

C프로그래밍

3 변수와 데이터 형식

01

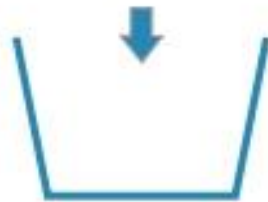
변수의 이해

변수 선언 (1/3)

- 요리를 하기에 앞서 그릇을 준비하듯이 C 프로그램을 작성하려면 변수 선언을 먼저 수행
 - ✓ 변수란? 값을 저장할 수 있는 메모리 공간에 붙인 이름 또는 공간 자체
 - ✓ 국그릇, 밥그릇, 반찬그릇 등 다양한 그릇이 있는 것처럼 변수의 종류도 다양
- 두 문장으로 다음 그림과 같이 새로운 변수(그릇) 2개를 생성
 - ✓ 소수점이 없는 값과 소수점이 있는 값을 담는 변수를 선언
 - ✓ 변수(그릇)에 각각 정수와 실수를 담을 수 있음

```
int a;  
float b;
```

정수를 담는 그릇



변수 a

실수를 담는 그릇(좀 더 크다)



변수 b

그림 3-6 정수형 변수와 실수형 변수의 개념

변수 선언 (2/3)

- 변수를 선언하는 방식은 다양

- ✓ 만약 정수형 변수 a와 b를 선언하고 싶다면 다음과 같은 방식을 사용할 수 있음
- ✓ 즉 변수의 종류가 같을 때는 변수를 개별적으로 선언해도 되고 콤마(,)를 사용하여 연속해서 선언해도 됨

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

==

```
int a, b;
```

변수 선언 (3/3)

- 정수형 변수 a, 실수형 변수 b, 정수형 변수 c, 실수형 변수 d를 선언하는 기본 방식

❶ 가능

```
int a;  
float b;  
int c;  
float d;
```

==

❷ 가능

```
int a, c;  
float b, d;
```

≠

❸ 불가능

```
int a, float b;  
int c, float d;
```

여기서 잠깐 세미콜론을 사용한 줄 표현

- 한 줄에 하나의 데이터 형식만 선언할 수 있다고 했으나 엄밀하게 말하면 '한 줄'이 아니라 '한 문장' 이라고 해야 옳음
- ❷는 올바른 형식이며 세미콜론(;)으로 구분된 것은 완전히 분리 된 문장으로 취급되므로 ❶과 ❷는 같은 의미

❶

```
int a;  
float b;  
int c;  
float d;
```

❷

```
int a; float b;  
int c; float d;
```

변수에 값을 담는 방법 (1/13)

- 기본적인 값의 대입

- 변수 a에 100을 대입하고 변수 b에 123.45를 대입 → 값을 저장



그림 3-7 정수형 변수와 실수형 변수에 값 대입

- 100은 정수, 123.45는 실수, 정수형 변수 a와 실수형 변수 b를 선언하고 변수에 값을 넣으려면 다음과 같이 나열

```
int a;  
float b;  
a = 100;  
b = 123.45;
```

==

```
int a = 100;  
float b = 123.45;
```

변수에 값을 담는 방법 (2/13)

- 기본적인 값의 대입

- 만약 변수 a, b가 모두 정수형 변수라면 형식이 동일하므로 한 줄로 해결할 수 있음

```
int a;  
int b;  
a = 100;  
b = 200;
```

==

```
int a = 100, b = 200;
```

- ✓ 변수에 값을 대입할 때는 지정된 데이터 형식만 대입해야 함

변수에 값을 담는 방법 (3/13)

- 기본적인 값의 대입

기본 3-7 변수에 값 대입 예

3-7.c

```
01 #include <stdio.h>
03 void main( )
04 {
05     int a;          ----- 정수형 변수 a를 선언한다.
06     float b;        ----- 실수형 변수 b를 선언한다.
07
08     a = 123.45;      ----- 정수형 변수에 실수를 대입한다. → 바람직하지 않다.
09     b = 200;         ----- 실수형 변수에 정수를 대입한다. → 바람직하지 않다.
10
11     printf("a의 값 ==> %d \n", a);
12     printf("b의 값 ==> %f \n", b);
13 }
```

실행 결과

a의 값 ==> 123

b의 값 ==> 200.000000

변수에 값을 담는 방법 (4/13)

- 기본적인 값의 대입

• 기본 3-7 결과 복기

- ✓ 실행은 되었지만 8행에서 정수형 변수 a에 실수 123.45를 대입해 결과가 123만 나왔음
- ✓ 8행에서 아래와 같은 처리가 이루어졌기 때문

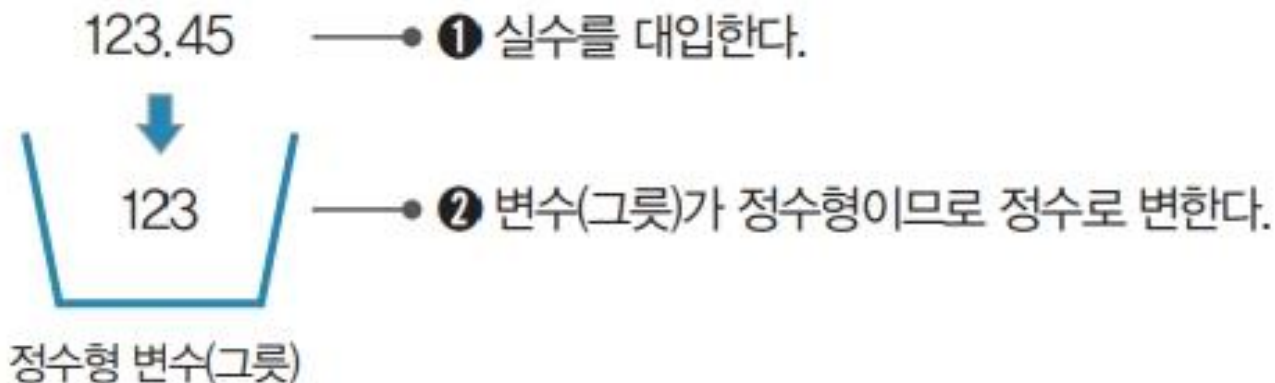


그림 3-8 정수형 변수에 실수 대입 시 처리 방식

- 실수(123.45)를 대입하더라도 그것을 담는 변수(그릇)가 정수형이므로 소수점 아래가 떨어져 나가고 정수(123)만 저장되어 결국 8행의 변수 a에는 123만 들어가게 됨

변수에 값을 담는 방법 (5/13)

- 기본적인 값의 대입

• 기본 3-7 결과 복기 (cont'd)

