



# 누구나 즐기는 C언어 콘서트

## 제3장 변수와 자료형





# 이번 장에서 학습할 내용



- 변수와 상수의 개념 이해
- 자료형
- 정수형
- 실수형
- 문자형



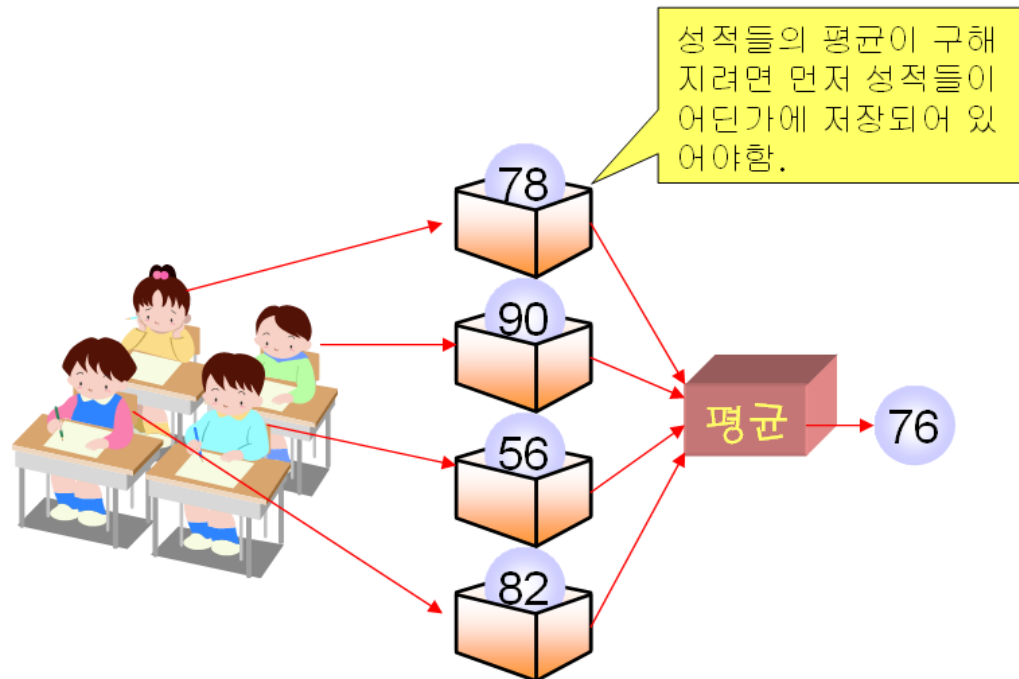
# 변수

Q) 변수(variable)이란 무엇인가?

A) 프로그램에서 일시적으로 데이터를 저장하는 공간

Q) 변수는 왜 필요한가?

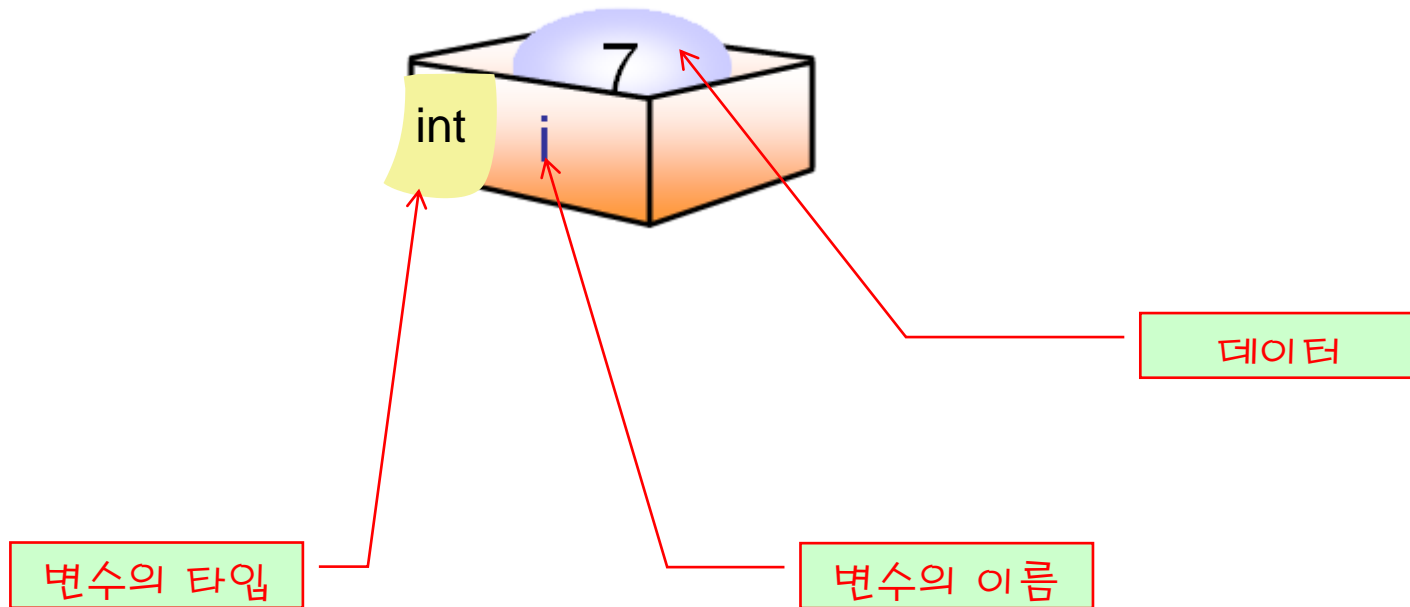
A) 데이터가 입력되면 어딘가에 저장해야만 다음에 사용할 수 있다.





# 변수 = 상자

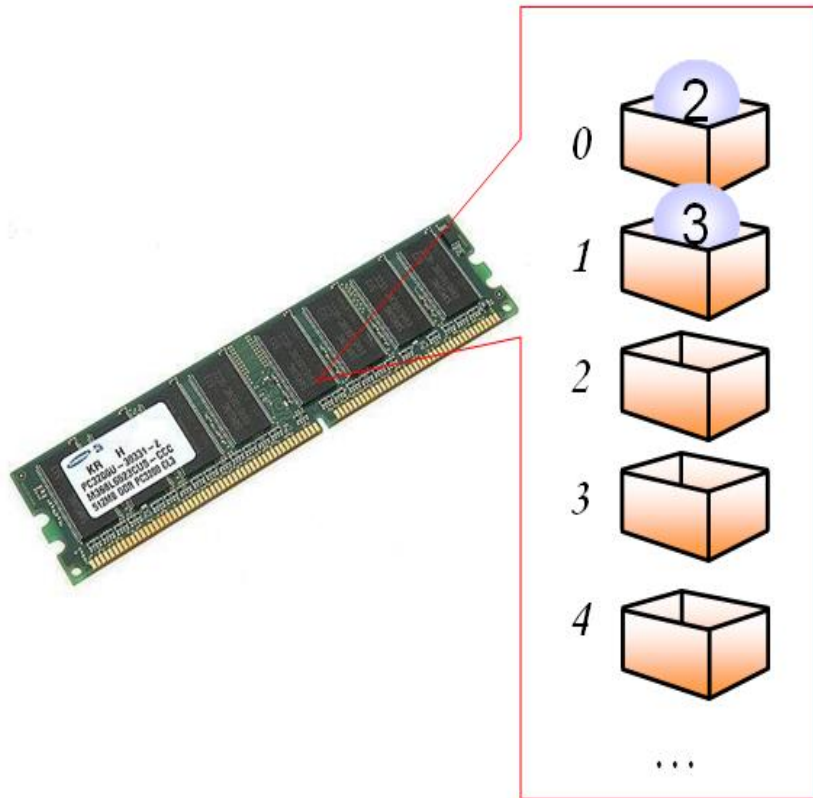
- 변수는 물건을 저장하는 상자와 같다.





