



인하공업전문대학
INHA TECHNICAL COLLEGE

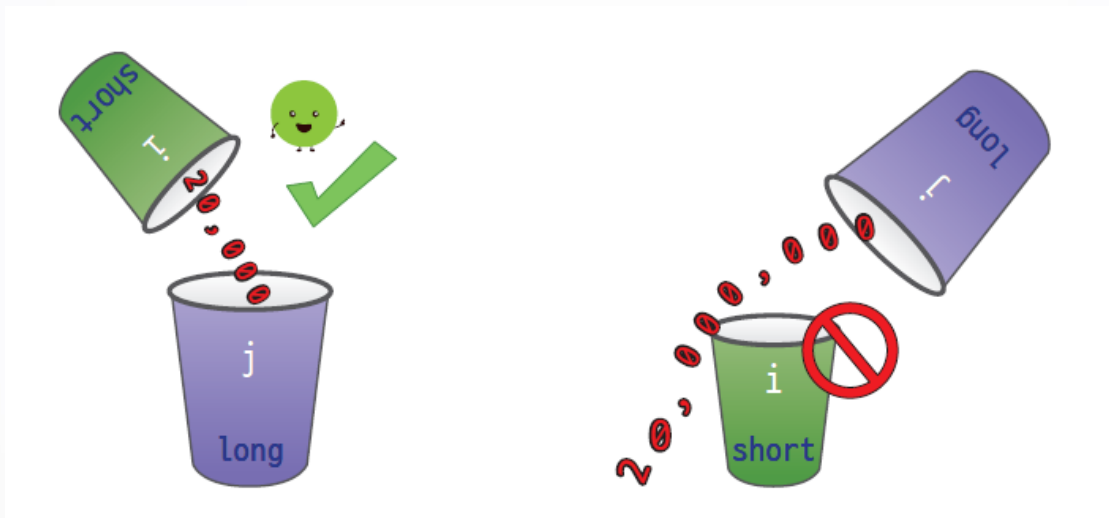
C 프로그래밍

5주차

인하공업전문대학 컴퓨터 정보과
김한결 강사

형 변환

- 형 변환(type conversion)이란 실행 중에 데이터의 타입을 변경하는 것이다

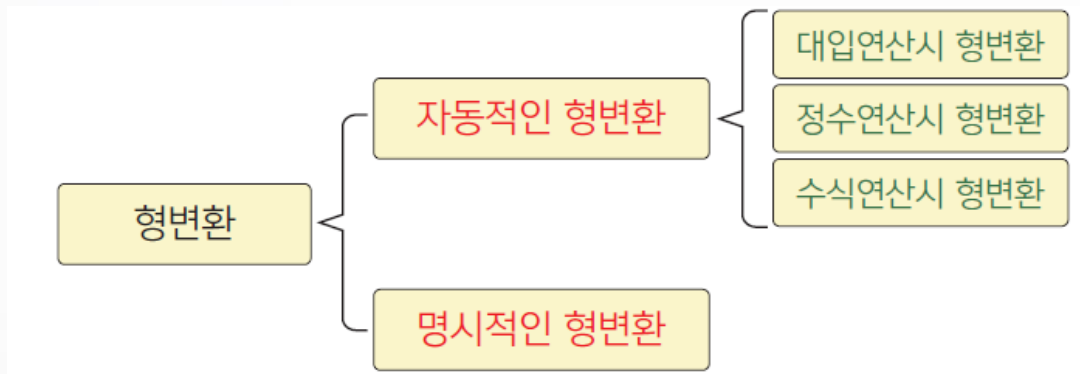


형변환을 잘못하면 데이터의 일부가 사라질 수도 있기 때문에 주의하여야 한다.



형 변환

- 연산시에 데이터의 유형이 변환되는 것



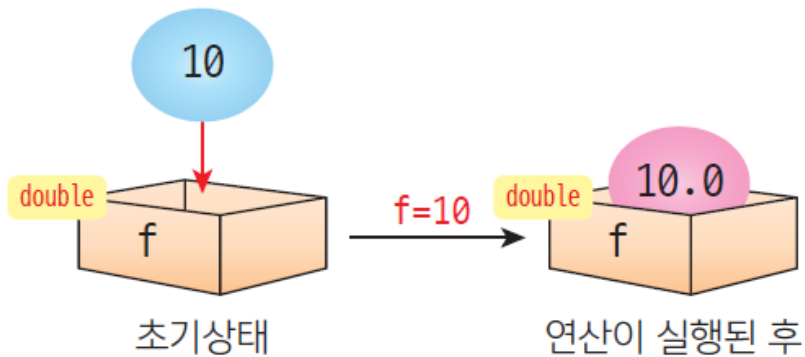
변수의 타입이 변경되는 것이 아니고 변수에 저장되는 데이터의 타입이 변경됩니다.



대입 연산시의 자동적인 형변환

- 올림 변환

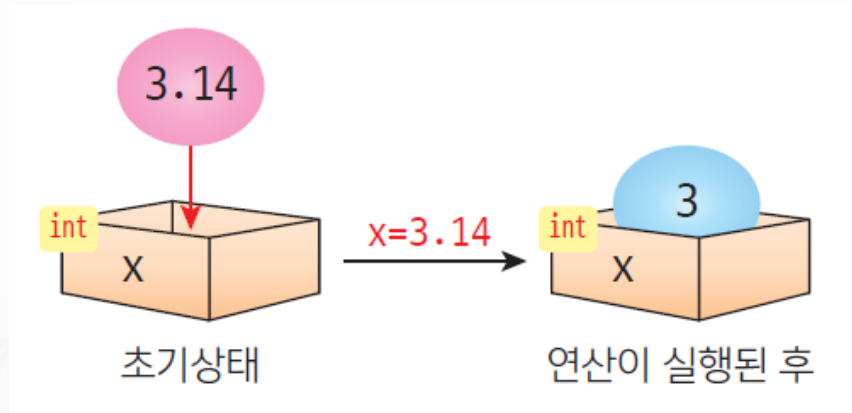
```
double f;  
f = 10 ;    // f에는 10.00이 저장된다.
```



대입 연산시의 자동적인 형변환

- 내림변환

```
int i;  
i = 3.141592;           // i에는 3이 저장된다.
```



정수형끼리 형변환

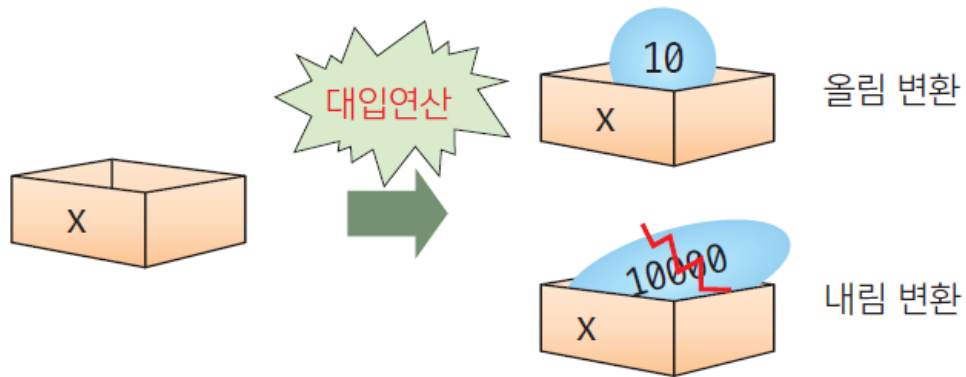
```
char x;
```

```
x = 10;
```

```
// OK
```

```
x = 10000;
```

```
// 상위 바이트는 없어진다.
```



```
#include <stdio.h>
int main(void)
{
    char c;
    int i;
    float f;

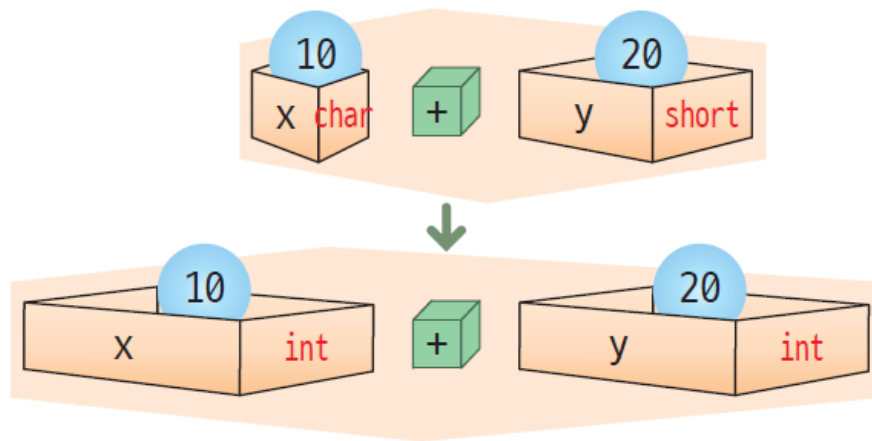
    c = 10000;           // 내림 변환
    i = 1.23456 + 10;    // 내림 변환
    f = 10 + 20;         // 올림 변환
    printf("c = %d, i = %d, f = %f \n", c, i, f);
    return 0;
}
```

c:\W...Wconvert1.c(10) : warning C4305: '=' : 'int'에서 'char'(으)로 잘립니다.
c:\W...Wconvert1.c(11) : warning C4244: '=' : 'double'에서 'int'(으)로 변환하면서 데이터가 손실될 수 있습니다.

c=16, i=11, f=30.000000

정수 연산시의 자동적인 형변환

- 정수 연산시 **char**형이나 **short**형의 경우, 자동적으로 **int**형으로 변환하여 계산한다.

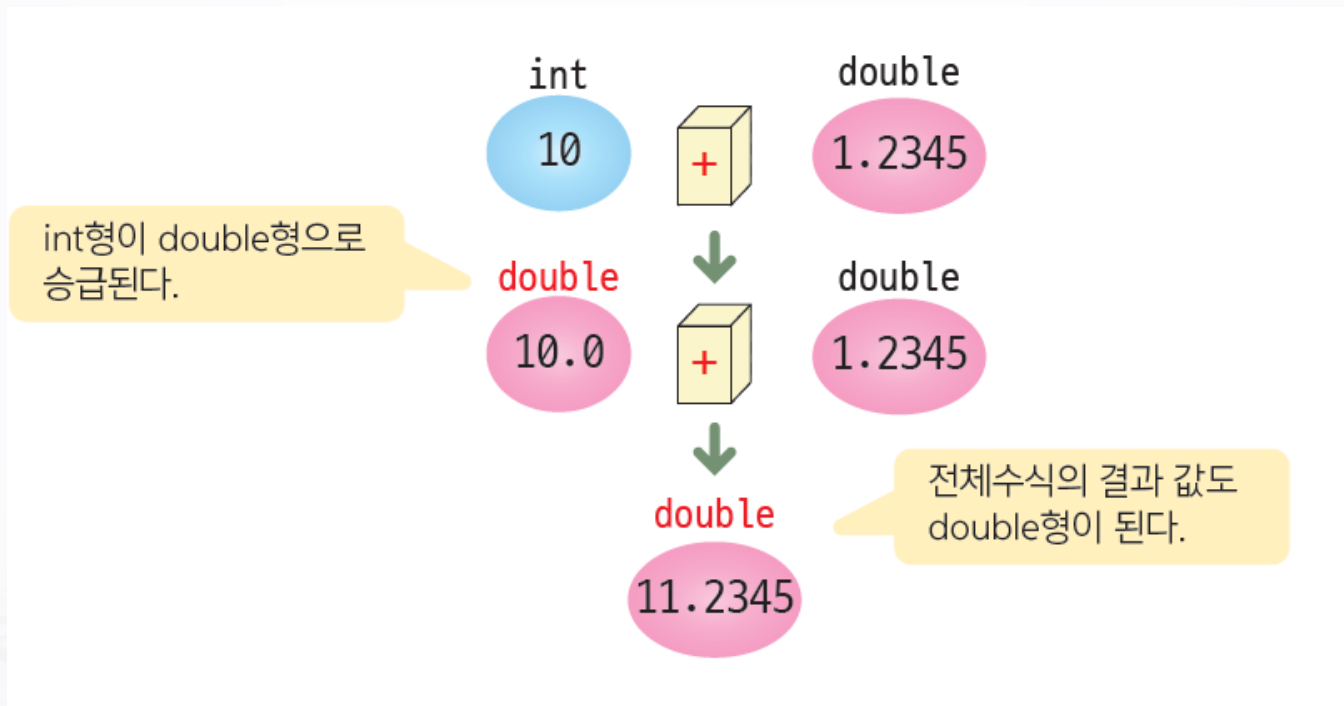


char나 short형은 int형으로
통일하여서 처리합니다.



수식에서의 자동적인 형변환

- 서로 다른 자료형이 혼합하여 사용되는 경우, 더 큰 자료형으로 통일된다.



