



문공 A0015

# R 프로그래밍

김 태 완

[kimtwan21@dongduk.ac.kr](mailto:kimtwan21@dongduk.ac.kr)

## 조건문, 반복문

---

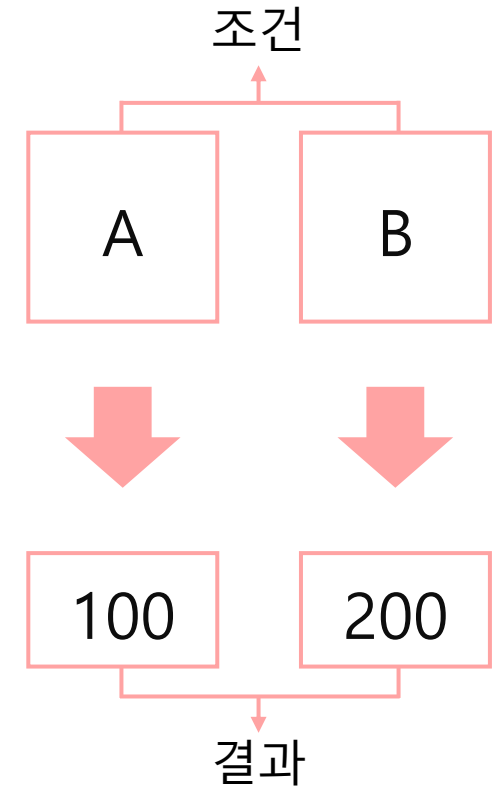
- 조건문
  - if-else문
    - if-else문의 문법

```
if(비교조건) {  
    조건이 참일 때 실행할 명령문(들)  
}else{  
    조건이 거짓을 때 실행할 명령문(들)  
}
```

## 조건문, 반복문

- 조건문
  - if-else문
    - 기본 if-else문

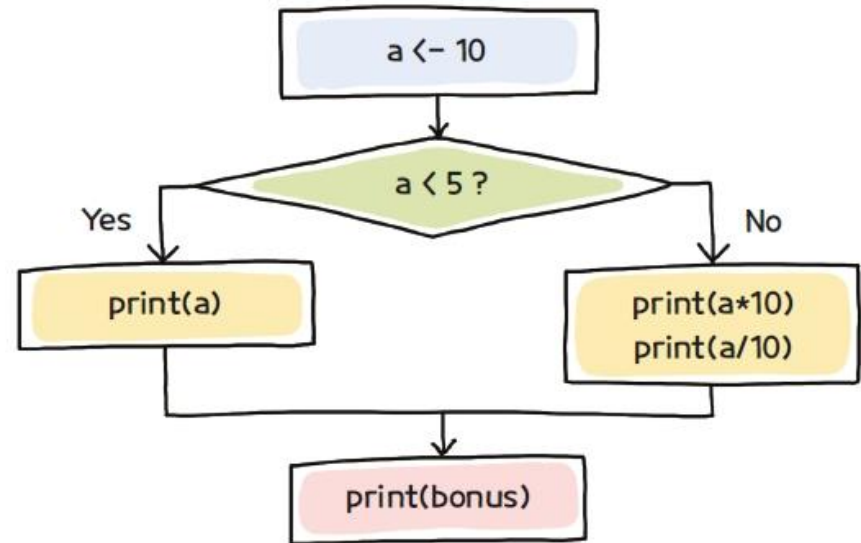
```
job.type <- 'A'
if (job.type == 'B') {
    bonus <- 200      # 직군이 B일 때 실행
} else {
    bonus <- 100      # 직군이 B가 아닌 나머지 경우 실행
}
print(bonus)
```



## 조건문, 반복문

- 조건문
  - if-else문
    - 기본 if-else문

```
> a <- 10  
  
> if (a<5) {  
+   print(a)  
+ } else {  
+   print(a*10)  
+   print(a/10)  
+ }
```



## 조건문, 반복문

- 조건문
  - if-else문
    - else가 생략된 if문

```
> job.type <- 'B'
```

```
> bonus <- 100
```

```
> if(job.type == 'A'){
```

```
+   bonus <- 200
```

```
+ }
```

```
> print(bonus)
```

```
[1] 100
```

# 직무 유형이 A일 때 실행

- 코드 블록(code block)

if와 else 다음에 있는 중괄호{ }

여러 명령문을 하나로 묶어주는 역할

코드 블록에 의해 묶인 명령문은 무조건 함께 실행됨.

