

3. 데이터유형 및 연산자

Autumn 2019

Today

1. 지난 시간 실습 복습

2. 연산자1

3. 연산자2

4. 실습

[복습] 용어 정의

■ 데이터유형(데이터형)

- 데이터형은 어떤 형태로 변수를 저장할지 결정한다.

■ 변수

- 변수는 저장공간을 지칭하는 것

■ 연산자

- 연산자는 구체적으로 데이터들을 조작하는 방법을 명시한다.
- +, -, /, %, // 등

■ 표현(Expressions)

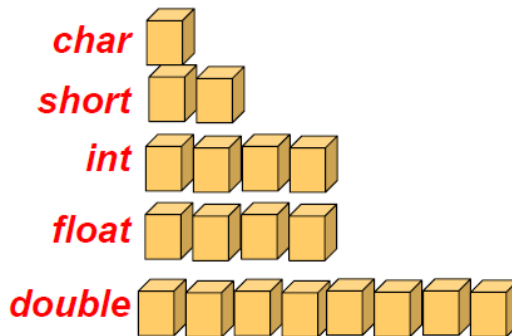
- 프로그래밍 언어에서 '표현'은 변수, 연산자, 함수들의 조합

■ 예를 들어, `int x=0, y=0, y=x+2;`

- x, y 는 변수
- `y=y+2` 는 표현
- + 는 연산자

[복습] c언어의 기본요소

- 변수: 프로그램이 실행되는 동안의 저장공간을 통칭
 - 변수 선언과 초기화
 - 변수 만들기 규칙
- 기본 데이터유형
 - char, int, float, double



[복습] 변수 선언하기

- ‘선언’하는 일반적인 형태는,

데이터형 변수명 [=값]

와 같이 사용함

- 예를 들면,
 - **char** x; /* uninitialized */
 - **char** x=' A' ; /* intialized to 'A'*/
 - **char** x=' A' ,y=' B' ; /*multiple variables initialized */

[복습] 변수명 규칙

- 다음에서 T/F 를 구분하시오.
 - `int money$owed;`
 - `int total_count`
 - `int score2`
 - `int 2ndscore`
 - `int long`

[복습] 비트, 바이트, 진수

- 비트(bit) 는 0과 1만 존재함 (끄고/키는 스위치와 비슷)
- 바이트(byte)는 비트 8개가 합쳐진 단위 (1byte = 8bit)
- 진수
 - 2진수: 비트를 표현
 - 10진수: 우리가 사용하고 있는 숫자체계
 - 16진수: 2진수의 네 자리를 한 자리로 용이하게 표현 (0~9, A~F)

비트 수	바이트 수	표현 개수	2진수	10진수	16진수
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} =약 42억	0~...	0~약 42억	0~FFFF FFFF
64	8	2^{64} =약 1800경	0~...	0~약 1800경	0~...

[복습] 데이터형 크기

- char 형의 크기는 1byte (= 8bit) → 약 2^8 개의 서로 다른 문자 표현가능
 - 정확한 값의 범위: $-2^7 \sim 2^7 - 1$
 - 8bit 중에서 음수/양수를 표현하는데 1bit를 사용해서 2^7 개가 됨
 - 부호 표현 하지 않는 unsigned char 인 경우에는 값의 범위가 $2^8 - 1$ 이 됨
 - 숫자를 입력하면 아스키(ASCII) 코드가 출력됨

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

```
char ch = 97;
```

위의 두 문장은 동일함

- int 형의 크기는 4byte (= 32bit) → 약 2^{32} 개의 서로 다른 숫자 표현가능
 - 정확한 값의 범위: $-2^{31} \sim 2^{31} - 1$
- float 형의 크기는 4byte (= 32bit) → 약 2^{32} 개의 서로 다른 숫자 표현가능
 - 정확한 값의 범위: $-2^{31} \sim 2^{31} - 1$
- double 형의 크기는 8byte (= 64bit) → 약 2^{64} 개의 서로 다른 숫자 표현가능
 - 정확한 값의 범위: $-2^{63} \sim 2^{63} - 1$

[복습] 변수형과 크기

- 숫자형 (int, float, double)
 - 문자형 (char)
 - 사용자정의 (struct, union)
-
- 변수형의 크기는 기계/컴파일러에 따라 다를 수 있음
하지만, 다음의 관계는 항상 성립함

sizeof(char) < sizeof(short) <= sizeof(int) <= sizeof(long)

sizeof(char) < sizeof(short) <= sizeof(float) <= sizeof(double)

[실습] 큰 실수 출력

- double 형을 출력할 때는 **%lf** 로 출력함
- float은 소수점 7자리, double 은 소수점 15자리까지 유효

```
#include <stdio.h>

int main()
{
    float a = 0.1234567890123456789012345;
    double b = 0.1234567890123456789012345;

    printf("%30.25f \n", a);
    printf("%30.25lf \n", b);
}
```

실행결과

```
0.1234567910432815551757812
0.1234567890123456773698862
```

: 유효범위 넘어가면 이상한 값 출력됨

- 다음 결과를 예상해보시오.

```
#include <stdio.h>

int main()
{
    float a = 1495982315.236;
    printf("%f \n", a);

    double b = 14959823154.236;
    printf("%f \n", b);
}
```

실행결과

1495982336.000000, 1495982315.236000

[복습] 문자형

[1] `int a = 65;` [2] `char a = 65;` [3] `char a = '65';` 의 차이점은?

