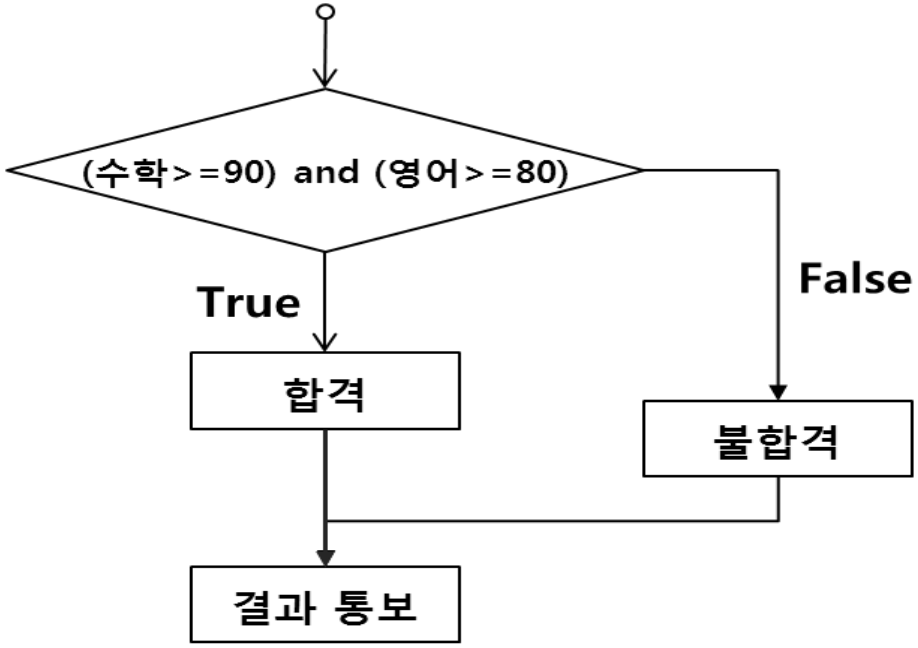
중첩 조건문
(Nested-if 문)

