인하공업전문대학
INHA TECHNICAL COLLEGE

C 프로그래밍(3주차)

인하공업전문대학 컴퓨터 정보과

김한결 강사

목차

Chapter 04
변수와 자료형

학습목표

- C 언어에서 사용하는 식별자와 예약어를 알아봅니다.
- 자료형의 종류를 알아봅니다.
- 프로그램에서 사용하는 변수와 상수를 알아봅니다.
- 자료형 변환을 이해합니다.

1. 식별자와 예약어

식별자 (identifier)

- 프로그래머가 필요에 따라 선언하여 특정 목적을 수행하는 독립적인 존재
- 다른 명령어와 구별하여 명령을 수행
- 알파벳과 숫자를 조합하여 임의로 선언
- 예약어는 식별자로 사용할 수 없음

식별자 선언 규칙

- 알파벳(대소 문자), 숫자, '_'(언더바)를 사용할 수 있음
- 알파벳 대문자와 소문자를 구분해서 사용해야 함
- 알파벳 또는 '_'로 식별자를 시작해야 함
- 숫자로 식별자를 시작해서는 안 됨
- 단어 중간에 공백은 허용하지 않음
- 한글은 사용할 수 없음
- 예약어는 사용할 수 없음

1. 식별자와 예약어

예약어 (reserved word)

- 컴파일러가 특정 목적의 프로그램을 수행하기 위해 사전에 예약해 놓은 명령어
- 고유의 명령을 수행하는 명령어
  → 변수나 함수의 이름으로 사용할 수 없음

자주 사용하는 예약어

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

2. 변수의 개념

■ 변수 (variables)

- 계속해서 변하는 값
- 프로그램이 실행되는 동안 수시로 데이터 값이 변하는 정보를 보유하고 있는 식별자
- 변수를 선언하면 컴파일러는 자료형의 크기에 따라 메모리 공간을 확보
- 상수 또는 문자와 문자열 등을 대입하기 위한 저장공간으로 사용

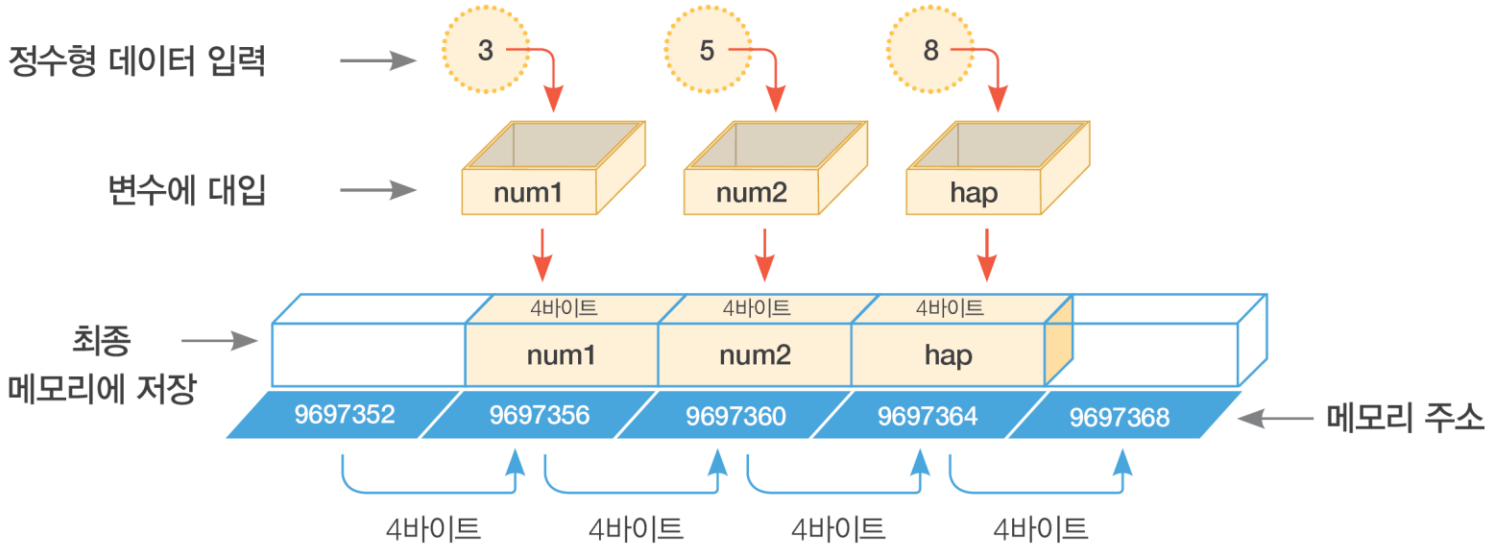


그림 4-1 변수를 선언하면 변수의 이름이 붙은 메모리 공간이 확보됨

2. 변수의 개념

■ 변수를 활용한 덧셈 연산식

- 변수를 선언하면 자료형의 크기만큼 메인 메모리인 RAM 공간을 확보
- 변수에 상수를 대입하면 확보한 메모리 공간에 저장

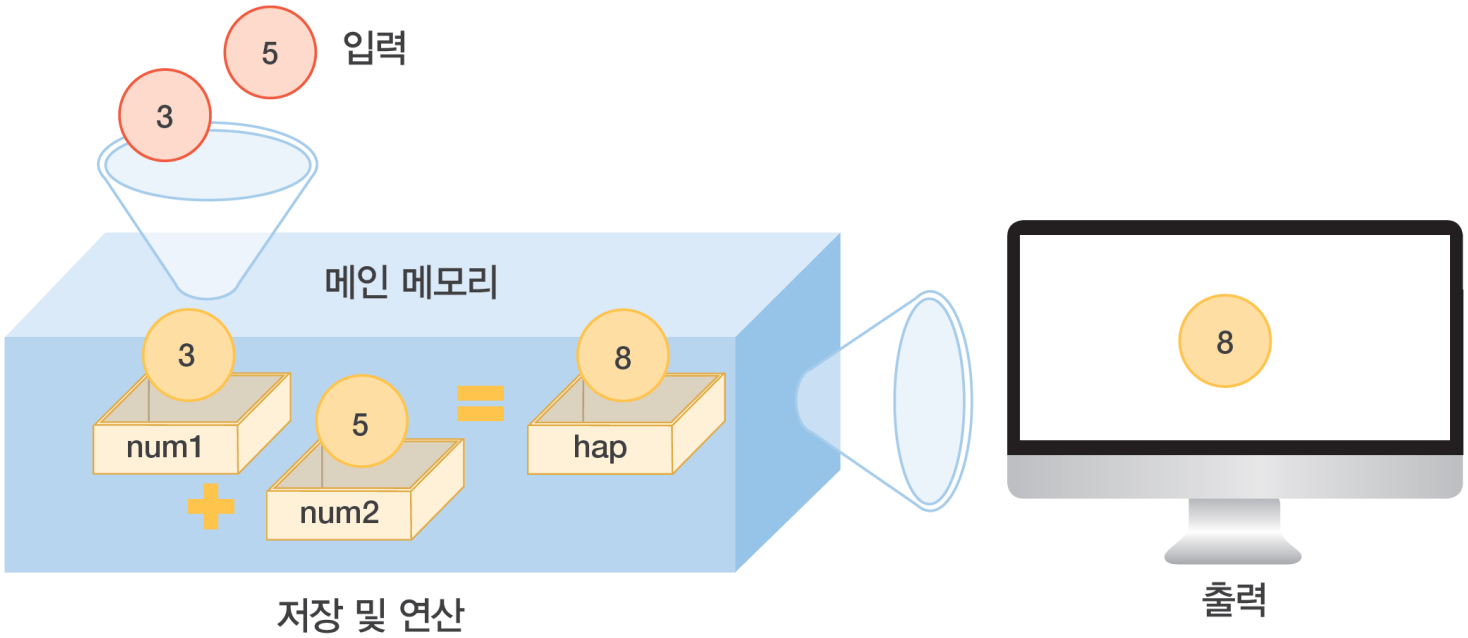


그림 4-2 변수에 값을 대입하고 덧셈 연삭식을 수행하는 과정

3. 변수 이름 표기법

■ 변수 이름 선언 규칙

- 알파벳 a~z, A~Z, 숫자 0~9 그리고 언더바(_)를 조합하여 구성
- 2바이트로 구성된 한글은 C 언어의 변수명으로 사용 불가
- 대문자와 소문자를 엄격하게 구분
- 용도에 맞는 이름으로 선언
- 대입하고자 하는 데이터의 성격을 쉽게 파악할 수 있는 단어 사용

```
count, student_name, _test, Data_8, TIME_zone, ···   // 올바른 변수 선언 (○)
5_data, 8888, space  zone, ···                       // 잘못된 변수 선언 (×)
```

```
Count,  count, COUNT, counT                          // 모두 다른 변수로 인식됨
```

