

# C 프로그래밍 실습

Practice of C programming language

## 4주차

송인석 박사과정 M514

[sis5041@naver.com](mailto:sis5041@naver.com)

010-6517-4500

# 수업 내용

## 1. 연산자와 연산식

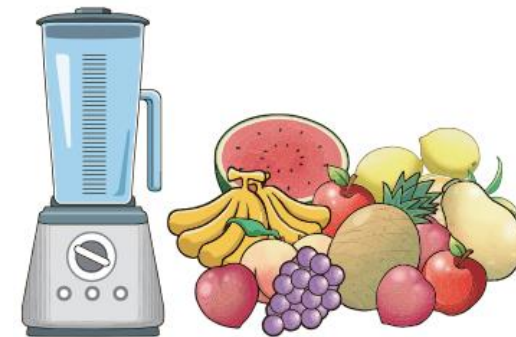
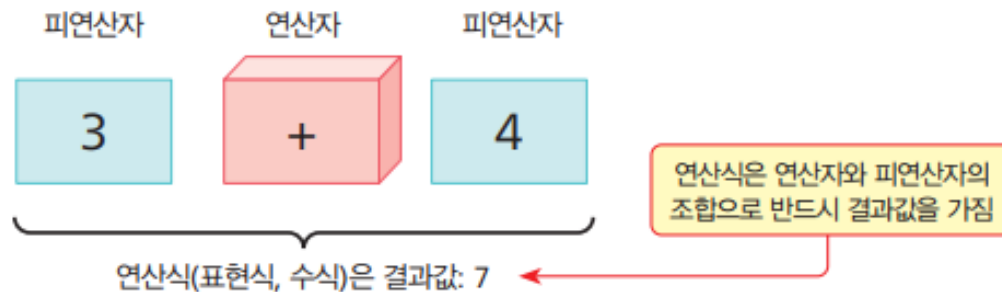
1. 연산식과 다양한 연산자
2. 관계와 논리, 조건, 비트연산자
3. 형변환 연산자와 연산자 우선순위

## 2. Quiz

# 1. 연산자와 연산식

# 연산자와 피연산자, 연산식과 연산값

- 연산식(expression)
  - 변수와 다양한 리터럴 상수 그리고 함수의 호출 등으로 구성되는 표현식
  - 연산식은 항상 하나의 결과값을 가짐
- 연산자(operator)
  - $+$   $-$   $*$   $/$ 
    - 이미 정의된 연산을 수행하는 문자 또는 문자조합 기호
- 피연산자(operand)
  - 연산(operation)에 참여하는 변수나 상수



# 다양한 연산자

- 단(일)항(unary), 이항(binary), 삼항(ternary) 연산자
  - 연산에 참여하는 피연산자(operand)의 개수
  - 삼항연산자
    - 조건연산자 '?' :가 유일
- 단항연산자는 연산자의 위치에 따라
  - ++a : 전위
    - 연산자가 앞에 있으면 전위(prefix) 연산자
  - a++ : 후위
    - 연산자가 뒤에 있으면 후위(postfix) 연산자

## 단항연산자

**연산자** **피연산자** (전위: prefix)

++a: 전위 증가연산자  
sizeof (int): sizeof 연산자  
(int) 3.46: 자료형 변환연산자  
-3: 부호연산자

**피연산자** **연산자** (후위: postfix)

b--: 후위 감소연산자  
a++: 후위 증가연산자



## 이항연산자

**피연산자 1** **연산자** **피연산자 2**

a % 3  
3 && 4  
5 >= 7

## 삼항연산자

**피연산자 1** **?** **피연산자 2** **:** **피연산자 3**

3 ? 4 : 5  
(5 >= 7) ? (3+5) : (10/2)

# 곱하기와 나누기, 나머지 연산자 활용

- 나머지 연산식  $a \% b$ 
  - $a$ 를  $b$ 로 나눈 나머지 값

$a \% b$  연산값:  $r$        $a / b$  연산값:  $n$

$$\begin{array}{r} n \\ b \overline{) a} \\ \underline{n \cdot b} \\ r \end{array}$$

$$\begin{array}{r} 3 \\ 3 \overline{) 10} \\ \underline{9} \\ 1 \end{array}$$

$a$ 를  $b$ 로 나눈 나머진  $r$ 과 몫인  $n$

$$a = b * n + r$$

10 % 3 결과: 1      10 / 3 결과: 3

실습예제 5-1	Prj01	01arithop.c	곱하기와 나누기, 나머지 연산자 활용	난이도: ★
	01	#include <stdio.h>		
	02			
	03	int main(void)		
	04	{		
	05	int amount = 4000 * 3 + 10000;		
	06			
	07	printf("총금액 %d 원\n", amount);		
	08	printf("오천원권 %d 개\n", amount / 5000);		
	09	printf("천원권 %d 개\n", (amount % 5000) / 1000);		
	10			
	11	return 0;		
	12	}		
결과	총금액 22000 원 오천원권 4 개 천원권 2 개			

# 대입연산자 =

- 대입연산자(assignment operator) =
  - 오른쪽 연산식 결과값을 왼쪽 변수에 저장하는 연산자
    - 왼쪽 부분에는 반드시 하나의 변수만이 올 수 있음
  - l-value, r-value

