

강03. 분기문/반복문

7장. 다양한 형태의 반복문

- 반복문의 기능
 - 특정 영역을 특정 조건이 만족하는 동안에 반복 실행하기 위한 문장
- 세 가지 형태의 반복문
 - while문에 의한 반복
 - do ~ while문에 의한 반복
 - for문에 의한 반복

- while문의 기본 원리와 의미

```
while( 반복 조건 )  
{  
    반복 내용  
}
```

“반복의 조건” 이 만족되는 동안
“반복 내용” 을 반복 실행하라.

```
while( i<10 )  
{  
    printf("Hello World! \n");  
    i++;  
}
```

“i<10” 이 만족되는 동안
“printf()와 i++” 을 반복 실행하라.

- while 문의 종괄호

- 반복하고자 하는 영역이 둘 이상의 문장으로 구성되는 경우에 필수

```
while(i<10)
    printf("Hello World! \n"), i++;
```

- 무한 루프(반복)

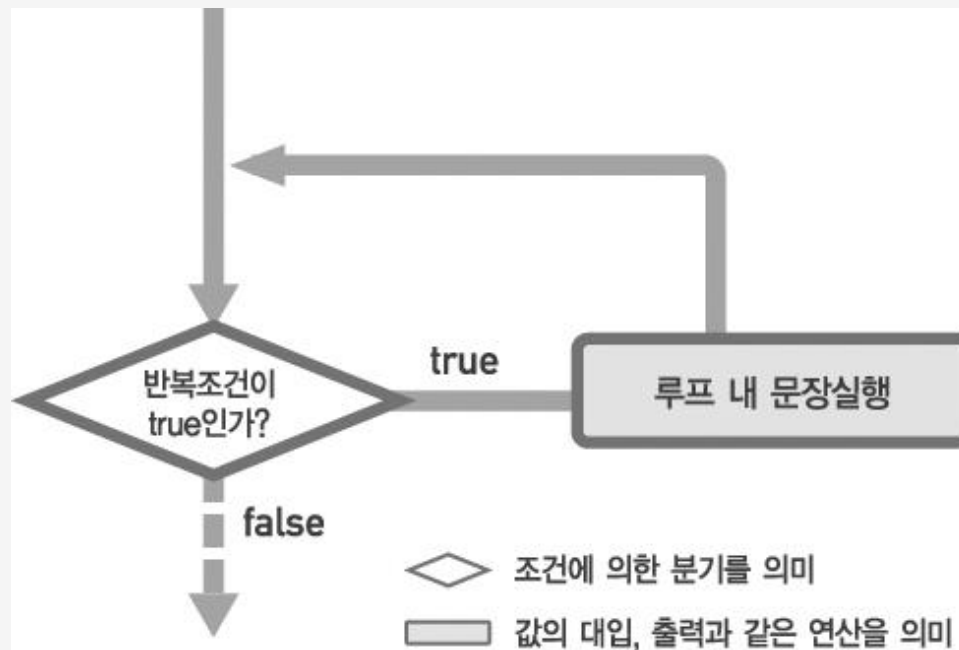
- 반복의 조건으로 true가 오면 발생

```
while( 1 ) // 반복의 조건 대신 0 이 아닌 정수를 넣는다.
{
    printf("Hello World! \n");
    i++;
}
```

- while 문의 중첩
 - while문 안에 while문을 포함시킨다는 뜻
 - 반복 구조 내에서 또 하나의 반복 구조 형성

```
int i=0, j=0;
int num=0;
while(i<10) {
    while(j<10) {
        num++;
        j++;
    }
    i++;
    j=0;
}
```

- while문의 순서도



- do~while문과 while문의 차이점
 - do~while문은 일단 한번 실행하고 나서 조건 검사를 진행

```
do
{
    반복 내용

} while( 반복의 조건 );
```

