

# Branching Statements

**Section 1** **if** Statements

**Section 2** Multiple Branches with **if** Statements

**Section 3** **switch** Statements



## 1 if Statements

### 분기문 - 조건문 또는 선택 제어문

#### ● 조건문(선택 제어문)

- 조건에 따라 다른 내용이 실행되도록 실행의 흐름을 제어하는 명령문으로 if문과 switch문이 있다.
- 선택의 기준을 제시하고 상황에 따라 다른 값이나 내용을 적용한다.
- 기준이 부합할 때까지 특정 작업을 반복한다.

#### ● if의 단독 사용

- 조건식이 참일 때 실행, 거짓일 때는 아무 것도 하지 않음

if (조건식) {  
~  
필수  
}

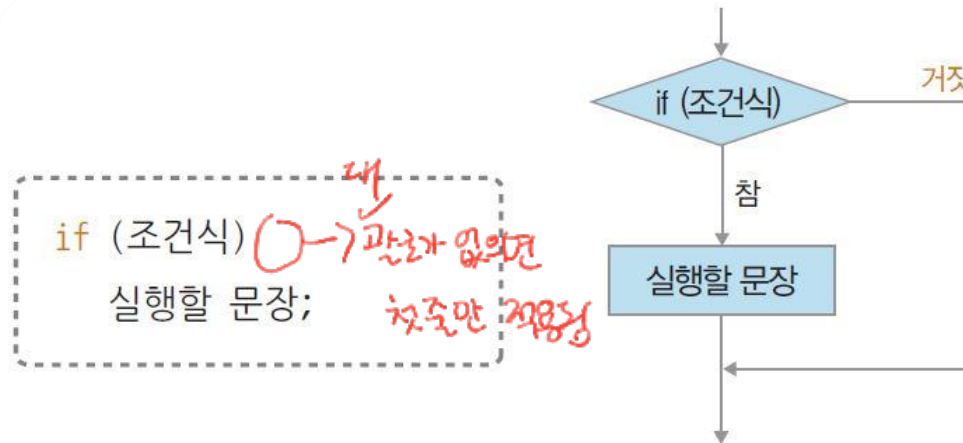


그림 5-1 if문의 형식과 순서도

그림 5-1 if문의 형식과 순서도

# 플로우차트 상에서의 조건문

## if문의 표기법

- ◇ : 분기문, ◇의 꼭지점에서 이어지는 화살표 - 흐름
- ◇ 안의 내용 : 조건 - 참 또는 거짓으로 나타낼 수 있는 물음
- 화살표 위의 문구: 조건식의 결과