실습예제 5-2

Prj02 02assignop.c 대입 연산자 활용 난이도: ★

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03
04 int main(void)
05 {
06     int cred2, cred3;
07
08     cred2 = cred3 = 0;
09
10     printf("2학점과 3학점의 수강 과목 수를 각각 입력 >> ");
11     scanf("%d %d", &cred2, &cred3);
12
13     cred2 = 2 * cred2;
14     printf("2학점 과목 총 학점: %d\n", cred2);
15     printf("3학점 과목 총 학점: %d\n", cred3 = 3 * cred3);
16     printf("총 학점: %d\n", cred2 + cred3);
17
18     return 0;
19 }
```

var

=

exp

왼쪽은 반드시 값을 저장할 수 있는 변수 이어야 한다.

오른쪽은 연산식으로 결과값이 왼쪽 변수에 저장

n = 2;

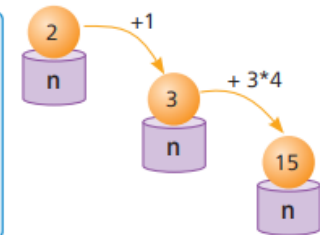
n = n + 1;

n = n + 3 \* 4;

대입 연산식의 연산값은 대입된 저장 값

결과

2학점과 3학점의 수강 과목 수를 각각 입력 >> 2 4  
2학점 과목 총 학점: 4  
3학점 과목 총 학점: 12  
총 학점: 16



# 축약 대입연산자

- `+=` `-=` `*=` `/=` `%=`

실습예제 5-3

Prj03

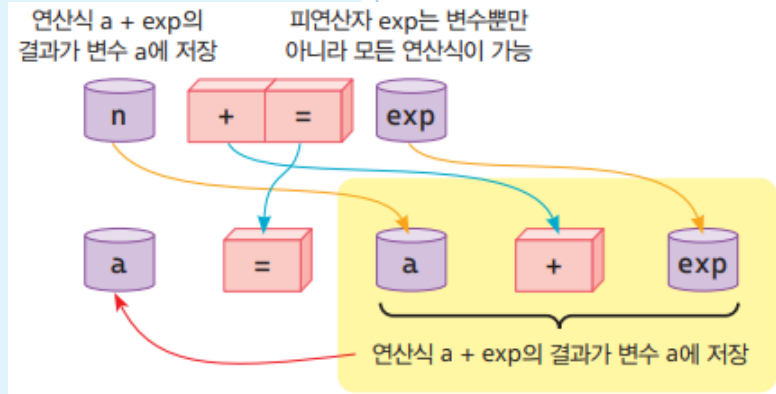
03compoundassign.c

복합 대입연산자 활용

난이도: ★

```
01 #define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의
02 #include <stdio.h>
03
04 int main(void)
05 {
06     int x = 0, y = 0;
07
08     printf("두 정수를 입력 >> ", &x, &y);
09     scanf("%d%d", &x, &y);
10
11     printf("%d\n", x += y);
12     printf("%d %d\n", x, y);
13     printf("%d\n", x -= y);
14     printf("%d %d\n", x, y);
15
16     return 0;
17 }
```

11줄의 출력 값과 같은 x의 값



결과

두 정수를 입력 >> 10 20  
30  
30 20  
10  
10 20

산술연산을 간략히 줄인 축약 대입연산자

<code>op1 += op2</code>	<code>op1 = op1 + op2</code>
<code>op1 -= op2</code>	<code>op1 = op1 - op2</code>
<code>op1 *= op2</code>	<code>op1 = op1 * op2</code>
<code>op1 /= op2</code>	<code>op1 = op1 / op2</code>
<code>op1 %= op2</code>	<code>op1 = op1 % op2</code>



$a=10$ ,  $b=2$ 인 경우, 다음 각각의 연산 결과는?

<code>a += b + 2; //a = a + (b + 2)</code>	14
<code>a -= b + 2;</code>	6
<code>a *= b + 2;</code>	40
<code>a /= b + 2;</code>	2
<code>a %= b + 2;</code>	2

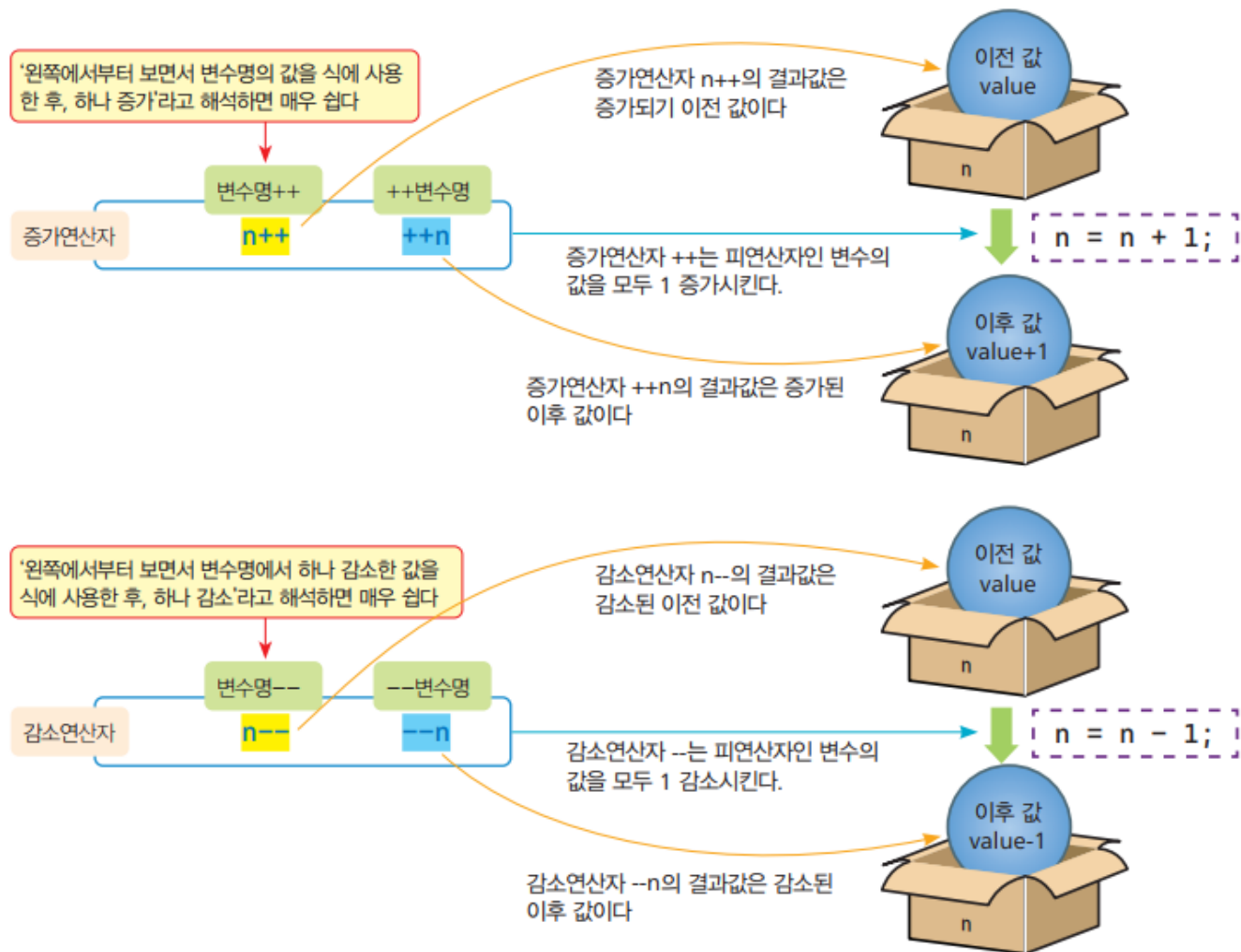
축약 대입연산자의 왼쪽 피연산자는 반드시 변수여야 하므로 다음은 잘못된 대입연산식