▶ [1]

▶ [2]

▶ [3]

```
#include <stdio.h>

int main()
{
    char a, b, c;

    a = 'A';
    printf("%c \n", a);
    printf("%d \n", a);

    b = 'a';
    c = b + 5;
    printf("%c \n", b);
    printf("%c \n", c);

    c = 90;
    printf("%c \n", c);
}
```

실행결과

```
A
65
a
f
Z
```

[실습] 형변환

```
#include <stdio.h>

int main()
{
    int a = 100, b = 200;
    float result;

    result = a / b;

    printf("%f \n", result);
}
```

실행결과

```
0.000000
Program ended with exit code: 0
```

- 숫자연산 시 연산하는 타입의 큰 쪽을 따라가게 됨

```
#include <stdio.h>

int main()
{
    int a = 100;
    float b = 200;
    float result;

    result = a / b;

    printf("%f \n", result);
}
```

1) a와 b 중의 하나를 float으로 선언

```
#include <stdio.h>

int main()
{
    int a = 100, b = 200;
    float result;

    result = (float)a / b;

    printf("%f \n", result);
}
```

2) 이미 선언된 변수의 형식을 변환 (형변환이라고 부름)

[실습] 원둘레와 원의 넓이 구하기

- 아래와 같이 출력하도록 하시오
 - 반지름 변수 `r`를 `double` 형태로 선언하고, 초기값은 0으로 할당한다.
 - 원주율 변수 `pi`를 `double` 으로 선언하고, 3.1415로 할당한다.
 - 원의 반지름을 다음 명령어를 써서 입력받으시오.
 - `scanf_s("%lf", &r);`
 - 계산한 원의 둘레는 `length` 변수에 할당하고,
 - 계산한 원의 면적은 `area` 변수에 할당하시오.
 - 둘레와 면적을 소수점 셋째자리까지 출력하시오.

원의 반지름을 입력하세요.

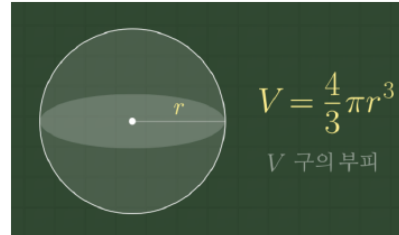
3.5

원의 둘레는 21.991 입니다.

원의 넓이는 38.483 입니다.

[실습] 구의 부피 구하기

- ▶ 반경이 r 인 구의 부피는 다음의 공식으로 구할 수 있다.



- ▶ 구의 반경 r 을 정수값으로 입력받아 구의 부피를 소수 둘째자리까지 출력하고, 이를 반올림한 값도 같이 출력해 보세요. 아래의 실행 결과를 참고하여 프로그램을 작성하세요.

구의 반경(cm)을 입력하세요.

12

반경이 12cm인 구의 부피는 5428.51(반올림하여 5429)cm³입니다.

구의 반경을 입력하세요.

12

반경이 12 인 구의 부피는 7238.02 (반올림하면 7238)입니다.

Today

1. 지난 시간 실습 복습
2. 연산자1
3. 연산자2
4. 실습

연산자 종류

- 산술 연산자
- 증감(증가감소) 연산자
- 대입(할당) 연산자
- 관계 연산자
- 논리 연산자
- 비트 연산자

산술 연산자

operator	meaning	examples
+	더하기	<code>x=3+2; /*constants*/</code> <code>y+z; /*variables*/</code> <code>x+y+2; /*both*/</code>
-	빼기	<code>3-2; /*constants*/</code> <code>int x=y-z; /*variables*/</code> <code>y-2-z; /*both*/</code>
*	곱하기	<code>int x=3*2; /*constants*/</code> <code>int x=y*z; /*variables*/</code> <code>x*y*2; /*both*/</code>
/	나누기	<code>float x=3/2; /*produces x=1 (int /)*/</code> <code>float x=3.0/2 /*produces x=1.5 (float /)*/</code> <code>int x=3.0/2; /*produces x=1 (int conversion)*/</code>
%	나머지	<code>int x=3%2; /*produces x=1*/</code> <code>int y=7;int x=y%4; /*produces 3*/</code> <code>int y=7;int x=y%10; /*produces 7*/</code>

[기본 4-1] 산술 연산자 사용 예시

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a, b = 5, c = 3;
06
07     a = b + c;      ----더하기로 연산해서 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 }
```

[4-2] 연산자 우선순위와 강제 형 변환 예시

■ 연산자 우선순위와 강제 형 변환

```
01 #include <stdio.h>
02
03 int 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 }
```

[4-2] 연산자 우선순위와 강제 형 변환 예시

- 데이터 형식의 강제 형 변환
[4-2]코드의 15행에서 형 변환을 하지 않을 경우

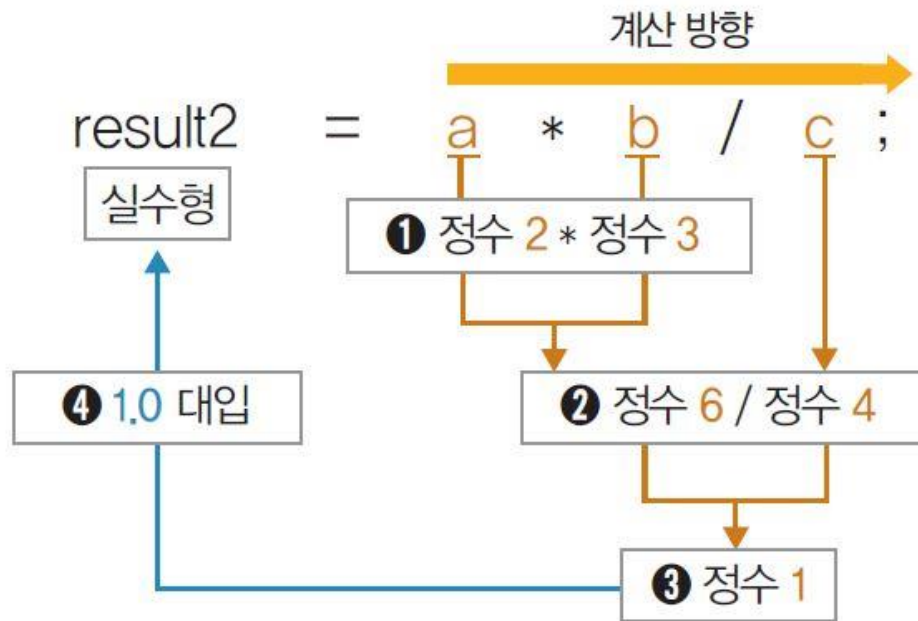


그림 4-1 강제 형 변환을 하지 않았을 때의 결과

[4-2] 연산자 우선순위와 강제 형 변환 예시

- 데이터 형식의 강제 형 변환

[4-2]코드의 15행에서 **형 변환**을 할 경우

- 강제 형 변환을 하려면 **형식은 변수 또는 상수 앞에 '(형식 이름)'**을 써줌

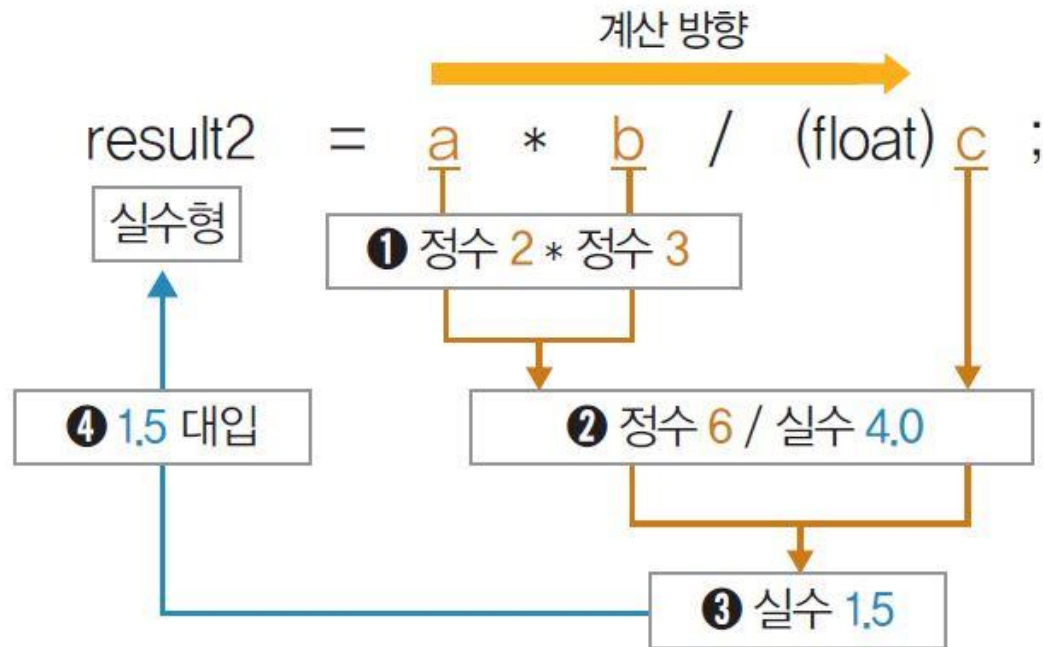


그림 4-2 강제 형 변환을 했을 때의 결과

증감(증가감소) 연산자

- 증감연산은 수리연산에서 일반적으로 사용되며, 두 가지 단축표현이다. `++`, `--`
- `x++` 은 `x = x+1` 의 단축표현
- `x--` 은 `x = x-1` 의 단축표현
- `y = x++` 는 `y=x; x=x+1;` 를 뜻하는 것으로 `x` 를 `y`에 대입한 후에 `x` 값이 1 증가됨

