데이터형과 변수

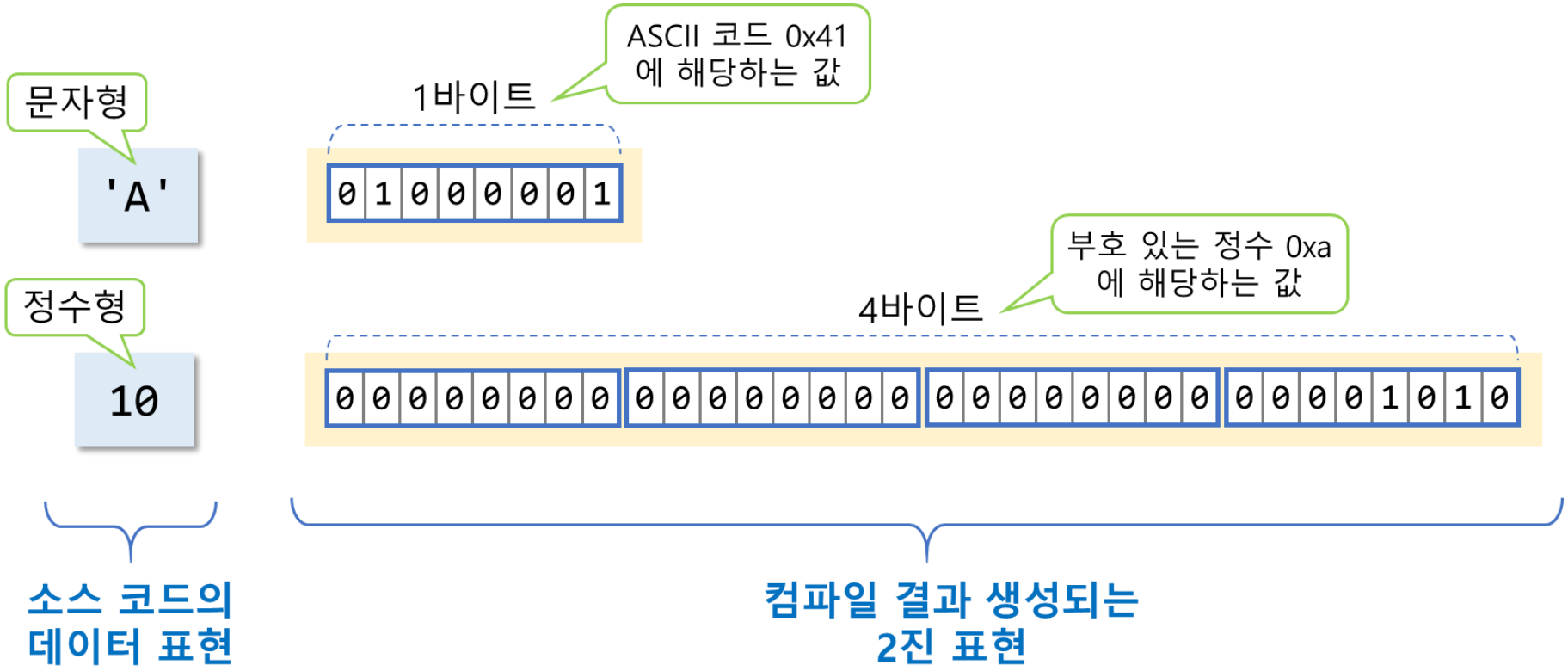
2020학년도 1학기

강원도립대학교 ICT드론과

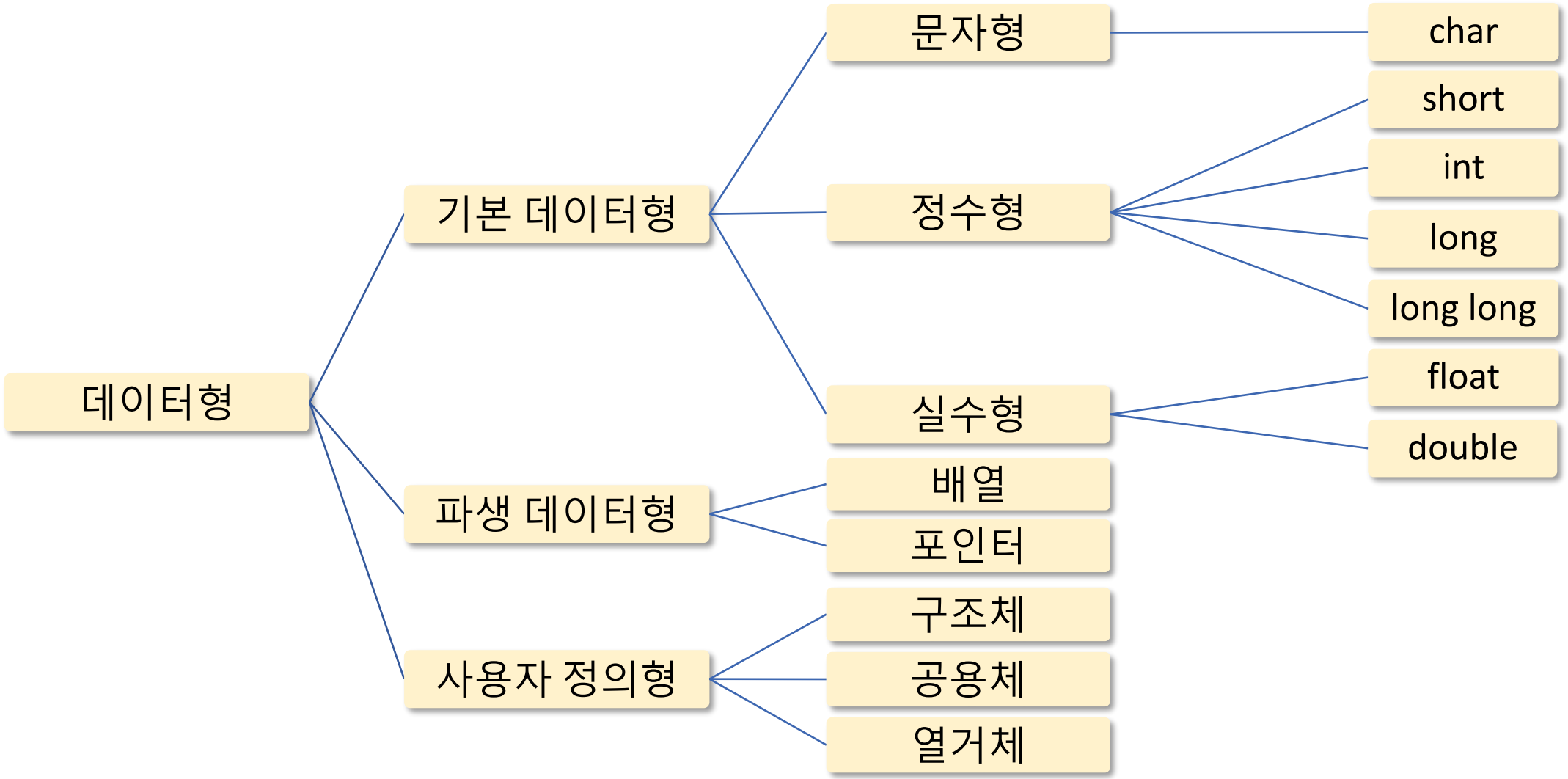
목차

데이터의 2진 표현

- 컴퓨터 시스템에서 2진 데이터로 값을 표현하고 저장하는 방식
- 컴파일러는 데이터형에 따라 값을 저장하는데 필요한 **메모리의 크기와 2진 표현을 결정**한다.