# 변수가 만들어지는 곳

- 변수는 메인 메모리에 만들어진다.



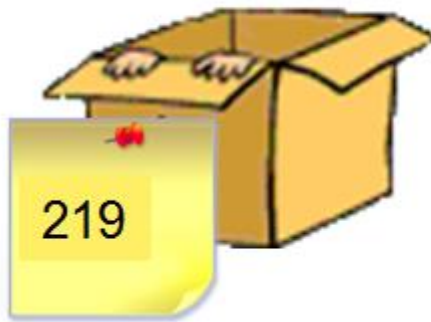


# 변수가 필요한 이유

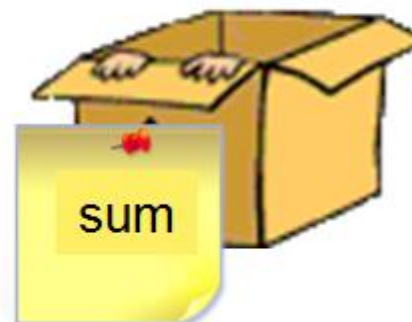
(Q) 만약 메모리를 변수처럼 이름을 가지고 사용하자 않고 주소로 사용한다면?

“219번지에 0을 대입하라”

(A) 충분히 가능하지만 불편하다. 인간은 숫자보다는 기호를 더 잘 기억한다.



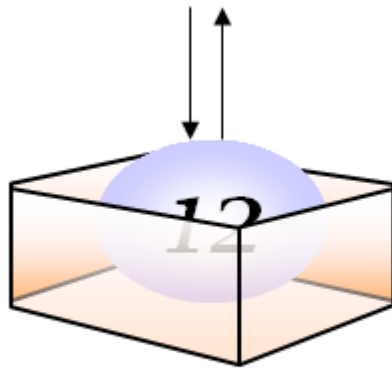
VS



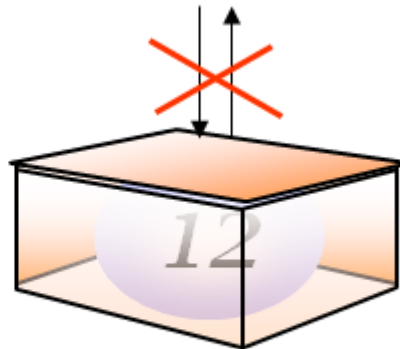


# 변수와 상수

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



변수



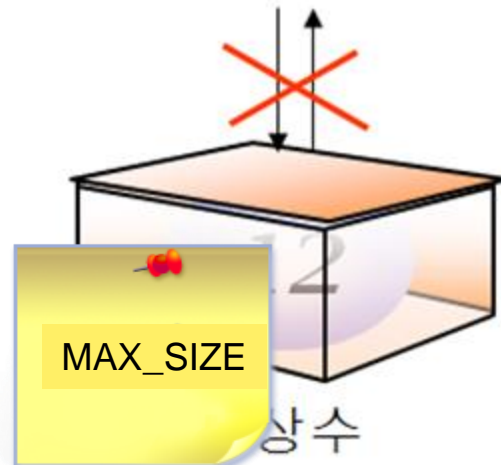
상수



# 상수의 이름


(Q) 상수도 이름을 가질 수 있는가?

(A) 보통 상수는 이름이 없다. 이러한 상수를 리터럴 (literal)이라고 한다. 하지만 필요하다면 상수에도 이름을 붙일 수 있다. 이것을 기호 상수라고 한다.





## 예제



```
/* 원의 면적을 계산하는 프로그램*/
```

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

```
int main(void)
```

```
{
```

```
float radius; // 원의 반지름
```

```
float area; // 원의 면적
```

```
printf("원의 반지름을 입력하시요:");
```

```
scanf("%f", &radius);
```

```
area = 3.141592 * radius * radius;
```


```
printf("원의 면적: %f", area);
```

```
return 0;
```

```
}
```

변수

상수



```
원의 반지름을 입력하시요:5.0  
원의 면적: 78.539803.
```



# 자료형

- 자료형(data type): 데이터의 타입(종류)
  - 정수형 데이터(100)
  - 실수형 데이터(3.141592)
  - 문자형 데이터('A')

자료형이  
다양한 이유는  
커피 전문점에  
다양한 컵의  
사이즈가 있는  
것과 같습니다.





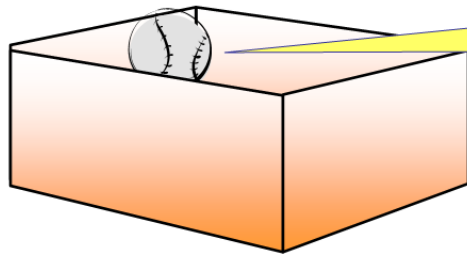
# 다양한 자료형

(Q) 다양한 자료형이 필요한 이유는?

(A) 상자에 물건을 저장하는 것과 같다.



물건이 상자보다 크면 들어가지 않을 것이다.



물건이 상자보다 너무 작으면 공간이 낭비될 것이다.