```
int x=0; int y=0;  
y= x++;  
printf("%d \t %d \n", x, y);
```

- `y = x--` 는 `y=x; x=x-1;` 를 뜻하는 것으로 `x`를 `y`에 대입한 후에 `x` 값이 1 감소됨

```
int x=0; int y=0;  
y= x--;  
printf("%d \t %d \n", x, y);
```

대입(할당) 연산자

- C언어에서 가장 빈번하게 나타나는 연산자는 “=”

$x = x + 1$

$x = x * 10$

$x = x / 2$

- C언어는 축약한 할당연산도 제공한다.

$x += 1$ /*is the same as $x = x + 1$ */

$x -= 1$ /*is the same as $x = x - 1$ */

$x *= 10$ /*is the same as $x = x * 10$ */

$x /= 2$ /*is the same as $x = x / 2$ */

$x \% = 2$ /*is the same as $x = x \% 2$ */

x++ 과 ++x 차이

```
#include <stdio.h>

int main()
{
    int a = 10, b;

    b = a++;
    printf("a = %d\n", a);
    printf("b = %d\n", b);

    b = ++a;
    printf("a = %d\n", a);
    printf("b = %d\n", b);
}
```

실행결과

```
a = 11
b = 10
a = 12
b = 12
```

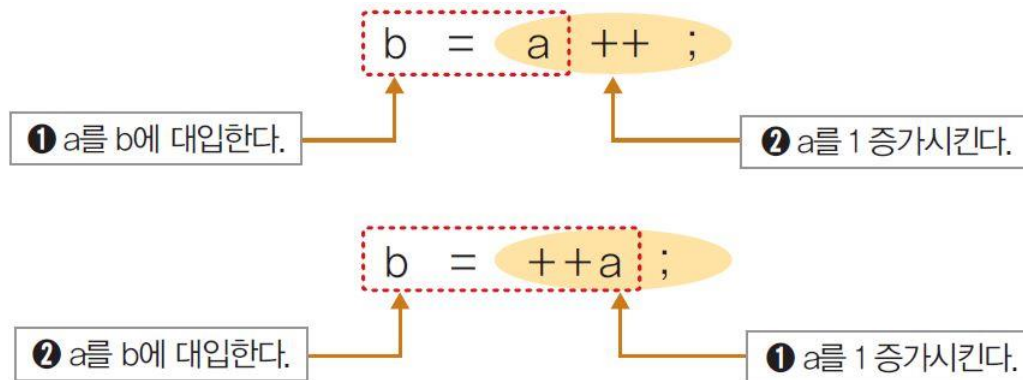


그림 4-3 a++와 ++a의 차이

관계 연산자

$$a < b = \begin{cases} \text{참: 1} \\ \text{거짓: 0} \end{cases}$$

그림 4-4 관계 연산자의 기본 개념

- 관계 연산자(또는 비교 연산자)는 어떤 것이 큰지, 작은지, 같은지 비교하는 것
 - 연산 결과는 참(True, 1)이나 거짓(False, 0)
(C언어에서는 보통 0이 아닌 값을 True로, 0은 False로 간주함)
 - 주로 조건문이나 반복문에 사용, 단독으로 쓰이지 않음.

operator	meaning	examples
>	greater than	3>2; /*evaluates to 1 */ 2.99>3 /*evaluates to 0 */
>=	greater than or equal to	3>=3; /*evaluates to 1 */ 2.99>=3 /*evaluates to 0 */
<	lesser than	3<3; /*evaluates to 0 */ 'A' < 'B' /*evaluates to 1*/
<=	lesser than or equal to	3<=3; /*evaluates to 1 */ 3.99<3 /*evaluates to 0 */
==	equal to	3==3; /*evaluates to 1 */ 'A' == 'a' /*evaluates to 0 */
!=	not equal to	3!=3; /*evaluates to 0 */ 2.99!=3 /*evaluates to 1 */

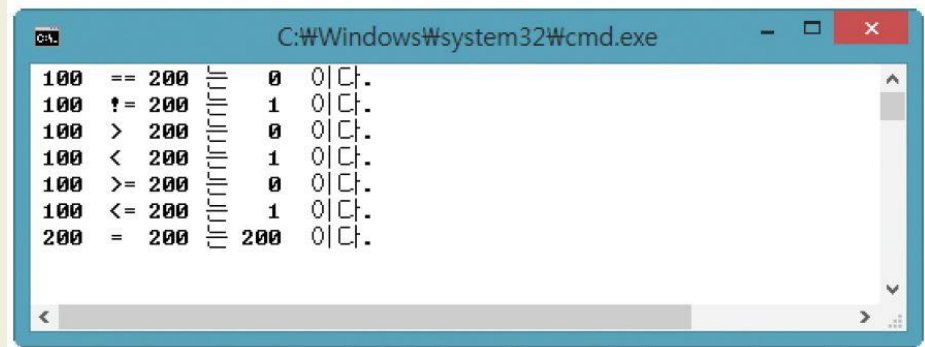
관계 연산자

- 같은지 비교하는 “==” 연산자는 “=”와 다르다.
- 프로그래밍에서 “=” 는 할당(대입)을 뜻하는 연산자
- “==” 사용시 주의사항
 - float 자료형을 비교할 때는, 소수점이 생기기 때문에 유의해서 사용해야 함

[4-5] 관계 연산자의 사용 예

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a = 100 , b = 200;