기본 데이터형


데이터형
기본 데이터형
파생 데이터형
사용자 정의형
문자형
정수형
실수형
배열
포인터
구조체
공용체
열거체
char
short
int
long
long long
float
double

sizeof 연산자

- 데이터형의 크기는 플랫폼에 따라 다르다.
- 데이터형이나 값의 바이트 크기를 구하려면 sizeof 연산자를 이용한다.

형식	sizeof(데이터형) sizeof(값) sizeof 값
사용예	sizeof(char) sizeof(int) sizeof(num) sizeof(num + 1) sizeof 3.141592

- 소스 코드에서 데이터형이나 값의 크기가 필요할 때는 sizeof 연산자로 구한 크기를 사용하는 것이 좋다.

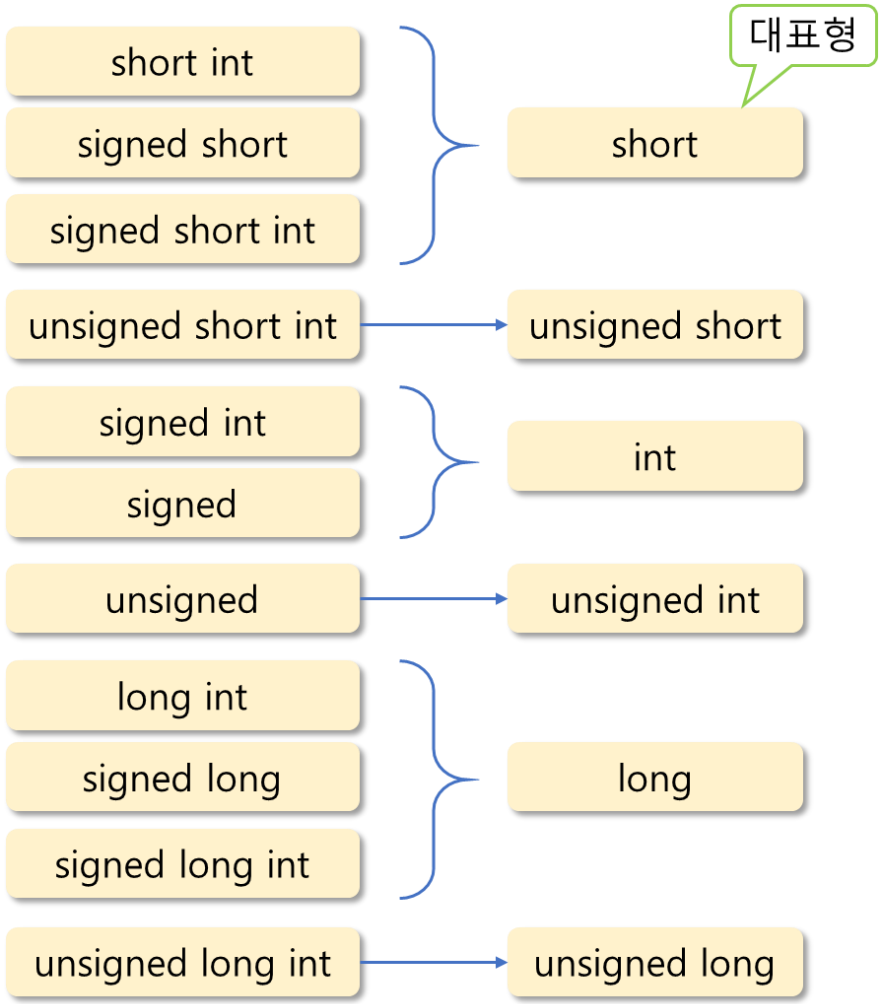
```
size1 = 4 * 10;                 // 의미가 불분명한 코드
size2 = sizeof(float) * 10;     // 의미가 명확한 코드
```

예제 3-1

정수형

- 프로그래머가 용도에 따라 적절한 데이터 형을 선택할 수 있도록 다양한 크기의 정수형을 제공한다.
  - sizeof(short) ≤ sizeof(int) ≤ sizeof(long) ≤ sizeof(long long)

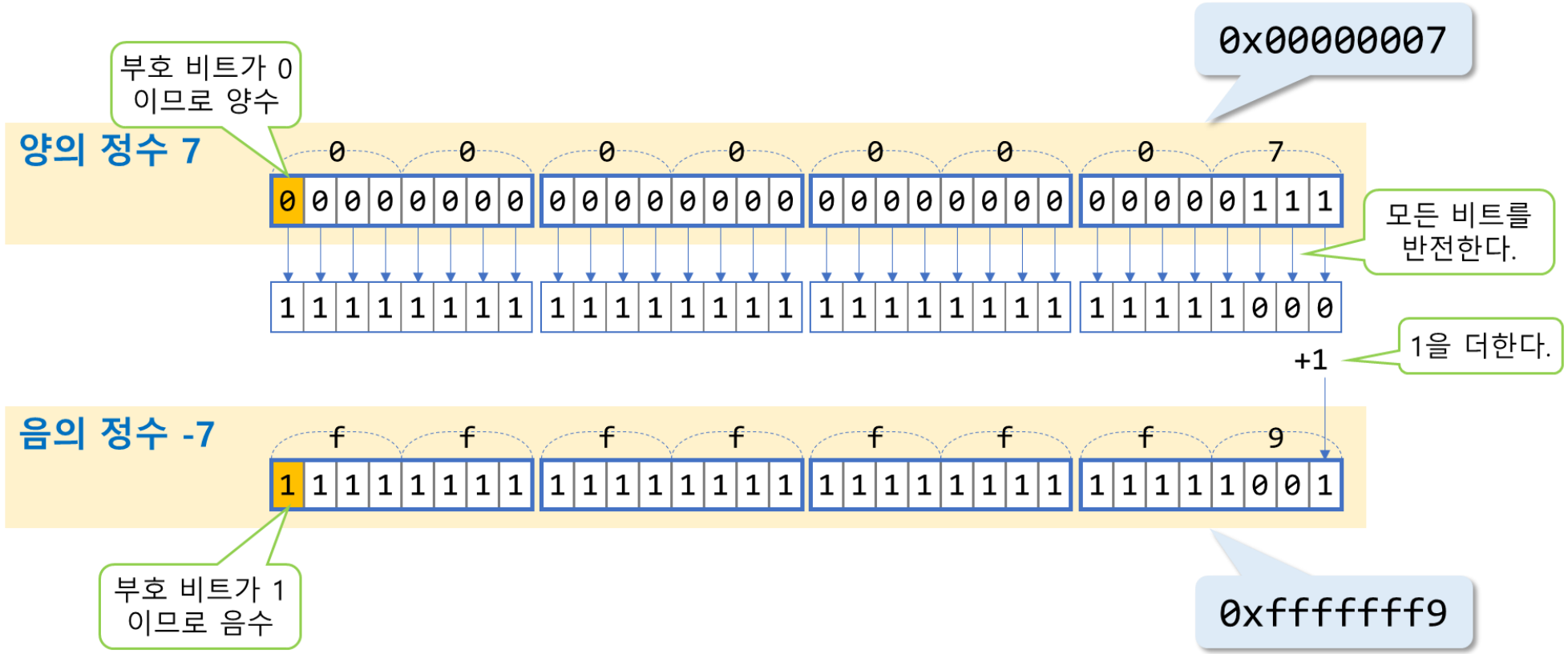
- 부호 있는 정수형과 부호 없는 정수형
  - signed는 생략 가능
  - unsigned : 부호 없는 정수형



