제4장 변수와 자료형

내부적인 정수 표현 방식

- 컴퓨터에서 정수는 이진수 형태로 표현됨

- 이진수는 전자 스위치로 생각할 수 있음
  - 0 또는 1 둘 중에 한 가지 값만 가질 수 있음

비트가 하나 있으면 표현할 수 있는 수는 0과 1 두 가지임

비트가 둘 있으면 표현할 수 있는 수는 0(00), 1(01), 2(10), 3(11) 네 가지임

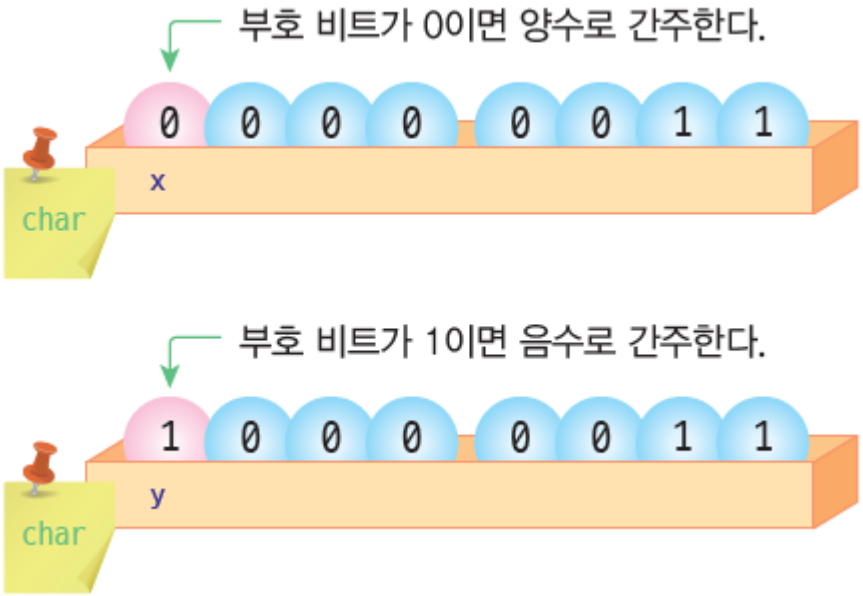
내부적인 정수 표현 방식

- short 형 정수 표현
  - short 형은 2바이트이므로 16비트로 표현됨
  - 예

0	0000000000000000
1	0000000000000001
2	0000000000000010
3	0000000000000011
4	0000000000000100

정수 표현 방법

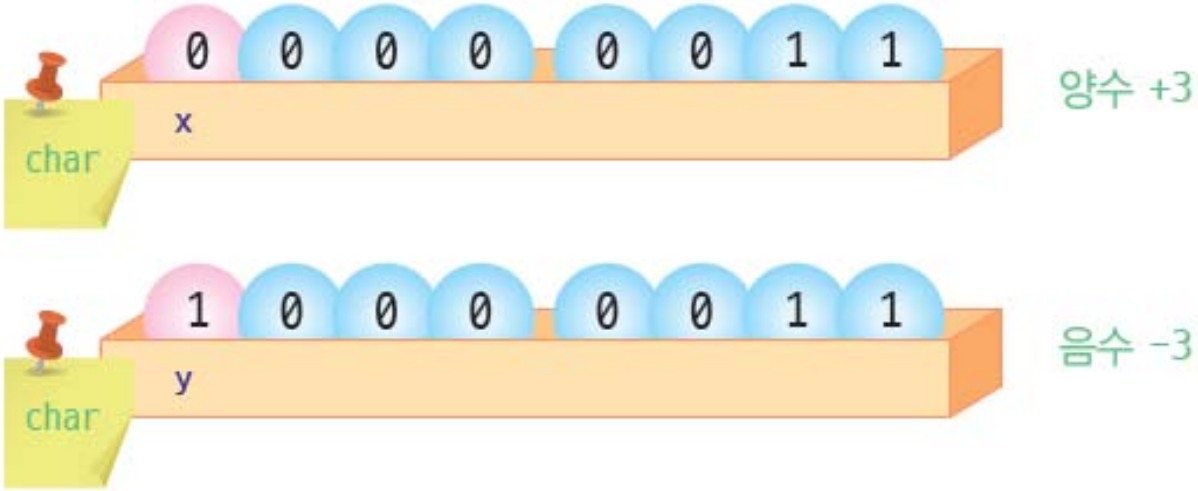
- 양수
  - 십진수를 이진수로 변환하여 저장하면 된다.
- 음수
  - 보통은 첫번째 비트를 부호 비트로 사용한다.



