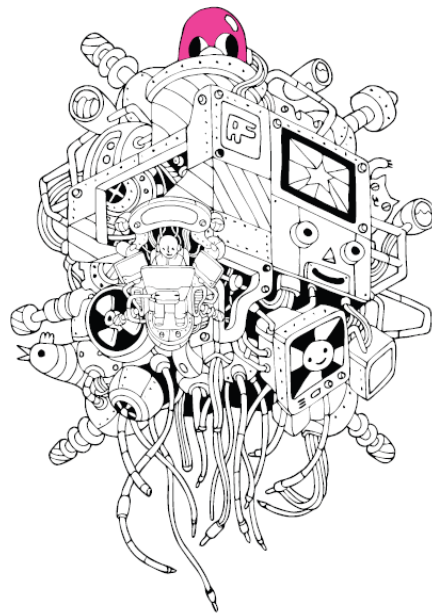


윤성우의 열혈 C 프로그래밍



윤성우 저 열혈강의 C 프로그래밍 개정판

Chapter 08. 조건에 따른 흐름의 분기

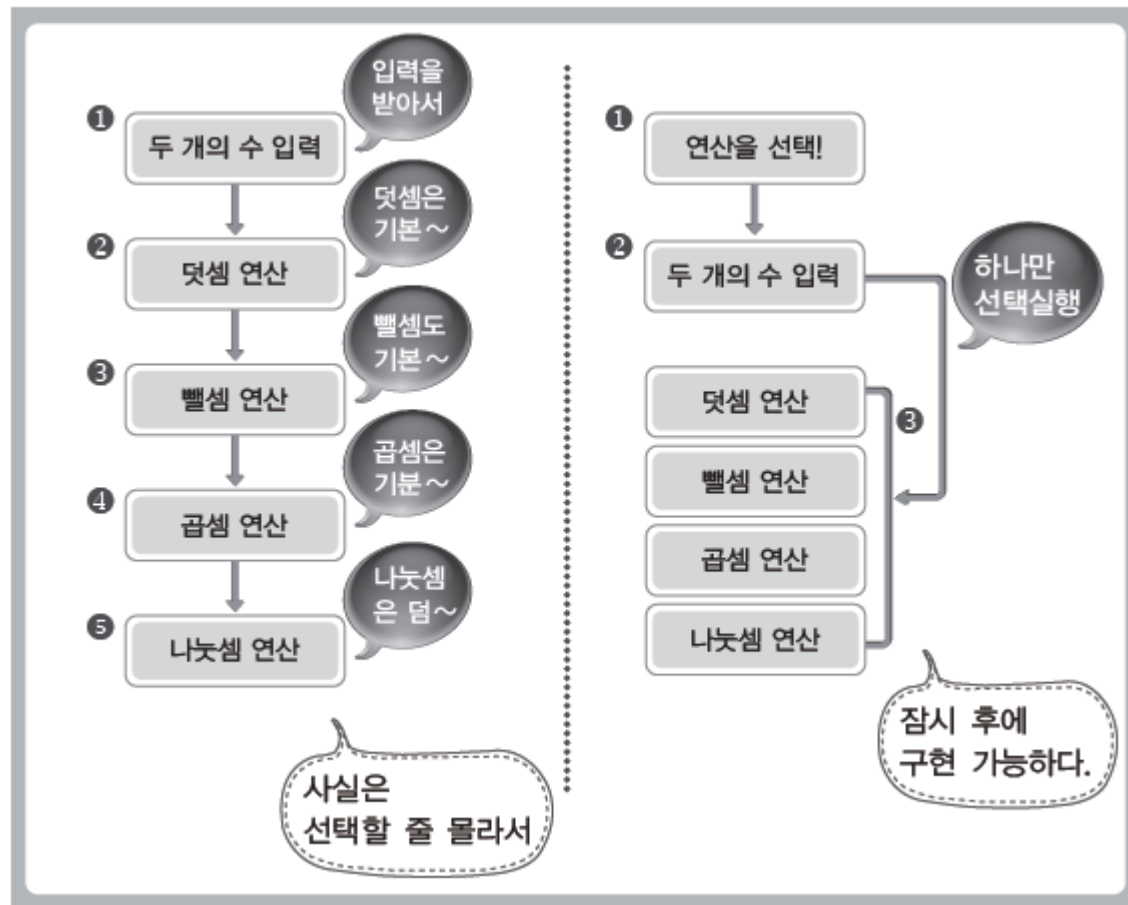
윤성우의 열혈 C 프로그래밍



Chapter 08-1. 조건적 실행과 흐름의 분기

윤성우 저 열혈강의 C 프로그래밍 개정판

흐름의 분기가 필요한 이유



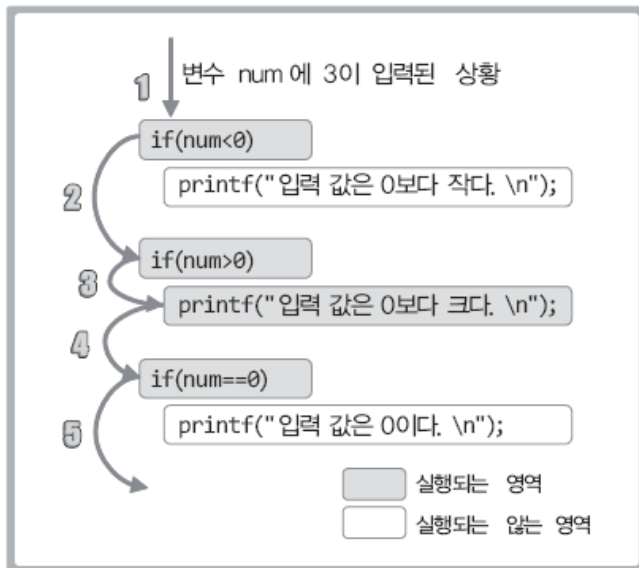
분기하지 못하면 프로그램 사용자는
사칙연산 중 하나를 선택하지 못한다!

프로그램을 구현하다 보면 상황에 따라서 선택적으로 실행해야 하는 영역도 존재하기 마련!

if문을 이용한 조건적 실행

```
if(num1>num2) num1이 num2보다 크면 실행
{
    printf("num1이 num2보다 큽니다. \n");
    printf("%d > %d \n", num1, num2);
}
```

```
if(num1>num2) 한 줄이면 중괄호 생략 가능
    printf("num1이 num2보다 큽니다. \n");
```



```
int main(void)
{
    int num;
    printf("정수 입력: ");
    scanf("%d", &num);

    if(num<0)    // num이 0보다 작으면 아래의 문장 실행
        printf("입력 값은 0보다 작다. \n");

    if(num>0)    // num이 0보다 크면 아래의 문장 실행
        printf("입력 값은 0보다 크다. \n");