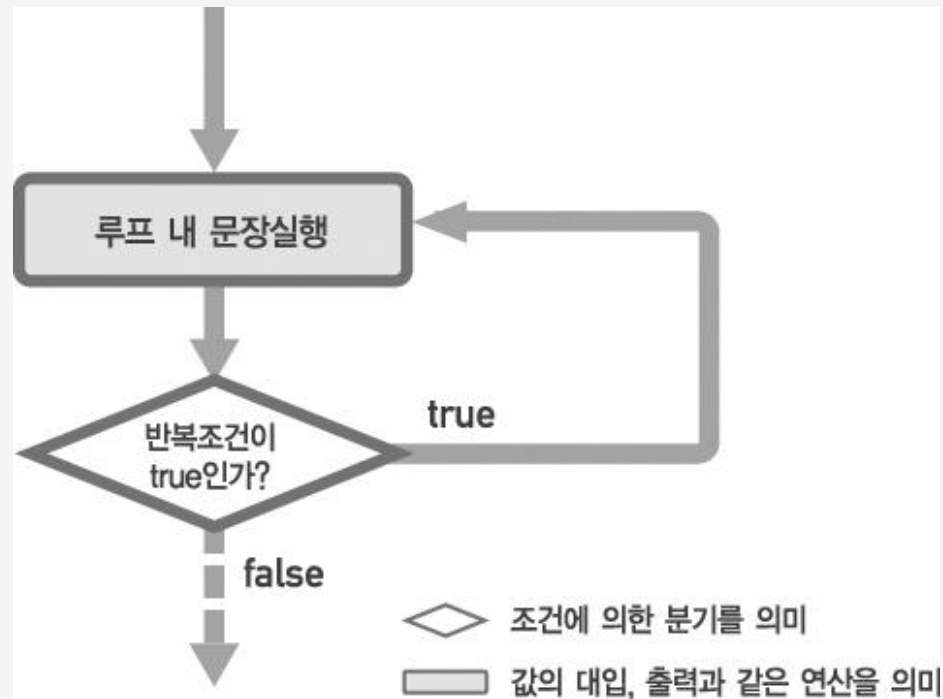
일단 루프를 한번 실행 후, “반복의 조건” 이 만족되는 동안 “반복 내용” 을 반복 실행하라.

```
do
{
    printf("%d*d=%d \n", val, i, val*i);
    i++;

} while( i<10 );
```

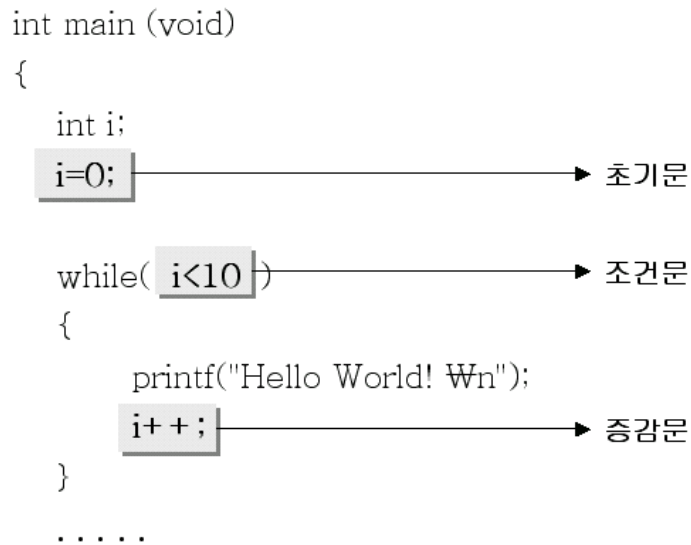
일단 한번 “printf()와 i++” 를 실행후 “i<10” 이 만족되는 동안 “printf()와 i++” 를 반복 실행하라.

