2009 Spring

Computer Engineering Programming 1

Lesson 3
- 제4장 변수와 자료형

Lecturer: JUNBEOM YOO
jbyoo@konkuk.ac.kr

※ 본 강의자료는 생능출판사의 "PPT 강의자료"를 기반으로 제작되었습니다.

이번 장에서 학습할 내용

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

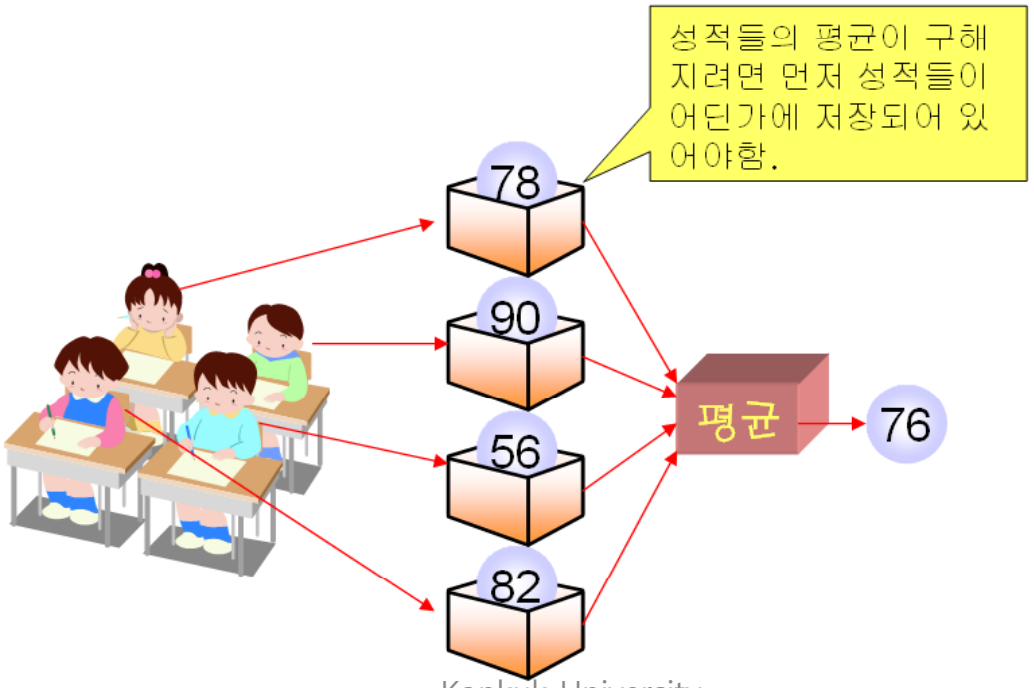
변수

Q) 변수(variable)이란 무엇인가?

A) 프로그램에서 일시적으로 데이터를 저장하는 공간

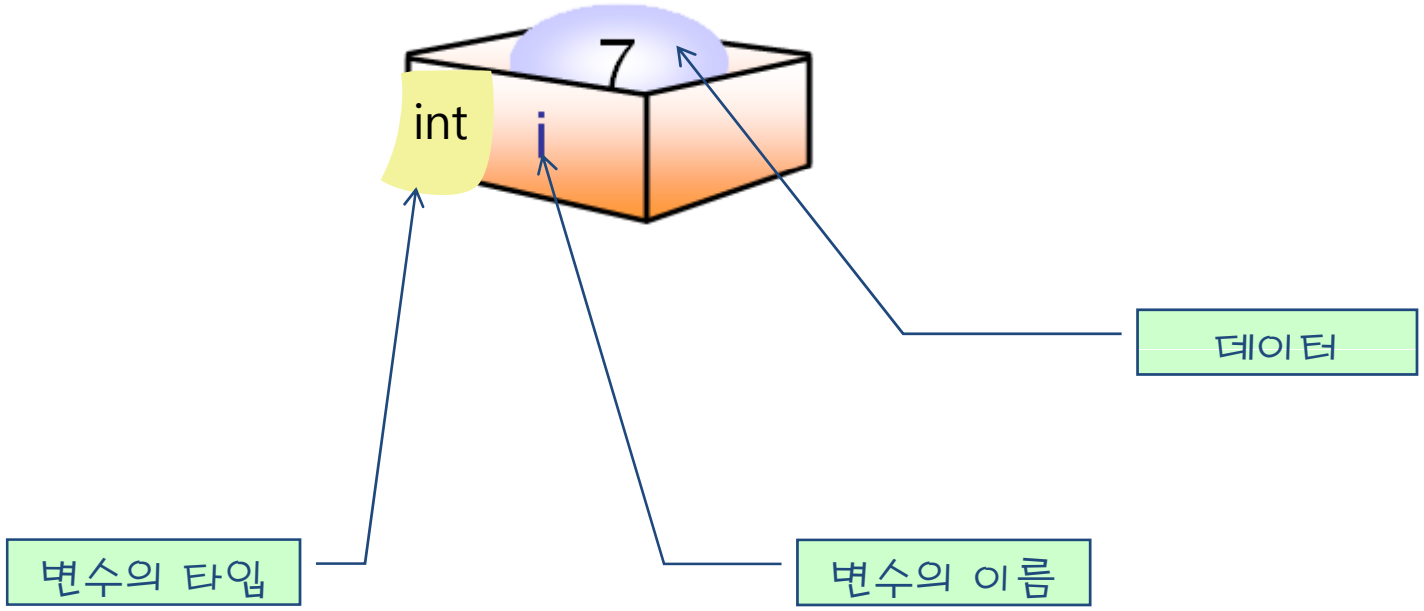
Q) 변수는 왜 필요한가?

A) 데이터가 입력되면 어딘가에 저장해야만 다음에 사용할 수 있다.



변수 = 상자

- 변수는 물건을 저장하는 상자와 같다.



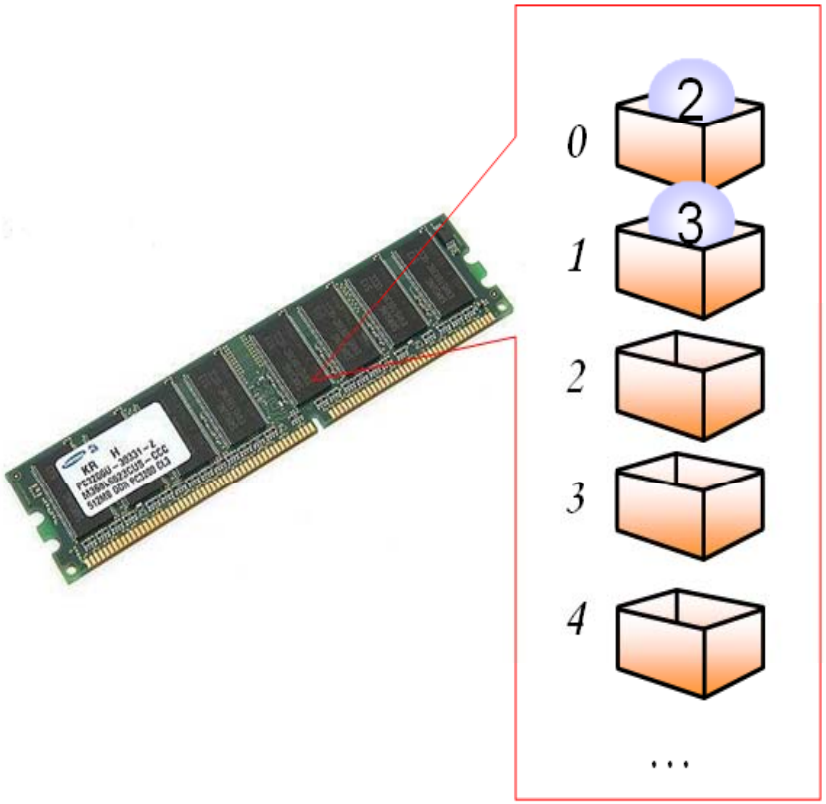
변수가 만들어지는 곳

- 변수는 메인 메모리에 만들어진다.

