고유한 의미를 가지고 있는 특별한 단어 예약어(reserved words) 라고도 한다.

|          |        |          |          |
|----------|--------|----------|----------|
| auto     | double | int      | struct   |
| break    | else   | long     | switch   |
| case     | enum   | register | typedef  |
| char     | extern | return   | union    |
| const    | float  | short    | unsigned |
| continue | for    | signed   | void     |
| default  | goto   | sizeof   | volatile |
| do       | if     | static   | while    |





# 변수의 이름

- `sum` *// 영문 알파벳 문자로 시작*
- `_count` *// 밑줄 문자로 시작할 수 있다.*
- `number_of_pictures` *// 중간에 밑줄 문자를 넣을 수 있다.*
- `King3` *// 맨 처음이 아니라면 숫자도 넣을 수 있다.*
- `2nd_base(X)` *// 숫자로 시작할 수 없다.*
- `money#` *// #과 같은 기호는 사용할 수 없다.*
- `double` *// double은 C 언어의 키워드이다.*

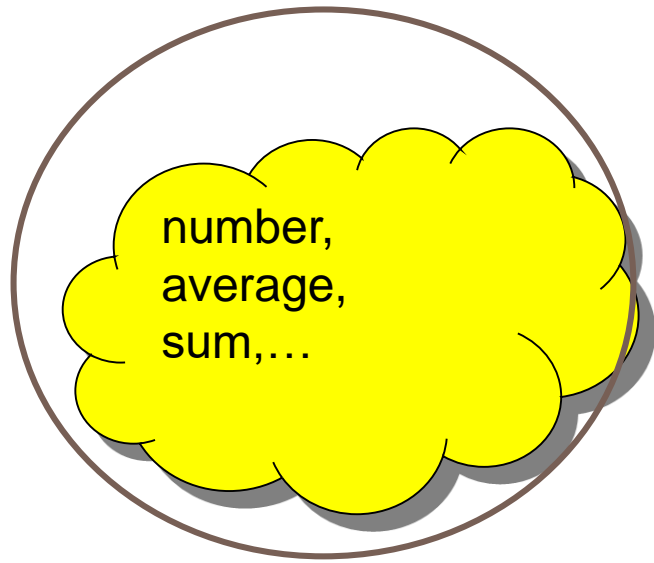


# 좋은 변수 이름

- 변수의 역할을 가장 잘 설명하는 이름
  - 밑줄 방식: `bank_account`
  - 단어의 첫번째 글자를 대문자: `BankAccount`



a, b, c, ,d,...

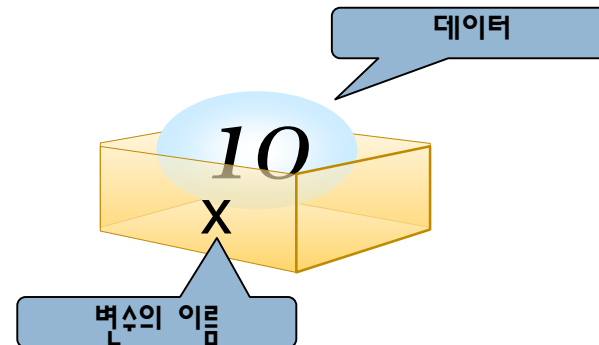
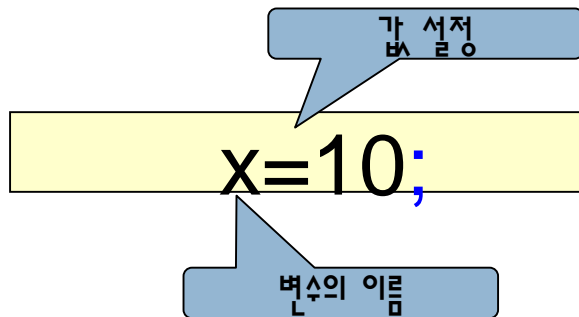


number,  
average,  
sum,...



# 변수의 사용

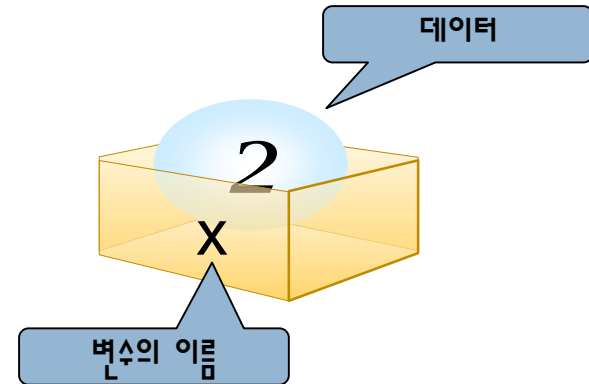
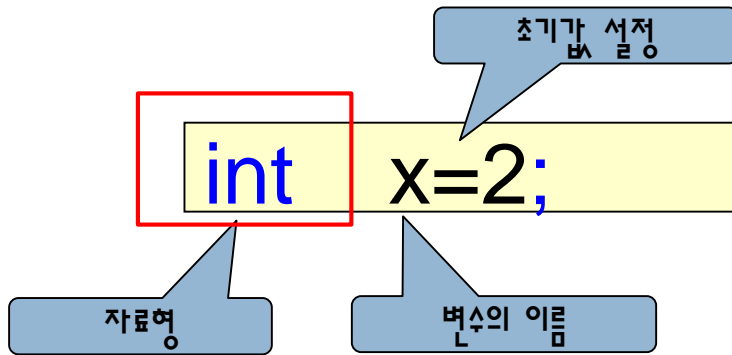
- 변수는 여러 개의 값을 변경해서 값을 저장 할 수 있다.
- 새로운 값을 저장 할 경우 이전의 값 없어 진다
- 형식 : 변수명 = 값



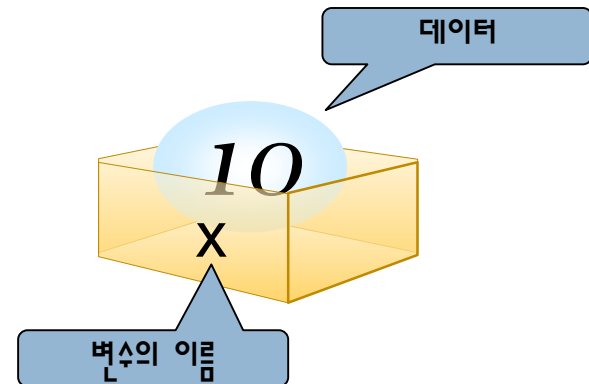
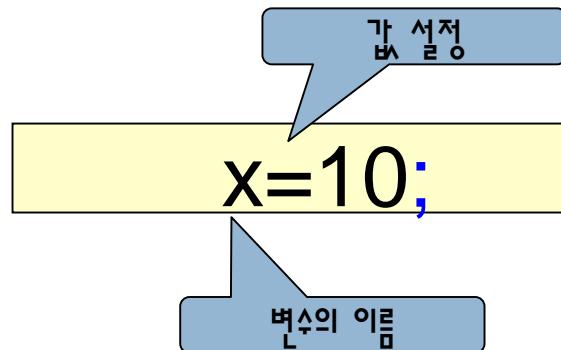