- do~while문의 순서도

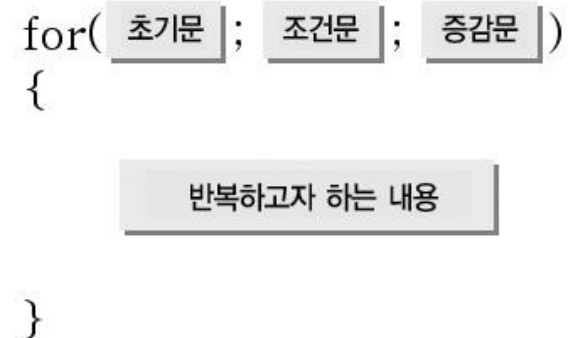


- for문의 기본 원리와 의미
 - 초기문, 조건문, 증감문 모두를 기본적으로 포함!
 - 가장 많이 사용되는 반복문

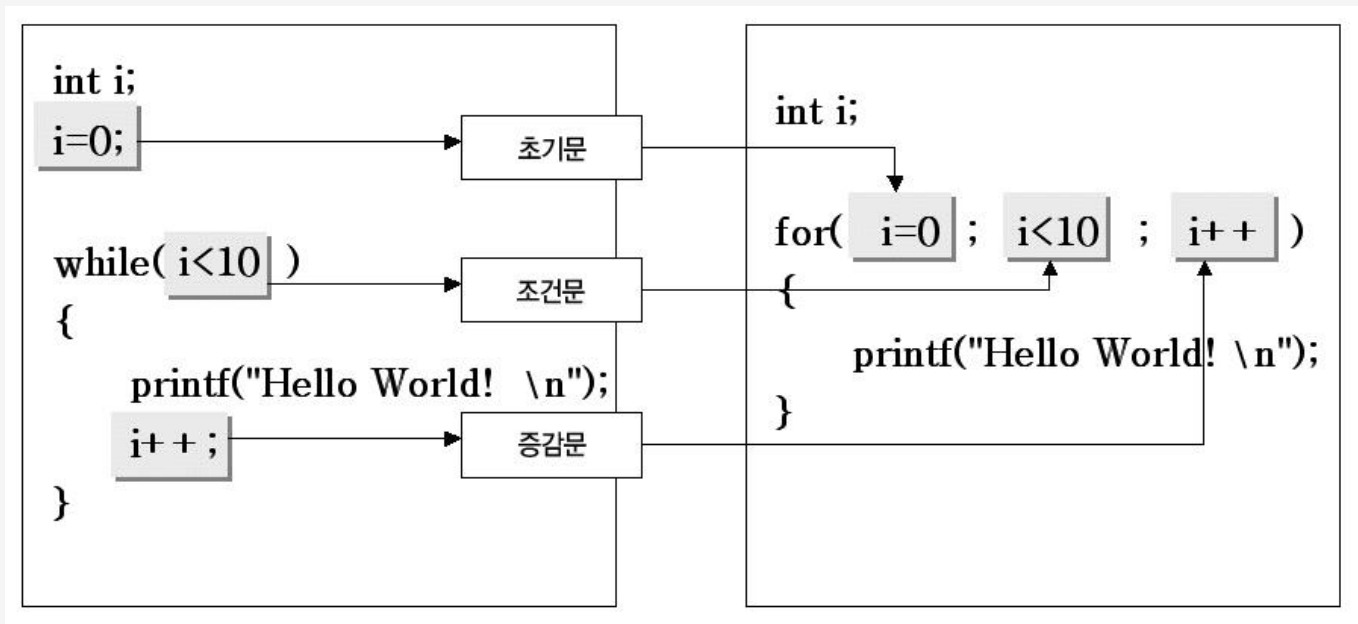
```
int main (void)
{
    int i;
    i=0;
    while( i<10 )
    {
        printf("Hello World! \n");
        i++;
    }
    . . . . .
```



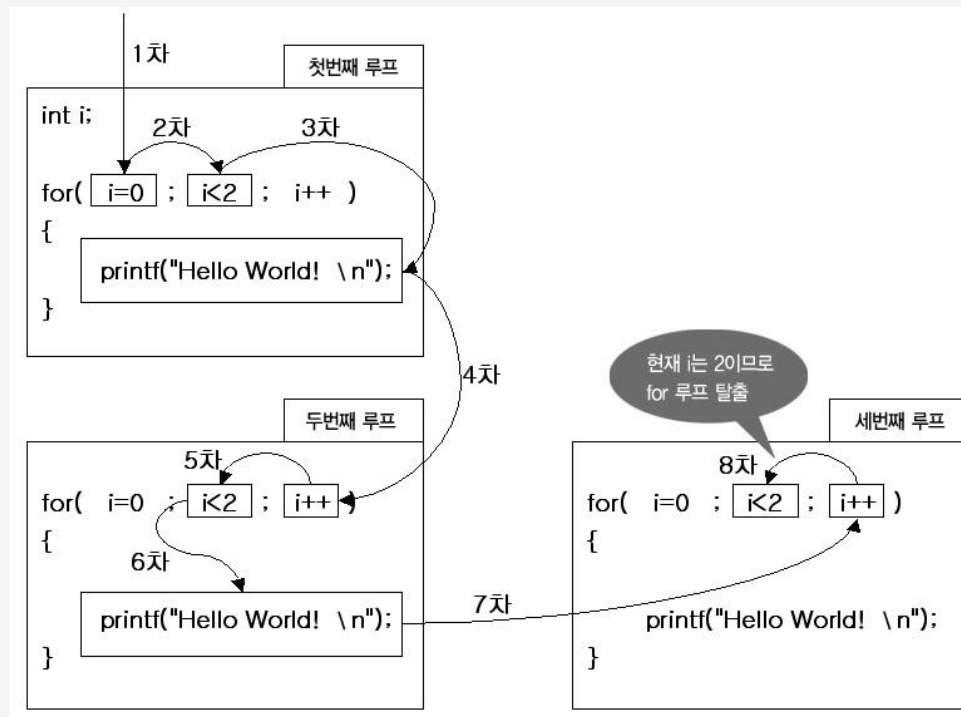
```
for( 초기문 ; 조건문 ; 증감문 )
{
    반복하고자 하는 내용
}
```