```
++a += b;
a+1 -= b;
a *= b; //=*가 아니라 *=
a /=a; //=/가 아니라 /=
```



# 증감 연산자 ++ --

- **n++**
  - 1 증가되기 전 값이 연산 결과값
- **++n**
  - 1 증가된 값이 연산 결과값



# 증가연산자 ++와 감소연산자 --

실습예제 5-4

Prj04

04incdecop.c

증가연산자 ++와 감소연산자 --

난이도: ★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int m = 1, n = 5;
06
07     printf("%d %d\n", m++, ++n); //1 6
08     printf("%d %d\n", m, n);      //2 6
09     printf("%d %d\n", m--, --n); //2 5
10     printf("%d %d\n", m, n);      //1 5
11
12     return 0;
13 }
```

출력 값은 결과값으로 증가되기 이전 값인 1이다.

출력 값은 결과값으로 증가된 값인 6이다.

결과

```
1 6
2 6
2 5
1 5
```

int n = 10;

출력

```
printf("%d\n", n++);
printf("%d\n", n);
```

10

11

int n = 10;

출력

```
printf("%d\n", ++n);
printf("%d\n", n);
```

11

11

int n = 10;

출력

```
printf("%d\n", n--);
printf("%d\n", n);
```

10

9

int n = 10;

출력

```
printf("%d\n", --n);
printf("%d\n", n);
```

9

9

# 관계 연산자

- 관계연산자는 두 피연산자의 크기를 비교하기 위한 연산자
  - 비교 결과가 참이면 1, 거짓이면 0

실습예제 5-5

Prj0505relationop.c관계 연산 활용난이도: ★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int speed = 80;
06     printf("%d ", (60 <= speed));
07     printf("%d\n", (60 > speed));
08
09     printf("%d ", ('a' > 'b'));
10     printf("%d\n", ('Z' <= 'a'));
11     printf("%d ", (4 == 4.0));
12     printf("%d\n", (4.0F != 4.0));
13
14     return 0;
15 }
```

모든 소문자는 대문자보다 코드 값이 크므로  
관계 연산도 참을 의미하고 결과는 1

4와 4.0은 같은 것으로 평가

결과

1 0
0 1
1 0

연산자	연산식	의미	예제	연산(결과)값
>	$x > y$	x가 y보다 큰가?	$3 > 5$	0(거짓)
>=	$x \geq y$	x가 y보다 크거나 같은가?	$5-4 \geq 0$	1(참)
<	$x < y$	x가 y보다 작은가?	'a' < 'b'	1(참)
<=	$x \leq y$	x가 y보다 작거나 같은가?	$3.43 \leq 5.862$	1(참)
!=	$x != y$	x와 y가 다른가?	$5-4 != 3/2$	0(거짓)
==	$x == y$	x가 y가 같은가?	'%' == 'A'	0(거짓)

# 논리연산자

- 논리연산자 &&, ||, !을 제공

- 각각 and, or, not 의 논리연산
- 결과가 참이면 1 거짓이면 0을 반환
  - 0, 0.0, '₩0'은 거짓을 의미
  - 0이 아닌 모든 정수와 실수, 그리고 널 (null) 문자 '₩0'이 아닌 모든 문자와 문자열은 모두 참을 의미

실습예제 5-6

Prj06      06logicop.c      논리 연산자 && || !      난이도: ★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     double grade = 4.21;
06
07     printf("%d ", (4.0 < grade) && (grade <= 5)); //1
08     printf("%d ", 0.0 || (4.0 > grade)); //0
09     printf("%d\n", (4.2 < grade) || !0.0); //1
10     printf("%d ", 'a' && 3.5); //1
11     printf("%d ", '\0' || "C"); //1
12     printf("%d\n", "java" && '\0'); //0
13
14     return 0;
15 }
16
```

결과

1 0 1  
1 1 0

x	y	x && y	x    y	!x
0(거짓)	0(거짓)	0	0	1
0(거짓)	0이 아닌 값(참)	0	1	1
0이 아닌 값(참)	0(거짓)	0	1	0
0이 아닌 값(참)	0이 아닌 값(참)	1	1	0

