


제 6장 조건문

이번 장에서 학습할 내용

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

필요에 따라서
조건이 만족되면
문장의 실행 순서를
변경할 수 있는
기능이 제공됩니다.

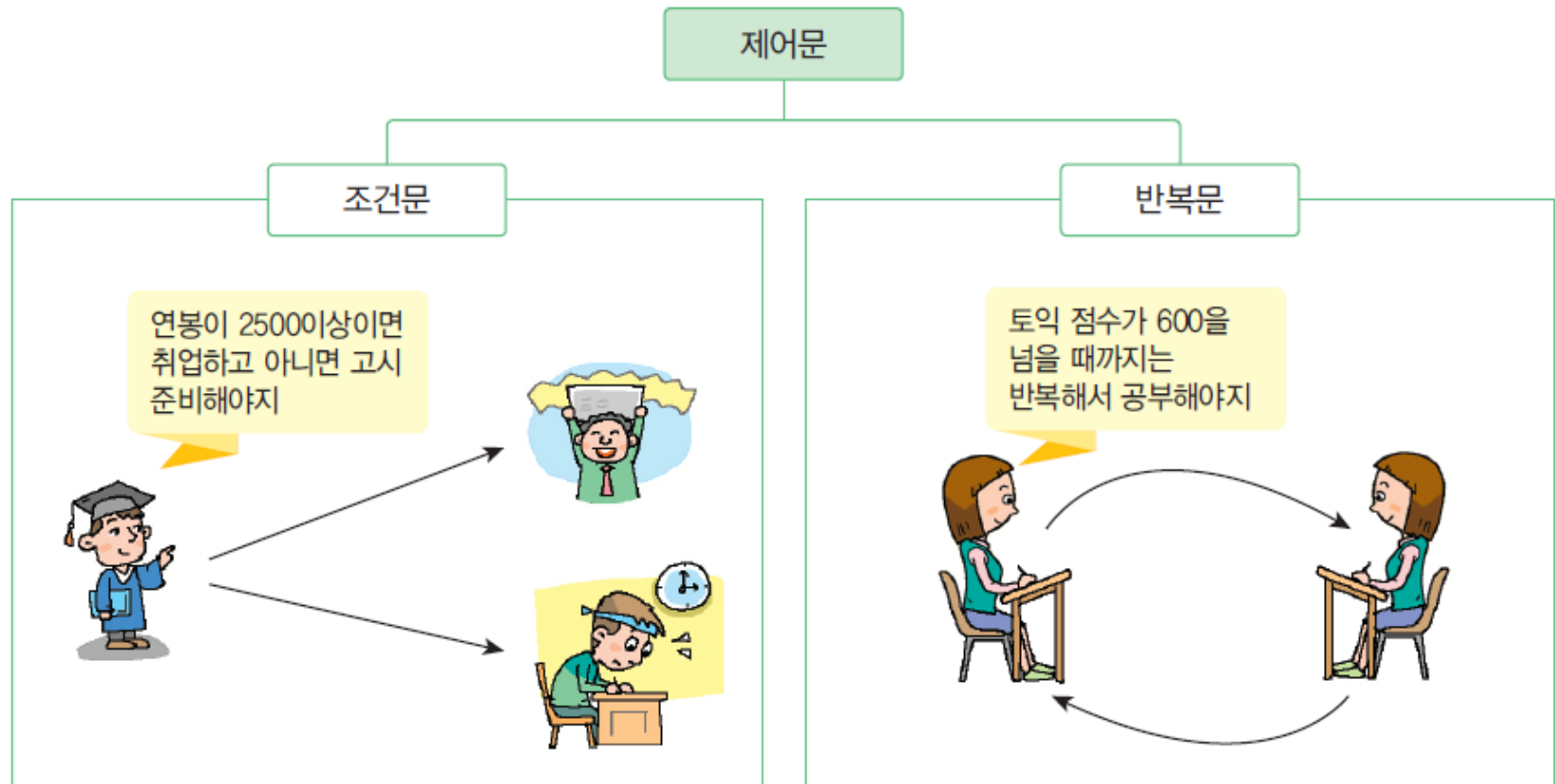


조건문

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

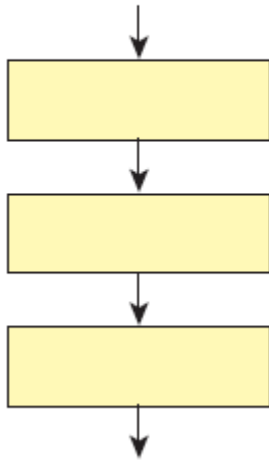


제어문

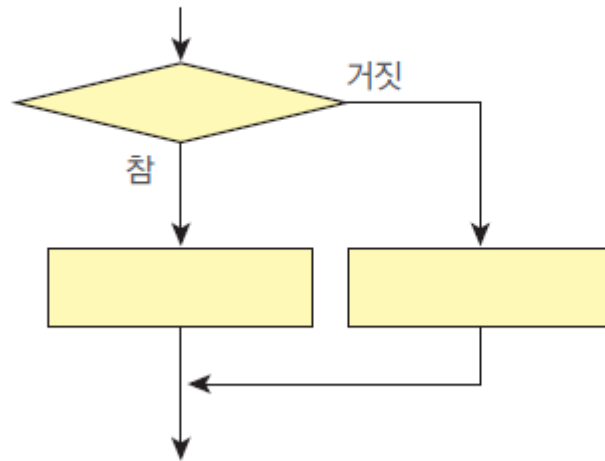


3가지의 제어구조

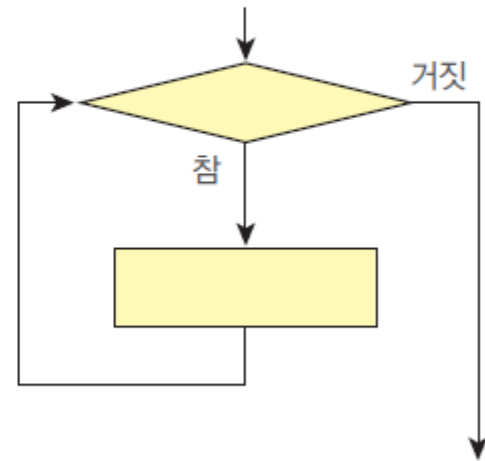
순차구조



선택구조

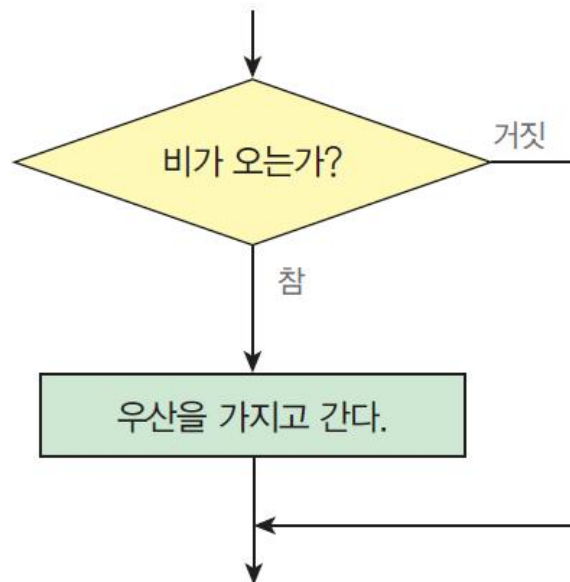