# 자료형의 종류

자료형			설명	바이트수	범위
정수형	부호있음	short	short형 정수	2	-32768 ~ 32767
		int	정수	4	-2147483648 ~ 2147483647
		long	long형 정수	4	-2147483648 ~ 2147483647
	부호없음	unsigned short	부호없는 short형 정수	2	0 ~ 65535
		unsigned int	부호없는 정수	4	0 ~ 4294967295
		unsigned long	부호없는 long형 정수	4	0 ~ 4294967295
문자형	부호있음	char	문자 및 정수	1	-128 ~ 127
	부호없음	unsigned char	문자 및 부호없는 정수	1	0 ~ 255
부동소수점형		float	단일정밀도 부동소수점	4	1.2E-38 ~ 3.4E38
		double	두배정밀도 부동소수점	8	2.2E-308 ~ 1.8E308



# 자료형의 크기

- sizeof 연산자 이용

형식

`sizeof(변수)`

`sizeof(자료형)`

예

`sizeof(x)` // 변수

`sizeof(10)` // 값

`sizeof(int)` // 자료형

`sizeof(double)` // 자료형

# 예제

```
#include <stdio.h>
int main(void)
{
    int x;
    printf("변수 x의 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;
}
```

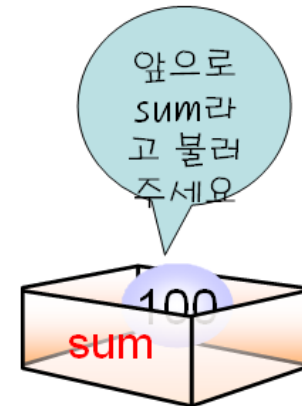
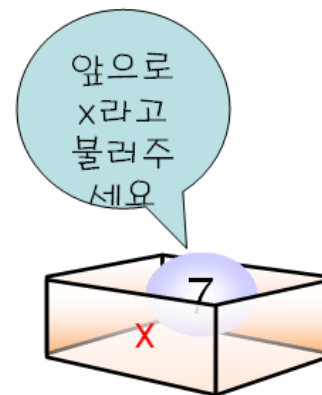
sizeof 연산자

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



# 변수의 이름짓기

- 식별자(identifier): 식별할 수 있게 해주는 이름
  - 변수 이름
  - 함수 이름





# 식별자를 만드는 규칙

- 알파벳 문자와 숫자, 밑줄 문자 \_로 구성
- 첫 번째 문자는 반드시 알파벳 또는 밑줄 문자 \_
- 대문자와 소문자를 구별
- C 언어의 키워드와 똑같은 이름은 허용되지 않는다.

(Q) 다음은 유효한 식별자인가?

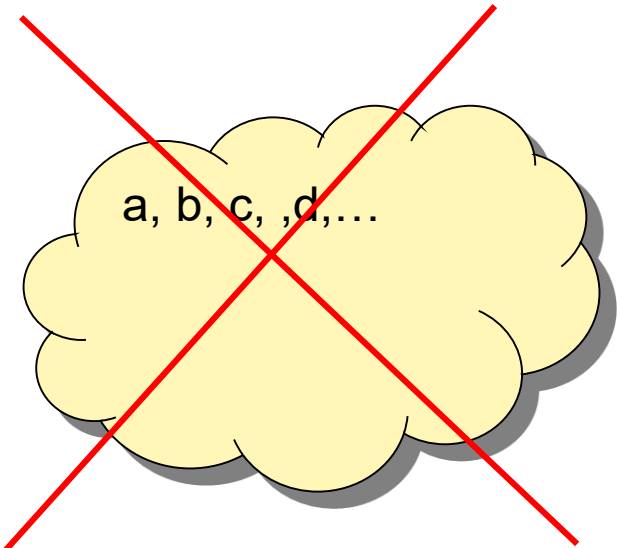
sum	O	
_count	O	
king3	O	
n_pictures	O	
2nd_try	X	// 숫자로 시작
dollar#	X	// #기호
double	X	// 키워드



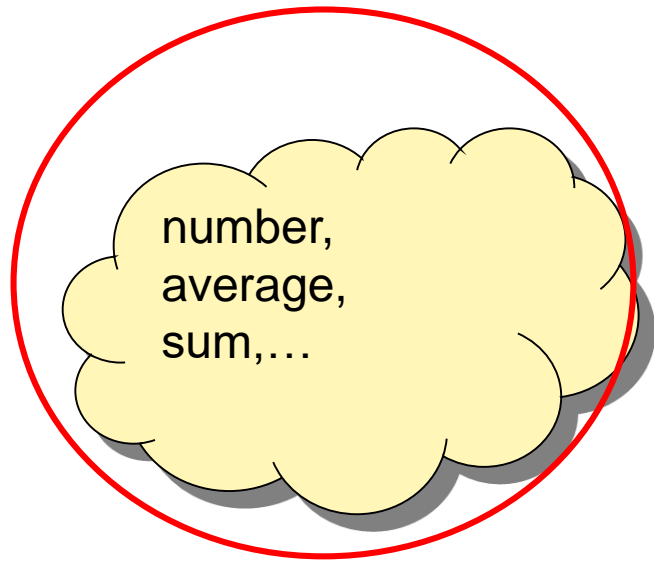


# 좋은 변수 이름

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



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



number,  
average,  
sum,...



# 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

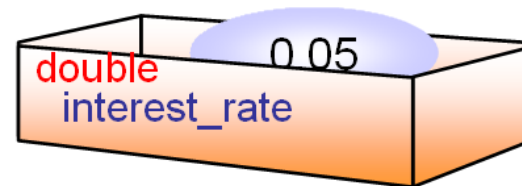
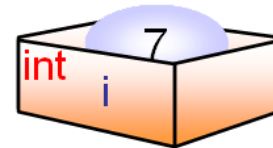


# 변수 선언

- 변수 선언: 컴파일러에게 어떤 변수를 사용하겠다고 미리 알리는 것

자료형    변수이름;

- 변수 선언의 예
  - char c;
  - int i;
  - double interest\_rate;
  - int w, h;