정수의 2진 표현

- 부호 있는 정수형은 최상위 비트를 **부호 비트**로 사용
- 음수를 나타내기 위해 **2의 보수**를 사용

예제 3-2

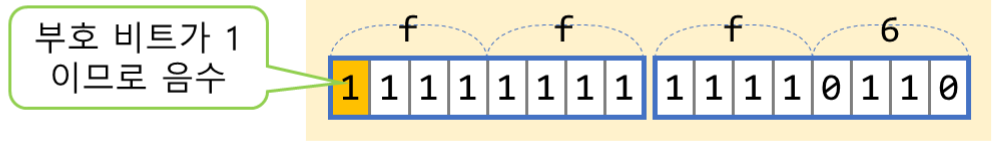


부호 없는 정수형

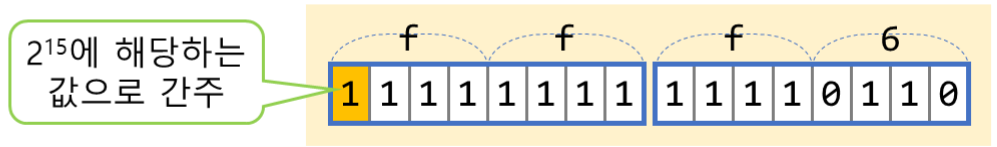
- 최상위 비트를 값을 저장하는 용도로 사용

예제 3-3

```
short a = -10;
```



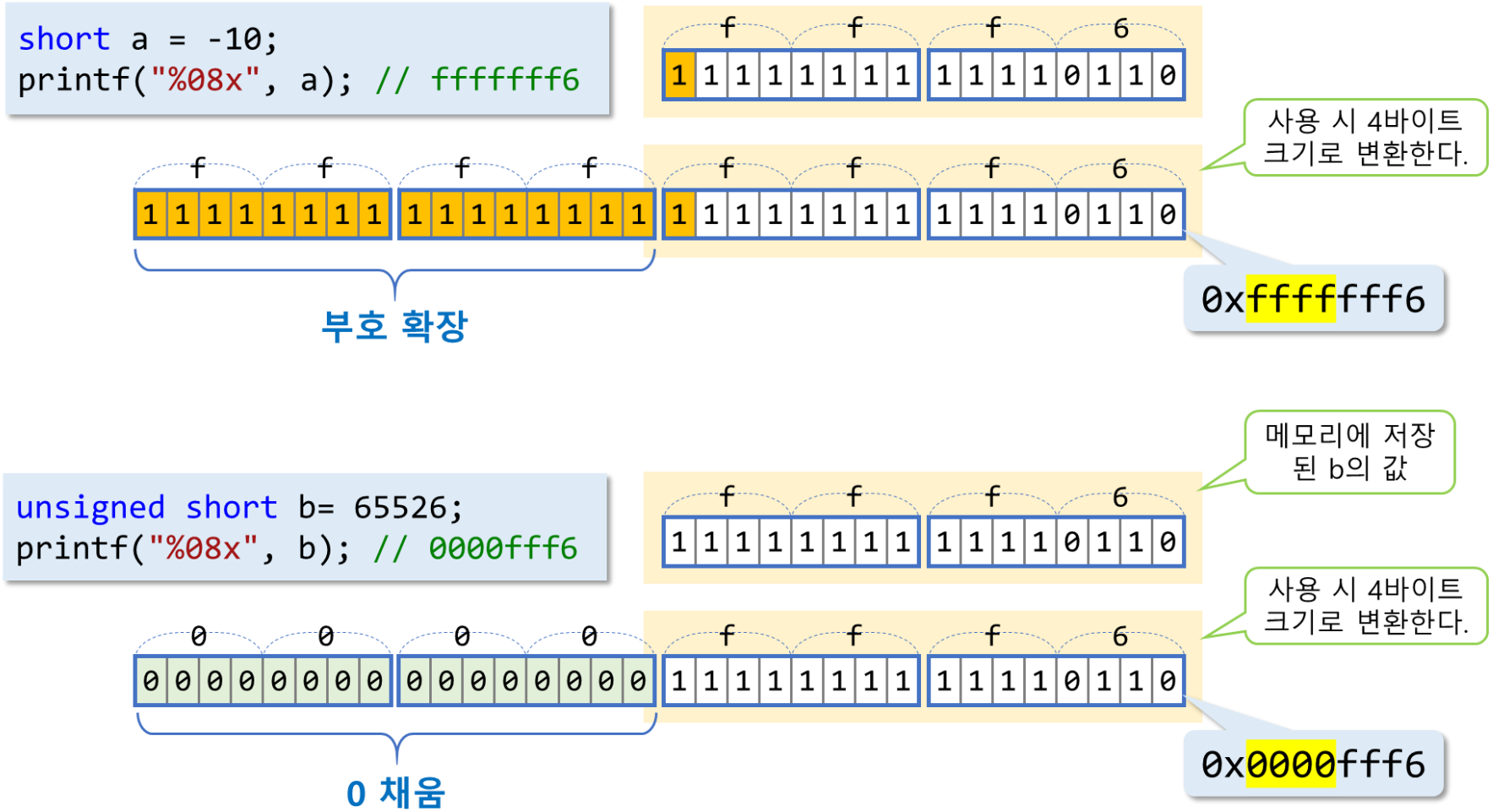

```
unsigned short b = 65526;
```



2진 표현은 같지만
의미가 다르다.

정수의 승격

- char, short형 변수는 사용 시 int형으로 변환된다.



정수형으로 사용되는 char형

- char형은 1바이트 크기의 정수형으로 사용
  - char형의 범위는 -128~127이므로 작은 크기의 정수를 저장할 때 유용하다.
  - char형도 정수형이므로 덧셈, 뺄셈과 같은 연산을 할 수 있다.