## 오른쪽 그림을 if문을 사용하여 C 언어의 코드로 나타내어 보자!

```
01
02
03
04
05
06
07
08
09
10
11
12 }
```

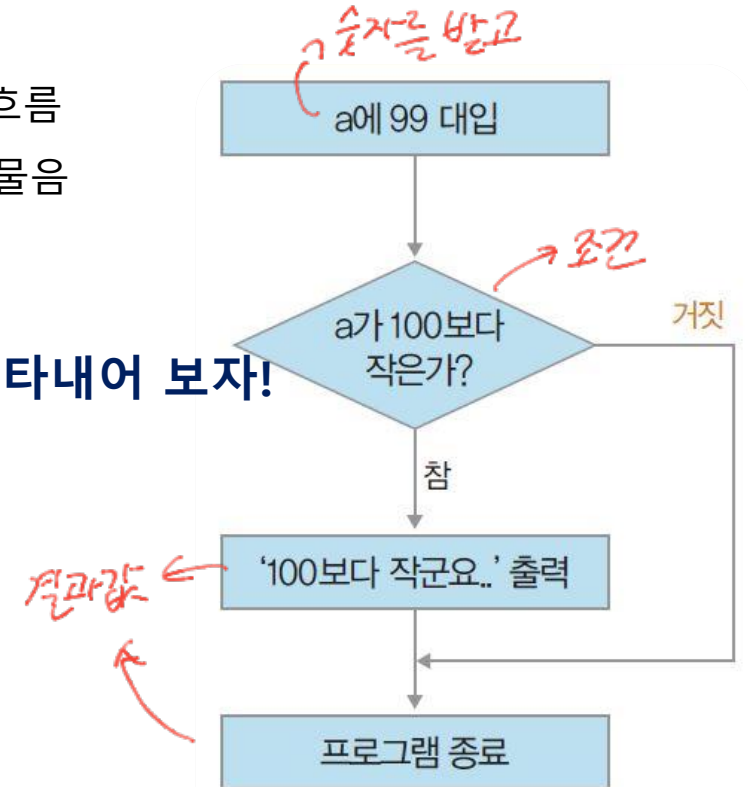


그림 5-2 [기본 5-1]의 실행 과정  
그림 2-5 [기초 2-1]의 실행 과정

프로그램 실행

## 1 if Statements

### if 문의 문법

#### ● 행 바꿈의 함정

- 조건식이 참이면 바로 아래에 있는 한 문장만 실행  
(7행의 조건식이 거짓이므로 그 아래 문장인 8행은 건너뛰고 9행부터 실행)
- 7~11행의 수정

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a = 200;
06
07     if (a < 100)
08         printf("100보다 작군요..\n");
09     printf("거짓이므로 이 문장은 안보이겠죠?\n");
10     printf("프로그램 끝! \n");
11 }
12 }
```

*Handwritten note: 7~11행 수정 (7~11 lines modified)*



```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a = 200;
06
07     if (a < 100)
08     {
09         printf("100보다 작군요..\n");
10         printf("거짓이므로 앞의 문장은 안보이겠죠?\n");
11     }
12
13     printf("프로그램 끝! \n");
14 }
```

*Handwritten note: 중괄호 (Curly braces) added around lines 8-11*

#### ● if문에서 두 문장 이상을 실행하게 할 경우

- 중괄호 ({ }) 로 묶어 블록으로 지정한다.

## 1 if Statements

### if 문의 코딩 규칙 #1

- 본 강의에서는 google 스타일 코딩 규칙을 지속적으로 강조

- 실험 실습 시 본 규칙을 준수하여 제출하지 않을 시 감점의 대상이 됨

- 규칙 #1 – Teams의 Wiki 탭 참고

- 수행할 명령이 하나이더라도 반드시 중괄호로 묶어 블록 처리할 것
- "if"와 '(' 사이에 공백을 둘 것
- 조건식의 괄호 '(', ')' 안에 불필요한 공백을 두지 말 것
- 조건식의 괄호 ')' 뒤에 공백을 사용하고 줄바꿈을 하지 않은 상태에서 중괄호를 열 것

```
if(condition) { ... // Bad - space missing after IF
if ( condition ) { ... // Bad - space between the parentheses and the condition
if (condition){ ... // Bad - space missing before {
if(condition){ ... // Doubly bad
if (int a = f();a == 10) { ... // Bad - space missing after the semicolon
if!(condition)!{ // Good
```

조건식에  
조건문  
등

## 1 if Statements

### if-else문

#### ● if의 단독 사용과의 차이

- 단독 사용 시: 조건을 만족할 때 수행하는 명령만 있음
- else와 짝을 이뤄 사용 시: 조건을 만족하지 않을 때 수행하는 명령도 있음
- 참일 때 실행하는 문장과 거짓일 때 실행하는 문장이 다를 때 사용함.
- 조건식이 참이면 '실행할 문장 1'을 실행하고, 그렇지 않으면 '실행할 문장 2'를 실행.

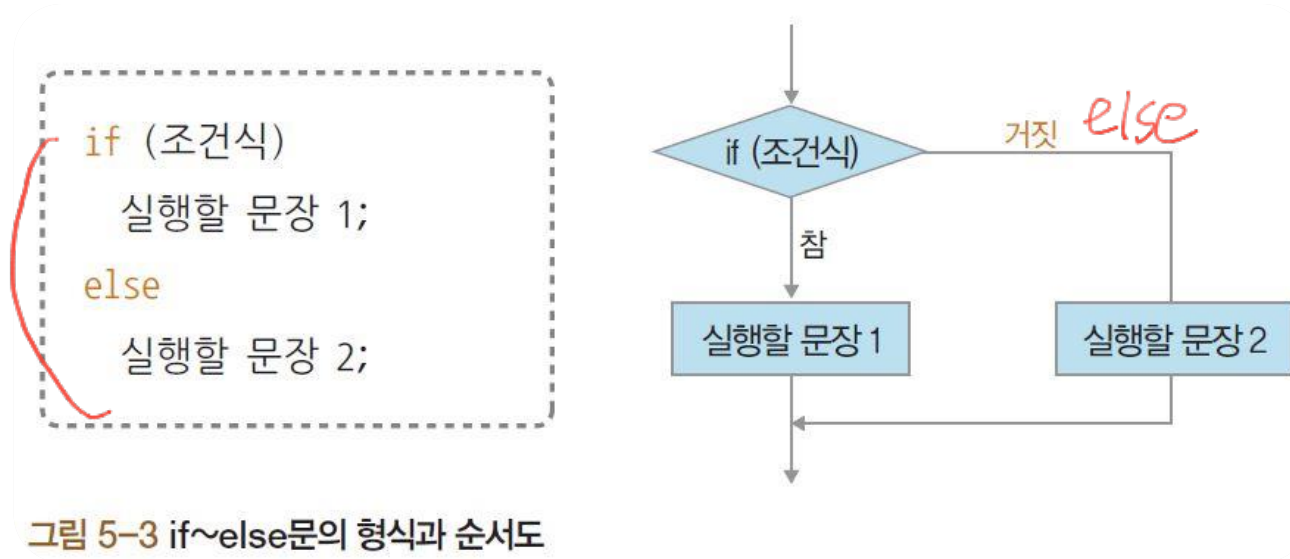


그림 5-3 if~else문의 형식과 순서도

그림 2-3 if~else문의 형식과 순서도

## 1 if Statements

### if-else 문의 코딩 규칙 #2

#### ● 규칙 #2 – Teams의 Wiki 탭 참고

- **if**와 짝을 이루는 **else**문은 구분을 위해 중괄호를 닫은 '}' 후에 줄바꿈을 하지 않는다
- **else** 앞 뒤에 공백을 둔다