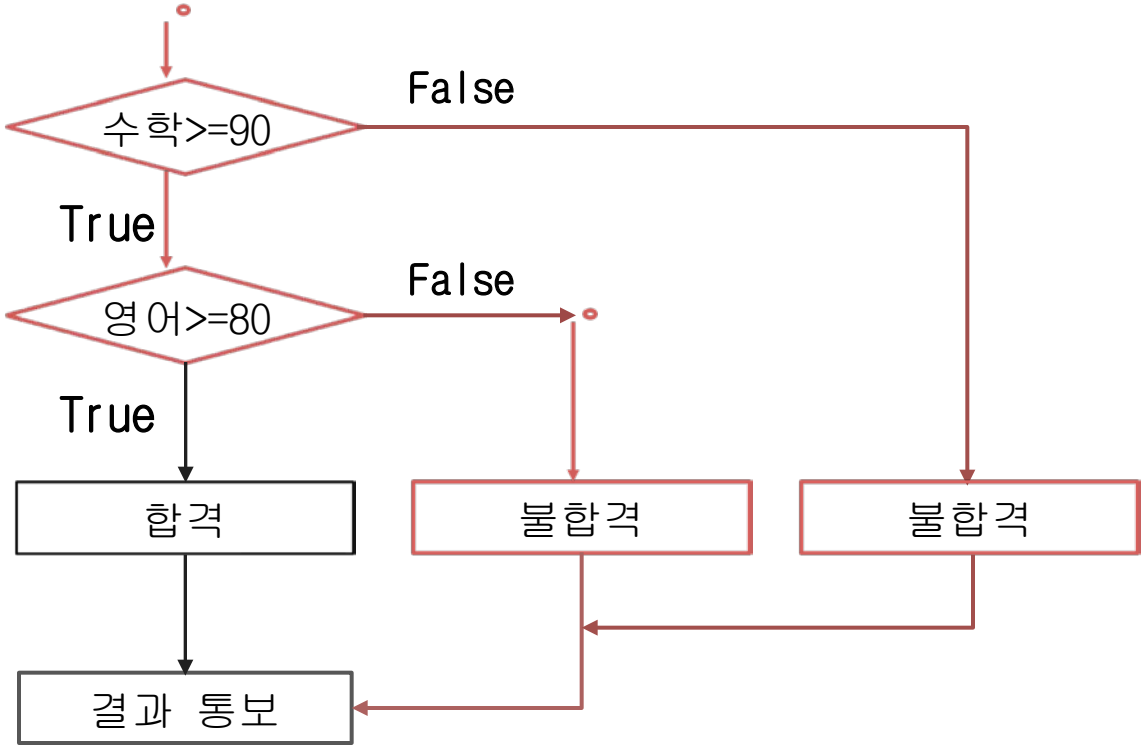
알고리즘 예제

- 수학점수가 90점 이상, 영어점수가 80점 이상이어야 합격일 때
  - *if* 문을 이용한 경우 문제점: 합격이면 상관없지만, 불합격일 경우 어떤 과목에 의해서 불합격 되었는지는 알 수 없음



보완된 알고리즘 예제

- 중첩 조건문을 이용하여 좀더 복잡한 상황의 알고리즘을 구현 가능

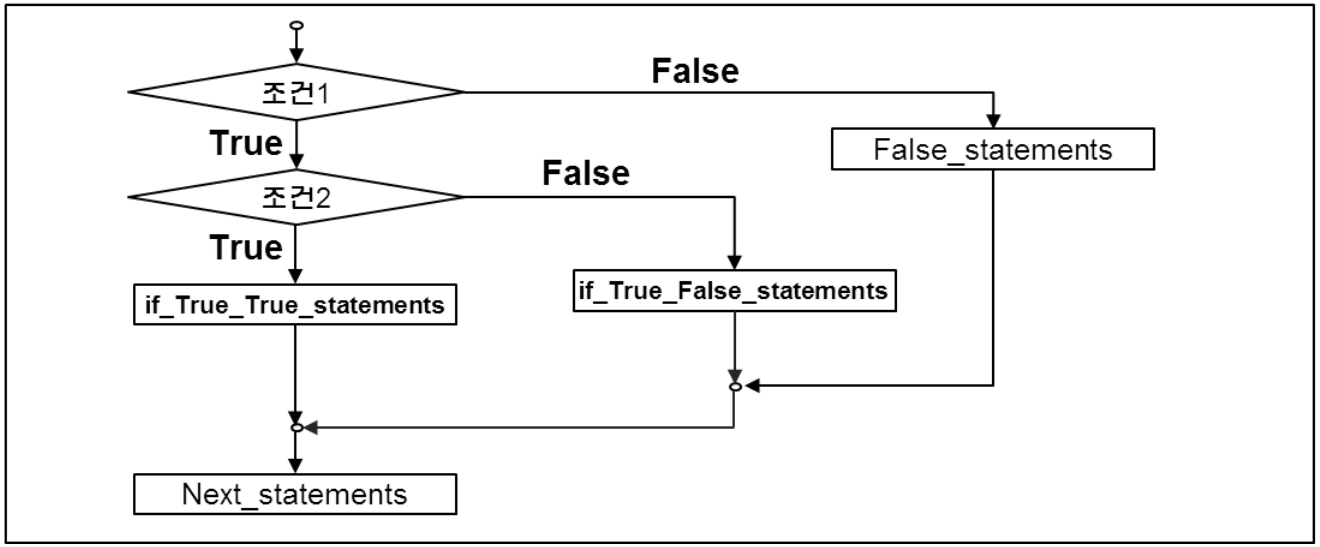


Nested-if 문 (1)

- if 문의 조건식이 True일 때 또 다른 if 문이 들어가는 경우

```
if 조건1 :
    if 조건2 :
        if_True_True_statements
    else :
        if_True_False_statements
else :
    False_statements
Next_statements
```

"조건1"이 True 이고 "조건2"도 True 이면 if_True_True_statements를,
"조건1"이 True 이고 "조건2"가 False 이면 if_True_False_statements를 실행하고,
"조건1"이 False 이면 False_statements 를 실행한 후
Next_statements를 실행