# 변수의 사용

## ● 1. 변수의 선언



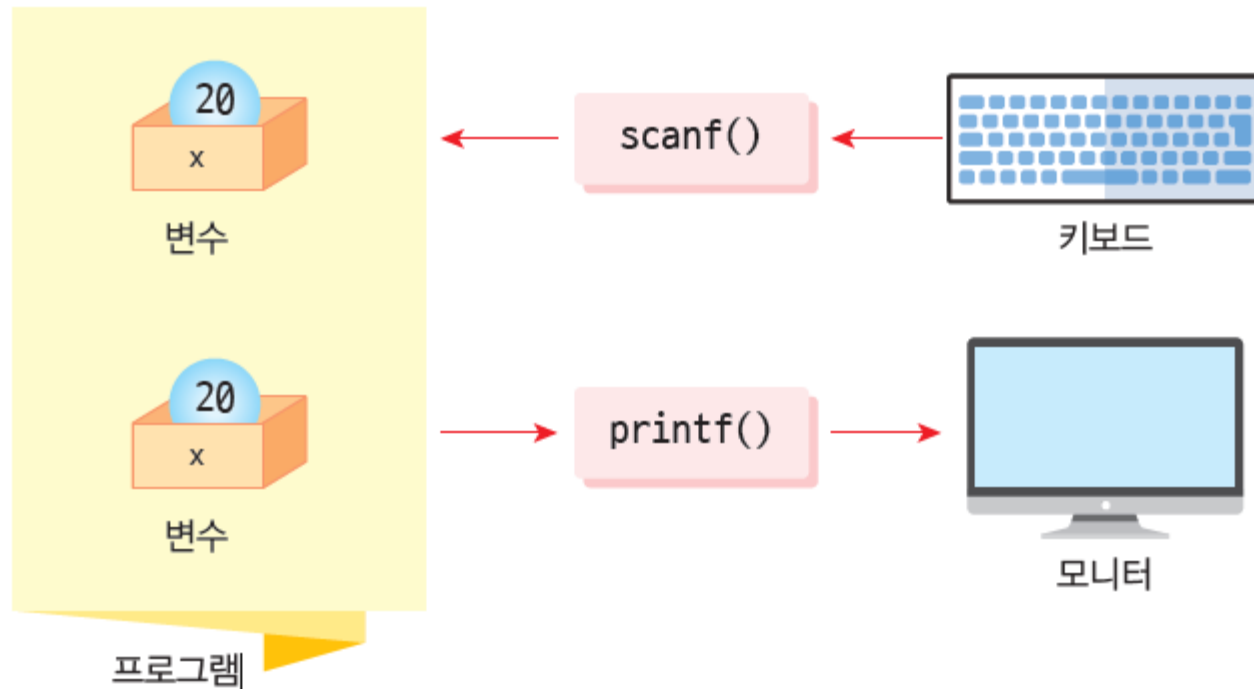
## ● 2. 변수의 사용





# printf()

- printf(): 모니터에 출력을 하기 위한 표준 출력 라이브러리 함수





# printf()

- 형식 :

- 1. print("") 의 형태의 경우 ""안의 내용이 그대로 출력

```
print("apple"); -> apple
```

- 2. print("",...) 의 경우 "" 안의 내용중 일부를 변수의 값을 출력하고자 하는 경우

- "" 안에 %d, %f, %c, %s의 숫자 만큼 이후 변수가 와야 한다.

```
int num=1;  
print("apple%d",num); -> apple1
```

```
int num=1;  
int x=20;  
print("apple%d%d",num,x); -> apple120
```



# 변수값 출력

```
int sum =30;
```

```
printf("두수의 합: %d", sum);
```

sum의 값이 정수형으로 출력된다.

두수의 합: 30

출력이 되는 것들은 print("")의 첫번째 "" 부분으로 출력된다.



# 형식 지정자

- 형식 지정자: `printf()`에서 값을 출력하는 형식을 지정한다.

| 형식 지정자 | 의미         | 예                                      | 실행 결과 |
|--------|------------|--|-------|
| %d     | 10진 정수로 출력 | <code>printf("%d \n", 10);</code>      | 10    |
| %f     | 실수로 출력     | <code>printf("%f \n", 3.14);</code>    | 3.14  |
| %c     | 문자로 출력     | <code>printf("%c \n", 'a');</code>     | a     |
| %s     | 문자열로 출력    | <code>printf("%s \n", "Hello");</code> | Hello |





# 변수의 사용

ch02\_ex1.c

```
#include <stdio.h>

int main(void)
{
    //1. 변수 선언
    int x = 0;
    int y=1;
    printf("변수 선언 x=%d, y=%d\n", x, y);

    //2. 변수의 사용
    x = 10;
    y = 20;
    printf("변수 사용1 x=%d, y=%d\n", x, y);

    x = 100;
    y = 200;
    printf("변수 사용2 x=%d, y=%d\n", x, y);
    return 0;
}
```



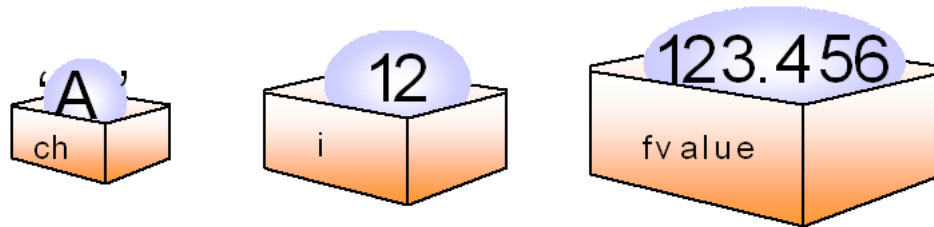
# 중간점검

- 100의 값을 저장하는 변수명  $x$ 를 선언해보자.
- 정수를 저장하는 변수  $y$ 를 선언해보자.
- 변수  $z$ 를 10의 값을 저장하며 선언해보자.
- 변수  $x$ 에 50 저장해보자
- 변수  $y$ 에 300 을 저장해 보자.
- 변수  $y$ 에 30을 저장해 보자



# 변수의 종류

- 변수에는 데이터의 종류에 따라 여러 가지 타입이 존재한다.