명시적인 형변환

Syntax

형변환

예

자료형

(int)1.23456

(double) x

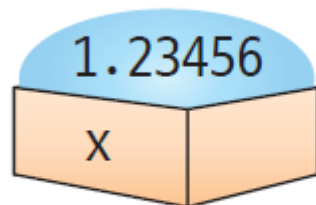
(long) (x+y)

수식

// int형으로 변환

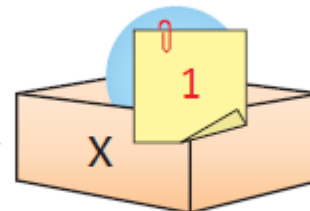
// double형으로 변환

// long형으로 변환



초기상태

(int)



연산이 실행된 후

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
```

```
int main(void)
{
```

```
    int i;
    double f;
```

```
    f = 5 / 4;
```

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

```
    f = (double)5 / 4;
    printf("%f\n", f);
```

```
    f = 5.0 / 4;
    printf("%f\n", f);
```

5/4는 1이 되고 이것이 1.0이 된다.

5가 5.0으로 되어서 전체 결과가 1.25가 된다.

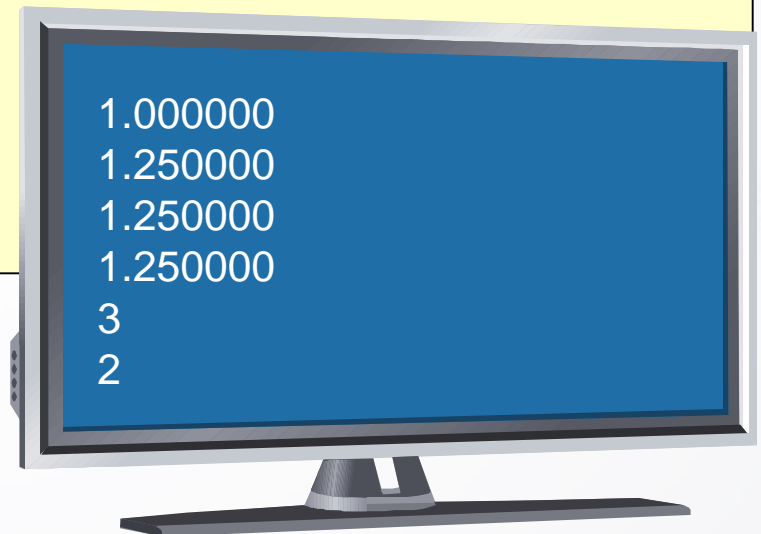
```
f = (double)5 / (double)4;
printf("%f\n", f);

i = 1.3 + 1.8;
printf("%d\n", i);

i = (int)1.3 + (int)1.8;

printf("%d\n", i);
return 0;
}
```

1.3이 1이 되고
1.8도 1이
되어서 최종
결과는 2가
된다.



중간 점검

1. 내림 변환과 올림 변환을 설명하라.
2. `int`형 변수 `x`를 `double`형으로 형변환하는 문장을 써보라.
3. 하나의 수식에 정수와 부동소수점수가 섞여 있으면 어떻게 되는가?



우선 순위

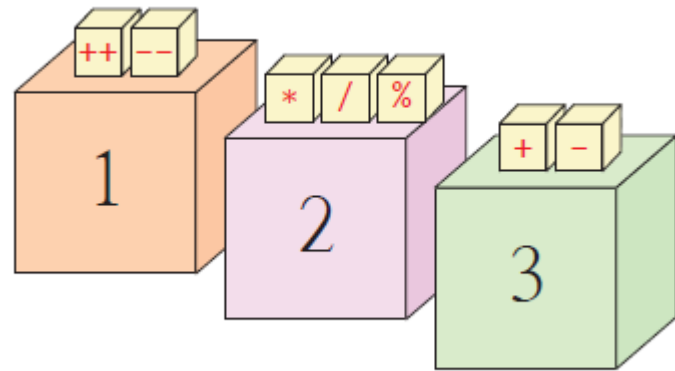
- 어떤 연산자를 먼저 계산할 것인지에 대한 규칙

$$x + y * z$$

Diagram illustrating operator precedence for the expression $x + y * z$. A bracket labeled ① groups $y * z$, indicating multiplication is performed first. A second bracket labeled ② groups the entire expression $x + (y * z)$, indicating addition is performed second.

$$(x + y) * z$$

Diagram illustrating operator precedence for the expression $(x + y) * z$. A bracket labeled ① groups $x + y$, indicating addition is performed first. A second bracket labeled ② groups the entire expression $(x + y) * z$, indicating multiplication is performed second.



우선 순위

우선순위	연산자	설명	결합성
1	++ --	후위 증감 연산자	→ (좌에서 우)
	()	함수 호출	
	[]	배열 인덱스 연산자	
	.	구조체 멤버 접근	
	->	구조체 포인터 접근	
	(type){list}	복합 리터럴(C99 규격)	
2	++ --	전위 증감 연산자	← (우에서 좌)
	+ -	양수, 음수 부호	
	! ~	논리적인 부정, 비트 NOT	
	(type)	형변환	
	*	간접 참조 연산자	
	&	주소 추출 연산자	
	sizeof	크기 계산 연산자	
	_Alignof	정렬 요구 연산자 (C11 규격)	

3	* / %	곱셈, 나눗셈, 나머지	→ (좌에서 우)
4	+ -	덧셈, 뺄셈	
5	<< >>	비트 이동 연산자	
6	< <=	관계 연산자	
	> >=	관계 연산자	
7	== !=	관계 연산자	
8	&	비트 AND	
9	^	비트 XOR	
10		비트 OR	
11	&&	논리 AND 연산자	
12		논리 OR 연산자	
13	?:	삼항 조건 연산자	← (우에서 좌)
14	=	대입 연산자	
	+= -=	복합 대입 연산자	
	*= /= %=	복합 대입 연산자	
	<<= >>=	복합 대입 연산자	
	&= ^= =	복합 대입 연산자	
15	,	coma 연산자	→ (좌에서 우)

우선 순위의 일반적인 지침

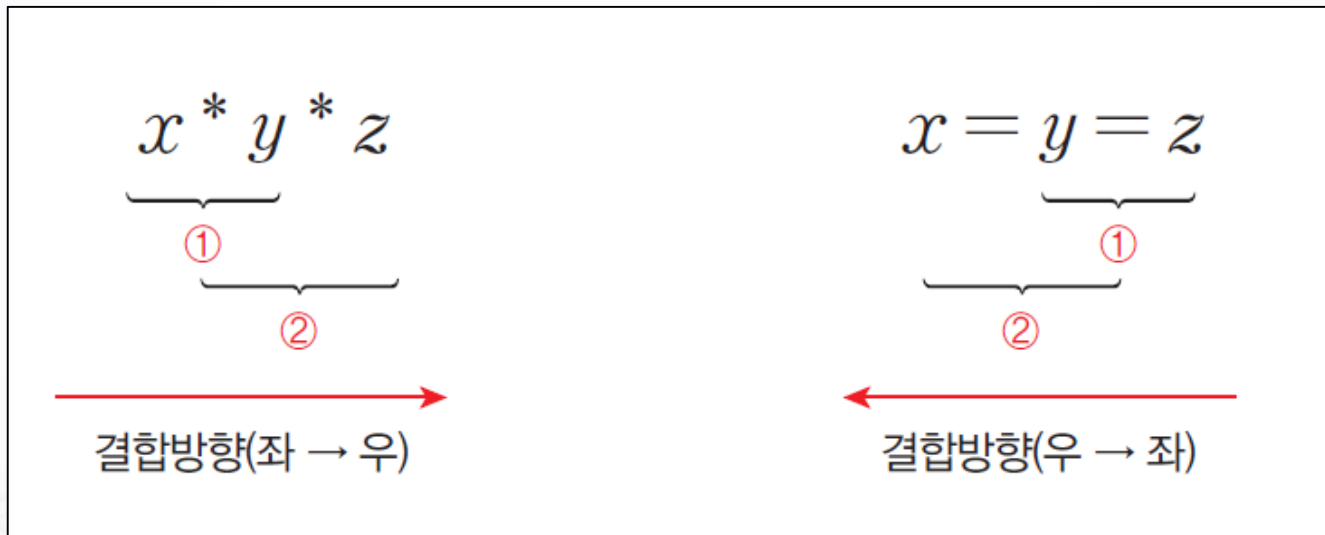
- 콤마 < 대입 < 논리 < 관계 < 산술 < 단항
- 괄호 연산자는 가장 우선순위가 높다.
- 모든 단항 연산자들은 이항 연산자들보다 우선순위가 높다.
- 콤마 연산자를 제외하고는 대입 연산자가 가장 우선순위가 낮다.
- 연산자들의 우선 순위가 생각나지 않으면 괄호를 이용
 - `(x <= 10) && (y >= 20)`
- 관계 연산자나 논리 연산자는 산술 연산자보다 우선순위가 낮다.
 - `x + 2 == y + 3`
- 관계 연산자는 논리 연산자보다 우선 순위가 높다. 따라서 다음과 같은 문장은 안심하고 사용하라.
 - `x > y && z > y` // `(x > y) && (z > y)`와 같다.

우선 순위의 일반적인 지침

- 논리 연산자 중에서 && 연산자가 || 연산자보다 우선 순위가 높다는 것에 유의하여야 한다.
 - $x < 5 \parallel x > 10 \&\& x > 0$ // $x < 5 \parallel (x > 10 \&\& x > 0)$ 와 같다
- 가끔 연산자들의 계산 순서가 상당히 혼돈스러운 경우도 있다. $x * y + w * y$ 에서 $x * y$ 와 $w * y$ 중에서 어떤 것이 먼저 계산될지는 불명확하다.

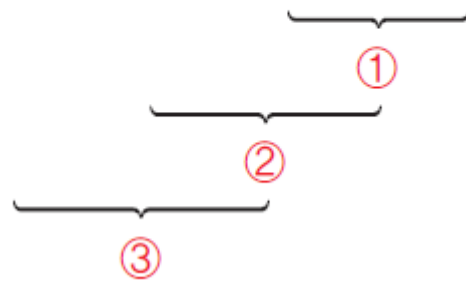
결합 규칙

- 만약 같은 우선순위를 가지는 연산자들이 여러 개가 있으면 어떤 것을 먼저 수행하여야 하는가의 규칙

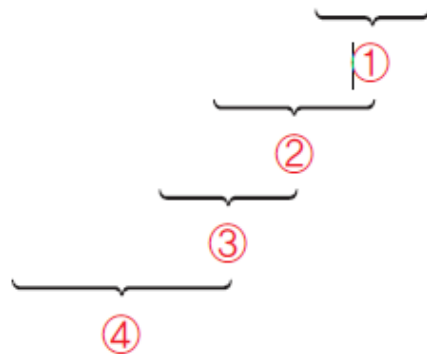


결합규칙의 예

`x = y = z = 5;`



`y = - ++ --x;`



결합규칙의 예

$$y = a \% b / c + d * (e - f);$$

The diagram illustrates the evaluation order of the expression $y = a \% b / c + d * (e - f);$ using numbered brackets:

- ①: $(e - f)$ (sub-expression inside parentheses)
- ②: $a \% b$ (sub-expression)
- ③: $a \% b / c$ (sub-expression)
- ④: $d * (e - f)$ (sub-expression)
- ⑤: $a \% b / c + d * (e - f)$ (sub-expression)
- ⑥: $y = a \% b / c + d * (e - f);$ (the entire statement)

```
#include <stdio.h>
int main(void)
{
    int x=0, y=0;
    int result;

    result = 2 > 3 || 6 > 7;
    printf("%d", result);

    result = 2 || 3 && 3 > 2;
    printf("%d", result);

    result = x = y = 1;
    printf("%d", result);

    result = - ++x + y--;
    printf("%d", result);

    return 0;
}
```



중간 점검

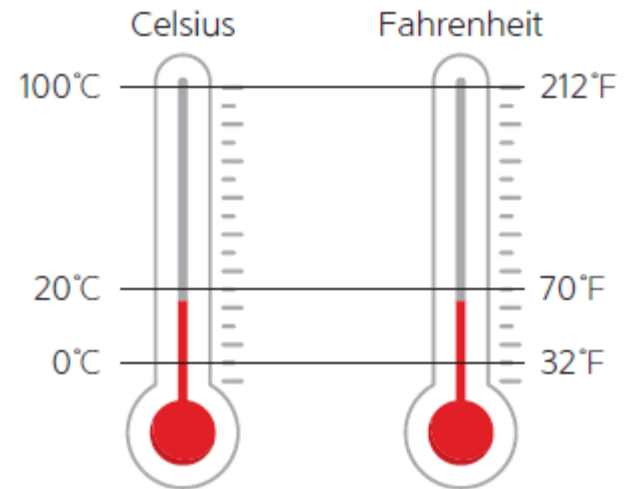
1. 연산자 중에서 가장 우선 순위가 낮은 연산자는 무엇인가?
2. 논리 연산자인 **&&**과 **||** 중에서 우선 순위가 더 높은 연산자는 무엇인가?
3. 단항 연산자와 이항 연산자 중에서 어떤 연산자가 더 우선 순위가 높은가?
4. 관계 연산자와 산술 연산자 중에서 어떤 연산자가 더 우선 순위가 높은가?



Mini Project: 화씨 온도를 섭씨로 바꾸기

- 화씨 온도를 섭씨 온도로 바꾸는 프로그램을 작성하여 보자.

$$\text{섭씨온도} = \frac{5}{9}(\text{화씨온도} - 32)$$