# 직무유형이 A가 아니므로

bonus 값을 200으로 변경하지 않음

(조건문의 명령을 실행하지 않음)

조건

A

200

결과

## 조건문, 반복문

- 조건문
  - if-else문
    - else가 생략된 if문

```
# -----  
a <- 10  
b <- 20  
if (a>5 & b>5) {           # and  
  print(a+b)  
}  
if (a>5 | b>30) {          # or  
  print(a*b)  
}
```

## 조건문, 반복문

- 조건문
  - if-else문

if-else문을 서술할 때 다음과 같이 쓰면 오류가 발생합니다.

```
job.type <- 'A'
if (job.type == 'B') {
  bonus <- 200
}
else {                # 에러 발생, 윗줄로 옮겨야 한다
  bonus <- 100
}
```

else는 반드시 if문의 코드블록이 끝나는 표시 }와 같은 줄에 서술해야 합니다.

```
if (job.type == 'B') {
  bonus <- 200
}
```

## 조건문, 반복문

- 조건문

- ifelse문 : 조건에 따라 두 값 중 하나를 선택하는 경우에는 ifelse문이 더욱 편리

if(비교조건, 조건이 참일 때 선택할 값, 조건이 거짓일 때 선택할 값)

### if-else문

```
> a <-10
> b <- 20
> if(a > b){
+   c <- a
+ } else {
+   c <- b
+ }
> print(c)
[1] 20
```

### ifelse문

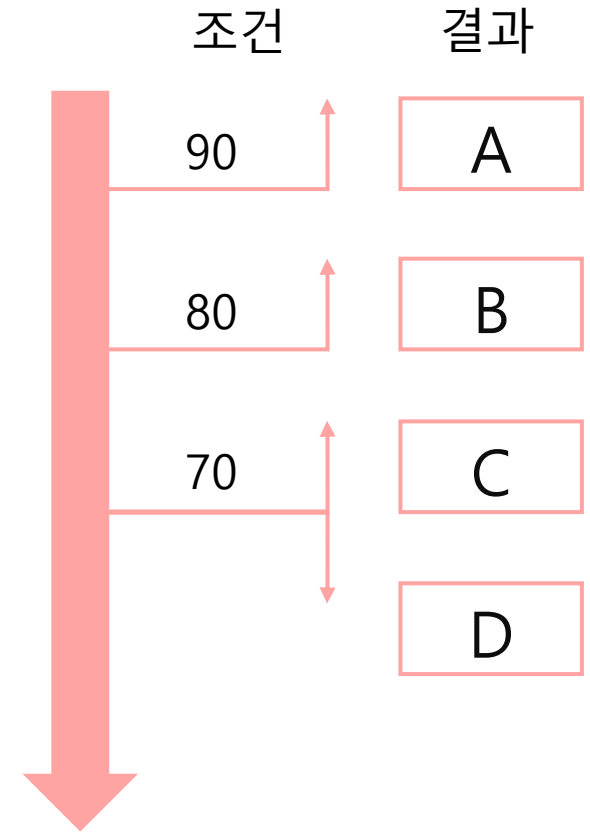
```
> a <-10
> b <- 20
> c <- ifelse(a>b, a, b)
> print(c)
[1] 20
# 비교 조건 : a > b
  조건이 참일 때 : c <- a
  조건이 거짓일 때 : c <- b
```



## 조건문, 반복문

- 조건문
  - if-else문
    - 다중 if-else문

```
> score <- 85
> if(score > 90){          # score가 90보다 크면 grade는 A
+   grade <- 'A'
+ } else if (score > 80){  # score가 80보다 크면 grade는 B
+   grade <- 'B'
+ } else if (score > 70){  # score가 70보다 크면 grade는 C
+   grade <- 'C'
+ } else {                # score가 70이하면 grade는 D
+   grade <- 'D'
+ }
> print(grade)
[1] "B"
```



## 조건문, 반복문

- 반복문

- for문 : 정해진 반복 횟수만큼 실행

```
for(반복 변수 in 반복 범위) {  
    반복할 명령문(들)  
}
```

```
> for(i in 1:4) {  
+   print('O')  
+ }
```

```
[1] O  
[1] O  
[1] O  
[1] O
```

# 반복 변수 : i, 반복 범위 : 1:4 -> 4번 반복  
반복할 명령 : 'O'를 출력

```
i = 1  
i = 2  
i = 3  
i = 4
```

## 조건문, 반복문

- 반복문

- for문 : 정해진 반복 횟수만큼 실행
  - 반복 범위에 따른 반복 변수의 값 변화

```
> for(i in 6: 10) {  
+   print(i)  
+ }
```

```
[1] 6  
[1] 7  
[1] 8  
[1] 9  
[1] 10
```