    if(num==0)   // num이 0이면 아래의 문장 실행
        printf("입력 값은 0이다. \n");

    return 0;
}
```

정수 입력: 3
입력 값은 0보다 크다. *실행결과1*

정수 입력: 0
입력 값은 0이다. *실행결과2*

if문을 이용한 계산기 프로그램

```
int main(void)
{
    int opt;
    double num1, num2;
    double result;

    printf("1.덧셈 2.뺄셈 3.곱셈 4.나눗셈 \n");
    printf("선택? ");
    scanf("%d", &opt);
    printf("두 개의 실수 입력: ");
    scanf("%lf %lf", &num1, &num2);

    if(opt==1)
        result = num1 + num2;
    if(opt==2)
        result = num1 - num2;
    if(opt==3)
        result = num1 * num2;
    if(opt==4)
        result = num1 / num2;

    printf("결과: %f \n", result);
    return 0;
}
```

이제 계산기 프로그램에 실질적으로 더 가까운 형태가 되었다.

프로그램 구성상 사칙연산 중 하나만 실행이 된다. 그럼에도 불구하고 프로그램 사용자가 덧셈연산을 선택할지라도 총 4번의 조건검사(if문을 통한)를 진행한다는 불합리한 점이 존재한다.

이러한 불합리한 점의 해결에 사용되는 것이 if~else문이다.