- ✓ 실수형 변수(그릇)에 정수를 담으면 아래와 같이 처리

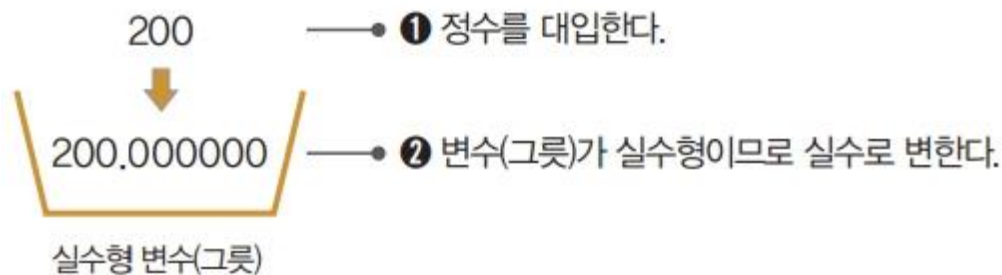


그림 3-9 실수형 변수에 정수 대입 시 처리 방식

- 대입된 정수(200)가 실수(200.000000)로 변함
- 200이나 200.00이나 정수 또는 실수라는 것만 다를 뿐 값의 크기는 차이가 없으므로 문제가 생기지 않음
- 하지만 변수의 데이터 형식과 실제 값의 종류가 다른 것은 그리 바람직하지 않으므로 정수형 변수에는 정수를 대입하고 실수형 변수에는 실수를 대입해야 함
- 즉 9행은 다음과 같이 고치는 것이 바람직함

```
b = 200.0;
```

변수에 값을 담는 방법 (6/13)

- 다양한 값의 대입 방법

응용 3-8 변수에 변수 대입 예 1

3-8.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int a, b;           —— 정수형 변수 2개를 선언한다.
06     float c, d;        —— 실수형 변수 2개를 선언한다.
07
08     a = 100;           —— a에 정수 100을 대입한다.
09     1                 —— b에 a 값을 대입한다.
10
11     c = 111.1f;        —— c에 실수 111.1을 대입한다.
12     2                 —— d에 c 값을 대입한다.
13
14     printf("a, b의 값 ==> %d , %d \n", a, b);
15     printf("c, d의 값 ==> %5.1f , %5.1f \n", c, d);
16 }
```

실행 결과

a, b의 값 ==> 100 , 100

c, d의 값 ==> 111.1 , 111.1

c = d 1 a = b 1

변수에 값을 담는 방법 (7/13)

- 다양한 값의 대입 방법

• 응용 3-8 결과 복기

- ✓ 8행에서 정수형 변수 a에 100을 대입
- ✓ 9행에서는 정수형 변수 b에 값 대신 a를 대입
 - 이때 9행은 다음과 같이 처리
 - 오른쪽에 있는 변수 a의 값인 100이 변수 b에 들어감
 - 결국 변수 a와 b는 같은 값인 100을 가지게 되므로 14행에서 같은 값을 출력

변수 a 값(100)만 뽑아서 변수 b에 대입한다.



그림 3-10 변수에 변수 대입 시 처리 방식

변수에 값을 담는 방법 (8/13)

- 다양한 값의 대입 방법

응용 3-9 변수에 변수 대입 예 2

3-9.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int a, b, c, d;
06
07     a = 100 + 100;      —— a에 두 수의 계산 결과를 대입한다.
08     b = a + 100;        —— b에 변수와 수의 계산 결과를 대입한다.
09     c = a + b - 100;    —— c에 변수와 수의 계산 결과를 대입한다.
10     1                  —— d에 a, b, c의 덧셈 결과를 대입한다.
11     printf("a, b, c, d 의 값 ==> %d, %d, %d, %d \n", a, b, c, d);
12
13     2                  —— a, b, c, d에 모두 100을 대입한다(한 문장으로 처리한다).
14     printf("a, b, c, d 의 값 ==> %d, %d, %d, %d \n", a, b, c, d);
15
16     a = 100;
17     a = a + 200;        —— a에 자신의 a 값과 200을 더한 값을 다시 대입한다.
18     printf("a 의 값 ==> %d \n", a);
19 }
```

실행 결과

a, b , c , d 의 값 ==> 200 , 300 , 400 , 900

a, b , c , d 의 값 ==> 100 , 100 , 100 , 100

a 의 값 ==> 300

변수에 값을 담는 방법 (9/13)

- 다양한 값의 대입 방법

• 응용 3-9 결과 복기

✓ 7행에서는 연산 결과를 변수에 대입

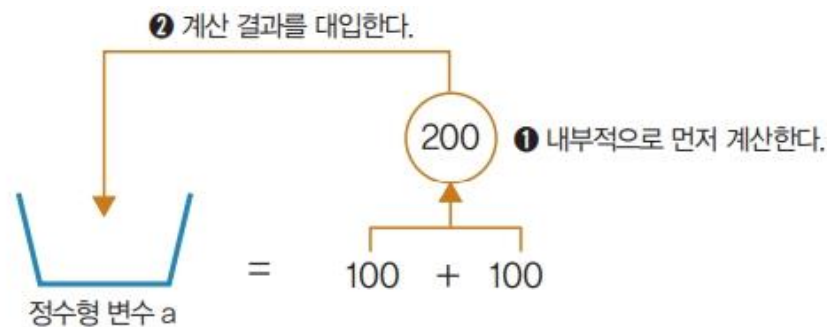


그림 3-11 숫자끼리의 계산 결과를 대입하는 방식

✓ 8행에서는 변수 a의 값과 100의 연산 결과를 변수 b에 대입

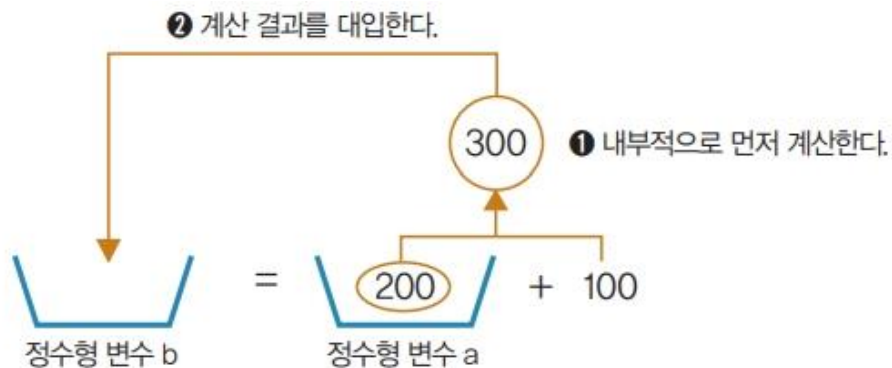


그림 3-12 변수와 숫자의 계산 결과를 대입하는 방식

변수에 값을 담는 방법 (10/13)

- 다양한 값의 대입 방법

• 응용 3-9 결과 복기 (cont'd)

✓ =는 맨 뒤부터 처리 되는데 결국 13행은 다음과 같이 풀어 쓸 수 있음

<code>a = b = c = d = 100;</code>	==	<code>d = 100; c = d; b = c; a = b;</code>
-----------------------------------	----	--