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

"100번지에 0을 대입하라"

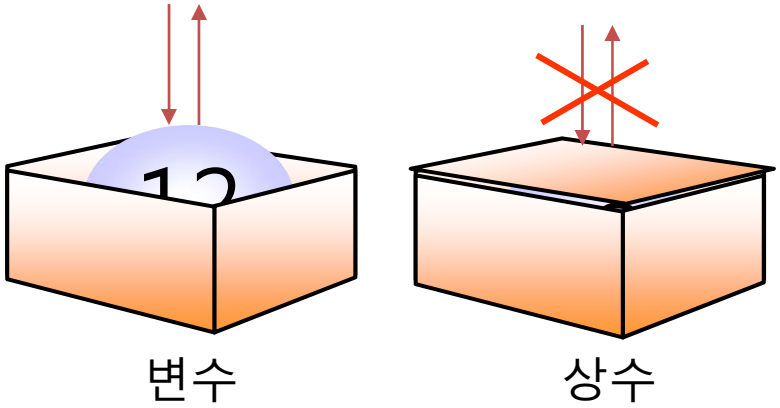
(A) 충분히 가능하지만 불편하다. 인간은 숫자보다는 기호를 더 잘 기억한다.



변수와 상수

- 변수(variable): 저장된 값의 변경이 가능한 공간
- 상수(constant): 저장된 값의 변경이 불가능한 공간

(예) 3.14, 100, 'A', "Hello World!"



(Q) 상수도 이름을 가질 수 있는가?

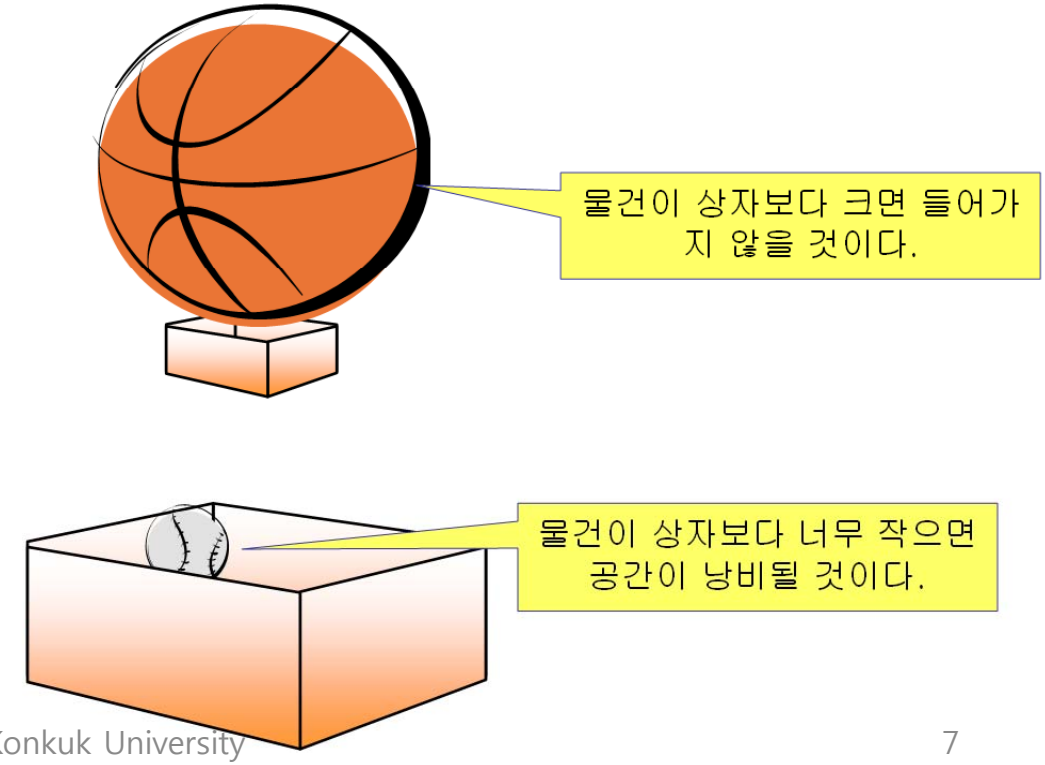
(A) 보통 상수는 이름이 없다. 이러한 상수를 리터럴(literal)이라고 한다. 하지만 필요하다면 상수에도 이름을 붙일 수 있다. 이것을 기호 상수라고 한다.

자료형

- 자료형(data type): 데이터의 타입(종류)
  - (예) 정수형 데이터(100)
  - 실수형 데이터(3.141592)

(Q) 디양한 자료형이 필요한 이유는?

(A) 상자에 물건을 저장하는 과 같다.

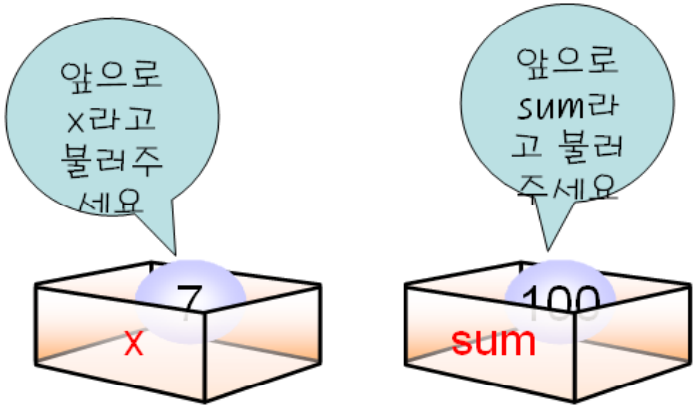


자료형의 종류

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

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
dollor$ X // $기호
double X // 키워드

키워드

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

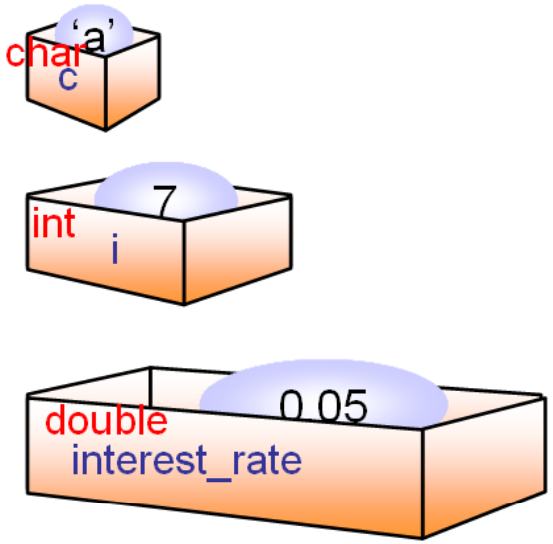
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