```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    double f_temp;
```

```
    double c_temp;
```

```
    printf("화씨 온도를 입력하시오");
```

```
    scanf("%lf", &f_temp);
```

```
    c_temp = 5 / 9 * (f_temp - 32);
```

```
    printf("섭씨 온도는 %f입니다", c_temp);
```

```
    return 0;
```

```
}
```

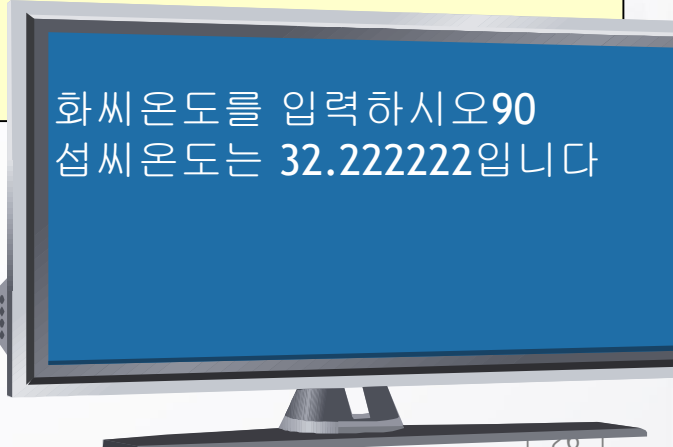
5/9가 먼저 계산되어서
0이 된다.

화씨 온도를 입력하시오: 90
섭씨 온도는 0.000000입니다.

```
#include <stdio.h>
int main(void)
{
    double f_temp;
    double c_temp;

    printf("화씨 온도를 입력하시오");
    scanf("%lf", &f_temp);
    c_temp = 5.0 / 9.0 * (f_temp - 32);
    printf("섭씨 온도는 %f입니다", c_temp);

    return 0;
}
```



화씨 온도를 입력하시오 90
섭씨 온도는 32.222222입니다

도전문제

1. 위에서 제시한 방법 외에 다른 방법은 없을까?
2. $((\text{double})5 / (\text{double})9) * (f_temp - 32);$ 가 되는지 확인하여 보자.
3. $((\text{double})5 / 9) * (f_temp - 32);$ 가 되는지 확인하여 보자.



이번 장에서 학습할 내용



- 조건문이란?
- if 문
- if, else 문
- 중첩 if 문
- switch 문
- break문
- continue문
- goto문

필요에 따라서 조건
이 만족되면 문장의
실행 순서를 변경할
수 있는 기능이 제
공됩니다.

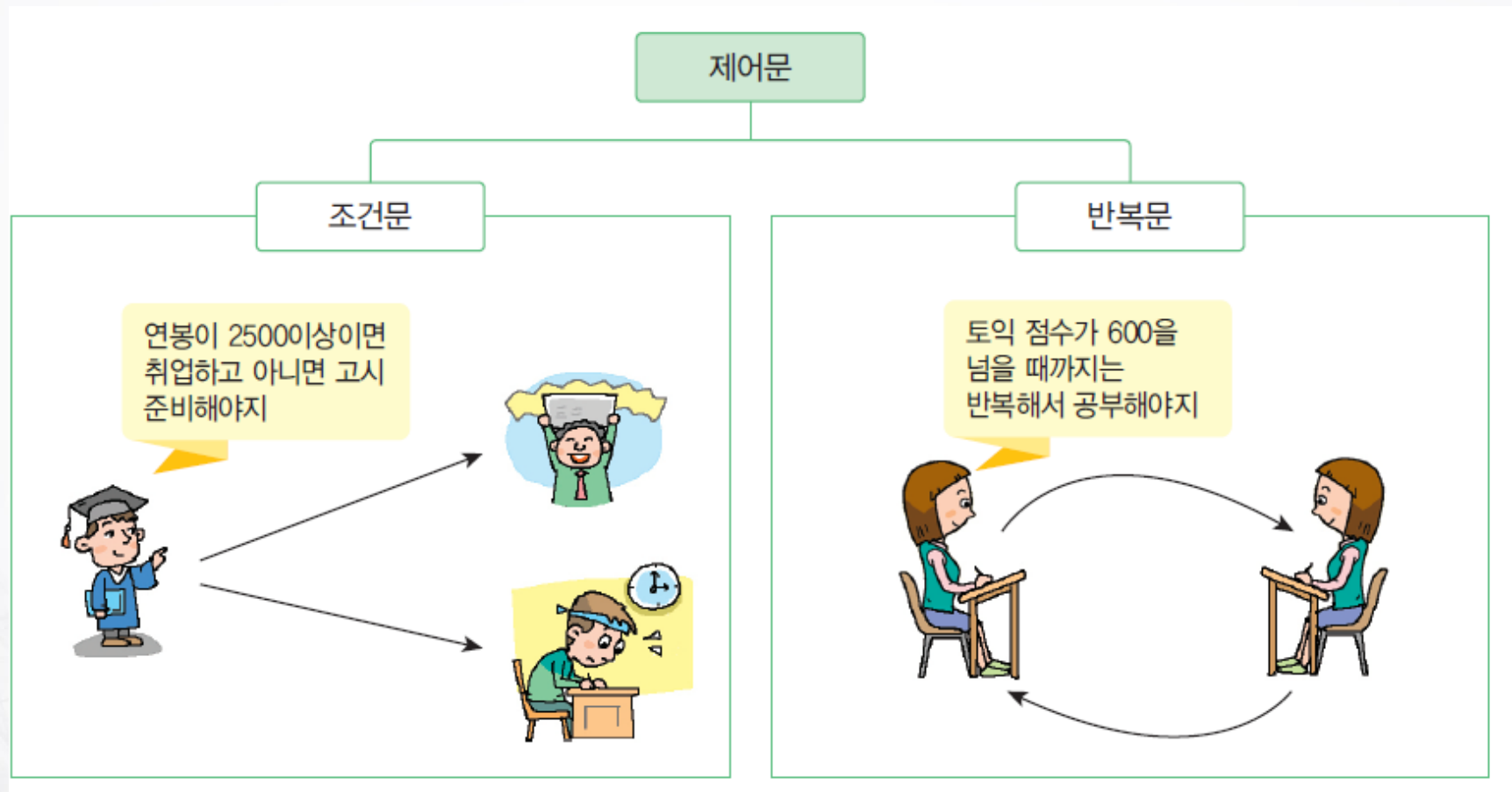


조건문

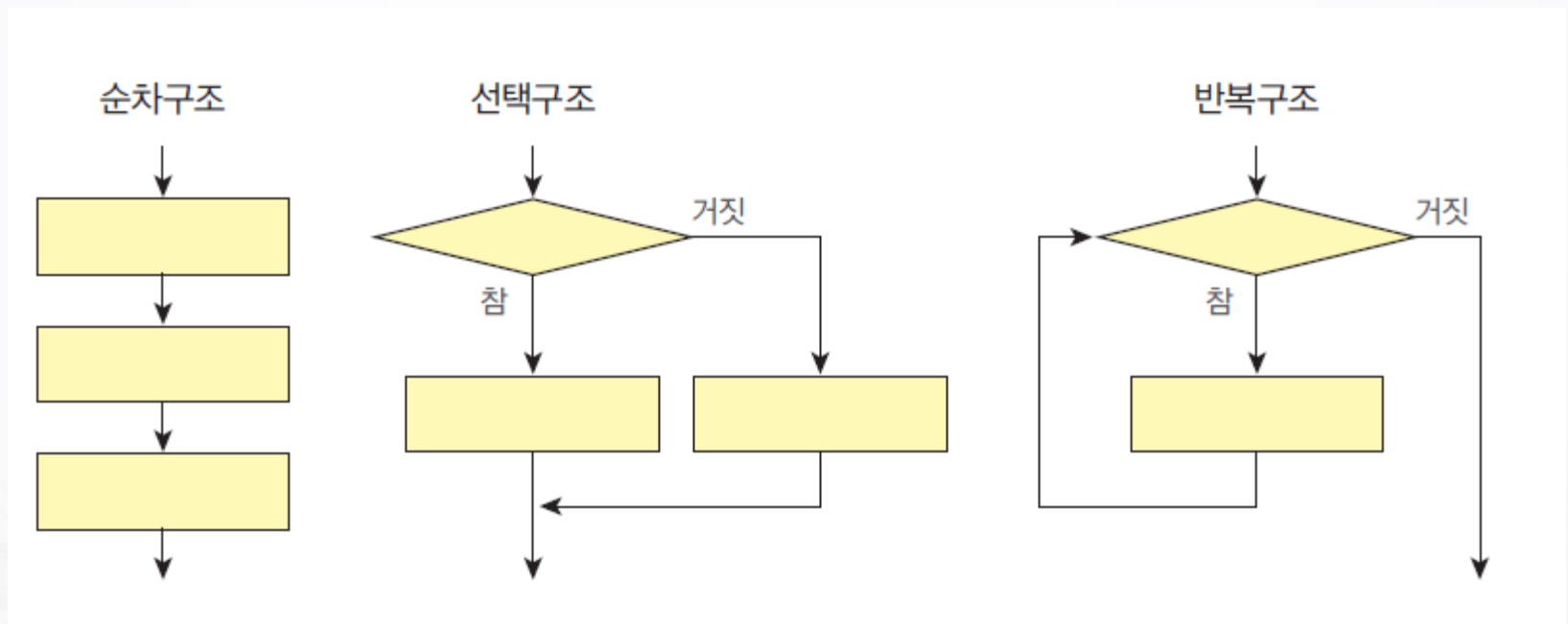
- 만약 프로그램에 선택 구조가 없다면 프로그램은 항상 동일한 동작만을 되풀이 할 것이다.



제어문

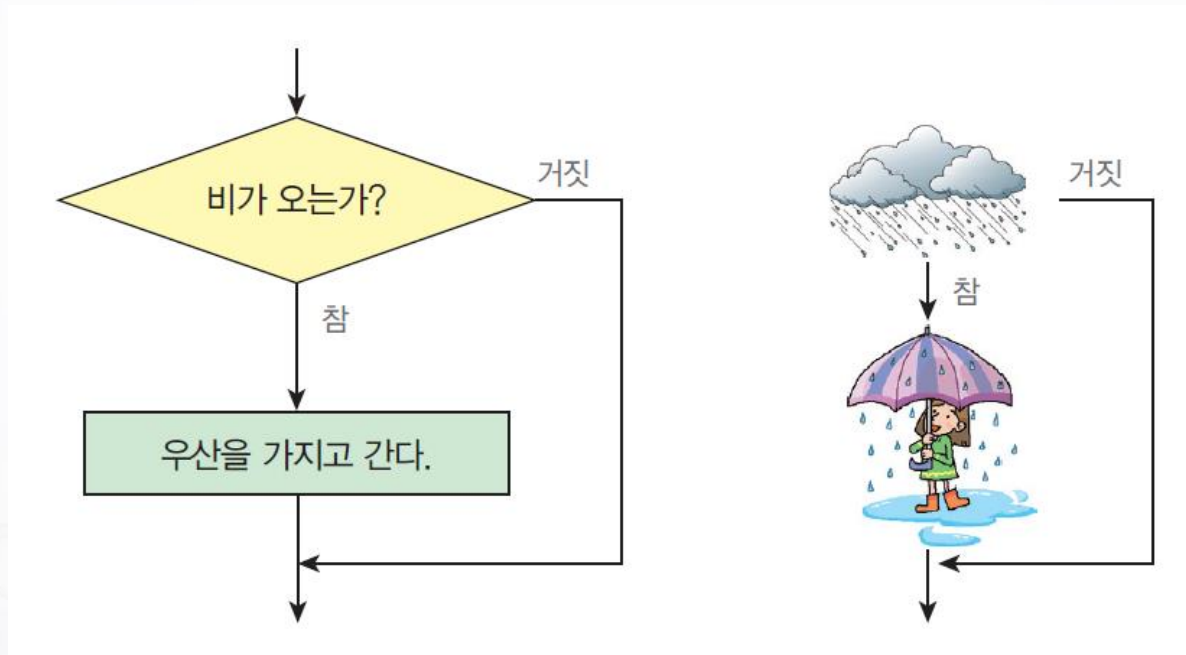


3가지의 제어구조



if문

- 일상생활에서도 조건에 따라서 결정을 내려야 하는 경우는 많이 있다.



if문의 구조

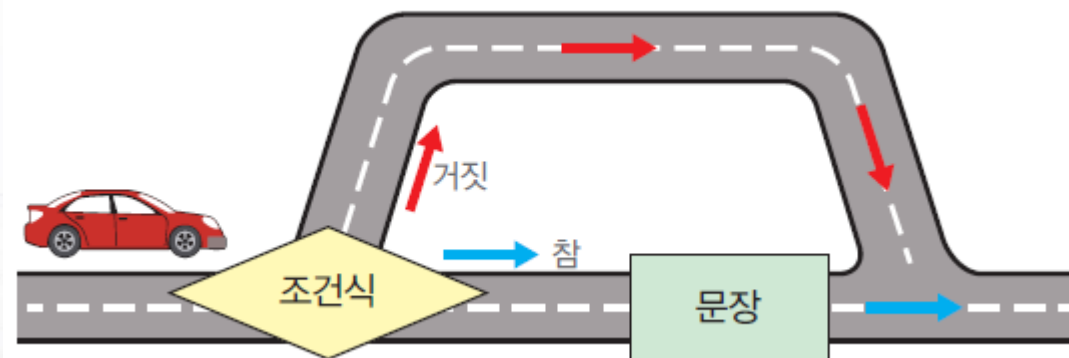
Syntax if 문

예

```
if( number > 0 )  
    printf("양수입니다.\n");
```

조건식

조건식이 참인 경우에만 문장이 실행된다.



if문의 예

number 가 0보다 크면


```
if( number > 0 )  
    printf("양수입니다\n");
```

“양수입니다”를 출력한다.