실행결과

```
1.덧셈 2.뺄셈 3.곱셈 4.나눗셈
선택? 3
두 개의 실수 입력: 2.14 5.12
결과: 10.956800
```

예제 Mul3Mul4.c도
공부하자!

if~else문을 이용한 흐름의 분기

```
if(num1>num2)    num/이 num2보다 크면 실행
{    // if 블록
    printf("num1이 num2보다 큽니다. \n");
    printf("%d > %d \n", num1, num2);
}
else    num/이 num2보다 크지 않으면 실행
{    // else 블록
    printf("num1이 num2보다 크지 않습니다. \n");
    printf("%d <= %d \n", num1, num2);
}
```

if~else문은 하나의 문장임에 주목하자!

따라서 if와 else 사이에 다른 문장이 삽입될 수 없다.

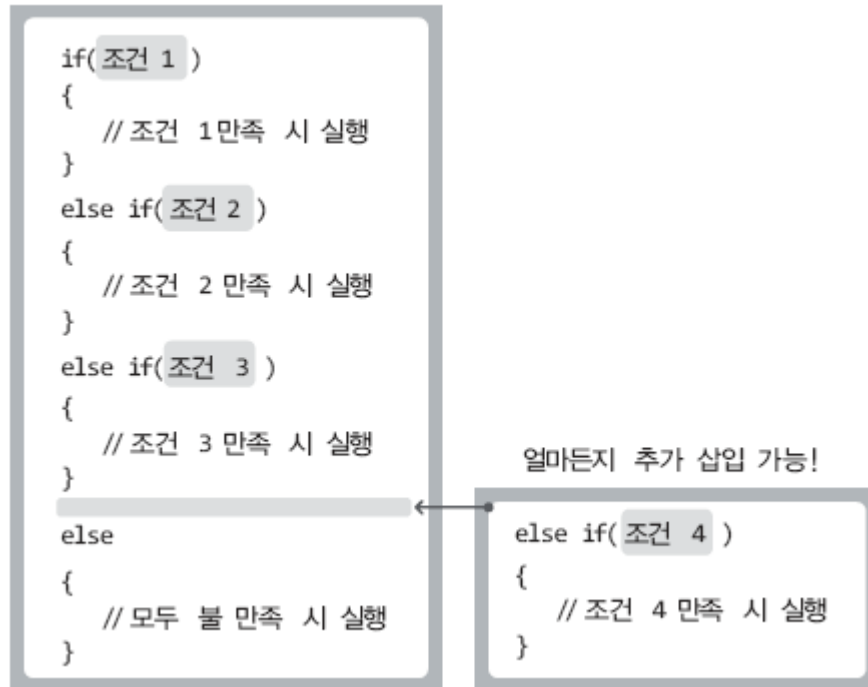
```
int main(void)
{
    int num;
    printf("정수 입력: ");
    scanf("%d", &num);
    if(num<0)
        printf("입력 값은 0보다 작다. \n");
    else
        printf("입력 값은 0보다 작지 않다. \n");

    return 0;
}
```

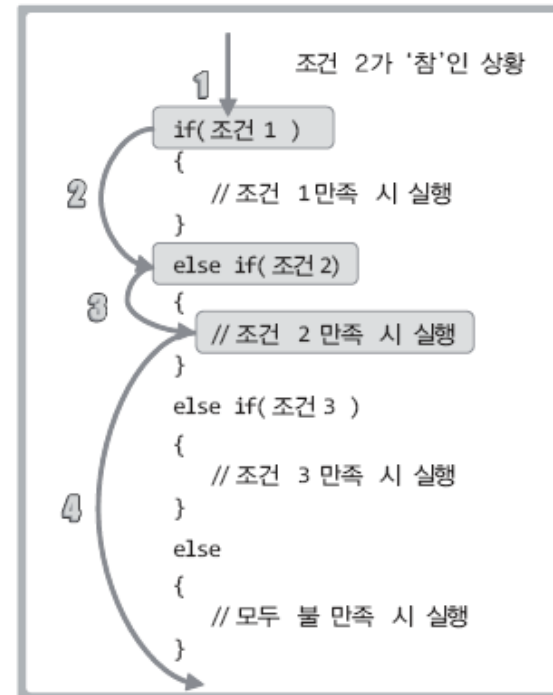
실행결과

정수 입력: 7
입력 값은 0보다 작지 않다.