# 단축 평가

- 논리연산자 &&와 ||

- 피연산자 두 개 중에서 왼쪽 피연산자만으로 논리연산 결과가 결정된다면 오른쪽 피연산자는 평가하지 않는 방식

실습예제 5-7

Prj0707shorteval.c&& 연산자로 일정액 이상의 구매액에 쿠폰 발행과 할인 계산난이도: ★★

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03
04 int main(void)
05 {
06     int amount = 0;
07     int coupons = 10; //각각 10 이상과 10 미만을 입력
08
09     printf("총 금액 >> ");
10     scanf("%d", &amount); // 각각 10000원 이상과 미만을 입력
11
12     int sale = (amount >= 10000) && (coupons++ >= 10);
13     printf("할인: %d, 쿠폰 수: %d\n", sale, coupons);
14
15     return 0;
16 }
```

&&의 왼쪽 (amount >= 10000)가 만족되어야 (coupons++ >= 10)를 실행하며, 이것도 만족해야 최종 결과가 1이 되어 할인이 가능하다.

결과	int coupons = 10;	
	총 금액 >> 9000 할인: 0, 쿠폰 수: 10	총 금액 >> 10000 할인: 1, 쿠폰 수: 11
	int coupons = 8;	
	총 금액 >> 9000 할인: 0, 쿠폰 수: 8	총 금액 >> 10000 할인: 0, 쿠폰 수: 9

# 조건연산자

- 연산자 ? :
  - 조건에 따라 주어진 피연산자가 결과값이 되는 삼항연산자

실습예제 5-8	Prj08	08condop.c	조건 연산자 활용	난이도: ★
	<pre>01 #define _CRT_SECURE_NO_WARNINGS 02 #include &lt;stdio.h&gt; 03 04 int main(void) 05 { 06     int a = 0, b = 0; 07 08     printf("두 정수 입력 &gt;&gt; "); 09     scanf("%d%d", &amp;a, &amp;b); 10 11     printf("최대값: %d ", (a &gt; b) ? a : b); 12     printf("최소값: %d\n", (a &lt; b) ? a : b); 13     printf("절대값: %d ", (a &gt; 0) ? a : -a); 14     printf("절대값: %d\n", (b &gt; 0) ? b : -b); 15 16     ((a % 2) == 0) ? printf("짝수 ") : printf("홀수 "); 17     printf("%s\n", ((b % 2) == 0) ? "짝수" : "홀수"); 18 19     return 0; 20 }</pre>			
결과	<p>두 정수 입력 &gt;&gt; 8 -9 최대값: 8 최소값: -9 절대값: 8 절대값: 9 짝수 홀수</p>			

조건 삼항연산자의 두 번째와 세 번째 피연산자는 문장도 가능

조건 삼항연산자의 두 번째와 세 번째 피연산자는 문자열을 비롯하여 모든 자료형도 가능

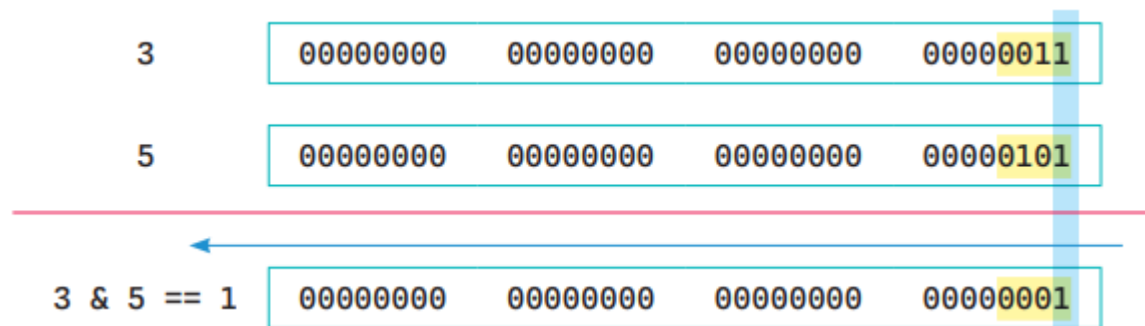
# 비트 논리연산자

- 연산자 &, |, ^, ~ 4가지

연산자	연산자 이름	사용	의미
&	비트 AND	op1 & op2	비트가 모두 1이면 결과는 1, 아니면 0
	비트 OR	op1   op2	비트가 적어도 하나 1이면 결과는 1, 아니면 0
^	비트 배타적 OR(XOR)	op1 ^ op2	비트가 서로 다르면 결과는 1, 같으면 0
~	비트 NOT(Negation) 또는 보수(complement)	~op1	비트가 0이면 결과는 1, 0이면 1

- 보수 연산자(bitwise complement operator) ~
  - 각 비트에서 0은 1, 1은 0이 결과

피연산자		보수 연산		
수	비트표현(2진수)	보수 연산 결과		10진수
1	000000000 000000000 000000000 000000001	111111111 111111111 111111111 111111110		~1 = -2
4	000000000 000000000 000000000 000000100	111111111 111111111 111111111 111111011		~4 = -5



# 비트 연산자 & | ^ ~

실습예제 5-9

Prj09

09bitop.c

비트 연산자 & | ^ ~

난이도: ★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int x = 15; // 1111
06
07     printf("%9x\n", -1);           // 1111
08     printf("%3d\n", 10 & -1);      // 10
09     printf("%3d\n\n", 10 | 0);     // 10
10
11     printf("%3d %08x\n", x, x);    // 1111
12     printf("%3d %08x\n", 1, x & 1); // 1111 & 0001
13     printf("%3d %08x\n", 15, x | 1); // 1111 | 0001
14     printf("%3d %08x\n", 14, x ^ 1); // 1111 ^ 0001
15     printf("%3d %08x\n", ~x, ~x);  // -16
16
17     return 0;
18 }
```

정수 15는 이진수로 1111

15 & 1의 결과는 1

결과

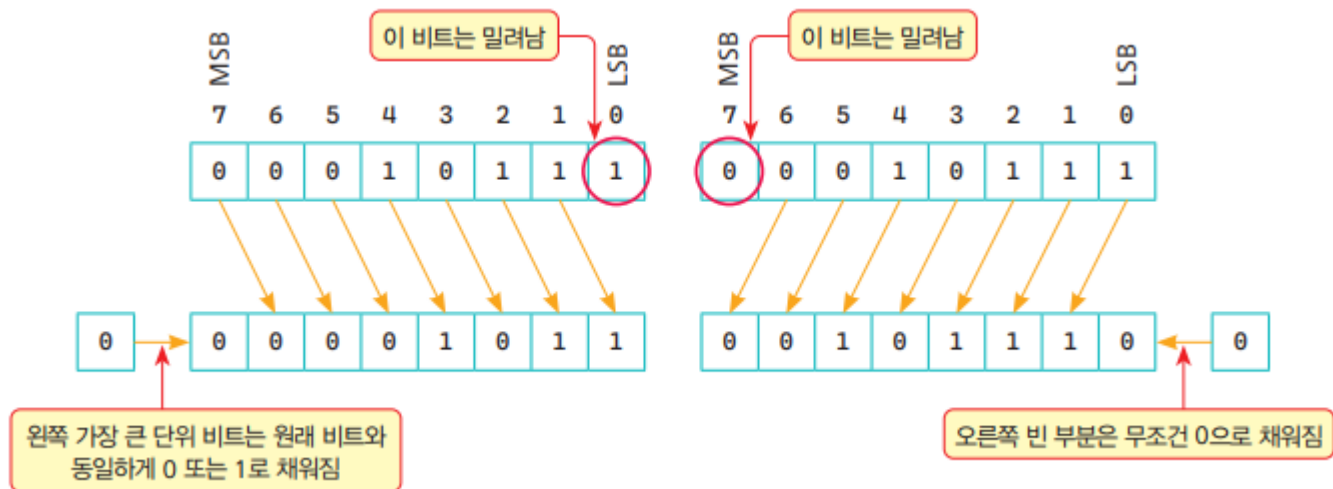
```
ffffffff
10
10

15 0000000f
1 00000001
15 0000000f
14 0000000e
-16 ffffffff0
```