```
// Bad – IF statement with ELSE is missing braces
if (x) DoThis();
else DoThat();

// Bad – IF statement with ELSE does not have braces everywhere
if (condition)
    foo;
else {
    bar;
}

// GOOD – IF statement with ELSE has braces everywhere
if (condition){           // Comments on condition must be here!
    foo;
} else {
    bar;
}
```

- 조건에 관한 주석은 '{'의 오른쪽에 기입한다.

## 1 if Statements if-else문

### ● 오른쪽 그림을 if-else문을 사용하여 C 언어의 코드로 나타내어 보자!

- 코딩 규칙 준수

```
01
02
03
04
05
06
07
08
09
10
11
```

```
if (num < 100) {
    printf("
}
else {
    printf("
}
```

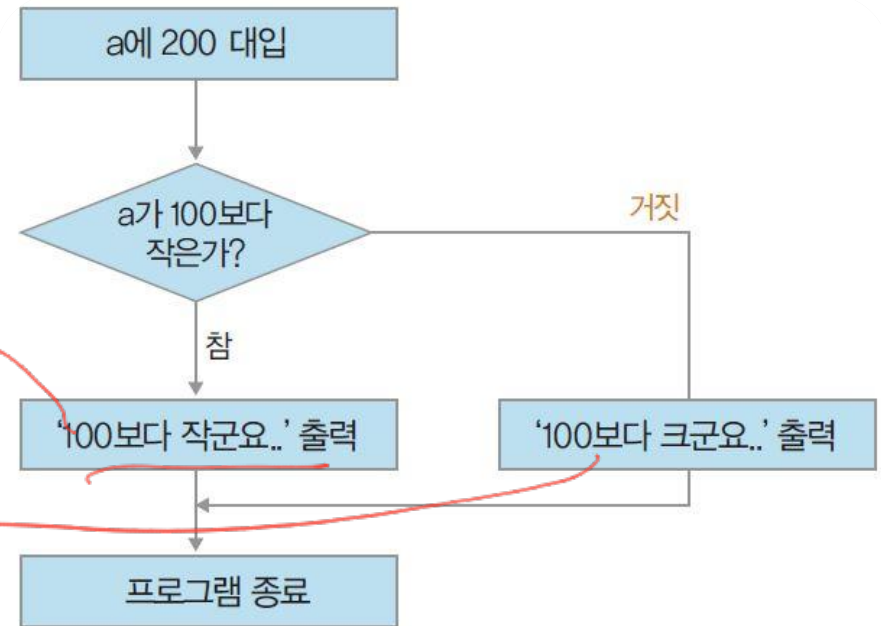


그림 5-4 [기본 5-4]의 실행 과정

그림 2-4 [이동 2-4]이 적용 됨

프로그램 실행



## 2 Multiple Branches with if Statements

### if문을 이용한 다중 분기문

#### ● 중첩 (Nested) if문 = 상황연산자

- if문을 한 번 실행하고 그 결과에 다시 다른 if문을 실행하는 것
- 가능하면 블록을 사용하여 조건식을 명확히 하는 것이 좋음