3. 변수 이름 표기법

카멜 표기법

- 연결되는 2개 단어의 첫 글자를 대문자로 표기
- 낙타의 등과 비슷하다는 유래에서 탄생한 표기법

```
myCount,  startTime, studyGroup, forSum
```

스네이크 표기법

- 변수의 용도와 특성을 알 수 있는 접두어를 붙이거나 2개 이상의 단어를 언더바(_)로 연결해서 변수를 선언하는 방법

```
int_num, circle_radius, member_count
```

4. 변수를 사용하는 이유

■ 프로그램 유지 · 보수의 효율성

- 소스 코드를 수정하거나 업데이트를 해야 할 때 변수에 대입되는 값만 수정하면 됨
- 변수는 변할 수 있는 값을 대입하기 위해 선언하고 사용하는 것이기 때문에 자료형은 신중히 결정
- 선언한 변수의 자료형에서 처리할 수 있는 값의 유효 범위를 고려하여 선언(오버플로 또는 언더플로 발생 예방)

```
int a, b, sum;
a = 10;
b = 20;
sum = a + b;
```

소스 코드에서 값이 변동되면 변수에 대입하는 값만 수정하면 됨

1. 자료형의 개념

■ 자료형 (Data type)

- 데이터를 저장하기 위한 공간인 변수를 선언할 때 메모리의 크기를 지정하는 단위
- 사용할 데이터의 크기에 맞게 변수의 자료형 선언

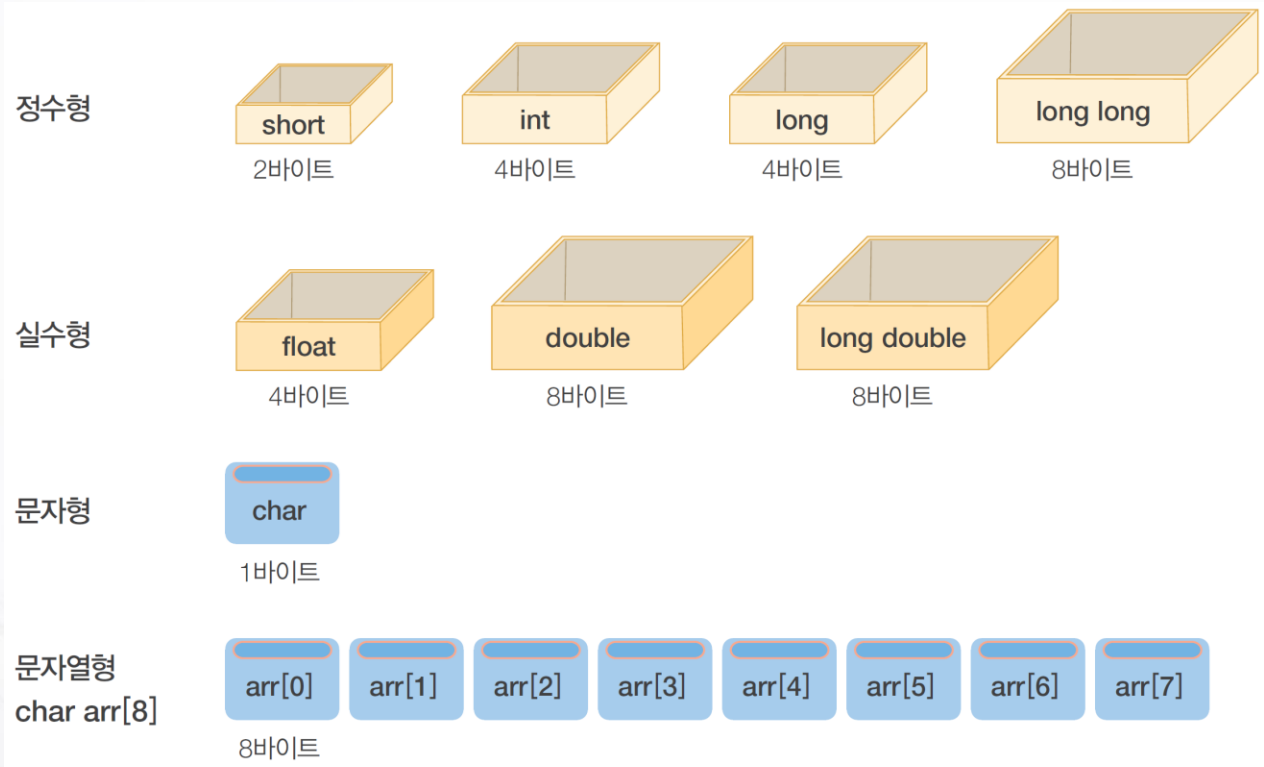


그림 4-3 자료형별 바이트 단위의 크기 비교

2. 자료형의 종류

■ C 언어 : 정수형, 실수형, 문자형 세가지 자료형 제공

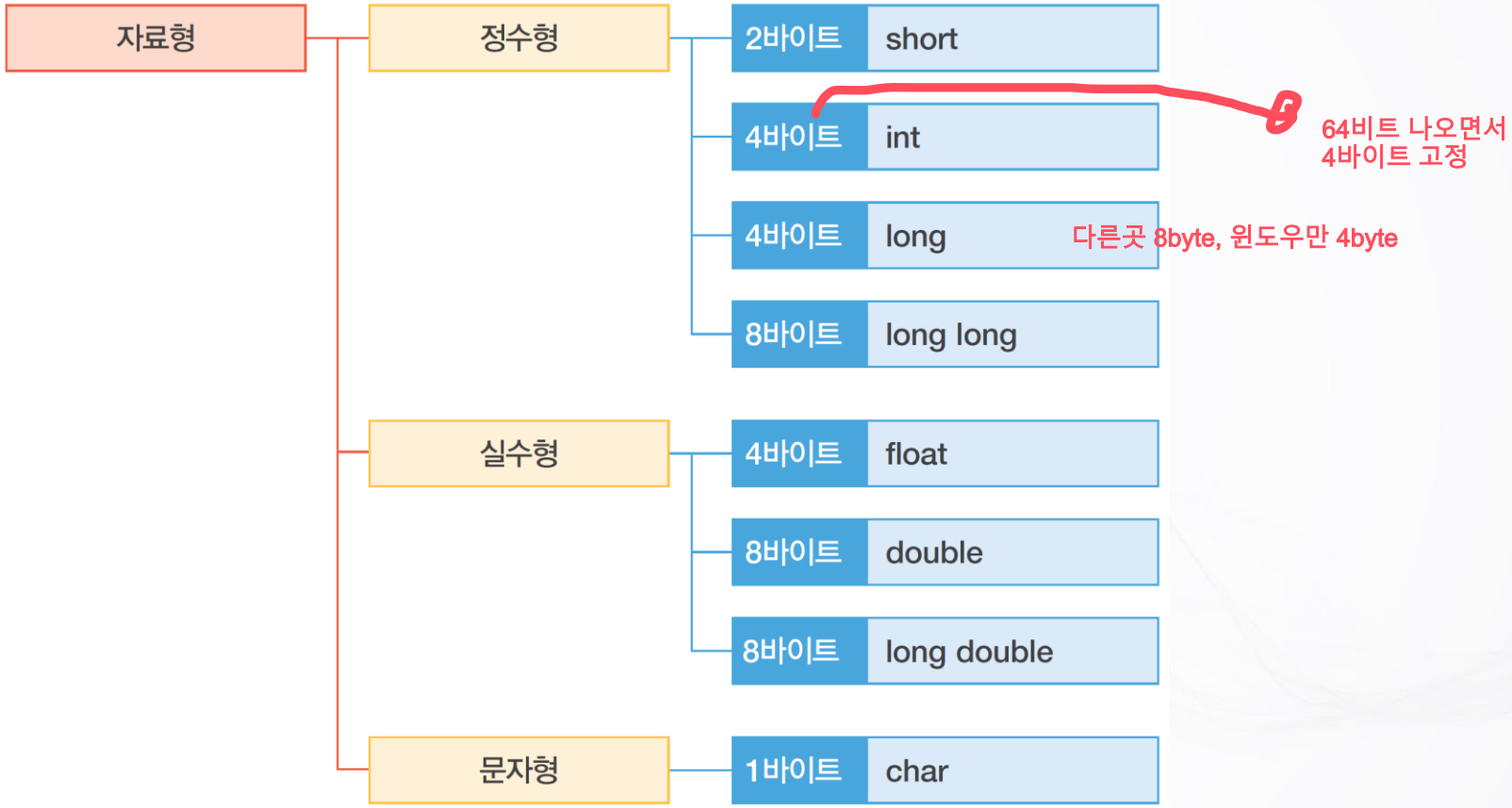


그림 4-4 C 언어에서 제공하는 자료형의 종류

3. 변수와 자료형의 관계

변수

- 데이터를 입력하기 위해 선언하는 식별자로 할당된 메모리 공간을 사용
- int a;
  → int형 자료형의 메모리 공간 4바이트 크기 확보
- double b;
  → 소수점을 포함하여 메모리 공간 8바이트 확보