```
if ( temperature < 0 )  
    printf("현재 영하입니다.\n");           // 조건이 참일 때만 실행  
  
printf("현재 온도는 %도 입니다.\n", temperature); // 항상 실행
```

if 문이 끝나면 if 문 다음 문장이 실행된다.

예제



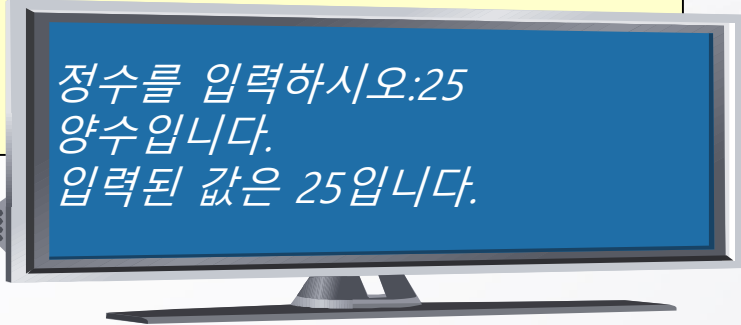
```
#include <stdio.h>
int main(void)
{
    int number;

    printf("정수를 입력하시오:");
    scanf("%d", &number);

    if( number > 0 )
        printf("양수입니다.");

    printf("입력된 값은 %d입니다.", number);

    return 0;
}
```



정수를 입력하시오:25
양수입니다.
입력된 값은 25입니다.

예제

```
// if 문을 사용하여 절대값을 구하는 프로그램  
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int number;
```

```
    printf("정수를 입력하시오:");  
    scanf("%d", &number);
```

```
    if( number < 0 )  
        number = -number;
```

```
    printf("절대값은 %d 입니다.\n", number);
```

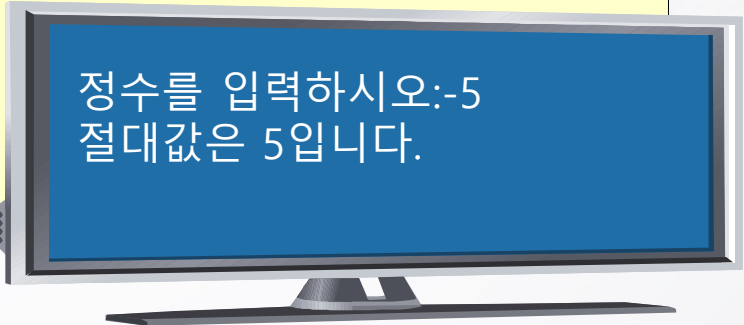
```
    return 0;
```

```
}
```

만약

사용자가 -5를 입력하였다면

-5 < 0이므로 해당 조건문 실행



정수를 입력하시오:-5
절대값은 5입니다.

복합문

- 복합문(compound statement)
 - 중괄호를 사용하여 문장들을 그룹핑하는 것,
 - 블록(block)이라고도 한다.
 - 단일문 대신 들어 갈 수 있다.

```
if( score >= 60 )  
{  
    printf("합격입니다.\n");  
    printf("장학금도 받을 수 있습니다.\n");  
}
```

조건식이 참이면 2개의 문
장이 묶여서 실행된다.

조건문의 간략한 표기

표준적인 방법	간략한 표기법
<pre>if(x != 0) printf("x가 0이 아닙니다.\n");</pre>	<pre>if(x) printf("x가 0이 아닙니다.\n");</pre>
<pre>if(x == 0) printf("x가 0입니다.\n");</pre>	<pre>if(!x) printf("x가 0입니다.\n");</pre>

오류 주의

경고: 오류 주의 #1

다음과 같이 if 문장의 조건식 뒤에 세미콜론을 찍으면 안 된다. if 문장은 조건식과 문장이 합쳐서 하나의 문장을 이룬다. 아래와 같이 작성하면 if 문은 `if(x > 0);`로 끝나고 `printf` 문장은 조건에 관계없이 실행된다.

```
if( x > 0 );  
    printf("양수입니다.\n");
```

경고: 오류 주의 #2

아주 많이 하는 오류가 두 값을 비교할 때 `==` 연산자를 사용하지 않고 `=` 연산자를 사용하는 것이다. 이 경우에는 비교가 되지 않고 값이 단순히 변수에 대입된다. 대입된 값에 따라서 참과 거짓이 결정된다.

```
if( x = 0 )  
    printf("x가 0이다.");
```

이 경우에는 `x`에 0이 대입되어서 항상 거짓이 된다. `x == 0`으로 작성하여야 한다. 이러한 오류를 방지하기 위하여 어떤 사람들은 `0 == x`와 같이 적는다. 만약 `0 = x`가 되면 문법 오류가 발생한다.

실수 비교

참고사항

실수와 실수를 비교할 때는 다음과 같은 문장을 사용하는 것은 문제가 될 수 있다.

```
if (result == expectedResult) { ... }
```

위의 비교는 참이 되기 힘들다. 왜냐하면 0.2와 같은 단순한 값은 정확하게 표현되지만 복잡한 값은 정확하게 표현되지 않기 때문이다. 따라서 부동소수점 수 2개가 같은지를 판별하려면 다음과 같이 오차를 감안하여서 비교하여야 한다. 즉 2개의 숫자가 오차 이내로 아주 근접하면 같은 것으로 판정하는 방법이다.

```
if (fabs(result - expectedResult) < 0.00001) { ... }
```

`fabs()` 함수는 실수의 절대값을 계산하여서 반환한다.

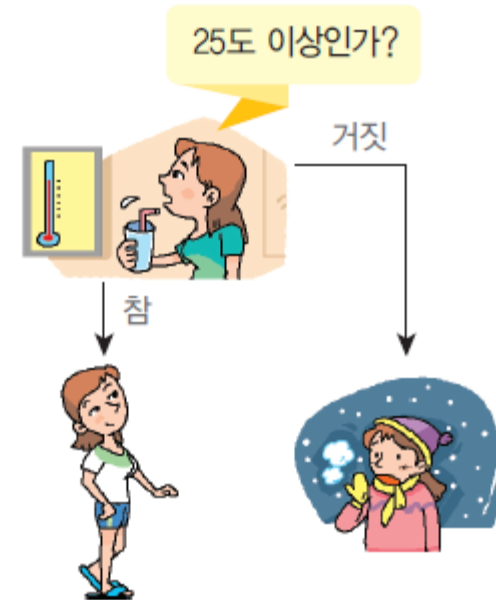
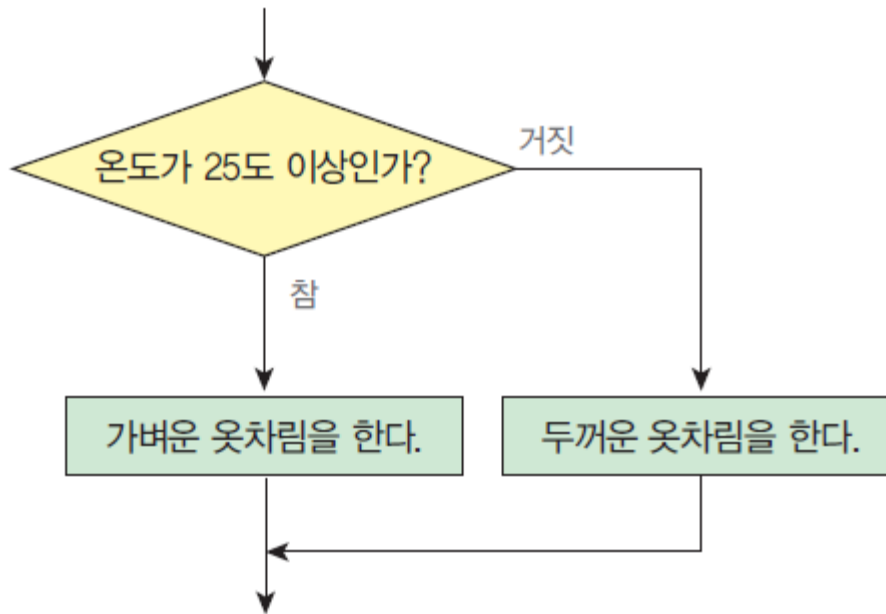
오차가 무시할 만 하면
같은 것으로 인정

중간 점검

1. 중괄호로 묶은 여러 개의 문장을 무엇이라고 하는가?
2. C에서 참과 거짓은 어떤 정수로 표시되는가?
3. if 문안의 조건식으로 많이 사용되는 수식의 종류는 무엇인가?
4. if 문이 끝나면 어떤 문장이 실행되는가?
5. 조건에 따라서 실행되어야 하는 문장이 두개 이상이면 어떻게 하여야 하는가?



if-else 문



Syntax

if-else 문

예

조건식
`if(number > 0)`

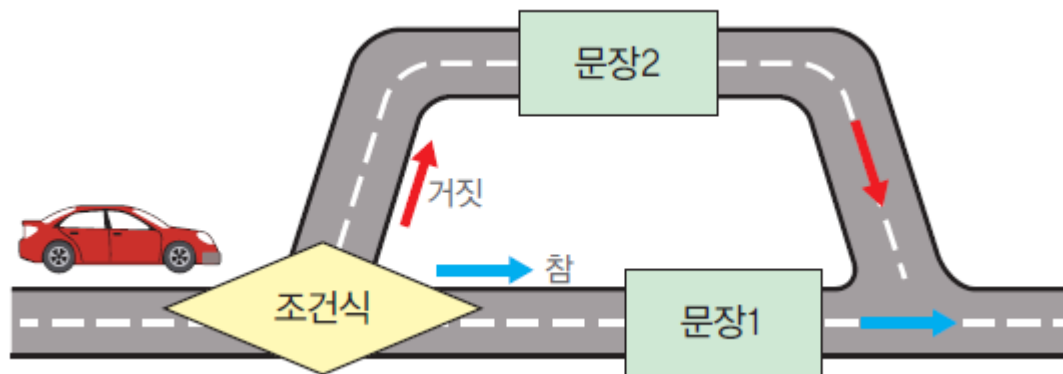
`printf("양수입니다.\n");`

`else`

`printf("양수가 아닙니다.\n");`

만약 조건식이 참이면
문장1이 실행된다.

그렇지 않으면 문장2가 실행
된다.



```
if ( score >= 60 )
```

```
    printf("합격입니다.\n");
```

```
else
```

```
    printf("불합격입니다.\n");
```

score가 60이상이면 실행

score가 60미만이면 실행

```
if ( score >= 60 )
```

```
{
```

```
    printf("합격입니다.\n");
```

```
    printf("장학금도 받을 수 있습니다.\n");
```

```
}
```

```
else
```

```
{
```

```
    printf("불합격입니다.\n");
```

```
    printf( " 다시 도전하세요.\n");
```

```
}
```

score가 60이상이면 실행

score가 60미만이면 실행

복잡한 조건식도 가능

- 학점 결정 코드

```
if( score >= 80 && score < 90 )  
    grade = 'B';
```

- 공백 문자들의 개수를 세는 코드

```
if( ch == ' ' || ch == '\n' || ch == '\t' )  
    white_space++;
```

조건 연산자

- 간단한 **if-else** 문은 4장에서 학습하였던 조건 연산자를 사용하여 표현할 수도 있다.