반복구조



if문

- 일상생활에서도 조건에 따라서 결정을 내려야 하는 경우는 많이 있다.



if문의 구조

Syntax

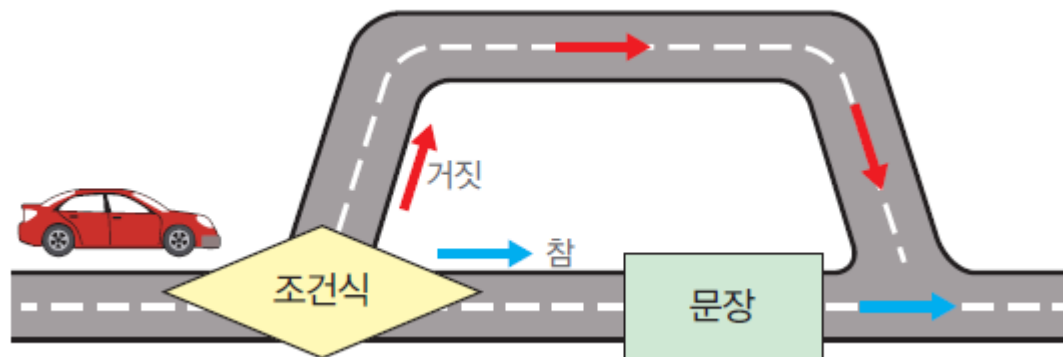
if 문

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.

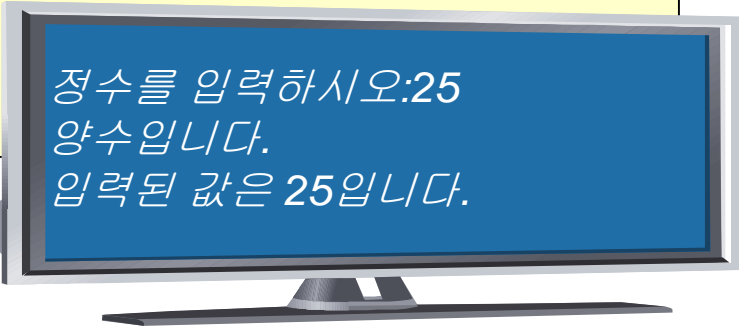
예제

```
#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);
```

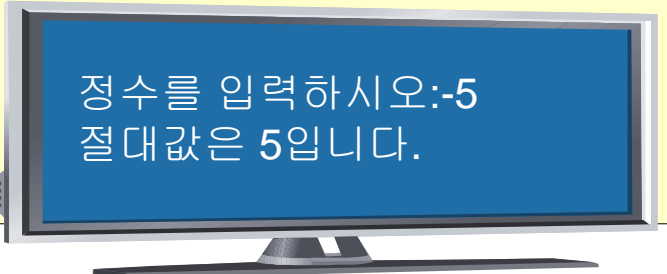
```
    return 0;
```

```
}
```

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.

