

# C 프로그래밍 1

## Lecture Note #08

---

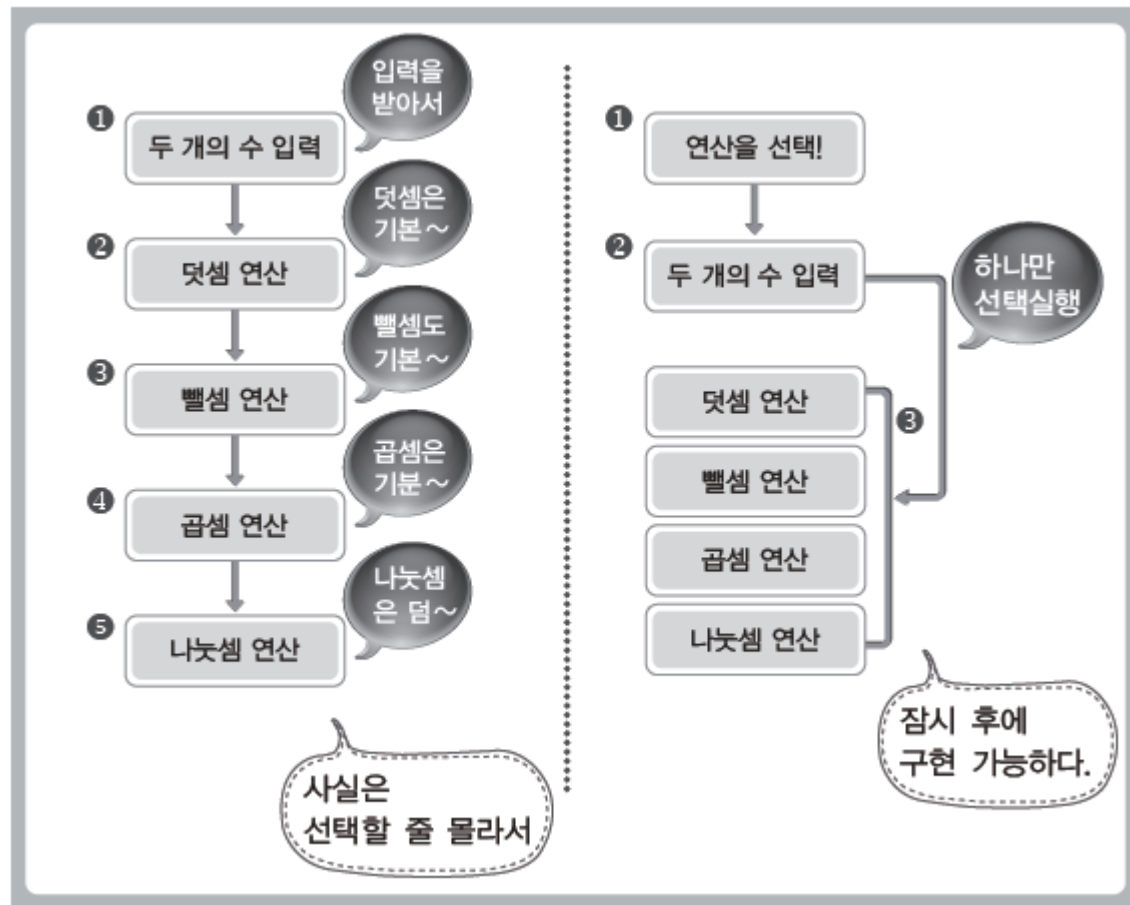
백윤철  
ybaek@smu.ac.kr

# 내용

---

- 조건과 분기
- if
- if else
- 3항연산자
- break, continue
- switch
- goto