변수 선언

- 변수 선언: 컴파일러에게 어떤 변수를 사용하겠다고 미리 알리는 것

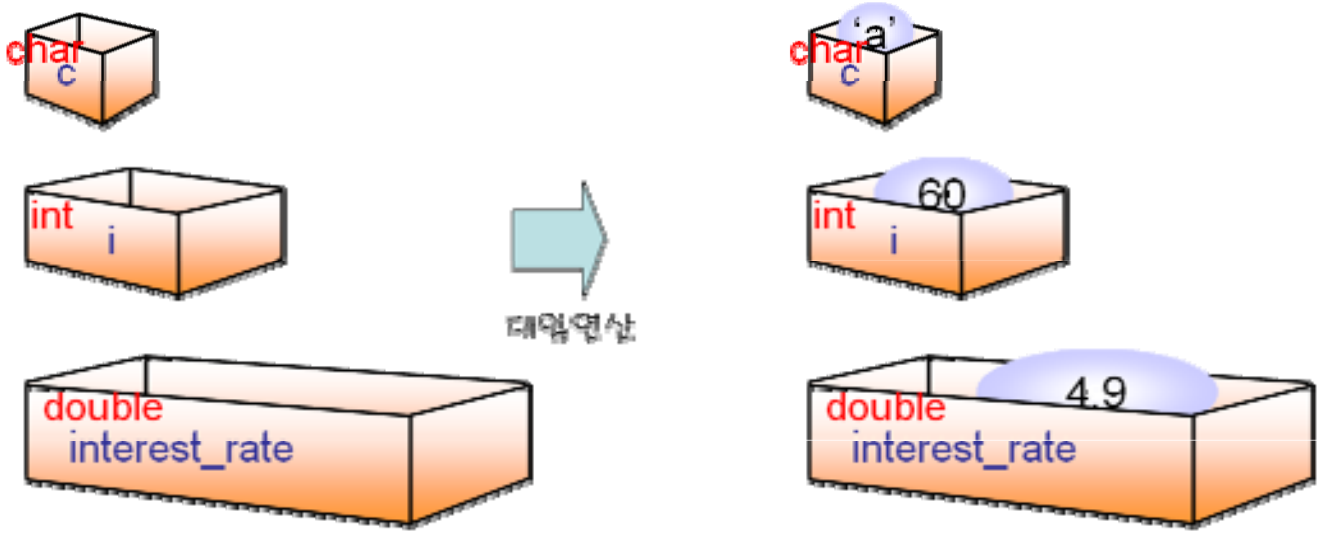
```
자료형      변수이름;
```

- 변수 선언의 예
  - char c;
  - int I;
  - double interest_rate;
  - int height, width;



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 82.9를 대입
```

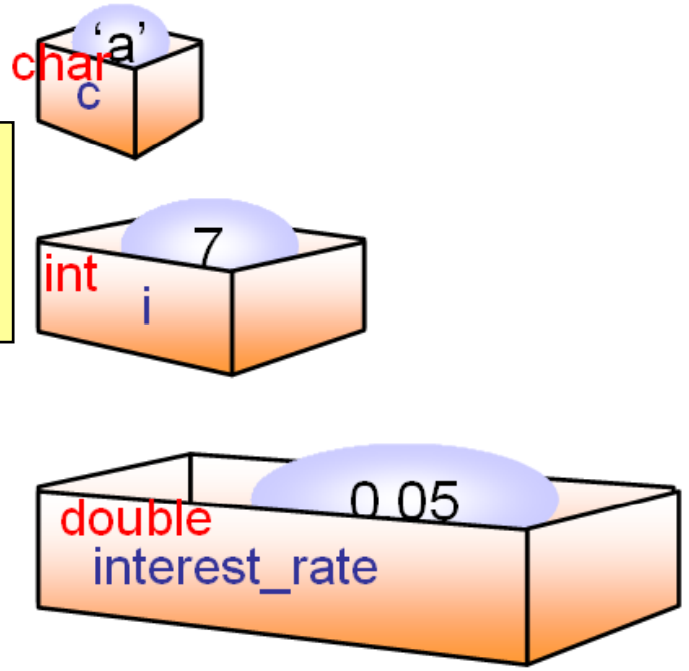


변수의 초기화

```
자료형    변수이름 = 초기값;
```

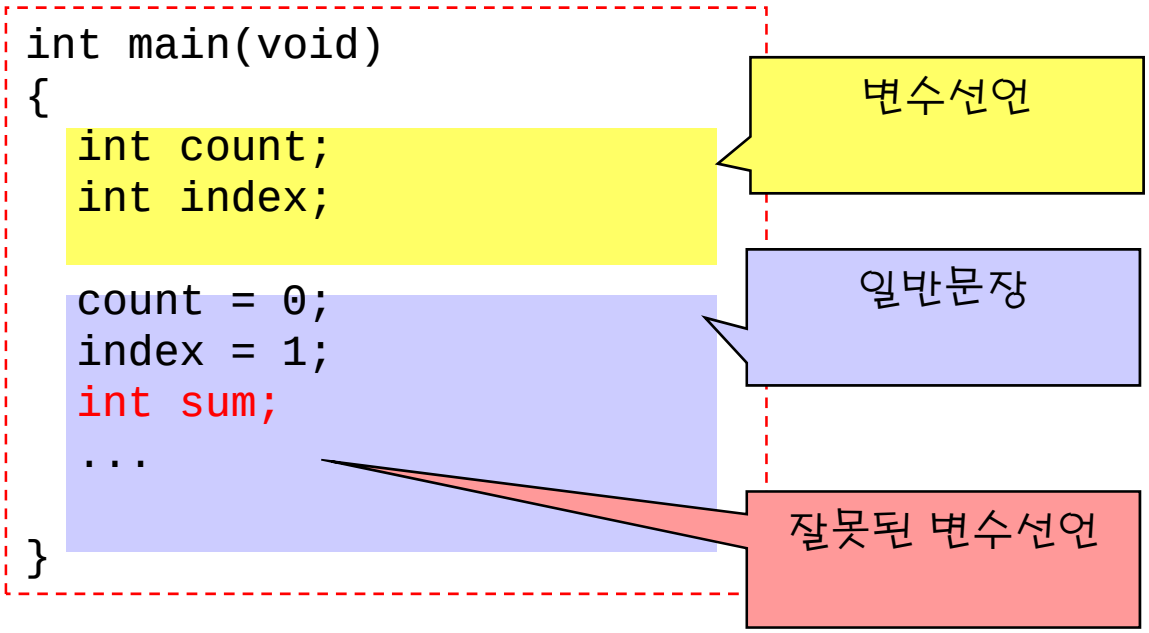
- 변수 초기화의 예

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



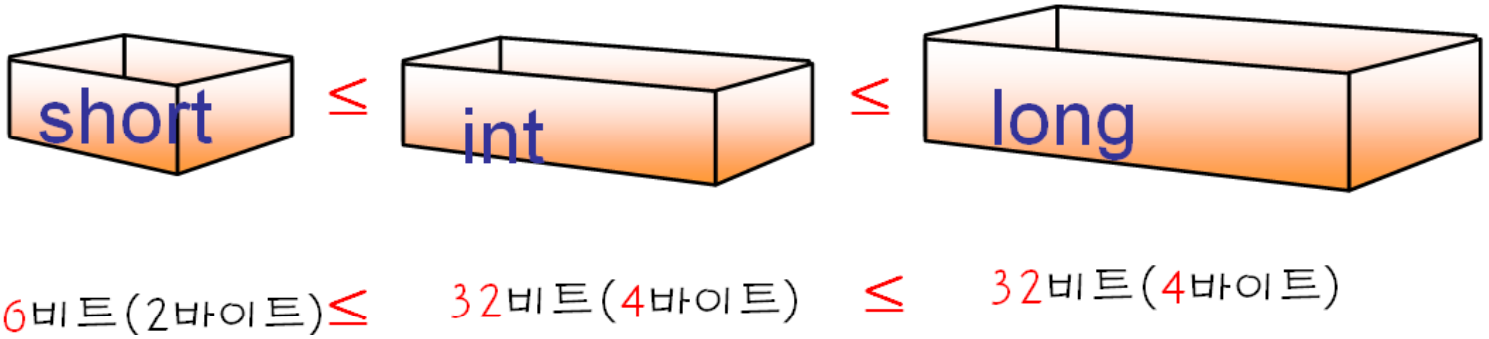
변수 선언 위치

- 변수는 함수의 첫부분에서만 선언할 수 있습니다.



정수형

- short, int, long



- 가장 기본이 되는 것은 int
  - CPU에 따라서 크기가 달라진다.
  - 16비트, 32비트, 64비트

(Q) 왜 여러 개의 정수형이 필요한가?

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

정수형의 범위

- int형

$$-2^{31},\ldots,-2,\ -1,\ 0,\ 1,2,\ldots,\ 2^{31}-1$$

$$-2147483648 \leqq n \leqq +2147483647$$

- short형

$$-2^{15},\ldots,-2,\ -1,\ 0,\ 1,2,\ldots,\ 2^{15}-1$$

$$-32768 \leqq n \leqq +32767$$

- long형
  - 보통 int형과 같음

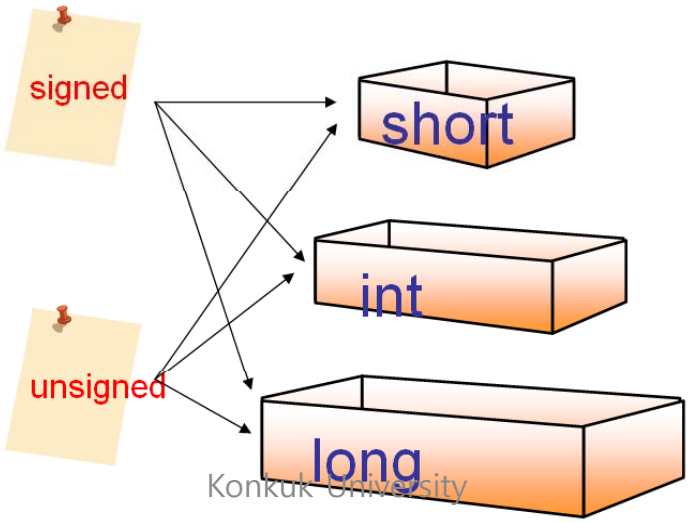
예제