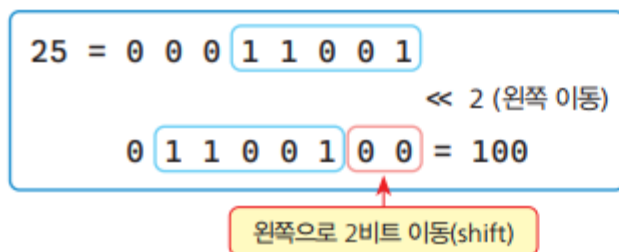
# 비트 이동연산자

- Shift right >>
- Shift left <<



연산자	이름	사용	연산 방법	새로 채워지는 비트
>>	shift left	$op1 \gg op2$	op1을 오른쪽으로 op2 비트만큼 이동	가장 왼쪽 비트인 부호 비트는 원래의 부호 비트로 채움
<<	shift right	$op1 \ll op2$	op1을 왼쪽으로 op2 비트만큼 이동	가장 오른쪽 비트를 모두 0으로 채움

왼쪽 이동 연산자(<<)



오른쪽 이동 연산자(>>)



# 예제: 비트 이동 연산자

실습예제 5-10	Prj10	10shiftop.c	비트 이동 연산자	난이도: ★
	01	#include <stdio.h>		
	02			
	03	int main(void)		
	04	{		
	05	int x = 0xffff; //정수 65535		
	06			
	07	printf("%d %08x\n", x, x); // 1111(f) 1111(f) 1111(f) 1111(f)		
	08	printf("%d %08x\n", x >> 1, x >> 1); // 0111(7) 1111(f) 1111(f) 1111(f)		
	09	printf("%d %08x\n", x >> 2, x >> 2); // 0011(3) 1111(f) 1111(f) 1111(f)		
	10	printf("%d %08x\n", x >> 3, x >> 3); // 0001(1) 1111(f) 1111(f) 1111(f)		
	11			
	12	printf("%d %08x\n", x << 1, x << 1); // 0001(1) 1111(f) 1111(f) 1111(f) 1110(e)		
	13	printf("%d %08x\n", x << 2, x << 2); // 0011(3) 1111(f) 1111(f) 1111(f) 1100(c)		
	14			
	15	return 0;		
	16	}		
설명	05	정수 65535는 16진수로 0000ffff		
	08~10	정수 >> n은 정수를 n번 2로 나눈 효과		
	12~13	정수 << n은 정수를 n번 2로 곱한 효과		
결과		65535 0000ffff		
		32767 00007fff		
		16383 00003fff		
		8191 00001fff		
		131070 0001fffe		
		262140 0003fffc		

# 정수에서 오른쪽 n번째 비트 값 알기

- 비트 연산자를 이용하여 정수에서 오른쪽 n번째 비트 값 알기



**TIP**

비트 연산자를 이용하여 정수에서 오른쪽 n번째 비트 값 알기

임의 정수  $x$ 의 비트 연산  $x \& 1$ 의 결과는 0 또는 1이다. 즉 결과는 정수  $x$ 의 가장 오른쪽 비트 값이 0이면 0, 1이면 1이된다. 그렇다면 어느 정수에서 오른쪽  $n$ 번째 비트 값을 알 수 있는 방법을 생각해 보자. 비트 연산  $x \gg (n-1)$ 은  $x$ 의 오른쪽  $n$ 번째 비트를 가장 오른쪽으로 이동시킨다. 그러므로 비트 연산  $(x \gg (n-1)) \& 1$ 의 결과가 바로 정수에서 오른쪽  $n$ 번째 비트 값이라는 것이라는 알 수 있다. 이와 같은 비트 연산을 이용하면 정수를 이진수로 표현할 수 있다.

```
int x = 0x2f;
```

```
printf("%d", x >> 7 & 1); //8 번째 자리
```

```
printf("%d", x >> 6 & 1); //7 번째 자리
```

```
printf("%d", x >> 5 & 1); //6 번째 자리
```

```
printf("%d", x >> 4 & 1); //5 번째 자리
```