```

if (조건식 1){
    if (조건식 2)
        실행할 문장 1;
    else
        실행할 문장 2;
}
else
    실행할 문장 3;
    
```

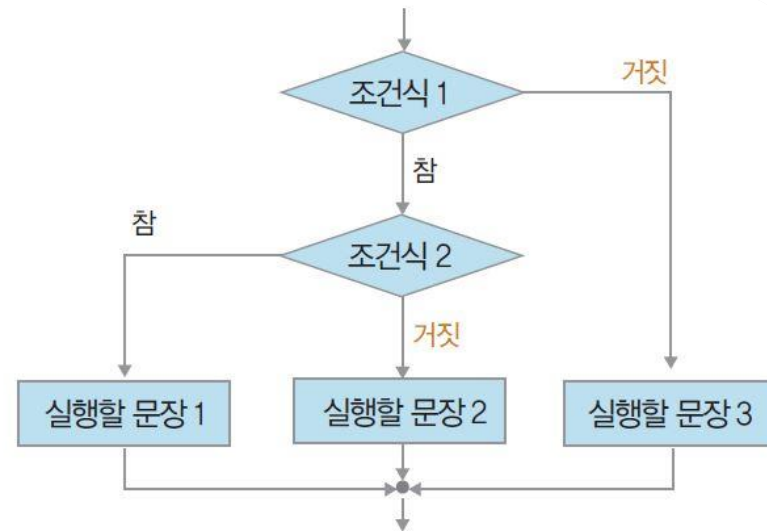


그림 5-5 중첩 if문의 형식과 순서도

그림 5-5 중첩 if문의 형식과 순서도

그림 5-5 중첩 if문의 형식과 순서도

## 2 Multiple Branches with if Statements

### if문을 이용한 다중 분기문

- 오른쪽 그림을 if-else문을 사용하여 C 언어의 코드로 나타내어 보자!

```

01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24 }
    
```

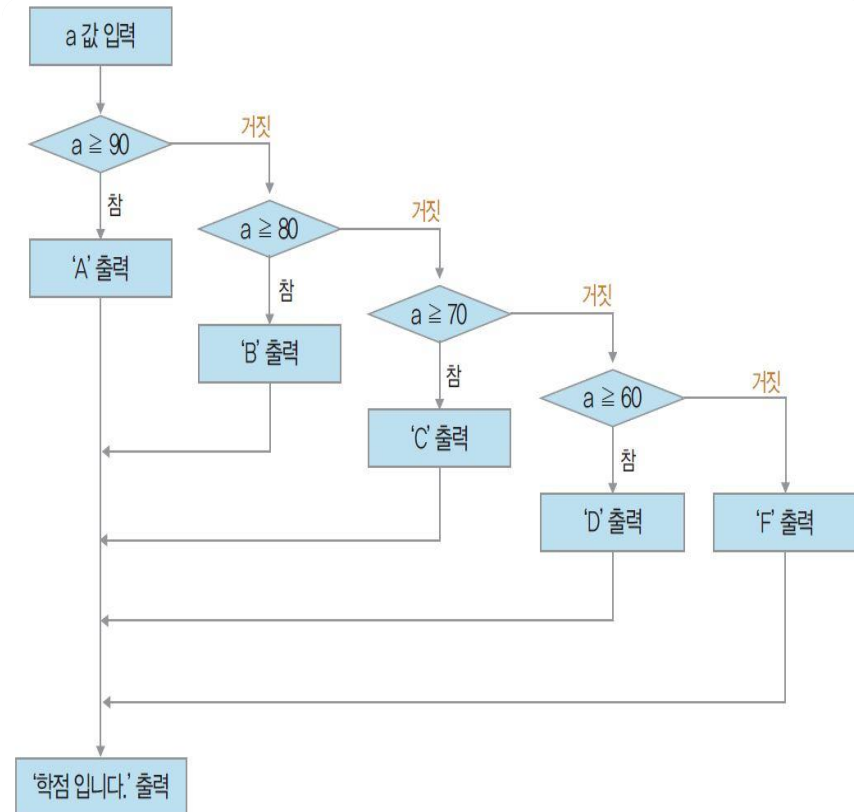


그림 5-6 [응용 5-8]의 실행 과정

그림 5-7 [응용 5-8]의 실행 과정

학점입니다.' 출력

## 2 Multiple Branches with if Statements

### if-else문을 이용한 다중 분기문

#### ● if-else 문 설계 상의 주의점

- 깊이는 최대한 얇게: 지나치게 깊으면 지옥간다.
- 조건의 수는 최대한 적게: 공통 규칙을 잘 관찰
- 하나의 조건이 너무 길지 않게: 줄바꿈 되지 않도록