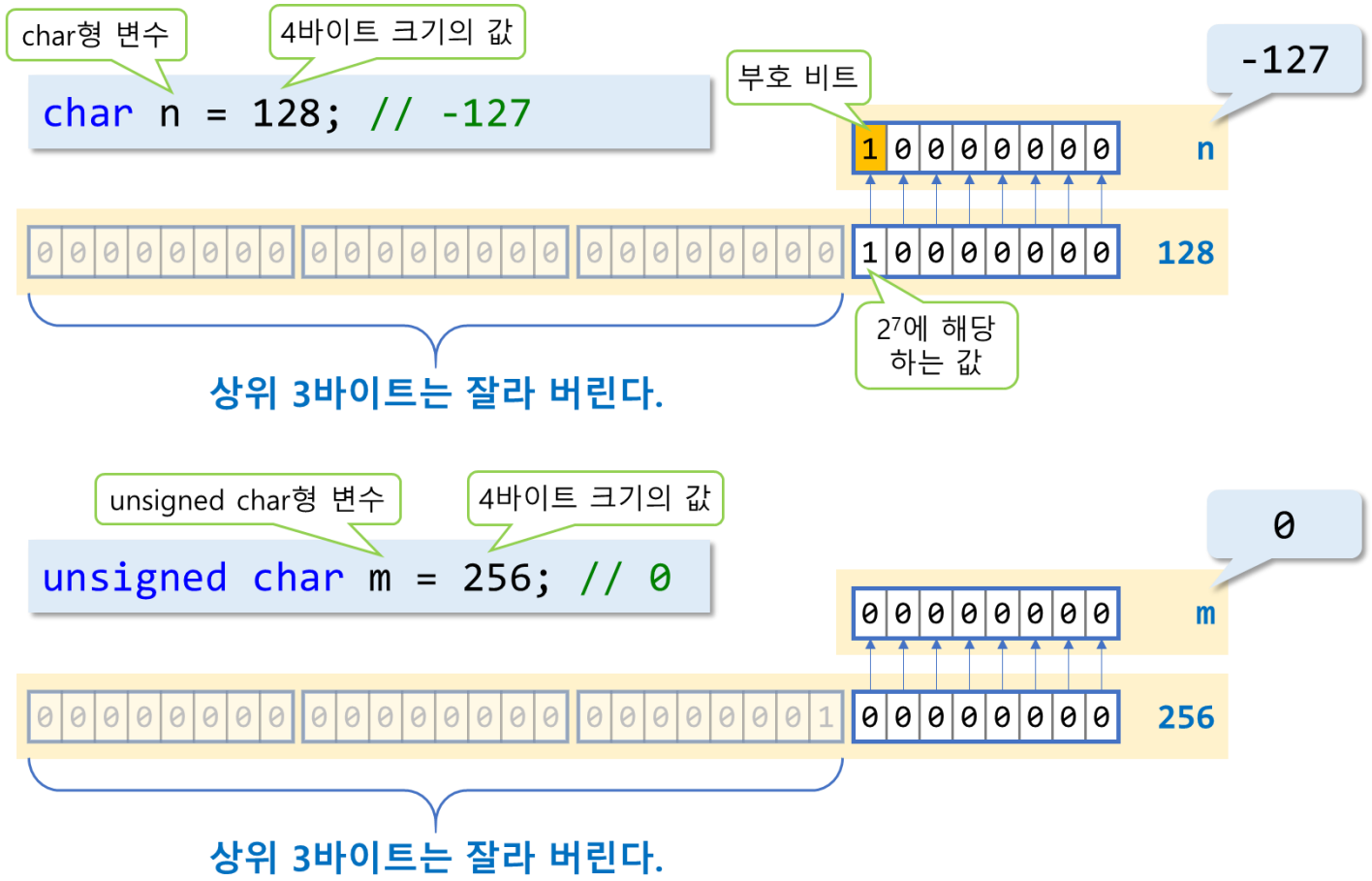
```
char n = 127;    // 정수형으로 사용되는 char형

printf("n   = %d\n", n);          // 정수처럼 10진수로 출력할 수 있다.
printf("n+1 = %d\n", n + 1);      // 정수처럼 덧셈 연산을 할 수 있다.
```

- unsigned char형은 1바이트 크기의 2진 데이터를 저장할 때 주로 사용

```
unsigned char control_flag = 0x47;      // 1바이트 크기의 컨트롤 플래그 0100 0111
unsigned char image[256];               // 256바이트 크기의 이미지
```

