Ch04 제어문과 함수

1. 연산자

산술연산자, 관계연산자, 논리연산자

1.1 산술연산자

사칙연산(+, -, *, /), 나머지 계산 연산자(%%), 거듭제곱(^ 또는 **)계산 연산자

실습 (산술연산자 사용)

```
num1 <- 100
num2 <- 20
result <- num1 + num2
result
result <- num1 - num2
result
result <- num1 * num2
result
result <- num1 / num2
result

result <- num1 %% num2
result
```

```
result <- num1 ^ 2
result
result <- num1 ^ num2
result
```

"1e+40" : "1 * 10^40"과 같은 결과를 나타냄

1.2 관계연산자

관계연산자(==, !=, >, >=, <, <=)

관계식의 결과가 참이면 TRUE, 거짓이면 FALSE값을 반환하는 연산자

실습 (관계연산자 사용)

```
boolean <- num1 == num2
boolean
boolean <- num1 != num2
boolean

boolean <- num1 > num2
boolean
boolean <- num1 >= num2
boolean
boolean <- num1 < num2
```

boolean

boolean <- num1 <= num2

boolean

1.3 논리연산자

TRUE / FALSE 값 반환

실습 (논리연산자 사용)

logical <- num1 >= 50 & num2 <= 10

logical

logical <- num1 >= 50 | num2 <= 10

logical

logical <- num1 >= 50

logical

logical <= !(num1 >= 50)

logical

x <- TRUE; y <- FALSE

xor(x, y)

xor() 함수:

X	Y	XOR
T	T	F
T	F	T
F	T	T
F	F	F

2. 조건문

if(), ifelse(), switch(), which()함수

2.1 if()문

형식:

if(조건식) { 참인 경우 처리문

} else { 거짓인 경우 처리문

}

{ }을 이용하여 if와 else의 블록을 지정한다.

실습 (if()함수 사용)

```
x <- 50; y <- 4; z <- x * y
if(x * y >= 40) {
   cat("x * y의 결과는 40이상입니다.₩n")
   cat("x * y = ", z)
} else {
   cat("x * y의 결과는 40미만입니다. x * y = ", z, "₩n")
}
```

실습 (if()함수 이용 점수의 학점 산출)

```
score <- scan()


score
result <- "노력"
if(score >= 80) {
   result <- "우수"
}
cat("당신의 학점은 ", result, score)
```

실습 (if~else if 형식으로 학점 산출)

```
score <- scan()
if(score >= 90) {
   result = "A학점"
} else if(score >= 80) {
   result = "B학점"
} else if(score >= 70) {
   result = "C학점"
} else if(score >= 60) {
   result = "D학점"
} else {
   result = "F학점"
}
cat("당신의 학점은", result)
```

```
print(result)
```

2.2 ifelse()함수

조건식이 참인 경우와 거짓인 경우 처리할 문장을 ifelse()문에 포함

형식:

Ifelse(조건식, 참인 경우 처리문, 거짓인 경우 처리문)

실습 (ifelse()함수 사용)

```
score <- scan()

ifelse(score >= 80, "우수", "노력")
ifelse(score <= 80, "우수", "노력")
```

실습 (ifelse()함수 응용)

```
excel <- read.csv("C:/Rwork/Part-I/excel.csv", header = T)
q1 <- excel$q1
q1
ifelse(q1 >= 3, sqrt(q1), q1)
```

실습 (ifelse()함수에서 논리연산자 사용)

```
ifelse(q1 >= 2 & q1 <= 4, q1 ^ 2, q1)
```

2.3 switch()함수

비교 문장의 내용에 따라 여러 개의 실행문 중 하나를 선택할 수 있도록 프로그램 작성

형식:

switch(비교문, 실행문1 [, 실행문2, 실행문 3, ...])