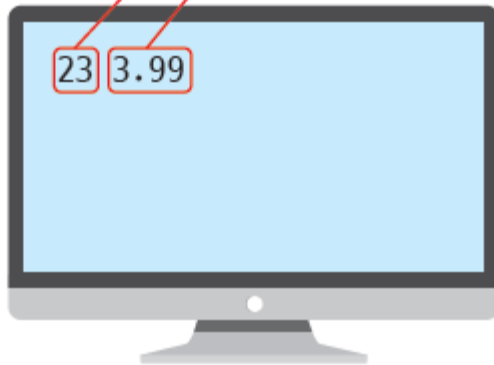


# 여러 개의 변수값 출력

```
int number = 23;  
double grade = 3.99;
```

형식제어 문자열

```
printf("%d %f", number, grade);
```



형식 지정자의 개수와  
변수의 개수와 순서는  
같아야 합니다.





# 여러 개의 변수값 출력

ch02\_ex2.c

```
#include <stdio.h>

int main(void)
{
    //1. 변수 선언
    int number = 6;
    double grade = 0.16;
    // 변수 출력
    printf("%d, %f\n", number, grade);

    return 0;
}
```



# 주의!

```
int sum =250;
```

실수 형식

정수형 변수

```
printf("%f \n", sum);
```

오류가 발생하였습니다.

형식 지정자와 변수의  
타입은 같아야 합니다.





# Lab: 변수 사용하기

- 1. 정수만 사용하는 변수  $x$ ,  $y$ 에는 초기값 0으로 선언한 후,  $x$ ,  $y$ 의 값을 200, 300 으로 변경후 출력하세요
- 2. 실수만 사용하는 변수  $i$ ,  $j$ 를 초기값 0으로 선언한 후,  $i$ ,  $j$ 의 값을 1.2, 10.1 로 변경 후 출력하세요

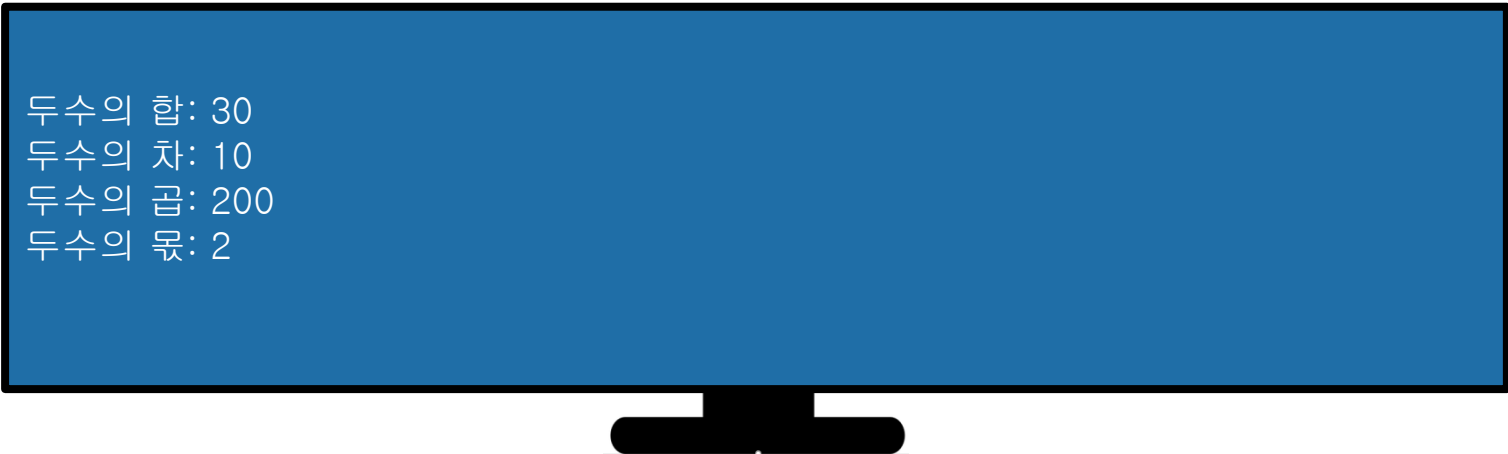






# Lab: 사칙 연산

- 변수  $x$ 와  $y$ 에 20과 10을 저장하고  $x+y$ ,  $x-y$ ,  $x*y$ ,  $x/y$ 을 계산하여서 변수에 저장하고 이들 변수를 화면에 출력하는 프로그램을 작성해보자.

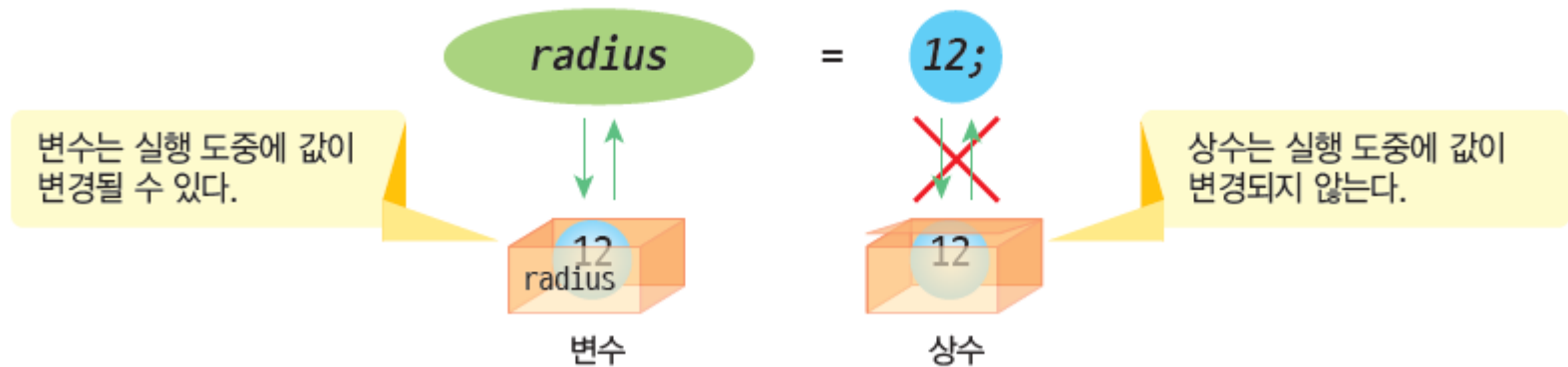


```
두수의 합: 30  
두수의 차: 10  
두수의 곱: 200  
두수의 몫: 2
```



# 변수와 상수

- **변수(variable)**: 저장된 값의 변경이 가능한 공간
- **상수(constant)**: 저장된 값의 변경이 불가능한 공간
  - (예) 3.14, 100, 'A', "Hello World!"