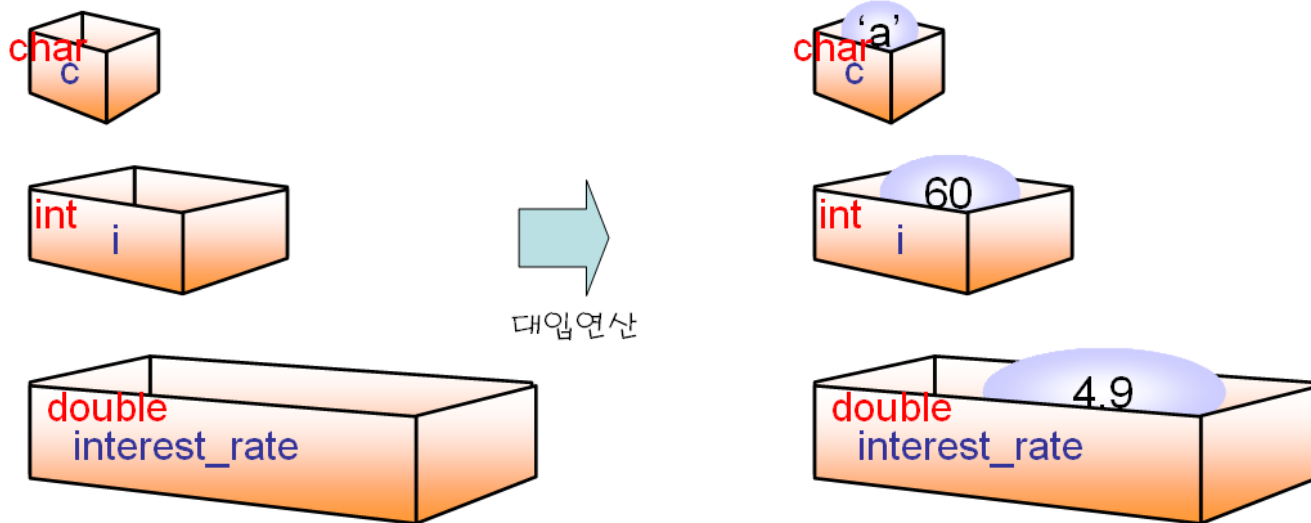




# 변수에 값을 저장하는 방법

```
char c;           // 문자형 변수 c 선언
int i;            // 정수형 변수 i 선언
double interest_rate; // 실수형 변수 interest_rate 선언
```

```
c = 'a';          // 문자형 변수 c에 문자 'a'를 대입
i = 60;           // 정수형 변수 i에 60을 대입
interest_rate = 4.9; // 실수형 변수 interest_rate에 4.9를 대입
```

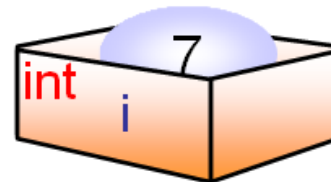




# 변수의 초기화

자료형    변수이름 = 초기값;

```
char c = 'a';  
int i = 7;  
double interest_rate = 0.05;
```





# 변수 선언 위치

- 변수는 함수의 첫부분에서만 선언할 수 있습니다. (ANSI C에서는 사실이 아님. 사용되기 전에 선언되기만 하면 됨)

```
int main(void)
```

```
{
```

```
int index=0;
```

```
int count=0;
```

```
count = 10;
```

```
index = 7;
```

```
int sum;
```

```
...
```

```
}
```

변수 선언

일반 문장

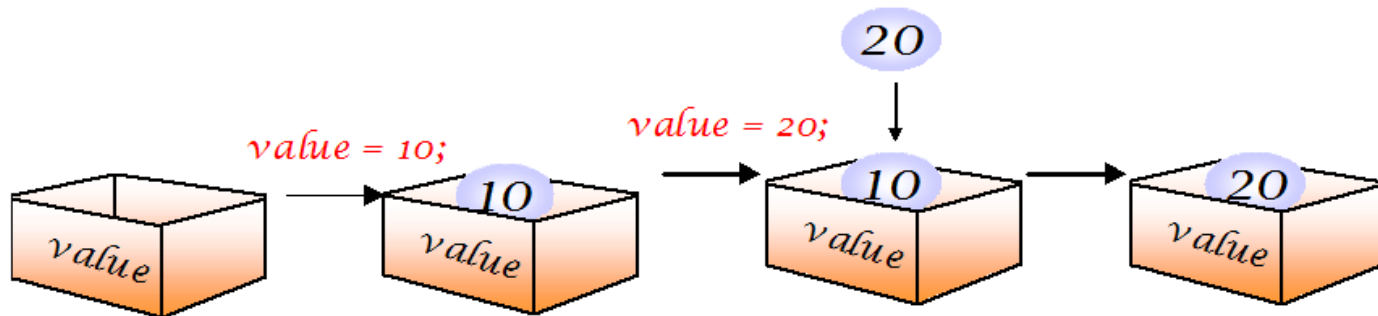
오류! 잘못된 변수 선언



# 변수의 사용

- 대입 연산자를 이용하여서 값을 저장한다.

```
int value;  
value = 10;  
...  
value = 20;
```

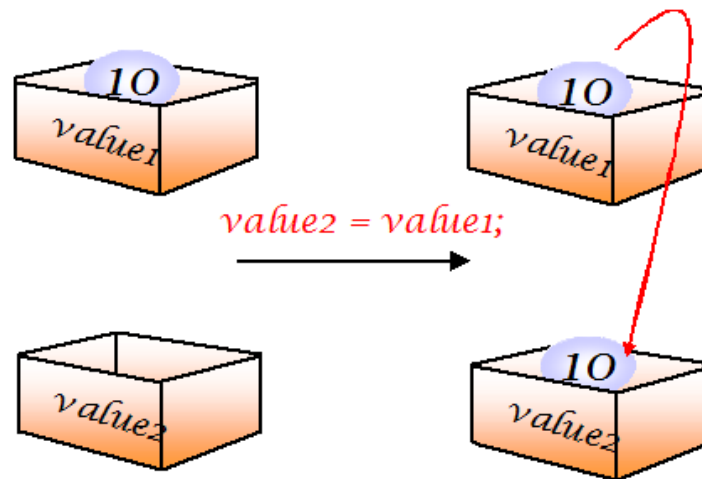




# 변수의 사용

- 저장된 값은 변경이 가능하다.

```
int value1 = 10;  
int value2;  
value2 = value1;
```





# 예제

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

```
int main(void)
```

```
{
```

```
    int usd;           // 달러화
```

```
    int krw;           // 원화
```

```
    printf("달러화 금액을 입력하시오: ");
```

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

```
    krw = 1120 * usd;
```

```
    printf("달러화 %d 달러는 %f원입니다.\n", usd, krw);
```

```
    return 0;
```

```
}
```

변수 선언

변수에 값을 입력받는다.

// 입력 안내 메시지

// 달러화 금액 입력

// 원화로 환산

// 계산 결과 출력

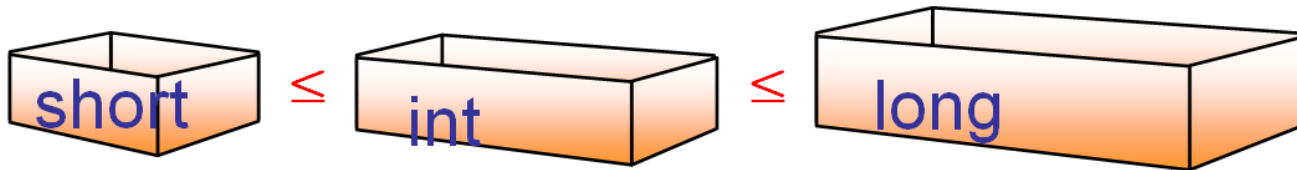
// 함수 결과값 반환

달러화 금액을 입력하시오: 100  
달러화 100달러는 112000원입니다.



# 정수형

- short, int, long



16비트(2바이트) ≤ 32비트(4바이트) ≤ 32비트(4바이트)

- 가장 기본이 되는 것은 int
  - CPU에 따라서 크기가 달라진다.
  - 16비트, 32비트, 64비트
  - 64-bit machine - long 타입은 64비트 사용

(Q) 왜 여러 개의 정수형이 필요한가?

