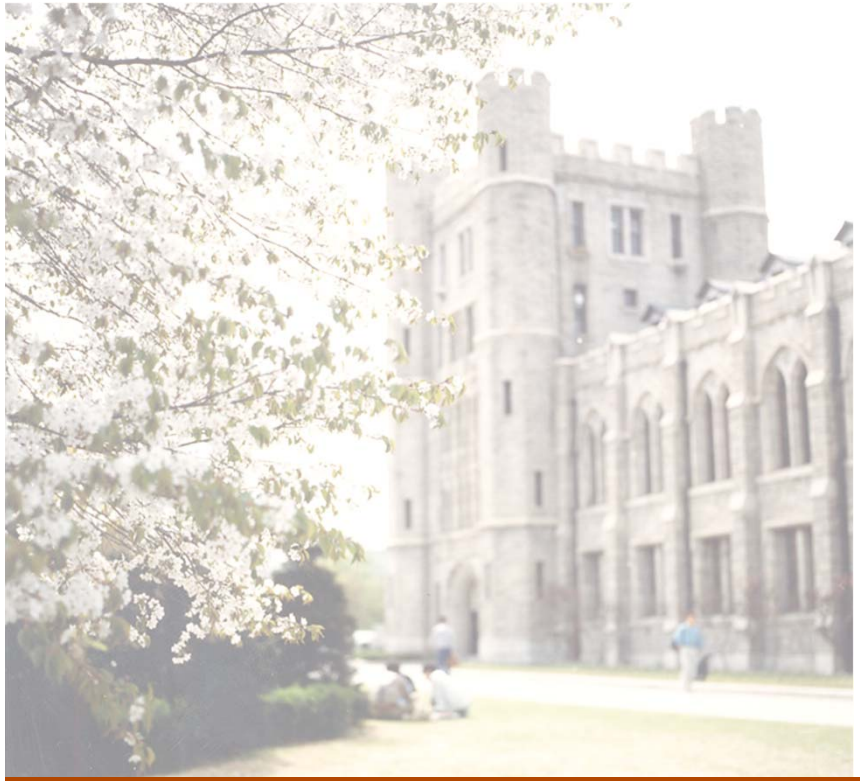
DAEGU UNIVERSITY

42경산 라피신(La piscine) 대비 사전 SW교육과정 - C



Heungwoo Nam

Computer Engineering
Daegu University
2023. 7. 11

Objective & Contents

❑ 수업목표

- 변수와 자료형의 이해

❑ Contents

- 4.1 변수와 상수
- 4.2 자료형
  - 정수형, 부동소수점형, 문자형
- 4.3 정수형
  - 정수형 범위, signed & unsigned,
  - 정수상수(10진법, 8진법, 16진법), 기호상수
  - 정소표현방법(2의보수)
- 4.4 부동 소수점형
  - 고정 소수점 방식, 부동 소수점 방식
- 4.5 문자형
  - ASCII코드, 문자상수, 제어문자

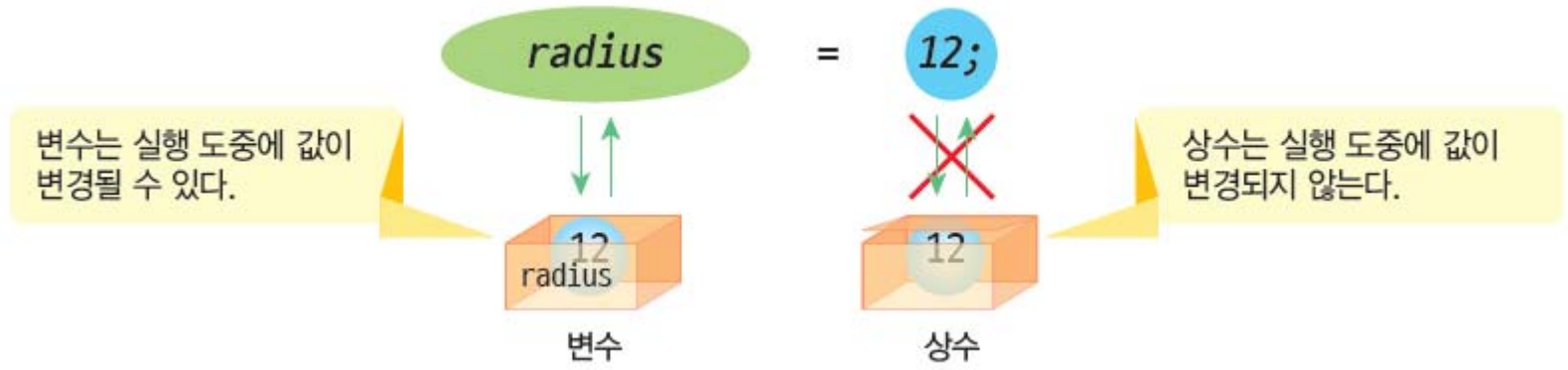
4.1 변수와 상수

❑ 변수(variable)

- 저장된 값의 변경이 가능한 공간

❑ 상수(constant)

- 저장된 값의 변경이 불가능한 공간
  - (예) 3.14, 100, 'A', "Hello World!"