# 예제: 변수와 상수

ch02\_ex4.c

```
/* 원의 면적을 계산하는 프로그램 */  
#include <stdio.h>  
int main(void)  
{  
    double radius=1.2; // 원의 반지름  
    double area=0;      // 원의 면적  
  
    area = 3.141592 * radius * radius;  
    printf("원의 면적: %f \n", area);  
  
    return 0;  
}
```



# 중간 점검

## 1. 변수와 상수는 어떻게 다른가?





# 자료형

- 자료형(data type): 데이터의 타입(종류)
  - short, int, long: 정수형 데이터(100)
  - double, float: 부동소수점형 데이터(3.141592)
  - char: 문자형 데이터('A', 'a', '한')

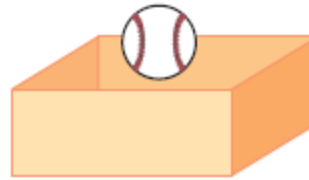




# 다양한 자료형이 필요한 이유

- 상자에 물건을 저장하는 것과 같다.

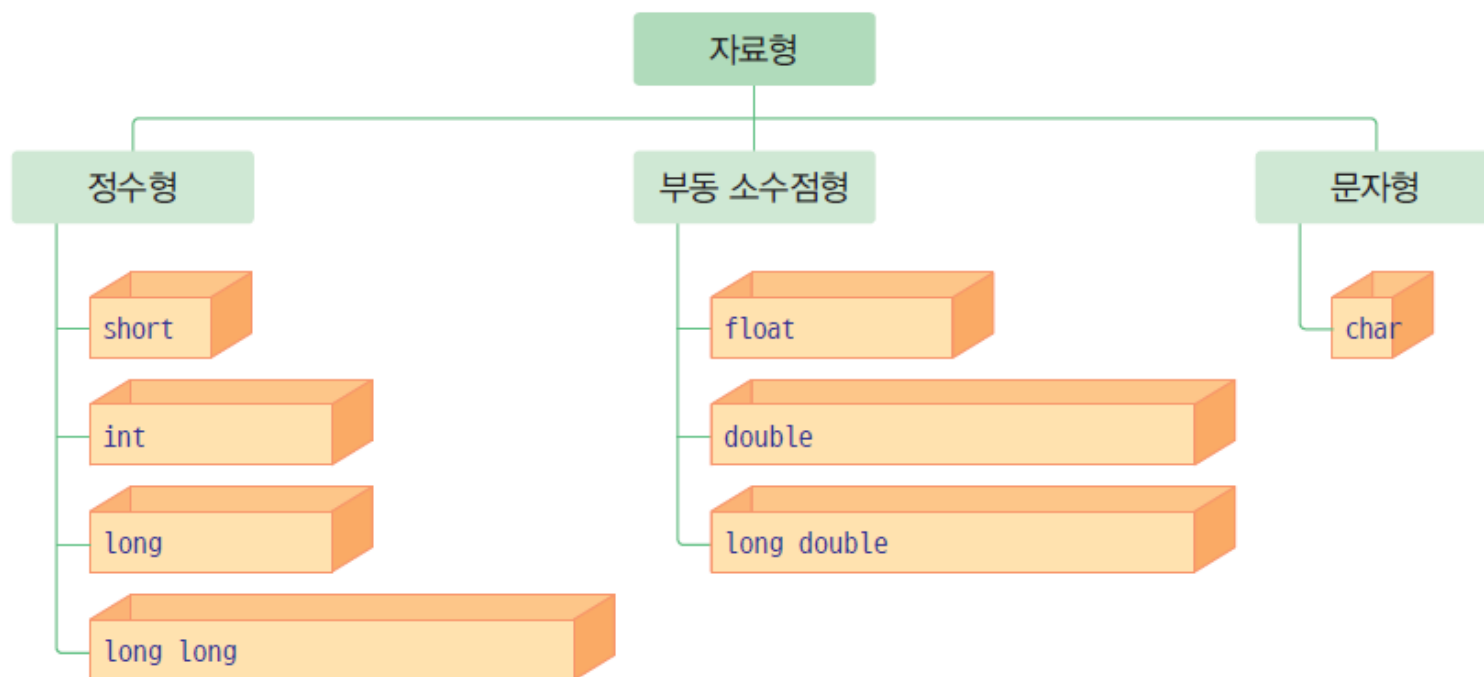
물건이 상자보다 크면  
들어가지 않을 것이다.



물건이 상자보다 너무 작으면  
공간이 낭비될 것이다.



# 자료형의 분류



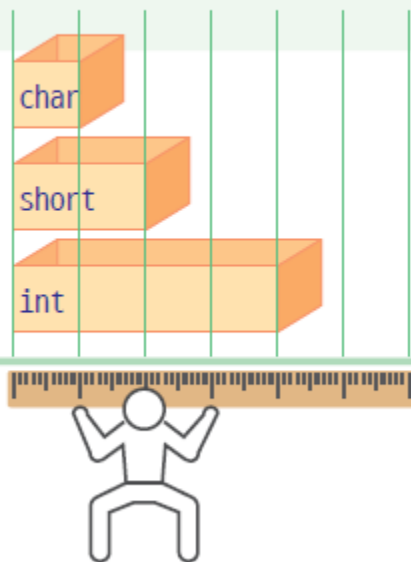


# 자료형의 크기

Syntax: sizeof()

예

```
sizeof(x)      // 변수  
sizeof(10)     // 값  
sizeof(int)    // 자료형  
sizeof(double) // 자료형
```



sizeof 연산자는 변수나  
데이터 타입의 크기를  
바이트 단위로 반환합니다.