(A) 용도에 따라 프로그래머가 선택하여 사용할 수 있게 하기 위하여



## 정수형 선언의 예

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



# 정수형의 범위

- int형

$$-2^{31}, \dots, -2, -1, 0, 1, 2, \dots, 2^{31} - 1$$
$$-2147483648 \leq n \leq +2147483647$$

약 -21억에서  
+21억

- short형

$$-2^{15}, \dots, -2, -1, 0, 1, 2, \dots, 2^{15} - 1$$
$$-32768 \leq n \leq +32767$$

- long형

- 보통 int형과 같음

# 예제

```
#include <stdio.h>
int main(void)
{
    short year = 0;           // 0으로 초기화한다.
    int sale = 0;             // 0으로 초기화한다.
    long total_sale = 0;      // 0으로 초기화한다.

    year = 10;                // 약 3만2천을 넘지 않도록 주의
    sale = 2000000000;        // 약 21억을 넘지 않도록 주의
    total_sale = year * sale;  // 약 21억을 넘지 않도록 주의
    printf("total_sale = %d \n", total_sale);

    printf("short형의 크기: %d바이트\n", sizeof(short));
    printf("int형의 크기: %d바이트\n", sizeof(int));
    printf("long형의 크기: %d바이트\n", sizeof(long));
    return 0;
}
```

자료형의 크기를 반환.

```
total_sale = 20000000000
short형의크기: 2바이트
int형의크기: 4바이트
long형의크기: 4바이트
```



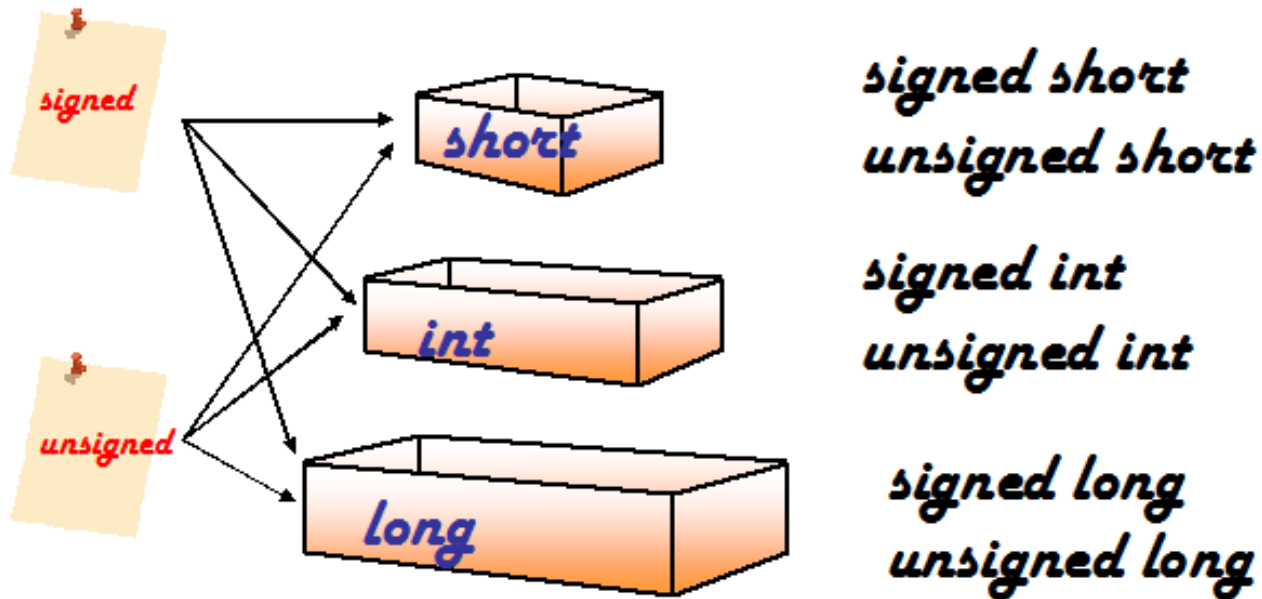
# signed, unsigned 수식자

- unsigned
  - 음수가 아닌 값만을 나타냄을 의미
- (예) unsigned int

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



# unsigned 와 signed





# unsigned 수식자

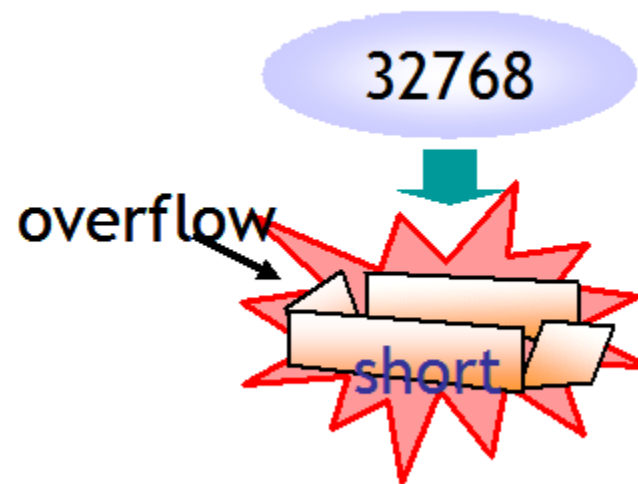
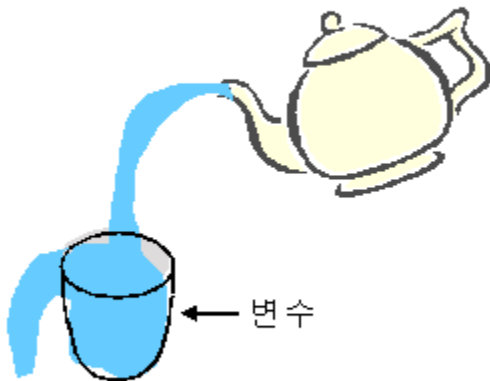
- *unsigned int* speed; // 부호없는 int형
- *unsigned* distance; // unsigned int distance와 같다.
- *unsigned short* players; // 부호없는 short형
- *unsigned long* seconds; // 부호없는 long형





# 오버플로우

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





# 오버플로우



```
#include <stdio.h>
#include <limits.h>
int main(void)
{
    short s_money = SHRT_MAX; // 최대값으로 초기화한다.
    unsigned short u_money = USHRT_MAX; // 최대값으로 초기화한다.

    s_money = s_money + 1;
    printf("s_money = %d\n", s_money);
    u_money = u_money + 1;
    printf("u_money = %d\n", u_money);
    return 0;
}
```