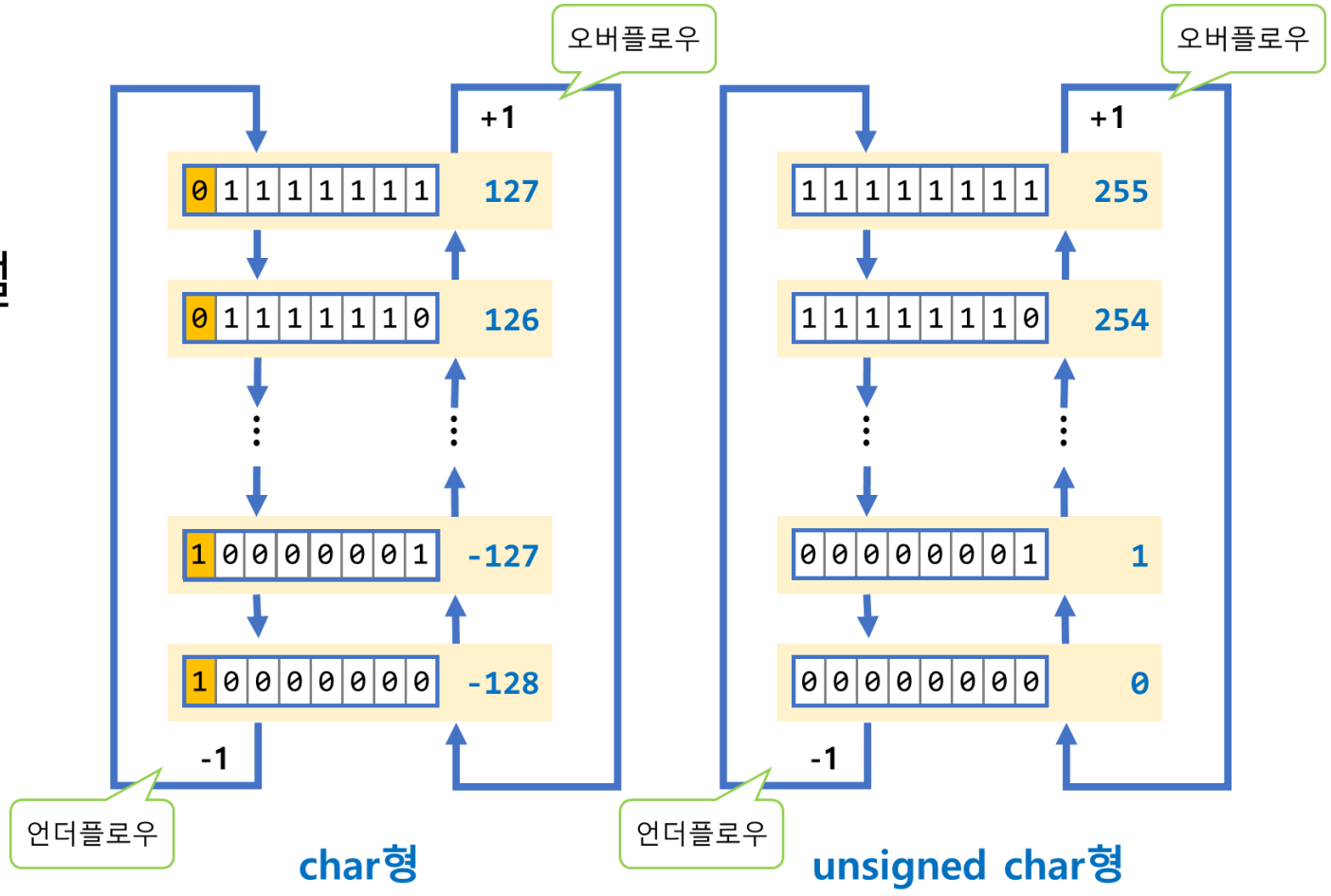
정수형의 유효 범위

- 정수형 변수에 유효 범위 밖의 값을 저장하면, 정수형의 크기에 맞춰 값의 나머지 부분을 잘라 버리고 유효 범위 내의 값만 저장된다.



오버플로우와 언더플로우

- 오버플로우
  - 정수형의 최대값보다 큰 값을 저장할 때 값이 넘쳐흘러서 유효 범위 내의 값으로 설정되는 것
- 언더플로우
  - 정수형의 최소값보다 작은 값을 저장할 때도 유효 범위 내의 값으로 설정되는 것




```
unsigned char red = 300;    // 오버플로우가 발생하므로 red에 실제로는 44가 저장된다.
```

예제 3-4

문자형

- 문자의 2진 표현 : ASCII 코드
  - 제어 문자 : 0~31, 127
  - 출력 가능한 문자 : 32~126
- 특수 문자
  - ' '안에 역슬래시(\)와 함께 정해진 문자를 이용해서 나타낸다.
  - '\' 다음에 8진수로 적어주거나 '\x' 다음에 16진수로 적어준다.

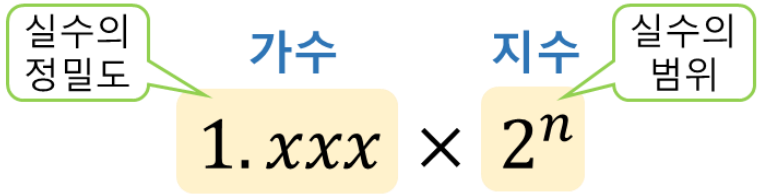
10진수	8진수	16진수	특수 문자	의미
0	000	00	'\0'	널 문자(null)
7	007	07	'\a'	경고음(bell)
8	010	08	'\b'	백스페이스(backspace)
9	011	09	'\t'	수평 탭(horizontal tab)
10	012	0A	'\n'	줄 바꿈(newline)
11	013	0B	'\v'	수직 탭(vertical tab)
12	014	0C	'\f'	폼 피드(form feed)
13	015	0D	'\r'	캐리지 리턴(carriage return)
34	042	22	'\"'	큰따옴표
39	047	27	'\''	작은따옴표
92	134	5C	'\\'	역슬래시(back slash)

```
char newline1 = '\012';     // newline1에 '\n'을 저장한다.
char newline2 = '\xa';      // newline2에 '\n'을 저장한다.
```

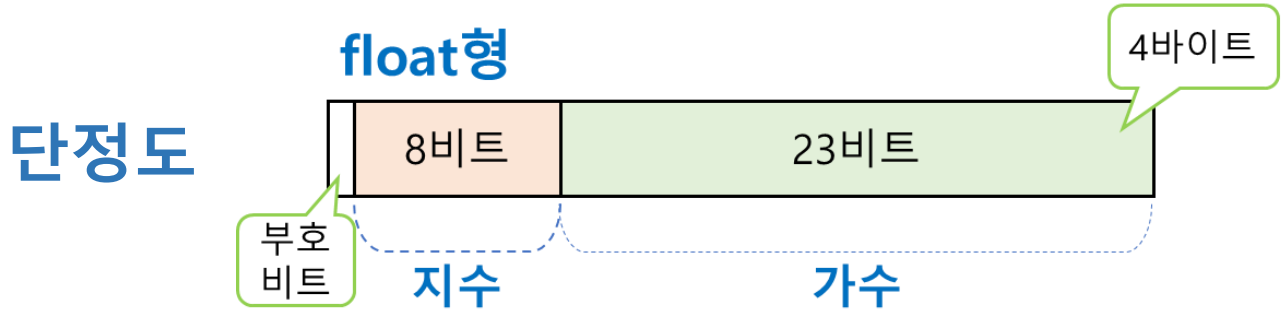
예제 3-5, 3-6

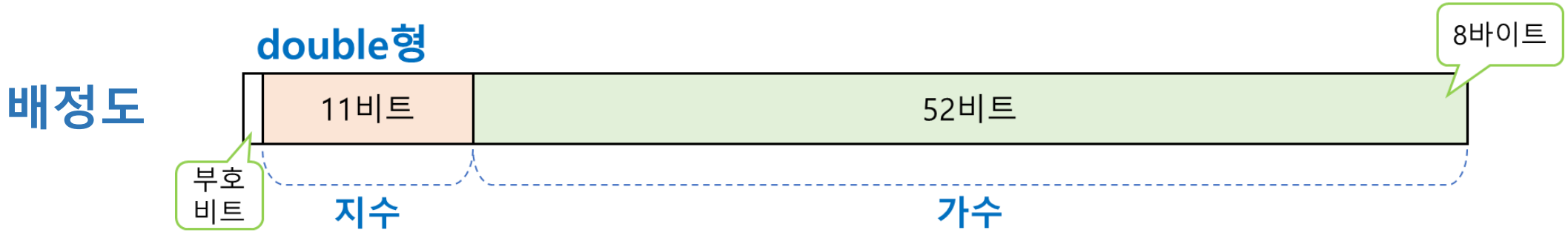
실수형

- 부동소수점 방식



예제 3-7





실수의 유효 범위

- 실수형의 오버플로우
  - 최대값보다 큰 값을 저장하려고 하면 무한대를 의미하는 INF로 설정
- 실수형의 언더플로우
  - 최소값보다 작은 값을 저장하려고 하면, 가수 부분을 줄이고 지수 부분을 늘려서 실수를 표현하거나, 만일 그것이 불가능해지면 0으로 만들어 버린다.
- 실수형의 크기와 유효 범위

예제 3-8

데이터형		크기	유효 범위
실수형	float	4바이트	$\pm 1.17549 \times 10^{-38} \sim \pm 3.40282 \times 10^{38}$
	double	8바이트	$\pm 2.22507 \times 10^{-308} \sim \pm 1.79769 \times 10^{308}$
	long double	8바이트	$\pm 2.22507 \times 10^{-308} \sim \pm 1.79769 \times 10^{308}$

변수의 필요성

- 변수를 이용하면 '어떤 값이 될지 모르는 값을 처리하는 프로그램'을 작성할 수 있다.