06
07     printf(" %d == %d 는 %d 이다.\n", a, b, a==b); ----관계 연산자 '같다'
08     printf(" %d != %d 는 %d 이다.\n", a, b, a!=b); ----관계 연산자 '같지 않다'
09     printf(" %d > %d 는 %d 이다.\n",a, b, a>b); ----관계 연산자 '크다'
10     printf(" %d < %d 는 %d 이다.\n",a, b, a<b); ----관계 연산자 '작다'
11     printf(" %d >= %d 는 %d 이다.\n", a, b, a>=b); ----관계 연산자 '크거나 같다'
12     printf(" %d <= %d 는 %d 이다.\n", a, b, a<=b); ----관계 연산자 '작거나 같다'
13
14     printf(" %d = %d 는 %d 이다.\n", a, b, a=b); ----관계 연산자가 아닌 대입 연산자를 사용했다.
15 }
```

실행 결과 ▼



```
C:\Windows\system32\cmd.exe
100 == 200 0 이다.
100 != 200 1 이다.
100 > 200 0 이다.
100 < 200 1 이다.
100 >= 200 0 이다.
100 <= 200 1 이다.
200 = 200 200 이다.
```

Today

1. 지난 시간 실습 복습
2. 연산자1
3. 연산자2
4. 실습

논리 연산자

■ 참/거짓을 판단

operator	meaning	examples
&&	AND	<code>((9/3)==3) && (2*3==6); /*evaluates to 1 */</code> <code>(' A' == ' a') && (3==3) /*evaluates to 0 */</code>
	OR	<code>2==3 ' A' == ' A' ; /*evaluates to 1 */</code> <code>2.99>=3 0 /*evaluates to 0 */</code>
!	NOT	<code>!(3==3); /*evaluates to 0 */</code> <code>!(2.99>=3) /*evaluates to 1 */</code>

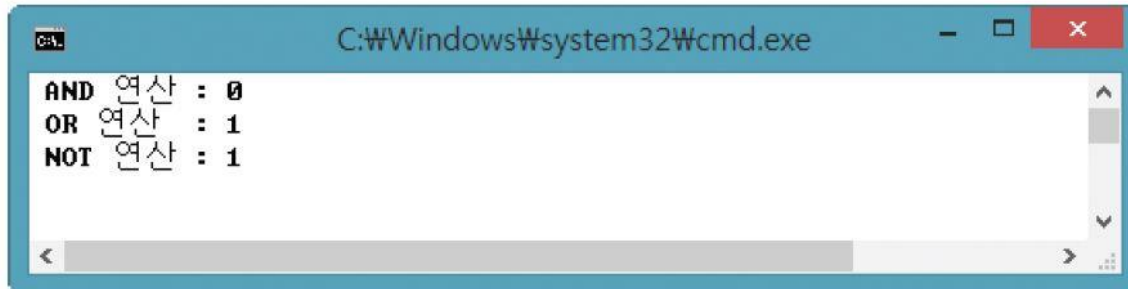
표 4-4 논리 연산자

연산자	의미		사용 예	설명
&&	~ 이고	그리고(AND)	<code>(a>100) && (a<200)</code>	둘 다 참이어야 참이다.
	~ 이거나	또는(OR)	<code>(a>100) (a<200)</code>	둘 중 하나만 참이어도 참이다.
!	~ 아니다	부정(NOT)	<code>!(a==100)</code>	참이면 거짓, 거짓이면 참이다.

[4-6] 논리 연산자 사용 예 ①

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a = 99;
06
07     printf(" AND 연산 : %d \n", (a >= 100) && (a <= 200)); ---AND 연산을 사용한다.
08     printf(" OR 연산 : %d \n", (a >= 100) || (a <= 200)); ---OR 연산을 사용한다.
09     printf(" NOT 연산 : %d \n", !(a == 100)); ---NOT 연산을 사용한다.
10 }
```

실행 결과 ▼



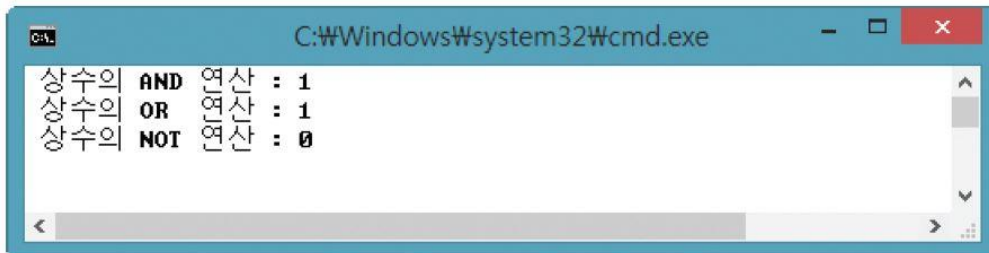
The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window contains the following output:

```
AND 연산 : 0
OR 연산 : 1
NOT 연산 : 1
```

[4-7] 논리 연산자 사용 예 ②

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a = 100, b = -200;
06
07     printf(" 상수의 AND 연산 : %d \n", a && b);    ---AND 연산을 사용한다.
08     printf(" 상수의 OR 연산 : %d \n", __①__);      ---OR 연산을 사용한다.
09     printf(" 상수의 NOT 연산 : %d \n", __②__);     ---NOT 연산을 사용한다.
10 }
```

실행 결과 ▼



```
C:\Windows\system32\cmd.exe
상수의 AND 연산 : 1
상수의 OR 연산 : 1
상수의 NOT 연산 : 0
```

기타연산자들 - (잘 쓰이지 않음)

- 비트연산자
 - 논리곱 (&)
 - 논리합 (|)
 - 배타적논리합 (^)
 - 부정 (~)
 - 왼쪽시프트 (<<)
 - 오른쪽시프트 (>>)

비트 연산자

- 비트 연산자는 정수나 문자 등을 **2진수로 변환한 후에** 각 자리의 비트끼리 연산 수행함

표 4-5 비트 연산자

연산자	명칭	설명
&	비트 논리곱(AND)	둘 다 1이면 1이다.
	비트 논리합(OR)	둘 중 하나만 1이면 1이다.
^	비트 배타적 논리합(XOR)	둘이 같으면 0, 둘이 다르면 1이다.
~	비트 부정	1은 0으로, 0은 1로 변경한다.
<<	비트 왼쪽 시프트(이동)	비트를 왼쪽으로 시프트(이동)한다.
>>	비트 오른쪽 시프트(이동)	비트를 오른쪽으로 시프트(이동)한다.

- AND : 둘 다 true 일 때, true
- OR : 둘 중에 하나만 true 면, true
- XOR: 한 개만 true면 true

비트 연산자 - 논리곱(&)

- 10진수를 2진수로 변환한 후 각 비트에 AND 연산 수행
`printf("%d\n", 10 & 7);`

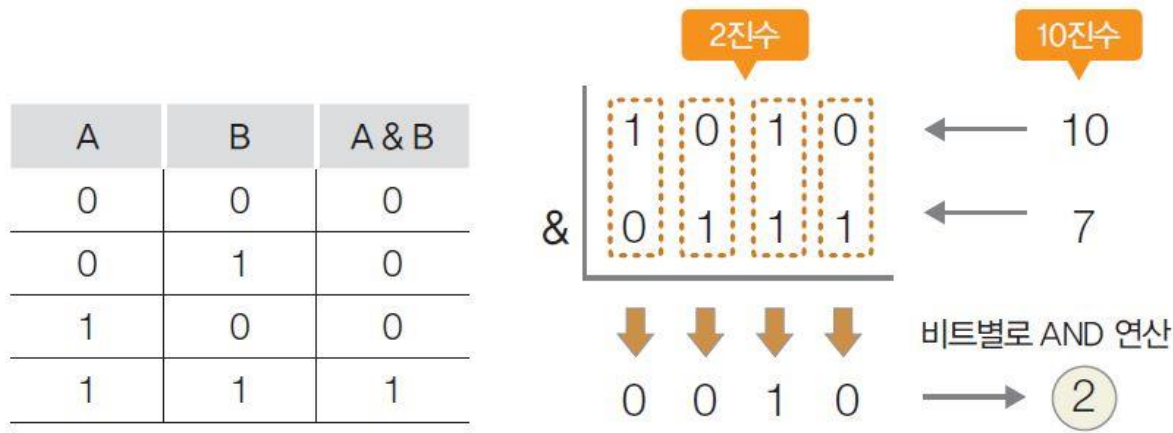


그림 4-6 비트 논리곱의 예

```
01 #include <stdio.h>
02
03 int main()
04 {
05     printf(" 10 & 7 = %d \n", 10 & 7);
06     printf(" 123 & 456 = %d \n", 123 & 456);
07 }
```

---10과 7의 비트 논리곱을 수행한다.

---123과 456의 비트 논리곱을 수행한다.

비트 연산자 - 논리합(|)

- 10진수를 2진수로 변환한 후 각 비트에 OR 연산 수행
`printf("%d\n", 10 | 7);`

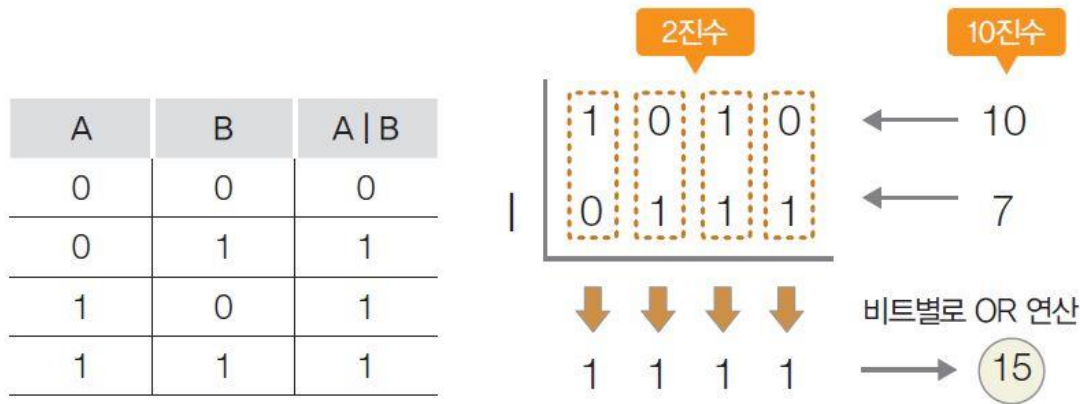


그림 4-7 비트 논리합의 예

```
01 #include <stdio.h>
02
03 int main()
04 {
05     printf(" 10 | 7 = %d \n", 10 | 7);
06     printf("123 | 456 = %d \n", 123 | 456);
07 }
```

---10과 7의 비트 논리합을 수행한다.