4.1 변수와 상수

❑ 예제

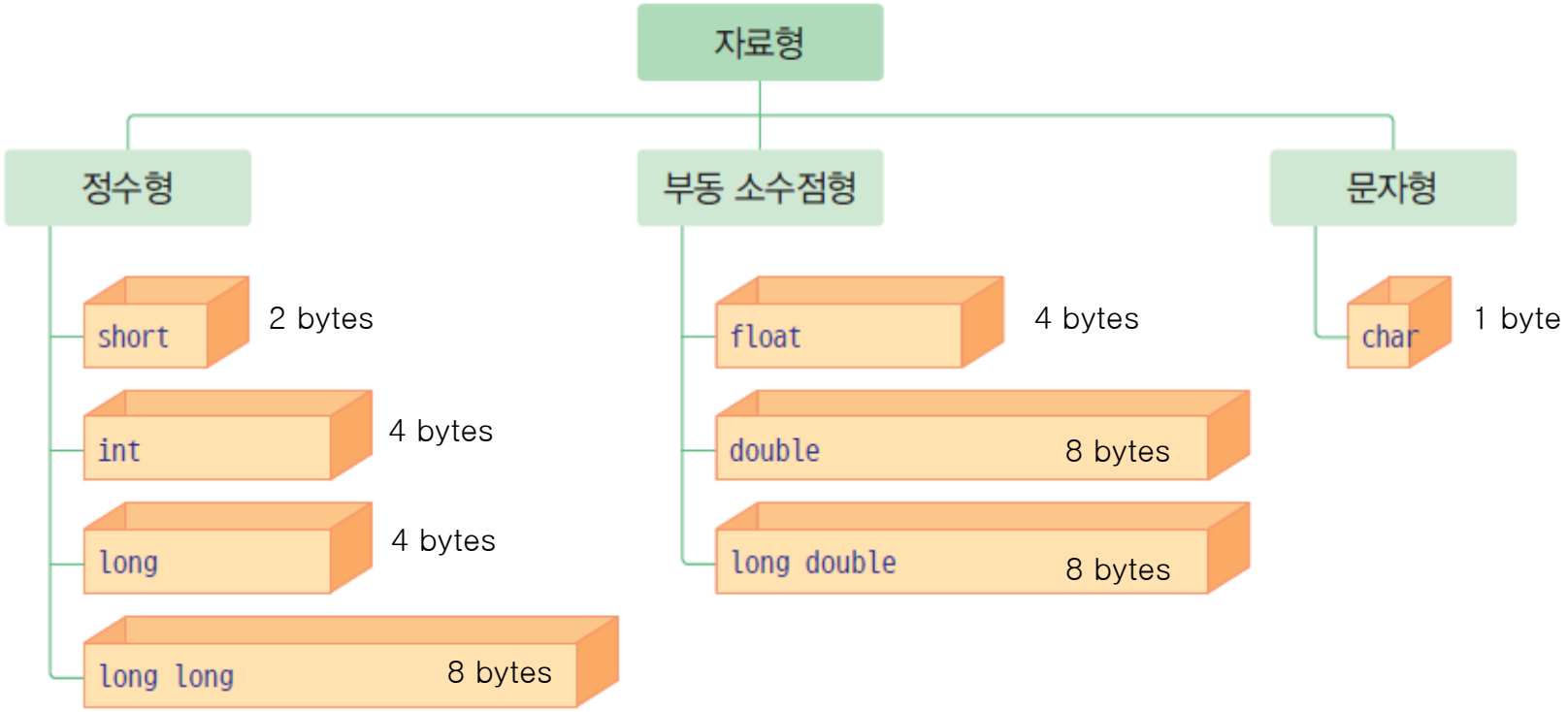
```
/* 원의 면적을 계산하는 프로그램 */
#include <stdio.h>
int main(void)
{
        float radius;       // 원의 반지름
        float area;         // 원의 면적

        printf("원의 면적을 입력하시요:");
        scanf("%f", &radius);

        area = 3.141592 * radius * radius;
        printf("원의 면적: %f \n", area);

        return 0;
}
```

변수

상수

4.2 자료형

❑ 자료형(data type)



4.2 자료형

❑ 예제: 자료형의 크기

- sizeof(): 변수나 자료형의 크기를 byte 수로 반환하는 연산자

```
#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("long long형의 크기: %d\n", sizeof(long long));
        printf("float형의 크기: %d\n", sizeof(float));
        printf("double형의 크기: %d\n", sizeof(double));
        return 0;
}
```

실행결과

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

4.3 정수형

❑ 정수형의 범위

- short grade; // short형의 변수를 생성한다.