음수를 표현하는 첫번째 방법

- 양수와 동일하게 하되 맨 처음 비트만 1로 바꾸는 방법

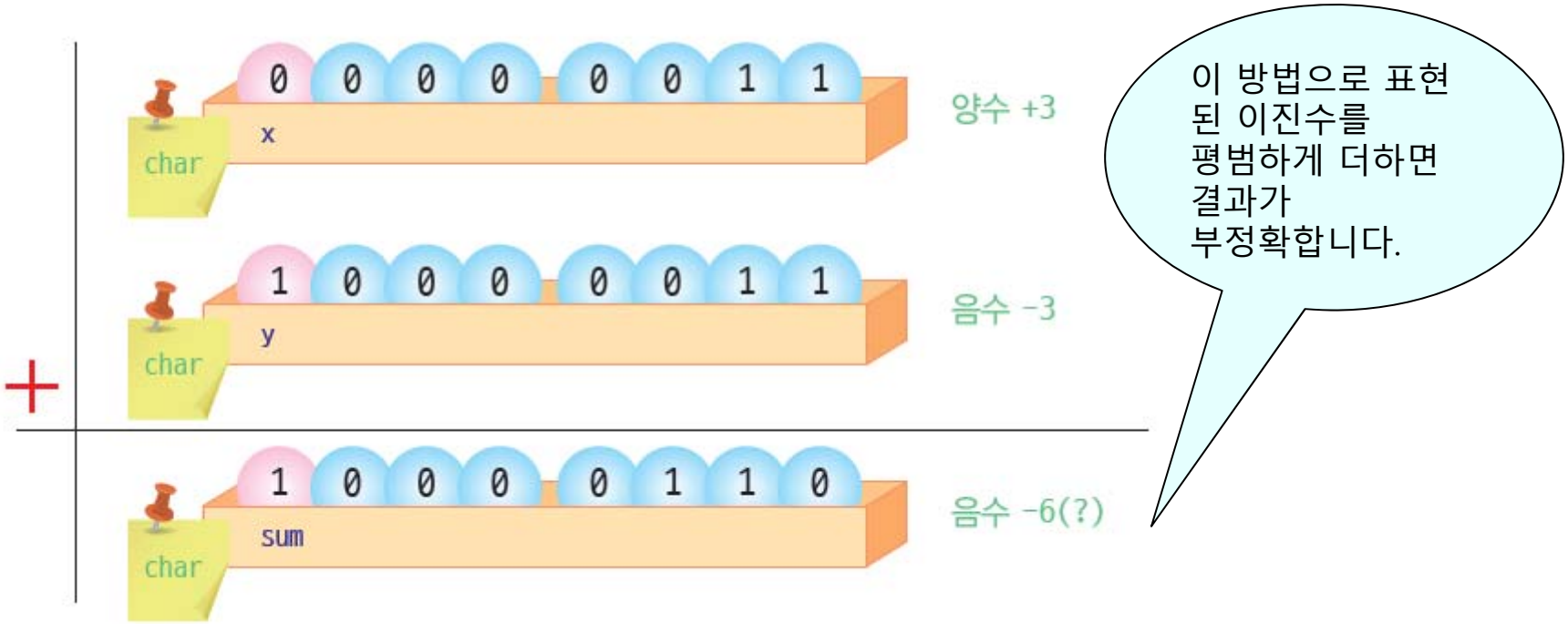
- 이렇게 하면 아무 문제가 없을까?



컴퓨터는 덧셈만 할 수 있다

- 컴퓨터는 덧셈 회로만 가지고 있다.
- 뺄셈은 다음과 같이 덧셈으로 변환한다.

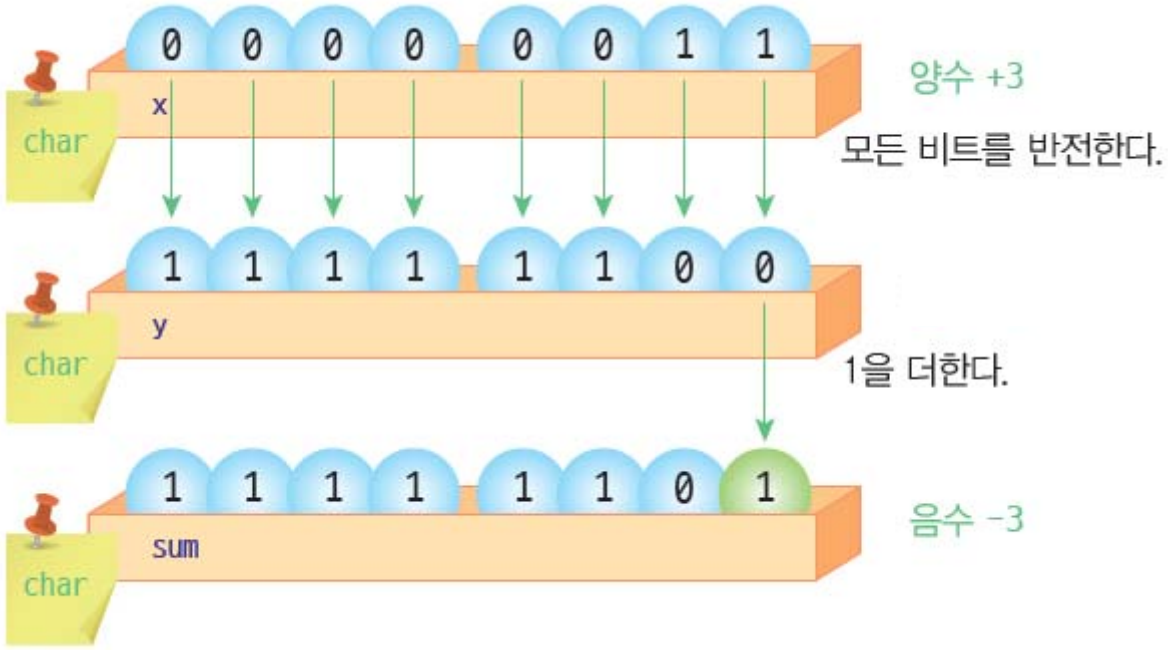
3-3 = 3+(-3)