```
if(condition){
    foo;
} else {
    if ... {
        ...
    } else {
        if .... {
            ...
        } else {
            if ... {
                ... → // hell
            } else {
                ...
            }
        }
    }
}
```

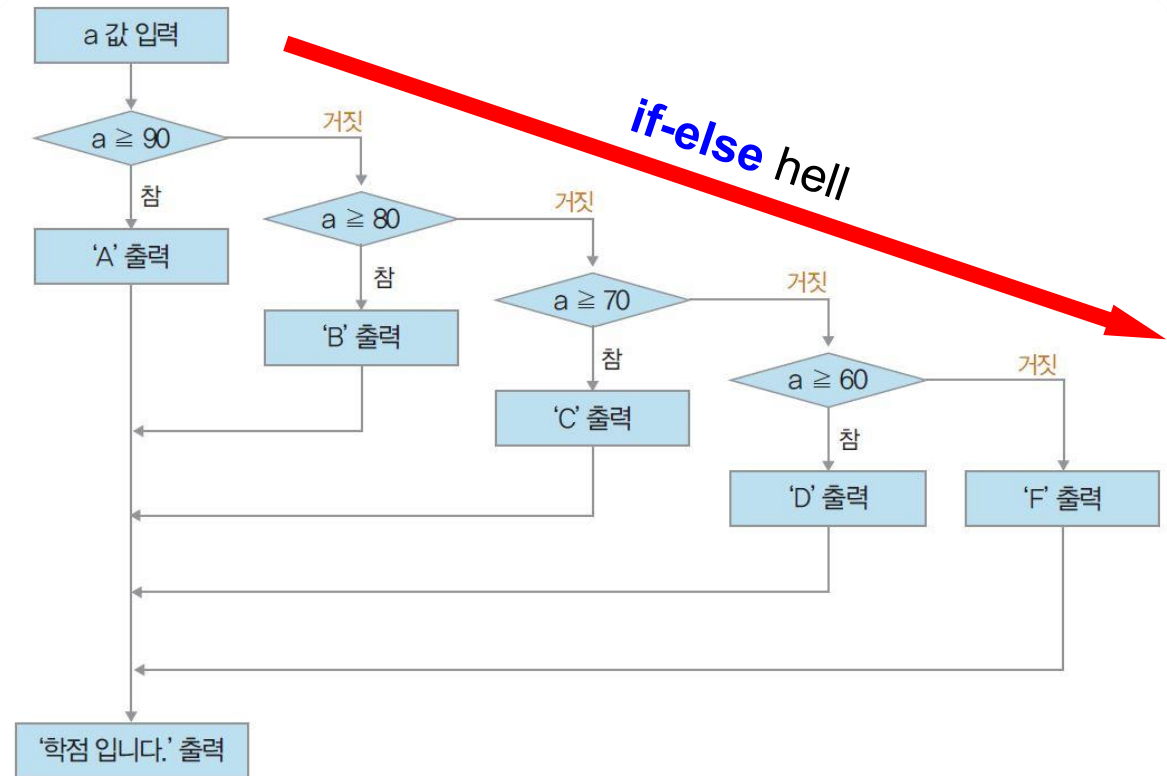


그림 5-6 [응용 5-8]의 실행 과정

그림 2-9 [응용 2-8]의 실행 과정

## 2 Multiple Branches with if Statements

### if-elseif-else문을 이용한 다중 분기문

#### ● if-else hell에 빠지지 않기 위한 최소한의 방어 장치

- **if-else** hell을 굳이 의역하자면, if-else의 높
- **else** 뒤에 "{"을 열면 **else** 블록의 시작이지만 **if**를 사용하면 추가 조건을 제시할 수 있다.
- **의미**: condition1은 **FALSE**이지만, condition2는 **TRUE**일 때