✓ 한 가지 유의할 점은 바로 전 단계(7~10행)의 a, b, c, d에 각각 200, 300, 400, 900 이라는 값이 들어 있지만 그 값들은 무시하고 새로운 값으로 덮어쓴다는 것

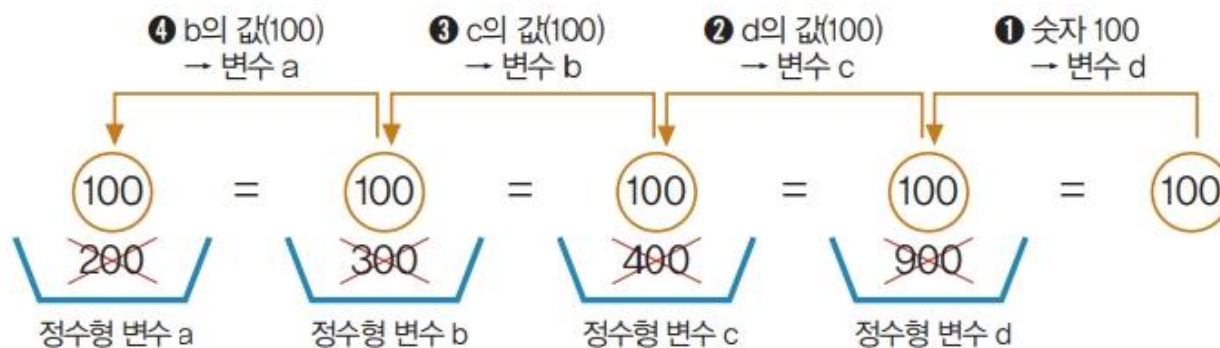


그림 3-13 연속된 값의 대입 방식

변수에 값을 담는 방법 (11/13)

- 다양한 값의 대입 방법

• 응용 3-9 결과 복기 (cont'd)

- ✓ 17행에서는 [그림 3-14]처럼 자신의 값으로 연산을 한 후 다시 자신에게 넣음

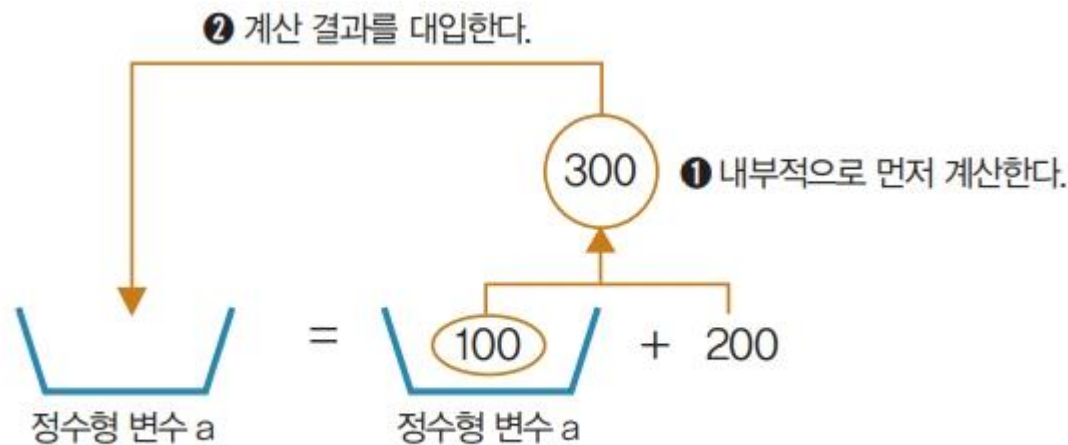


그림 3-14 자신의 값을 다시 계산 결과에 대입하는 방식

변수에 값을 담는 방법 (12/13)

- 대입 연산자와 변수의 위치

- **대입 연산자(=)를 사용하면 오른쪽의 것이 왼쪽에 대입**

- ✓ 대입 연산자(=)의 왼쪽에는 반드시 무엇을 담는 그릇인 변수만 온다는 것을 알 수 있음

10=100;



그림 3-15 값을 넣을 그릇이 없는 경우

변수에 값을 담는 방법 (13/13)

- 대입 연산자와 변수의 위치

- 대입 연산자(=)의 오른쪽에는 상수(숫자), 변수, 계산값이 올 수 있음
- 결론적으로 대입 연산자의 왼쪽에는 변수만 올 수 있고 오른쪽에는 상수, 변수, 계산 값, 함수 등 무엇이든지 올 수 있음

a=100;

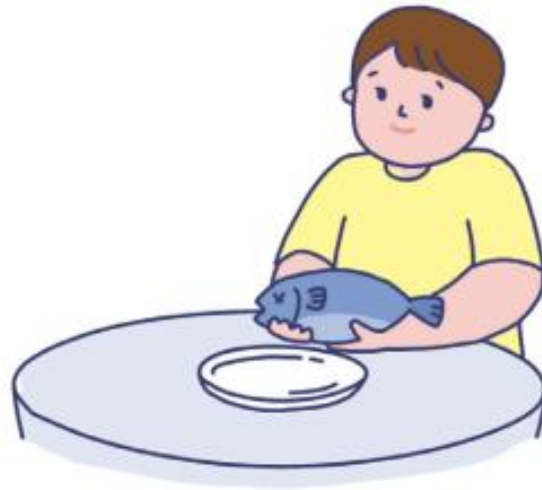


그림 3-16 값을 넣을 그릇이 있는 경우

02

데이터 형식 (1/2)

비트, 바이트, 진수 (1/3)



- 비트

- **비트(bit)는 전기 스위치와 비슷한 개념**

- ✓ 전기 스위치에는 OFF와 ON만 있듯이 비트에도 0(OFF)과 1(ON)만 있음
- ✓ 1비트라면 전기 스위치가 1개이고 2비트라면 전기 스위치가 2개인 것
- ✓ 전기 스위치 2개(2비트)로 표현할 수 있는 가짓수는 4개이고 이를 2진수로 표현하면 각각 00, 01, 10, 11, 이를 10진수로 나타내면 0, 1, 2, 3

전기 스위치 n개로 표현할 수 있는 가짓수 = 2^n

표 3-3 전기 스위치로 표현 가능한 가짓수

전기 스위치	의미	2진수(0, 1)	10진수
	꺼짐, 꺼짐	00	0
	꺼짐, 켜짐	01	1
	켜짐, 꺼짐	10	2
	켜짐, 켜짐	11	3

비트, 바이트, 진수 (2/3)

- 진수

- 10진수는 숫자 10개(0~9)로 모든 숫자를 표현
- 2진수는 숫자 2개(0, 1)로만 모든 수를 표현

표 3-4 10진수, 2진수, 16진수로 나타낸 0~15

10진수(0~9)	2진수(0, 1)	16진수(0~F)
00	0000	0
01	0001	1
02	0010	2
03	0011	3
04	0100	4
05	0101	5
06	0110	6
07	0111	7
08	1000	8
09	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

✓ 16진수가 필요한 이유 : 2진수의 네 자리와 16진수의 한 자리가 맞아 떨어짐

비트, 바이트, 진수 (3/3)

- 바이트

- 비트와 더불어 C에서 가장 많이 사용되는 단위는 바이트(byte)

표 3-5 비트와 바이트의 크기에 따른 숫자의 범위

비트 수	바이트 수	표현 개수	2진수	10진수	16진수
1		$2^1=2$	0~1	0~1	0~1
2		$2^2=4$	0~11	0~3	0~3
4		$2^4=16$	0~1111	0~15	0~F
8	1	$2^8=256$	0~11111111	0~255	0~FF
16	2	$2^{16}=65536$	0~11111111 11111111	0~65535	0~FFFF
32	4	$2^{32}=\text{약 } 42\text{억}$	0~...	0~약 42억	0~FFFF FFFF
64	8	$2^{64}=\text{약 } 1800\text{경}$	0~...	0~약 1800경	0~...