# 예제: 자료형의 크기

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

ch02\_ex5.c

```
int main(void)
```

```
{
```

```
    int x=0;
```

```
    printf("변수x의   크기: %d\n", sizeof(x));
```

```
    printf("char형의   크기: %d\n", sizeof(char));
```

```
    printf("int형의   크기: %d\n", sizeof(int));
```

```
    printf("short형의   크기: %d\n", sizeof(short));
```

```
    printf("long형의   크기: %d\n", sizeof(long));
```

```
    printf("float형의   크기: %d\n", sizeof(float));
```

```
    printf("double형의   크기: %d\n", sizeof(double));
```

```
    return 0;
```

```
}
```



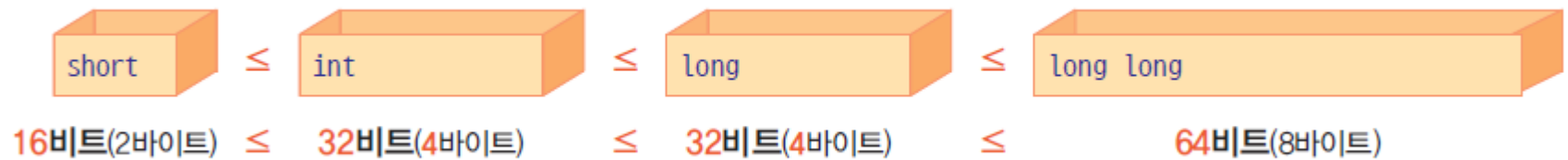
# 실행 결과

변수x의 크기: 4  
char형의 크기: 1  
int형의 크기: 4  
short형의 크기: 2  
long형의 크기: 4  
float형의 크기: 4  
double형의 크기: 8



# 정수형

- short형
- int형
- long형
- long long형





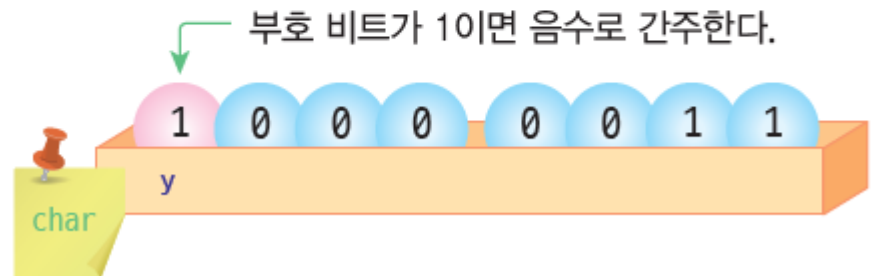
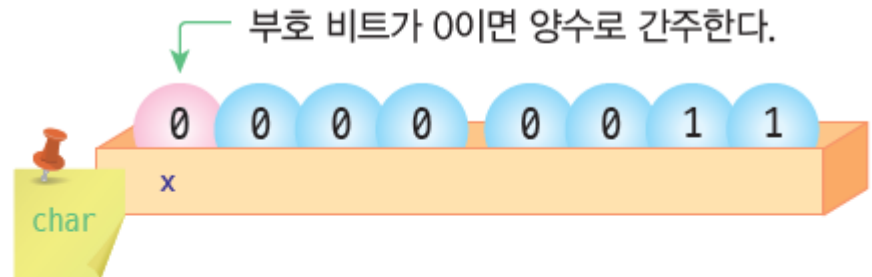
# 정수형

| 자료형 |                    |            | 비트   | 범위   |
|-----|--------------------|------------|------|--|
| 정수형 | short              | 부호있는<br>정수 | 16비트 | -32768~32767   |
|     | int                |            | 32비트 | -2147483648~2147483647                                   |
|     | long               |            |      | -2147483648~2147483647                                   |
|     | long long          |            | 64비트 | -9,223,372,036,854,775,808<br>~9,223,372,036,854,775,807 |
|     | unsigned short     | 부호없는<br>정수 | 16비트 | 0~65535  |
|     | unsigned int       |            | 32비트 | 0~4294967295   |
|     | unsigned long      |            |      | 0~4294967295   |
|     | unsigned long long |            | 64비트 | 0~18,446,744,073,709,551,615                             |



# 정수 표현 방법

- 양수
  - 십진수를 이진수로 변환하여 저장하면 된다.
- 음수
  - 보통은 첫번째 비트를 부호 비트로 사용한다.
  - 문제점이 발생한다.





# 정수형 선언의 예

- `short grade;`                      `// short형의 변수를 생성한다.`
- `int count;`                      `// int형의 변수를 생성한다.`
- `long distance;`                      `// distance형의 변수를 생성한다.`

## ★ 참고사항

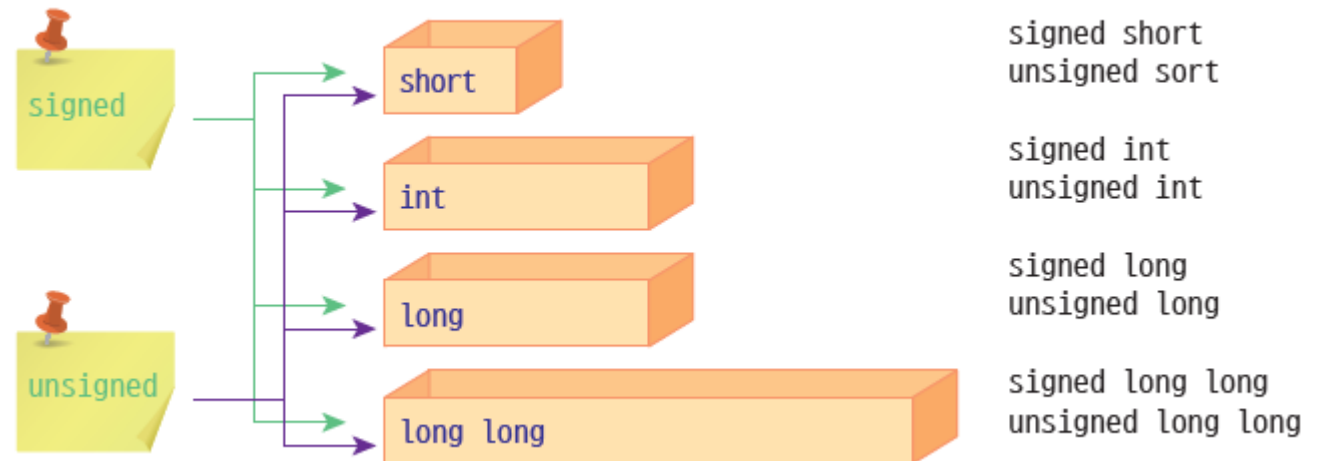
Q: 왜 C에서는 각 자료형들의 크기가 정확하게 정의되지 않는가?