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

조건식이 참이면 2개의
문장이 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 x에 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) { ... }
```

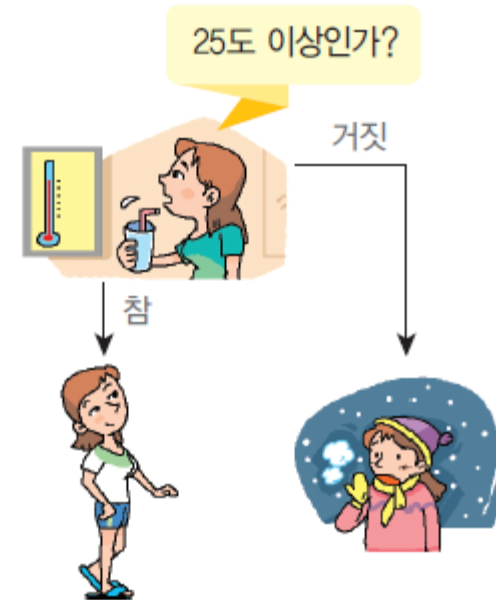
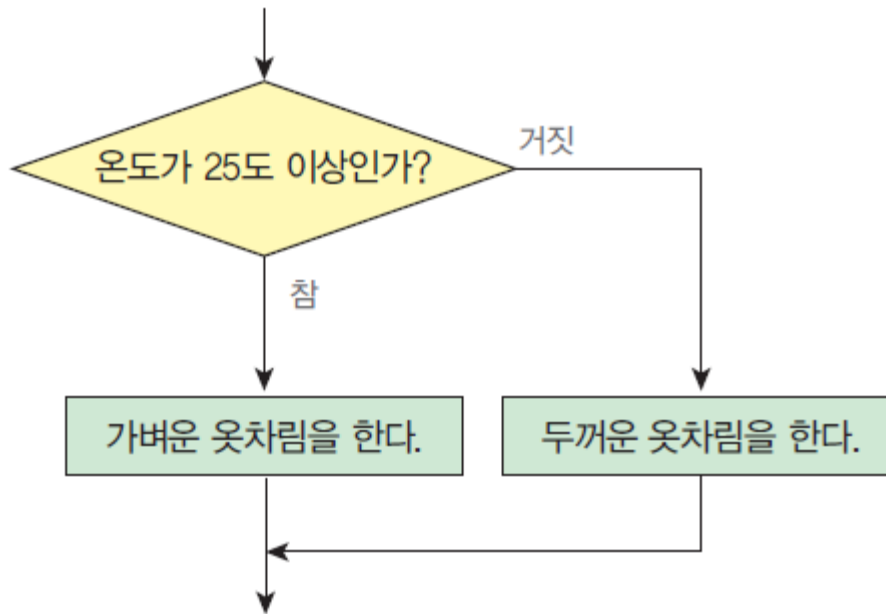
위의 비교는 참이 되기 힘들다. 왜냐하면 0.2와 같은 단순한 값은 정확하게 표현되지만 복잡한 값은 정확하게 표현되지 않기 때문이다. 따라서 부동소수점 수 2개가 같은 지를 판별하려면 다음과 같이 오차를 감안하여서 비교하여야 한다. 즉 2개의 숫자가 오차 이내로 아주 근접하면 같은 것으로 판정하는 방법이다.

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

fabs() 함수는 실수의 절대값을 계산하여서 반환한다.

오차가 무시할 만 하면
같은 것으로 인정

if-else 문



if-else 문

Syntax

if-else 문

예

조건식
`if(number > 0)`

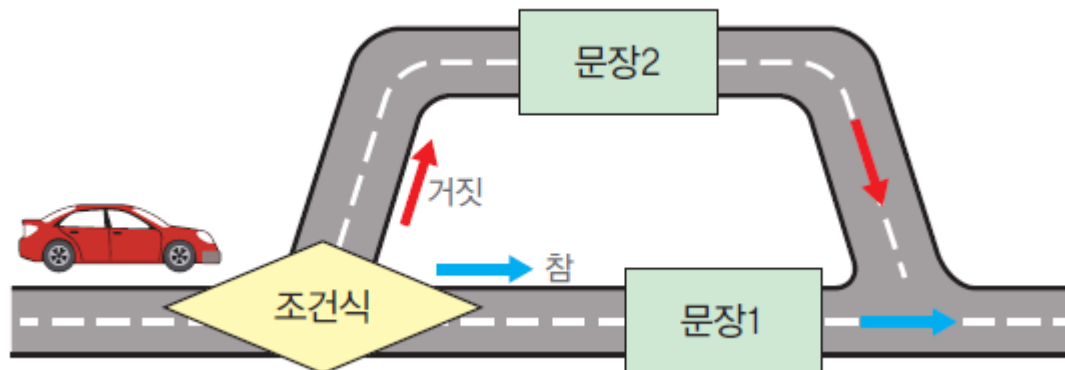
`printf("양수입니다.\n");`

`else`

`printf("양수가 아닙니다.\n");`

만약 조건식이 참이면
문장1이 실행된다.

그렇지 않으면 문장2가 실행
된다.



if-else 문