# 내림변환과 올림변환

- 자료형 크기

- char < short < int < long < long long < float < double < long double

- 올림변환

- 작은 범주의 자료형(int)에서 보다 큰 범주인 형(double)으로의 형변환 방식

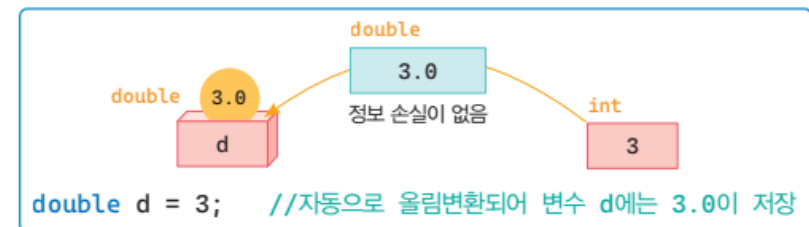
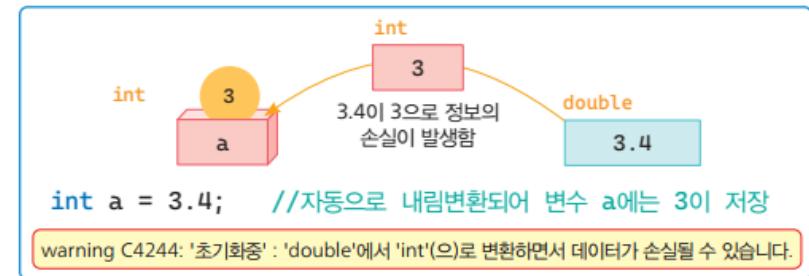
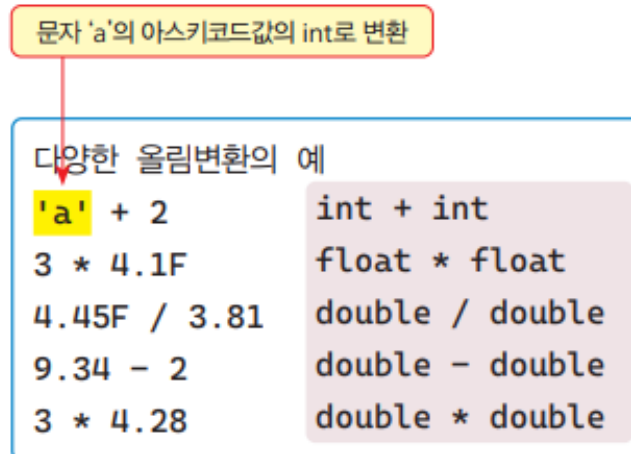
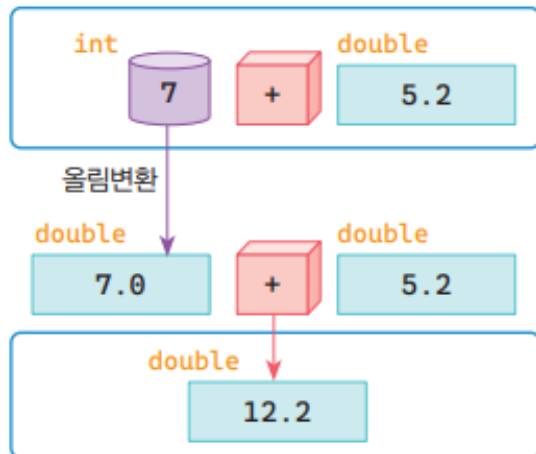
- 내림변환

- 큰 범주의 자료형(double)에서 보다 큰 범주인 형(int)으로의 형변환 방식

- 자료형 변환 구분 방식: 명시적 형변환과 묵시적 형변환

- 명시적(강제) 형변환: 소스에서 직접 형변환 연산자를 사용하는 방식

- 묵시적(자동) 형변환: 컴파일러가 알아서 자동으로 수행하는 방식



# 형변환 연산자

- (int) 30.525

실습예제 5-11

Prj11 11typecast.c 형변환 연산자 활용 난이도: ★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int a = 7.8;    //자동으로 내림변환되어 변수 a에는 7.8이 저장
06     double b = 5;  //자동으로 올림변환되어 변수 b에는 5.0이 저장
07
08     printf("%d %f ", a, b);
09     printf("%d %f ", (int) 3.56, (double) 3);
10     printf("%f %d\n ", 3.56 + 7.87, (int)(3.56 + 7.87));
11
12     printf("%d %f %f\n", 5 / 2, (double) 5 / 2, (double) (5 / 2));
13
14     return 0;
15 }
```

결과

7 5.000000 3 3.000000 11.430000 11  
2 2.500000 2.000000

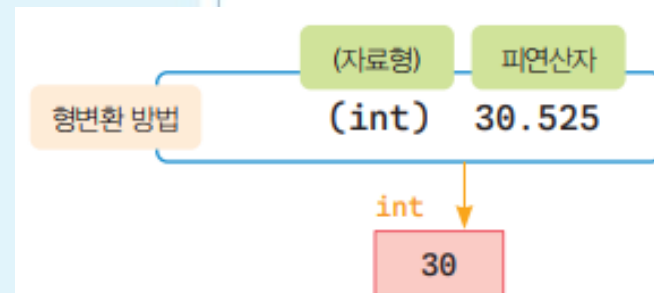
형변환 방법

(자료형)    피연산자  
(int)    30.525

int  
30

결과는 정수인 2

5를 먼저 double로 변환한 후  
5.0 / 2를 연산하므로 결과는 2.5



# sizeof 연산자와 콤마연산자

- **sizeof 3.14**

- 연산값 또는 자료형의 저장장소의 크기를 구하는 연산자
  - 바이트 단위의 정수
- 피연산자 앞에 위치하는 전위 연산자
- 피연산자가 int와 같은 자료형인 경우 반드시 괄호를 사용
- 피연산자가 상수나 변수 또는 연산식이면 괄호는 생략 가능

- **콤마연산자**

- 콤마연산자 ,는 왼쪽과 오른쪽 연산식을 각각 순차적으로 계산하며 결과값은 가장 오른쪽에서 수행한 연산의 결과
- 콤마 연산자는 연산의 우선순위가 가장 늦어 대입연산자보다 나중에 계산

- **반환 값 자료형**

- 대부분의 변수는 동작 환경에 따라 크기가 달라짐. 하지만, size\_t는 항상 4바이트로 고정되어 있음. 따라서, 어디서든 4바이트의 크기를 가지는 변수를 선언하고 싶을 때 size\_t로 선언
- printf()에서 형식제어문자 %zu로 출력 => z: size / u: unsigned를 의미

```
size_t sz = sizeof (short);  
printf("%zu\n", sz);
```

# 연산자 sizeof와 콤마 연산자의 활용

실습예제 5-12

Prj12

12sizeofcomma.c

연산자 sizeof와 콤마 연산자의 활용

난이도: ★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int a, x;
06     a = x = 0;
07
08     x = 3 + 4, 2 * 3;    // (x = 3+4), 2*3;
09     printf("x = %d ", x);
10     x = (3 + 4, 2 * 3); // x = (3+4, 2*3);
11     printf("x = %d\n", x);
12
13     int byte = sizeof (double);
14     printf("double 형: %d bytes, %d bits\n", byte, byte * 8);
15     int bit = (byte = sizeof a, byte * 8);
16     printf("int 형: %d bytes, %d bits\n", byte, bit);
17
18     size_t sz = sizeof (short);
19     printf("%zu\n", sz);
20
21     return 0;
22 }
```