---123과 456의 비트 논리합을 수행한다.

비트 연산자 - 배타적 논리합(^)

- 1이 한 개만 있어야 참(1), 그 외는 거짓(0)
`printf("%d\n", 10 ^ 7);`

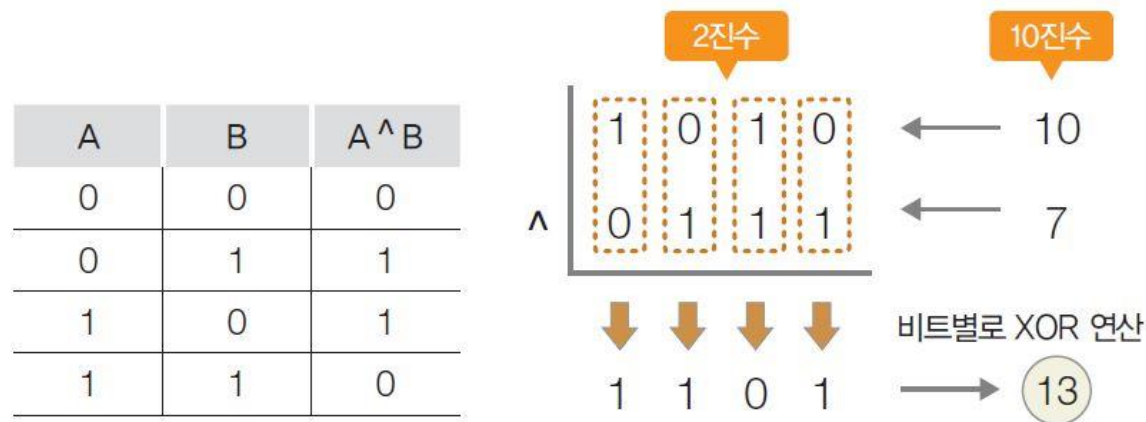


그림 4-8 비트 배타적 논리합의 예

```
01 #include <stdio.h>
02
03 int main()
04 {
05     printf(" 10 ^ 7 = %d \n", 10 ^ 7);
06     printf(" 123 ^ 456 = %d \n", 123 ^ 456);
07 }
```

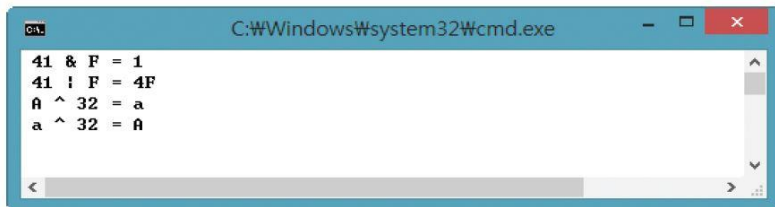
---10과 7의 비트 배타적 논리합을 수행한다.

---123과 456의 비트 배타적 논리합을 수행한다.

[4-11] 비트 연산에 마스크를 사용한 예

```
01 #include <stdio.h>
02
03 int main()
04 {
05     char a = 'A', b, c;
06     char mask = 0x0F; ----마스크 값(0000 11112)을 설정한다.
07
08     printf(" %X & %X = %X \n", a, mask, a & mask); ----'A'와 0x0F의 논리곱을 수행한다.
09     printf(" %X | %X = %X \n", a, mask, a | mask); ----'A'와 0x0F의 논리합을 수행한다.
10
11     mask = 'a' - 'A'; ----'a'와 'A'의 차이는 32이다.
12
13     b = ____①____ ----'A'와 마스크(32)의 배타적 논리곱을 수행한다.
14     printf(" %c ^ %d = %c \n", a, mask, b);
15     a = ____②____ ----'a'와 마스크(32)의 배타적 논리곱을 수행한다.
16     printf(" %c ^ %d = %c \n", b, mask, a);
17 }
```

실행 결과 ▼



```
C:\Windows\system32\cmd.exe
41 & F = 1
41 | F = 4F
a ^ 32 = a
a ^ 32 = a
```

실행 결과 ▼

[4-11] 비트 연산에 마스크를 사용한 예

- 6행 : 마스크(mask) 값 선언, 16진수 $0x0F_{16}$
- 8행 : 마스크를 사용한 비트 논리곱 결과

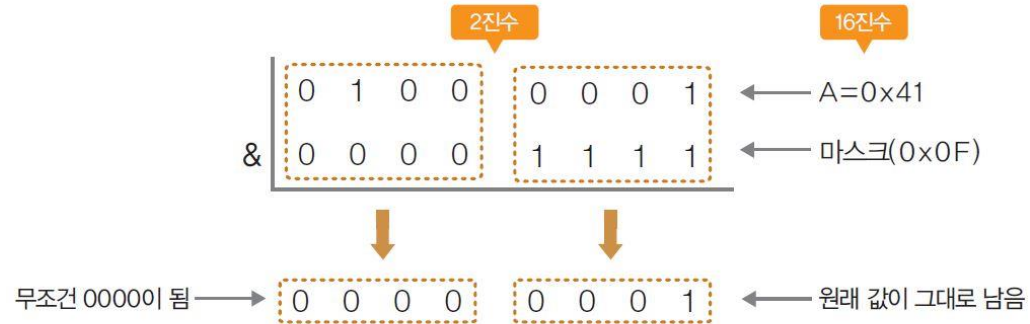


그림 4-9 마스크 0x0F를 사용한 비트 논리곱의 예

- 9행 : 마스크를 사용한 비트 논리합 결과
- 11행 : 마스크 값 선언
- 13행, 15행 : 마스크를 사용하여 'a', 'A'의 비트 배타적 논리합 수행