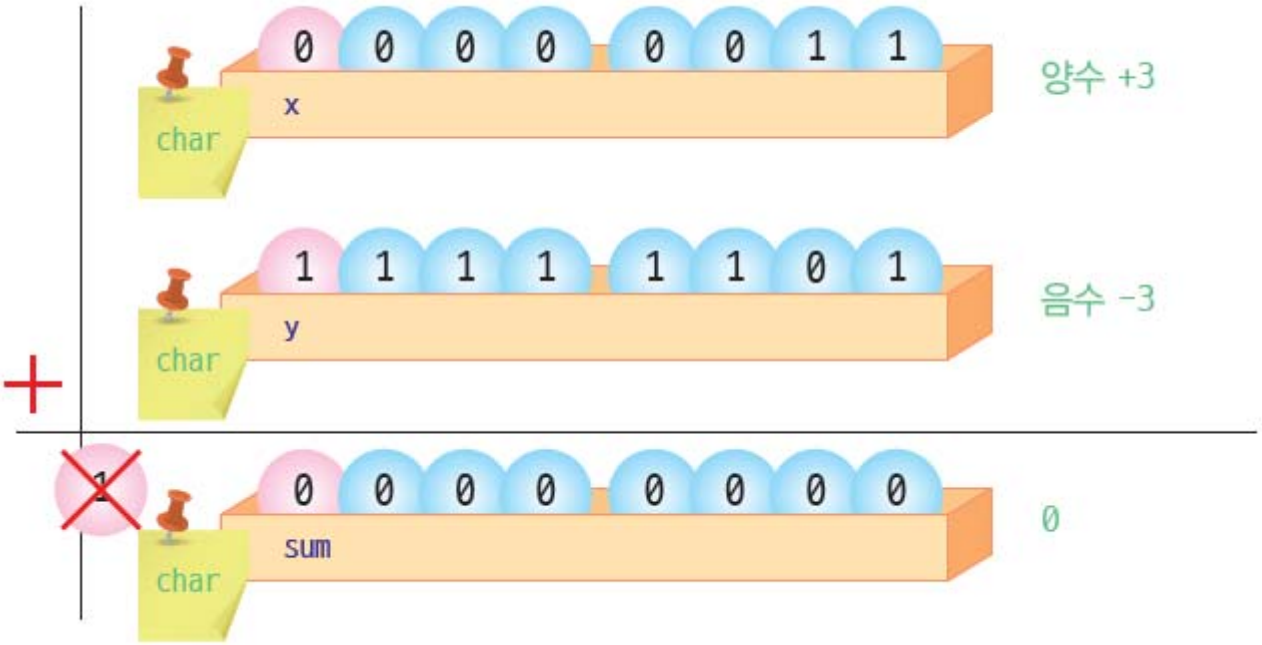
음수를 표현하는 두번째 방법

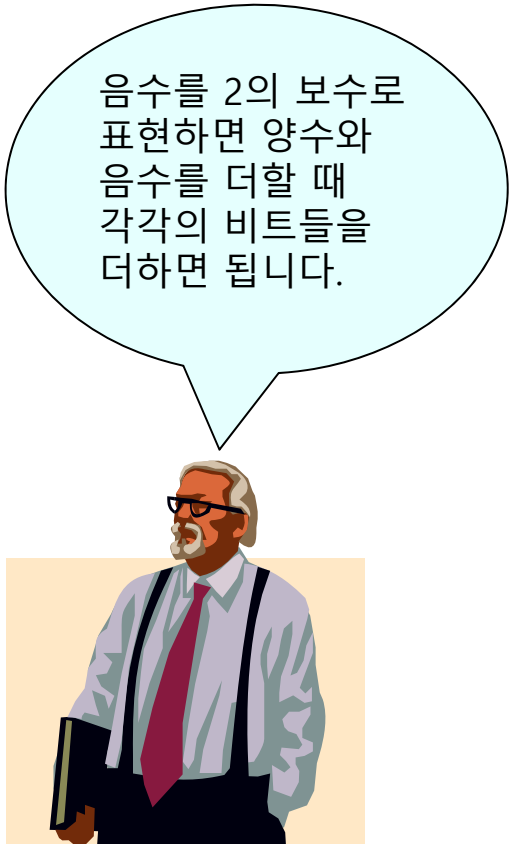
- 2의 보수로 음수를 표현 → 표준적인 음수 표현 방법

- 2의 보수를 만드는 방법



2의 보수로 양수와 음수를 더하면





예제

```
/* 2의 보수 프로그램*/
#include <stdio.h>

int main(void)
{
        int x = 3;
        int y = -3;

        printf("x = %08X\n", x);          // 8자리의 16진수로 출력한다.
        printf("y = %08X\n", y);          // 8자리의 16진수로 출력한다.
        printf("x+y = %08X\n", x+y);      // 8자리의 16진수로 출력한다.

        return 0;
}
```

음수가 2의 보수로
표현되는지를 알아보자.

```
x = 00000003
y = FFFFFFFD
x+y = 00000000
```

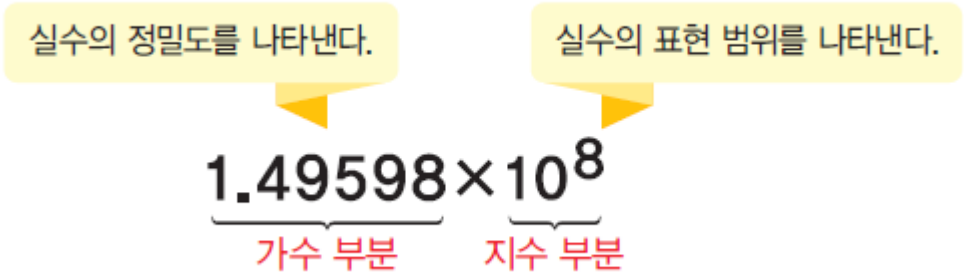
16진수 F → 2진수 1111

중간 점검

- 음수의 표현 방법으로 2의 보수를 사용하는 이유는 무엇인가?
- 2진수 01000011의 2의 보수를 구해보자.