A: 특히 int형은 CPU에 따라 달라진다.  
비록 C가 저수준 언어이기는하지만  
C는 자료형의 정확한 크기는 구현 세부 사항이라는 입장을 가지고 있었다.



# signed, unsigned 수식자

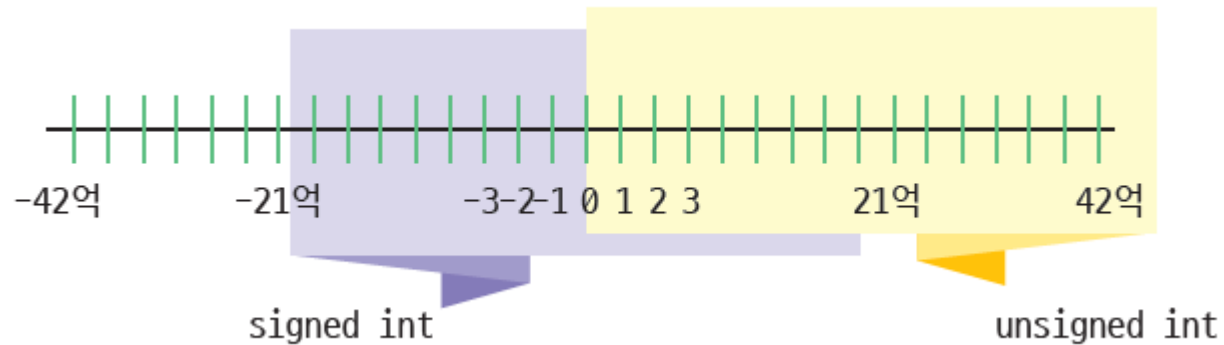
- unsigned
  - 음수가 아닌 값만을 나타냄을 의미
  - unsigned int
- signed
  - 부호를 가지는 값을 나타냄을 의미
  - 흔히 생략





# unsigned int

$0, 1, 2, \dots, 2^{32} - 1$   
(0 ~ +4294967295)







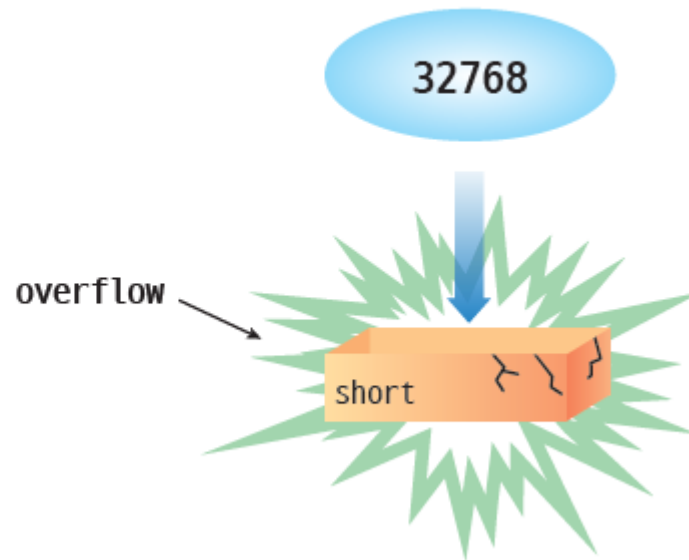
# unsigned 수식자

```
unsigned int  speed;           // 부호없는 int형
unsigned      distance;        // unsigned int distance와 같다.
unsigned short players;        // 부호없는 short형
unsigned long  seconds;        // 부호없는 long형
```



# 오버플로우

- **오버플로우(overflow)**: 변수가 나타낼 수 있는 범위를 넘는 숫자를 저장하려고 할 때 발생





# 오버플로우

ch02\_ex6.c

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

```
int main()
```

```
{
```

```
    short a = 32767; // short 최대값
```

```
    short b = -32768; // short 최소값
```

```
    short c = -32768; // short 최소값
```

```
    a = a + 2;
```

```
    b = b - 2;
```

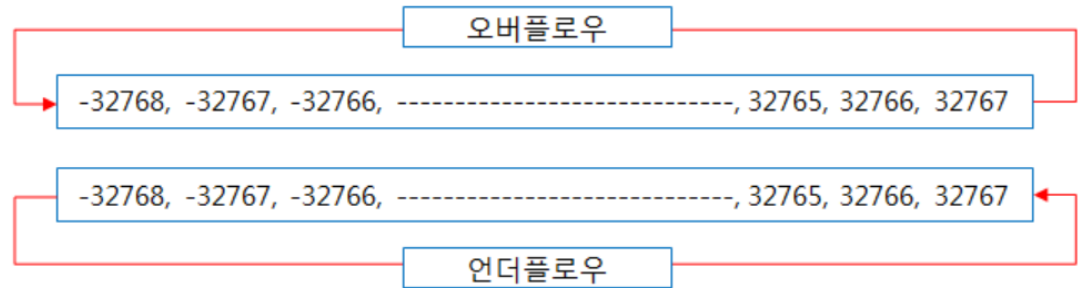
```
    printf("a overflow : %d\n", a);
```

```
    printf("b underflow : %d\n", b);
```

```
    printf("c : %d\n", c);
```

```
    return 0;
```

```
}
```



오버플로우 발생!!

언더플로우 발생!!

정상

a overflow : -32767

b underflow : 32766

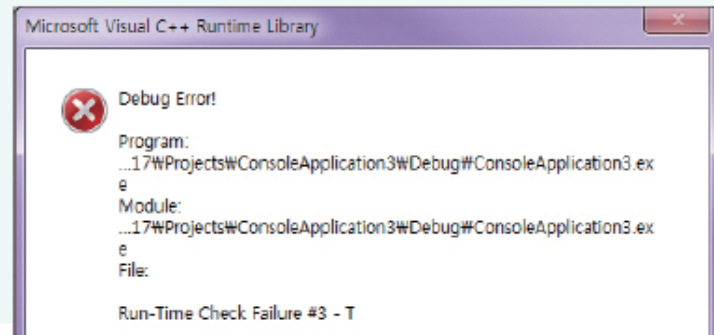
c : -32768



# Lab: 변수의 초기값

sum\_error.c

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int x, y, z, sum;
5      printf("3개의 정수를 입력하세요 (x, y, z): ");
6      scanf("%d %d %d", &x, &y, &z);
7      sum += x;
8      sum += y;
9      sum += z;
10     printf("3개 정수의 합은 %d\n", sum);
11     return 0;
12 }
```





# 무엇이 문제일까?

sum\_error.c

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int x, y, z, sum;
5
6      sum = 0;
7      printf("3개의 정수를 입력하세요 (x, y, z): ");
8      scanf("%d %d %d", &x, &y, &z);
9      sum += x;
10     sum += y;
11     sum += z;
12     printf("3개 정수의 합은 %d\n", sum);
13     return 0;
14 }
```

변수는 사용하기 전에 반드시 초기화  
시켜야 함!