비교문 위치에 있는 변수의 이름이 실행문 위치에 있는 변수의 이름과 일치할 때 일치하는 변수에 할당된 값을 출력

```
# switch() 함수 사용 예
switch("name", id = "hong", pwd = "1234", age = 105, name = "홍길동")
```

실습 (switch()함수를 사용하여 사원명으로 급여정보 보기)

```
empname <- scan(what = "")
empname
switch(empname,
        hong = 250,
        lee = 350,
```

```
    kim = 200,
    kang = 400
)
```

2.4 which()함수

벡터 객체를 대상으로 특정 데이터를 검색하는데 사용되는 함수

조건식이 결과가 참인 벡터 원소의 위치(인덱스)가 출력되며, 조건식의 결과가 거짓이면 0이 출력된다.

형식: which(조건)

실습 (벡터에서 which()함수 사용하여 index값 반환)

```
name <- c("kim", "lee", "choi", "park")
which(name == "choi")
```

실습 (데이터프레임에서 which()함수 사용)

데이터프레임에서 해당 원소가 없으면 0이 출력

which함수는 크기가 큰 데이터프레임이나 테이블 구조의 자료를 대상으로 특정 정보를 검색하는데 사용

```
# 데이터프레임 생성
no <- c(1:5)
```

```
name <- c("홍길동", "이순신", "강감찬", "유관순", "김유신")
score <- c(85, 78, 89, 90, 74)
exam <- data.frame(학번 = no, 이름 = name, 성적 = score)
exam

# 일치하는 이름의 인덱스 반환
which(exam$이름 == "유관순")
exam[4, ]
```

3. 반복문

조건에 따라서 특정 실행문을 지정된 횟수만큼 반복적으로 수행할 수 있는 문

for()함수, while()함수

3.1 for()함수

지정한 횟수만큼 실행문을 반복 수행하는 함수

형식: for(변수 in qustn) {실행문}

실습 (for()함수 사용)

*반복할 문장이 하나 뿐일 때는 { } 생략 가능

```
i <- c(1:10)
for(n in i) {
   print(n * 10)
   print(n)
}
```

실습 (짝수 값만 출력)

```
i <- c(1:10)
for(n in i)
   if(n %% 2 == 0) print(n)
```

%%연산자: 나머지 계산

실습 (짝수이면 넘기고, 홀수 값만 출력)

```
i <- c(1:10)
for(n in i) {
   if(n %% 2 == 0) {
      next
   } else {
      print(n)
   }
}
```

next문: for()함수의 반복 범위에서 문장을 실행하지 않고 계속 반복할 때 사용

실습 (변수의 컬럼명 출력)

```
name <- c(names(exam))
for(n in name) {
   print(n)
}
```

실습 (벡터 데이터 사용)

```
score <- c(85, 95, 98)
name <- c("홍길동", "이순신", "강감찬")

i <- 1
for(s in score) {
   cat(name[i], " -> ", s, "₩n")
   i <- i + 1
}
```

3.2 while()함수

for()함수는 반복 회수를 결정하는 변수를 사용. While()함수는 사용자가 블록 내에서 증감식을 이용하여 반복 회수를 지정해야 한다.

형식: while(조건) { 실행문 }

실습 (while()함수 사용)

```
i = 0
while(i < 10) {
   i <- i + 1
   print(i)
}
```

4. 함수의 정의

사용자 정의 함수: 사용자가 직접 함수 내에 필요한 코드를 작성하여 필요한 경우 함수를 호출하여 사용하기 위하여 작성한 함수

형식: 함수명 <- function(매개변수) { 실행문 }

4.1 사용자 정의 함수

실습 (매개변수가 없는 사용자 함수 정의)

```
f1 <- function() {
  cat("매개변수가 없는 함수")
}

f1()
```

매개변수가 없는 함수는 '함수명()' 형태로 함수명에 빈 괄호를 붙여 호출

정의된 함수를 호출하지 않으면 함수의 내용은 실행되지 않는다.