변수를 사용하지 않는 경우

InfinityWarPlayer.exe

```
int main(void)
{
  "InfinitWar.mp4" 파일을 연다.
  동영상을 재생한다.
  파일을 닫는다.

  return 0;
}
```

BlackPantherPlayer.exe

```
int main(void)
{
  "BlackPanther.mp4" 파일을 연다.
  동영상을 재생한다.
  파일을 닫는다.

  return 0;
}
```

파일 이름이 달라지면 프로그램을 다시 만든다.

⋮

같은 일을 하는 프로그램을 매번 다시 만들어야 하므로 비효율적이다.

변수를 사용하는 경우

MoviePlayer.exe

```
int main(void)
{
  char filename[80]; // 변수 사용
  filename에 동영상 파일 이름을 입력받는다.
  filename 파일을 연다.
  동영상을 재생한다.
  파일을 닫는다.

  return 0;
}
```

입력된 파일 이름을 저장하는 변수를 이용한다.

한 프로그램으로 여러 가지 동영상을 재생할 수 있다.

변수의 선언과 사용

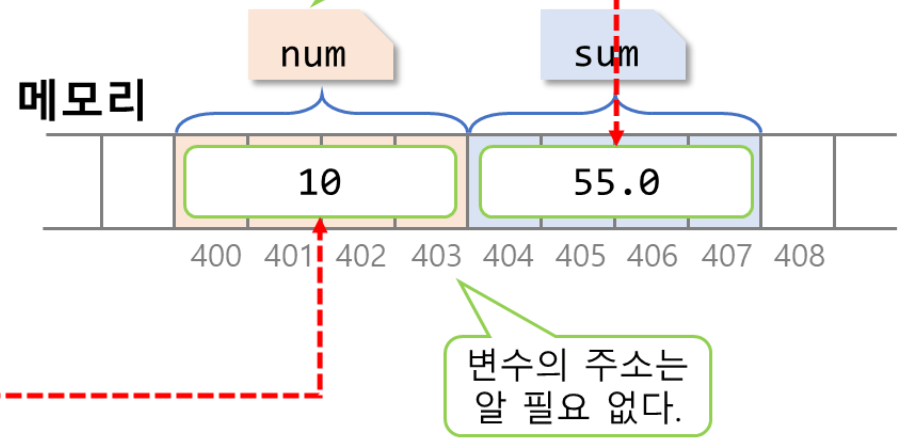
변수가 사용할 메모리 할당

연산의 결과를 변수에 저장

입력받은 값을 변수에 저장

```
int main(void)
{
   int num;
   float sum;

   scanf("%d", &num);
   sum = num * (num + 1) * 0.5
   printf("sum = %.1f\n", sum);

   return 0;
}
```

변수의 이름을 사용해서 접근



변수의 주소는 알 필요 없다.

[실행 결과]

```
10
sum = 55.0
```

변수의 선언

- 변수를 선언하려면 변수의 데이터형과 이름이 필요하다.

형식 데이터형 변수명;
데이터형 변수명1, 변수명2, … ;

사용예

```
float discount_rate;
int width, height;
unsigned char red, green, blue;
```

- **식별자를 만드는 규칙**
  - 반드시 영문자, 숫자, 밑줄 기호(_)만을 사용해야 한다.
  - 첫 글자는 반드시 영문자 또는 밑줄 기호(_)로 시작해야 한다. 식별자는 숫자로 시작해서는 안 된다.
  - 밑줄 기호(_)를 제외한 다른 기호를 사용할 수 없다.
  - 대소문자를 구분해서 만들어야 한다. red, Red, RED은 모두 다른 이름이다.
  - C 언어의 키워드는 식별자로 사용할 수 없다.

변수의 선언 예

- 올바른 변수 선언의 예

```
int salary2018;      // 변수명의 첫 글자 외에는 숫자를 사용할 수 있다.
double _rate;        // 변수명은 _로 시작할 수 있다.
int width, height;   // 여러 개의 변수를 선언할 때는 ,를 사용한다.
long discount_rate;  // 여러 단어를 연결할 때는 _를 사용한다.
int discountRate;    // 연결되는 단어의 첫 글자를 대문자로 지정한다.
```

- 잘못된 변수 선언의 예

```
long text-color;     // 변수명에 - 기호를 사용할 수 없다.
int elapsed time;    // 변수명에 빈칸을 포함할 수 없다.
int 2018salary;      // 변수명은 숫자로 시작할 수 없다.
char case;           // C 키워드는 변수명으로 사용할 수 없다.
```

변수 선언문의 위치

ANSI C

```
int main(void)
{
   int num;
   float sum;

   scanf("%d", &num);
   sum = num * (num + 1) * 0.5;
   printf("sum = %.1f\n", sum);

   return 0;
}
```

변수 선언문이 함수 시작 부분에 모여 있어야 한다.

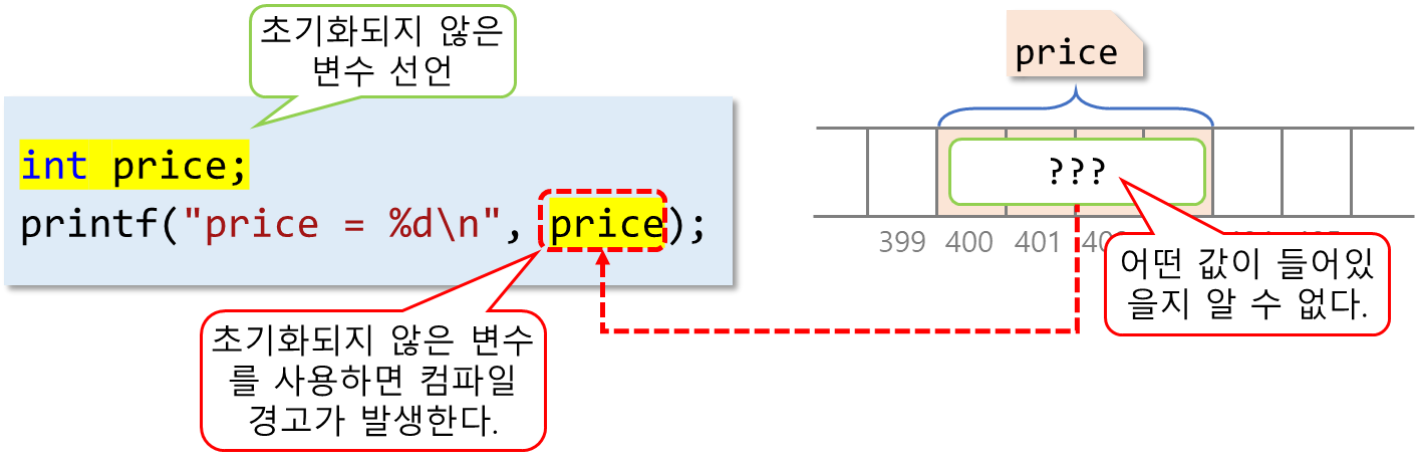
C99

```
int main(void)
{
   int num;
   scanf("%d", &num);

   float sum;
   sum = num * (num + 1) * 0.5;
   printf("sum = %.1f\n", sum);

   return 0;
}
```

변수를 필요한 곳에서 선언할 수 있다.