변수의 자료형 결정

- 변수에 대입할 값의 최소값과 최대값 고려
- 데이터 값에 대한 소수점 여부
- 소수점 이하 자릿수가 모두 1.000000과 같이 출력되는 논리 오류 예방

1. 변수 선언과 초기화

■ 변수 선언

- 컴파일러에게 사용할 변수를 미리 알리는 것
- 변수 선언 시 사용 목적과 유효 범위에 적합한 자료형을 지정
- 변수를 선언하면 변수 이름으로 자료형의 크기만큼 메모리 공간 확보
- 데이터의 유효 범위 값에 해당하는 자료형으로 변수 선언
  → 같은 4바이트의 int와 float이더라도 소수점 유무로 구분하여 선언

```
int a;
int b;
int sum;
```

또는 int a, b, sum;

1. 변수 선언과 초기화

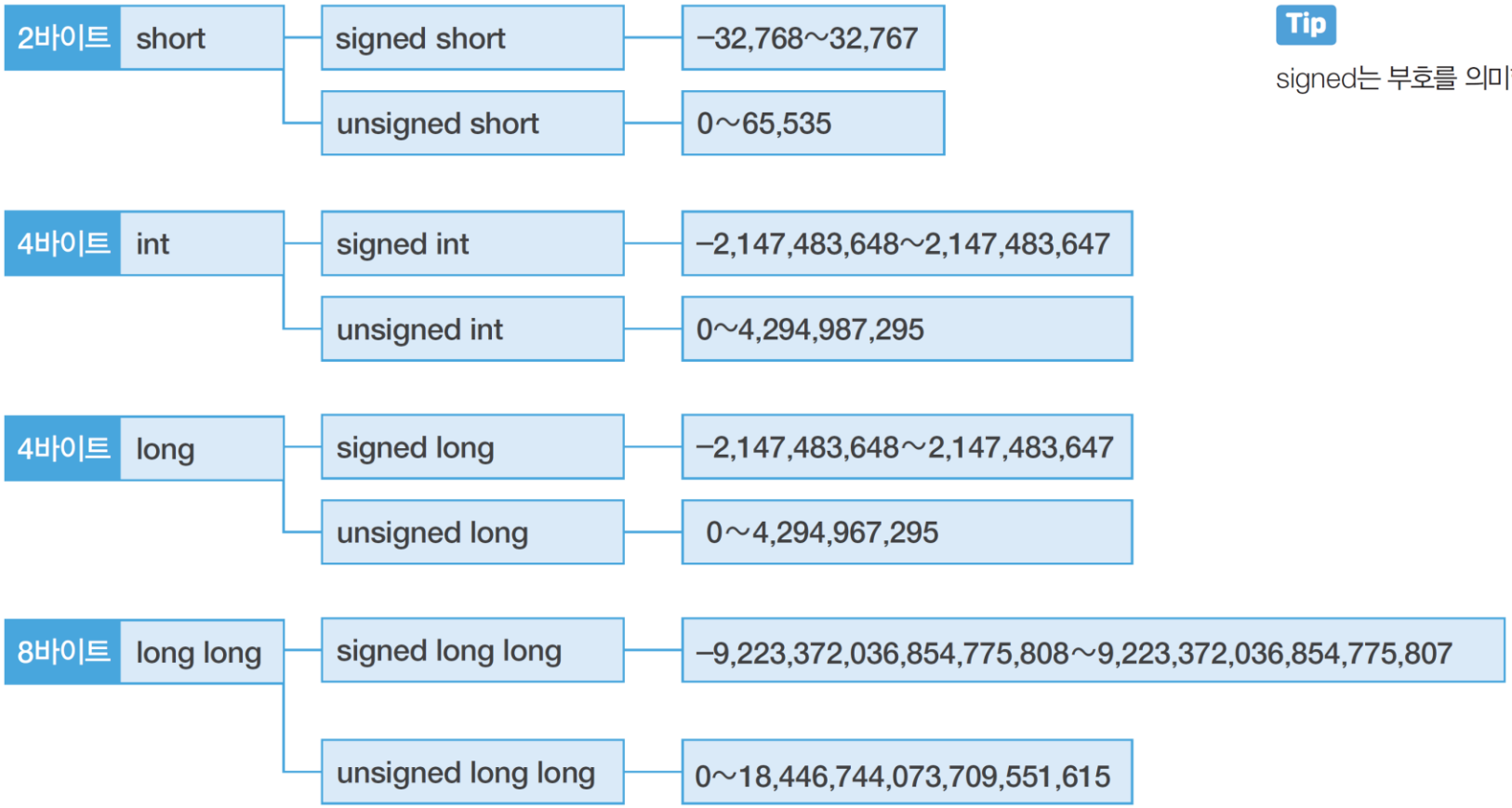
■ 변수 초기화

- 변수를 선언하면서 초깃값을 변수에 대입하는 과정
- 메모리에는 알 수 없는 가비지값이 있을 수 있음
  → 변수를 선언함과 동시에 초기화 수행
  → 누적합계를 저장할 변수는 반드시 0으로 초기화 선언

```
int num = 0;              // 4바이트 크기의 int형 변수값을 0으로 초기화
```

2. 정수형 변수

■ 정수형 자료형의 종류



Tip

signed는 부호를 의미합니다.