- for문과 while문의 비교

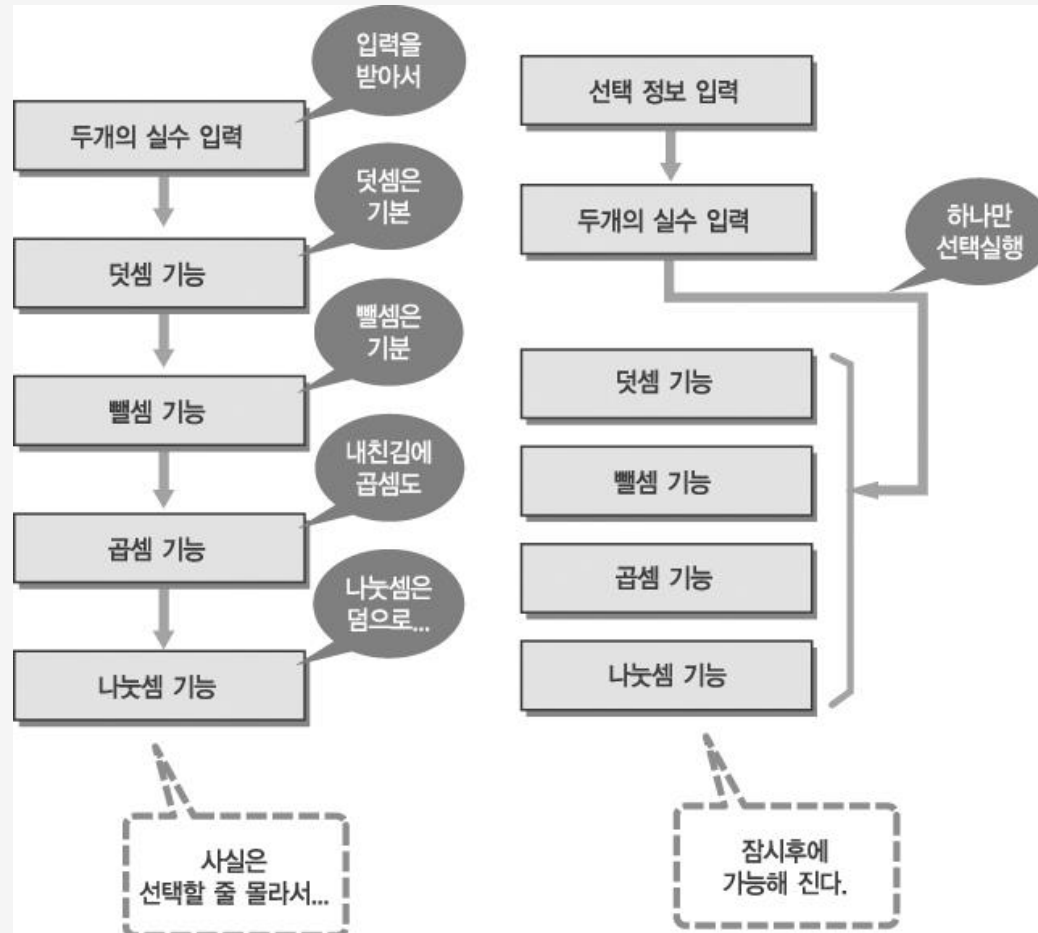


- 반복 과정의 이해



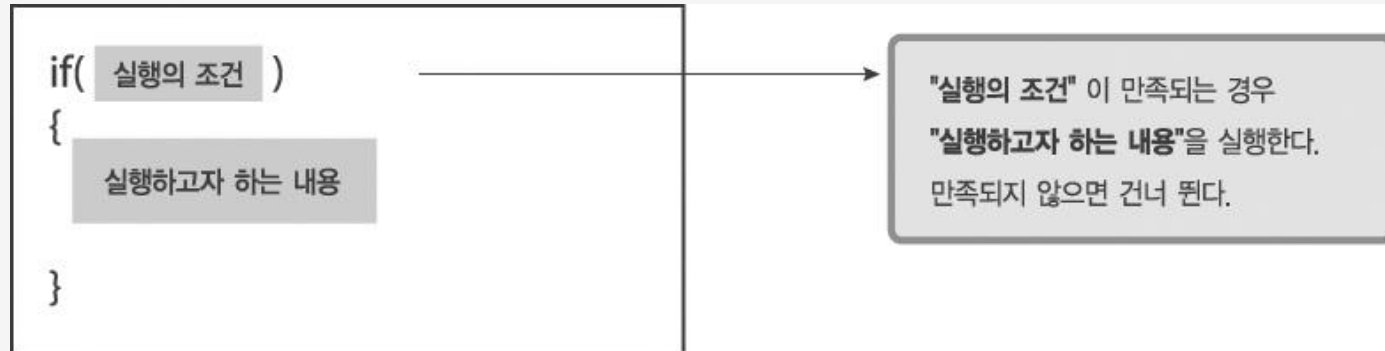
8장. 조건에 따른 흐름의 분기

- 상황에 따른 프로그램의 유연성 부여



C/C++ 8-2 if와 else

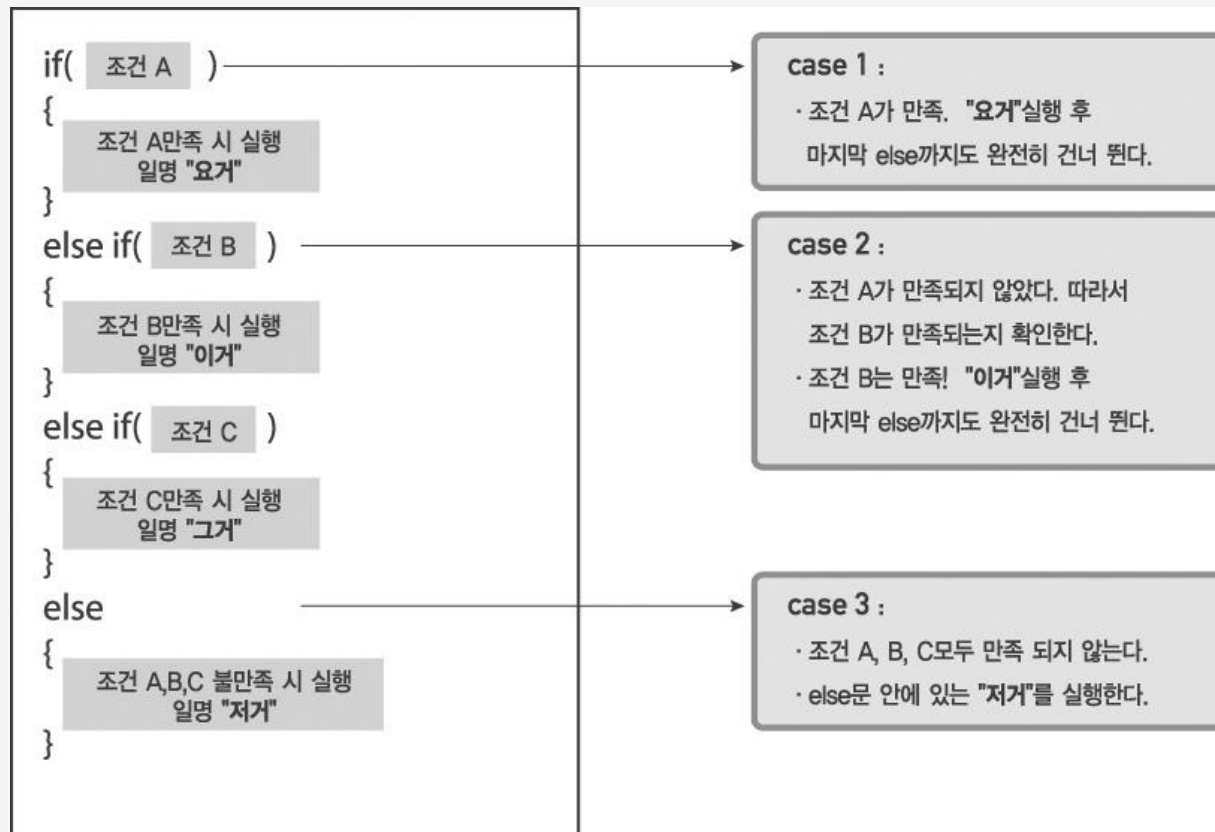
- if문에 의한 조건적 실행
 - 조건이 만족되는 경우에 한해서 실행



- if~else 에 대해서
 - 단점: 불필요한 연산을 하게 된다.



- if, else if, else...



C/C++ 8-2 if와 else

- if, else if, else에 대한 진실
 - if~else문은 하나의 문장이다.
 - if~else문의 중첩된 형태에 지나지 않는다.

- 조건 연산자(삼항 연산자)
 - if~else문을 간결히 표현하는데 사용될 수 있다.