변수의 초기화 (1/2)

- 선언 시 초기화되지 않은 변수의 값은 쓰레기 값이다.



- 변수의 초기화 방법

형식 데이터형 변수명 = 초기값;
데이터형 변수명1 = 초기값1, 변수명2 = 초기값2, … ;

사용예

```
float discount_rate = 0.1;
char size = 'L';
int width = 100, height = 100;
```

변수의 초기화 (2/2)

- 변수를 초기화할 때 변수의 데이터형과 같은 형의 값으로 초기화해야 한다.

```
int area = 3.14 * 5 * 5;       // 정수형 변수를 실수 값으로 초기화하므로 컴파일 경고 발생
float discount_rate = 0.1;     // 0.1은 double형이므로 컴파일 경고 발생
```

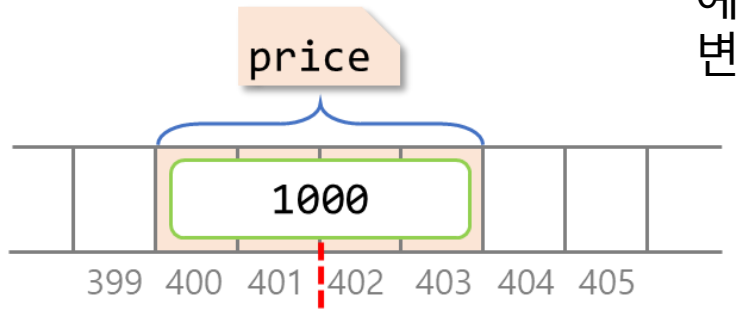
- 변수가 어떤 초기값을 가져야 하는지 알 수 없을 때에는 0으로라도 초기화하는 것이 안전하다.

```
int amount = 0;      // 어떤 값이 될지 아직 알 수 없으면 0으로 초기화한다.
float sum = 0;       // 어떤 값이 될지 아직 알 수 없으면 0으로 초기화한다.
```

변수의 사용

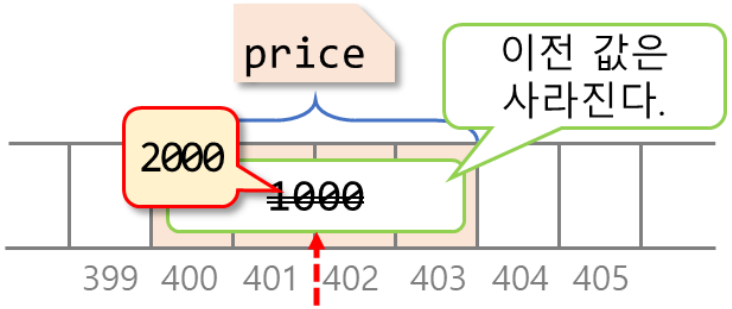
예제 3-9
변수의선언과 사용

```
int price;
price = 1000;
printf("price = %d\n", price);
```



price에 저장된 값인 1000을 읽어온다.

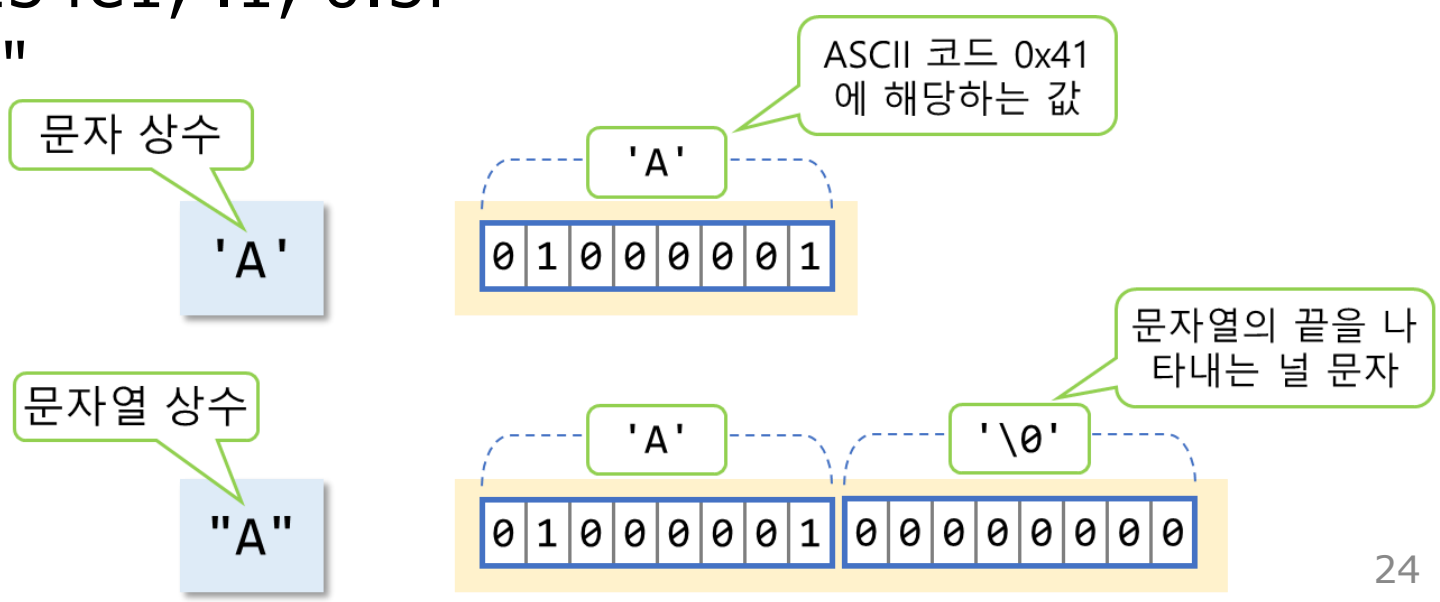
```
int price;
price = 1000;
printf("price = %d\n", price);
price = 2000 ;
```



상수

- 프로그램에서 값이 변경되지 않는 요소
  - 메모리에 저장되지 않고, 한 번만 사용된 다음 없어져 버리는 임시 값
- 리터럴 상수 : 값 자체
  - 문자 상수 : 'A', '\xa'
  - 정수형 상수 : 0x12, 123u, 1234567L
  - 실수형 상수 : 12.34, 1.234e1, .1, 0.5F
  - 문자열 상수 : "hello", "A"