```
(score >= 60 ) ? printf("합격입니다.\n") : printf("불합격입니다.\n");
```

```
bonus = (( years > 30 ) ? 500 : 300 );
```

if-else 문의 스타일

스타일

if-else 문은 보통 다음의 2가지 중의 하나의 스타일을 이용하는 것이 좋다. 이 책에서는 주로 첫 번째 방법을 사용하지만 지면이 부족할 때는 두 번째 방법도 사용하였다.

복합문은 들여쓰기를 하는 편이 읽기가 쉬워진다.

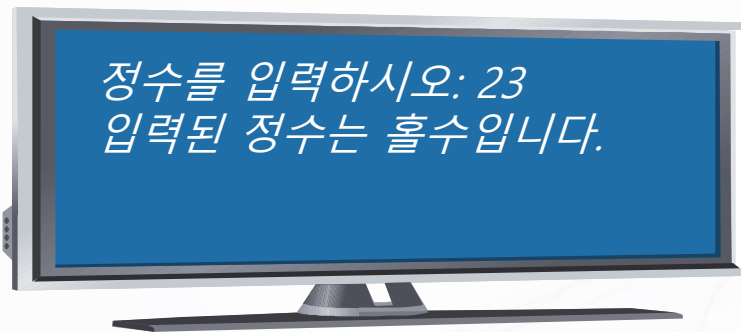
```
if( expression )
{
    → statement11;
    statement12;
    ...
}
else
{
    → statement21;
    statement22;
    ...
}
```

공간의 절약을 위하여 이런 형태로 작성하기도 한다.

```
if( expression ){
    statement11;
    statement12;
    ...
}
else {
    statement21;
    statement22;
    ...
}
```

예제 #1

- 키보드에서 입력받은 정수가 홀수인지 짝수인지를 말해주는 프로그램을 작성하여 보자. 홀수와 짝수는 어떻게 구별할 수 있는가?



예제 #1

// if-else 문을 이용하여 홀수와 짝수를 구분한다.

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int number;
```

```
    printf("정수를 입력하시오:");
```

```
    scanf("%d", &number);
```

```
    if( number % 2 == 0 )
```

```
        printf("입력된 정수는 짝수입니다.\n");
```

```
    else
```

```
        printf("입력된 정수는 홀수입니다.\n");
```

```
    return 0;
```

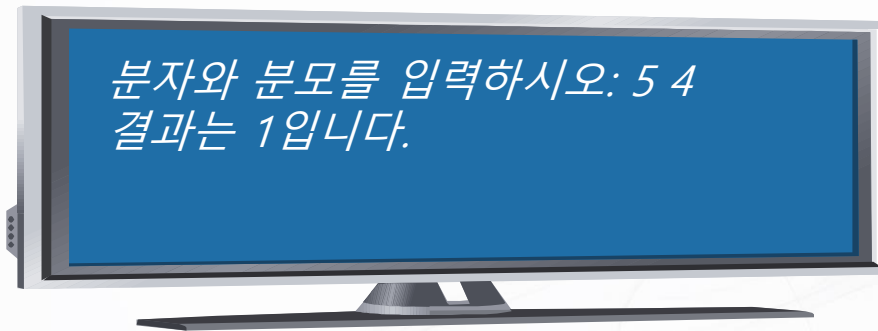
```
}
```

2로 나누어서 나머지가 0이면 짝수이다.

정수를 입력하시오: 23
입력된 정수는 홀수입니다.

예제 #2

- 사용자로부터 두 개의 정수를 입력받아서 정수 간의 나눗셈을 실행한다. 나눗셈을 하기 전에 분모가 0인지를 if 문을 이용하여 검사한다.

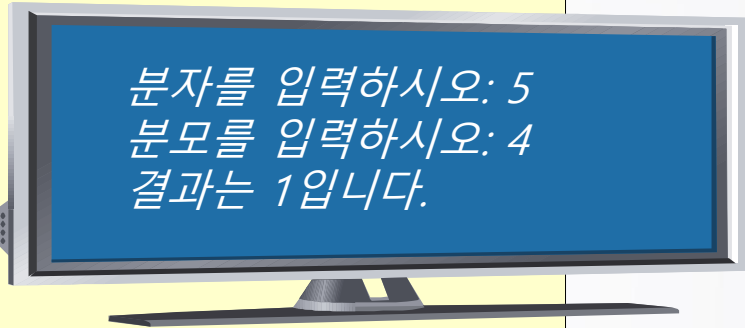


예제 #2

```
// 나눗셈을 하기 전에 분모가 0인지를 if-else 문을 이용하여 검사
#include <stdio.h>
```

```
int main(void)
{
    int n, d, result;

    printf("분자와 분모를 입력하시오: ");
    scanf("%d %d", &n, &d);
    if( d == 0 )
    {
        printf("0으로 나눌 수는 없습니다.\n");
    }
    else
    {
        result = n / d;
        printf("결과는 %d입니다.\n", result);
    }
    return 0;
}
```



분자를 입력하시오: 5
분모를 입력하시오: 4
결과는 1입니다.

예제 #3

- 앞에서 등장하였던 윤년인지 아닌지를 판단하는 프로그램을 if 문을 사용하여 다시 작성하여 보자.
 - 연도가 4 로 나누어 떨어지면서 100으로 나누어 떨어지지 않은 연도
 - 400으로 나누어 떨어지는 연도



예제 #3

```
// 윤년 판단 프로그램
```

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int year;
```

```
    printf("연도를 입력하시오: ");
```

```
    scanf("%d", &year);
```

```
    if((year % 4 == 0 && year % 100 != 0) || year % 400 == 0)
```

```
        printf("%d년은 윤년입니다.\n", year);
```

```
    else
```

```
        printf("%d년은 윤년이 아닙니다.\n", year);
```

```
    return 0;
```

```
}
```



연도를 입력하시오: 2012
2012년은 윤년입니다.

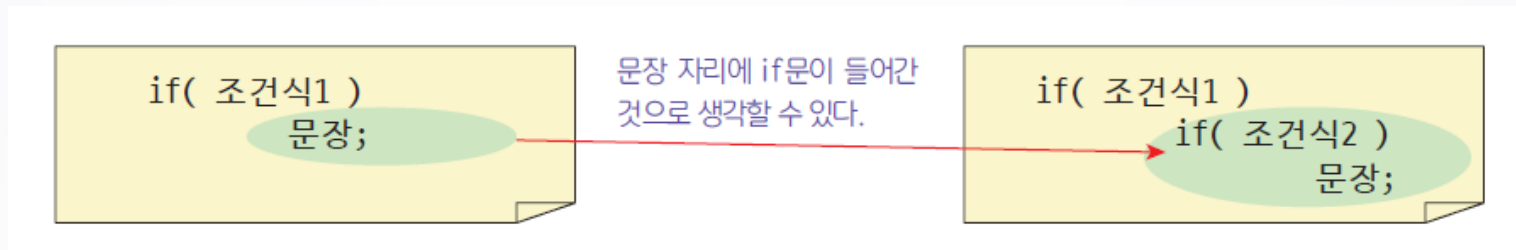
중간 점검

1. 변수 n 의 값이 100보다 크거나 같으면 "large", 100보다 작으면 "small"을 출력하는 if-else 문을 작성하라.



중첩 if

- if 문에 다시 if 문이 포함



중첩 if

```
if( score >= 80 )  
    if( score >= 90 )  
        printf("당신의 학점은 A입니다.\n");
```

if 문안의 문장 자리에
if문이 들어간 경우

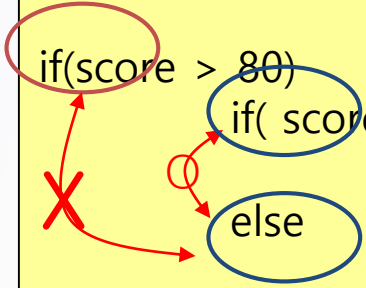
```
if( score >= 80 )  
    if( score >= 90 )  
        printf("당신의 학점은 A입니다.\n");  
    else  
        printf("당신의 학점은 B입니다.\n");
```

if 문안의 문장 자리에
if-else 문이 들어간 경우


if와 else의 매칭 문제

else 절은 가장 가까운 if절
과 매치된다.

```
if(score > 80)
    if( score >= 90)
        printf("당신의 학점은 A입니다\n");
    else
        printf("당신의 학점은 B입니다\n");
```



```
if( score >= 80 )
{
    if( score >= 90 )
        printf("당신의 학점은 A입니다.\n");
}
else
    printf("당신의 학점은 A나 B가 아닙니다.\n");
```



만약 다른 if절과 else 절을 매치시키려면
중괄호를 사용하여 블록으로 묶는다.

연속적인 if

Syntax

연속적인 if 문

문법

```
if( 조건식1 )
```

```
    문장1;
```

```
else if( 조건식2 )
```

```
    문장2;
```

```
else if( 조건식3 )
```

```
    문장3;
```

```
else
```

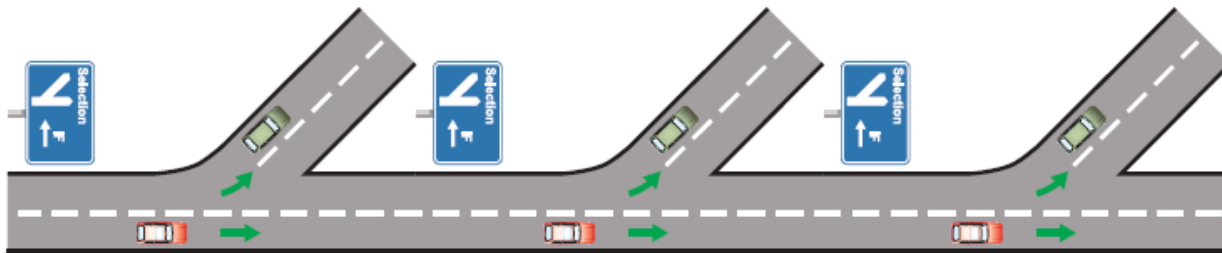
```
    문장4;
```

만약 조건식1이 참이면 문장1이 실행된다.

그렇지 않고 조건식2가 참이면 문장2가 실행된다.

그렇지 않고 조건식3이 참이면 문장3이 실행된다.

그렇지 않으면 문장4가 실행된다.



학점 결정 예제

- 학생들의 성적을 받아서 학점을 출력하는 프로그램을 작성하여 실행하여보자.



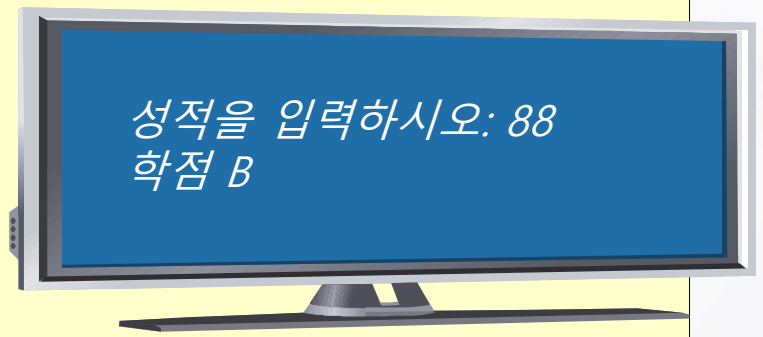
학점 결정 예제

```
#include <stdio.h>

int main(void)
{
    int score;

    printf("성적을 입력하시오: ");
    scanf("%d", &score);

    if (score >= 90)
        printf("합격: 학점A\n");
    else if (score >= 80)
        printf("합격: 학점B\n");
    else if (score >= 70)
        printf("합격: 학점C\n");
    else if (score >= 60)
        printf("합격: 학점D\n");
    else
        printf("불합격: 학점F\n");
    return 0;
}
```



문자 분류 예제

- 키보드에서 문자를 받아서 문자들을 대문자(A-Z), 소문자(a-z), 숫자(0-9), 그 외의 문자들로 구분하여 보자.
- 문자를 받아들이는 함수로는 `getchar()`를 사용하자

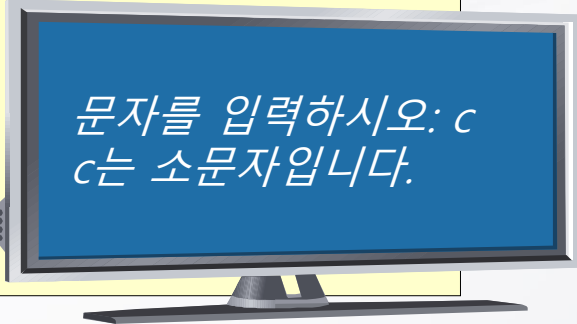


문자 분류 예제

```
// 문자들을 분류하는 프로그램
#include <stdio.h>
int main(void)
{
    char ch;
    printf("문자를 입력하시오: ");

    ch = getchar();
    if( ch >= 'A' && ch <= 'Z' )
        printf("%c는 대문자입니다.\n", ch);
    else if( ch >= 'a' && ch <= 'z' )
        printf("%c는 소문자입니다.\n", ch);
    else if( ch >= '0' && ch <= '9' )
        printf("%c는 숫자입니다.\n", ch);
    else
        printf("%c는 기타문자입니다.\n", ch);

    return 0;
}
```



문자를 입력하시오: c
c는 소문자입니다.

중간 점검

1. n 의 값이 각각 -1, 0, 5인 경우에 다음의 코드에 의하여 생성되는 출력은 무엇인가?