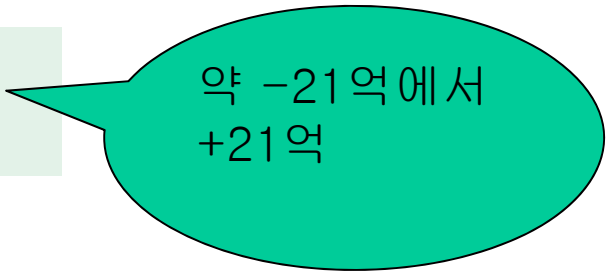
-2^{15}, ..., -2, -1, 0, 1, 2, ..., 2^{15} -1
(-32768 ~ +32767)

- int count; // int형의 변수를 생성한다.

-2^{31}, ..., -2, -1, 0, 1, 2, ..., 2^{31} -1
(-2147483648 ~ +2147483647)

- long distance; // distance형의 변수를 생성한다.

-2^{31}, ..., -2, -1, 0, 1, 2, ..., 2^{31} -1
(-2147483648 ~ +2147483647)



4.3 정수형

❑ 예제

```c
/* 정수형 자료형의 크기를 계산하는 프로그램*/
#include <stdio.h>

int main(void)
{
        short year = 0;              // 0으로 초기화한다.
        int sale = 0;               // 0으로 초기화한다.
        long total_sale = 0;         // 0으로 초기화한다.
        long long large_value;    // 64비트 자료형

        year = 10;                    // 약 3만2천을 넘지 않도록 주의
        sale = 200000000;             // 약 21억을 넘지 않도록 주의
        total_sale = year * sale;   // 약 21억을 넘지 않도록 주의

        printf("total_sale = %d \n", total_sale);

         return 0;
}
```

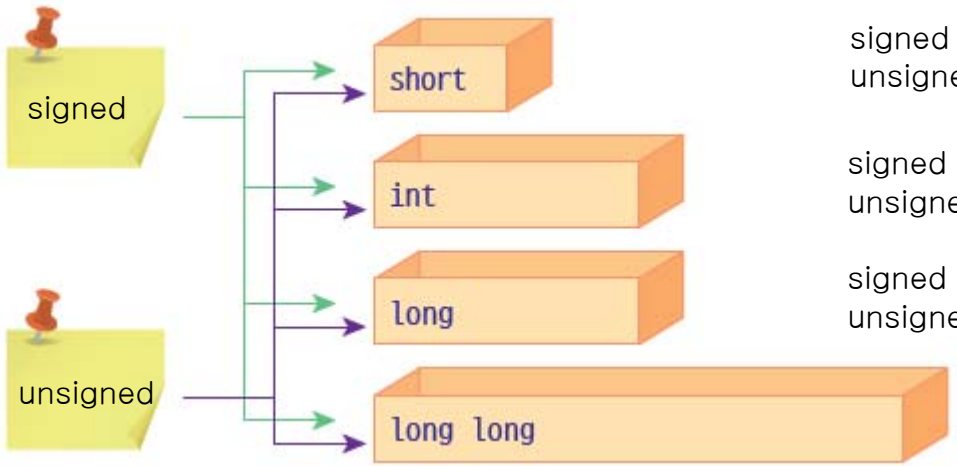
실행결과

```
total_sale = 2000000000
```

4.3 정수형

❑ signed, unsigned 수식자

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



signed short → short
unsigned short (0, 1, 2,…, 2^{16} −1 (65,535)

signed int → int
unsigned int (0, 1, 2,…, 2^{32} −1 (4,294,967,295)

signed long → long
unsigned long (0, 1, 2,…, 2^{32} −1 (4,294,967,295)