# 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");
```

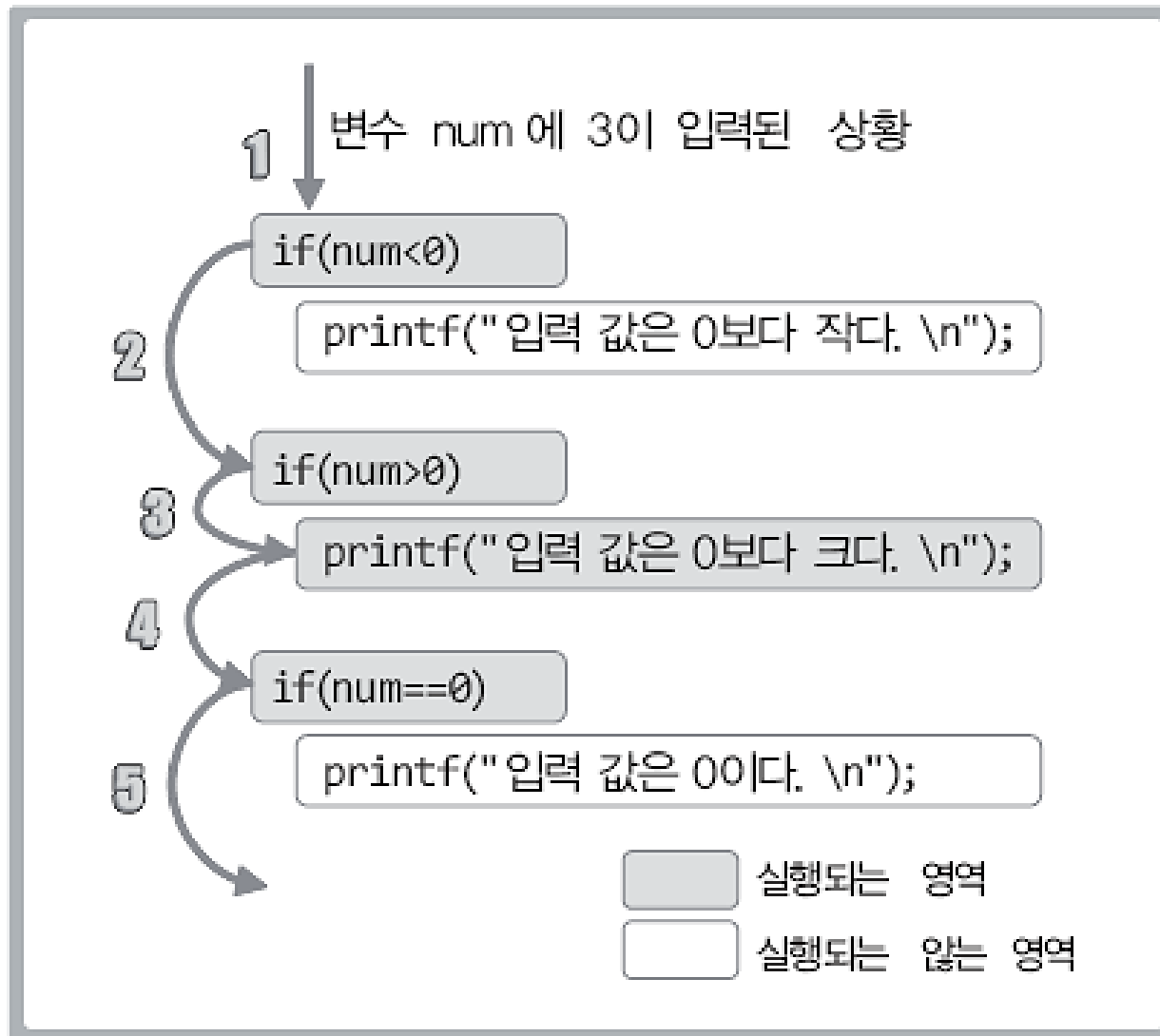
# if문을 이용한 조건적 실행

```
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      실행결과1  
입력 값은 0보다 크다.

정수 입력: 0      실행결과2  
입력 값은 0이다.