진수 변환 (1/5)

- 10진수 변환

- 2진수 1001 0011을 10진수로 변환하는 과정

1	0	0	1			0	0	1	1	→ 2진수
×	×	×	×			×	×	×	×	
2^7	2^6	2^5	2^4			2^3	2^2	2^1	2^0	
128	0	0	16			0	0	2	1	→ 10진수
+										
147										→ 10진수

그림 3-17 2진수를 10진수로 변환하는 방법

진수 변환 (2/5)

- 10진수 변환

- 2진수 1001 0011을 16진수로 변환 후 10진수로 변환하는 과정

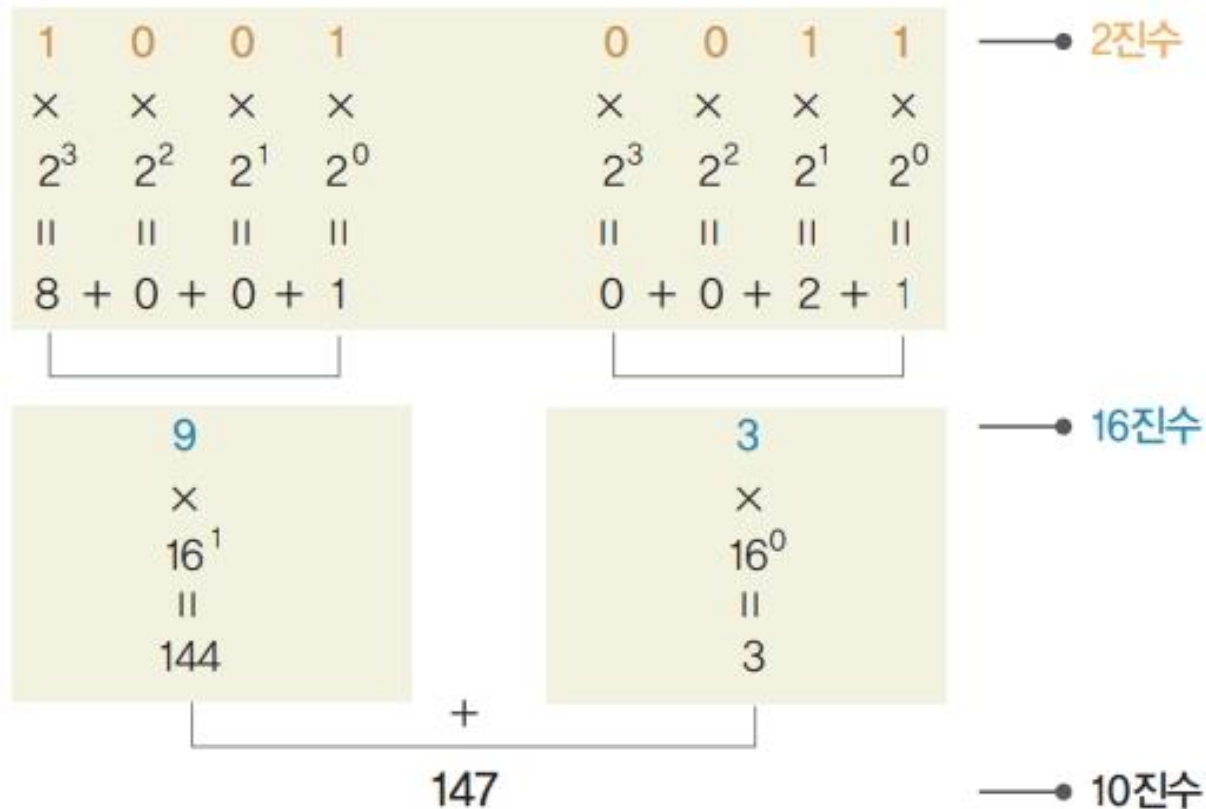


그림 3-18 2진수를 16진수로 변환한 후 10진수로 변환하는 방법

진수 변환 (3/5)

- 2진수 변환

• 10진수 13을 2진수로 변환하는 과정

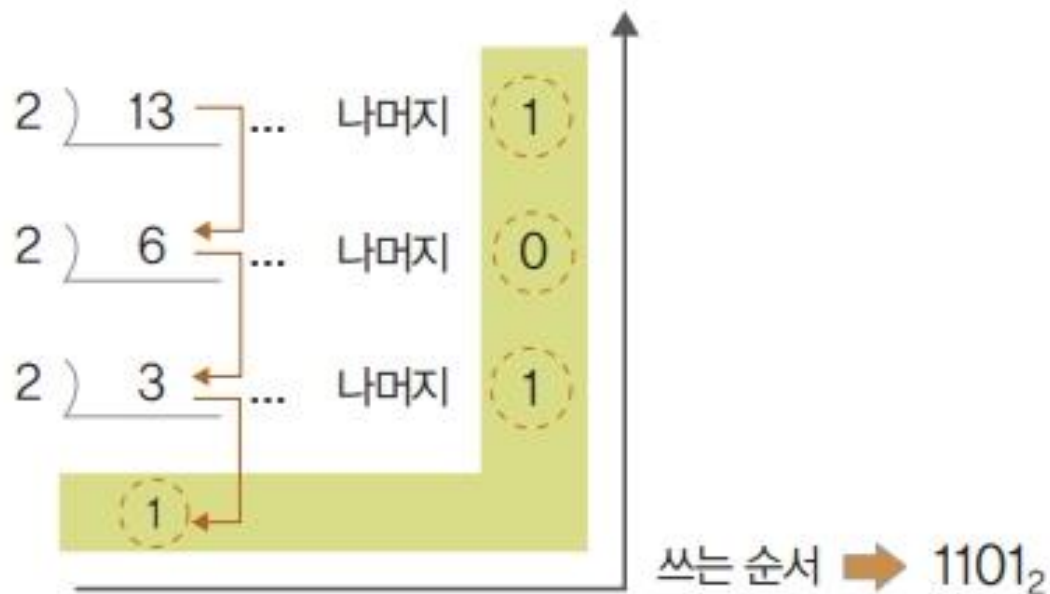


그림 3-19 10진수를 2진수로 변환하는 방법

- ✓ 처음에 13을 2로 나누면 몫은 6, 나머지는 1이고, 6을 다시 2로 나누면 몫은 3, 나머지는 0
- ✓ 다시 3을 2로 나누면 몫은 1, 나머지는 1
- ✓ 이렇게 해서 나온 마지막 몫과 나머지값들을 나열하면 2진수가 됨