조건 ? A : B

조건이 true인 경우 A를 반환
조건이 false인 경우 B를 반환

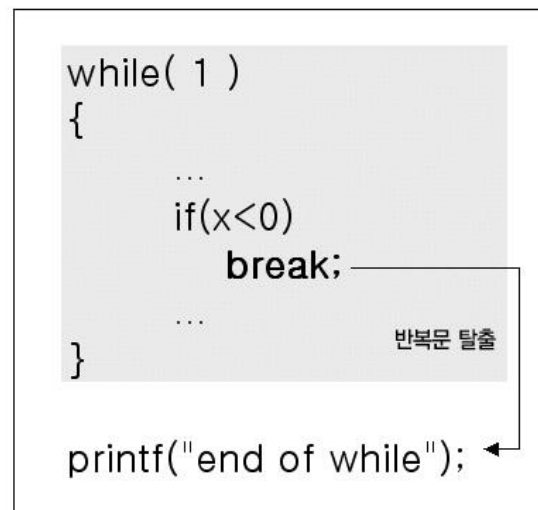
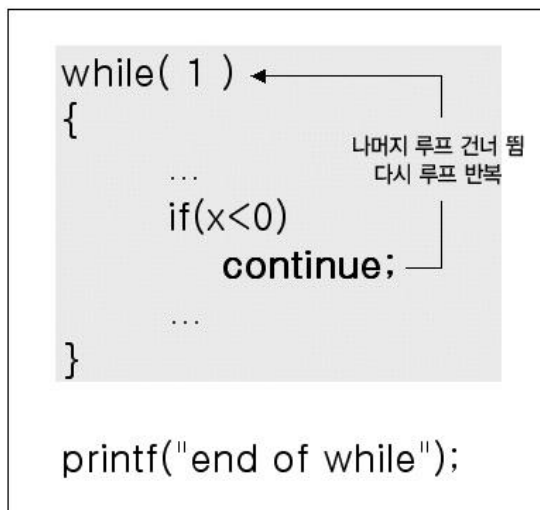
$X = (y < 0) ? 10 : 20;$

"y<0"이 true면 10이 반환되어 x에 대입
"y<0" 이 false면 20이 반환되어 x에 대입

$X = (y > 0) ? a * b : a / b;$

"y>0"이 true면 a*b이 연산결과 x에 대입
"y>0" 이 false면 a/b이 연산결과 x에 대입

- 이제 그만 break!(탈출)
 - 반복문을 빠져 나올 때 사용
- 다음으로 넘어가자 continue!(생략)
 - 다음 번 반복으로 넘어갈 때 사용



- switch문의 구조

```
switch ( n )
```

```
{
```

```
case 1 :
```

n이 1인 경우 실행되는 영역
break;

```
case 2 :
```

n이 2인 경우 실행되는 영역
break;