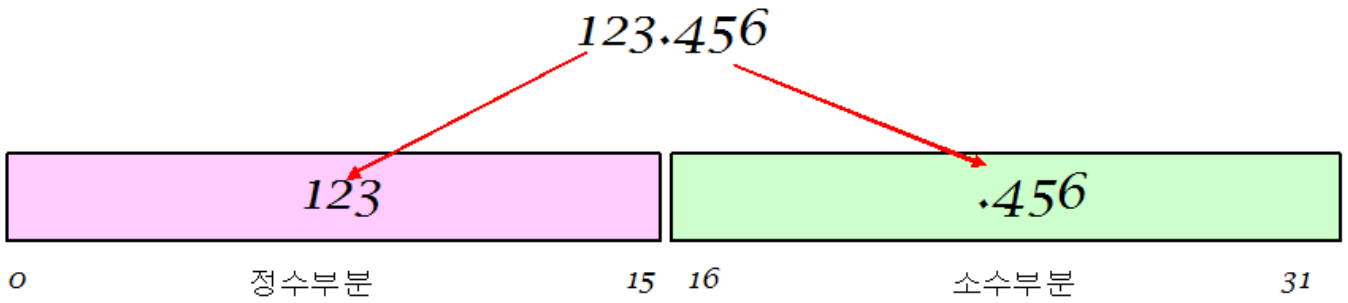
부동소수점형

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



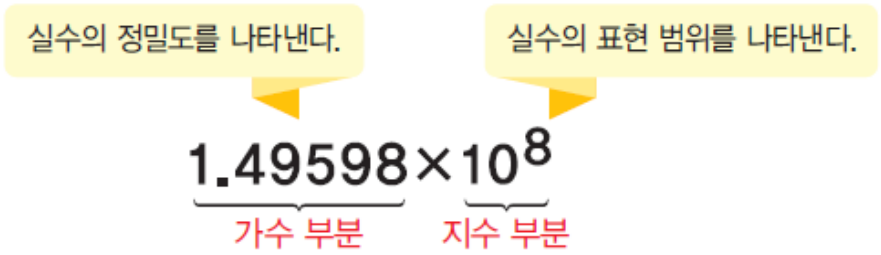
실수를 표현하는 방법

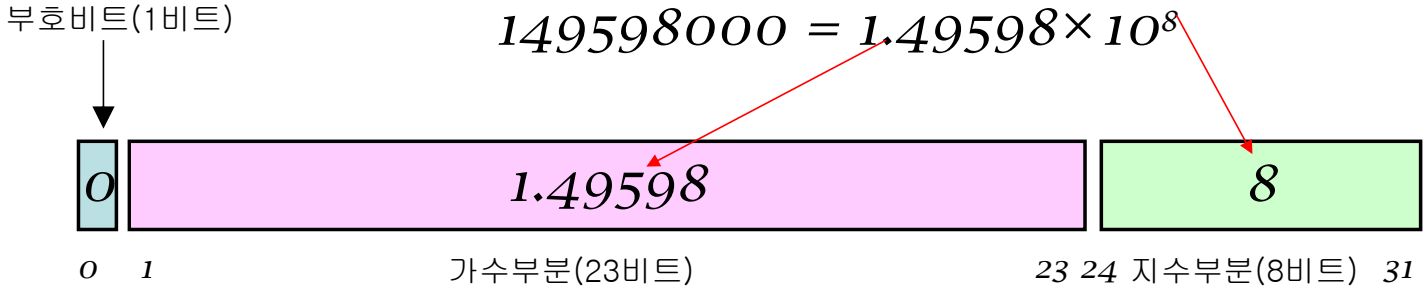
- #1 고정 소수점 방식
  - 정수 부분을 위하여 일정 비트를 할당하고 소수 부분을 위하여 일정 비트를 할당
  - 전체가 32비트이면 정수 부분 16비트, 소수 부분 16비트 할당
  - 과학과 공학에서 필요한 아주 큰 수를 표현할 수 없다
    - 지구와 태양 사이의 거리: 149,598,000km



실수를 표현하는 방법

- #2 부동 소수점 방식





- 표현할 수 있는 범위가 대폭 늘어난다.
- 지수 범위: 10^{-38} 에서 10^{+38}

부동소수점형

자료형	명칭	크기	범위
float	단일 정밀도(single-precision) 부동 소수점	32비트	$\pm 1.17549 \times 10^{-38} \sim \pm 3.40282 \times 10^{+38}$
double long double	두배 정밀도(double-precision) 부동 소수점	64비트	$\pm 2.22507 \times 10^{-308} \sim \pm 1.79769 \times 10^{+308}$

예제

```c
/* 부동 소수점 자료형의 크기 계산*/
#include <stdio.h>
int main(void)
{
        float x = 1.23456789012345678 9;
        double y = 1.23456789012345678 9;

        printf("float의 크기=%d\n", sizeof(float));
        printf("double의 크기=%d\n", sizeof(double));

        printf("x = %30.25f\n",x);