# if문을 이용한 조건적 실행



# 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;
```

# if문을 이용한 계산기 프로그램

```
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;
}
```

실행결과

1.덧셈 2.뺄셈 3.곱셈 4.나눗셈  
선택? 3

두 개의 실수 입력: 2.14 5.12  
결과: 10.956800

- 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.



# if~else문을 이용한 흐름의 분기

```
if (num1 > num2) {  
    /* if 블록 */  
    printf("num1이 num2보다 큽니다.\n");  
    printf("%d > %d\n", num1, num2);  
}  
else {  
    /* 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.

# if~else문을 이용한 흐름의 분기

```
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.

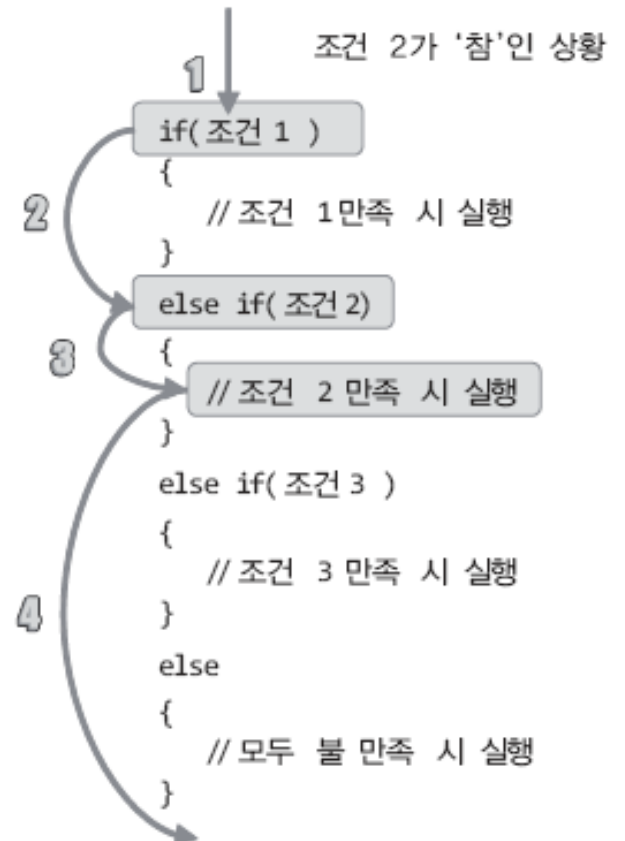
# if...else if...else의 구성

```
if( 조건 1 )  
{  
    // 조건 1만족 시 실행  
}  
else if( 조건 2 )  
{  
    // 조건 2 만족 시 실행  
}  
else if( 조건 3 )  
{  
    // 조건 3 만족 시 실행  
}  
else  
{  
    // 모두 불 만족 시 실행  
}
```

얼마든지 추가 삽입 가능!

```
else if( 조건 4 )  
{  
    // 조건 4 만족 시 실행  
}
```

## 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;
```

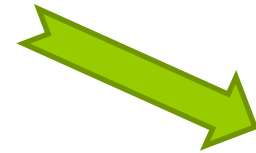
## if...else if...else문의 적용

```
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");
```



else에 하나의  
if~else문이 속한 상황.

```
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...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");  
}
```

# 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
```



## 조건 연산자: 피 연산자가 세 개인 '삼항 연산자'

```
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
```

# break - 이제 그만 빠져나가자

```
int main(void) {  
    int sum = 0, num = 0;  
    while (1) {  
        sum += num;  
        if (sum > 5000)  
            break; /* 반복문 탈출 */  
        num++;  
    }  
    printf("sum: %d\n", sum);  
    printf("num: %d\n", num);  
    return 0;  
}
```

- break문은 자신을 감싸는 반복문 하나를 빠져나감
- if문과 함께 사용되어서 특정 조건이 만족될 때 반복문을 빠져나가는 용도로 주로 사용됨

실행결과

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

## continue - 반복조건 확인하러