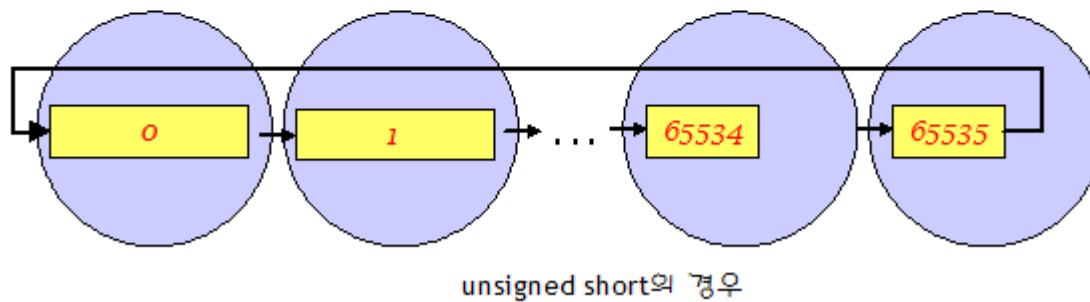
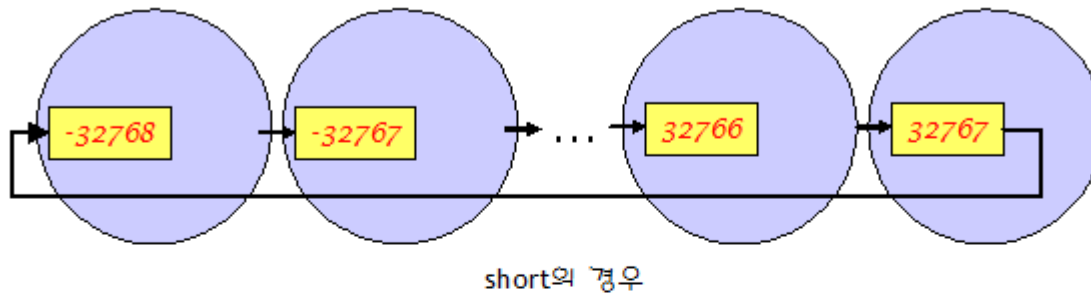


```
s_money = -32768
u_money = 0
```



# 오버플로우

- 규칙성이 있다.
  - 수도 계량기나 주행거리계와 비슷하게 동작





# 정수 상수

- 정수 상수: 정수형의 상수
  - (예) 12, 100
- 정수 상수는 기본적으로 `int`형으로 간주
- 상수의 자료형을 명시하려면 다음과 같이 한다.

접미사	자료형	예
u 또는 U	unsigned int	123u 또는 123U
l 또는 L	long	123l 또는 123L
ul 또는 UL	unsigned long	123ul 또는 123UL



# 다양한 진법 가능

- 10진수뿐만 아니라 8진수, 16진수으로도 표기 가능

- 8진수

(예) `int n = 012;`    // 012는 8진수

$$012_8 = 1 \times 8^1 + 2 \times 8^0 = 10_{10}$$

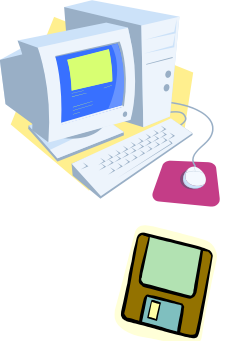
- 16진수

(예) `int n = 0xA;`    // 0xA는 16진수

$$0xA_{16} = 10 \times 16^0 = 10_{10}$$

10진수	8진수	16진수
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	10	8
9	11	9
10	12	a
11	13	b
12	14	c
13	15	d
14	16	e
15	17	f
16	20	10
17	21	11

자리수증가



# 예제

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

```
int main(void)
```

```
{
```

```
    int x = 10;
```

```
    int y = 010;
```

```
    int z = 0x10;
```

// 10은 10진수이고 십진수로 10이다.

// 010은 8진수이고 십진수로 8이다.

// 010은 16진수이고 십진수로 16이다.

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

```
    printf("y = %d\n", y);
```

```
    printf("z = %d\n", z);
```

```
    return 0;
```

```
}
```



x = 10

y = 8

z = 16



# 기호 상수

- 기호 상수(symbolic constant): 기호를 이용하여 상수를 표현한 것
- (예)
  - $\text{area} = 3.141592 * \text{radius} * \text{radius};$
  - $\text{area} = \text{PI} * \text{radius} * \text{radius};$
  - $\text{income} = \text{salary} - 0.15 * \text{salary};$
  - $\text{income} = \text{salary} - \text{TAX\_RATE} * \text{salary};$
- 기호 상수의 장점
  - 가독성(readability)이 높아진다.
  - 여러 곳에서 사용되는 값을 쉽게 변경할 수 있다.



# 기호 상수의 장점

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

```
int main(void)
```

```
{
```

```
...
```

```
won1 = 1120 * dollar1;
```

```
won2 = 1120 * dollar2;
```

```
...
```

```
}
```

1050

1050

리터럴 상수를 사용하는 경우:  
등장하는 모든 곳을  
수정하여야 한다.

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

```
#define EXCHANGE_RATE 1120
```

```
int main(void)
```

```
{
```

```
...
```

```
won1 = EXCHANGE_RATE * dollar1;
```

```
...
```

```
won2 = EXCHANGE_RATE * dollar2;
```

```
...
```

```
}
```

1050

기호 상수를 사용하는 경우:  
기호 상수가 정의된 곳만 수정  
하면 된다.





# 기호 상수를 만드는 방법

## ① #define 기호상수이름 값

```
/* 기호 상수 프로그램*/
#include <stdio.h>
#define PI 3.141592

int main(void)
{
    float radius, area, circumference;    // 변수 선언

    printf("반지름을 입력하시요:");      // 입력 안내문
    scanf("%f", &radius);                // 사용자로부터 반지름 입력

    area = PI * radius * radius;          // 면적 계산
    circumference = 2.0 * PI * radius;    // 둘레 계산

    printf("반지름은 %f입니다.\n", radius); // 반지름 출력
    printf("원의 면적은 %f이고 둘레는 %f입니다.\n", area, circumference);

    return 0;
}
```

기호 상수 정의



# 기호 상수를 만드는 방법

## ② const 키워드 이용

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

```
int main(void)
```

```
{
```

```
    const int MONTHS = 12; // 기호 상수 선언
```

```
    int m_salary, y_salary; // 변수 선언
```

```
    printf("월급을 입력하시요: "); // 입력 안내문
```

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

```
    y_salary = 12 * m_salary; // 순수입 계산
```

```
    printf("연봉은 %d입니다.\n", y_salary);
```

```
    return 0;
```

```
}
```

기호 상수 정의



# 부동소수점형

- 컴퓨터에서 실수는 부동소수점형으로 표현
  - 소수점이 떠서 움직인다는 의미
  - 과학자들이 많이 사용하는 과학적 표기법과 유사

실수의 정밀도를 나타낸다.