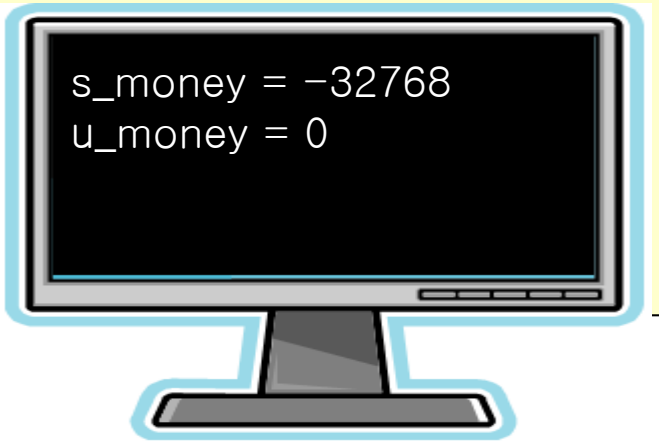
signed long long→ long long
unsigned long long (0, 1, 2,…, 2^{64} −1)

4.3 정수형

❑ 오버플로우 (Overflow)

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

int main(void)
{
        short s_money = SHRT_MAX;          // 최대값으로 초기화한다. 32767
        unsigned short u_money = USHRT_MAX; // 최대값으로 초기화한다. 65535

        s_money = s_money + 1;
        printf("s_money = %d", s_money);

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

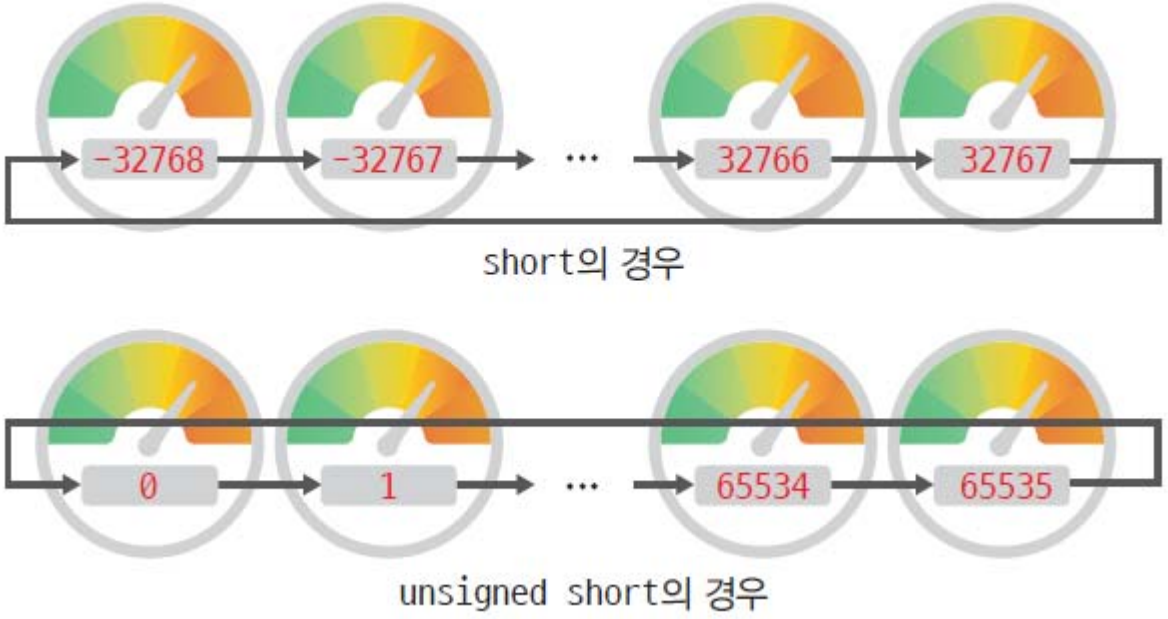
오버플로우 발생!!



4.3 정수형

❑ 오버플로우 (Overflow)

- 규칙성이 있다.
  - 수도 계량기나 자동차의 주행거리계와 비슷하게 동작



4.3 정수형

❑ 정수상수: 10진법, 8진법, 16진법

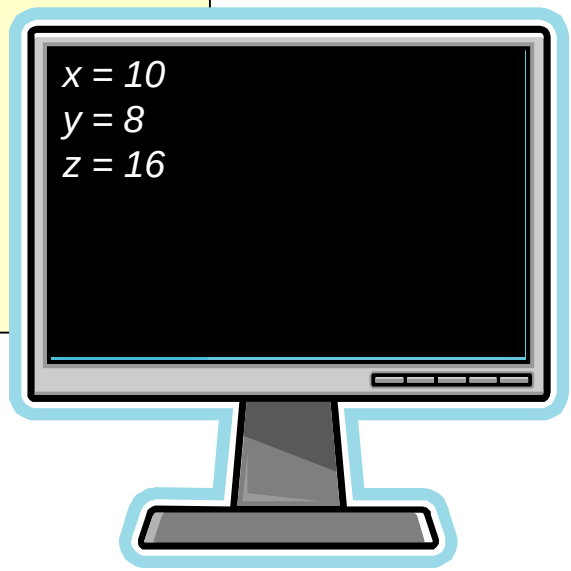
- 8진법
  - $012_8 = 1 \times 8^1 + 2 \times 8^0 = 10$
- 16진법
  - $0xA_{16} = 10 \times 16^0 = 10$