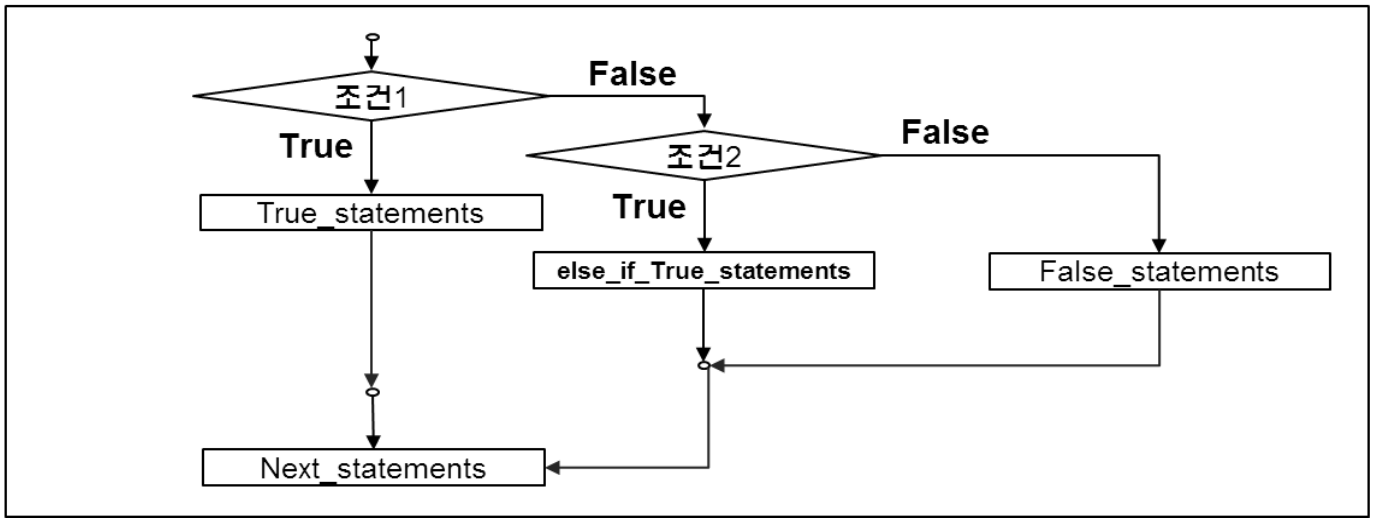


Nested-if 문 (2)

- if 문의 조건식이 False일 때 else 뒤에 또 다른 if 문이 들어가는 경우

```
if 조건1 :
    True_statements
else :
    if 조건2 :
        else_if_True_statements
    else :
        False_statements
Next_statements
```

"조건1"이 True 이면 True_statements를 실행한 후 Next_statements를 실행
"조건1"이 False 이고 "조건2"가 True 이면 else_if_True_statements를,
"조건1"이 False 이고 "조건2"도 False 이면 False_statements를 실행한 후,
Next_statements를 실행



Nested-if 문 (3)

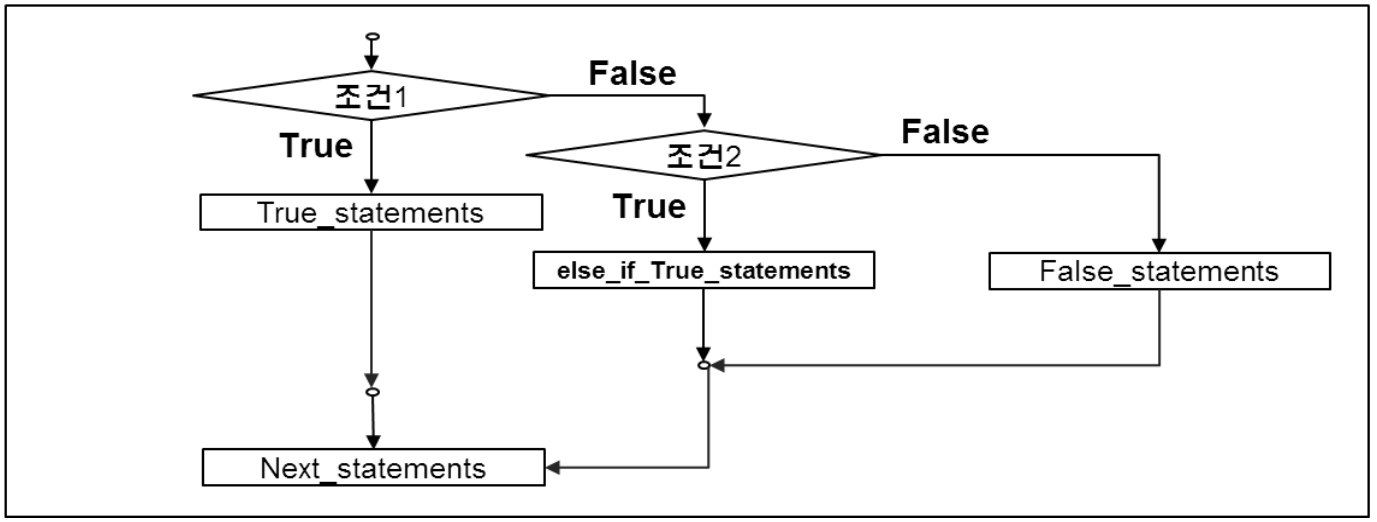
- if 문의 조건식이 False일 때 else 뒤에 또 다른 if 문이 들어가는 경우 elif로 대체가능
  - 실행 시간이 길어진다는 단점이 있지만 가독성이 좋아지는 장점

```
if 조건1 :
    True_statements_a
elif 조건2 :
    True_statements_b
else :
    False_statements
Next_statements
```