if...else if...else의 구성



if...else if...else문의 구성



if...else if...else문의 흐름

if...else if...else문의 적용

```
int main(void)
{
    int opt;
    double num1, num2;
    double result;
    printf("1.덧셈 2.뺄셈 3.곱셈 4.나눗셈 \n");
    printf("선택? ");
    scanf("%d", &opt);
    printf("두 개의 실수 입력: ");
    scanf("%lf %lf", &num1, &num2);

    if(opt==1)
        result = num1 + num2;
    else if(opt==2)
        result = num1 - num2;
    else if(opt==3)
        result = num1 * num2;
    else
        result = num1 / num2;

    printf("결과: %f \n", result);
    return 0;
}
```

합리적으로 완성된 사칙연산 계산기 프로그램

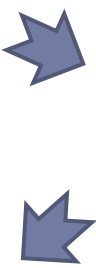


if...else if...else의 진실

```
if(num<0)
    printf("입력 값은 0보다 작다. \n");
else if(num>0)
    printf("입력 값은 0보다 크다. \n");
else
    printf("입력 값은 0이다. \n");
```

```
if(num<0)
{
    printf("입력 값은 0보다 작다. \n");
}
else
{
    if(num>0)
        printf("입력 값은 0보다 크다. \n");
    else
        printf("입력 값은 0이다. \n");
}
```

if~else문은 하나의 문장임을 상기!



```
if(num<0)
    printf("입력 값은 0보다 작다. \n");
else
    if(num>0)
        printf("입력 값은 0보다 크다. \n");
    else
        printf("입력 값은 0이다. \n");
```

else에 하나의 if~else문이 속한 상황.

속한 문장이 하나일 때에는 중괄호를 생략할 수 있다!

조건 연산자: 피 연산자가 세 개인 '삼항 연산자'

```
(num1>num2) ? (num1) : (num2);
```

```
(조건) ? data1 : data2
```

조건이 참이면 data1 반환, 거짓이면 data2 반환

```
int num3 = (num1>num2) ? (num1) : (num2);
```

```
int num3 = num1;  num1>num2가 참이면
```

```
int num3 = num2;  num1>num2가 거짓이면
```

```
int main(void)
{
    int num, abs;
    printf("정수 입력: ");
    scanf("%d", &num);

    abs = num>0 ? num : num*(-1);
    printf("절댓값: %d \n", abs);
    return 0;
}
```

실행결과

정수 입력: -79
절댓값: 79

윤성우의 열혈 C 프로그래밍



Chapter 08-2. 반복문의 생략과 탈출:
continue & break

윤성우 저 열혈강의 C 프로그래밍 개정판

break! 이제 그만 빠져나가자!

```
int main(void)
{
    int sum=0, num=0;

    while(1)
    {
        sum+=num;
        if(sum>5000)
            break; // break문 실행! 따라서 반복문 탈출
        num++;
    }

    printf("sum: %d \n", sum);
    printf("num: %d \n", num);
    return 0;
}
```

break문은 자신을 감싸는 반복문 하나를 빠져나간다.

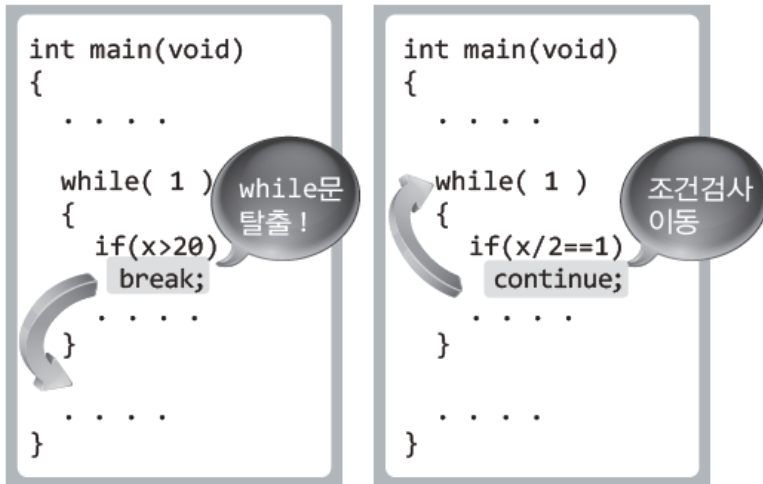
if문과 함께 사용이 되어서 특정 조만이 만족될 때 반복문을 빠져나가는 용도로 주로 사용된다.