진수 변환 (4/5)

- 2진수 변환

• 16진수 13을 2진수로 변환하는 과정

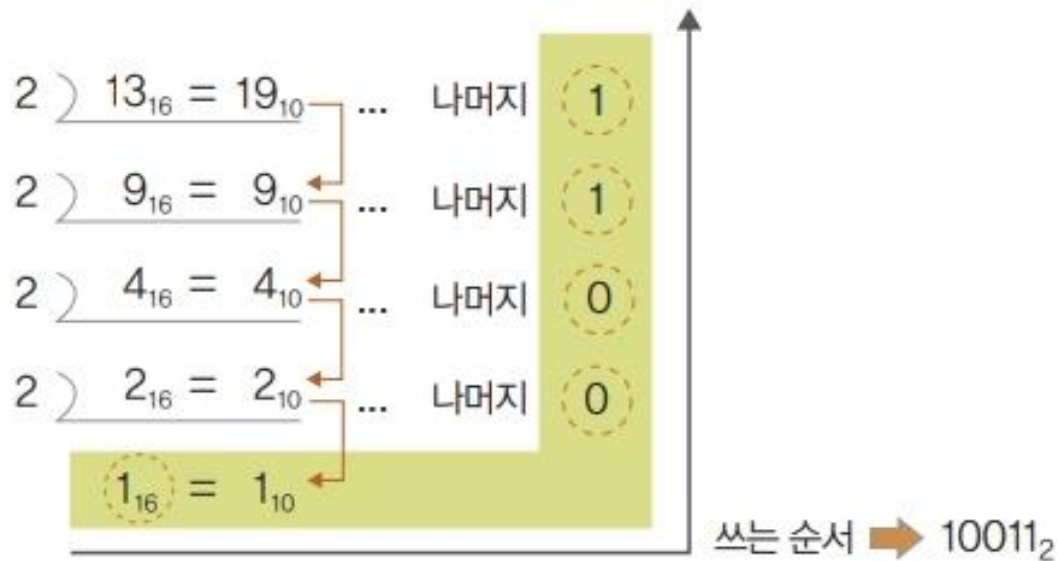


그림 3-20 16진수를 2진수로 변환하는 방법

- ✓ 13_{16} 을 10진수로 바꾸면 19가 되고 이것을 2로 나누면 몫은 9, 나머지는 1
- ✓ 이후 10진수와 동일하게 계산하면 13_{16} 은 2진수 10011_2 이 됨

진수 변환 (5/5)

- 2진수 변환

• 16진수를 2진수로 변환하는 간편한 방법

표 3-6 16진수와 2진수

16진수	2진수	16진수	2진수
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

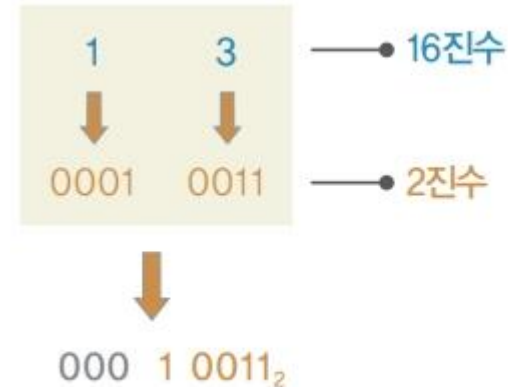


그림 3-21 16진수를 2진수로 변환하는 간편한 방법 1

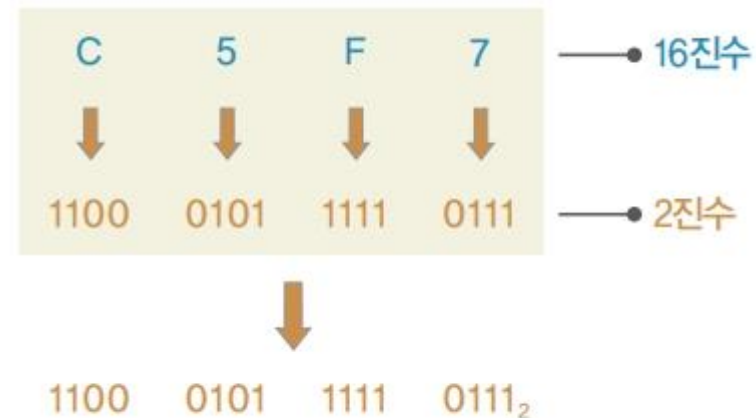


그림 3-22 16진수를 2진수로 변환하는 간편한 방법 2

03

데이터 형식 (2/2)

〈기본 자료형〉

숫자형 데이터 형식 (1/5)

- 소수점이 없는 정수형

- **정수형 : 소수점이 없는 데이터를 입력하기 위해 사용하는 데이터 형식**
 - ✓ unsigned가 붙은 데이터 형식은 마이너스(-) 값이 없는 경우에 사용
 - ✓ int와 long int의 차이점은?
 - ✓ 8byte int를 위해 long long int를 사용할 수도 있음 (C99 표준 이후에 추가)

표 3-7 정수형 데이터 형식

데이터 형식	의미	크기	값의 범위
short	작은 정수형	2바이트	$-2^{15}(-32768) \sim 2^{15}-1(32767)$
unsigned short	부호 없는 작은 정수형	2바이트	$0 \sim 2^{16}-1(65535)$
int	정수형	4바이트	$-2^{31}(\text{약 } -21\text{억}) \sim 2^{31}-1(\text{약 } 21\text{억})$
unsigned int	부호 없는 정수형	4바이트	$0 \sim 2^{32}-1(\text{약 } 42\text{억})$
long int(또는 long)	큰 정수형	4바이트	$-2^{31} \sim 2^{31}-1$
unsigned long	부호 없는 큰 정수형	4바이트	$0 \sim 2^{32}-1$

숫자형 데이터 형식 (2/5)