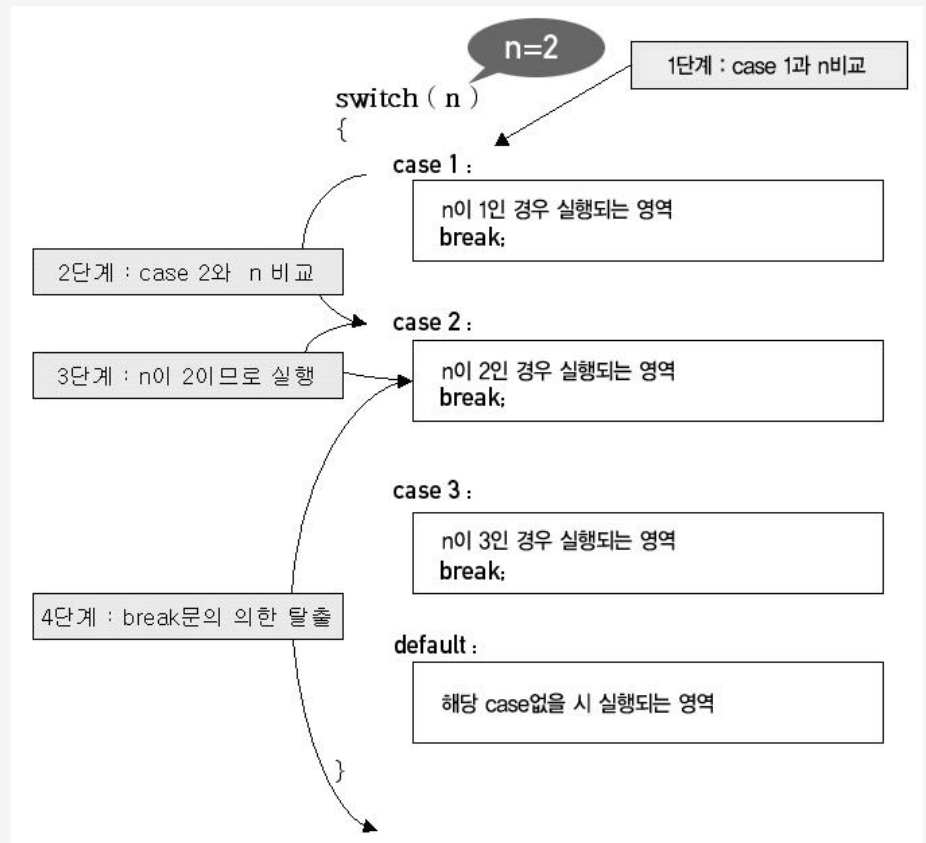
```
case 3 :
```

n이 3인 경우 실행되는 영역
break;

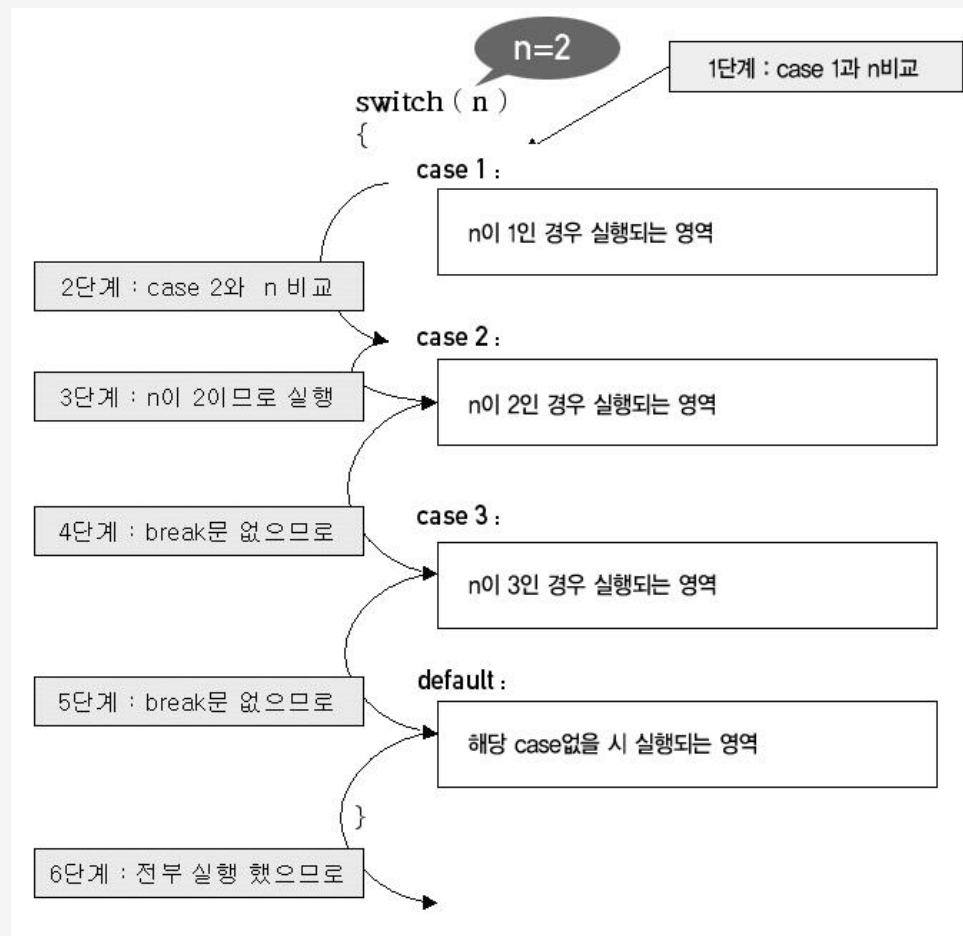
```
default :
```

해당 case 없을 시 실행되는 영역

```
}
```



- switch문에서 break문의 의미



- switch vs. if~else 1
 - 분기의 경우 수가 많아지면 가급적 switch문으로...

if ~else문에 의한 구현

```
if(n==1)
{
    printf("AAA");
}
else if(n==2)
{
    printf("BBB");
}
else if(n==3)
{
    printf("CCC");
}
else
{
    printf("EEE");
}
```

switch문에 의한 구현

```
switch(n)
{
    case 1 :
        printf("AAA");
        break;

    case 2 :
        printf("BBB");
        break;

    case 3 :
        printf("CCC");
        break;

    default :
        printf("EEE");
}
```