```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.

        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);

        printf("short형의 크기: %d바이트₩n", sizeof(short));
        printf("int형의 크기: %d바이트₩n", sizeof(int));
        printf("long형의 크기: %d바이트₩n", sizeof(long));

        return 0;
}
```

```
short형의 크기: 2바이트
int형의 크기: 4바이트
long형의 크기: 4바이트
```

signed, unsigned 수식자

- unsigned
  - 음수가 아닌 값만을 나타냄을 의미
  - unsigned int $0,\ 1,\ 2,\ldots,\ 2^{32}-1$ $(0 \sim +4294967295)$
- signed
  - 부호를 가지는 값을 나타냄을 의미
  - 흔히 생략



signed short
unsigned short

signed int
unsigned int

signed long
unsigned long

오버플로우

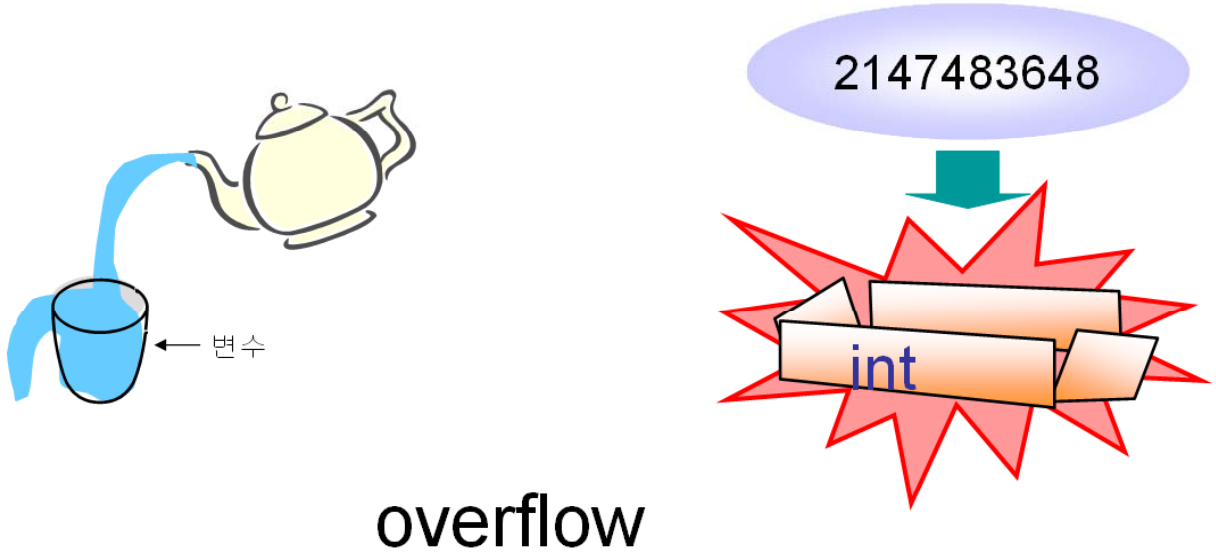
```
#include <stdio.h>
int main(void)
{
        int x;
        unsigned int y;

        x = 2147483647;
        printf("x = %d₩n",x);
        printf("x+1 = %d₩n",x+1);
        printf("x+2 = %d₩n",x+2);
        printf("x+3 = %d₩n",x+3);


        y = 4294967295;
        printf("y = %u₩n",y);      // unsigned를 출력하는 형식 지정자는 %u
        printf("y+1 = %u₩n",y+1);
        printf("y+2 = %u₩n",y+2);
        printf("y+3 = %u₩n",y+3);
}
```