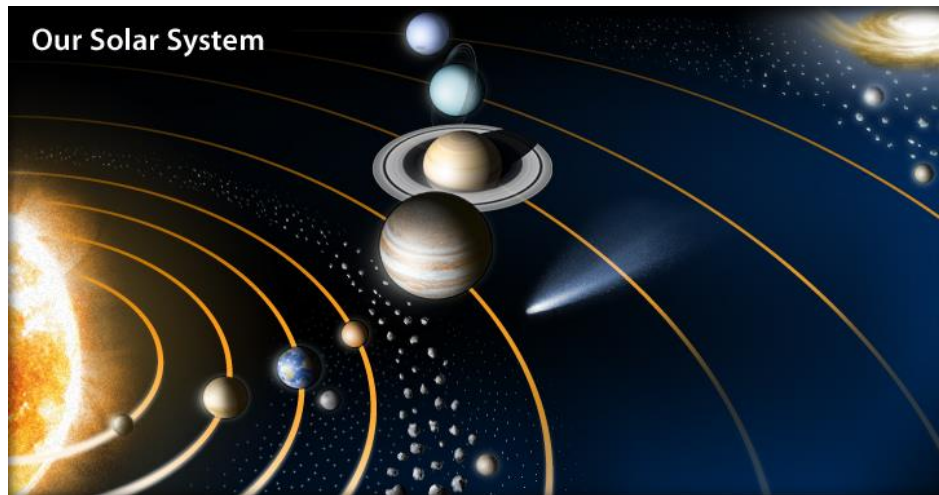
## 실행결과

3개의 정수를 입력하세요 (x, y, z): 10 20 30  
3개의 정수의 합은 60



# Lab : 태양빛 도달 시간

- 태양에서 오는 빛이 몇 분 만에 지구에 도착하는 지를 컴퓨터로 계산해보고자 한다.
- 빛의 속도는 1초에 30만 km를 이동한다.
- 태양과 지구 사이의 거리는 약 1억 4960만 km이다.





# 실행 결과

빛의 속도는 300000.000000km/s  
태양과 지구와의 거리 149600000.000000km  
도달 시간은 498.666667초



# 힌트

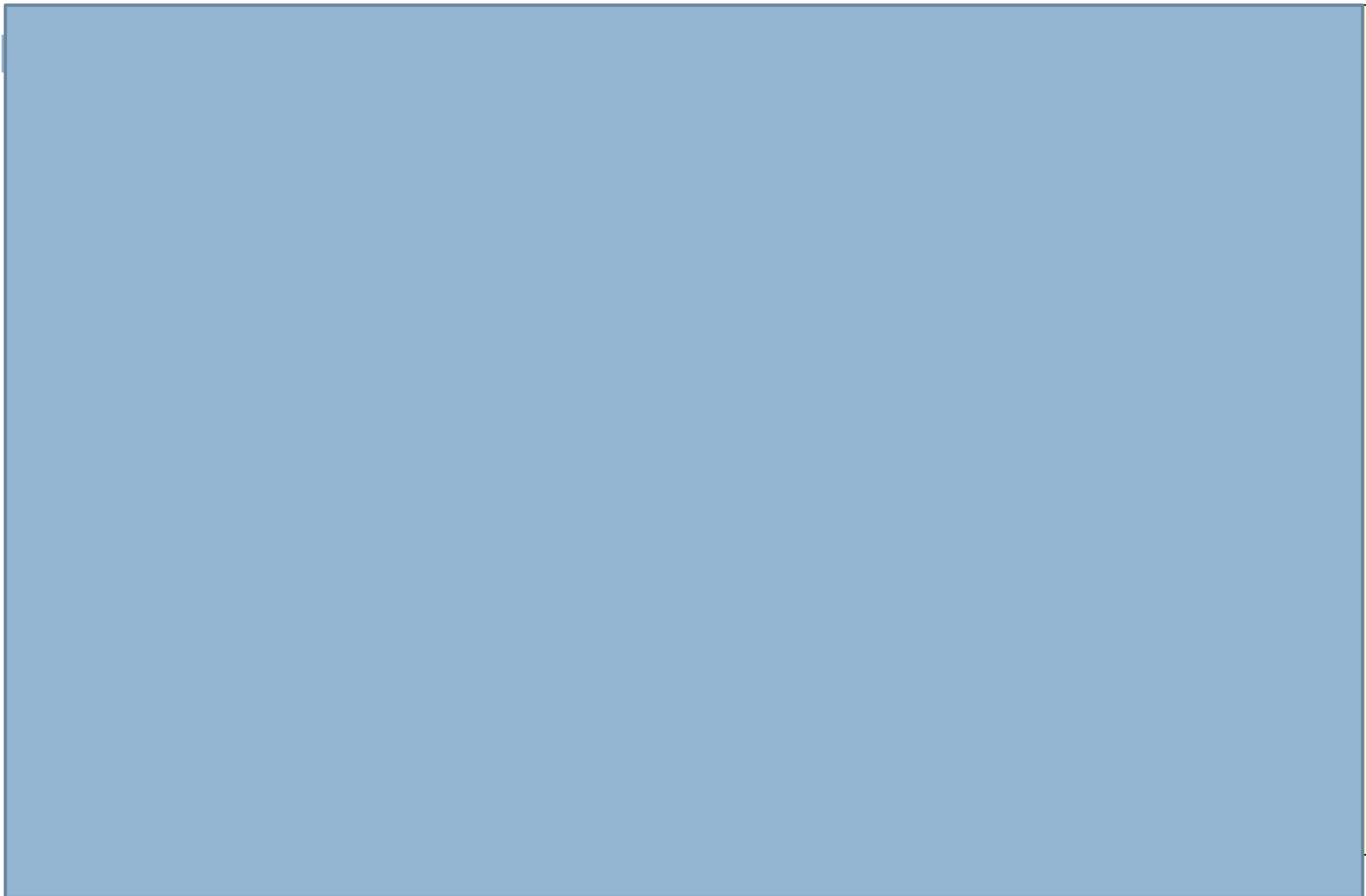
- 문제를 해결하기 위해서는 먼저 필요한 변수를 생성하여야 한다. 여기서는 빛의 속도, 태양과 지구 사이의 거리, 도달 시간을 나타내는 변수가 필요하다.
- 변수의 자료형은 모두 실수형이어야 한다. 왜냐하면 매우 큰 수들이기 때문이다.
- 빛이 도달하는 시간은 (도달 시간 = 거리 / (빛의 속도))으로 계산할 수 있다.
- 실수형을 `printf()`로 출력할 때는 `%f`나 `%lf`를 사용한다.