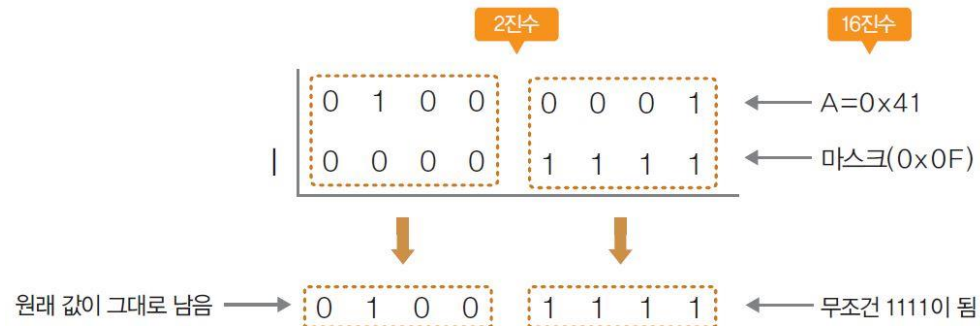


그림 4-10 마스크 0x0F를 사용한 비트 논리합의 예

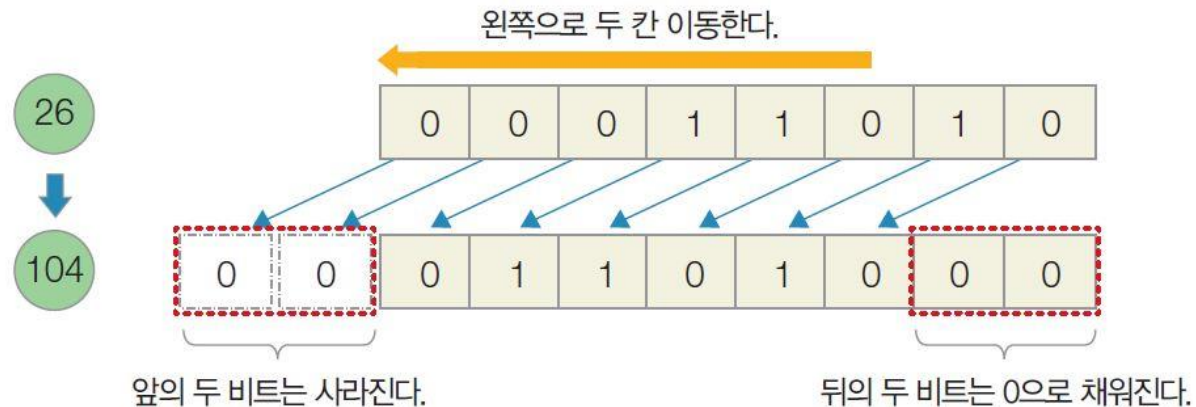
비트 연산자 - 부정(~)

- 두 수에 대한 연산이 아니라 비트 하나의 값을 반대로 만듦
 - 어떤 수의 음숫값(-)을 찾을 때 사용
 - 2의 보수(음수) = { 1의 보수(각 비트의 값을 반전시킨 값) } + 1

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a = 12345;
06
07     printf(" %d \n", ~a + 1); ---2의 보수(a 값)를 구한다.
08 }
```

비트 연산자 - 왼쪽 시프트(<<)

- 나열된 비트를 왼쪽으로 시프트(shift)하는 연산자
 - 왼쪽 시프트를 할 때마다 $2^n(2^1, 2^2, 2^3...)$ 을 곱한 효과
 - 예시, 26_{10} 을 왼쪽으로 시프트 연산
 - $0001\ 1010_2$ 로 변환한 후 비트 이동

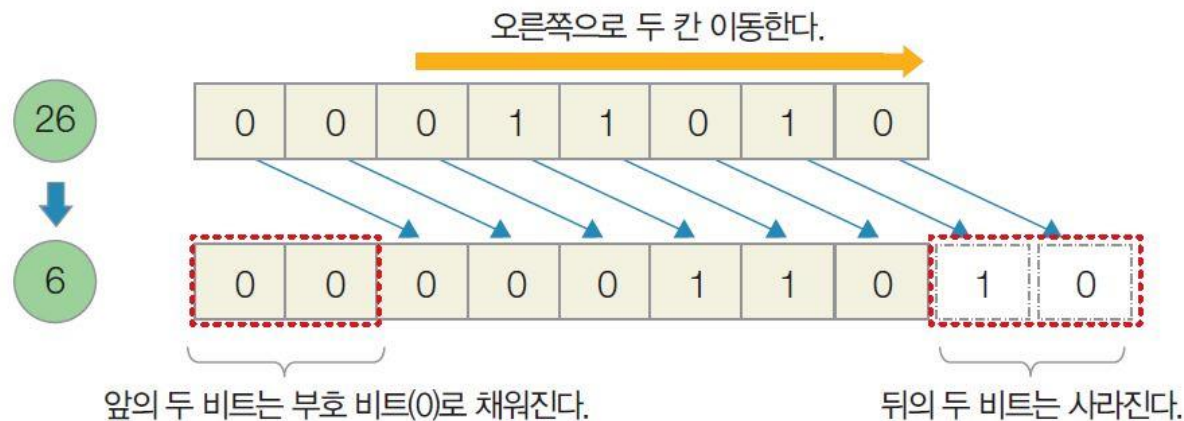


```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a = 10;
06     printf ("%d를 왼쪽 1회 시프트하면 %d이다.\n", a, a<<1);
07     printf ("%d를 왼쪽 2회 시프트하면 %d이다.\n", a, a<<2);
08     printf ("%d를 왼쪽 3회 시프트하면 %d이다.\n", a, a<<3);
```

---원쪽으로 시프트 한 결과를 출력한다.

비트 연산자 - 오른쪽 시프트(>>)

- 나열된 비트를 오른쪽으로 시프트(shift)하는 연산자
 - 왼쪽 시프트를 할 때마다 $2^n(2^1, 2^2, 2^3...)$ 을 **나눈** 효과
 - 예시, 26_{10} 을 오른쪽으로 시프트 연산



```
01. #include <stdio.h>
02
03 int main()
04 {
05     int a = 10;
06     printf ("%d.를 오른쪽 1회 시프트하면 %d 이다.\n", a, a>>1);
07     printf ("%d.를 오른쪽 2회 시프트하면 %d 이다.\n", a, a>>2);
08     printf ("%d.를 오른쪽 3회 시프트하면 %d 이다.\n", a, a>>3);
09     printf ("%d.를 오른쪽 4회 시프트하면 %d 이다.\n", a, a>>4);
```

--- 오른쪽으로 시프트 한 결과를 출력한다.

연산자 우선순위

표 4-6 연산자 우선순위

우선순위	연산자	명칭	순위가 같을 경우 진행 방향
1	() [] . ->	1차 연산자	➡
2	+ - ++ -- ~ ! * &	단항 연산자(변수 또는 상수 앞에 붙음)	⬅
3	* / %	산술 연산자	➡
4	+ -	산술 연산자	➡
5	<< >>	비트 시프트 연산자	➡
6	<<= >>=	비교 연산자	➡
7	== !=	동등 연산자	➡
8	&	비트 연산자	➡
9	^	비트 연산자	➡
10		비트 연산자	➡
11	&&	논리 연산자	➡
12		논리 연산자	➡
13	?:	삼항 연산자	➡
14	= += -= *= /= %= &= ^= = <<= >>=	대입 연산자	⬅
15	,	coma 연산자	➡

Today

1. 지난 시간 실습 복습
2. 연산자1
3. 연산자2
4. 실습

[실습] 입력된 두 실수의 산술 연산