```
x = 2147483647
x+1 = -2147483648
x+2 = -2147483647
x+3 = -2147483646
y = 4294967295
y+1 = 0
y+2 = 1
y+3 = 2
```

오버플로우 발생!!

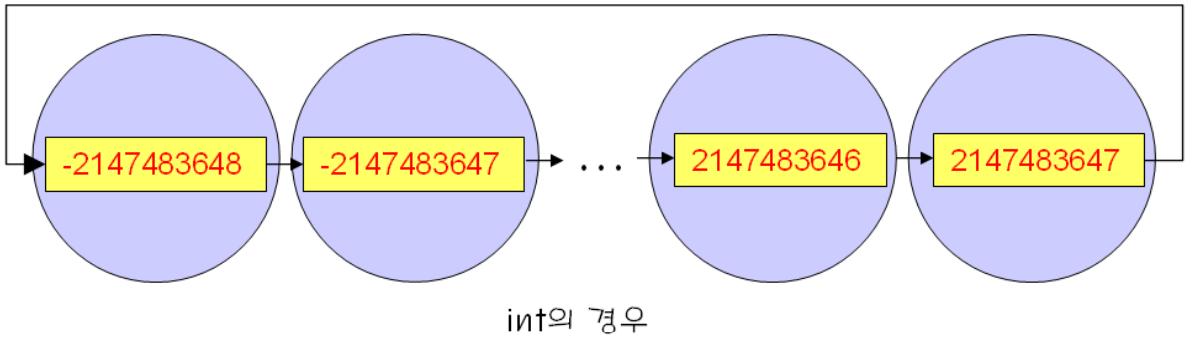
오버플로우

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

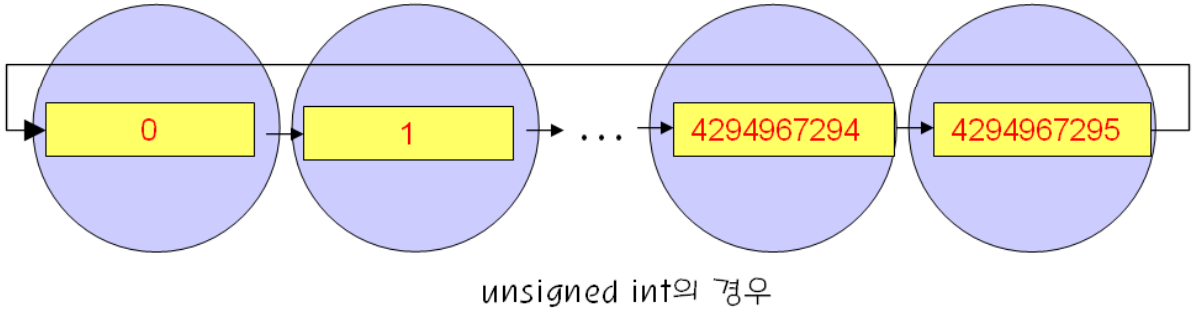


오버플로우

- 규칙성이 있다.
  - 수도 계량기나 주행거리계와 비슷하게 동작



int의 경우



unsigned int의 경우

정수 상수

- 숫자를 적으면 컴파일러가 가장 작은 자료형을 자동으로 선택
- 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

- 10진법이외의 진법으로도 표기 가능

```
int x = 10;   // 10은 10진수이고 int형이고 값은 십진수로 10이다.
int y = 010;  // 010은 8진수이고 int형이고 값은 십진수로 8이다.
int z = 0x10; // 010은 16진수이고 int형이고 값은 십진수로 16이
    다.
```

예제

```c
/* 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; // 010은 16진수이고 int형이고 값은 십진수로 16이다.

        printf("sizeof(10L) = %d\n", sizeof(10L));
        printf("x = %d y = %d z = %d\n", x, y, z);
        printf("x = %d x = %#o x = %#x\n", x, x, x);

        return 0;
}
```

```
sizeof(10L) = 4
x = 10 y = 8 z = 16
x = 10 x = 012 x = 0xa
```

기호 상수

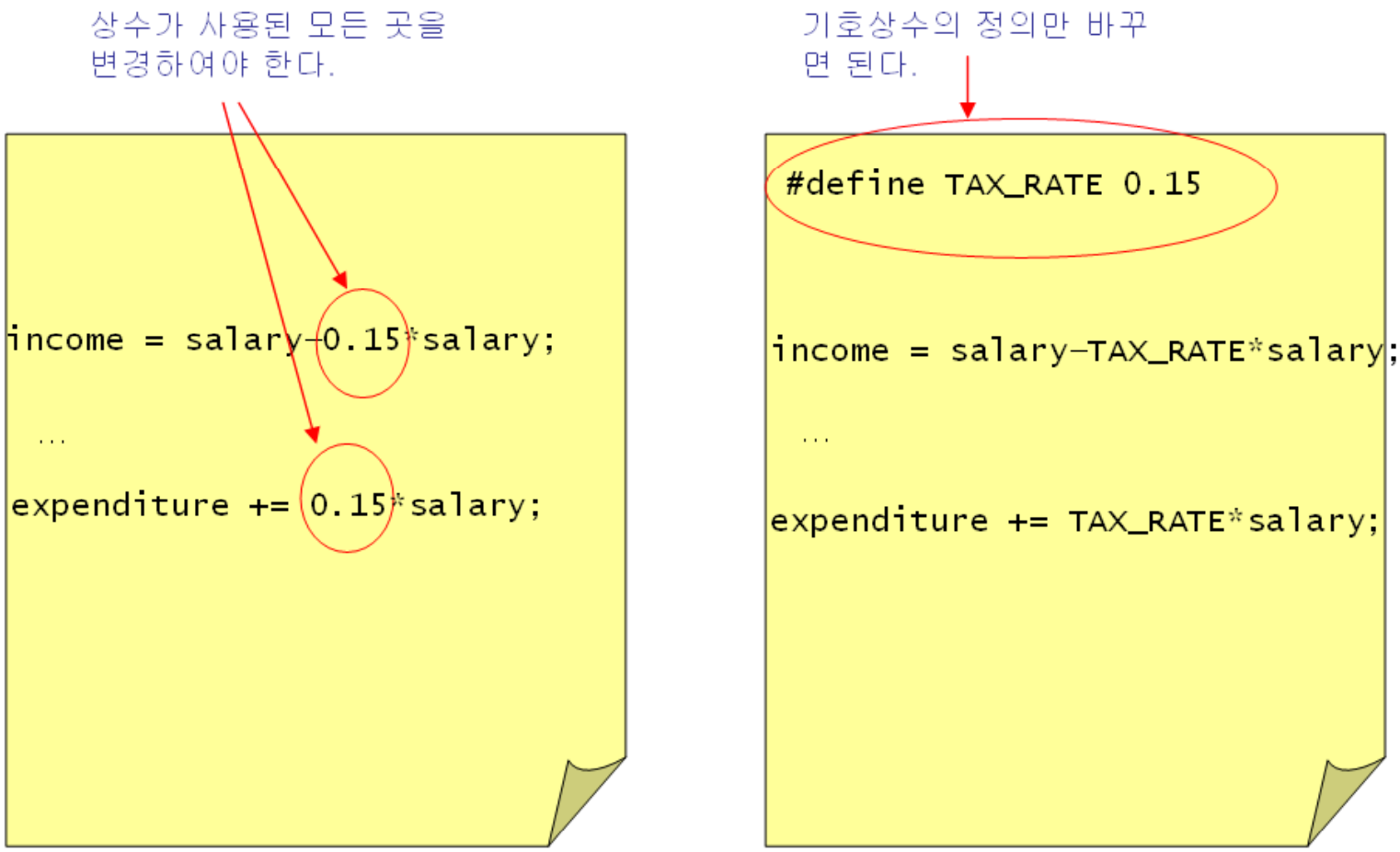
- 기호 상수(symbolic constant): 기호를 이용하여 상수를 표현한 것

- (예)
  - area = 3.141592 * radius * radius;
  - area = PI * radius * radius;

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

- 기호 상수의 장점
  - 가독성이 높아진다.
  - 값을 쉽게 변경할 수 있다.

기호 상수의 장점


상수가 사용된 모든 곳을
변경하여야 한다.
income = salary-0.15*salary;
...
expenditure += 0.15*salary;
기호상수의 정의만 바꾸
면 된다.
#define TAX_RATE 0.15
income = salary-TAX_RATE*salary;
...
expenditure += TAX_RATE*salary;

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

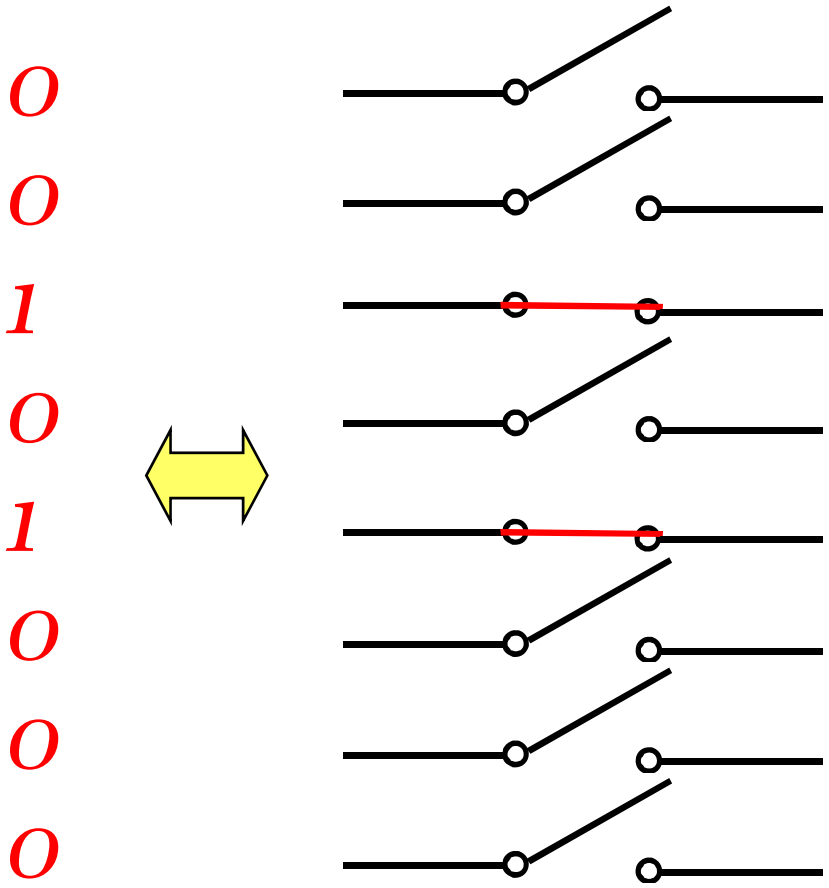
② const 키워드 이용