- 소수점이 없는 정수형

기본 3-10 소수점이 없는 정수형 사용 예

3-10.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int a=100, b=200;  —— 정수형 변수 a와 b에 값을 지정한다.
06     float result;      —— 실수형 변수 result를 선언한다.
07
08     result = a / b;     —— a를 b로 나눈 결과를 실수형 변수 result에 대입한다.
09
10     printf("%f \n", result);
11 }
```

실행 결과

0.000000

숫자형 데이터 형식 (3/5)

- 소수점이 없는 정수형

- 기본 3-10 복기

- ✓ 100을 200으로 나눈 결과는 0.5
- ✓ 8행의 a/b 는 $100/200$ 이고 100과 200은 모두 정수
- ✓ 그 결과도 실수 0.5가 아니라 소수 부분이 떨어진 정수 0 이 되어 8행의 result에 0의 실수 값인 0.000000이 저장

정수 +, -, *, / 정수 = 정수

정수 +, -, *, / 실수 = 실수

실수 +, -, *, / 실수 = 실수

숫자형 데이터 형식 (4/5)

- 소수점이 있는 정수형

- **float** 형은 대개 소수 아래 일곱 자리까지의 정밀도를 나타냄
- **double** 형은 소수점 아래 열여섯 자리 정도까지의 정밀도를 나타낼 수 있음

표 3-8 실수형 데이터 형식

데이터 형식	의미	크기	값의 범위
float	실수형	4바이트	약 $-3.4 \times 10^{38} \sim 3.4 \times 10^{38}$
double	큰 실수형	8바이트	약 $-1.79 \times 10^{308} \sim 1.79 \times 10^{308}$
long double	큰 실수형	8바이트	약 $-1.79 \times 10^{308} \sim 1.79 \times 10^{308}$

숫자형 데이터 형식 (5/5)

- 소수점이 있는 정수형

기본 3-11 소수점이 있는 실수형 사용 예

3-11.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     float a = 0.1234567890123456789012345f;
06     double b = 0.1234567890123456789012345;
07
08     printf("%30.25f \n", a);
09     printf("%30.25lf \n", b);
10 }
```

float 형 변수 a에 정밀도 스물다섯 자리 실수를 입력한다(맨 뒤의 f는 빼도 된다).

double 형 변수 b에 소수점 아래 스물다섯 자리 실수를 입력한다.

a와 b를 소수점 아래 스물다섯 자리까지 출력하는데 float는 %f로, double은 %lf로 출력한다.

실행 결과

```
0.1234567910432815551757812
0.1234567890123456773698862
```

문자형 데이터 (1/4)

- 아스키코드

- 컴퓨터에서 표현하는 문자(특히 키보드에 있는 영문자, 기호, 숫자 등)를 0~127에 대응한 코드

표 3-9 아스키코드

아스키코드	10진수	16진수
0 ~ 9	48 ~ 57	0x30 ~ 0x39
A ~ Z	65 ~ 90	0x41 ~ 0x5A
a ~ z	97 ~ 122	0x61 ~ 0x7A

- C에서는 숫자를 문자로도 표현

```
char ch = 'a';
```

==

```
char ch = 97;
```

문자형 데이터 (2/4)

- 한 글자를 뜻하는 문자형

표 3-10 문자형 데이터 형식

데이터 형식	의미	크기	값의 범위
char	문자형 또는 정수형	1바이트	$-2^7(-128) \sim 2^7-1(127)$
unsigned char	문자형 또는 부호 없는 정수형	1바이트	$0 \sim 2^8-1(255)$

- **char 형에는 문자 뿐만 아니라 값의 범위에 해당하는 정수를 대입할 수 있음**
 - ✓ char 형을 1바이트 크기의 정수형으로 취급해도 상관없음
 - ✓ char 형의 크기는 1바이트(8비트)이므로 표현할 수 있는 글자 수는 256가지이며 값의 범위는 -128~127
 - ✓ 아스키코드의 0~127을 담을 수 있음

문자형 데이터 (3/4)

- 한 글자를 뜻하는 문자형

기본 3-12 문자형 변수 사용 예 1

3-12.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     char a, b, c;           —— 문자형 변수 3개를 선언한다.
06
07     a = 'A';               —— 문자형 변수 a에 'A'를 대입한다.
08
09     printf(" %c \n", a);    —— 문자형 변수 a를 문자형과 정수형으로 출력한다.
10     printf(" %d \n", a);
11
12     b = 'a';               —— 문자형 변수 b에 'a'를 대입한다.
13     c = b + 5;            —— 문자형 변수 b에 5를 더해서 문자형 변수 c에 대입한다.
14     printf(" %c \n", b);
15     printf(" %c \n", c);
16
17     c = 90;               —— 문자형 변수 c에 90을 대입한다.
18     printf(" %c \n", c);
19 }
```

실행 결과

A
65
a
f
Z

문자형 데이터 (4/4)

- 한 글자를 뜻하는 문자형

응용 3-13 문자형 변수 사용 예 2

3-13.c

실행 결과

P

A

#의 ASCII 값은 35 입니다

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int a, b;
06     char c, d;
07
08     a = 0x41;      —— 정수형 변수 a, b에 16진수 0x41과 0x50을 대입한다.
09     b = 0x50;
10
11     1             —— 정수형 변수 b를 문자형으로 출력한다.
12
13     c = a;         —— 문자형 변수 c에 정수형 변수 a 값을 대입한다.
14     2             —— 문자형 변수 c를 문자형으로 출력한다.
15
16     d = '#';      —— 문자형 변수 d에 '#'를 대입한다.
17     printf("%c의 ASCII 값은 %d 입니다 \n", d, d); —— 문자형 변수 d를 두 가지 형태로 출력한다.
18 }
```

실행 결과: 1 printf("%c \n", b); 2 printf("%d \n", c);

02

산술 연산자

기본 산술 연산자 (1/2)

연산자	설명	사용 예	사용 예에 대한 설명
=	대입 연산자	$a = 3$	정수 3을 a에 대입한다.
+	더하기	$a = 5 + 3$	정수 5와 3을 더한 값을 a에 대입한다.
-	빼기	$a = 5 - 3$	정수 5와 3을 뺀 값을 a에 대입한다.
*	곱하기	$a = 5 * 3$	정수 5와 3을 곱한 값을 a에 대입한다.
/	나누기	$a = 5 / 3$	정수 5를 3으로 나눈 값을 a에 대입한다.
%	나머지 값	$a = 5 \% 3$	정수 5를 3으로 나눈 뒤 나머지 값을 a에 대입한다.

기본 산술 연산자 (2/2)