```
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;  
}
```

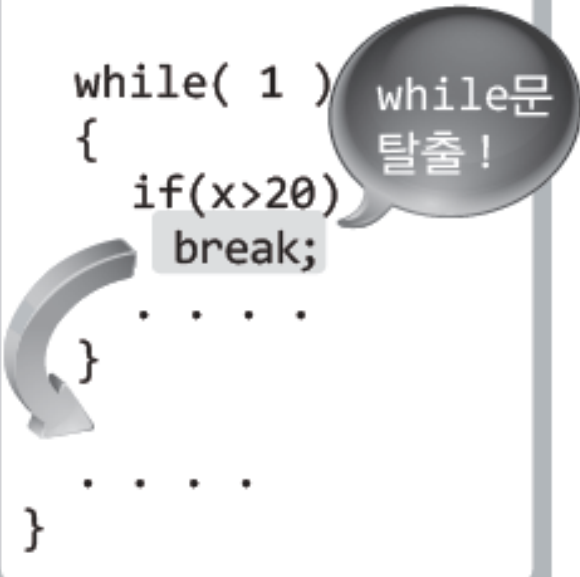
실행결과

start! 1 5 7 11 13 17 19 end!

# break와 continue

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

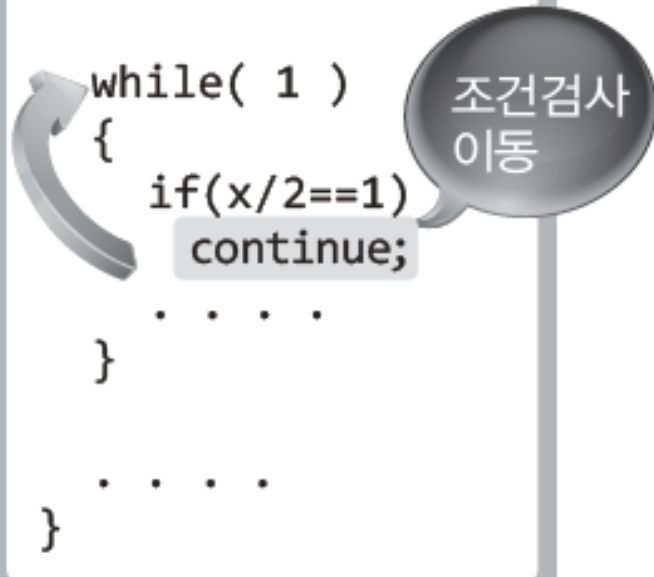
    while( 1 )
    {
        if(x>20)
            break;
        . . . .
    }
    . . . .
}
```



A diagram illustrating the use of the `break` statement. A curved arrow points from the `break;` line to the closing brace of the `while` loop, indicating an exit from the loop. A speech bubble next to the loop contains the text "while문 탈출!" (Exit while loop!).