예제 3-10 리터럴상수의크기

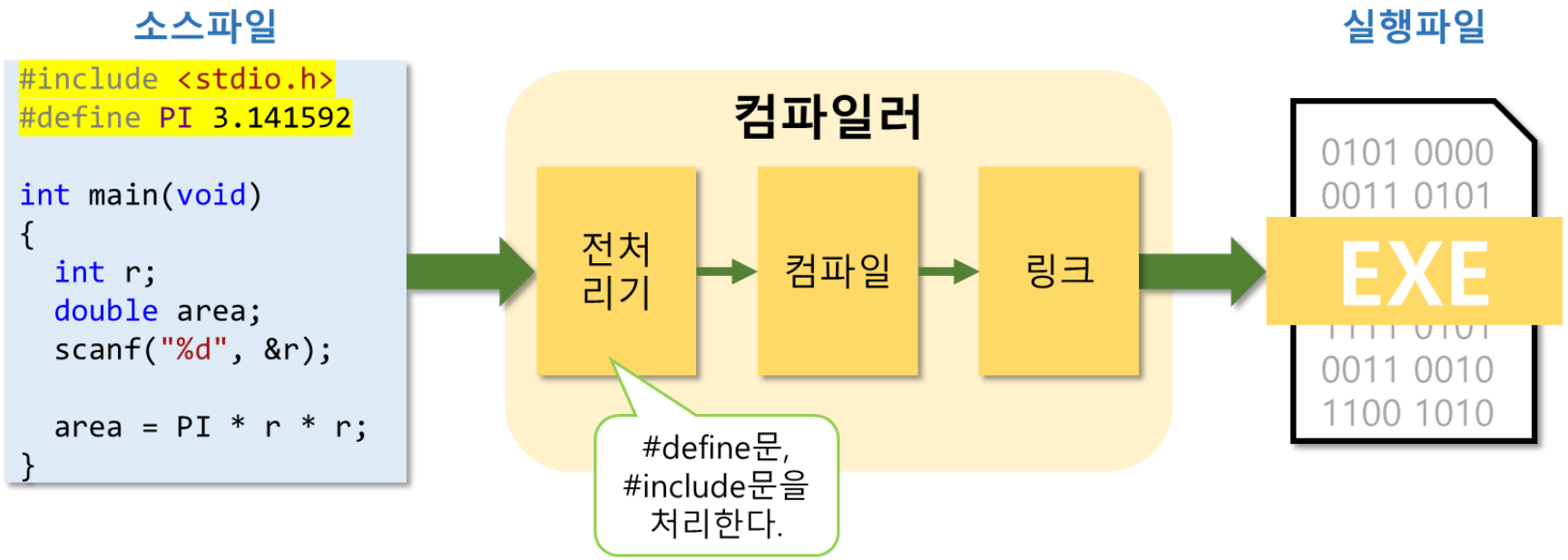


매크로 상수

- **#define**문으로 정의되는 상수

예제 3-11 매크로 상수

  - 전처리기가 처리하는 문장
  - 전처리기는 컴파일러가 소스 파일을 컴파일하기 전에 먼저 수행되며, 프로그래머가 작성한 소스 파일을 컴파일할 수 있도록 변환해서 준비한다.



전처리기의 #define 처리

- 전처리기는 매크로 상수를 특정 값으로 대치(replace)한다.
  - 문자열 상수 안에 포함된 매크로 상수는 문자열의 일부이므로 대치되지 않는다.

전처리기 수행 전

```
#define PI 3.14

int main(void)
{
   double r;
   double area

   scanf("%lf", &r);
   area = PI * r * r;
   printf("PI = %.2f\n", PI);
}
```

전처리기 수행 후

```
int main(void)
{
   double r;
   double area;

   scanf("%lf
   area = 3.14 * r * r;
   printf("PI = %.2f\n", 3.14);
}
```

#define문은 사라진다.

매크로 대치

매크로 대치

문자열의 일부는 대치되지 않는다.

표준 C 라이브러리의 매크로 상수

limits.h

```
#define SCHAR_MIN   (-128)
#define SCHAR_MAX     127
#define UCHAR_MAX     0xff
#define CHAR_MIN    SCHAR_MIN
#define CHAR_MAX    SCHAR_MAX
#define SHRT_MIN    (-32768)
#define SHRT_MAX      32767
#define USHRT_MAX     0xffff
#define INT_MIN     (-2147483647 - 1)
#define INT_MAX       2147483647
#define UINT_MAX      0xffffffff
#define LONG_MIN    (-2147483647L - 1)
#define LONG_MAX      2147483647L
#define ULONG_MAX     0xffffffffUL
#define LLONG_MAX     9223372036854775807i64
#define LLONG_MIN   (-9223372036854775807i64 - 1)
#define ULLONG_MAX    0xffffffffffffffffui64
```

예제 3-12

const 변수

- 값을 변경할 수 없는 변수

예제 3-13 const 변수의 사용

```
const int MAX_BUF = 32;
MAX_BUF = 128;      // const 변수는 변경할 수 없으므로 컴파일 에러 발생
```

- const 변수는 반드시 선언 시 초기화해야 한다.

```
const float VAT_RATE;          // 초기화 안하면 컴파일 에러는 아니지만 사용할 수 없게 된다.
VAT_RATE = 0.1;                // const 변수에 대입을 할 수 없으므로 컴파일 에러 발생
```

기호 상수를 사용해야 하는 이유 (1/2)

- 기호 상수를 사용하면 프로그램을 수정하기 쉽다.

리터럴 상수를 사용하는 경우

```
double area, perimeter;
int r = 5;

area = 3.14 * r * r;
perimeter = 2 * 3.14 * r;
```

3.141592

3.141592

리터럴 상수를 사용하는
모든 곳을 수정해야 한다.

기호 상수를 사용하는 경우

```
#define PI 3.14

double area, perimeter;
int r = 5;
area = PI * r * r;
perimeter = 2 * PI * r;
```

3.141592

기호 상수를 정의하는 곳만
수정하면 된다.

기호 상수를 사용해야 하는 이유 (2/2)

- 기호 상수를 사용하면 프로그램이 이해하기 쉽다.

리터럴 상수를 사용하는 경우

```
int amount, price, total;
scanf("%d %d", &amount, &price);

total = amount*price*(1+0.1);
```

0.1이 어떤 의미인지
알 수 없다.

기호 상수를 사용하는 경우

```
const double VAT_RATE = 0.1;

int amount, price, total;
scanf("%d %d", &amount, &price);

total = amount*price*(1+VAT_RATE);
```

VAT_RATE를 사용하면
의미를 명확히 알 수 있다.

학습과제

- 학습과제
  - 3장 데이터형과 변수 교재 내용 이해하고 연습문제 풀이하여 그결과를 메일로 보내기
  - Exercise 3
    - 페이지 119~123페이지
  - 메일 주소 : wykim@gw.ac.kr
  - 메일 제목 : 학번_이름_3차~4차 과제