기본 4-1 산술 연산자 사용 예

4-1.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int a, b = 5, c = 3;
06
07     a = b + c;      ----- 더하기 연산을 해서 a에 대입한다.
08     printf(" %d + %d = %d \n", b, c, a);
09
10     a = b - c;      ----- 빼기 연산을 해서 a에 대입한다.
11     printf(" %d - %d = %d \n", b, c, a);
12
13     a = b * c;      ----- 곱하기 연산을 해서 a에 대입한다.
14     printf(" %d * %d = %d \n", b, c, a);
15
16     a = b / c;      ----- 나누기 연산을 해서 a에 대입한다.
17     printf(" %d / %d = %d \n", b, c, a);
18
19     a = b % c;      ----- 나머지값 연산을 해서 a에 대입한다.
20     printf(" %d %d = %d \n", b, c, a);
21 }
```

실행 결과

$$5 + 3 = 8$$

$$5 - 3 = 2$$

$$5 * 3 = 15$$

$$5 / 3 = 1$$

$$5 \% 3 = 2$$

연산자 우선순위와 강제 형 변환 (1/5)

응용 4-2 연산자 우선순위와 강제 형 변환 예

4-2.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int a = 2, b = 3, c = 4;          —— 정수형 변수를 선언한다.
06     int result1, mok, namugi;
07     float result2;                   —— 실수형 변수를 선언한다.
08
09     result1 = a + b - c;              —— 더하기와 빼기 연산을 동시에 수행한다.
10     printf(" %d + %d - %d = %d \n", a, b, c, result1);
11
12     result1 = a + b * c;              —— 더하기와 곱하기 연산을 동시에 수행한다.
13     printf(" %d + %d * %d = %d \n", a, b, c, result1);
14
15     result2 = a * b / (float) c;      —— 정수 c를 실수로 강제 형 변환한 후 연산한다.
16     printf(" %d * %d / %d = %f \n", a, b, c, result2);
17
18     1 = c / b;                       —— 몫을 구한다.
19     printf(" %d / %d 의 몫은 %d \n", c, b, mok);
20
21     2 = c % b;                       —— 나머지를 구한다.
22     printf(" %d %d 의 나머지는 %d \n", c, b, namugi);
23 }
```

실행 결과

$$2 + 3 - 4 = 1$$

$$2 + 3 * 4 = 14$$

$$2 * 3 / 4 = 1.500000$$

$$4 / 3 \text{ 의 몫은 } 1$$

$$4 / 3 \text{ 의 나머지는 } 1$$

연산자 우선순위와 강제 형 변환 (2/5)

- 간단한 연산자 우선순위

• 응용 4-2 복기

✓ $\text{result1} = a + b - c;$

❶ $\text{result1} = (a + b) - c;$

❷ $\text{result1} = a + (b - c);$

✓ 답은 ❶이지만 덧셈과 뺄셈의 경우에는 어떤 것을 먼저 계산하든 결과가 같음

✓ 괄호가 없을 때는 왼쪽에서 오른쪽 방향으로 계산

✓ 덧셈과 곱셈이 같이 있는 12행을 보면 계산 순위에 따라 결과가 다르게 나옴

여기서 잠깐 괄호를 사용한 연산자 우선순위

- 덧셈, 뺄셈, 곱셈, 나눗셈이 함께 나와 연산자 우선순위가 혼란스러울 때는 괄호를 사용
- 다음 ❶과 ❷은 동일한 결과를 출력하지만 ❷가 더 나은 코딩이라고 할 수 있음
- ❶ $a = b + c * d;$
- ❷ $a = b + (c * d);$

연산자 우선순위와 강제 형 변환 (3/5)

- 데이터 형식의 강제 형 변환

- **대입연산에서 자동 형 변환 (auto casting)**

- ✓ `double num1 = 245; // int형 정수 245를 double형으로 자동 형 변환`
- ✓ 데이터 표현범위가 보다 넓은 데이터 형식으로 변환 → 데이터 손실 X
- ✓ 데이터 표현범위가 보다 좁은 데이터 형식으로 변환 → 데이터 손실 O
 - `int num2 = 1.5; // num2에는 1이 저장됨`
 - `char num3 = 129; // num3에는 -127이 저장됨`

연산자 우선순위와 강제 형 변환 (4/5)

- 데이터 형식의 강제 형 변환

- 정수의 승격에 의한 자동 형 변환

- ✓ int보다 작은 크기의 정수형 데이터는 int형 데이터로 형 변환이 되어 연산 진행

- 피연산자의 데이터 형식 불일치로 발생하는 자동 형 변환

- ✓ `Double num1 = 5.15 + 19;` //정수 19는 실수 19.0로 자동 형 변환

- ✓ 형 변환 규칙

`Int` → `long` → `long long` → `float` → `double` → `long double`

연산자 우선순위와 강제 형 변환 (5/5)

- 데이터 형식의 강제 형 변환

• 명시적 형 변환 : 강제 형 변환

- ✓ `Double num1 = 5.15 + (double)19; //`19는 19.000000로 강제 형 변환
- ✓ 응용 4-2 복기

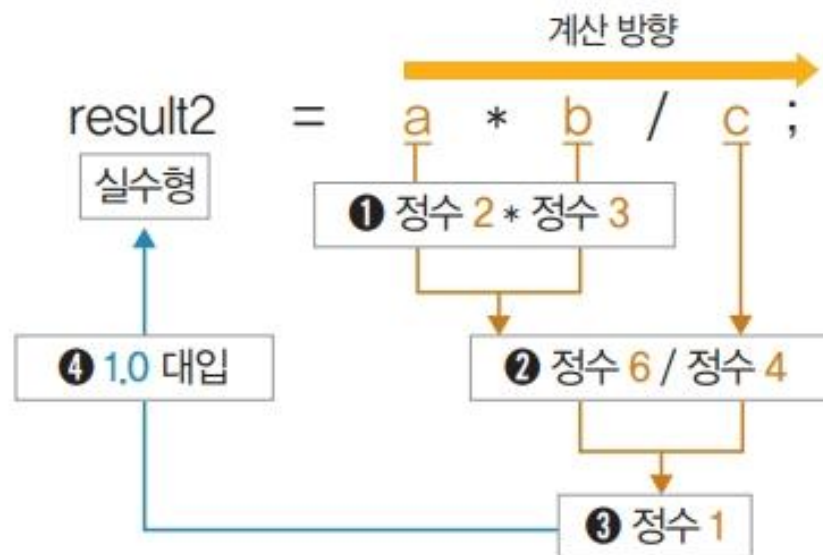


그림 4-1 강제 형 변환을 하지 않았을 때의 결과

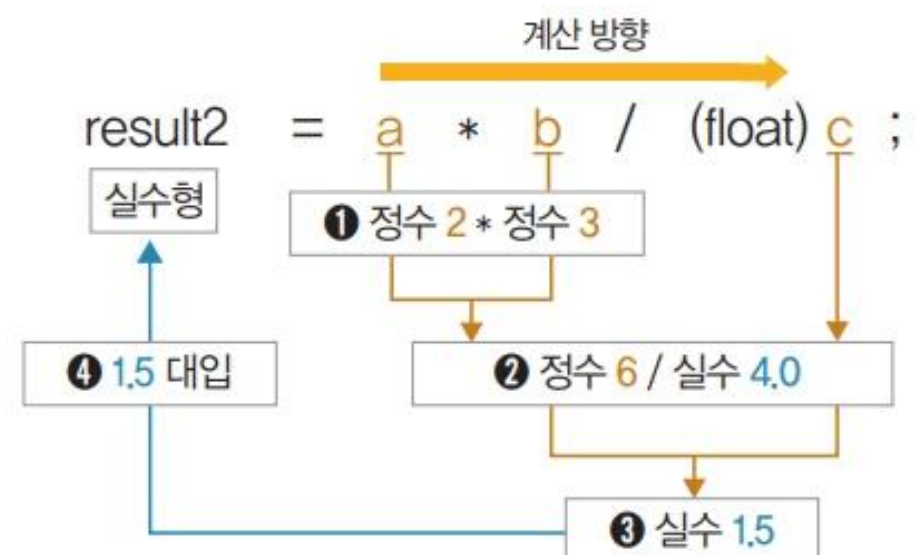


그림 4-2 강제 형 변환을 했을 때의 결과

대입 연산자와 증감 연산자 (1/4)