        printf("y = %30.25f\n",y);
       return 0;
}
```

```
float의 크기=4
double의 크기=8
x =     1.2345678806304932000000000
y =     1.2345678901234567000000000
```

부동 소수점 상수

실수	지수 표기법	의미
123.45	1.2345e2	1.2345×10^{2}
12345.0	1.2345e5	1.2345×10^{5}
0.000023	2.3e-5	2.3×10^{-5}
2,000,000,000	2.0e9	2.0×10^{9}

```
1.23456
2.          // 소수점만 붙여도 된다.
.28         // 정수부가 없어도 된다.
2e+10       // +나 -기호를 지수부에 붙일 수 있다.
9.26E3      // 9.26×10³
0.67e-7     // 0.67×10⁻⁹
```

부동 소수점 오버플로우

```
#include <stdio.h>

int main(void)
{
  float x = 1e39;
  printf("x = %e\n",x);
}
```

숫자가 커서 오버플로우 발생

```
x = 1.#INF00e+000
계속하려면 아무 키나 누르십시오 . . .
```

부동 소수점 언더플로우

```
#include <stdio.h>

int main(void)
{
        float x = 1.23456e-38;
        float y = 1.23456e-40;
        float z = 1.23456e-46;

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

숫자가 작아서
언더플로우 발생

```
x = 1.234560e-038
y = 1.234558e-040
z = 0.000000e+000
```

부동 소수점형 사용시 주의사항

- 오차가 있을 수 있다!

```
#include <stdio.h>

int main(void)
{
        double x;