그림 4-5 정수형 자료형의 종류와 유효 범위

2. 정수형 변수

■ 정수형 자료형의 크기

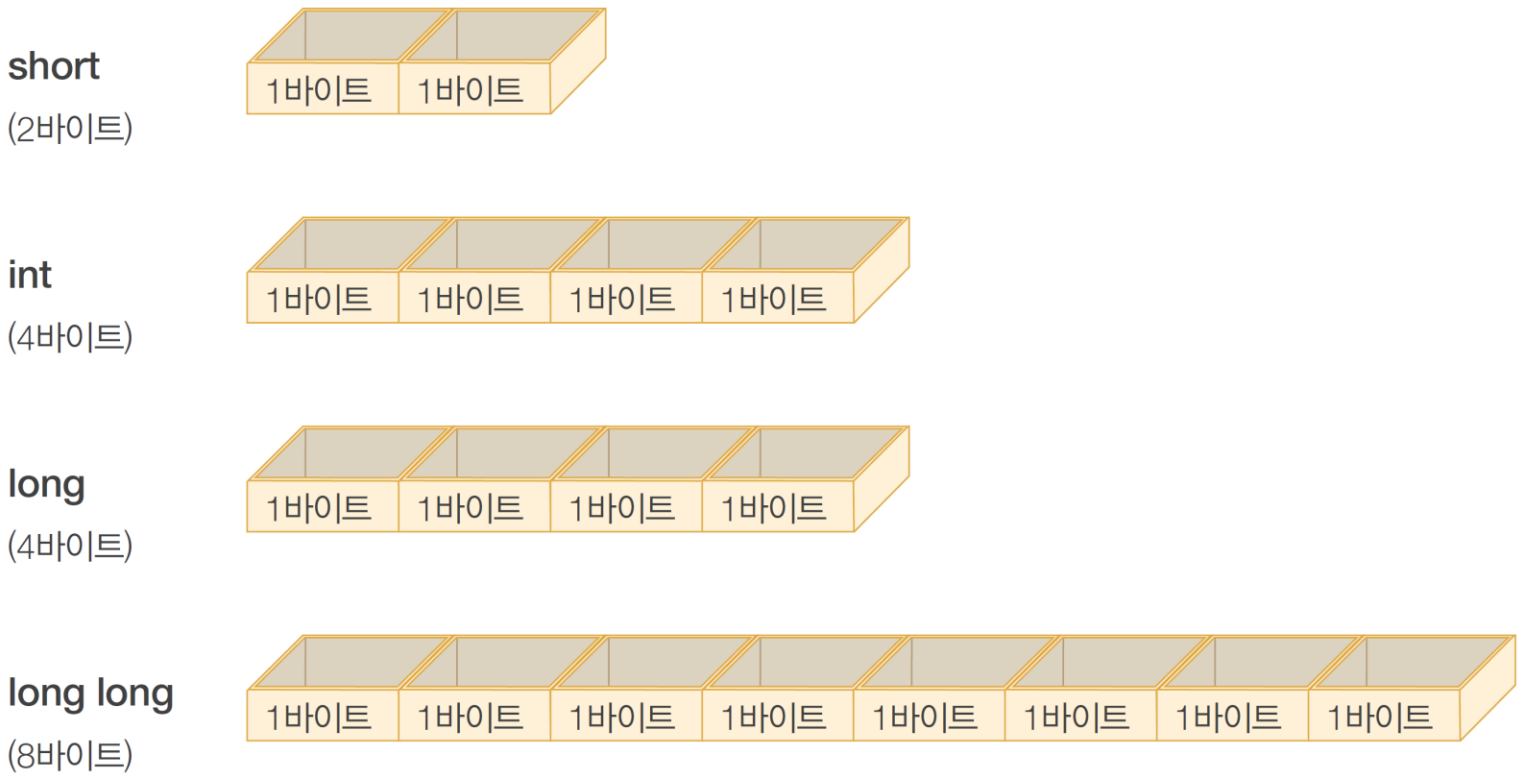


그림 4-6 정수형 자료형의 크기 비교

2. 정수형 변수

■ sizeof( ) 함수를 사용한 자료형 크기 알아보기

예제 4-1 정수형 자료형의 크기 출력 ex04_01.c

```c
#include <stdio.h>

int main(void)
{
    short num1;             // short 정수형 변수
    int num2;               // int 정수형 변수
    long long num3;         // long long 정수형 변수

    printf("자료형의 크기를 구하는 함수 : sizeof( ) 함수\n");
    printf("1.short형 변수 : %d바이트\n", sizeof(num1));
    printf("2.int형 변수 : %d바이트\n", sizeof(num2));
    printf("3.long long형 변수 : %d 바이트\n", sizeof(num3));
    printf("1바이트는 8비트입니다.\n");
    printf("4.short형 변수 : %d비트\n", sizeof(num1) * 8);
    printf("5.int형 변수 : %d비트\n", sizeof(num2) * 8);
    printf("6.long long형 변수 : %d비트\n", sizeof(num3) * 8);
    return 0;
}
```

```
자료형의 크기를 구하는 함수 : sizeof( ) 함수
1.short형 변수 : 2바이트
2.int형 변수 : 4바이트
3.long long형 변수 : 8바이트
1바이트는 8비트입니다.
4.short형 변수 : 16비트
5.int형 변수 : 32비트
6.long long형 변수 : 64비트
```

2. 정수형 변수

■ 정수형 자료형의 부호 표시

- 2바이트(16비트) 정수형 자료형의 변수를 선언 시 부호 표현 여부 결정

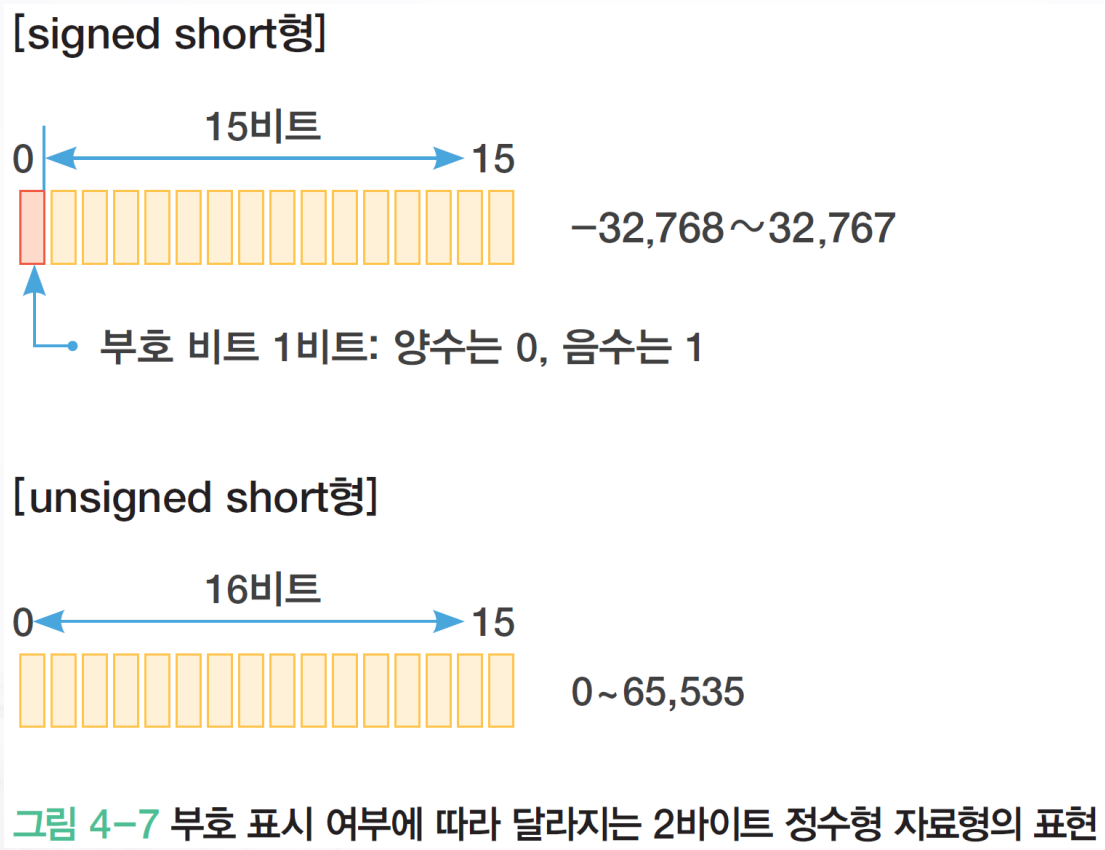
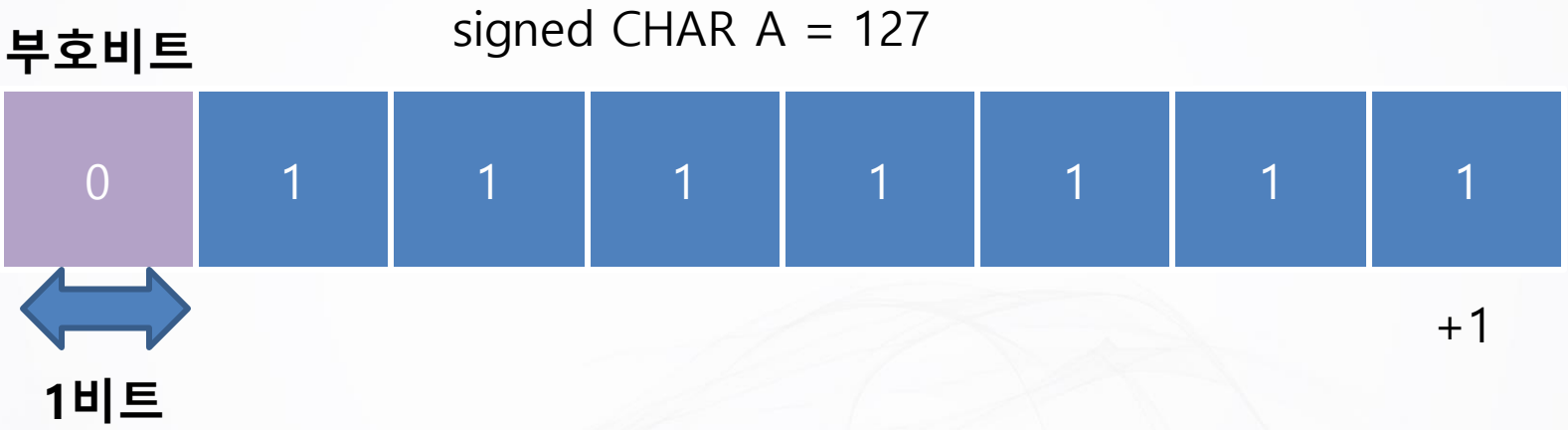


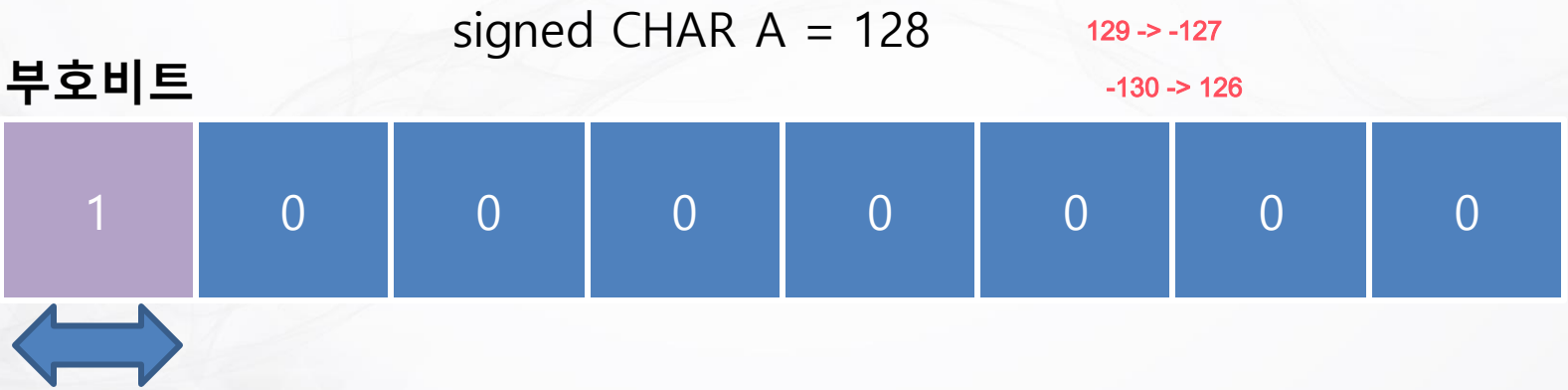
그림 4-7 부호 표시 여부에 따라 달라지는 2바이트 정수형 자료형의 표현