10진수	8진수	16진수
0	00	0x0
1	01	0x1
2	02	0x2
3	03	0x3
4	04	0x4
5	05	0x5
6	06	0x6
7	07	0x7
8	010	0x8
9	011	0x9
10	012	0xa
11	013	0xb
12	014	0xc
13	015	0xd
14	016	0xe
15	017	0xf
16	020	0x10
17	021	0x11
18	022	0x12

4.3 정수형

❑ 정수상수: 예제

```
/* 정수 상수 프로그램*/
#include <stdio.h>

int main(void)
{
      int x = 10;   // 10은 10진수이고 int형이고 값은 십진수로 10이다.
      int y = 010;  // 010은 8진수이고 int형이고 값은 십진수로 8이다.
      int z = 0x10; // 0x10은 16진수이고 int형이고 값은 십진수로 16이다.

      printf("x = %d", x);
      printf("y = %d", y);
      printf("z = %d", z);

      return 0;
}
```



4.3 정수형

❑ 기호상수

- *기호 상수(symbolic constant):* 기호를 이용하여 상수를 표현한 것
- (예)
  - area = 3.141592 * radius * radius;
  - area = PI * radius * radius;

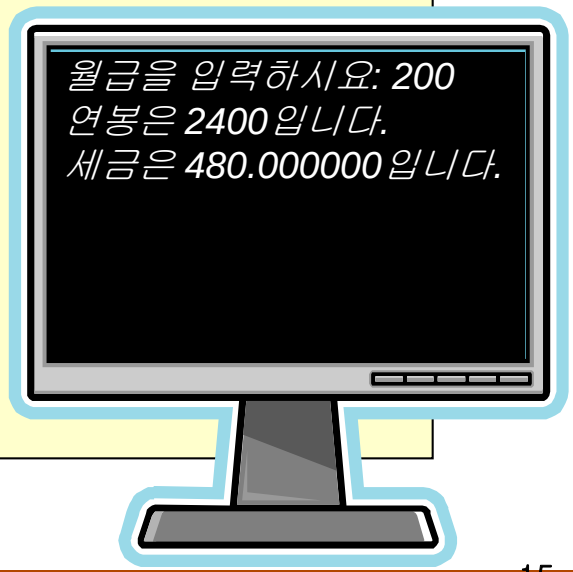
  - income = salary – 0.15 * salary;
  - income = salary – TAX_RATE * salary;

4.3 정수형

❑ 기호상수 예제

```
#include <stdio.h>
#define TAX_RATE  0.2

int main(void)
{
        const int MONTHS = 12;
        int m_salary, y_salary;          // 변수 선언

        printf( "월급을 입력하시요: ");   // 입력 안내문
        scanf("%d", &m_salary);
        y_salary = MONTHS * m_salary;// 순수입 계산
        printf("연봉은 %d입니다.",  y_salary);
        printf("세금은 %f입니다.",  y_salary*TAX_RATE);

        return 0;
}
```