- switch vs. if~else 2
 - switch문에서는 비교 연산이 올 수 없다.

if ~else문에 의한 구현

```
if(0<n && n<5)
{
    printf("AAA");
}
else if(5<n && n<10)
{
    printf("BBB");
}
else if(n<10 && n<15)
{
    printf("CCC");
}
else
{
    printf("EEE");
}
```

switch문에 의한 구현

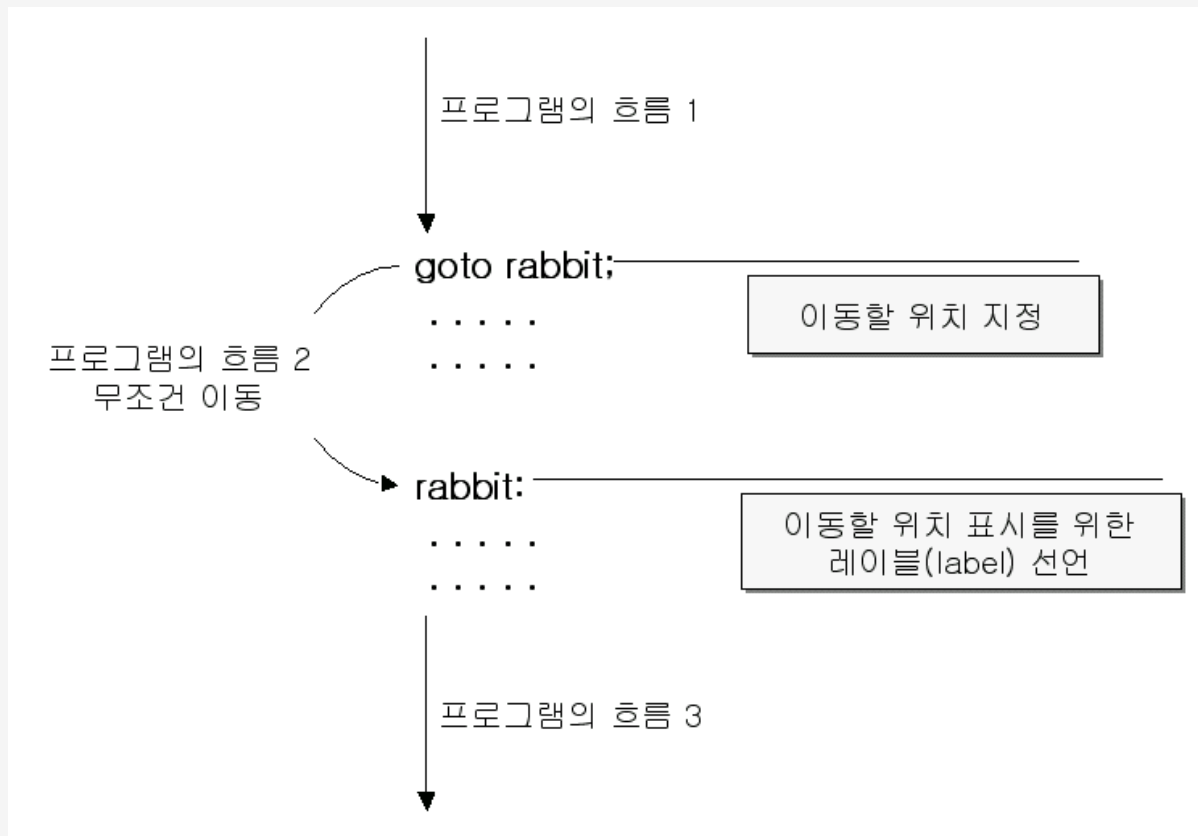
```
switch(n)
{
    case ?? :
        printf("AAA");
        break;

    case ??:
        printf("BBB");
        break;

    case ?? :
        printf("CCC");
        break;

    default :
        printf("EEE");
}
```


- GOTO label 문
 - 프로그램의 흐름을 복잡하게 한다.
 - 주의해서 사용. 가급적 사용하지 말자!



- 실습 문제

- if 문을 이용해서 숫자키를 누르면 해당하는 영어 단어를 출력하는 프로그램을 작성하라.

```
>1 : One
>2 : Two
>3 : Three
>4 : Four
>5 : Five
>6 : Six
>7 : Seven
>8 : Eight
>9 : Nine
>0 : Zero
```

- switch ~case 문을 이용해서 작성하라