연산자 sizeof (자료형)에서 괄호는 필수이며, 결과 자료형은 size\_t이나 간단히 int형 자료형에 저장도 가능

변수 bit에 저장되는 것은 나중에 계산된 byte \* 8

결과

```
x = 7 x = 6
double 형: 8 bytes, 64 bits
int 형: 4 bytes, 32 bits
2
```

# 연산자 우선순위와 결합성

## • 연산 규칙

- 첫 번째 규칙: 괄호가 있으면 먼저 계산
- 두 번째 규칙: 연산의 우선순위(priority)
- 세 번째 규칙: 동일한 우선순위인 경우, 연산을 결합하는 방법인 결합성(또는 결합규칙)
  - 왼쪽부터 오른쪽으로 차례로 계산
  - 제곱승과 같은 정해진 연산은 오른쪽에서 왼쪽으로 차례로 계산

### 이항연산자

coma < 대입 < 조건(삼항) < 논리 < 관계 < 산술 < 단항 < 괄호와 대괄호

- 괄호와 대괄호는 무엇보다도 가장 먼저 계산한다.
- 모든 단항연산자는 어느 이항연산자보다 먼저 계산한다.
- 산술연산자 \*, /, %는 +, -보다 먼저 계산한다.
- 산술연산자는 이항연산자 중에서 가장 먼저 계산한다.
- 관계연산자는 논리연산자보다 먼저 계산한다.
- 조건 삼항연산자는 대입연산자보다 먼저 계산하나, 다른 대부분의 연산보다는 늦게 계산한다.
- 조건 > 대입 > 콤마연산자 순으로 나중에 계산한다.



표 5-9 C 언어의 연산자 우선순위

우선순위	연산자	설명	분류	결합성(계산방향)
1	( )	함수 호출 및 우선 지정	단항	-> (좌에서 우로)
	[ ]	인덱스		
	. -> a++ a--	필드(유니온) 멤버 지정 필드(유니온)포인터 멤버 지정 후위 증가, 후위 감소		
2	++a --a ! ~ sizeof - + & *	전위 증가, 전위 감소 논리 NOT, 비트 NOT(보수) 변수, 자료형, 상수의 바이트 단위 크기 음수 부호, 양수 부호 주소 간접, 역참조		<- (우에서 좌로)
3	(형변환)	형변환		
4	* / %	곱하기 나누기 나머지	산술	-> (좌에서 우로)
5	+ -	더하기 빼기		
6	<< >>	비트 이동	이동	-> (좌에서 우로)
7	< > <= >=	대소 비교	관계	-> (좌에서 우로)
8	= = !=	동등 비교		
9	&	비트 AND 또는 논리 AND	비트	-> (좌에서 우로)
10	^	비트 XOR 또는 논리 XOR		
11		비트 OR 또는 논리 OR		
12	&&	논리 AND(단락 계산)	논리	-> (좌에서 우로)
13		논리 OR(단락 계산)		
14	? :	조건	조건	<- (우에서 좌로)
15	= += -= *= /= %= <<= >>= &=  = ^=	대입	대입	<- (우에서 좌로)
16	,	coma	coma	-> (좌에서 우로)

# 연산자 우선순위에 위한 계산

실습예제 5-13

Prj13

13oppriority.c

연산자 우선순위에 위한 계산

난이도: ★★

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      int speed = 90;
06      int x = 1, y = 2, z = 3;
07
08      printf("%d ", 60 <= speed && speed <= 80 + 20); //산술 > 관계 > 논리
09      printf("%d ", ( (60 <= speed) && (speed <= (80 + 20)) ));
10
11      printf("%d ", x % 2 == 0 ? y + z : y * z); //산술 > 관계 > 조건
12      printf("%d ", (x % 2 == 0) ? (y + z) : (y * z));
13
14      printf("%d ", speed += ++x && y - 2); //단항++ > 산술 > 논리 > 대입
15      printf("%d\n", speed += ( (++x) && (y - 2) ));
16
17      return 0;
18  }
```

결과

1 1 6 6 90 90

# 연산자의 결합성에 따른 계산 순서 확인

실습예제 5-14

Prj14

14opassociation.c

연산자의 결합성에 따른 계산 순서 확인

난이도: ★★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int m = 5, n = 10;
06     printf("%d\n", m += n /= 3); //우측에서 좌측으로 결합, (m += (n /= 3))
07     printf("%d %d\n", m, n); //8, 3
08
09     //우측에서 좌측으로 결합
10     printf("%d ", 3 > 4 ? 3 - 4 : 3 > 4 ? 3 + 4 : 3 * 4); //12
11     printf("%d\n", 3 > 4 ? 3 - 4 : (3 > 4 ? 3 + 4 : 3 * 4)); //위와 같은 12
12
13     printf("%d ", 10 * 3 / 2); //좌측에서 우측으로 결합, 15
14     printf("%d\n", 10 * (3 / 2)); //우측에서 좌측으로 결합, 10
15
16     return 0;
17 }
```

설명

06 연산식  $m += n /= 3$ 은  $(m += (n /= 3))$ 이므로 결과값은 m에 대입된 8 출력  
10 조건연산자는 결합성이 오른쪽에서 왼쪽으로  
13~14 연산식  $10 * 3 / 2$ 은  $(10 * 3) / 2$ 이므로  $10 * (3 / 2)$ 과 결과가 다름