실수의 표현범위를 나타낸다.

$$1.49598 \times 10^8$$

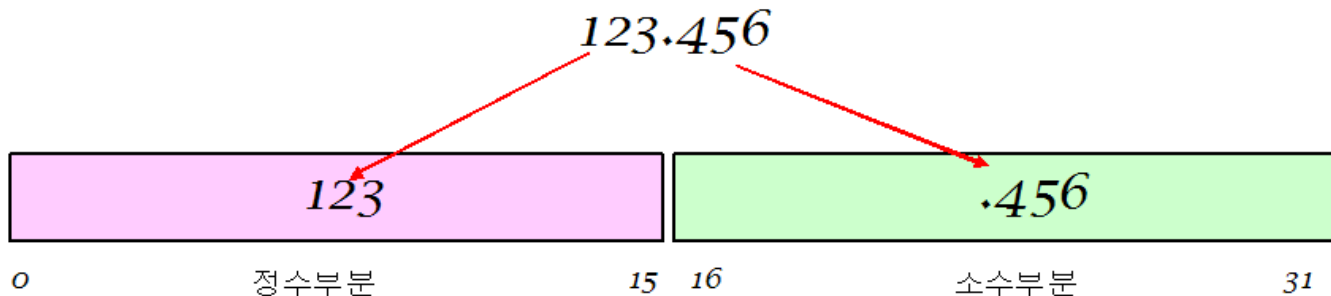
가수부분
지수부분

실수	과학적 표기법	지수 표기법
123.45	$1.2345 \times 10^2$	1.2345e2
12345.0	$1.2345 \times 10^5$	1.2345e5
0.000023	$2.3 \times 10^{-5}$	2.3e-5
2,000,000,000	$2.0 \times 10^9$	2.0e9



# 실수를 표현하는 방법

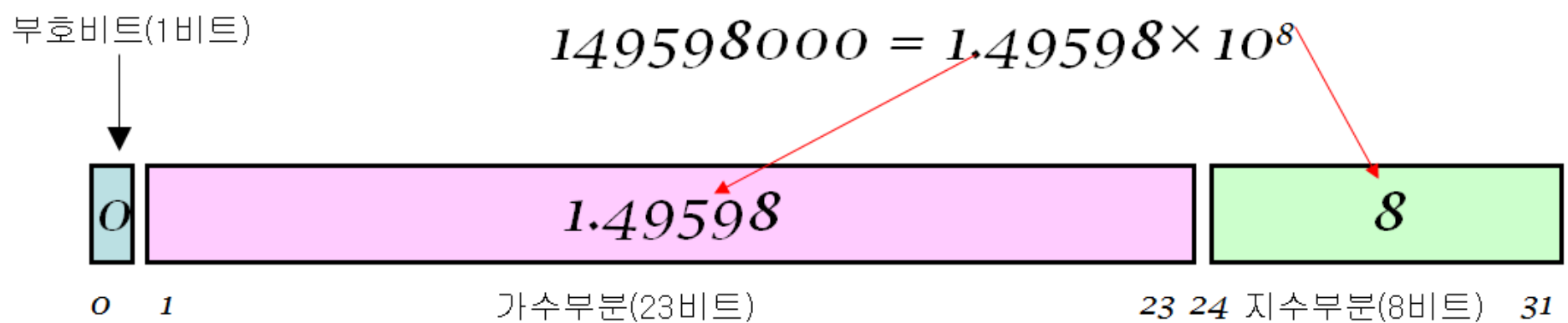
- #1 고정 소수점 방식
  - 정수 부분을 위하여 일정 비트를 할당하고 소수 부분을 위하여 일정 비트를 할당
  - 전체가 32비트이면 정수 부분 16비트, 소수 부분 16비트 할당
  - 과학과 공학에서 필요한 아주 큰 수를 표현할 수 없다





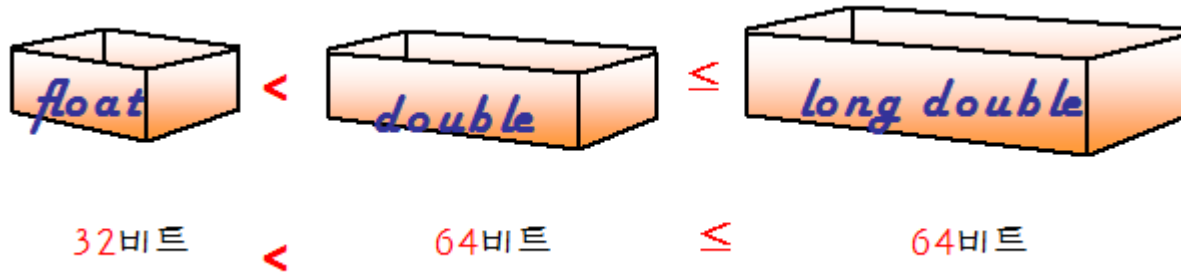
# 실수를 표현하는 방법

- #2 부동 소수점 방식
  - 표현할 수 있는 범위가 대폭 늘어난다.
  - $10^{-38}$  에서  $10^{+38}$





# 부동 소수점 형



자료형	명칭	크기	유효 숫자	범위
float	단일 정밀도 (single-precision)	32비트	6자리	$\pm 1.17549 \times 10^{-38} \sim \pm 3.40282 \times 10^{+38}$
double	두배 정밀도 (double-precision)	64비트	16자리	$\pm 2.22507 \times 10^{-308} \sim \pm 1.79769 \times 10^{+308}$
long double	두배 확장 정밀도 (double-extension-precision)	64비트	16자리	$\pm 2.22507 \times 10^{-308} \sim \pm 1.79769 \times 10^{+308}$



# 부동 소수점 상수

- 일반적인 실수 표기법
  - 3.141592 (double형)
  - 3.141592F (float형)
- 지수표기법
  - $1.23456e4 = 12345.6$
  - $1.23456e-3 = 0.00123456$
- 유효한 표기법의 예
  - 1.23456
  - 2. // 소수점만 붙여도 된다.
  - .28 // 정수부가 없어도 된다.
  - 0e0
  - 2e+10 // +나 -기호를 지수부에 붙일 수 있다.
  - 9.26E3 //
  - 9.26e3 //



# 예제

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

```
int main()
```

```
{
```

```
    float fvalue = 1234567890.12345678901234567890;
```

```
    double dvalue = 1234567890.12345678901234567890;
```

```
    printf("float형 변수=%30.25f\n", fvalue);
```

```
    printf("double형 변수=%30.25f\n", dvalue);
```

```
    return 0;
```

```
}
```



float형 변수=1234567936.000000000000000000000000000000

double형 변수=1234567890.123456700000000000000000000000



# 오버플로우와 언더플로우

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

```
int main(void)
```

```
{
```

```
    float x = 1e39;
```

오버플로우 발생

```
    float y = 1.23456e-46;
```

언더플로우 발생

```
    printf("x=%e\n", x);
```

```
    printf("y=%e\n", y);
```

```
    return 0;
```

```
}
```

```
x=1.#INF00e+000
```

```
y=0.000000e+000
```



# 부동소수점형 사용시 주의사항

- 오차가 있을 수 있다!