```
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++;
```

조건연산자

- 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.

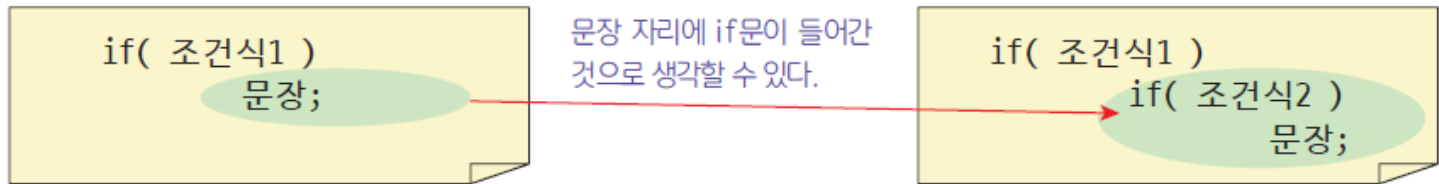
```
if( expression )
{
    → statement11;
    statement12;
    ...
}
else
{
    → statement21;
    statement22;
    ...
}
```

공간의 절약을 위하여 이런 형태로 작성하기도 한다.

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

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
```

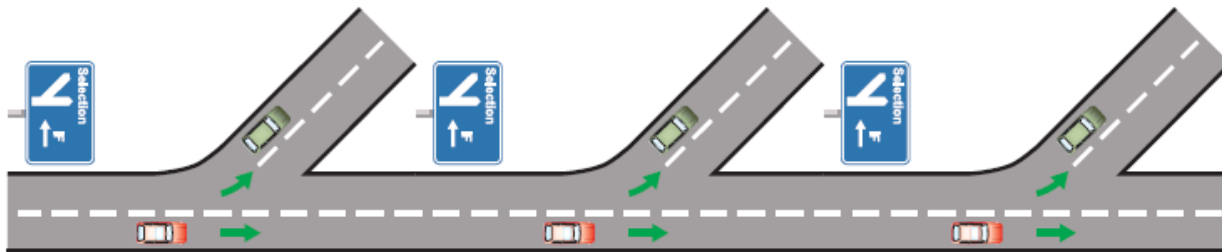
```
    문장4;
```

만약 조건식1이 참이면 문장1이 실행된다.

그렇지 않고 조건식2가 참이면 문장2가 실행된다.

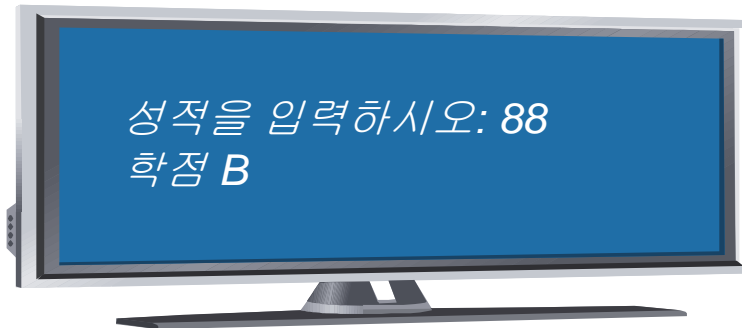
그렇지 않고 조건식3이 참이면 문장3이 실행된다.

그렇지 않으면 문장4가 실행된다.



학점 결정 예제

- 학생들의 성적을 받아서 학점을 출력하는 프로그램을 작성하여 실행해보자.



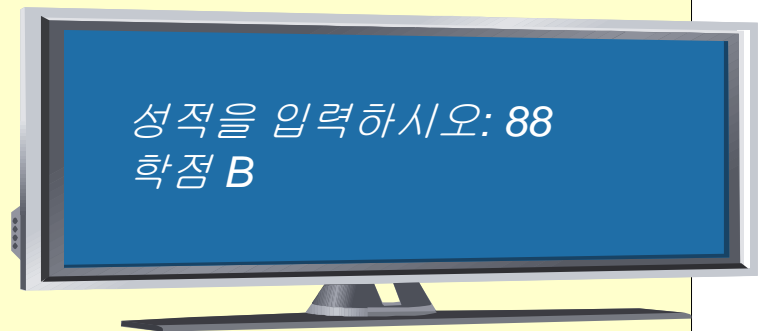
학점 결정 예제

```
#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

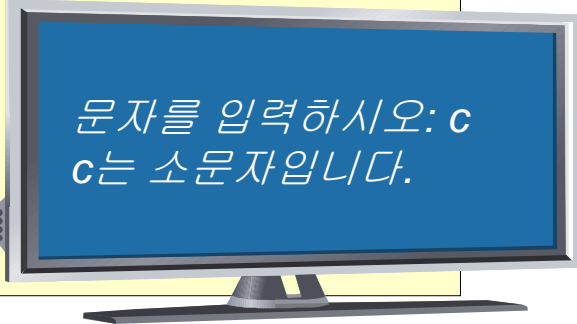


문자 분류 예제

```
// 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.