실습 (결과를 반환하는 사용자 함수 정의)

```
f3 <- function(x, y) {
  add <- x + y
  return(add)
```

```
}

add <- f3(10, 20)
```

return()함수: 사용자 정의 함수 내에 있는 값을 함수를 호출하는 곳으로 반환하는 역할

4.2 기술통계량을 계산하는 함수 정의

요약통계량, 빈도 수 등의 기술통계량을 계산하는 함수를 정의

실습 (기본함수를 이용하여 요약통계량과 빈도수 구하기)

```
# 파일 불러오기
setwd("C:/Rwork/")
test <- read.csv("test.csv", header = TRUE)
head(test)
```

summary()함수, table()함수

```
# 요약 통계량 구하기
summary(test)

# 특정 변수의 빈도수 구하기
```

```
table(test$A)


# 각 칼럼 단위의 비도수와 최대값, 최소값 계산을 위한 사용자 함수 정의하기
data_pro <- function(x) {
  for(idx in 1:length(x)) {
    cat(idx, "번째 칼럼의 빈도 분석 결과")
    print(table(x[idx]))
    cat("₩n")
  }

  for(idx in 1:length(x)) {
    f <- table(x[idx])
    cat(idx, "번째 칼럼의 최대값/최소값₩n")
    cat("max = ", max(f), "min = ", min(f), "₩n")
  }
}

data_pro(test)
```

사용자 정의함수 data_pro(): 컬럼 단위로 빈도수, 최대값, 최소값 계산

실습 (분산과 표준편차를 구하는 사용자 함수 정의)

표본분산: x변량을 대상으로 "변량의 차의 제곱이 합 / (변량의 개수-1) "

표본분산 식: var <-sum((x-산술평균)^2) / (length(x)-1)

표본표준편차 식: sqrt(var)

Ex. X = {7, 5, 12, 9, 15, 6}

X의 평균?

X의 중위수?

X의 분산?

X의 표준편차?

```
x <- c(7, 5, 12, 9, 15, 6)

var_sd <- function(x) {
   var <- sum(x - mean(x) / 2) / (length(x) - 1)
   sd <- sqrt(var)
   cat("표본분산: ", var, "₩n")
   cat("표본표준편차: ",sd)
}

var_sd(x)
```

4.3 피타고라스와 구구단 함수 정의

실습 (피타고라스식 정의 함수 만들기)

```
pytha <- function(s, t) {
  a <- s ^ 2 - t ^ 2
  b <- 2 * s * t
  c <- s ^ 2 + t ^ 2
  cat("피타고라스 정리: 3개의 변수: ", a, b, c)
}

pytha(2, 1)
```

실습 (구구단 출력 함수 만들기)

```
gugu <- function(i, j) {
  for(x in i) {
    cat("**", x, "단**₩n")
    for(y in j) {
      cat(x, " * ", y, " = ", x * y, "₩n")
    }
    cat("₩n")
  }
}

i <- c(2:9)
j <- (1:9)
```

```
gugu(i, j)
```

4.4 결측치 포함 자료의 평균 계산 함수 정의

결측치 데이터(NA)를 포함하는 데이터의 평균을 구하기 위해서는 먼저 결측치를 처리한 후 평균을 계산해야 한다.

실습 (결측치를 포함하는 자료를 대상으로 평균 구하기)

```
# 단계 1: 결측치(NA)를 포함하는 데이터 생성
data <- c(10, 20, 5, 4, 40, 7, NA, 6, 3, NA, 2, NA)
```

단계2에서 1차: NA제거하고 평균 산출, 2차: NA를 0으로 대체하여 평균을 구한다.

3차: NA를 평균으로 대체하여 평균을 구한다.

```
# 단계 2: 결측치 데이터를 처리하는 함수 정의
na <- function(x) {
  # 1차: NA 제거
  print(x)
  print(mean(x, na.rm = T))

  # 2차: NA를 0으로 대체
  data = ifelse(!is.na(x), x, 0)
  print(data)
```

```
  print(mean(data))

  # 3차: NA를 평균으로 대체
  data2 = ifelse(!is.na(x), x, round(mean(x, na.rm = TRUE), 2))
  print(data2)
  print(mean(data2))
}

# 단계 3: 결측치 처리를 위한 사용자 함수 호출
na(data)
```