2. 정수형 변수

■ 오버플로(overflow) 발생

- 자료형의 유효 범위 최댓값보다 더 큰 값이 대입된 논리 오류
- 오버플로 발생 → 자료형의 유효 범위 최솟값부터 다시 시작하여 연산한 후 결과값 출력





2. 정수형 변수

■ 오버플로(overflow) 발생

- 자료형의 유효 범위 최댓값보다 더 큰 값이 대입된 논리 오류
- 오버플로 발생 → 자료형의 유효 범위 최솟값부터 다시 시작하여 연산한 후 결과값 출력

main.c

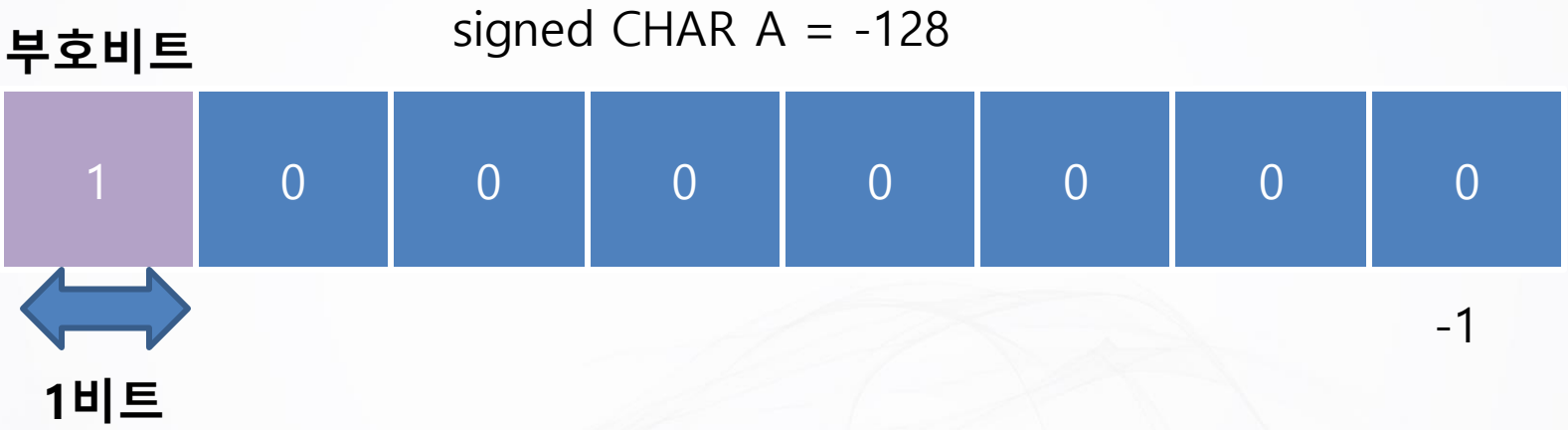
```c
#include <stdio.h>

int main()
{
    char a;
    a = 127;
    printf("%d\n", a);
    a = 128;
    printf("%d\n", a);
    return 0;
}
```

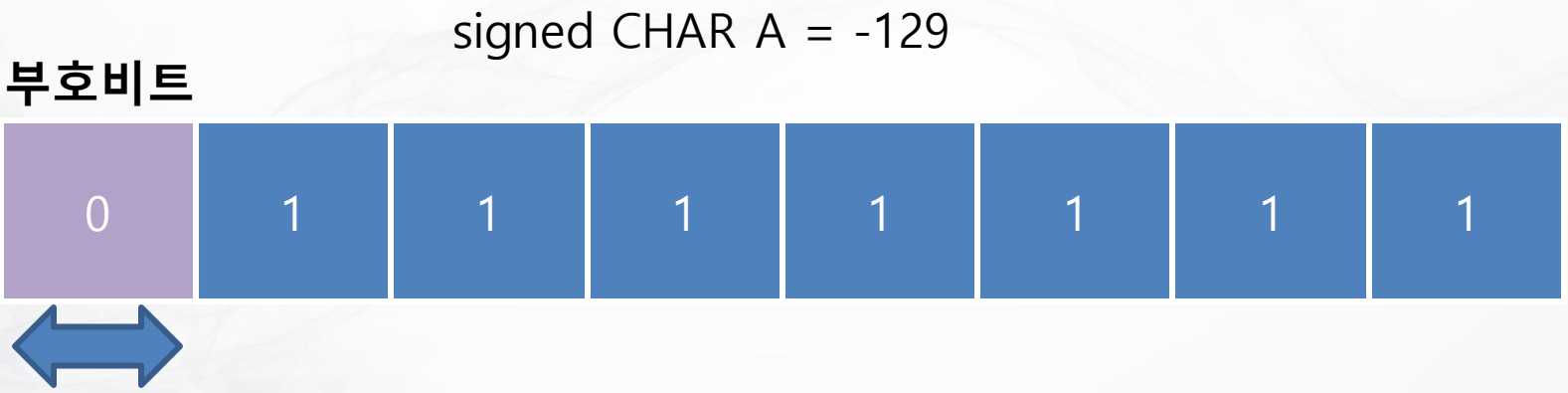
```
127
-128


...Program finished with exit code 0
Press ENTER to exit console.
```

2. 정수형 변수

■ 언더플로(underflow) 발생

- 자료형 유효 범위의 최솟값보다 더 작은 값이 변수에 대입되는 논리오류
- 언더플로 발생 → 자료형 유효 범위의 최댓값으로 돌아가서 다시 연산하여 실행 결과 출력





2. 정수형 변수

main.c

```c
#include <stdio.h>

int main()
{
    char a;
    a = -128;
    printf("%d\n", a);
    a = -129;
    printf("%d\n", a);
    return 0;
}
```

```
main.c: In function 'main':
main.c:8:13: warning: overflow in
    8 |          a = -129;
      |              ^
-128
127
```

3. 실수형 변수

■ 실수형 자료형의 종류

중간고사 이후

- 실수형 변수 : 소수점을 포함하는 데이터를 처리하기 위한 저장 공간
- 부동 소수점 : 소수점이 특정 위치에 고정되어 있지 않고 따로 지정

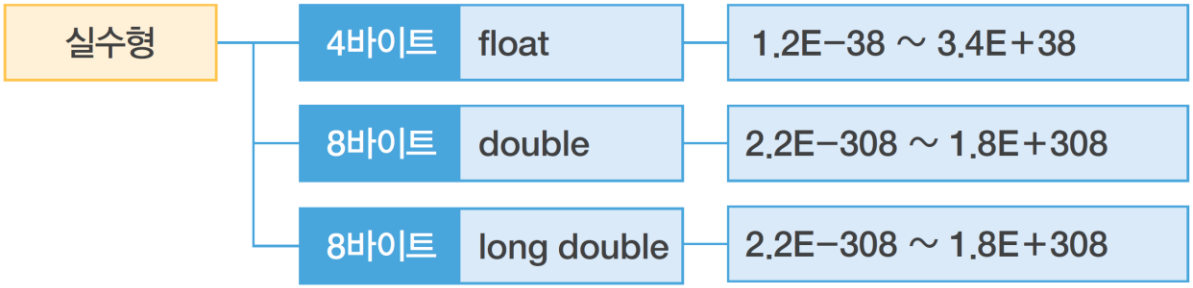


그림 4-12 실수형 자료형의 종류와 유효 범위

2진수로 소수점 구현
하면 정확도가 떨어짐.
0.1 + 0.2 != 0.3

3. 실수형 변수

■ 실수형 자료형의 크기

그림 4-13 실수형 자료형의 크기 비교

3. 실수형 변수

■ 4바이트 실수형 자료형의 표현

- float형 변수 : 소수 여섯째 자리까지만 유효한 값을 출력
  → 일곱째 자리에서 반올림이 자동으로 수행됨

```
printf("float형 출력 = %f", 3.12345678901234);
```

출력 결과

```
float형 출력 = 3.123457
```

3. 실수형 변수

■ 실수형 자료형 변수 선언

- 체질량 지수 BMI 산출 공식

```
BMI = 체중(kg) ÷ (키(m) × 키(m))    // 키의 단위는 센티미터(cm)가 아닌 미터(m)임을 주의해야 함
```