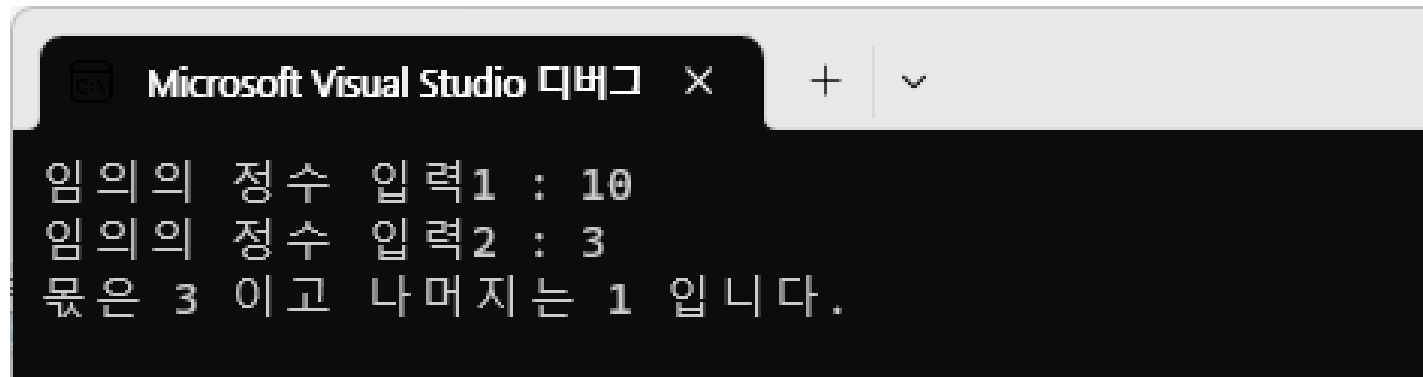
결과

8  
8 3  
12 12  
15 10

## 2. Quiz

## 문제 1

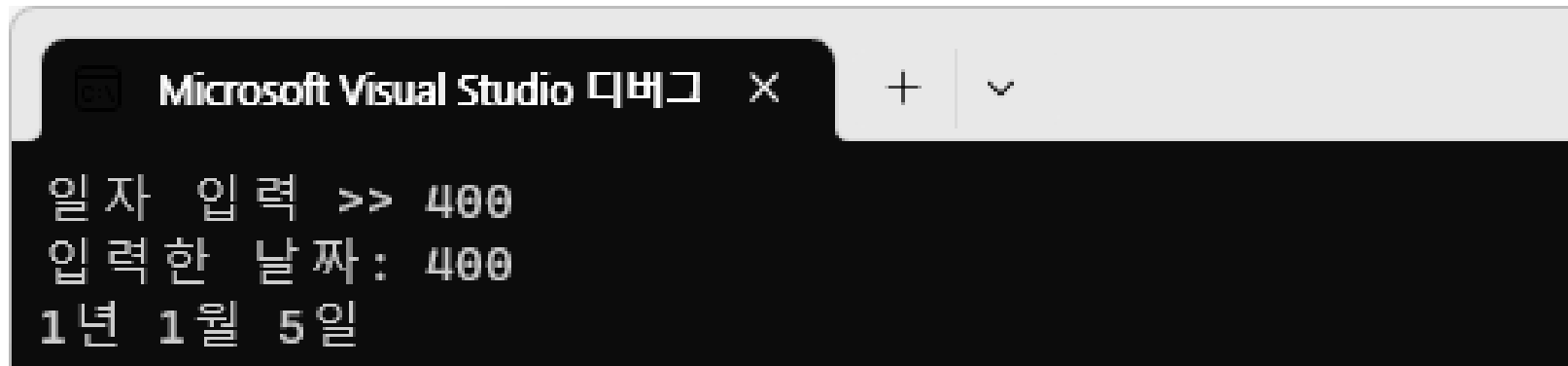
- 표준입력으로 두 정수를 입력 받아 큰 수를 작은 수로 나눔 몫과 나머지를 각각 출력하는 프로그램을 작성하시오.



```
Microsoft Visual Studio 디버그 × + ▾  
임의의 정수 입력1 : 10  
임의의 정수 입력2 : 3  
몫은 3 이고 나머지는 1 입니다.
```

## 문제 2

- 일자(dates)를 정수로 입력 받아 년, 월, 일을 출력하는 프로그램을 작성하시오.
  - 1년은 365일로, 1달은 30일로 계산



```
Microsoft Visual Studio 디버그
일자 입력 >> 400
입력한 날짜: 400
1년 1월 5일
```

## 문제 3

- 원금이 1,000,000인 경우, 예치 기간을 년 단위로 입력 받아 단리 이자 만기 시 총 금액을 출력하는 프로그램을 작성하시오.
  - 연 단위(단리 이자) = 원금 \* 이율(4.5%) \* 년(예치기간)
  - 만기 시 총 수령액(단리 적용) = 원금(1 + 이율(4.5%) \* 년(예치 기간))



```
Microsoft Visual Studio 디버그 X + v
예치 기간 입력(년) >> 5
이율: 4.5% 총금액: 1225000.00
```

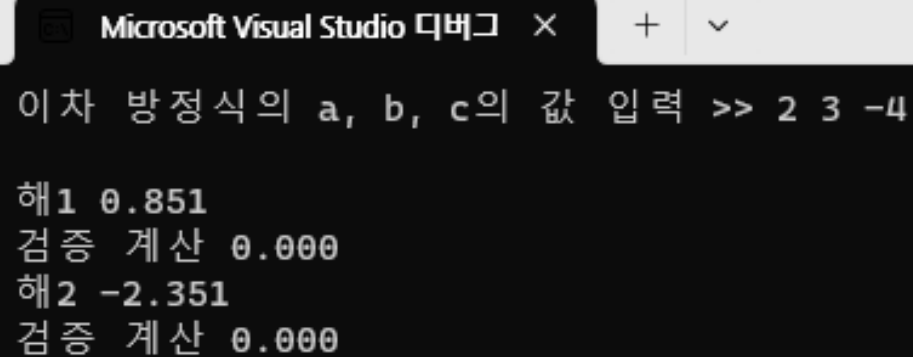
## 문제 4

- 이차 방정식  $ax^2 + bx + c = 0$ 에서  $a, b, c$ 의 값을 입력 받아 해를 구한 후 다시 검증하여 출력하는 프로그램을 작성하시오.

- 근의 공식:  $ax^2 + bx + c = 0$ 일 때(단,  $a \neq 0$ )

$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- 제곱근 계산은 `#include <math.h>`로 함수 `sqrt()` 사용



```
Microsoft Visual Studio 디버그 × + v
이차 방정식의 a, b, c의 값 입력 >> 2 3 -4

해1 0.851
검증 계산 0.000
해2 -2.351
검증 계산 0.000
```