제4강
조건문,반복문,함수

Section 01

조건문

1. 조건문

1. if-else문

조건문(conditional statement)에 따라 특정 명령을 실행을 하도록 하는 프로그래밍 명령문

조건에 따라 실행할 명령문을 달리해야 하는 경우에 사용

if-else문의 기본 문법

```
if(비교 조건) {
  조건이 참일 때 실행할 명령문(들)
} else {
  조건이 거짓 일 때 실행할 명령문(들)
}
```

1. 조건문

1.1 기본 if-else문

코드 4-1

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

```
> job.type <- 'A'
> if(job.type == 'B') {
+   bonus <- 200                # 직무 유형이 B일 때 실행
+ } else {
+   bonus <- 100                # 직무 유형이 B가 아닌 나머지 경우 실행
+ }
> print(bonus)
[1] 100
```

1. 조건문

1.2 else가 생략된 if문

코드 4-2

```
job.type <- 'B'
bonus <- 100
if(job.type == 'A') {
        bonus <- 200          # 직무 유형이 B일 때 실행
}
print(bonus)
```

```
> job.type <- 'B'
> bonus <- 100
> if(job.type == 'A') {
+   bonus <- 200              # 직무 유형이 B일 때 실행
+ }
> print(bonus)
[1] 100
```

1. 조건문

1.3 다중 if-else문

코드 4-3

```
score <- 85

if (score > 90) {
            grade <- 'A'
} else if (score > 80) {
              grade <- 'B'
} else if (score > 70) {
              grade <- 'C'
} else if (score > 60) {
              grade <- 'D'
} else {
              grade <- 'F'
}
```

```
> score <- 85
...(중간 생략)
> print(grade)
[1] "B"
```

1. 조건문

1. if와 else 다음에 있는 중괄호 { }는 프로그래밍에서 코드블록이라고 부름

2. 여러 명령문을 하나로 묶어주는 역할

```
> a <- 10
> if(a<5) {
   print(a)
}
else {
 print(a*10)
 print(a/10)
}
[1] 100
[1] 1
```

1.4 조건문에서 논리 연산자의 사용

- if문에 논리 연산자를 사용하면 복잡한 조건문을 서술할 수 있음
- 대표적인 논리연산자는 &(and)와 |(or)

코드 4-4

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

1. 조건문

```
> a <- 10
> b <- 20
> if(a>5 & b>5) {                # and 사용
+   print (a+b)
+ }
[1] 30
> if(a>5 | b>30) {               # or 사용
+   print (a*b)
+ }
[1] 200
```

2. ifelse문

- 조건에 따라 둘 중 하나의 값 또는 변수를 선택할 때 사용
- ifelse문의 문법

코드 4-5

```
a <- 10
b <- 20

if (a>b) {
 c <- a
} else {
 c <- b
}
print(c)
a <- 10
b <- 20
c <- ifelse(a>b, a, b)
print(c)
```

1. 조건문

```
> a <- 10
> b <- 20
...(중간 생략)
> print(c)
[1] 20
>
> a <- 10
> b <- 20
>
> c <- ifelse(a>b, a, b)
> print(c)
[1] 20
```

1. 조건문 코드블록

1. if-else문에서 발생할 수 있는 오류

2. else는 반드시 if문의 코드블록이 끝나는 부분에 있는 }와 같은 줄에 작성해야 함

```
job.type <- 'A'

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

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

Section 02

반복문

2. 반복문

1. for문

- 반복문(repetitive statement)은 정해진 동작을 반복적으로 수행할 때 사용하는 명령문
- 동일 명령문을 여러 번 반복해서 실행할 때 사용
- for문의 문법

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

2. 반복문

1.1 기본 for문

코드 4-6

```
for(i in 1:5) {
 print('*')
}
```

```
> for(i in 1:5) {
+   print('*')
+ }
[1] "*"
[1] "*"
[1] "*"
[1] "*"
[1] "*"
```

2. 반복문

1.2 반복 범위에 따른 반복 변수의 값 변화

코드 4-7

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

```
> for(i in 6:10) {
+     print(i)
+ }
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

2. 반복문

1.3 반복 변수를 이용한 구구단 출력

코드 4-8

```
for(i in 1:9) {
 cat('2 *', i,'=', 2*i,'₩n')
}
```

```
> for(i in 1:9) {
+     cat('2 *', i,'=', 2*i,'\n')
+ }
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
```

2. 반복문

1.4 for문 안에서 if문의 사용

코드 4-9

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

2. 반복문

```
> for(i in 1:20) {
+     if(i%%2==0) {                    # 짝수인지 확인
+         print(i)
+     }
+ }
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
[1] 12
[1] 14
[1] 16
[1] 18
[1] 20
```