```
if( n == 0 )  
    printf("A");  
else if( n > 3 )  
    printf("B");  
else  
    printf("C");
```

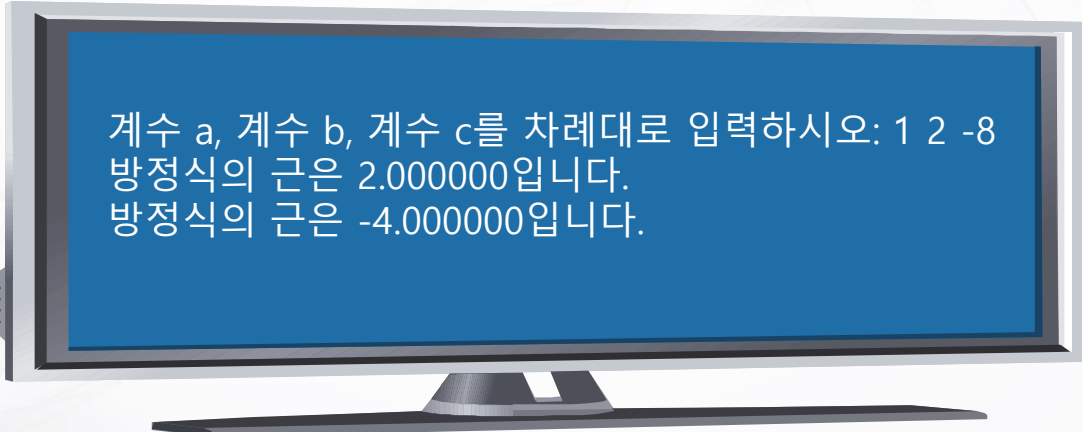
2. 컵의 사이즈를 받아서 100ml미만은 small, 100ml이상 200ml미만은 medium, 200ml 이상은 large라고 출력하는 연속적인 if-else 문을 작성하십시오.



Lab: 이차 방정식

1. 사용자에게 이차 방정식의 계수 a , b , c 를 입력하도록 한다.
2. 만약 a 가 0이면 근은 $-c/b$ 이다.
3. 판별식 ($b^2 - 4ac$)가 음수이면 실근은 존재하지 않는다.
4. 위의 조건에 해당되지 않으면 다음과 같은 공식을 이용하여 실근을 구한다.

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



계수 a , 계수 b , 계수 c 를 차례대로 입력하시오: 1 2 -8
방정식의 근은 2.000000입니다.
방정식의 근은 -4.000000입니다.

알고리즘

사용자로부터 a , b , c 를 읽는다.

if $a == 0$

일차 방정식의 근을 구한다.

실근을 출력한다.

else

판별식을 계산한다.

if 판별식 ≥ 0

근의 공식을 이용하여 실근을 구한다.

실근을 출력한다.

else

실근은 없다는 메시지 출력

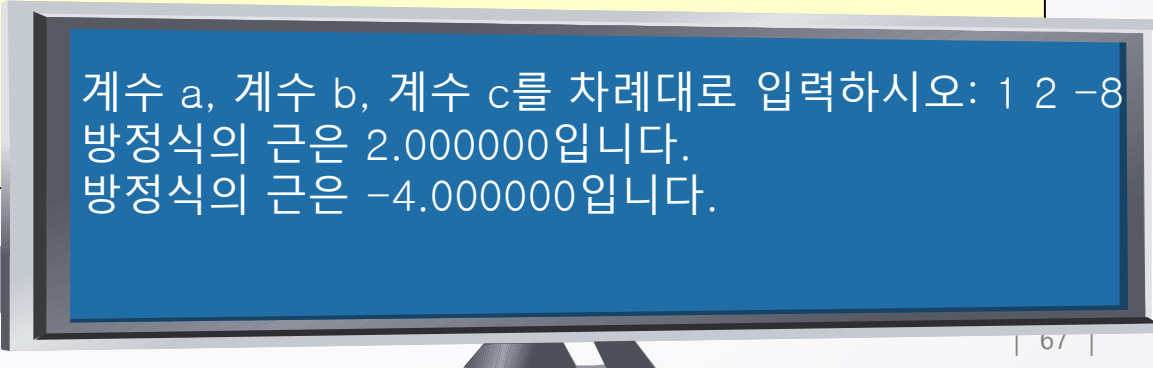
소스

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>
int main(void)
{
    double a, b, c, dis;

    printf("계수 a, 계수 b, 계수 c를 차례대로 입력하시오: ");
    scanf("%lf %lf %lf", &a, &b, &c);
```

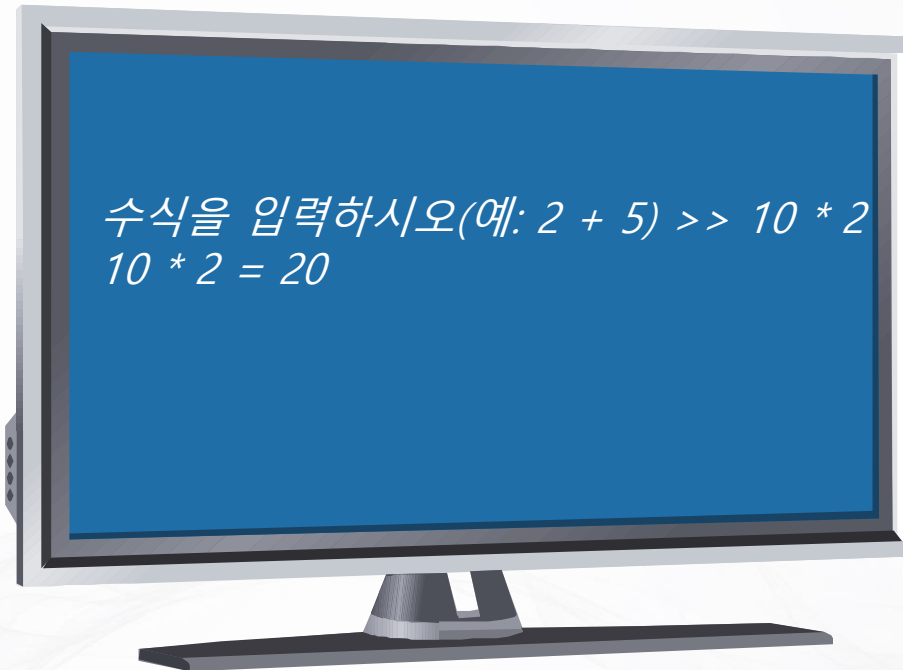
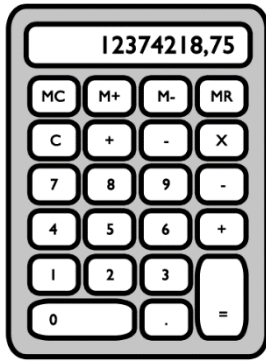
소스

```
if (a == 0)
    printf("방정식의 근은 %f입니다.", -c / b);
else
{
    dis = b * b - 4.0 * a * c;
    if (dis >= 0)
    {
        printf("방정식의 근은 %f입니다.\n", (-b + sqrt(dis)) / (2.0 * a));
        printf("방정식의 근은 %f입니다.\n", (-b - sqrt(dis)) / (2.0 * a));
    }
    else
        printf("실근이 존재하지 않습니다\n");
}
return 0;
}
```



계수 a, 계수 b, 계수 c를 차례대로 입력하시오: 1 2 -8
방정식의 근은 2.000000입니다.
방정식의 근은 -4.000000입니다.

Lab: 산술 계산기



Solution

```
#include <stdio.h>

int main(void)
{
    char op;
    int x, y, result;

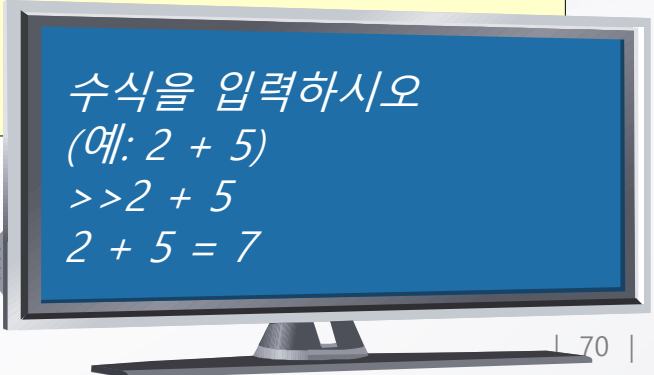
    printf("수식을 입력하시오(예: 2 + 5) >> ");
    scanf("%d %c %d", &x, &op, &y);
```

Solution

```
if( op == '+' )
    result = x + y;
else if( op == '-' )
    result = x - y;
else if( op == '*' )
    result = x * y;
else if( op == '/' )
    result = x / y;
else if( op == '%' )
    result = x % y;
else
    printf("지원되지 않는 연산자입니다. ");

printf("%d %c %d = %d \n", x, op, y, result);
return 0;
```

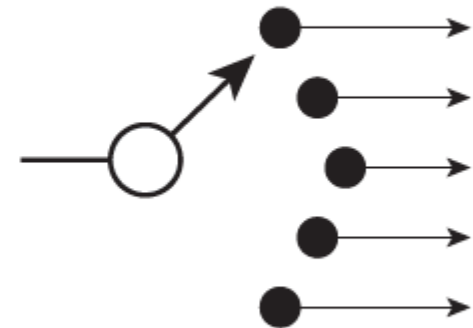
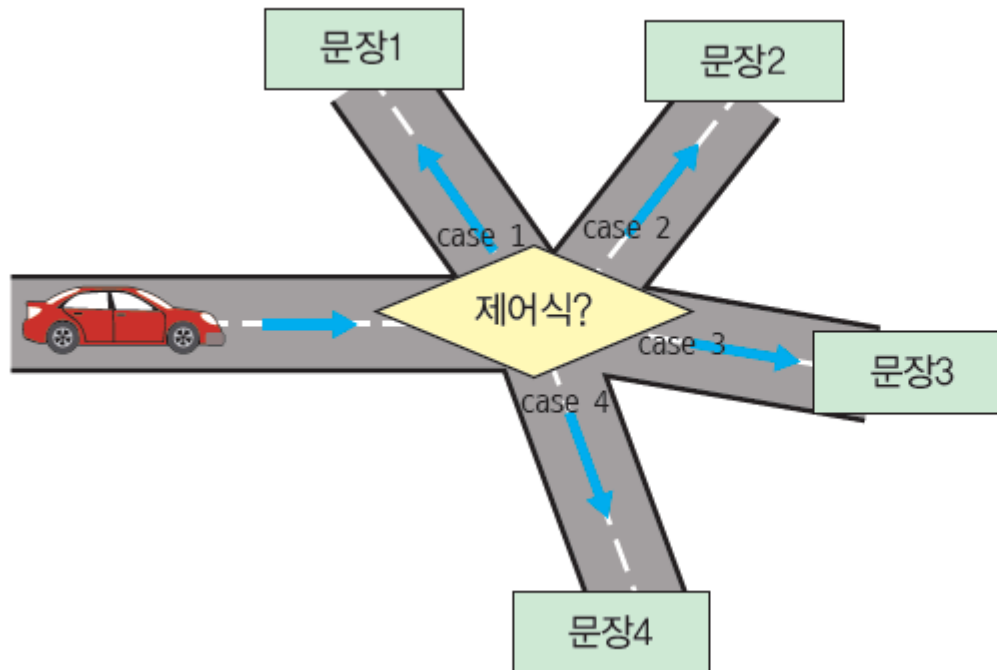
```
}
```



수식을 입력하십시오
(예: 2 + 5)
>> 2 + 5
2 + 5 = 7

switch 문

- 제어식의 값에 따라서 여러 경로 중에서 하나를 선택할 수 있는 제어 구조



switch 문

Syntax

switch 문

문법

```
switch(제어식)
```

```
{
```

```
case c1:
```

```
문장1;
```

```
break;
```