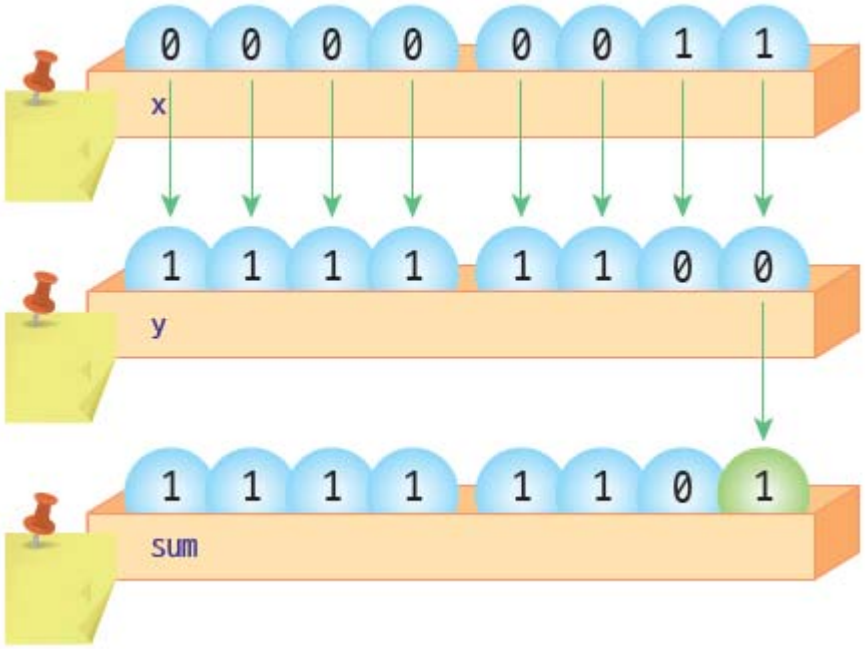
기호상수



4.3 정수형

❑ 음수를 표현하는 두번째 방법

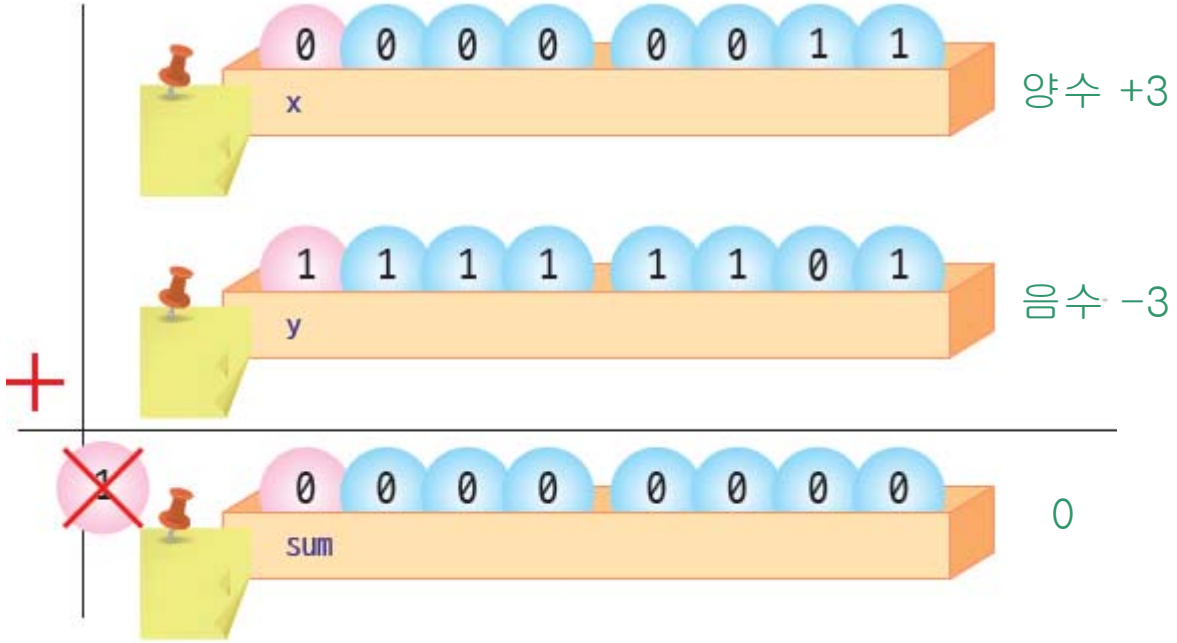
- 2의 보수로 음수를 표현 -> 표준적인 음수 표현 방법
- 2의 보수를 만드는 방법



4.3 정수형

❑ 음수를 표현하는 두번째 방법

- 2의 보수로 양수와 음수를 더하면
  - (예) +3 + (−3)