```
/* 기호상수 프로그램*/
#include <stdio.h>


int main(void)
{
      const double TAX_RATE = 0.15;     // 기호 상수 선언
       double income, salary;           // 변수 선언


       printf("월급을 입력하시요:");       // 입력 안내문
       scanf("%lf", &salary);           // double형은 %lf로 지정하여야함


       income = salary - TAX_RATE * salary;     // 순수입 계산
       printf("순수입은 %lf입니다.\n", income);   // 순수입 출력


       return 0;
}
```

기호 상수 정의

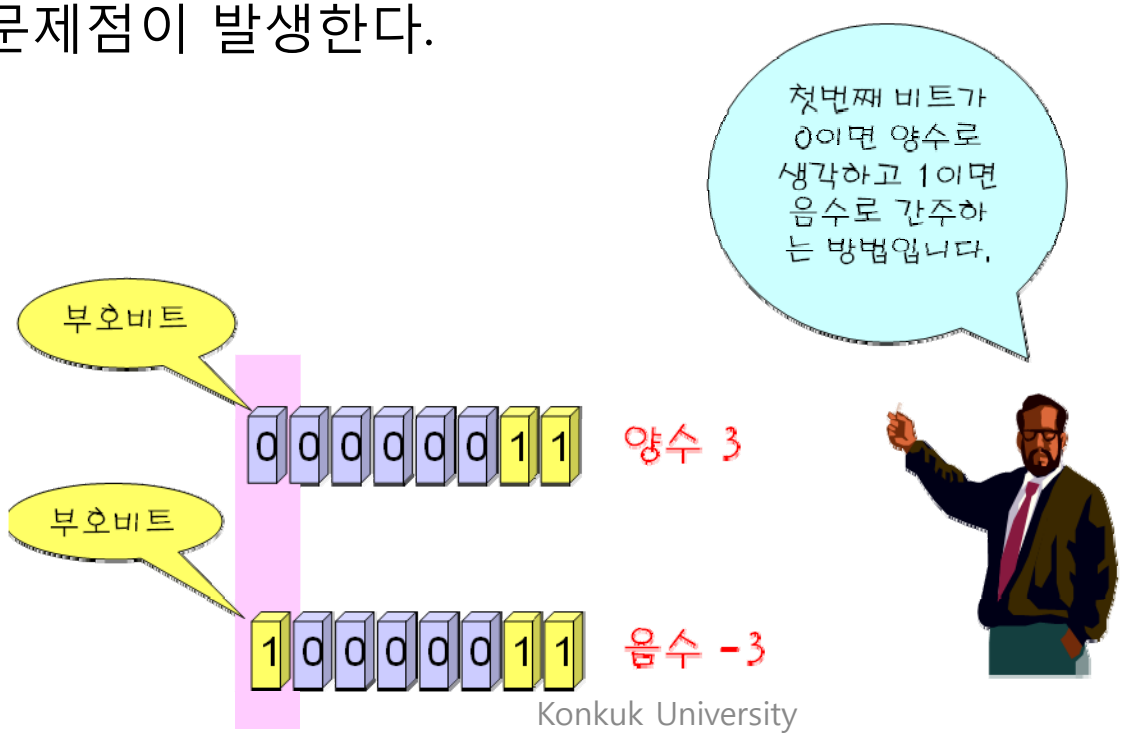
정수 표현 방법

- 컴퓨터에서 정수는 이진수 형태로 표현되고 이진수는 전자 스위치로 표현된다.



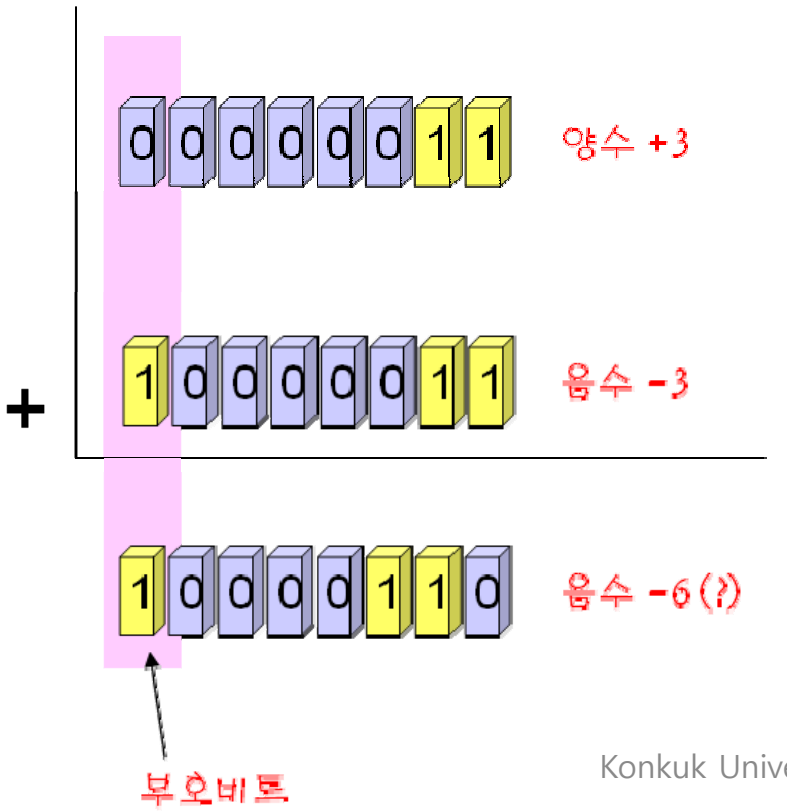
정수 표현 방법

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



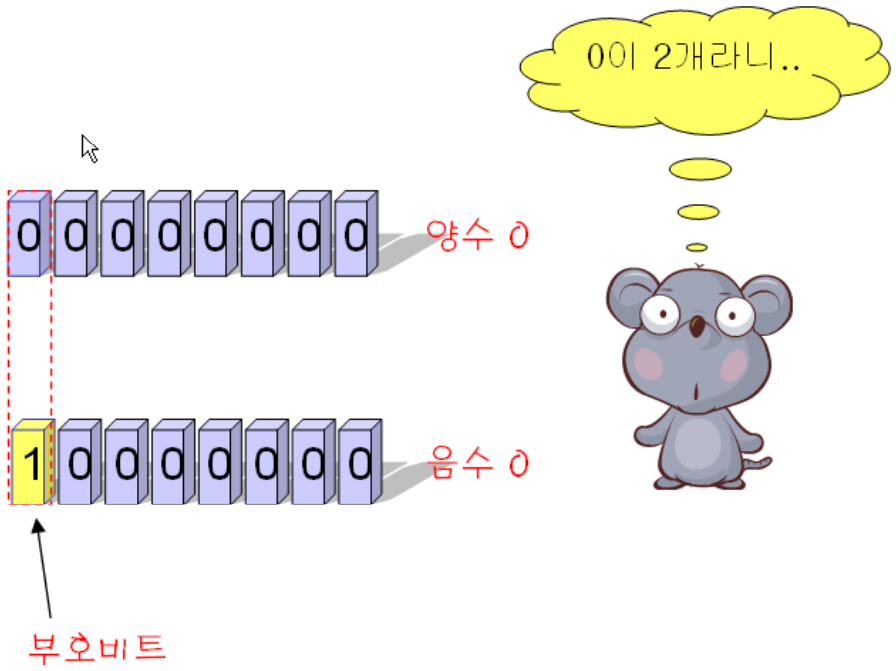
음수를 표현하는 첫번째 방법

- 첫번째 방법은 맨 처음 비트를 부호 비트로 간주하는 방법입니다.
- 양수와 음수의 덧셈 연산을 하였을 경우, 결과가 부정확하다.
  - (예) +3 + (-3)



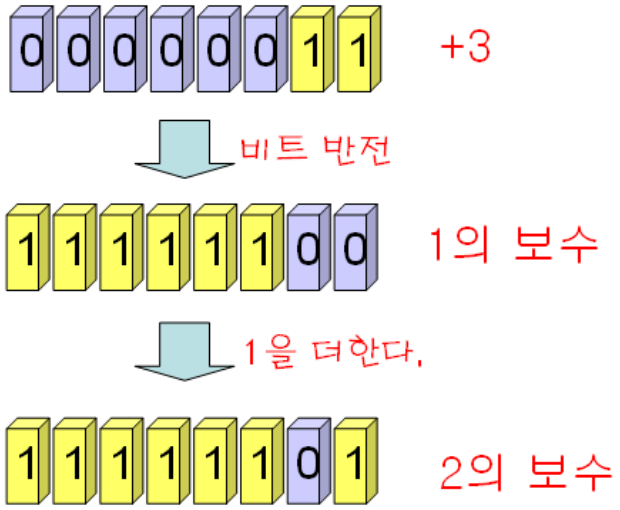
음수를 표현하는 첫번째 방법(cont.)

- 양수 0과 음수 0이 존재



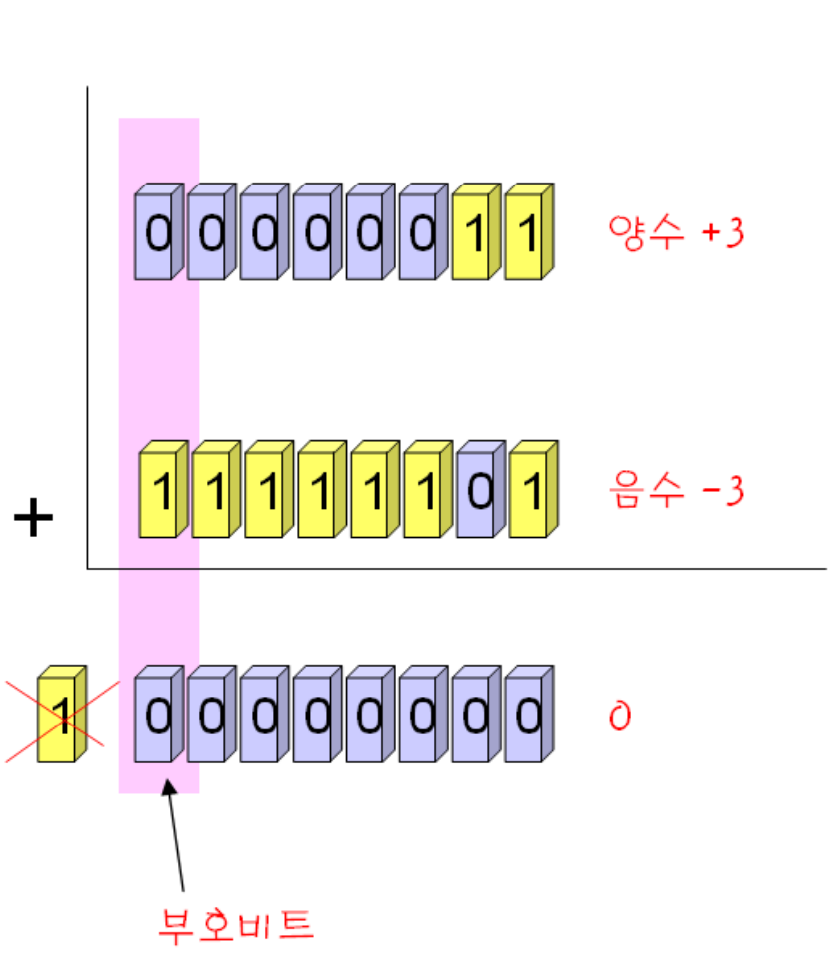
음수를 표현하는 두번째 방법

- 2의 보수로 음수를 표현한다.