제어식의 값이 c1이면 실행된다.

```
case c2:
```

```
문장2;
```

```
break;
```

제어식의 값이 c2이면 실행된다.

```
...
```

```
default:
```

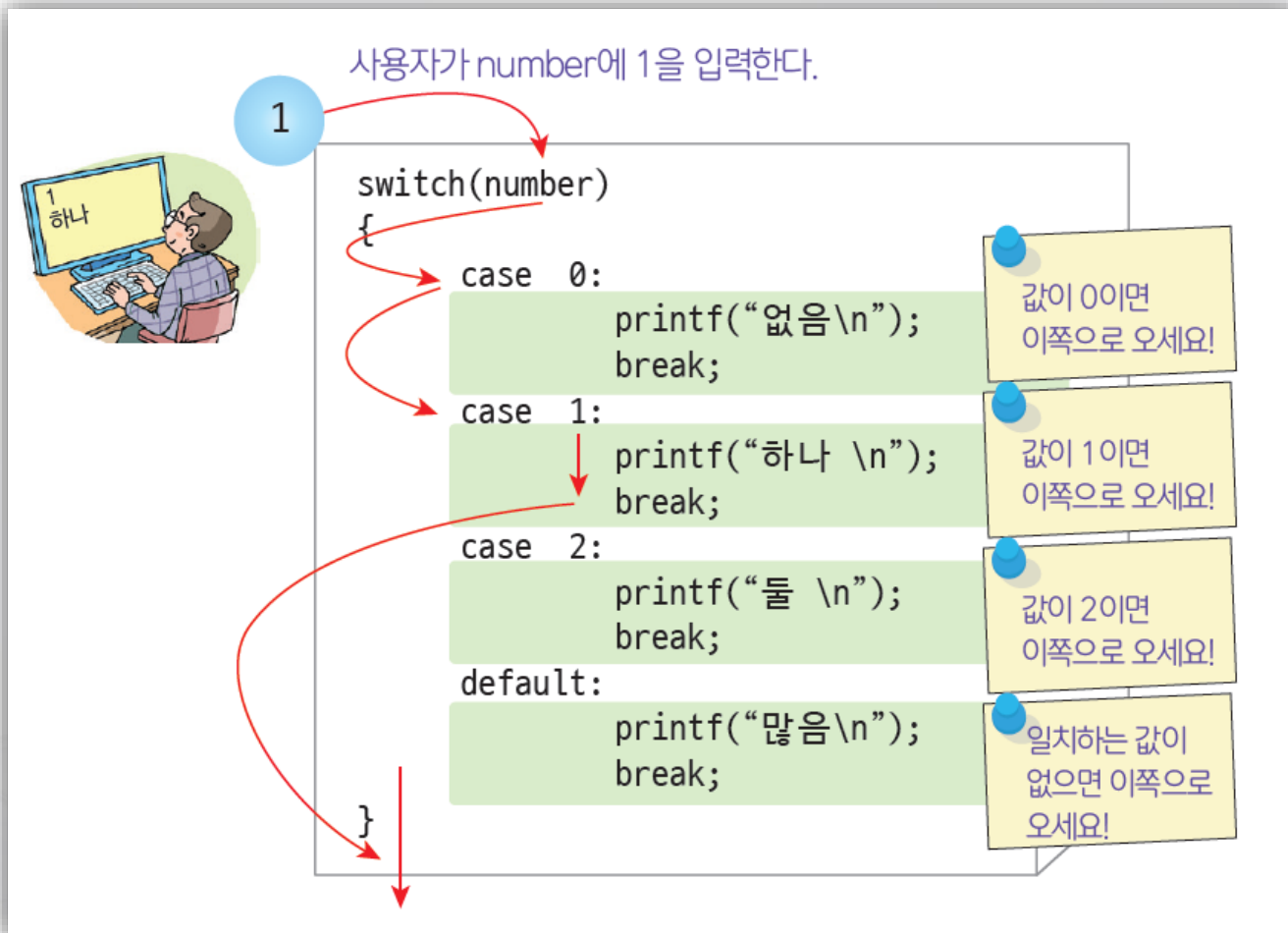
```
문장d;
```

```
break;
```

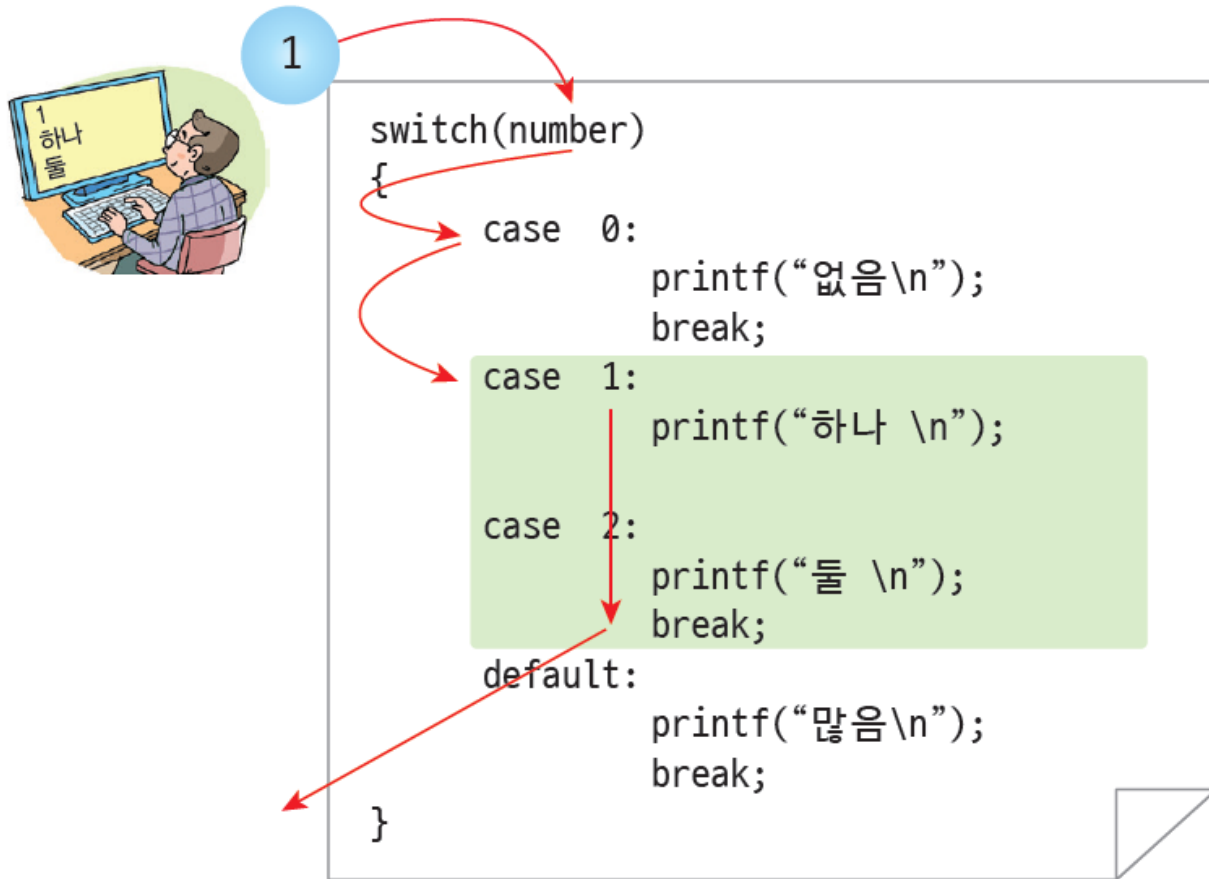
일치하는 값이 없으면 실행된다.

```
}
```


사용자가 1을 입력하는 경우



break가 생략되는 경우



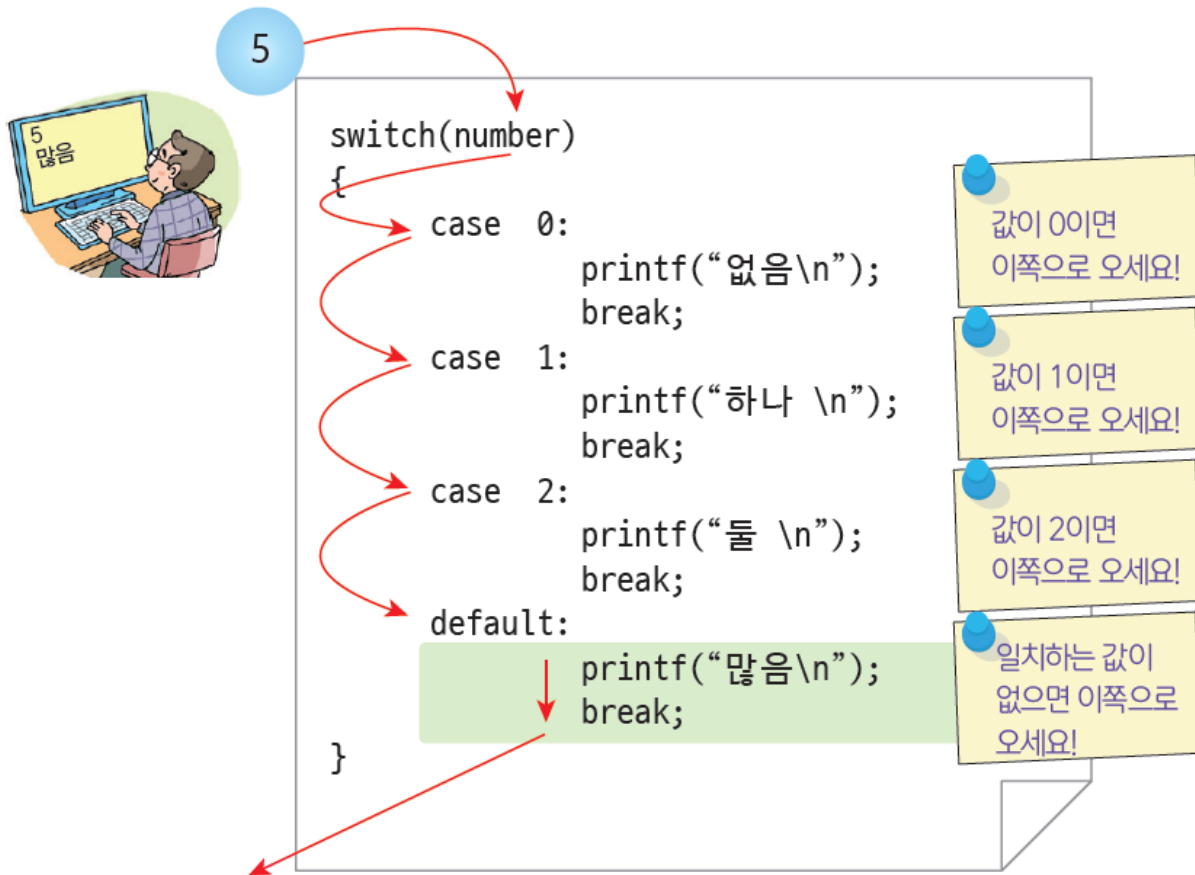
의도적인 break생략



2

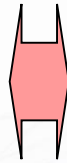
```
switch(number)
{
    case 0:
        printf("없음\n");
        break;
    case 1:
        printf("하나 \n");
        break;
    case 2:
    case 3:
        printf("두서너 개 \n");
        break;
    default:
        printf("많음\n");
        break;
}
```

default 문



switch 문과 if-else 문

```
switch(number) {  
    case 0:  
        printf("없음\n");  
        break;  
    case 1:  
        printf("하나\n");  
        break;  
    case 2:  
        printf("둘\n");  
        break;  
    default:  
        printf("많음\n");  
        break;  
}
```



```
if( number == 0 )  
    printf("없음\n");  
else if( number == 1 )  
    printf("하나\n");  
else if( number == 2 )  
    printf("둘\n");  
else  
    printf("많음\n");
```

switch 문에서 주의할 점

```
switch(number)
{
    case x:                                // 변수는 사용할 수 없다.
        printf("x와 일치합니다. \n ");
        break;
    case (x+2):                            // 변수가 들어간 수식은 사용할 수 없다.
        printf("수식과 일치합니다. \n ");
        break;
    case 0.001:                            // 실수는 사용할 수 없다.
        printf("실수 \n ");
        break;
    case 'a':                              // OK! 문자는 사용할 수 있다.
        printf("문자 \n ");
        break;
    case "001":                            // 문자열은 사용할 수 없다.
        printf("문자열 \n ");
        break;
}
```

정수의 범위를 나타낼 때

```
switch (score) {  
    case 100:  
    case 99:  
    case 98:  
    ...  
    case 90:  
        printf("A학점입니다.\n");  
        break;  
    ...  
}
```



```
if( score >= 90 && score <= 100 )  
    printf("A학점입니다.\n");
```

정수의 범위도 표현할 수 있으나 번거롭다.