4.5 몬테카를로 시뮬레이션 함수 정의

몬테카를로 시뮬레이션은 현실적으로 불가능한 문제의 해답을 얻기 위해 난수의 확률 분포를 이용하는 모의 시험으로 근사적 해를 구하는 기법

실습 (동전 앞면과 뒷면에 대한 난수 확률분포의 기대확률 모의시험)

```
# 동전 앞면과 뒷면의 난수 확률분포 함수 정의
coin <- function(n) {
  r <- runif(n, min = 0, max = 1)
  result <- numeric()
  for(i in 1:n) {
    if(r[i] <= 0.5)
      result[i] <- 0
```

```
    else
      result[i] <- 1
  }
  return(result)
}

# 동전 던지기 횟수가 10회인 경우 앞면(0)과 뒷면(1)이 나오는 vector 값
coin(10)

# 몬테카를로 시뮬레이션 함수 정의
montaCoin <- function(n) {
  cnt <- 0
  for(i in 1:n) {
    cnt <- cnt + coin(1)
  }

  result <- cnt / n
  return(result)
}

# 몬테카를로 시뮬레이션 함수 호출
montaCoin(10)
montaCoin(30)
montaCoin(100)
montaCoin(1000)
```

montaCoin(10000)

runif() 함수: 난수 발생 함수

5. 주요 내장함수

5.1 기술통계량 처리 관련 내장함수

[표 4.2] 기술통계량 처리 관련 내장함수

Min(vec), max(vec), range(vec), mean(vec), median(vec), sum(vec), sort(x),

order(x): 벡터의 정렬된 값의 index를 보여주는 함수

rank(x), sd(x), summary(x), table(x),

sample(x,y): x범위에서 y만큼 sample데이터를 생성하는 함수

실습 (행/컬럼 단위의 합계와 평균 구하기)

```
library(RSADBE)
data("Bug_Metrics_Software")
Bug_Metrics_Software[ , , 1]

# 행 단위 합계와 평균 구하기
rowSums(Bug_Metrics_Software[ , , 1])
rowMeans(Bug_Metrics_Software[ , , 1])


# 열 단위 합계와 평균 구하기
colSums(Bug_Metrics_Software[ , , 1])
```

colMeans(Bug_Metrics_Software[, , 1])

rowSums()함수: 행 단위 합계

rowMeans()함수: 행단위 평균

colSums()함수: 열 단위 합계

colMeans()함수: 열 단위 평균

실습 (기술통계량 관련 내장함수 사용하기)

[표 4.2] 기술통계량 처리 관련 내장함수 사용

seq(-2, 2, by = .2)

vec <- 1:10

min(vec)

max(vec)

range(vec)

mean(vec)

median(vec)

sum(vec)

sd(rnorm(10))

table(vec)

실습 (정규분포의 난수 생성)

```
n <- 1000

rnorm(n, mean = 0, sd = 1)

hist(rnorm(n, mean = 0, sd = 1))
```

rnorm()함수

형식: rnorm(n, mean, sd) #평균과 표준편차 이용

실습 (균등분포의 난수 생성하기)

형식: runif(n, min, max) #최소값, 최대값 이용

```
n <- 1000

runif(n, min = 0, max = 10)

hist(runif(n, min = 0, max = 10))
```

실습 (이항분포의 난수 생성하기)

형식: rbinom(n, size, prob) #독립적 n의 반복

```
n <- 20

rbinom(n, 1, prob = 1 /2 )

rbinom(n, 2, 0.5)

rbinom(n, 10, 0.5)

n <- 1000
```

rbinom(n, 5, prob = 1 / 6)

실습 (종자값으로 동일한 난수 생성)

종자(seed)값을 지정하면 동일한 난수 발생 가능

종자(seed)값 지정을 위해 seed()함수 사용

형식: set.seed(임의의 정수) #임의의 정수를 종자값으로 하여 동일한 난수 생성

rnorm(5, mean = 0, sd = 1)

set.seed(123)

rnorm(5, mean = 0, sd = 1)

set.seed(123)

rnorm(5, mean = 0, sd = 1)

set.seed(345)

rnorm(5, mean = 0, sd = 1)

실습에서 난수가 동일한지 확인

5.2 수학 관련 내장함수

[표 4.3] 수학 관련 내장함수

abs(x), sqrt(x), ceiling(x), floor(), round(), factorial(x),

which.min(x), which.max(x): 벡터 내 최소값과 최대값의 인덱스를 구하는 함수,

Pmin(x, pmax(x): dufj 벡터에서의 원소 단위 최소값과 최대갑을 구하는 함수,

Prod(): 곱

cumsum(), cumprod(), cos(x), sin(x), tan(x), log(x), log10(x), exp(x)