• 대입 연산자

- ✓ C에서는 대입 연산자 = 외에도 +=, -=, *=, /=, %=를 사용할 수 있음

• 증감 연산자

- ✓ 값을 1씩 증가 시키는 역할을 하는 ++ 연산자와 1씩 감소시키는 역할을 하는 -- 연산자

표 4-2 대입 연산자와 증감 연산자

연산자	명칭	사용 예	설명
+=	대입 연산자	a += 3	a = a + 3과 동일하다.
-=	대입 연산자	a -= 3	a = a - 3과 동일하다.
*=	대입 연산자	a *= 3	a = a * 3과 동일하다.
/=	대입 연산자	a /= 3	a = a / 3과 동일하다.
%=	대입 연산자	a %= 3	a = a % 3과 동일하다.
++	증가 연산자	a++ 또는 ++a	a += 1 또는 a = a + 1과 동일하다.
--	감소 연산자	a-- 또는 --a	a -= 1 또는 a = a - 1과 동일하다.

대입 연산자와 증감 연산자 (2/4)

기본 4-3 대입 연산자와 증감 연산자 사용 예

4-3.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int a = 10;
06
07     a++;           ----- a = a + 1과 동일하다.
08     printf(" a ++ ==> %d \n", a);
09
10     a--;           ----- a = a - 1과 동일하다.
11     printf(" a -- ==> %d \n", a);
12
13     a += 5;        ----- a = a + 5와 동일하다.
14     printf(" a += 5 ==> %d \n", a);
15
16     a -= 5;        ----- a = a - 5와 동일하다.
17     printf(" a -= 5 ==> %d \n", a);
18
```

대입 연산자와 증감 연산자 (3/4)

```
19  a *= 5;      ----- a = a * 5와 동일하다.
20  printf("a *= 5 ==> %d \n", a);
21
22  a /= 5;      ----- a = a / 5와 동일하다.
23  printf("a /= 5 ==> %d \n", a);
24
25  a %= 5;      ----- a = a % 5와 동일하다.
26  printf("a %= 5 ==> %d \n", a);
27 }
```

실행 결과

```
a ++ ==> 11
a -- ==> 10
a += 5 ==> 15
a -= 5 ==> 10
a *= 5 ==> 50
a /= 5 ==> 10
a %= 5 ==> 0
```


대입 연산자와 증감 연산자 (4/4)

- **a++** : a가 있고 a 값을 1 증가
- **++a** : a 값을 1 증가시키고 a가 있음

응용 4-4 증감 연산자 사용 예

4-4.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     int a = 10, b;
06
07     b = a++;          ----- b=a를 수행한 후 a를 1 증가시킨다.
08     printf(" %d \n", b);
09
10     b = ++a;          ----- a를 1 증가시킨 후 b=a를 수행한다.
11     printf(" %d \n", b);
12 }
```

:e++ = q 1 10

실행 결과

10
12

예 제

[예제 01] 정수형을 출력하는 프로그램

예제 설명 정수를 하나 입력받아 10진수, 16진수, 8진수로 출력하는 프로그램이다.

실행 결과

정수를 입력하세요 ==> 9999

10진수 ==> 9999

16진수 ==> 270F

8진수 ==> 23417

[예제 01] 정수형을 출력하는 프로그램

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     int data;
06
07     printf("정수를 입력하세요 ==> ");
08     scanf("%d", &data);
09
10
11
12
13 }
```

—— 정수형 변수를 선언한다.

—— 키보드로 정수를 입력받는다.

—— 10진수(%d), 16진수(%X), 8진수(%o)를 출력한다.

[예제 02] 입력하는 정수의 진수 결정

예제 설명 10진수, 16진수, 8진수 중 어떤 진수의 값을 입력받을지 결정하고, 입력받은 수를 10진수, 16진수, 8진수로 출력하는 프로그램이다.

실행 결과

입력진수 결정 <1>10 <2>16 <3>8 : 2

값 입력 : FF

10진수 ==> 255

16진수 ==> FF

8진수 ==> 377

[예제 02] 입력하는 정수의 진수 결정

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     int type, data;
06
07     printf("입력진수 결정 <1>10 <2>16 <3>8 : ");
08     scanf("%d", &type);
09
10     printf("값 입력 : ");
11
12     if(type == 1)
13     { scanf("%d", &data); }
14
15     if(type == 2)
16     { }
17
18     if(type == 3)
19     { }
20
21     //
22     //
23     //
24 }
```

—— 키보드로 1~3 중 하나를 입력받는다.

—— 입력값이 1이면 10진수를 입력받는다.

—— 입력값이 2이면 16진수를 입력받는다.

—— 입력값이 3이면 8진수를 입력받는다.

—— 입력받은 data 값을 10진수, 16진수, 8진수로 변환하여 출력한다.

[예제 03] 데이터 형의 크기 확인

sizeof() : 연산자

예제 설명 sizeof() 함수를 사용해서 각 데이터형의 크기를 확인하는 프로그램이다.

실행 결과

int 형의 크기	=> 4
unsigned int 형의 크기	=> 4
short 형의 크기	=> 2
unsigned short 형의 크기	=> 2
long int 형의 크기	=> 4
unsigned long int 형의 크기	=> 4
float 형의 크기	=> 4
double 형의 크기	=> 8
long double 형의 크기	=> 8
char 형의 크기	=> 1
unsigned char 형의 크기	=> 1

[예제 03] 데이터 형의 크기 확인

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     printf("int 형의 크기\t\t\t ==> %d\n", sizeof(int));
06     printf("unsigned int 형의 크기\t\t ==> %d\n",      );
07     printf("short 형의 크기\t\t\t\t ==> %d\n",      );
08     printf("unsigned short 형의 크기\t ==> %d\n", sizeof(unsigned short));
09     printf("long int 형의 크기\t\t\t ==> %d\n",      );
10     printf("unsigned long int 형의 크기\t ==>
        %d\n", sizeof(unsigned long int));
11     printf("float 형의 크기\t\t\t\t ==> %d\n",      );
12     printf("double 형의 크기\t\t\t ==> %d\n",      );
13     printf("long double 형의 크기\t\t ==> %d\n",      );
14     printf("char 형의 크기\t\t\t\t ==> %d\n", sizeof(char));
15     printf("unsigned char 형의 크기\t\t ==> %d\n",      );
16 }
```

—— sizeof() 함수로 각 데이터형의 크기(바이트 수)를 출력한다. 이때 컴파일러에 따라서 long double 형은 16바이트 크기일 수도 있다.

Q & A