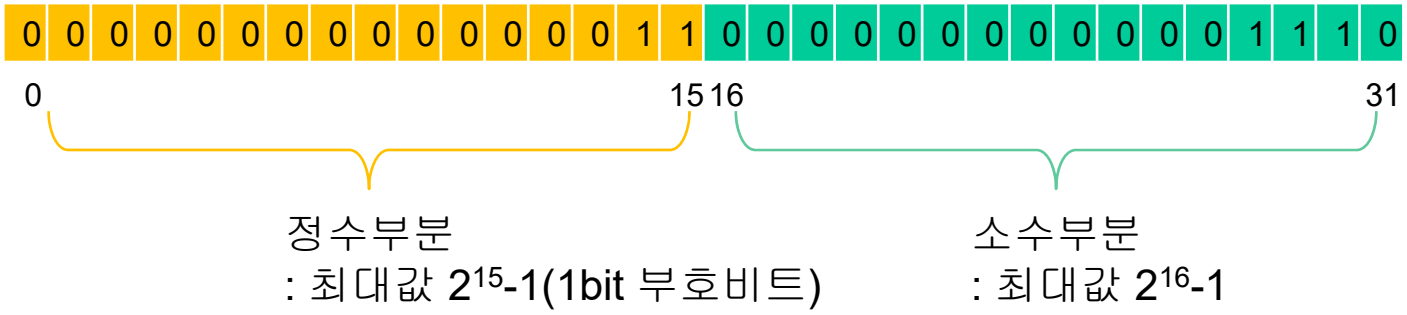


4.4 부동 소수점형

❑ 컴퓨터에서 실수를 나타내는 방법

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

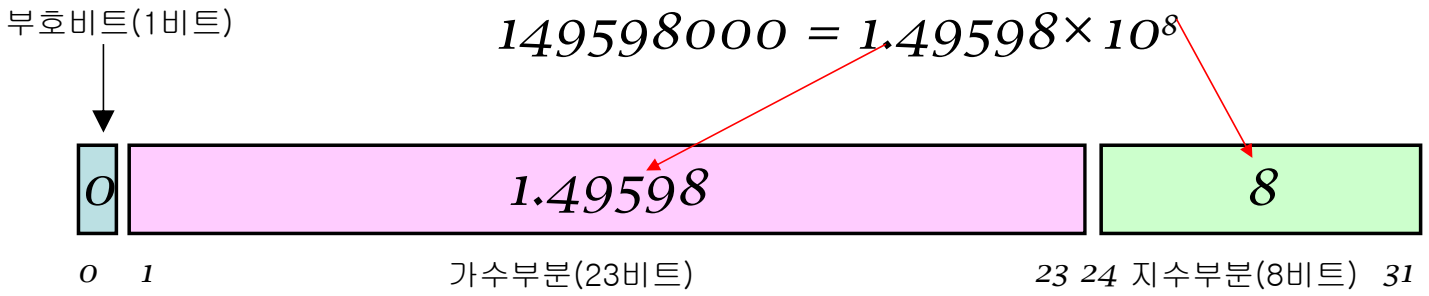
Ex) 3.14



4.4 부동 소수점형

❑ 컴퓨터에서 실수를 나타내는 방법

- #2 부동 소수점 방식
  - 소수점의 위치가 떠서 움직인다는 의미, 즉 소수점의 위치를 움직임으로써 한정된 비트로 정밀도를 보다 높게 표현할 수 있음.
  - 과학자들이 많이 사용하는 과학적 표기법과 유사



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

4.4 부동 소수점형

❑ 부동 소수점 상수

- 부동 소수점 상수는 기본적으로 소수점을 이용하여 표현하고, double형으로 저장됨.
  1) 3.141592 // double 형 상수 (64비트)
  2) 3.141592F // float형 상수 (32비트)

- 부동 소수점 상수의 지수 표기법

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}

- 유효한 부동 소수점 상수의 예

```
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-9     // 0.67×10⁻⁹
```

4.4 부동 소수점형

❑ 부동 소수점 오버플로우 (Overflow)

- 변수에 대입된 수가 너무 커서 변수가 저장할 수 없는 상황 의미
- **float**형 변수는 약 $1x10^{38}$ 이상을 넘는 수는 저장하지 못함.
  이 보다 큰 값을 대입하면 오버플로우 발생.

```
#include <stdio.h>

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

숫자가 커서 오버플로우 발생

```
x = inf
계속하려면 아무 키나 누르십시오 . . .
```

- 오버플로우가 발생하면 컴파일러는 해당 변수에 무한대를 의미하는 inf 대입 및 출력함.

4.4 부동 소수점형

❑ 부동 소수점 언더플로우 (Underflow)

- 부동 소수점 수가 너무 작아서 표현하기가 힘든 상황을 의미
- **float**형 변수는 **1.23456x10^{-38}** 부근이 최대 정밀도를 가지면서 비교적 작은 수가 됨. 언더플로우시 **0**으로 출력함

```
#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);
}
```

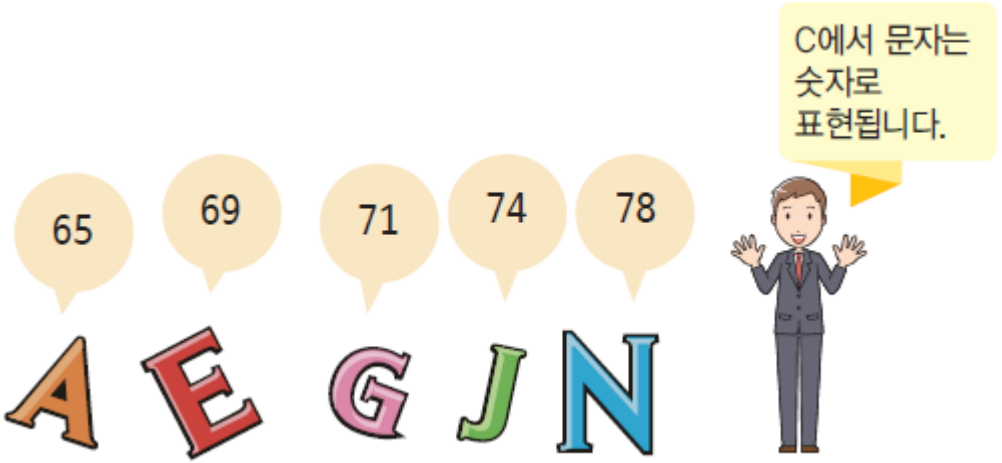
숫자가 작아서
언더플로우 발생