실습 (수학 관련 내장함수 사용)

```
vec <- 1:10
proc(vec)
prod(vec)
factorial(5)
abs(-5)
sqrt(16)
vec
cumsum(vec)
log(10)
log10(10)
```

5.3 행렬연산 관련 내장함수

행렬 분해(matrix decomposition)는 행렬을 특정한 구조를 가진 다른 행렬의 곱으로 나타내는 것

eigen(), svd(), qr(), chol()

[표 4.4] 행렬연산 관련 내장함수

ncol(x), nrow(x), t(x), cbind(...), rbind(...), diag(x), det(x), appply(x, m, fun),

solve(x): 역행렬,

eigen(x): 정방행렬을 대상으로 고유값 분해하는 함수

svd(x): mxn 행렬을 대상으로 특이값을 분해하는 함수

x%*%y: 두 행렬의 곱을 구하는 수식

실습 (행렬연산 내장함수 사용하기)

[표 4.4] 행렬연산 관련 내장함수 사용

```
x <- matrix(1:9, nrow = 3, ncol = 3, byrow = T)
y <- matrix(1:3, nrow = 3)
ncol(x)
nrow(x)
t(x)
cbind(x, 1:3)
rbind(x, 10:12)
diag(x)
det(x)
apply(x, 1, sum)
apply(x, 2, mean)
svd(x)
eigen(x)
```

x %*% y

==========

R %any%특별연산자

1. %% 나머지 연산자

형식: a %% b
a를 b로 나눈 나머지
> x <-c(1,2,3,4,5)
>y <-c(5)
>x %% y
[1] 1 2 3 4 0

2. %*% 연산자

형식: a %*% b
연산자는 a와 b를 곱한다.
Cf) 교재 p131 & p132

Ex.
x <- matrix(c(1, 4, 2, 3), nrow = 2)
x
y <- matrix(c(1, 3, 2, 4, 5, 6), nrow = 2)
y

2행 2열인 Matrix와 2행 3열인 Matrix를 곱하기 연산을 한다.
x%*%y

z <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3)
z

2행 2열인 Matrix와 3행 3열인 Matrix를 곱하기 연산을 하면 오류가 발생한다.

x%*%z

2행 3열인 Matrix와 3행 3열인 Matrix를 곱하기 연산을 한다.
y%*%z

# Vector의 곱
c(1, 2, 3) %*% c(4, 5, 6)
결과는?

3. %/% 연산자

형식: a %/% b
A를 b로 나눈다.
요소별로 연산한다.
%/% 연산자로 나눈 결과에서 소수점은 절삭을 하여 반환한다.

Ex.
x <- matrix(c(1, 4, 2, 3), nrow = 2)
x
y <- matrix(c(1, 3, 2, 4), nrow = 2)
y

나누기 연산을 한다.
x %/% y
z <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3)

크기가 맞지 않아 오류가 발생한다.
x %/% z

4. **벡터 내 특정 값 포함 여부 확인 연산자 %in%**

형식: a %in% b
벡터 내 특정값이 포함되었는지 여부 확인

TRUE, FALSE 논리형 벡터 출력

Ex.

```
x %in% y
[1] TRUE TRUE TRUE TRUE

sum(x %in% y)
[1] 1
```

==========

5.4 집합연산 관련 내장함수

집합을 대상으로 합집합, 교집합, 차집합 등의 집합연산을 수행하는 R의 내장함수

[표 4.5] 집합연산 관련 내장함수

union(x,y), setequal(x,y), intersect(x,y), setdiff(x,y), c%in%y

실습 (집합연산 관련 내장함수 사용)

[표 4.5] 집합연산 관련 내장함수 사용

```
x <- c(1, 3, 5, 7, 9)

y <- c(3, 7)

union(x, y)

setequal(x, y)

intersect(x, y)

setdiff(x, y)
```

setdiff(y, x)

5 %in% y

ch4 연습문제