알고리즘

사용자로부터 a, b, c를 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

소 스

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

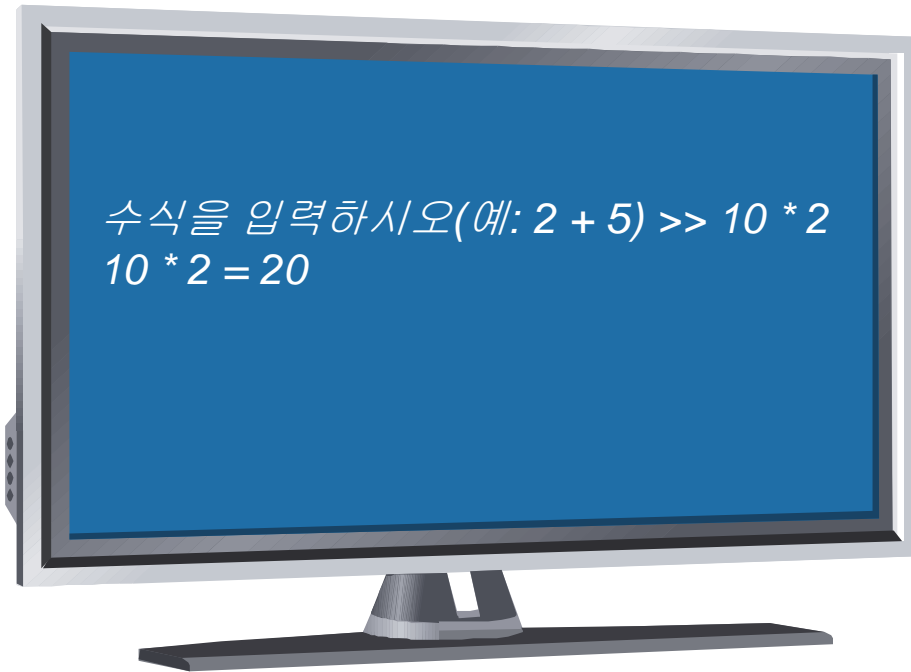
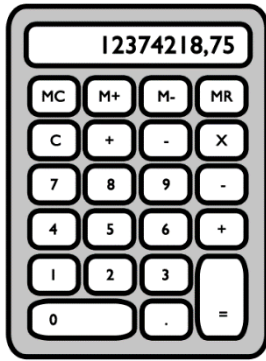
    printf("계수 a, 계수 b, 계수 c를 차례대로 입력하시오: ");
    scanf("%lf %lf %lf", &a, &b, &c);
```

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;

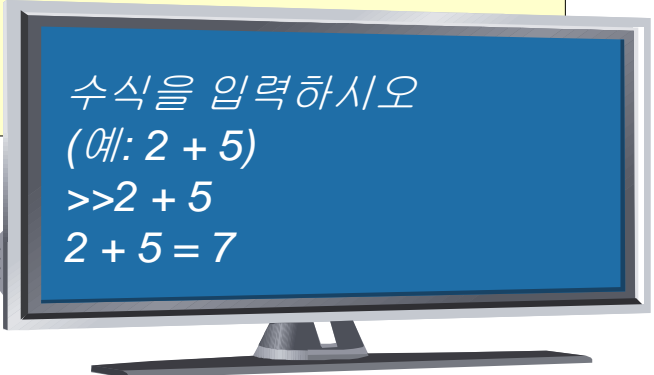
    printf("수식을 입력하시오(예: 2 + 5) >> ");
    scanf("%d %c %d", &x, &op, &y);
```