        x = (1.0e20 + 5.0)-1.0e20;
        printf("%f \n",x);
        return 0;
}
```

부동소수점 연산에서는
오차가 발생한다.
5.0이 아니라 0으로 계산
된다.

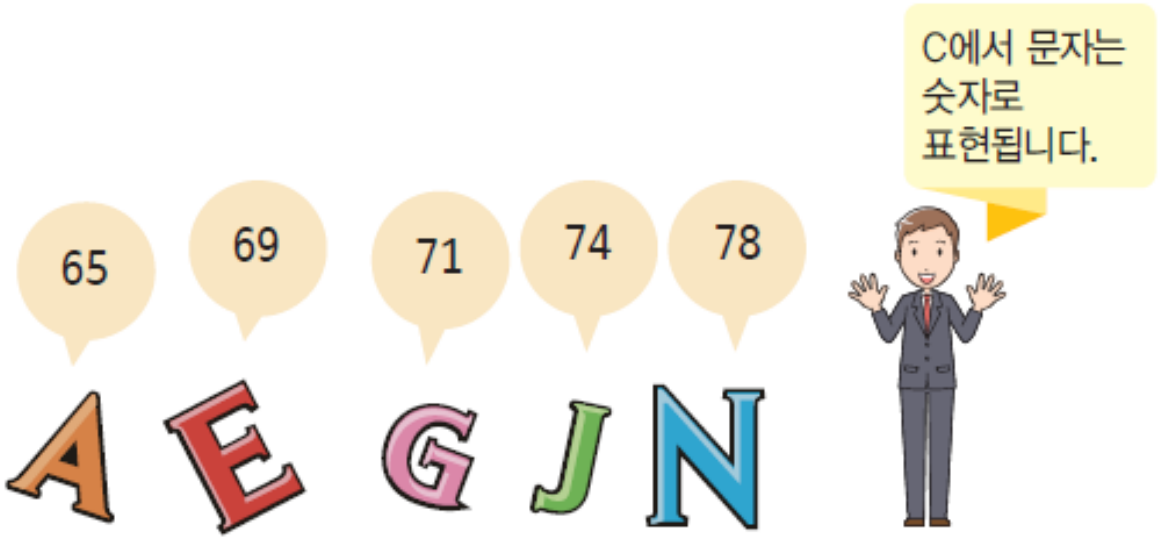
```
0.000000
```

중간 점검

- float형과 double형의 크기는?
- 부동 소수점 상수인 1.0×1025를 지수 형식으로 표기하여 보자.
- 부동 소수점형에서 오차가 발생하는 근본적인 이유는 무엇인가?

문자형

- 문자는 컴퓨터보다는 인간에게 중요
- 문자도 숫자를 이용하여 표현



문자형

- 문자는 컴퓨터보다는 인간에게 중요
- 문자도 숫자를 이용하여 표현

- 공통적인 규격이 필요하다.
- 아스키 코드(**ASCII:** American Standard Code for Information Interchange)

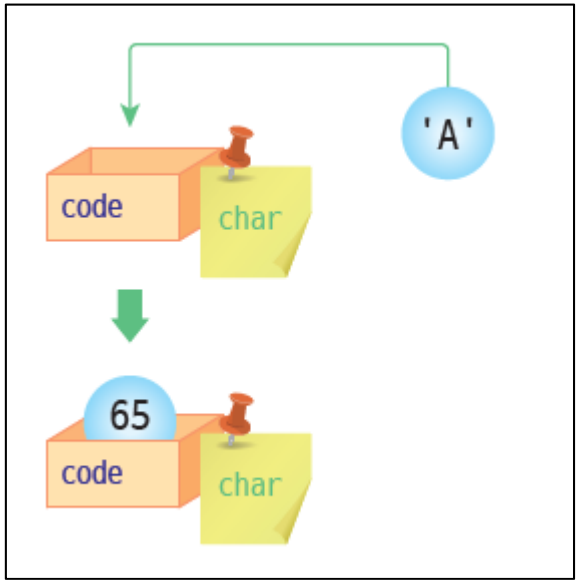
아스키 코드표 (일부)

Dec	Hex	문자	Dec	Hex	문자	Dec	Hex	문자	Dec	Hex	문자
0	0	NULL	20	14	DC4	40	28	(	60	3C	<
1	1	SOH	21	15	NAK	41	29	)	61	3D	=
2	2	STX	22	16	SYN	42	2A	*	62	3E	>
3	3	ETX	23	17	ETB	43	2B	+	63	3F	?
4	4	EOL	24	18	CAN	44	2C	,	64	40	@
5	5	ENQ	25	19	EM	45	2D	-	65	41	A
6	6	ACK	26	1A	SUB	46	2E	.	66	42	B
7	7	BEL	27	1B	ESC	47	2F	/	67	43	C
8	8	BS	28	1C	FS	48	30	0	68	44	D
9	9	HT	29	1D	GS	49	31	1	69	45	E
10	A	LF	30	1E	RS	50	32	2	70	46	F
11	B	VT	31	1F	US	51	33	3	71	47	G
12	C	FF	32	20	space	52	34	4	72	48	H
13	D	CR	33	21	!	53	35	5	73	49	I
14	E	SO	34	22	"	54	36	6	74	4A	J
15	F	SI	35	23	#	55	37	7	75	4B	K
16	10	DLE	36	24	$	56	38	8	76	4C	L
17	11	DC1	37	25	%	57	39	9	77	4D	M
18	12	DC2	38	26	&	58	3A	:	78	4E	N
19	13	DC3	39	27	'	59	3B	;	79	4F	O

문자 변수와 문자 상수

- char형을 사용하여 문자를 저장한다.