# 반복 변수 : i, 반복 범위 : 6:10 -> 6에서 10까지 **5번 반복**  
(반복 범위의 숫자는 반드시 1부터 시작하지 않아도 됨)  
반복할 명령 : i를 출력

## 조건문, 반복문

- 반복문

- for문 : 정해진 반복 횟수만큼 실행
  - 반복 변수를 이용한 구구단 출력

```
> for(i in 1:9) { 단순 문자열(i의 값 변경과 상관 없음) -> 반복이 진행되어도 값이 바뀌지 않음
+   cat('2*', i, '=', 2*i, '\n') # cat( ) 함수 : 한 줄에 여러 개의 값을 결합하여 출력
+ }   반복이 진행되면서 i의 값이 변경 -> i, 2*i의 값도 변경      ('2*', i, '=', 2*i, '\n'을 결합하여 출력)
                                     <-> print( ) 함수 : 하나의 값을 출력
                                     # '\n'은 줄 바꿈([Enter])을 하도록 하는 특수문자
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
...
```

## 조건문, 반복문

- 반복문

- for문 : 정해진 반복 횟수만큼 실행
  - for문 안에서 if문의 사용(반복문 + 조건문)

```
> for(i in 1:10) {  
+   if(i%%2==0) {  
+     print(i)  
+   }  
+ }
```

```
[1] 2  
[1] 4  
[1] 6  
[1] 8  
[1] 10
```

```
# 반복 변수 : i, 반복 범위 : 1:10 -> 10번 반복  
# 반복할 명령 -> 조건문  
# 조건 : i를 2로 나누었을 때 나머지가 0 (짝수)  
# 조건이 참일 때 -> i를 출력
```

## 조건문, 반복문

---

- 반복문
  - for문 : 정해진 반복 횟수만큼 실행
    - for문 안에서 if문의 사용(반복문 + 조건문)

```
for(i in 1:20) {  
  if(i%%2==0) {           # 짝수인지 확인  
    cat(i, ' ')  
  }  
}
```

```
[1] 2  4  6  8 10 12 14 16 18 20
```

## 조건문, 반복문

---

- 반복문
  - for문 : 정해진 반복 횟수만큼 실행

```
sum <- 0
for(i in 1:100) {
  sum <- sum + i      # sum에 i 값을 누적
}
print(sum)

[1] 5050
```

## 조건문, 반복문

- 반복문

- for문 : 정해진 반복 횟수만큼 실행

- 데이터프레임에서 for문 적용하기 : iris에서 꽃잎의 길이에 따른 분류 작업

<b>norow &lt;- nrow(iris)</b>	# iris의 행의 수를 norow 변수에 저장
<b>mylabel &lt;- c( )</b>	# 비어있는 벡터 선언(결과 값이 벡터이므로)
<b>for(i in 1:norow) {</b>	# 반복 변수 : i, 반복 범위 : 1:norow -> 1행부터 마지막행까지
<b>if (iris\$Petal.Length[i] &lt;= 1.6) {</b>	# 조건 : iris의 Petal.Length열의 i행의 값이 1.6 이하
<b>mylabel[i] &lt;- 'L'</b>	# 조건이 참일 때 : mylabel 벡터의 i번째 값에 'L'추가
<b>} else if (iris\$Petal.Length[i] &gt;= 5.1) {</b>	# 조건 : iris의 Petal.Length열의 i행의 값이 5.1 이상
<b>mylabel[i] &lt;- 'H'</b>	# 조건이 참일 때 : mylabel 벡터의 i번째 값에 'H'추가
<b>} else {</b>	# iris의 Petal.Length열의 i행의 값이 1.6보다 크고 5.1보다 작음
<b>mylabel[i] &lt;- 'M'</b>	# mylabel 벡터의 i번째 값에 'M'추가
<b>}</b>	
<b>}</b>	

조건문

반복문\_for문

반복문\_while문



## 조건문, 반복문

- 반복문

- for문 : 정해진 반복 횟수만큼 실행
  - 데이터프레임에서 for문 적용하기 : iris에서 꽃잎의 길이에 따른 분류 작업

```
> print(mylabel)
```

```
[1] "L" "L" "L" "L" "L" "M" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "M"
```

```
[20] "L" "M" "L" "L" "M" "M" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L"
```

```
[39] "L" "L" "L" "L" "L" "L" "M" "L" "L" "L" "L" "L" "M" "M" "M" "M" "M" "M" "M"
```

```
...
```

```
# Petal.Length의 값이 'L', 'M', 'H' 중의 하나로 결정
```

```
# 벡터 값의 개수는 iris의 행의 값인 150개(nrow개)
```