소 스

Ch02\_lab2.c





# scanf()

- 키보드로부터 값을 받아서 변수에 저장한다.
- 변수의 주소를 필요로 한다.

- 형식 지정자

- 값을 저장할 변수의 주소

scanf("%d", &x);



# 주소가 필요한 이유

변수의 주소는 &를  
붙여서 보냈어요!



주소만 있으면 배송이  
가능합니다.



# 형식지정자

| 형식 지정자 | 의미                  | 예  |
|--------|---------------------|--|
| %d     | 정수를 10진수로 입력한다      | <code>scanf("%d", &amp;i);</code>        |
| %f     | float 형의 실수로 입력한다.  | <code>scanf("%f", &amp;f);</code>        |
| %lf    | double 형의 실수로 입력한다. | <code>scanf("%lf", &amp;d);</code>       |
| %c     | 문자 형태로 입력한다.        | <code>scanf("%c", &amp;ch);</code>       |
| %s     | 문자열 형태로 입력한다.       | <code>char s[10]; scanf("%s", s);</code> |

아직 학습하지 않았음!  
너무 신경쓰지 말것!



# 실수 입력시 주의할 점

```
float ratio = 0.0;  
scanf("%f", &ratio);
```

- float 형은 %f 사용

```
double scale = 0.0;  
scanf("%lf", &scale);
```

- double 형은 %lf 사용

잘못 사용하면 오류가  
발생합니다.

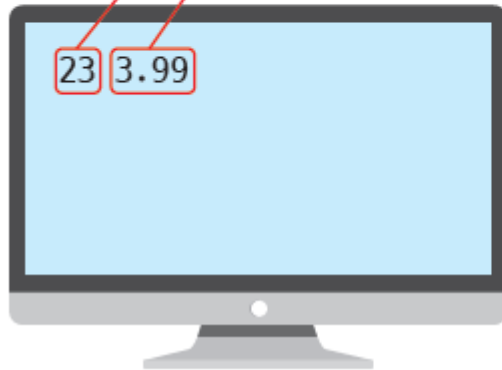




# scanf()

형식제어 문자열

```
scanf("%d %f", &number, &grade);
```



형식 지정자의 개수와  
변수의 개수와 순서는  
같아야 합니다.





# 비주얼 스튜디오에서 scanf() 오류

The screenshot shows the Visual Studio IDE with a C program named 'hello.c' open. The code is as follows:

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int i;
6     printf("정수를 입력하시오: ");
7     scanf("%d", &i);
8     return;
9 }
```

The error list at the bottom shows two errors. The second error, C4996, is highlighted in a red box and pointed to by a yellow lightning bolt:

| 코드    | 설명  | 프로젝트  | 파일      | 줄 |
|-------|---|-------|---------|---|
| C6031 | 반환 값이 무시되었습니다. 'scanf'.   | hello | hello.c | 7 |
| C4996 | 'scanf': This function or variable may be unsafe. Consider using scanf_s instead. To disable deprecation, use _CRT_SECURE_NO_WARNINGS. See online help for details. | hello | hello.c | 7 |



# scanf() 사용시 컴파일 오류가 난다면?

! C6031 반환 값이 무시되었습니다. 'scanf'.  
✖ C4996 'scanf': This function or variable may be unsafe. Consider using scanf\_s instead. To disable deprecation, use \_CRT\_SECURE\_NO\_WARNINGS. S

scanf() 가 안전하지 않으니 scanf\_s()를  
사용하라는 의미이다.

만약 이런 오류가 발생하면 다음 페이지와 같이  
\_CRT\_SECURE\_NO\_WARNINGS를  
정의해준다.





# 해결책

The screenshot shows the Visual Studio IDE with a C program named 'hello.c' open. The code is as follows:

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int i;
7      printf("정수를 입력하시오: ");
8      scanf("%d", &i);
9      return;
10 }
```

The first line, `#define _CRT_SECURE_NO_WARNINGS`, is highlighted with a red rectangular box. A yellow lightning bolt icon points to this line, indicating a warning suppression technique. The IDE's status bar at the bottom shows '모두 다시 빌드했습니다.' (Rebuild all).

The right sidebar shows the '솔루션 탐색기' (Solution Explorer) with the project 'hello' and its files: '참조' (References), '외부 종속성' (External Dependencies), '리소스 파일' (Resource Files), '소스 파일' (Source Files) containing 'hello.c', and '헤더 파일' (Header Files).

The bottom panel shows the '출력' (Output) window with the following text:

```
출력 보기 선택(S): 빌드
다시 빌드 시작...
1>----- 모두 다시 빌드 시작: 프로젝트: hello, 구성: Debug Win32 -----
1>hello.c
1>hello.vcxproj -> C:\Users\k\source\repos\hello\Debug\hello.exe
===== 모두 다시 빌드: 성공 1, 실패 0, 생략 0 =====
```



# 정수를 받아들이는 프로그램

ch02\_ex6.c

```
#define _CRT_SECURE_NO_WARNINGS
```

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

```
int main(void)
```

```
{
```

```
    int x;                // 정수를 저장할 변수
```

```
    printf("정수를 입력하시오: ");
```

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

```
    printf("입력된 정수 = %d \n", x);
```

```
    return 0;
```

```
}
```

만약 scanf() 오류가 발생하면 소스 파일의  
처음에서  
\_CRT\_SECURE\_NO\_WARNINGS를  
정의해준다.

Microsoft Visual Studio 디버그 콘솔

정수를 입력하시오: 20  
입력된 정수 = 20

C:\Users\chun\source\repos\Project8\Debug\Project8.exe(28548 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.



# scanf() 정리

책의 소스에서 scanf() 오류가 발생하면 소스  
파일의 첫부분에서  
`_CRT_SECURE_NO_WARNINGS`를  
정의해준다.



소스의 첫 부분에 한  
줄을 추가한다.

```
#define _CRT_SECURE_NO_WARNINGS  
...  
...
```



소스를 그대로 사용하  
다.

```
...  
...
```