```
01 #include <stdio.h>
02
03 int main()
04 {
05     float a, b;
06     float result;
07
08     printf("첫번째 계산할 값을 입력하세요 == > ");
09     scanf("%f", &a);
10     printf("두번째 계산할 값을 입력하세요 == > ");
11     scanf("%f", &b);
12
13     result = a + b;
14     printf(" %5.2f + %5.2f = %5.2f \n", a, b, result);
15     result = a - b;
16     printf(" %5.2f - %5.2f = %5.2f \n", a, b, result);
17     result = a * b;
18     printf(" %5.2f * %5.2f = %5.2f \n", a, b, result);
19     result = a / b;
20     printf(" %5.2f / %5.2f = %5.2f \n", a, b, result);
21     result = (int)a % (int)b;
22     printf(" %d %% %d = %d \n", (int)a, (int)b, (int)result);
23 }
```

----실수형 변수를 선언한다.

---실수를 입력받는다.

---실수를 입력받는다.

---실수의 덧셈이다.

---실수의 뺄셈이다.

---실수의 곱셈이다.

---실수의 나눗셈이다.

---나머지 연산을 위해 실수를 정수로 강제 형 변환한다.

[실습1] 동전 교환해주기

- 입력된 돈의 액수만큼 500원, 100원, 50원, 10원짜리 동전으로 교환해주는 프로그램 작성하고 다음과 같이 출력되도록 하기
 - 단, 고액 동전부터 먼저 바꿔줄것!!

교환할 돈의 액수를 입력하세요.

2791

500원 5개

100원 2개

50원 1개

10원 4개

바꾸지 못한 잔돈 1원

[실습1] 동전 교환 프로그램

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int money, c500, c100, c50, c10;    ---입력한 돈과 각 동전의 개수를 저장할 변수이다.
06
07     printf(" ## 교환할 돈은 ? ");
08     scanf("%d", &money);    ---교환할 액수를 입력한다
09
10     c500 = money / 500;    ---500원짜리 동전의 개수를 계산한다.
11     money = money % 500;    ---500원짜리로 바꾼 후 나머지 금액이다.
12
13     c100 = money / 100;    ---100원짜리 동전의 개수를 계산한다.
14     money = money % 100;    ---100원짜리로 바꾼 후 나머지 금액이다.
15
16     c50 = money / 50;    ---50원짜리 동전의 개수를 계산한다.
17     money = money % 50;    ---50원짜리로 바꾼 후 나머지 금액이다.
18
19     c10 = money / 10;    ---10원짜리 동전의 개수를 계산한다.
20     money = money % 10;    ---10원짜리로 바꾼 후 나머지 금액이다.
21
```

[실습1] 동전 교환 프로그램

```
22  printf("\n 오백원짜리 == > %d 개 \n", c500);
23  printf(" 백원짜리 == > %d 개 \n", c100);
24  printf(" 오십원짜리 == > %d 개 \n", c50);
25  printf(" 십원짜리 == > %d 개 \n", c10);
26  printf(" 바꾸지 못한 잔돈 == > %d 원 \n", money); ---바꾸지 못한 나머지 돈은 money에 들어 있다.
27 }
```

[실습2] 3의 배수 판별하기

- 입력받은 숫자가 3의 배수인지 판별하는 프로그램을 작성하고 다음과 같이 출력하기
 - 입력받은 수를 3으로 나눈 나머지가 0이면 3의 배수로 판별함
- 실행 결과

정수를 입력하세요.

4

3의 배수가 아닙니다.

정수를 입력하세요.

15

3의 배수가 맞습니다.

[실습2] 3의 배수 판별하기

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int a;
06      printf("정수를 입력하세요.\n");
07      scanf("%d", &a);
08      if (a % 3 == 0)
09          printf("3의 배수가 맞습니다.\n");
10      else
11          printf("3의 배수가 아닙니다.\n");
12  }
```

[실습3] 윤년판별하기

- 입력받은 연도가 윤년인지 판별하는 프로그램을 작성하고 다음과 같이 출력되도록 하기
 - 윤년은 입력한 연도가 4로 나누어 떨어지고 100으로는 나누어 떨어지지 않아야 한다.
 - 또는 400으로 나누어 떨어지면 윤년으로 판별한다.

- 실행결과

년도를 입력하세요. : 2019
2019 년은 윤년이 아닙니다.

년도를 입력하세요. : 2020
2020 년은 윤년입니다.

[실습3] 윤년 계산 프로그램

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int year;
06
07     printf("년도를 입력하세요. : ");
08     scanf("%d", &year);
09
10     if ( ((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0) )
11         printf ("%d 년은 윤년입니다. \n", year);
12     else
13         printf ("%d 년은 윤년이 아닙니다. \n", year);
14 }
```

---계산할 연도를 입력한다.

---윤년은 입력한 연도가 4로 나누어 떨어지고 100으로는 나누어 떨어지지 않아야 한다. 또는 400으로 나누어 떨어져도 된다.

Summary

- 연산자
 - 산술연산자, 대입연산자, 증감연산자,
 - 관계연산자, 논리연산자, 비트연산자
- 모호함을 피하기 위해서 연산자 사용 시 () 를 사용하는 것을 권장
 - $y = x * 3 + 2$ /*same as $y = (x * 3) + 2$ */
 - $x != 0 \ \&\& \ y == 0$ /*same as $(x != 0) \ \&\& \ (y == 0)$ */
 - $d = c >= '0' \ \&\& \ c <= '9'$ /*same as $d = (c >= '0') \ \&\& \ (c <= '9')$ */