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;
```

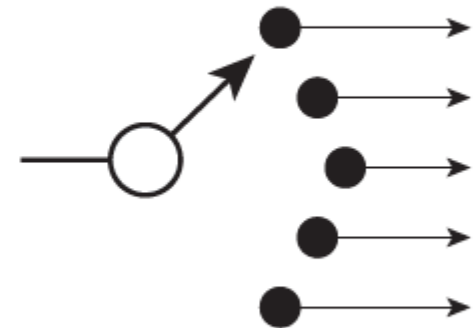
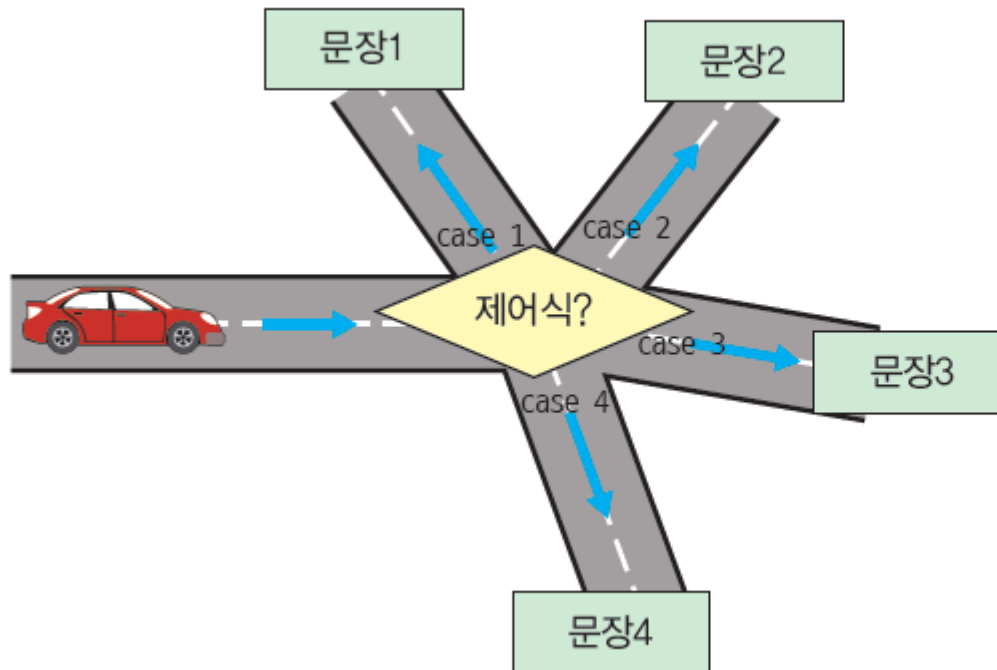
```
}
```



수식을 입력하시오
(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:
```