실행결과

```
sum: 5050
num: 100
```



continue! 나머지 생략하고 반복조건 확인하러



continue문은 반복문을 빠져나가지 않는다!
다만 반복조건을 확인하러 올라갈 뿐이다.
그리고 반복조건이 여전히 '참'이라면 반복영역을
처음부터 실행하게 된다.

```
int main(void)
{
    int num;
    printf("start! ");

    for(num=1; num<20; num++)
    {
        if(num%2==0 || num%3==0)
            continue;
        printf("%d ", num);
    }
    printf("end! \n");
    return 0;
}
```

start! 1 5 7 11 13 17 19 end!

실행결과

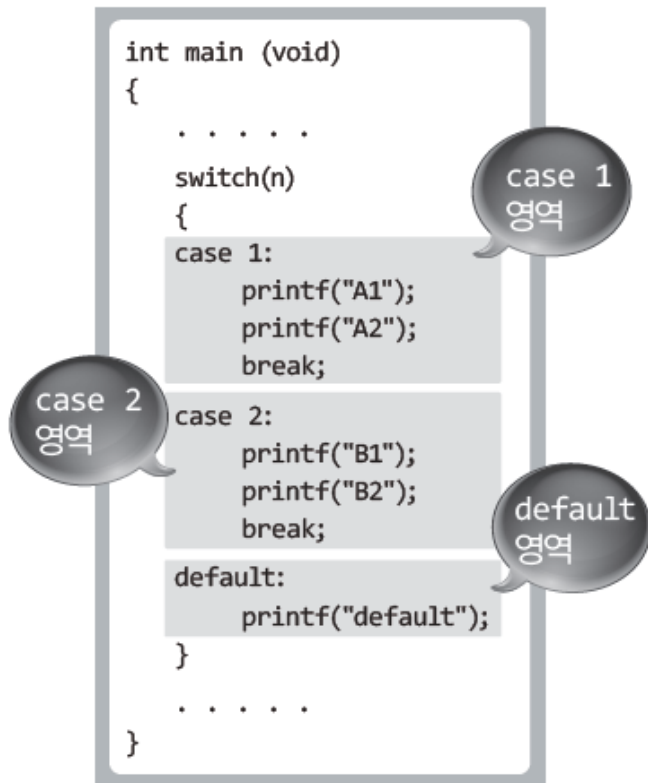
윤성우의 열혈 C 프로그래밍



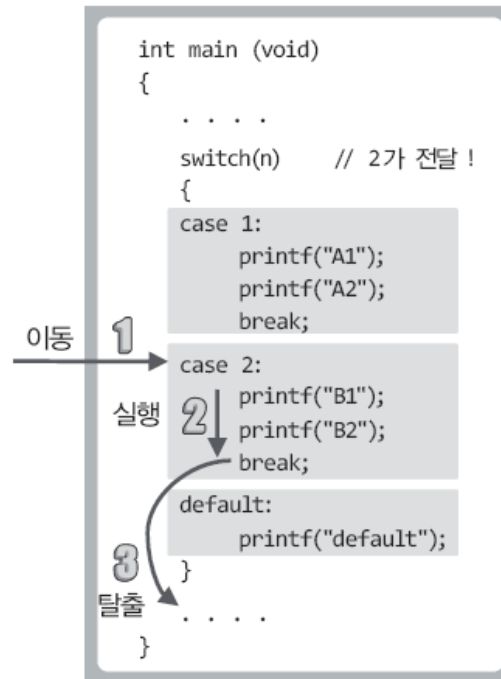
Chapter 08-3. switch문에 의한 선택적
실행과 goto문

윤성우 저 열혈강의 C 프로그래밍 개정판

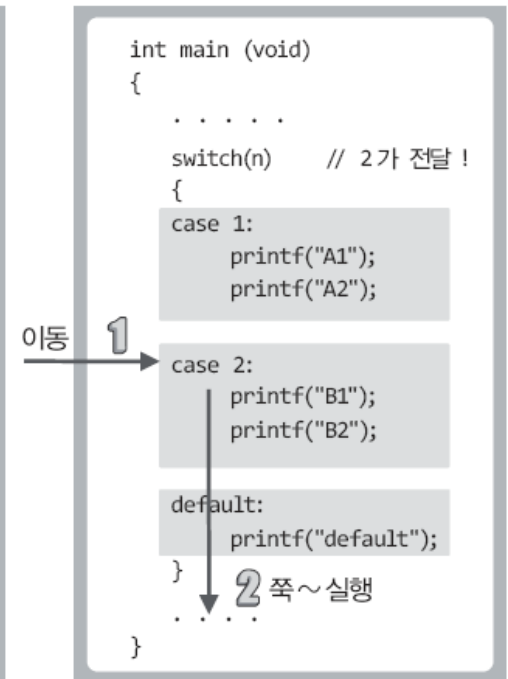
switch문의 구성과 기본기능



switch문의 기본구성



삽입되어 있는 break문이 갖는 의미



switch문 관련 예제

```
int main(void)
{
    int num;
    printf("1이상 5이하의 정수 입력: ");
    scanf("%d", &num);
    switch(num)
    {
        case 1:
            printf("1은 ONE \n");
            break;
        case 2:
            printf("2는 TWO \n");
            break;
        case 3:
            printf("3은 THREE \n");
            break;
        case 4:
            printf("4는 FOUR \n");
            break;
        case 5:
            printf("5는 FIVE \n");
            break;
        default:
            printf("I don't know! \n");
    }
    return 0;
}
```

실행결과1

1이상 5이하의 정수 입력: 3
3은 THREE

실행결과2

1이상 5이하의 정수 입력: 5
5는 FIVE

실행결과3

1이상 5이하의 정수 입력: 7
I don't know!

break문을 생략한 형태의 switch문 구성

```
int main(void)
{
    char sel;
    printf("M 오전, A 오후, E 저녁 \n");
    printf("입력: ");
    scanf("%c", &sel);

    switch(sel)
    {
        case 'M':
        case 'm':
            printf("Morning \n");
            break;
        case 'A':
        case 'a':
            printf("Afternoon \n");
            break;
        case 'E':
        case 'e':
            printf("Evening \n");
            break; // 사실 불필요한 break문!
    }
    return 0;
}
```

원편의 예제와 같은 경우 다음과 같이 두 case 레이블을 한 줄에 같이 표시하기도 한다.

case 'M': case 'm':

.....

case 'A': case 'a':

.....