```
x = 1.234560e-038
y = 1.234558e-040   //이미 부정확한 수로 결과 출력됨
z = 0.000000e+000
```

4.5 문자형

❑ 문자와 아스키 코드

- 문자는 컴퓨터보다는 인간에게 중요
- 문자도 숫자를 이용하여 표현
- 공통적인 규격이 필요하다.
  - 아스키 코드(ASCII: American Standard Code for Information Interchange)



4.5 문자형

❑ 문자와 아스키 코드

*	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	₩	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

4-bit (b_4, b_3, b_2, b_1)

3-bit (b_7, b_6, b_5)

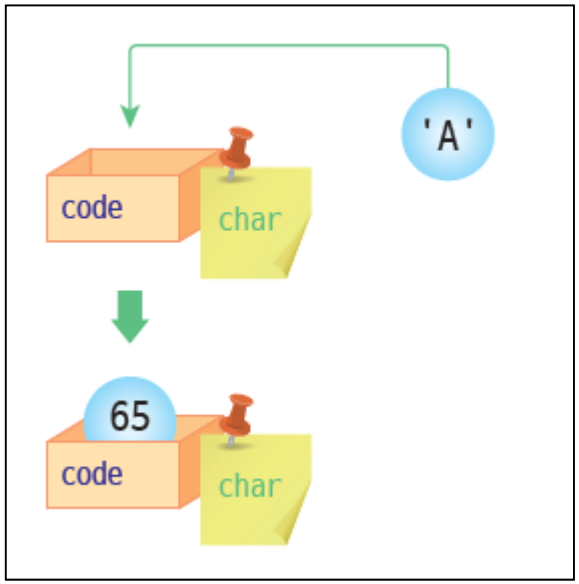
. A=$(41)_{16}$=$(100\ 0001)_2$=65
. a=$(61)_{16}$=$(110\ 0001)_2$=97

4.5 문자형

❑ 문자 변수와 문자 상수

- 문자 변수: char형을 사용하여 문자 저장
  - 문자가 정수로 저장됨
  - 8비트로 2^8=256개의 문자 저장 가능
- 문자 상수: 'A' 처럼 작은 따옴표로 감싸진 문자

```
char code;

code = 'A';
     [==]
code = 65;
```



4.5 문자형

❑ 문자 변수와 문자 상수 예제

```
/* 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
```

4.5 문자형

❑ 이스케이프 시퀀스(escape sequence): 특수 문자열

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

4.5 문자형

❑ 정수형으로서의 char형

- 8비트의 정수를 저장하는데 char 형을 사용할 수 있음

```
#include <stdio.h>

int main(void)
{
        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
```

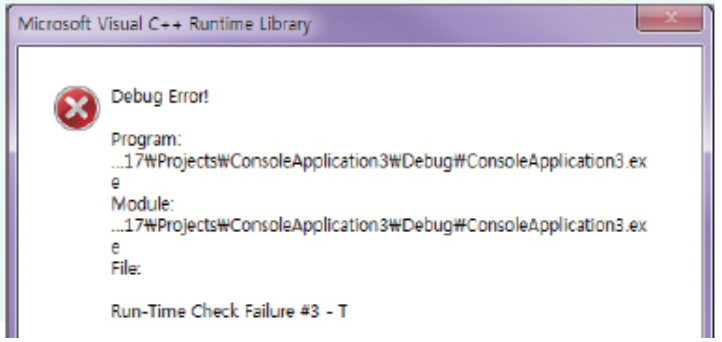
4.5 문자형

❑ 변수의 초기값

- 무엇이 문제일까?

sum_error.c

```
1   #include <stdio.h>
2   int main(void)
3   {
4       int x, y, z, sum;                          //sum은 초기화가
5       printf("3개의 정수를 입력하세요 (x, y, z): ");    //없어서 쓰레기값으로 채워짐
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  }
```



4.5 문자형

❑ 변수의 초기값

- 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
```