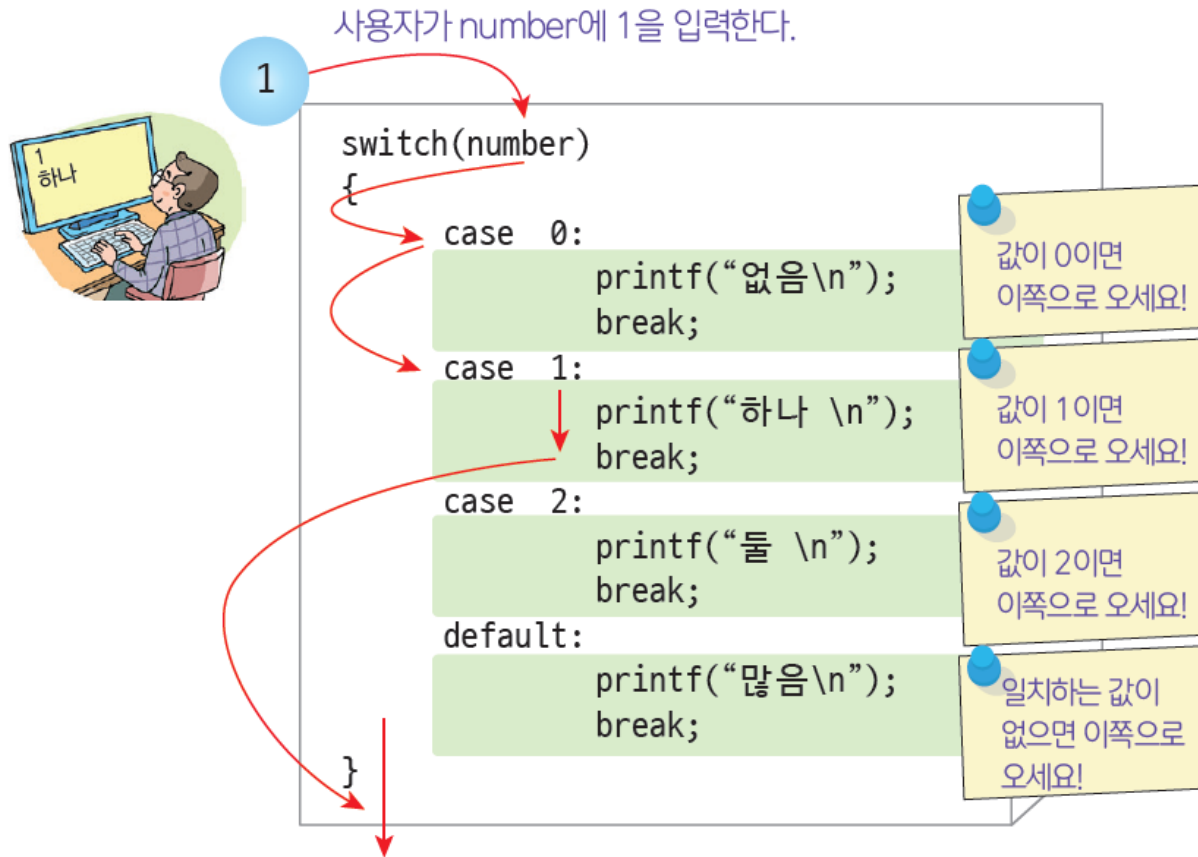
```
문장d;
```

```
break;
```

일치하는 값이 없으면 실행된다.

```
}
```

사용자가 1을 입력하는 경우



break가 생략되는 경우



1

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

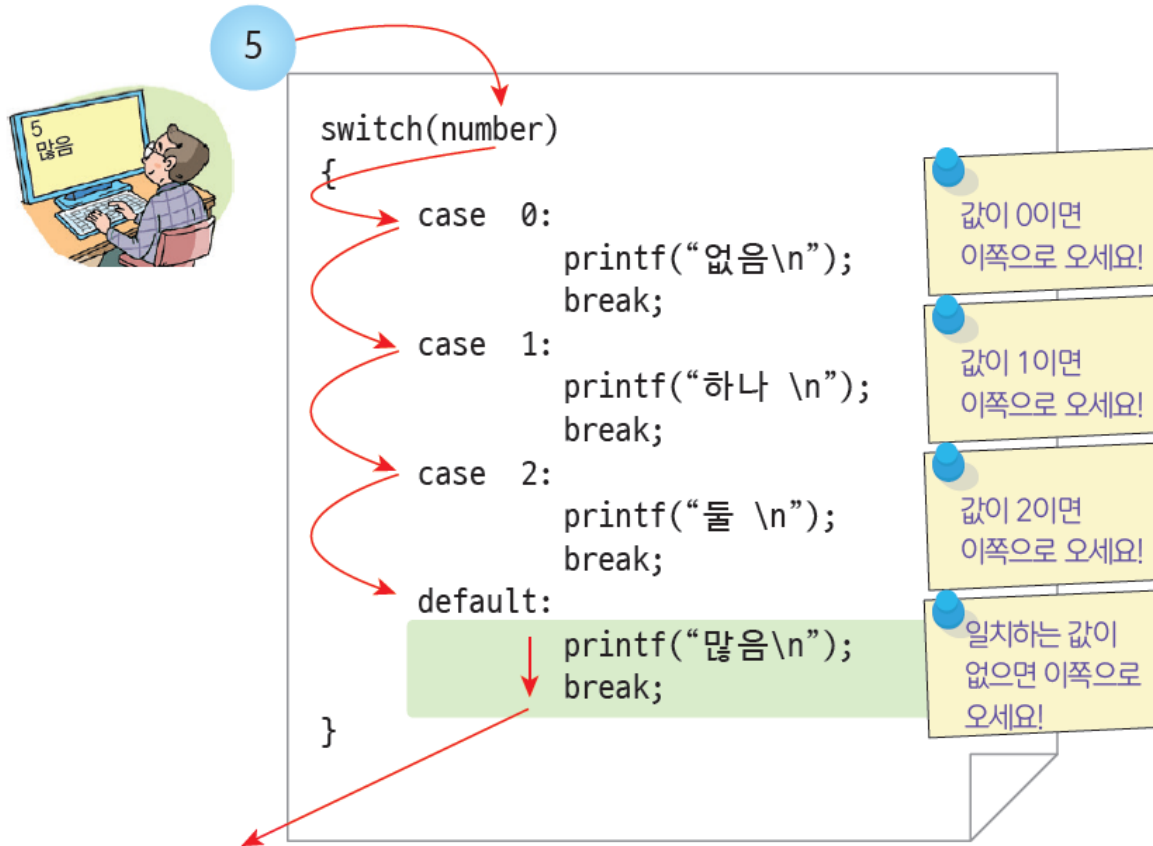
의도적인 break생략



2

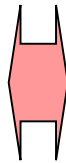
```
switch(number)
{
    case 0:
        printf("없음\n");
        break;
    case 1:
        printf("하나 \n");
        break;
    case 2:
    case 3:
        printf("두서너 개 \n");
        break;
    default:
        printf("많음\n");
        break;
}
```