정수의 범위를 나타낼 때

```
int iscore;
...
iscore = score/10;           // 정수 나눗셈의 경우, 나머지는 없어진다.
switch (iscore) {
    case 9: grade = 'A'; break; // 90-100은 A 학점
    case 8: grade = 'B'; break; // 80-89은 B 학점
    case 7: grade = 'C'; break; // 70-79은 C 학점
    case 6: grade = 'D'; break; // 60-69은 D 학점
    default: grade = 'F'; break; // 59점 이하는 F 학점
}
```

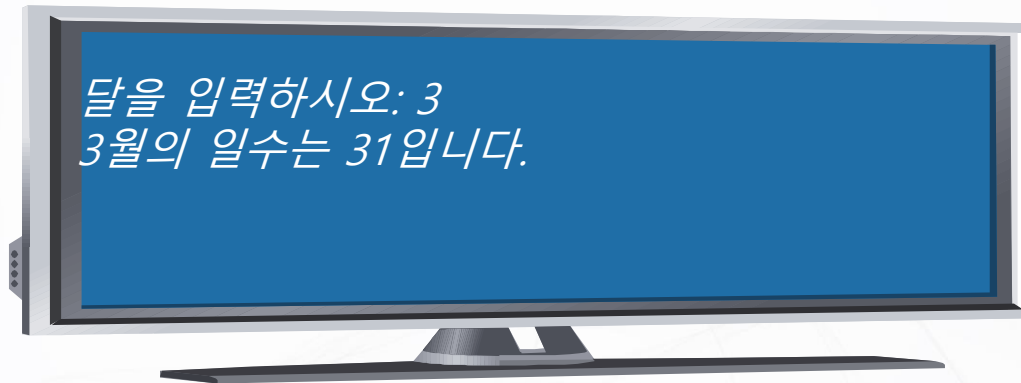


switch 문과 if/else 체인 중에서 어떤 것이 더 효율적인가?

차이는 미소하다. 하지만 switch 문은 간략한 점프 테이블로 효율적으로 구현이 가능하도록 설계되었다. 따라서 대부분의 경우 switch를 사용하는 것이 좋다. 코드가 간결하고 아마 약간은 효율적이다.

예제

- 각 달의 일수를 출력하는 프로그램을 작성해보자. 즉 달이 주어지면 그 달의 일수를 출력한다.



예제 #1

// 달의 일수를 계산하는 프로그램

`#include <stdio.h>`

`int main(void)`

`{`

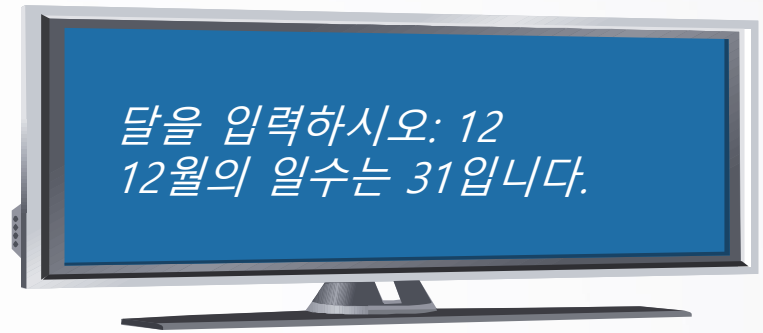
`int month, days;`

`printf("달을 입력하시오: ");`

`scanf("%d", &month);`

예제

```
switch(month)
{
    case 2:
        days = 28;
        break;
    case 4:
    case 6:
    case 9:
    case 11:
        days = 30;
        break;
    default:
        days = 31;
        break;
}
printf("%d월의 일수는 %d입니다.\n", month, days);
return 0;
}
```



중간 점검

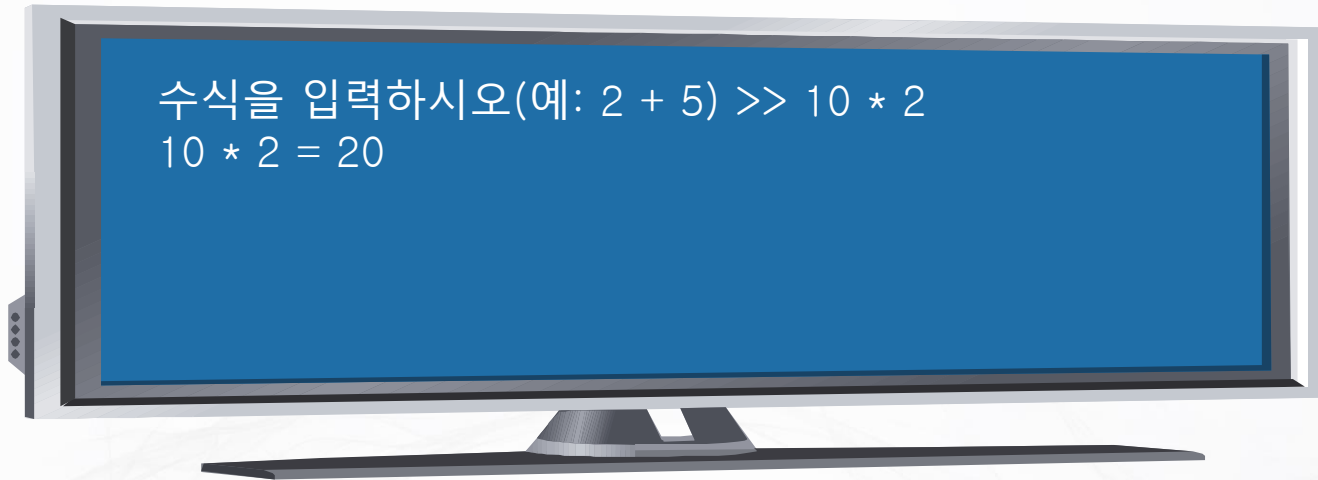
1. case 절에서 break 문을 생략하면 어떻게 되는가?
2. 변수 fruit의 값이 각각 1, 2, 5일 때, 다음의 코드의 출력을 쓰시오.

```
switch(fruit) {  
    case 1:  
        printf("사과");  
        break;  
  
    case 2:  
        printf("배");  
  
    case 3:  
        printf("바나나");  
        break;  
  
    default:  
        printf("과일");  
        break;  
}
```



Lab: 산술 계산기(switch 버전)

- 앞의 산술 계산기 예제를 **switch** 문을 이용하여 다시 작성하여 보자.



Lab: 산술 계산기

```
// 간단한 산술 계산기 프로그램
```

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    char op;
```

```
    int x, y, result;
```

```
    printf("수식을 입력하십시오(예: 2 + 5) >> ");
```

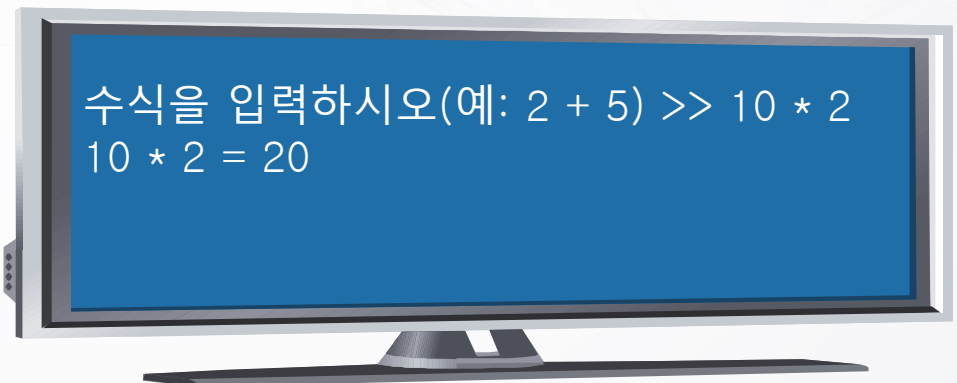
```
    scanf("%d %c %d", &x, &op, &y);
```

Lab: 산술 계산기

```
switch (op)
{
    case '+':
        result = x + y;
        break;
    case '-':
        result = x - y;
        break;
    case '*':
        result = x * y;
        break;
    case '/':
        result = x / y;
        break;
```

Lab: 산술 계산기

```
case '%':  
    result = x % y;  
    break;  
default:  
    printf("지원되지 않는 연산자입니다. \n");  
    break;  
}  
printf("%d %c %d = %d \n", x, op, y, result);  
return 0;  
}
```

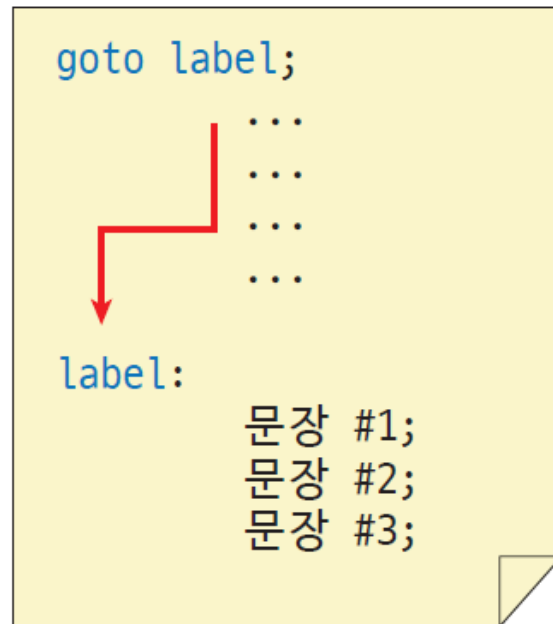


goto문

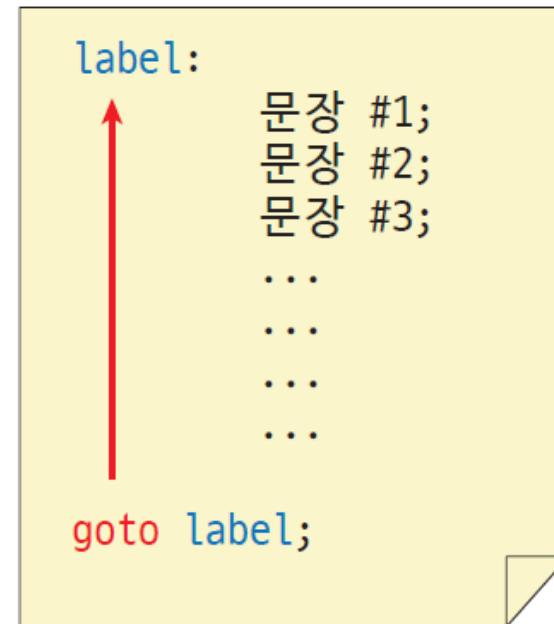
- 조건없이 어떤 위치로 점프
- 사용하지 않는 것이 좋음



goto 문



전향 참조



후향 참조

예제

// 구구단 출력 프로그램

#include <stdio.h>

int main(void)

{

int i = 1;

loop:

printf("%d * %d = %d \n", 3, i, 3 * i);

i++;

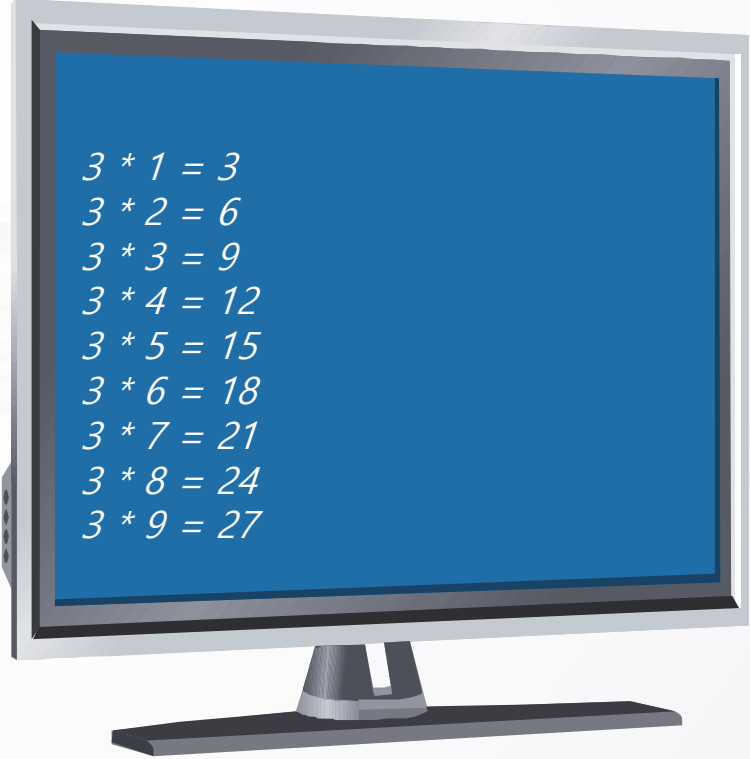
if(i == 10) goto end;

goto loop;

end:

return 0;

}



3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27

Mini Project: 소득세 계산기 만들기



소득세 계산표

- 산출세액 = 과세표준 * 세율 - 누진공제액 = 35000000 * 15% - 1080000 = 4170000

과세표준	세율	누진공제
0	6%	0
12,000,000	15%	1,080,000
46,000,000	24%	5,220,000
88,000,000	35%	14,900,000
150,000,000	38%	19,400,000
300,000,000	40%	25,400,000
500,000,000	42%	35,400,000
1,000,000,000	45%	65,400,000

Q & A