- 실습 코드 : 다음 슬라이드

3. 실수형 변수

■ 실수형 자료형 변수 선언

예제 4-7 실수형 데이터로 연산한 결과값을 실수형으로 출력 ex04_07.c

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     float height, weight, swap, bmi;   // 4바이트 실수형 자료형 변수
06
07     printf("키와 체중은 소수 둘째 자리까지 입력\n");
08
09     printf("키 입력(cm) : ");
10     scanf_s("%f", &height);
11     printf("체중 입력(kg) : ");
12     scanf_s("%f", &weight);
13
14     swap = height / 100;
15     bmi = weight / (swap * swap);
16     printf("BMI = 체중(kg) / (키(m) x 키(m))\n");
17     printf("센티미터로 입력한 키를 미터로 환산해서 사용\n");
18     printf("BMI = %.2f\n", bmi);
19     return 0;
20 }
```

```
키와 체중은 소수 둘째 자리까지 입력
키 입력(cm) : 185.65
체중 입력(kg) : 83.38
BMI = 체중(kg) / (키(m) x 키(m))
센티미터로 입력한 키를 미터로 환산해서 사용
BMI = 24.19
```

데이터 입력

3. 실수형 변수

■ 실수형 데이터의 소수점 이하 값이 모두 0으로 출력되는 경우

예제 4-8 정수형 데이터로 연산한 실수형 결과값을 정수형으로 출력 ex04_08.c

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int num1, num2;    // 4바이트 정수형 변수 선언
06     float result;      // 4바이트 실수형 변수 선언
07
08     printf("정수 2개를 입력하세요.\n");
09     printf("첫 번째 정수 num1 = ");
10     scanf_s("%d", &num1);
11     printf("두 번째 정수 num2 = ");
12     scanf_s("%d", &num2);
13
14     result = num1 / num2;
15     printf("나눗셈 연산 = num1 / num2 = %d / %d = %f\n", num1, num2, result);
16     return 0;
17 }
```

정수와 정수의 연산에서 실수값은 절대 산출될 수 없음

```
정수 2개를 입력하세요.
첫 번째 정수 num1 =  5
두 번째 정수 num2 =  3
나눗셈 연산 = num1 /num2 = 5 / 3 = 1.000000
```

1.666667을 생각했던 논리 오류 발생

3. 실수형 변수

■ 캐스트 연산자로 자료형 변환

- 캐스트 연산자 : 자료형을 강제로 변환하는 연산자

예제 4-9 캐스트 연산자로 강제 자료형 변환 ex04_09.c

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int num1, num2;                  // 4바이트 정수형 변수 선언
06     float result;                    // 4바이트 실수형 변수 선언
07
08     printf("정수 2개를 입력하세요.\n");
09     printf("첫 번째 정수 num1 = ");
10     scanf_s("%d", &num1);
11     printf("두 번째 정수 num2 = ");
12     scanf_s("%d", &num2);
13
14     result = (float)num1 / num2;     // 캐스트 연산자로 자료형 강제 변환
15     printf("나눗셈 연산 = (float)num1 / num2 = %d / %d = %f\n", num1, num2, result);
16     return 0;
17 }
```

캐스트 연산자

```
정수 2개를 입력하세요.
첫 번째 정수 num1 = 5
두 번째 정수 num2 = 3
나눗셈 연산 = (float)num1 / num2 = 5 / 3 = 1.666667
```

논리 오류 해결

3. 실수형 변수

8바이트 실수형 자료형의 표현

- double형 변수 : 소수 16째 자리까지만 유효한 값으로 출력

```
printf("double형 출력 = %.16f", 3.1234567890123456789);
```

출력 결과

```
double형 출력 = 3.1234567890123457
```

3. 실수형 변수

■ float형과 double형으로 삼각형의 넓이 구하기

예제 4-10 실수형 자료형인 float형과 double형 사용 ex04_10.c

```c
#include <stdio.h>

int main(void)
{
    float base, height;      // 4바이트 float 실수형 변수 선언
    double result;           // 8바이트 double 실수형 변수 선언

    printf("밑변과 높이에 소수 둘째 자리까지 데이터 입
    printf("밑변 입력(cm) : ");
    scanf_s("%f", &base);
    printf("높이 입력(cm) : ");
    scanf_s("%f", &height);

    result = (base * height) / 2;

    printf("삼각형 넓이(소수 6째 자리까지) 출력 : %f\n", result);
    printf("삼각형 넓이(소수 15째 자리까지) 출력 : %.15f\n", result);
    printf("삼각형 넓이(소수 16째 자리까지) 출력 : %.16f\n", result);
    printf("삼각형 넓이(소수 17째 자리까지) 출력 : %.17f\n", result);
    printf("삼각형 넓이(소수 18째 자리까지) 출력 : %.18f\n", result);
    return 0;
}
```

```
밑변과 높이에 소수 둘째 자리까지 데이터 입력
밑변 입력(cm) : 12.87
높이 입력(cm) : 33.98
삼각형 넓이(소수 6째 자리까지) 출력 : 218.661301
삼각형 넓이(소수 15째 자리까지) 출력 : 218.661300659179688
삼각형 넓이(소수 16째 자리까지) 출력 : 218.6613006591796875
삼각형 넓이(소수 17째 자리까지) 출력 : 218.66130065917968750
삼각형 넓이(소수 18째 자리까지) 출력 : 218.661300659179687500
```

데이터 입력

4. 문자형 변수

■ 문자형 자료형

- 문자형 변수 : 알파벳 문자 1개당 1바이트로 처리
- 아스키코드 : 모든 문자를 아스키코드값으로 표현
- 큰 범주에서는 정수형 자료형에 포함 : 아스키코드값으로 연산 가능
- 변수에 문자를 대입 : 문자를 작은따옴표로 반드시 감싸줘야 함

```
char alphabet;
alphabet = 'A';
```

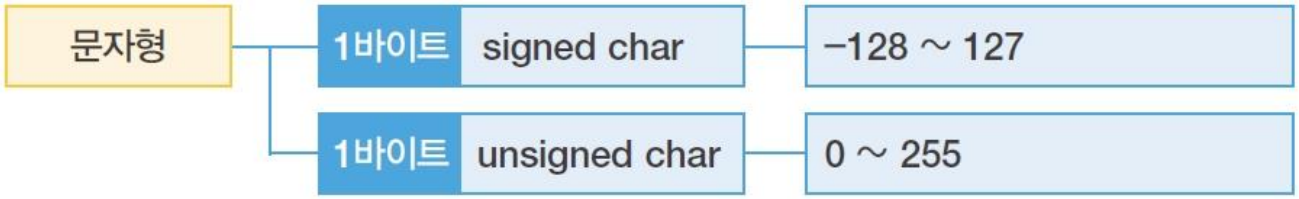


그림 4-14 문자형 자료형의 종류와 유효 범위

4. 문자형 변수

■ 알파벳에 해당하는 아스키코드값 출력

예제 4-11 알파벳 대문자를 아스키코드값으로 출력 ex04_11.c

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     char alphabet;      // 1바이트 문자형 변수 선언
06     alphabet = 'A';     // 문자는 반드시 작은따옴표로 감싸야 함
07
08     printf("알파벳 대문자를 문자형 변수에 대입 : alphabet = \'A\'\n");
09     printf("알파벳 출력 : %c\n", alphabet);
10     printf("아스키코드값 출력 : %d\n", alphabet);
11     return 0;
12 }
```

```
알파벳 대문자를 문자형 변수에 대입 : alphabet = 'A'
알파벳 출력 : A
아스키코드값 출력 : 65
```

4. 문자형 변수

■ 문자형 변수를 정수형처럼 사용

예제 4-12 문자형 변수에 숫자를 더한 후 알파벳으로 출력 ex04_12.c

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     char alphabet, add;      // 1바이트 문자형 변수 선언
06     alphabet = 'A';          // 문자는 반드시 작은따옴표로 감싸야 함
07     add = alphabet + 5;      // 문자형 변수에 5를 더하는 덧셈 연산
08
09     printf("알파벳 대문자를 문자형 변수에 대입 : alphabet = \'A\'\n");
10     printf("처음 대입한 알파벳 출력 : %c\n", alphabet);
11     printf("처음 알파벳에 5를 더한 알파벳 : %c\n", add);
12     printf("변경된 알파벳의 아스키코드값 : %d\n", add);
13     return 0;
14 }
```

```
알파벳 대문자를 문자형 변수에 대입 : alphabet = 'A'
처음 대입한 알파벳 출력 : A
처음 알파벳에 5를 더한 알파벳 : F
변경된 알파벳의 아스키코드값 : 70
```

4. 문자형 변수

■ 입력한 문자에 해당하는 아스키코드값 출력

예제 4-13 입력한 알파벳을 아스키코드값으로 출력 ex04_13.c

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     char input_alpha;        // 1바이트 문자형 변수 선언
06
07     printf("알파벳 입력 : ");
08     scanf_s("%c", &input_alpha, sizeof(input_alpha));
09
10     printf("입력한 알파벳 출력 : %c\n", input_alpha);
11     printf("입력한 알파벳의 아스키코드값 : %d\n", input_alpha);
12     return 0;
13 }
```

```
알파벳 입력 : a
입력한 알파벳 출력 : a
입력한 알파벳의 아스키코드값 : 97
```