```
char code;
code = ‘A’;
```



예제

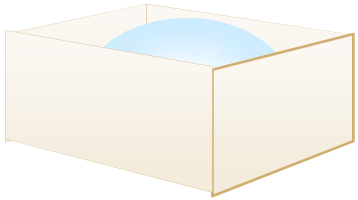
```c
/* 문자 변수와 문자 상수*/
#include <stdio.h>

int main(void)
{
        char code1 = 'A';     // 문자 상수로 초기화
        char code2 = 65;      // 아스키 코드로 초기화

        printf("code1 = %c\n", code1);
        printf("code2 = %c\n", code2);
}
```

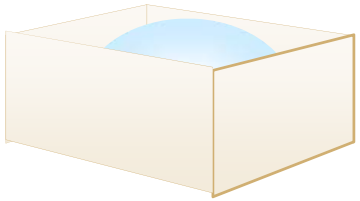
```
code1 = A
code2 = A
```

Quiz

(Q) 1과 '1'의 차이점은?

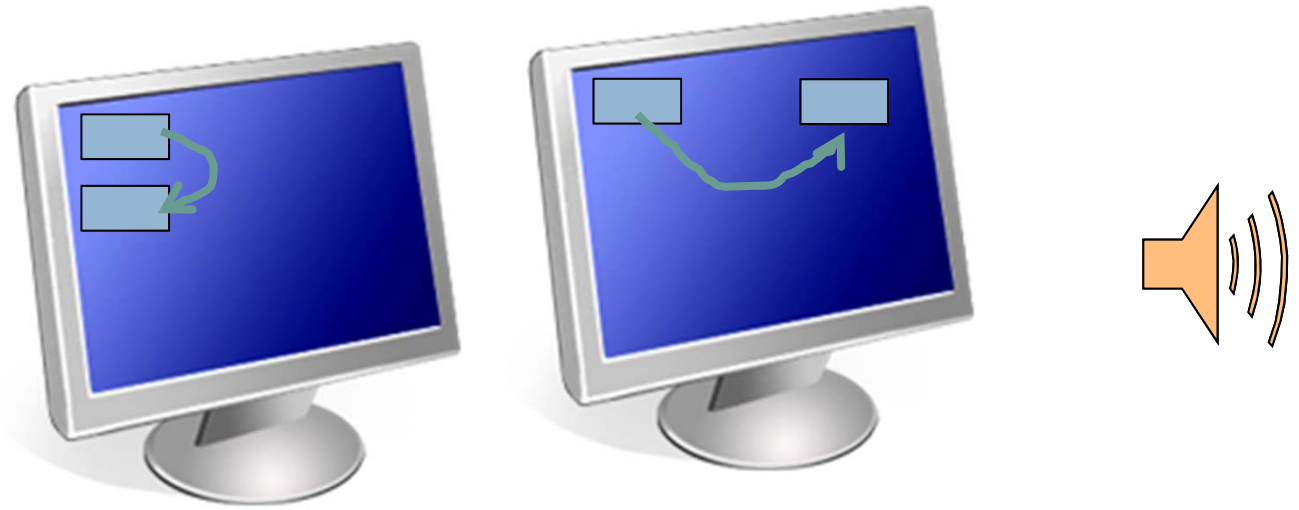
(A) 1은 정수이고 '1'은 문자 '1'을 나타내는 아스키코드이다.





제어 문자

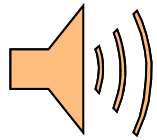
- 인쇄 목적이 아니라 제어 목적으로 사용되는 문자들
  - (예) 줄바꿈 문자, 탭 문자, 벨소리 문자, 백스페이스 문자



제어 문자를 나타내는 방법

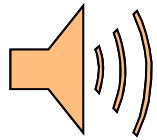
- 아스키 코드를 직접 사용

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



- 이스케이프 시퀀스 (escape sequence) 사용

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



이스케이프 시퀀스

제어 문자	이름	의미
\0	널문자	
\a	경고(bell)	"삐"하는 경고음 발생
\b	백스페이스(backspace)	커서를 현재의 위치에서 한 글자 뒤로 옮긴다.
\t	수평탭(horizontal tab)	커서의 위치를 현재 라인에서 설정된 다음 탭 위치로 옮긴다.
\n	줄바꿈(newline)	커서를 다음 라인의 시작 위치로 옮긴다.
\v	수직탭(vertical tab)	설정되어 있는 다음 수직 탭 위치로 커서를 이동
\f	폼피드(form feed)	주로 프린터에서 강제적으로 다음 페이지로 넘길 때 사용된다.
\r	캐리지 리턴(carriage return)	커서를 현재 라인의 시작 위치로 옮긴다.
\"	큰따옴표	원래의 큰따옴표 자체
\'	작은따옴표	원래의 작은따옴표 자체
\\	역슬래시(back slash)	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.", 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..