- 현재 사용되는 표준적인 음수 표현 방법
- 2의 보수를 만드는 방법



부호 비트									
0	1	1	1	1	1	1	1	=	127
0	1	1	1	1	1	1	0	=	126
		...				...			...
0	0	0	0	0	0	1	1	=	3
0	0	0	0	0	0	1	0	=	2
0	0	0	0	0	0	0	1	=	1
0	0	0	0	0	0	0	0	=	0
1	1	1	1	1	1	1	1	=	−1
1	1	1	1	1	1	1	0	=	−2
1	1	1	1	1	1	0	1	=	−3
		...				...			...
1	0	0	0	0	0	0	1	=	−127
1	0	0	0	0	0	0	0	=	−128

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
```

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{|}~
```

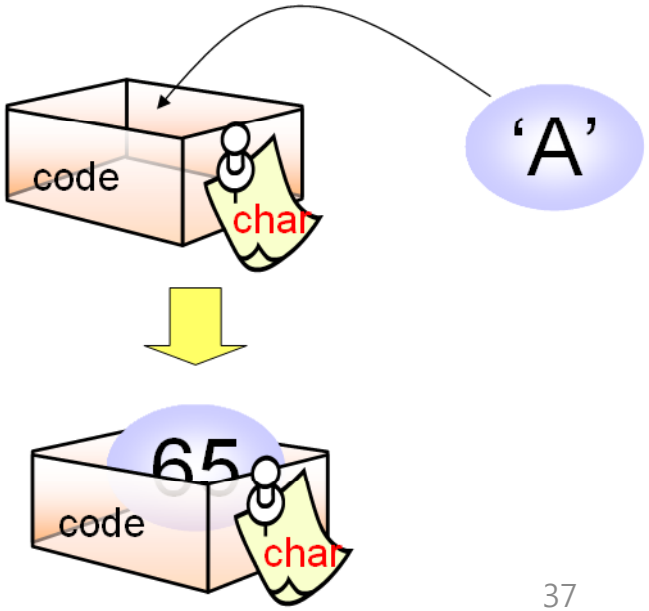
문자 변수

- char형의 변수가 문자 저장

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

- char형의 변수에 문자를 저장하려면 아스키 코드 값을 대입

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



예제

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

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

        printf("문자 상수 초기화 = %c\n", code1);
        printf("아스키 코드 초기화 = %c\n", code2);
}
```

```
문자 상수 초기화 = A
아스키 코드 초기화 = A
```

(Q) 1과 '1'의 차이점은?