1. 상수의 개념

상수 (Constant)

- 변수에 대입되는 값
- 프로그램 수행 중에는 변하지 않는 데이터
- C 언어에서 정수형 상수, 실수형 상수, 문자형 상수, 문자열형 상수 제공
- 문자형 상수 : 작은따옴표(‘ ’)로 묶어줌
- 문자열형 상수 : 큰따옴표(“ ”)로 묶어줌

2. 정수형 상수

■ 정수형 상수의 의미

- 소수점 없이 정수형 자료형으로 선언한 변수에 대입되는 값
- 모든 정수형 상수 앞에 + 또는 – 부호를 붙일 수 있음
- 부호를 생략하면 무조건 양의 정수로 취급
- 진법에 따라 10진수, 8진수, 16진수 상수로 구분

표 4-1 정수형 상수를 나타내는 접미사

자료형	접미사	사용 예
unsigned int	U 또는 u	123U 또는 123u
long	L 또는 l	456L 또는 456l
unsigned long	UL 또는 ul	789UL 또는 789ul

2. 정수형 상수

표 4-2 10진수에 대한 2진수, 8진수, 16진수 숫자 표현

10진수	2진수	8진수	16진수
0	0000 0000 0000 0000	00	0x0
1	0000 0000 0000 0001	01	0x1
2	0000 0000 0000 0010	02	0x2
3	0000 0000 0000 0011	03	0x3
4	0000 0000 0000 0100	04	0x4
5	0000 0000 0000 0101	05	0x5
6	0000 0000 0000 0110	06	0x6
7	0000 0000 0000 0111	07	0x7
8	0000 0000 0000 1000	010	0x8
9	0000 0000 0000 1001	011	0x9
10	0000 0000 0000 1010	012	0xA
11	0000 0000 0000 1011	013	0xB
12	0000 0000 0000 1100	014	0xC
13	0000 0000 0000 1101	015	0xD
14	0000 0000 0000 1110	016	0xE
15	0000 0000 0000 1111	017	0xF
16	0000 0000 0001 0000	020	0x10
17	0000 0000 0001 0001	021	0x11
18	0000 0000 0001 0010	022	0x12
19	0000 0000 0001 0011	023	0x13
20	0000 0000 0001 0100	024	0x14

Tip

8진수는 숫자 앞에 0을 붙이고 16진수는 0x 또는 0X를 붙인 형태입니다.

1바이트 계산 나옴

2. 정수형 상수

2진수를 10진수로 변환

2진수	0	1	0	1	1	0	0	0
자릿수	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
10진수	128	64	32	16	8	4	2	1

64 + 16 + 8 = 88(10)

2진수를 8진수로 변환

10진수 3 10진수 5

2진수	00011101	00010101	00010011	00011111
8진수	35(8)	25(8)	23(8)	37(8)

자릿수	4	2	1
2진수	1	0	1
8진수	5		

2. 정수형 상수

■ 2진수를 16진수로 변환

2진수	1010	1011	1100	1101	1110	1111	10000	10001
10진수	10	11	12	13	14	15	16	17
16진수	A	B	C	D	E	F	10	11

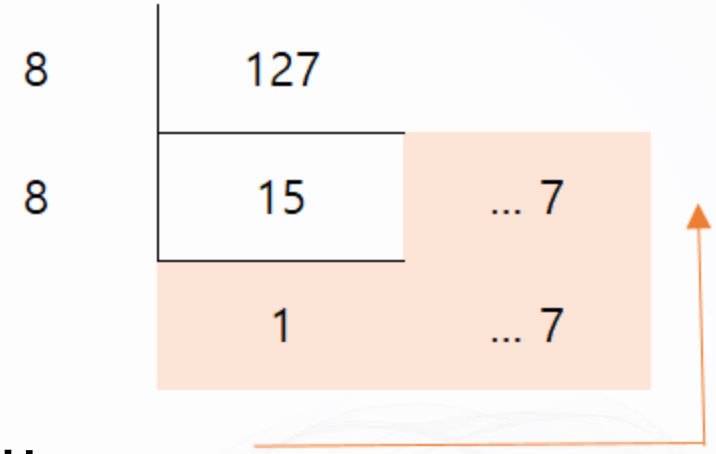
10진수 12 -> c

2진수	0111100	1110101	1001000	10000000000
16진수	1C(16)	75(16)	48(16)	400(16)

2. 정수형 상수

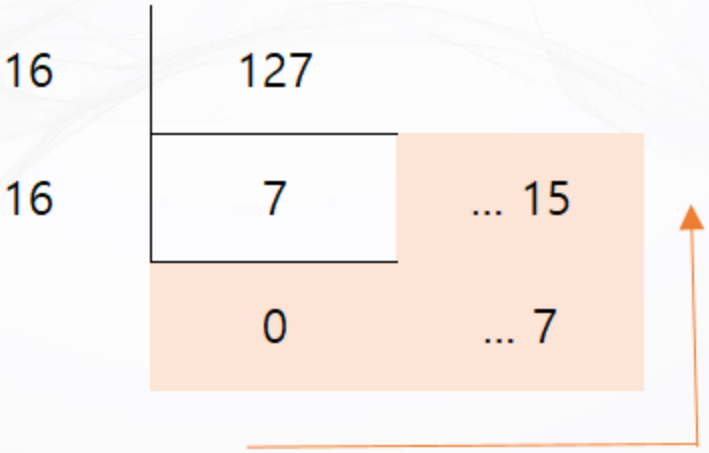
10진수를 8진수로 변환

127(10) -> 177(8)



10진수를 16진수로 변환

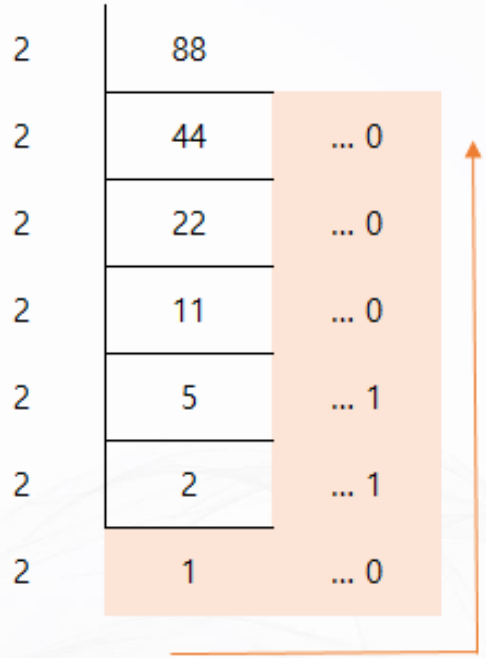
127(10) -> 7F(16)



2. 정수형 상수

- **10진수를 2진수로 변환**

88(10) -> 01011000(2)



2. 정수형 상수

■ 8진수를 10진수로 변환

8진수	1	2	7
자릿수	8^2	8^1	8^0
10진수	64	16	7

64+16+7 = 87 → 팔진수: 127

16진수	1	3	F
자릿수	16^2	16^1	16^0
10진수	256	48	15

256+48+15 = 319 → 16진수: 13F

2. 정수형 상수

8진수를 16진수로 변환

37(8) -> 1F(16)

- 37(8) -> 2진수 변환 : 011111(2)
- 11111(2) -> 16진수 변환 : 1F(16)

16진수를 8진수로 변환

1F(16) -> 37(8)

- 1F(16) -> 2진수 변환 : 00011111(2)
- 00011111(2) -> 8진수 변환 : 37(8)