## 조건문, 반복문

- 반복문

- for문 : 정해진 반복 횟수만큼 실행
  - 데이터프레임에서 for문 적용하기 : 새로 데이터프레임 생성

```
newds <- data.frame(iris$Petal.Length, mylabel) # 꽃잎 길이와 레이블 결합
head(newds)                                     # 새로운 데이터셋 내용 출력
```

```
iris.Petal.Length mylabel
```

1	1.4	L
2	1.4	L
3	1.3	L
4	1.5	L
5	1.4	L
6	1.7	M

## 조건문, 반복문

- 반복문

- while문 : 특정 조건이 만족되는 동안 실행 (조건이 거짓일 경우 반복을 종료)

```
while(비교조건) {  
    반복할 명령문(들)  
}
```

while문은 for문과 달리 몇 번이나 반복이 실행될지 쉽게 알 수 없음

```
> i <- 1  
> while(i <=4) {      # 비교조건 : i가 4이하  
+   print('O')        # 반복할 명령문 : 'O'를 출력  
+   i <- i + 1         # i값을 1씩 증가시킴  
+ }                   # 증가/감소시키지 않으면 i값에 변화가 일어나지 않음 -> 무한 반복  
[1] O                 i = 1  
[1] O                 i = 2  
[1] O                 i = 3  
[1] O                 i = 4  
                        i = 5 -> i가 4보다 크므로 조건을 만족시키지 못함 -> 반복이 종료됨
```

## 조건문, 반복문

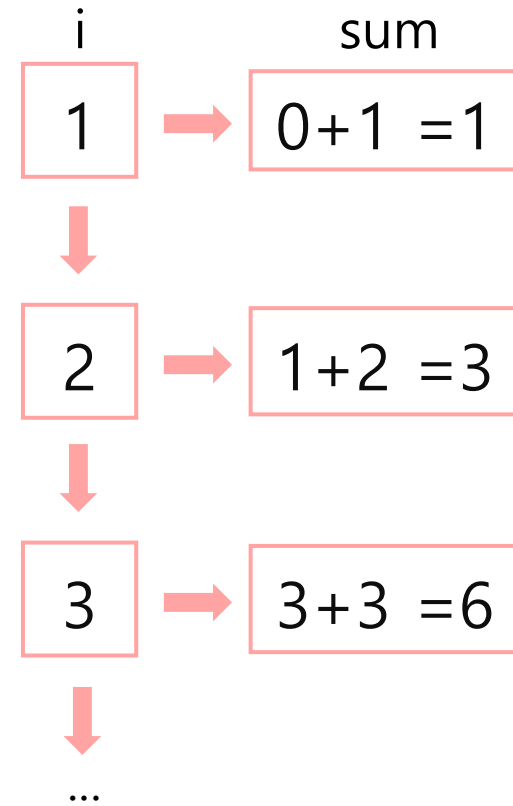
- 반복문

- while문 : 특정 조건이 만족되는 동안 실행 (조건이 거짓일 경우 반복을 종료)
  - 1부터 100까지 숫자의 합 출력하기

```
> sum <- 0
> i <- 1
> while(i <= 100) {
  sum <- sum + i
  i <- i + 1
}
> print(sum)

[1] 5050
```

# 합을 출력할 sum 값을 미리 선언하여 초기값을 저장  
# i값은 1부터 시작  
# 비교 조건 : i가 100이하  
# 반복할 명령문 : sum값을 i와 sum(누적 합계값)으로 재설정  
# i값은 1씩 증가시킴



## 조건문, 반복문

- 반복문
  - break와 next
    - break : 반복문을 중단시킴

```
> sum <- 0
> for( i in 1:10) {
  sum <- sum + i
  if (i>=5) break
}
> sum
[1] 15
```