```
#include <stdio.h>

int main()
{
        char code = 'A';
        printf("%d %d %d \n", code, code+1, code+2); // 65 66 67이 출력된다.
        printf("%c %c %c \n", code, code+1, code+2); // A B C가 출력된다.
        return 0;
}
```

```
65 66 67
A B C
```

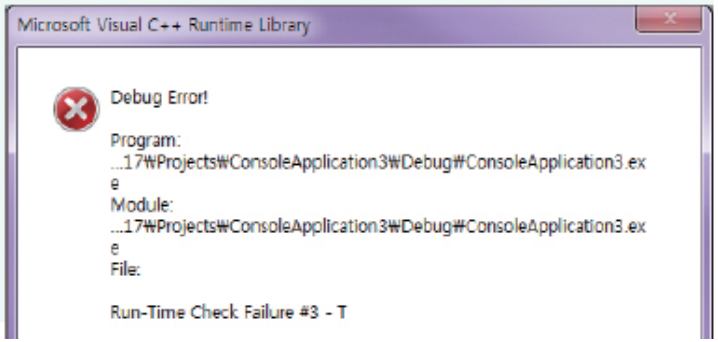
중간 점검

- 컴퓨터에서는 문자를 어떻게 나타내는가?
- C에서 문자를 가장 잘 표현할 수 있는 자료형은 무엇인가?
- 경고음이 발생하는 문장을 작성하여 보자.

무엇이 문제일까?

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

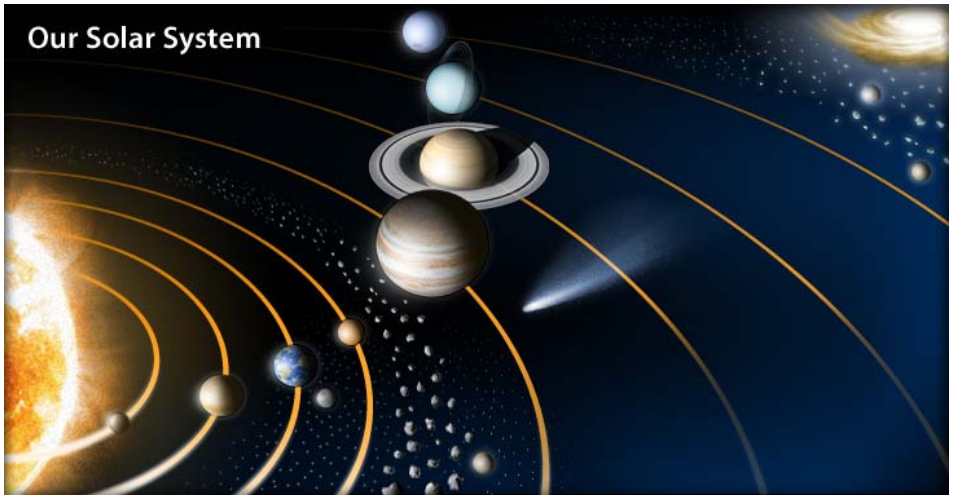
```c
#include <stdio.h>
int main(void)
{
    int x, y, z, sum;

    sum = 0;
    printf("3개의 정수를 입력하세요 (x, y, z): ");
    scanf("%d %d %d", &x, &y, &z);
    sum += x;
    sum += y;
    sum += z;
    printf("3개 정수의 합은 %d\n", sum);
    return 0;
}
```

실행결과

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

mini project: 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.



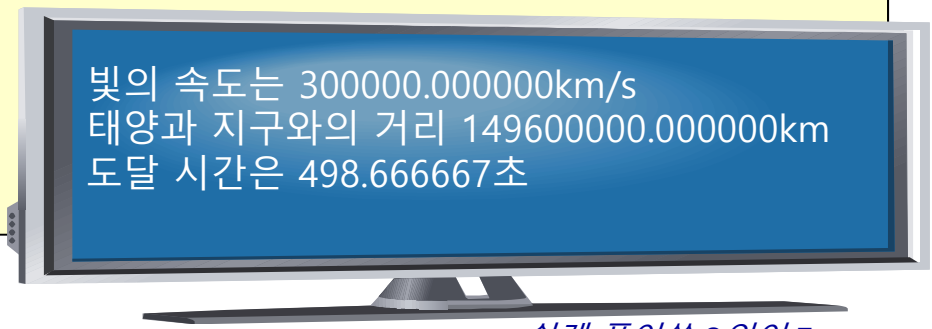
실행 결과

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 %f나 %lf를 사용한다.

소스

```
#include <stdio.h>
int main(void)
{
        double light_speed = 300000;        // 빛의 속도 저장하는 변수
        double distance = 149600000;        // 태양과 지구 사이 거리 저장하는 변수
                                            // 149600000km로 초기화한다.
        double time;                        // 시간을 나타내는 변수

        time = distance / light_speed;      // 거리를 빛의 속도로 나눈다.


        printf("빛의 속도는 %fkm/s \n", light_speed);
        printf("태양과 지구와의 거리 %fkm \n", distance);
        printf("도달 시간은 %f초\n", time);        // 시간을 출력한다.

        return 0;
}
```



도전문제

- 위의 프로그램의 출력은 498.666667초로 나온다. 이것을 분과 초로 나누어서 8분 20초와 같은 식으로 출력하도록 변경하라. 나머지를 계산하는 연산자는 %이다. 추가적인 정수 변수를 사용하여도 좋다.

Q & A