```
if(condition1){
    foo;
} else {
    if (condition2) ... {
        ...
    } else {
        if (condition3) ... {
            ...
        } else {
            if (condition4) ... {
                ... ➔ // hell
            } else {
                ...
            }
        }
    }
}
```



```
if(condition1){
    foo;
} else if (condition2) ... {
    ...
} else if (condition3) ... {
    ...
} else if (condition4) ... {
    ... ➔ // by hell
} else {
    ...
}
```

## 2 Multiple Branches with if Statements

### if-else 문의 코딩 규칙 #3

- 규칙 #3: 코딩 규칙 #2의 확장 – Teams의 Wiki 탭 참고

- if와 짝을 이루는 else if문은 구분을 위해 중괄호를 닫은 '}' 후에 줄바꿈을 하지 않는다
- else 앞 뒤에 공백을 둔다

```
// F*** BAD – too many useless new lines
if(condition1)
{
    // no spaces inside parentheses, space before brace
    DoOneThing();           // two/four space indent
    DoAnotherThing();
}
else if(condition2)
{
    // doesn't satisfy condition1, but satisfy condition2
    DoAThirdThing(a);
}
else
{
    DoNothing();
}

// GOOD – IF statement with ELSE IF and ELSE has braces everywhere
if(condition1){           // no spaces inside parentheses, space before brace
    DoOneThing();           // two/four space indent
    DoAnotherThing();
} else if(condition2) {    // doesn't satisfy condition1, but satisfy condition2
    DoAThirdThing(a);
} else {
    DoNothing();
}
```

## 4 Switch Statements

### switch-case문

#### ● 분기문 이면서 점프문 (jump)

- **TRUE**와 **FALSE** 이외의 다른 여러 개의 선택이 가능한 경우에 사용
- 다중 분기: 여러 조건 중 하나를 만족하는 블록을 수행
- 조건: 정수로 표현될 수 있는 모든 값 - 에 따라 **case**문 실행

#### ● 엄밀히 따지면 아래 플로우 차트는 if-elseif-else를 나타냄