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

    while( 1 )
    {
        if(x/2==1)
            continue;
        . . . .
    }
    . . . .
}
```



A diagram illustrating the use of the `continue` statement. A curved arrow points from the `continue;` line to the condition check of the `while` loop, indicating a jump to the next iteration. A speech bubble next to the loop contains the text "조건검사 이동" (Move to condition check).

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

# switch 문의 구성과 기본 기능

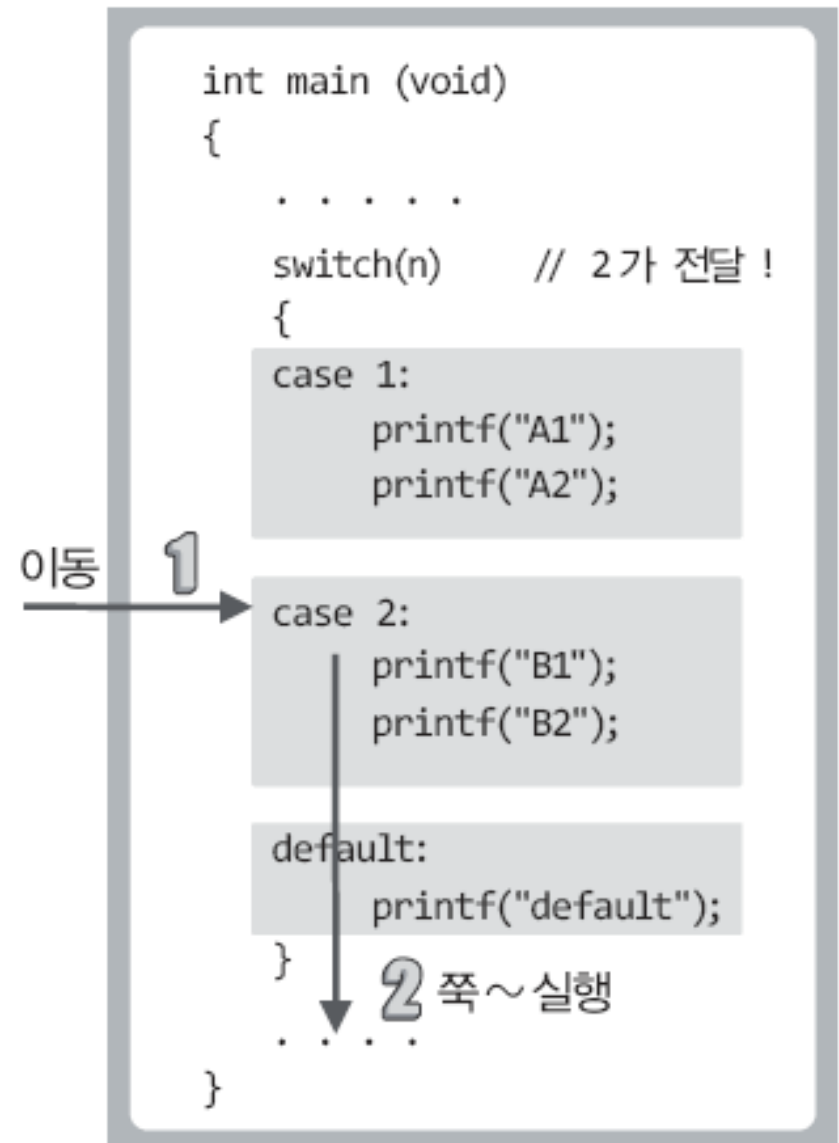
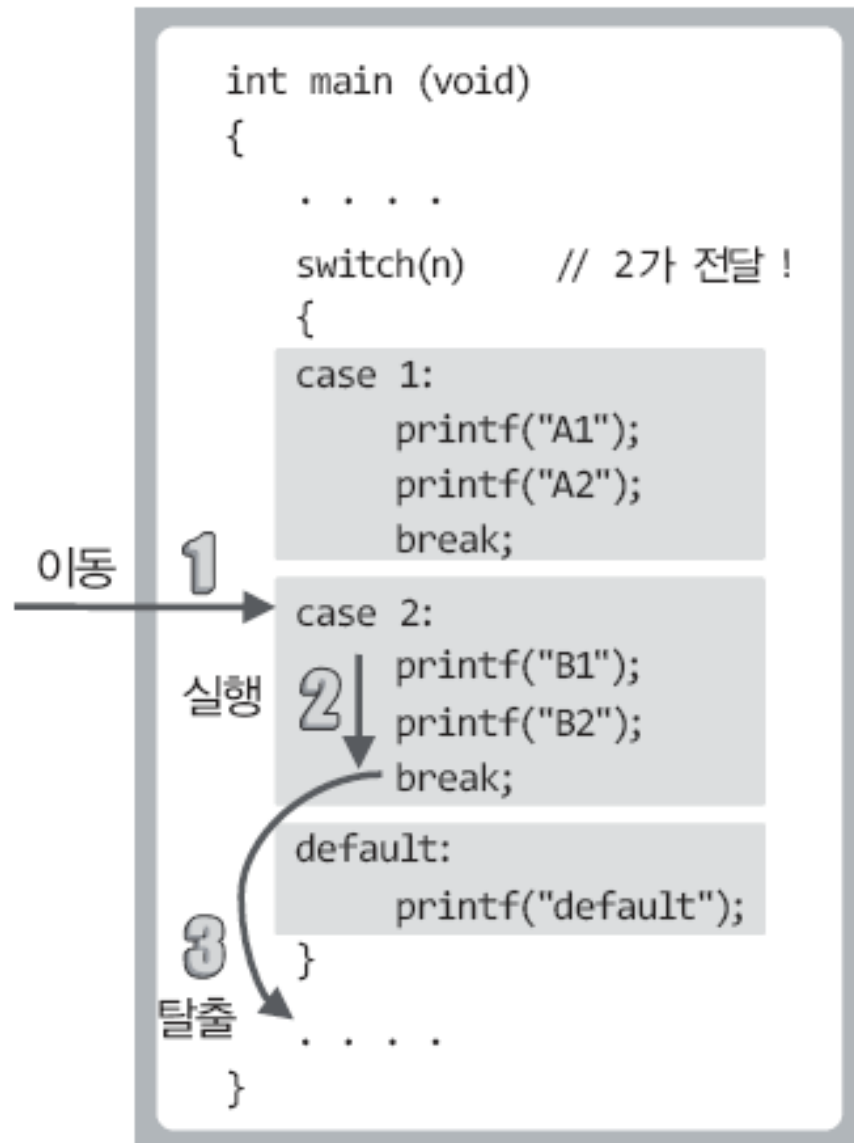
```
int main (void)
{
    . . . . .
    switch(n)
    {
        case 1:
            printf("A1");
            printf("A2");
            break;
        case 2:
            printf("B1");
            printf("B2");
            break;
        default:
            printf("default");
    }
    . . . . .
}
```

case 1  
영역

case 2  
영역

default  
영역

# switch 문의 구성과 기본 기능



## 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;
```

```
}
```