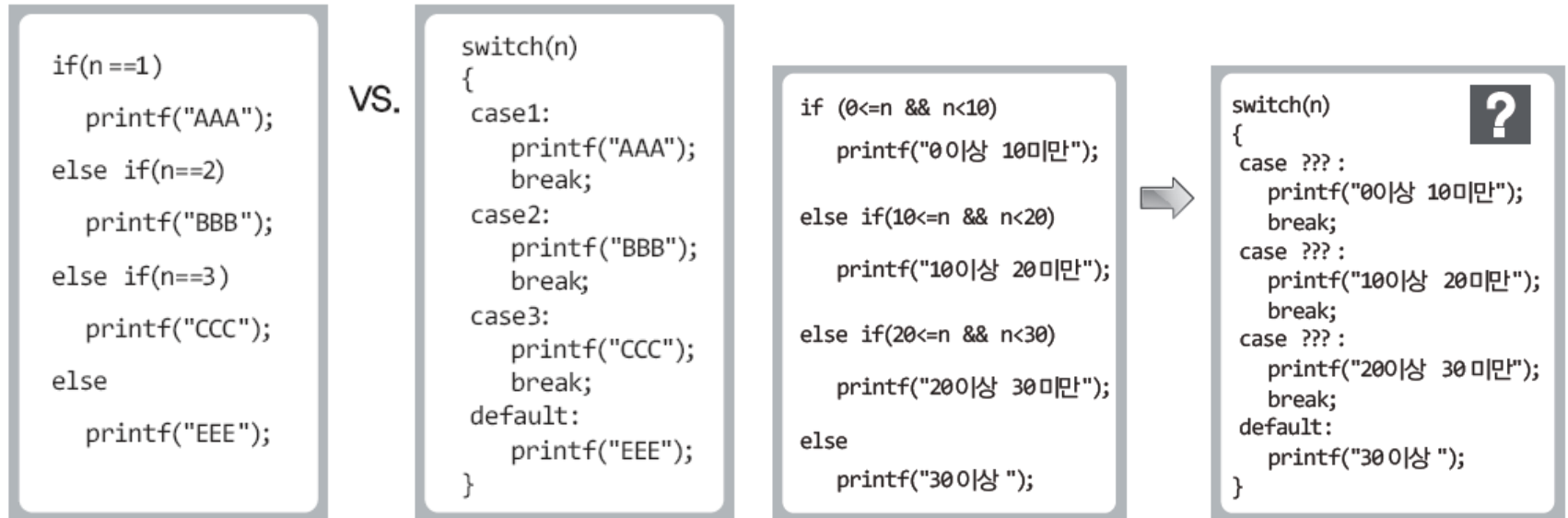
case 'E': case 'e':

.....

실행결과

```
M 오전, A 오후, E 저녁
입력: M
Morning
```

switch vs. if...else if...else



if...else if...else보다 switch문을 선호한다.
switch문이 더 간결해 보이기 때문이다.

모든 if...else if...else문을 switch문으로
대체할 수 있는 것은 아니다.

마지막으로 goto에 대해서 소개합니다.

```
int main(void)
```

```
{
```

```
    . . . .
```

```
    rabbit: 위치를 표시하는 rabbit 레이블
```

```
    . . . .
```

```
    goto rabbit: 레이블 rabbit으로 무조건 이동!
```

```
    . . . .
```

```
}
```

**goto는 단점이 많다. 따라서 이해는 하되 활용은
하지 말자!**

실행결과

```
자연수 입력: 2
2를 입력하셨습니다!
```

```
int main(void)
```

```
{
```

```
    int num;
```

```
    printf("자연수 입력: ");
```

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

```
    if(num==1)
```

```
        goto ONE;
```

```
    else if(num==2)
```

```
        goto TWO;
```

```
    else
```

```
        goto OTHER;
```

```
ONE:
```

```
    printf("1을 입력하셨습니다! \n");
```

```
    goto END;
```

```
TWO:
```

```
    printf("2를 입력하셨습니다! \n");
```

```
    goto END;
```

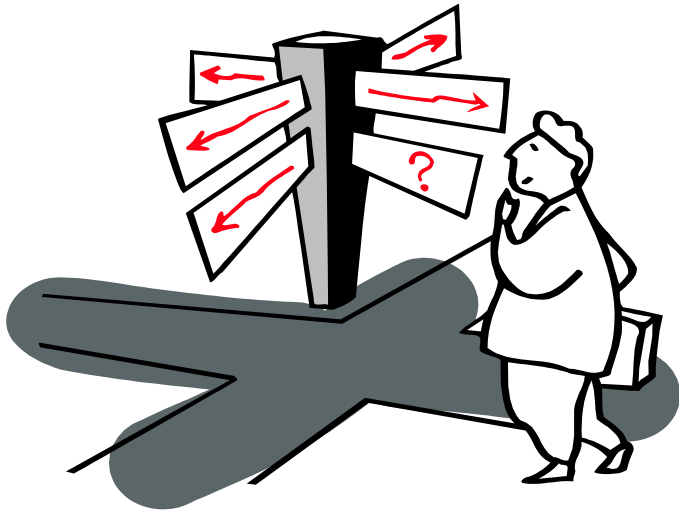
```
OTHER:
```

```
    printf("3 혹은 다른 값을 입력하셨습니다! \n");
```

```
END:
```

```
    return 0;
```

```
}
```



Chapter 08이 끝났습니다. 질문 있으신지요?