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 "abc":            // 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
}
```

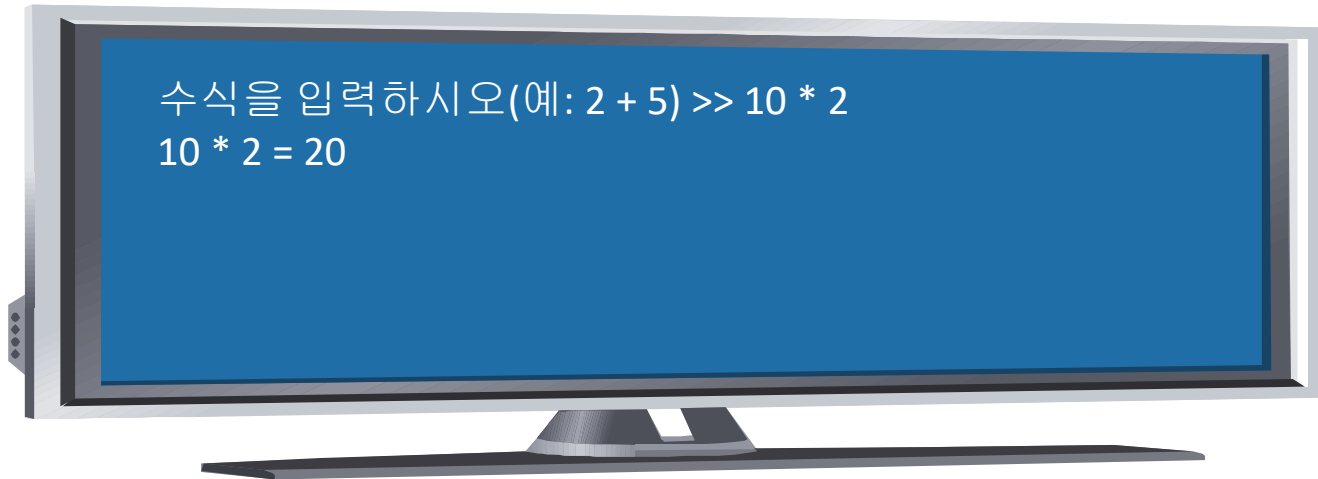


switch 문과 if/else 체인 중에서 어떤 것이 더 효율적인가?

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

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)
{

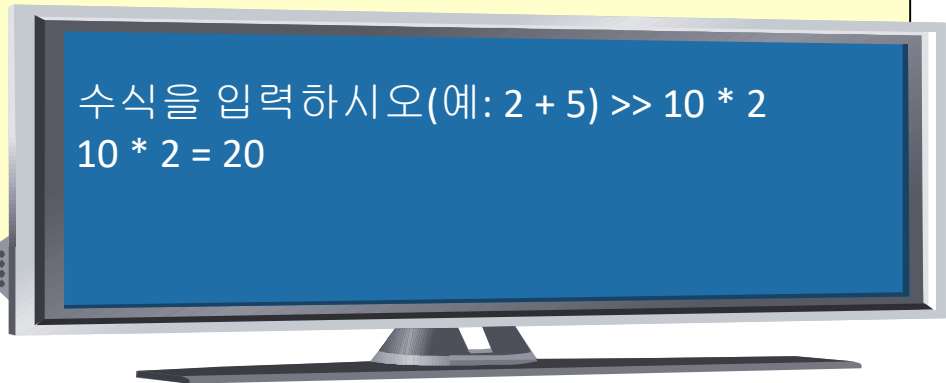
    char op;
    int x, y, result;
    printf("수식을 입력하시오(예: 2 + 5) >> ");
    scanf("%d %c %d", &x, &op, &y);
```

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;  
}
```



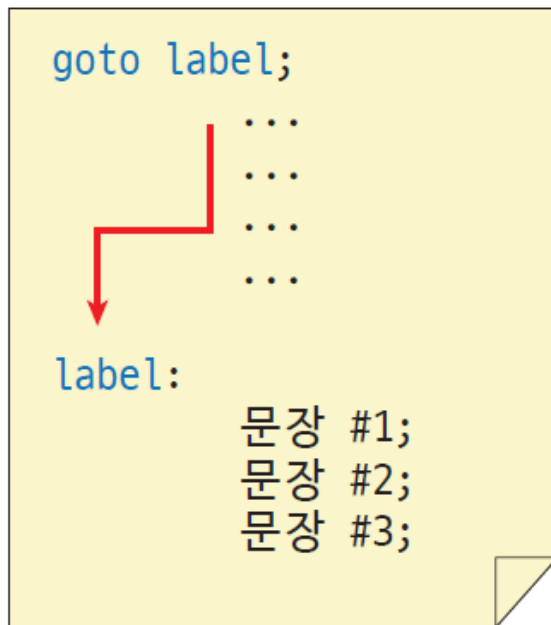
수식을 입력하시오(예: 2 + 5) >> 10 * 2
10 * 2 = 20

goto문

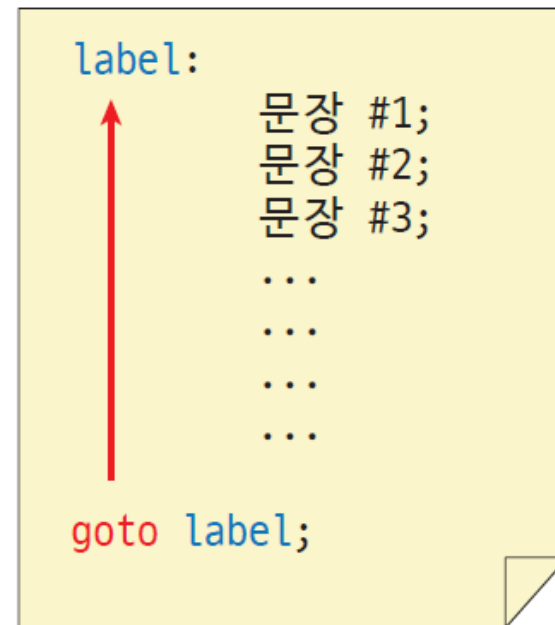
- 조건없이 어떤 위치로 점프
- 사용하지 않는 것이 좋음



goto 문



전향 참조



후향 참조

예제

```
// 구구단출력프로그램
```

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

```
int main(void)
```

```
{
```

```
    int i = 1;
```

```
loop:
```

```
    printf("%d * %d = %d Wn", 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

Q & A