# i가 5이상이면 break ->for문이 중단

# 1에서 5까지의 합계가 출력

## 조건문, 반복문

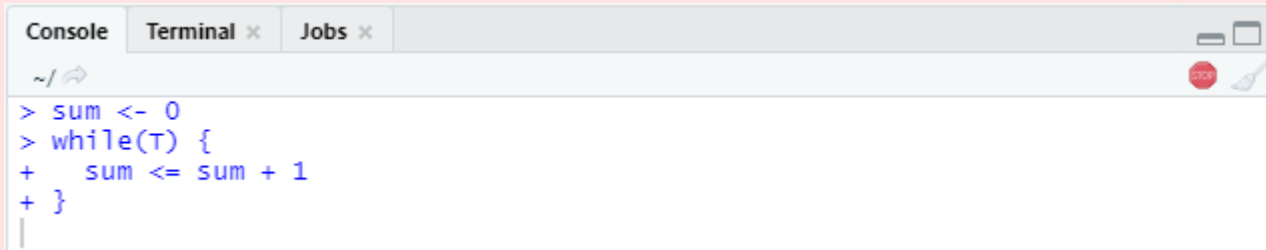
- 반복문
  - break와 next
    - next : 반복문의 시작 지점으로 되돌아감

```
> sum <- 0
> for( i in 1:10) {
  if (i%%2==0) next          # i를 2로 나누었을 때 나머지가 0(짝수)이면 next
                             -> sum <- sum + i 가 실행되지 않고 다음 반복으로 넘어감
  sum <- sum + i
}
> sum                        # 홀수들의 합계가 출력
[1] 25
```

## 조건문, 반복문

- 반복문
  - 무한 반복

R 스튜디오에서 현재 실행 중인 작업을 중단시키려면 콘솔창 상단 오른쪽에 있는 빨간색 아이콘을 클릭하거나 키보드에서 <ESC>를 누르면 됩니다. 무한 루프에 빠진 경우에도 이 기능을 사용하여 중단시킬 수 있습니다.

A screenshot of the R Studio Console window. The window has tabs for 'Console', 'Terminal', and 'Jobs'. The 'Console' tab is active, showing a prompt '~/' and a red stop icon. The code entered is:

```
> sum <- 0
> while(T) {
+   sum <= sum + 1
+ }
```

## 예시

- 예시 1 : 1~100 사이의 정수 중 3의 배수 출력하기

```
for [ ] {  
    [ ] {  
        [ ]  
    }  
}
```

# for문 이용  
# 조건 : 24의 약수

```
[1] 3  
[1] 6  
[1] 9  
.  
.  
.
```



## 예시

- 예시 2 : 24의 약수 구하기

```
for [ ] {  
  if [ ] {  
    print(i)  
  }  
}  
[1]  1  
[1]  2  
[1]  3  
...
```

# for문 이용  
# 조건 : 24의 약수

## 예시

- 예시 3 : 1~100 사이의 정수 중 3의 배수들의 합과 개수

```
num <- 0
sum <- 0
for ( ) {
  if ( ) {
    num <- 
    sum <- 
  }
}
[1] 33 1683
```

# 3의 배수의 개수(초기값 설정)

# 3의 배수의 합(초기값 설정)

# for문을 이용

# 조건 : 3의 배수

# num과 sum을 함께 출력

## 예시

- 예시 4 : while 문을 이용하여 5! 출력하기

```
i <- 1
k <- 1
while  {
  k <- 
  i <- 
}
print(k)
[1] 120
```

# 변수 i의 값을 미리 설정  
# 팩토리얼 값을 저장할 k값 미리 설정  
  
# 팩토리얼 -> 누적 합계값

## 예시

- 예시 5 : while 문을 이용하여 구구단 7단 출력하기

```
i <- 1  
while( ) {  
    
    
}
```

# 7단이므로 7은 유지

```
1 x 7 = 7  
2 x 7 = 14  
3 x 7 = 21  
.  
.  
9 x 7 = 63
```

# 0 x 0 = 0 형태로 출력

## 예시

- 예시 6 : while 문을 이용하여 1~100의 정수 중 4의 배수를 \*로 바꾸어 출력하기

```
i <- 1
while (i <= 100){
  if (  ){
    
  } 
  
  i <- i + 1
}
```

# 4의 배수라면 \*를 출력

# 4의 배수가 아니면 정수 그대로 출력

```
[1] 1
[1] 2
[1] 3
[1] *
[1] 5
...
```

## 예시

- 예시 7 : airquality 데이터프레임에서 Temp가 90이상인 날의 month와 day를 출력하고, 총 몇 일이 조건에 해당하는지 출력하시오.

```
n <- nrow(airquality)
m <- 0
for(i in 1:n){
  if ( ) {
    cat( )
      

  }
}
print(m)
```

```
6 9
6 11
.
9 4
[1] 17
```

```
# airquality 열의 개수를 n에 저장
# 조건에 맞는 행의 개수를 m에 저장

# 조건 : airquality의 Temp가 90 이상
# 90 이상인 행의 Month와 Day를 출력
# 총 몇 일 Temp가 90 이상이었는지 출력
```

감사합니다

[kimtwan21@dongduk.ac.kr](mailto:kimtwan21@dongduk.ac.kr)

김 태 완