## 실행결과 1

1이상 5이하의 정수 입력: 3

3은 THREE

## 실행결과 2

1이상 5이하의 정수 입력: 5

5는 FIVE

## 실행결과 3

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;  
    }  
}
```

# break문을 생략한 형태의 switch문 구성

```
case 'A':  
case 'a':  
    printf("Afternoon\n");  
    break;  
case 'E':  
case 'e':  
    printf("Evening\n");  
    break; /* 사실 불필요한 break문 */  
}  
return 0;  
}
```

실행결과

M 오전, A 오후, E 저녁  
입력: M  
Morning

# break문을 생략한 형태의 switch문 구성

- 위 예제 같은 경우, 다음과 같이 두 case 레이블을 한 줄에 같이 표시하기도 함

```
case 'M': case 'm':
```

```
...
```

```
case 'A': case 'a':
```

```
...
```

```
case 'E': case 'e':
```

```
...
```

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

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

VS.

```
switch(n)
{
    case1:
        printf("AAA");
        break;
    case2:
        printf("BBB");
        break;
    case3:
        printf("CCC");
        break;
    default:
        printf("EEE");
}
```


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

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

```
if (0<=n && n<10)
    printf("0 이상 10미만");
else if(10<=n && n<20)
    printf("10 이상 20미만");
else if(20<=n && n<30)
    printf("20 이상 30미만");
else
    printf("30 이상 ");
```



```
switch(n)
{
    case ??? :
        printf("0 이상 10미만");
        break;
    case ??? :
        printf("10 이상 20미만");
        break;
    case ??? :
        printf("20 이상 30미만");
        break;
    default:
        printf("30 이상 ");
}
```



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

# goto

```
int main(void)
```

```
{
```

```
    . . . .
```

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

```
    . . . .
```

```
    goto rabbit;
```

```
    . . . .
```

```
}
```

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

# goto 예제

```
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 입력하셨군요! \n");
END:
    return 0;
}
```

실행결과

자연수 입력: 2  
2를 입력하셨습니다!

# 정리

---

- 조건과 분기
- if
- if else
- 3항연산자
- break, continue
- switch
- goto