(A) 1은 정수 상수이고 '1'은 문자 상수이다.

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 사용

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

이스케이프 시퀀스

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 강제적으로 다음 페이지로 넘길 때 사용된다.
캐리지 리턴(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

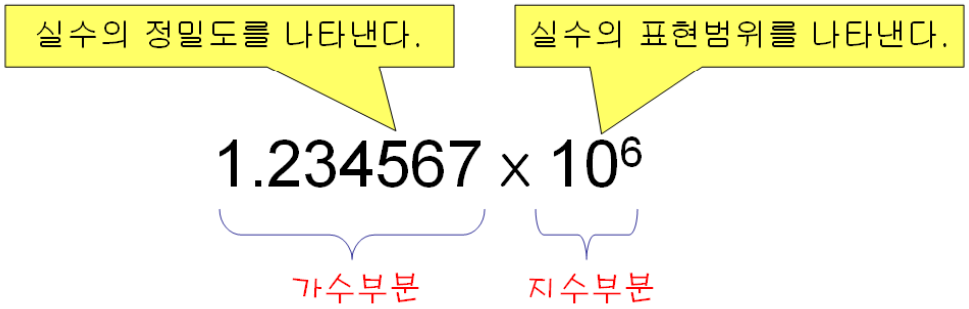
예제

```c
/* 이스케이프 시퀀스 */
#include <stdio.h>

int main(void)
{
        printf("이스케이프 시퀀스는 \\와 의미를 나타내는 글자를 붙여서 기술\n");
        printf("'\\a'는 경고를 나타내는 제어문자이다. \n");
        printf("'\\007'로도 표현이 가능하다. \n");
        printf("경고를 출력해 보자'\\007'을 출력한다\007 \n");

}
```

```
이스케이프 시퀀스는 ₩와 의미를 나타내는 글자를 붙여서 기술
'₩a'는 경고를 나타내는 제어 문자이다.
'₩007'로도 표현이 가능하다.
경고를 출력해보자 '₩007'을 출력한다
```

부동소수점형

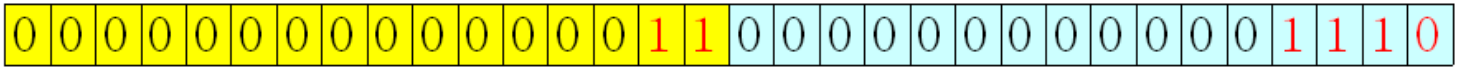
- 컴퓨터에서 실수는 부동소수점형으로 표현
  - 소수점이 떠서 움직인다는 의미
  - 과학자들이 많이 사용하는 과학적 표기법과 유사



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

실수를 표현하는 방법

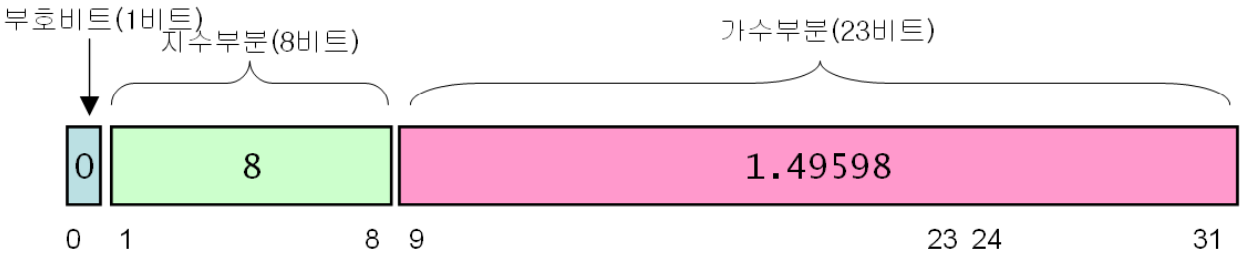
- #1 고정 소수점 방식
  - 정수 부분을 위하여 일정 비트를 할당하고 소수 부분을 위하여 일정 비트를 할당
  - 전체가 32비트이면 정수 부분 16비트, 소수 부분 16비트 할당
  - (예) 3.14



  - 과학과 공학에서 필요한 아주 큰 수를 표현할 수 없다

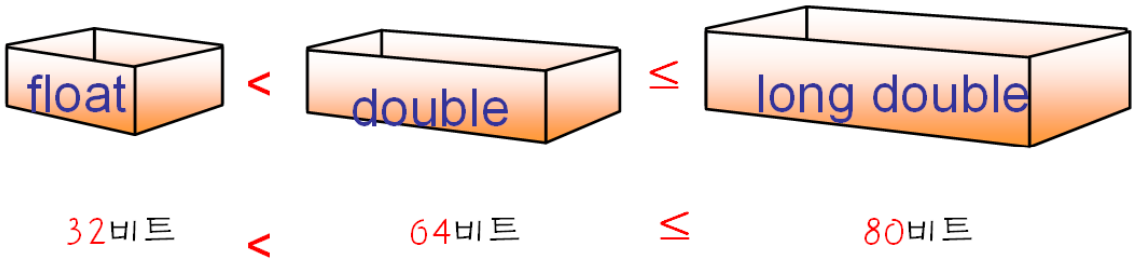
실수를 표현하는 방법

- #2 부동 소수점 방식



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

부동 소수점 형



자료형	명칭	크기	범위
float	단일정밀도(single-precision) 부동소수점	32비트	$\pm 1.17549 \times 10^{-38} \sim \pm 3.40282 \times 10^{+38}$
double	두배정밀도(double-precision) 부동소수점	64비트	$\pm 2.22507 \times 10^{-308} \sim \pm 1.79769 \times 10^{+308}$
long double	두배확장정밀도 (double-extension-precision) 부동소수점	64비트 또는 80비트	$\pm 2.22507 \times 10^{-308} \sim \pm 1.79769 \times 10^{+308}$

예제

```c
/* 부동 소수점 자료형의 크기 계산*/
#include <stdio.h>

int main(void)
{
        float x = 1.234567890123456789;
        double y = 1.234567890123456789;

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

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

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

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 //

부동 소수점 오버플로우

```
#include <stdio.h>

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

숫자가 커서 오버플로우 발생

```
C:₩CPROGRAM₩test₩test.c(5) : warning C4056: overflow in floating-point constant arithmetic
```

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
```

부동소수점형 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;
}
```

5.0은 부동 소수점수가 저장할 수 있는 한계때문에 저장되지 않는다.

```
0.000000
```

Q & A