```
switch(정숫값){
  case 정숫값 1:
    실행할 문장 1;
    break;
  case 정숫값 2:
    실행할 문장 2;
    break;
  default: → else 랑 같음
    실행할 문장 3;
    break;
}
```

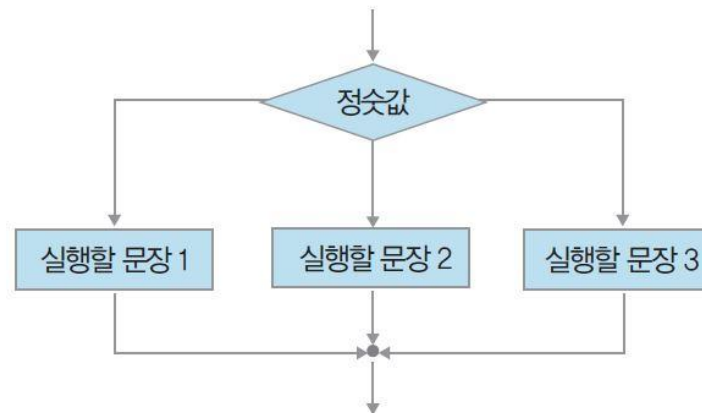


그림 5-7 switch~case문의 형식과 순서도

그림 5-8 switch~case문의 유효한 문법

## 4 Switch Statements

# 플로우 차트로 이해하는 switch-case문

- 정해진 위치를 찾아서 jump한다.
  - 플로우 차트를 아래 공백에 새로 그려보자.

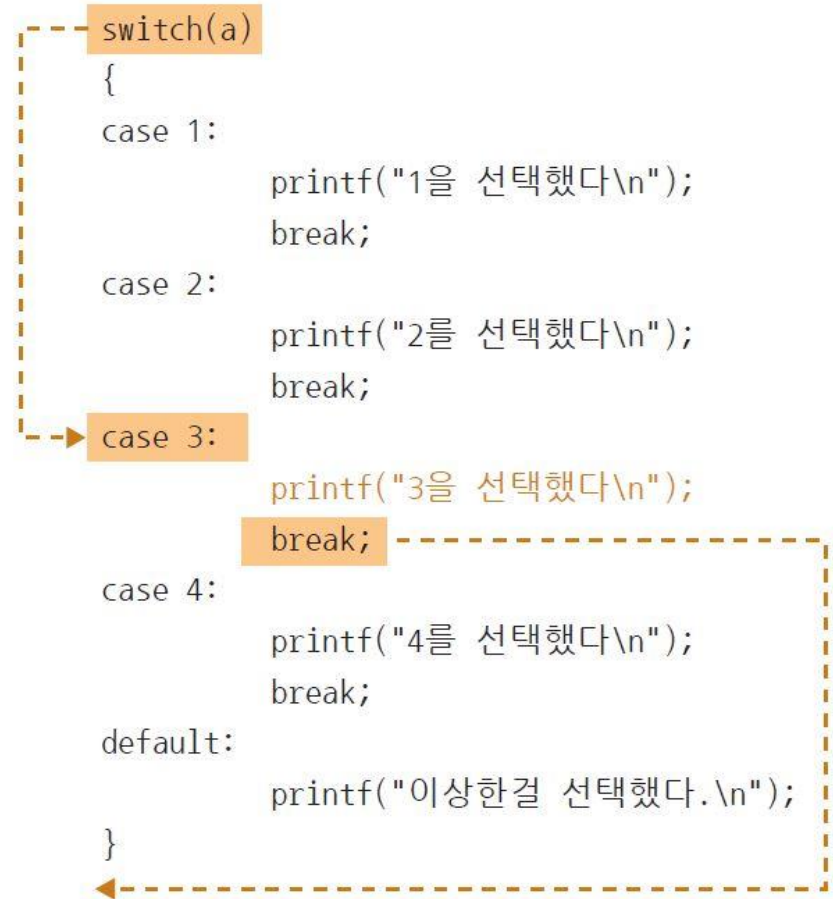


그림 5-8 a가 3일 때의 switch~case문 흐름도

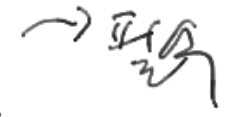
# switch-case-default문

### ● default

- 디폴트: "기본 값"의 의미로 사용하는 용어
- else의 역할과 유사: switch의 조건이 모든 case를 만족하지 않을 때 이동하는 곳
- 일반적으로 모든 케이스를 나열하고 맨 마지막에 사용한다.
- 단, 일반적일 뿐 맨 앞에 사용할 수도 있다. ➔ 엔지니어의 역량과 재치에 따라 결정된다.



### 제어문 break;

- **break;** 
  - 깨다, 중단시키다, 벗어나다
  - Prison Break: 감옥 깨기? ← 도장깨기?
- **Switch~case문의 블록을 벗어나가는 역할**
  - 실행문의 마지막에 반드시 써줘야 함
- **break문을 빼고 실행한 결과는 어떻게 될까?**

# switch-case-default 문의 코딩 규칙

- switch와 '(' 사이, ')'와 '{' 사이에 공백을 둔다.
- switch-case-default문에서 case가 변경될 때마다 반드시 행을 바꾼다.
- switch-case-default문에서 마지막 조건은 반드시 break를 사용한다.

// GOOD

```
switch(x%2){
    case 0:
    case 1:
    case 2:
        .... do something ...
    default:
        break;
}
```

// GOOD

```
switch(x%2){
    case 1:
    case 2:
        .... do something ...
    default:
        break;
    case 0:
        break;
}
```

// ALSO GOOD

```
switch(x%2){
    case 0:
    case 1:
    case 2:
        .... do something ...
    default:
        break;
}
```

- switch와 case는 들여쓰기 수준을 동일하게 적용할 수 있다.

# Branching Statements

## ● 2가지 형태의 분기문

- 조건문이라고도 한다.
- TRUE와 FALSE로 표현할 수 있는 조건식을 사용할 때 → if-elseif-else
- 정수로 표현할 수 있는 조건식을 사용할 때 → switch-case-default

## ● 코딩 규칙 (공통) – google St. *각각 방식*

- 조건문에 해당하는 괄호 (Round Bracket)는 좌우에 공백을 사용한다.
- 블록에 해당하는 괄호 (Curly Bracket)는 열 때 줄바꿈을 하지 않는다.