2. 정수형 상수

8421 값으로 진수 변환

- 2진수의 자리에 따라 8421 값으로 1, 2, 4, 8, 16, 32, 64, … , 16384 를 사용해 계산하여 빠르게 변환

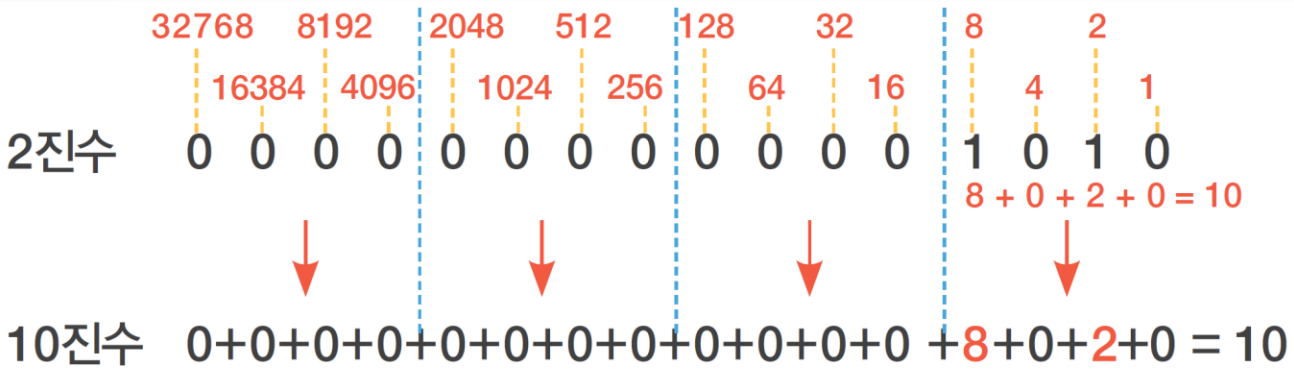


그림 4-16 8421 값으로 2진수와 10진수의 진법 변환

2. 정수형 상수

8진수와 10진수 변환

- 8진수 자릿수를 계산하여 8의 거듭제곱을 곱한 다음 더함
- 숫자 한 자를 6비트로 표현하는 캐릭터 머신의 비트 상태 표현할 때 주로 사용

$013_8 = 1 \times 8^1 + 3 \times 8^0 \quad \rightarrow \quad 013_8 = 11_{10}$

8진수 013을 2진수로 변환

$013_8 = 001\ 011_2$

8진수를 출력 변환 기호에 #o(알파벳)를 붙여 변환

printf("%#o", 013); → 8진수 013으로 출력

2. 정수형 상수

16진수 표현 방법

- 0부터 9까지의 숫자와 A부터 F 또는 a부터 f까지 알파벳 조합하여 숫자를 표현
- 0xA와 0xa 는 같은 의미
- 자릿수가 지나치게 커져서 처리가 불편한 경우에 사용
- 메모리 주소값을 표현할 때 사용
- 웹 페이지에서 색상을 표현할 때 16진수 색상 코드 사용

16진수 0xB16을 10진수로 변환

$$0xB_{16} = 11 \times 16^{0} \quad \rightarrow \quad 0xB_{16} = 11_{10}$$

16진수 0xB를 2진수로 변환

$$0xB_{16} = 1011_{2}$$

2. 정수형 상수

- **16진수를 출력 변환 기호에 #x(알파벳)를 붙여 변환**

```
printf("%#x", 0xB);   →   16진수 0xB로 출력
```

2. 정수형 상수

■ 2진수를 8진수와 16진수로 변환

- 2진수를 오른쪽 끝자리부터 차례대로 세 자리씩 묶어주면 8진수, 네 자리씩 묶으면 16진수로 쉽게 변환 가능

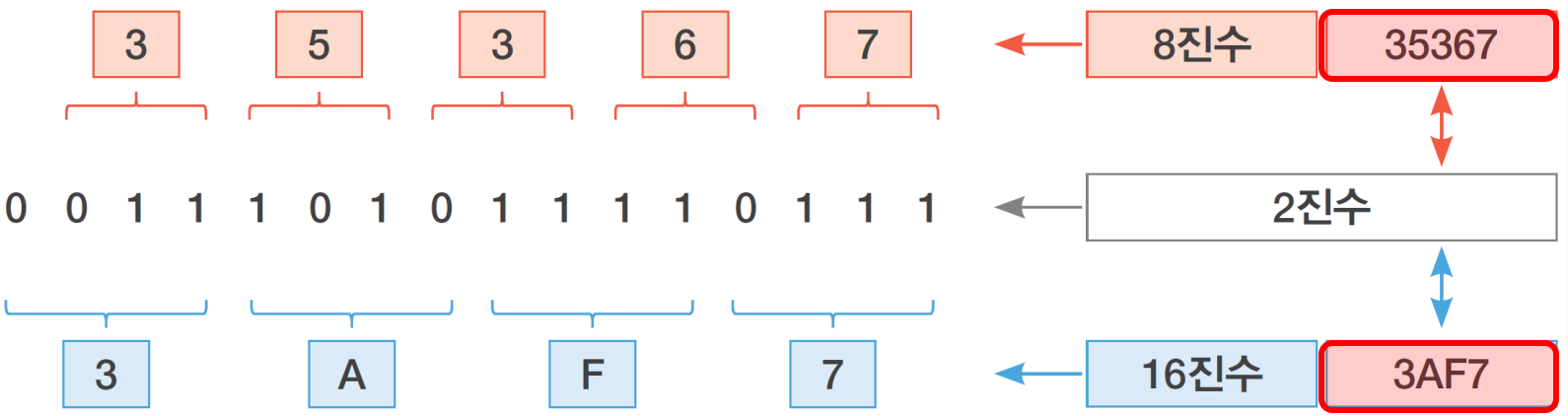


그림 4-17 2진수를 8진수와 16진수로 변환하는 방법

2. 정수형 상수

■ 정수형 상수를 10진수, 8진수, 16진수로 출력

예제 4-14 정수형 상수를 10진수, 8진수, 16진수로 출력 ex04_14.c

```c
#include <stdio.h>

int main(void)
{
    int dc_num = 123;       // 10진수 상수 대입
    int oc_num = 0123;      // 8진수 상수 대입
    int hx_num = 0x123;     // 16진수 상수 대입

    printf("10진수 상수 : %d\n", dc_num);
    printf("8진수 상수 : %#o\n", oc_num);
    printf("16진수 상수 : %#x\n", hx_num);
    printf("10진수 123을 10진수로 출력 : %d\n", dc_num);
    printf("8진수 0123을 10진수로 출력 : %d\n", oc_num);
    printf("16진수 0x123을 10진수로 출력 : %d\n", hx_num);
    printf("10진수 123을 8진수로 출력 : %o\n", dc_num);
    printf("8진수 0123을 8진수로 출력 : %o\n", oc_num);
    printf("16진수 0x123을 8진수로 출력 : %o\n", hx_num);
    printf("10진수 123을 16진수로 출력 : %x\n", dc_num);
    printf("8진수 0123을 16진수로 출력 : %x\n", oc_num);
    printf("16진수 0x123을 16진수로 출력 : %x\n", hx_num);
    return 0;
}
```

```
10진수 상수 : 123
8진수 상수 : 0123
16진수 상수 : 0x123
10진수 123을 10진수로 출력 : 123
8진수 0123을 10진수로 출력 :  83
16진수 0x123을 10진수로 출력 : 291
10진수 123을 8진수로 출력 : 173
8진수 0123을 8진수로 출력 : 123
16진수 0x123을 8진수로 출력 : 443
10진수 123을 16진수로 출력 : 7b
8진수 0123을 16진수로 출력 : 53
16진수 0x123을 16진수로 출력 : 123
```

2. 정수형 상수

■ 부호가 없는 상수와 long형 상수를 대입하여 출력

예제 4-15 부호가 없는 상수와 long형 상수 출력 ex04_15.c

```c
#include <stdio.h>

int main(void)
{
    unsigned int un_num = 12345U;          // 부호
    long lg_num = 1234567890L;             // long

    printf("부호가 없는 10진수 상수값 대입 : 12345U\n");
    printf("long형 10진수 상수값 대입 : 1234567890L\n");
    printf("부호가 없는 상수값을 10진수로 출력 : %u\n", un_num);
    printf("long형 상수값을 10진수로 출력 : %ld\n", lg_num);
    printf("부호가 없는 상수값을 8진수로 출력 : %o\n", un_num);
    printf("long형 상수값을 8진수로 출력 : %o\n", lg_num);
    printf("부호가 없는 상수값을 16진수로 출력: %X\n", un_num);
    printf("long형 상수값을 16진수로 출력 : %X\n", lg_num);
    return 0;
}
```

```
부호가 없는 10진수 상수값 대입 : 12345U
long형 10진수 상수값 대입 : 1234567890L
부호가 없는 상수값을 10진수로 출력 : 12345
long형 상수값을 10진수로 출력 : 1234567890
부호가 없는 상수값을 8진수로 출력 : 30071
long형 상수값을 8진수로 출력 : 11145401322
부호가 없는 상수값을 16진수로 출력 : 3039
long형 상수값을 16진수로 출력 : 499602D2
```

2. 정수형 상수

■ 입력받은 정수형 상수를 10진수, 8진수, 16진수로 출력

예제 4-16 입력한 정수형 상수를 10진수, 8진수, 16진수로 출력 ex04_16.c

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int input_num;
06
07     printf("정수 입력 : ");
08     scanf_s("%d", &input_num);
09
10     printf("10진수로 출력 : %d\n", input_num);
11     printf("8진수로 출력 : %o\n", input_num);
12     printf("16진수로 출력 : %X\n", input_num);
13     printf("입력한 정수에 해당하는 문자 : %c\n", input_num);
14     return 0;
15 }
```

```
정수 입력 : 998
10진수로 출력 : 998
8진수로 출력 : 1746
16진수로 출력 : 3E6
입력한 정수에 해당하는 문자 : ?
```

C 언어에서 지원하는 아스키코드값은 0~127까지이므로 128부터는 ?가 출력됩니다.

수고하셨습니다.