2. 반복문

1.5 1~100 사이의 숫자의 합 출력

코드 4-10

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

```
> sum <- 0
> for(i in 1:100) {
+     sum <- sum + i                    # sum에 i 값을 누적
+ }
> print(sum)
[1] 5050
```

2. 반복문

1.6 iris에서 꽃잎의 길이에 따른 분류 작업

코드 4-11

```
norow <- nrow(iris)                          # iris의 행의 수
mylabel <- c( )                              # 비어있는 벡터 선언
for(i in 1:norow) {
        if (iris$Petal.Length[i] <= 1.6) { # 꽃잎의 길이에 따라 레이블 결정
                mylabel[i] <- 'L'
        } else if (iris$Petal.Length[i] >= 5.1) {
                mylabel[i] <- 'H'
        } else {
                mylabel[i] <- 'M'
        }
}
print(mylabel)                                       # 레이블 출력
newds <- data.frame(iris$Petal.Length, mylabel) # 꽃잎의 길이와 레이블 결합
head(newds)                                  # 새로운 데이터셋 내용 출력
```

2. 반복문

```
> norow <- nrow(iris)                                    # iris의 행의 수
> mylabel <- c()                                         # 비어있는 벡터 선언
> for(i in 1:norow) {
+     if (iris$Petal.Length[i] <= 1.6) {                 # 꽃잎의 길이에 따라 레이블 결정
+         mylabel[i] <- 'L'
+     } else if (iris$Petal.Length[i] >= 5.1) {
+         mylabel[i] <- 'H'
+     } else {
+         mylabel[i] <- 'M'
+     }
+ }
> print(mylabel)                                         # 레이블 출력
  [1] "L" "L" "L" "L" "L" "M" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L"
 [18] "L" "M" "L" "M" "L" "L" "M" "M" "L" "L" "L" "L" "L" "L" "L" "L" "L"
 [35] "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "M" "L" "L" "L" "L" "L" "M"
 [52] "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M"
```

2. 반복문

```
 [69] "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "H" "M"
 [86] "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "H" "H"
[103] "H" "H" "H" "H" "M" "H" "H" "H" "H" "H" "H" "M" "H" "H" "H" "H" "H"
[120] "M" "H" "M" "H" "M" "H" "H" "M" "M" "H" "H" "H" "H" "H" "H" "H" "H"
[137] "H" "H" "M" "H" "H" "H" "H" "H" "H" "H" "M" "H" "H" "H"
> newds <- data.frame(iris$Petal.Length, mylabel) # 꽃잎의 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
```

2. while문

- while문은 어떤 조건이 만족하는 동안 코드블록을 수행하고, 해당 조건이 거짓일 경우 반복을 종료하는 명령문

```
while (비교조건) {
 반복할  명령문(들)
}
```

코드 4-12

```
sum <- 0
i <- 1
while(i <=100) {
        sum <- sum + i    # sum에 i 값을 누적
        i <- i + 1        # i 값을 1 증가시킴
}
print(sum)
```

2. 반복문

```
> sum <- 0
> i <- 1
> while(i <=100) {
+     sum <- sum + i                # sum에 i 값을 누적
+     i <- i + 1                    # i 값을 1 증가시킴
+ }
> print(sum)
[1] 5050
```

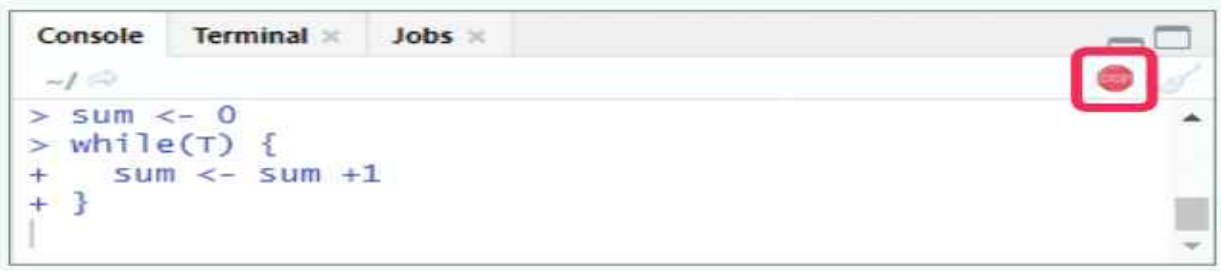

그림 4-1 콘솔(Console) 창의 STOP 아이콘

3. break와 next

3.1 break

코드 4-13

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

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

2. 반복문

3.2 next

코드 4-14

```
sum <- 0
for(i in 1:10) {
  if (i%%2==0) next
  sum <- sum + i
}
```

```
> sum <- 0
> for(i in 1:10) {
+     if (i%%2==0) next
+     sum <- sum + i
+ }
> sum
[1] 25
```

감사합니다.