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

```
int main(void)
```

```
{
```

```
    double x;
```

```
    x = (1.0e20 + 5.0) - 1.0e20;
```

```
    printf("x = %f \n", x);
```

```
    return 0;
```

```
}
```

5.0은 부동 소수점수가  
저장할 수 있는  
한계때문에 저장되지  
않는다.



```
x = 0.000000
```



## 중간 점검

1. 부동 소수점형에 속하는 자료형을 모두 열거하라.
2. `float`형 대신에 `double`형을 사용하는 이유는 무엇인가?
3. 부동 소수점형에서 오차가 발생하는 근본적인 이유는 무엇인가?
4. 12.345처럼 소수점이 있는 실수를 `int`형의 변수에 넣을 경우, 어떤 일이 발생하는가?
5. 수식  $(1.0/3.0)$ 을 `float`형 변수와 `double`형 변수에 각각 저장한 후에 출력하여 보자.  $(1.0/3.0)$ 은 0.333333.... 값을 출력하여야 한다. 소수점 몇 자리까지 정확하게 출력되는가?





# 문자형

- 문자는 컴퓨터보다는 인간에게 중요
- 문자도 숫자를 이용하여 표현
- 공통적인 규격이 필요하다.
- 아스키 코드(**ASCII**: American Standard Code for Information Interchange)
  - 8비트를 사용하여 영어 알파벳 표현
  - (예) !는 33, 'A'는 65, 'B'는 66, 'a'는 97, 'b'는 98

```
!"#$%&'()*+,-./0123456789:;<=>?  
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_  
`abcdefghijklmnopqrstuvwxyz{|}~
```



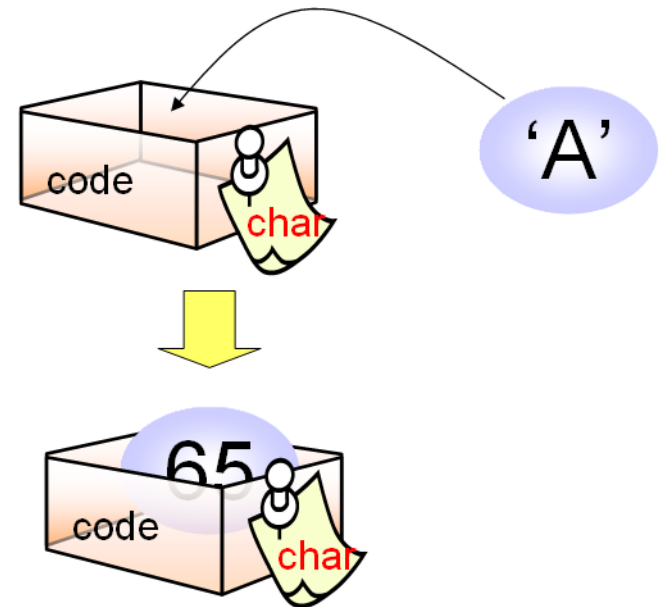
# 문자 변수

- char형의 변수가 문자 저장

```
char c;  
char answer;  
char code;
```

- char형의 변수에 문자를 저장하려면 아스키 코드 값을 대입

```
code = 65;           // 'A' 저장  
code = 'A';
```





# 문자 입출력 예제

```
#include <stdio.h>

int main()
{
    char c; // 변수 선언

    printf("문자를 입력하시오: "); // 입력 안내문
    scanf("%c", &c);
    printf("입력된 문자는 %c입니다\n", c);

    return 0;
}
```



문자를 입력하시오: m  
입력된 문자는 m입니다



# 예제

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

```
int main()
```

```
{
```

```
    char ch = 'A'; // 변수선언
```

```
    printf("%c의 아스키 코드= %d\n", ch, ch);
```

```
    ch = ch + 1;
```

```
    printf("%c의 아스키 코드= %d\n", ch, ch);
```

```
    return 0;
```

```
}
```



A의 아스키 코드 = 65

B의 아스키 코드 = 66



# 제어 문자

- 인쇄 목적이 아니라 제어 목적으로 사용되는 문자들
  - (예) 줄바꿈 문자, 탭 문자, 벨소림 문자, 백스페이스 문자
- 제어 문자를 나타내는 방법
  - 아스키 코드를 직접 사용

```
char beep = 7;  
printf("%c", beep);
```

- 이스케이프 시퀀스 사용

```
char beep = '\a';  
printf("%c", beep);
```





# 제어 문자

제어 문자 이름	제어 문자 표기	값	의미
널문자	\0	0	문자열의 끝을 표시
경고(bell)	\a	7	"삐"하는 경고 벨소리 발생
백스페이스(backspace)	\b	8	커서를 현재의 위치에서 한 글자 뒤로 옮긴다.
수평탭(horizontal tab)	\t	9	커서의 위치를 현재 라인에서 설정된 다음 탭 위치로 옮긴다.
줄바꿈(newline)	\n	10	커서를 다음 라인의 시작 위치로 옮긴다.
수직탭(vertical tab)	\v	11	설정되어 있는 다음 수직 탭 위치로 커서를 이동
폼피드(form feed)	\f	12	주로 프린터에서 강제로 다음 페이지로 넘길 때 사용된다.
캐리지리턴(carriage return)	\r	13	커서를 현재 라인의 시작 위치로 옮긴다.
큰따옴표	\“	34	원래의 큰따옴표 자체
작은따옴표	\‘	39	원래의 작은따옴표 자체
역슬래시(back slash)	\\	92	원래의 역슬래시 자체



# 예제



```
#include <stdio.h>
int main()
{
    int id, pass;
    printf("아이디와 패스워드를 4개의 숫자로 입력하세요:\n");
    printf("id: ____\b\b\b\b");
    scanf("%d", &id);
    printf("pass: ____\b\b\b\b");
    scanf("%d", &pass);
    printf("\a입력된 아이디는 \"%d\"이고 패스워드는 \"%d\"입니다.\n", id, pass);

    return 0;
}
```



아이디와 패스워드를 4개의 숫자로 입력하세요:  
id: 1234  
pass: 5678  
입력된 아이디는 "1234"이고 패스워드는 "5678"입니다.



# 정수형으로서의 char형

- 8비트의 정수를 저장하는데 char 형을 사용할 수 있다..

```
char code = 65;
```

```
printf("%d %d %d", code, code+1, code+2); // 65 66 67이 출력된다.
```

```
printf("%c %c %c", code, code+1, code+2); // A B C가 출력된다.
```



65 66 67A B C