"조건1"이 True 이면 True_statements_a를 실행한 후, Next_statements를 실행

"조건1"이 False 이고 "조건2"가 True 이면 True_statements_b를 실행,

"조건1"이 False 이고 "조건2"도 False 이면 False_statements를 실행한 후 Next_statements를 실행



비교 예제

- 1~8까지 입력 받은 숫자를 판별하는 프로그램

```
num = int(input('숫자:'))
if num < 5 :
  if num < 3 :
    if num < 2 :
        print('1')
    else:
        print('2')
  else :
    if  num < 4 :
        print('3')
    else :
        print('4')

else :
  if num < 7 :
    if num < 6 :
        print('5')
    else :
        print('6')
  else :
    if num < 8 :
        print('7')
    else :
        print('8')
```

3번의 비교연산

조건의 수가 많아질 경우
프로그램 실행 시간이
→ 짧아진다

But 가독성이 낮다

```
num = int(input('숫자:'))

if num == 1 :
    print('1')

elif num == 2 :
    print('2')

elif num == 3 :
    print('3')

elif num == 4 :
    print('4')

elif num == 5 :
    print('5')

elif num == 6 :
    print('6')

elif num == 7 :
    print('7')

elif num == 8 :
    print('8')
```

8번의 비교연산

조건의 수가 많아질 경우
프로그램 실행시간이
→ 길어진다

But 가독성이 높다

실습 예제들 (1)

- *예제1)* 수학점수 및 영어점수로 두 가지 유형으로 판단:
  "합격", "불합격"
  - '복합'조건식에 논리연산자 'and' 사용하여 단일if로 구현
  - '복합'조건식에 논리연산자 'or' 사용하여 단일if로 구현

- *예제2)* 수학점수 및 영어점수로 세 가지 유형으로 판단:
  "합격", "수학 불합격", "영어 불합격"
  - '단일'조건식을 사용하여 if에 중첩
  - '단일'조건식을 사용하여 else에 중첩
    - elif 사용
  - '복합'조건식에 논리연산자 'and' 사용하여 else에 중첩
    - elif 사용
  - '복합'조건식에 논리연산자 'or' 사용하여 if에 중첩

실습 예제들 (2)

- *예제3)* 수학점수 및 영어점수로 네 가지 유형으로 판단:
"합격", "수학 불합격", "영어 불합격", "수학 영어 모두 불합격"
  - '복합'조건식에 논리연산자 'and' 사용하여 else에 두번 중첩
    - elif 사용
  - '복합'조건식에 논리연산자 'or' 사용하여 if에 두번 중첩
    - elif 사용

- *예제4)* 윤년(leap year)을 판단하는 프로그램을 작성:
해당 연도가 4로 나누어 떨어지고, 100으로 나누어 떨어지지 않으면 윤년
하지만 해당 연도가 400으로 나누어 떨어지면 윤년
  - 단일if로 구현
  - 중첩if로 구현