pr6

황인규

2023-10-19

조건문과 반복문

1. 반복문

- 동일한 연산을 여러번 반복하고 싶을때 사용

1.1 for()

- 데이터를 순서대로 i로 받아서 사용됨

```
a = c(1,2,4); a
```

```
## [1] 1 2 4
```

```
for(i in a) print(i)
```

```
## [1] 1
## [1] 2
## [1] 4
```

```
for(i in c(1,2,4))
  print(i)
```

```
## [1] 1
## [1] 2
## [1] 4
```

```
string = c("test", "for", "for()")
for(i in string) print(i)
```

```
## [1] "test"
## [1] "for"
## [1] "for()"
```

```
for(i in c("test", "for", "for()"))
  print(i)
```

```
## [1] "test"
## [1] "for"
## [1] "for()"
```

- for 문에서 {}없이 연산을 할 경우, for 문에 첫 명령문만 반복되고 그 다음줄은 별개의 코드로 인식

```
for (i in 1:3)
  print(i)
```

```
## [1] 1
## [1] 2
## [1] 3
```

```
print("다음줄")
```

```
## [1] "다음줄"
```

```
print(i + 1)
```

```
## [1] 4
```

- 여러 연산을 같이 반복하고 싶다면 {} 안에 명령문을 작성해야 함

```
for (i in 1:3)
  print(i)
```

```
## [1] 1
## [1] 2
## [1] 3
```

```
  print("다음줄")
```

```
## [1] "다음줄"
```

```
  print(i + 1)
```

```
## [1] 4
```

- 응용, vector J의 짝 수 번째 데이터들만 출력하기

```
j = 1:6
for (i in seq(2,6,by =2)){
  print(paste0(i, "번째 데이터 :"))
  print(j[i])
}
```

```
## [1] "2번째 데이터 :"
## [1] 2
## [1] "4번째 데이터 :"
## [1] 4
## [1] "6번째 데이터 :"
## [1] 6
```

- 꼭 i 를 쓰지 않아도 됨

```
for (a in 1:10)
  print(a)
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

- break라는 명령어를 수행하면 반복문은 종료됨

```
for (i in 1:10){
  print(i)
  break
}
```

```
## [1] 1
```

1.2 while()

- while 문은 조건식과 같이 사용, 주어진 조건식이 참일 경우 반복 수행
- while(조건)(명령): 조건이 참일때 명령문 실행
- 주의사항: 조건이 계속 참이면 무한반복이 발생

```
i = 1
while (i<5){
  print(i)
  i = i + 1
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

1.3. repeat

- 반복과 정지에 대한 조건문을 활용하여 명령문을 반복함

```
i = 1
repeat {
  i = i + 30
  if (i > 100) break
  print(i)
}
```

```
## [1] 31
## [1] 61
## [1] 91
```

2.조건문

2.1 if 문과 if-else 문

- if (조건식) 연산이_ 방법으로 사용
- 단순 if 문은 if(조건)명령문 의 형태로 작성

```
x = 80
if (x<=90) print("B")
```

```
## [1] "B"
```

```
if (x<=80) print("C")
```

```
## [1] "C"
```

- 여러개의 명령문을 수행할 땐 {}연산자 사용
- 명령문 추가 줄 바꿈해서 작성

```
x = 91
if (x> 90){
  print("B")
  x = x +10
  print(x)
}
```

```
## [1] "B"
## [1] 101
```

- 단순 if-else문은 if(조건){조건이 참일때 명령} else {조건이 거짓일 때 명령문}의 형태로 작성

```
x = 91
if (x< 90){
  print("B")
} else {
  print("C")
}
```

```
## [1] "C"
```

```
if (x< 90){
  print("B")
  x = x +10
} else {
  print("C")
}
```

```
## [1] "C"
```

- if 와 else를 사용해서 2개 이상의 조건문을 만들고 싶을때 else if 를 사용
- else if 는 else인 조건 상황에서 다시 if문 조건을 쓸때 사용

```
x = 100
if (x<70){
  print("F")
} else if (x<80){
  print("C")
} else if (x<90){
  print("B")
} else{
  print("A")
}
```

```
## [1] "A"
```

2.2 ifesle

- 한 줄의 명령문으로 조건이 참일 때와 거짓일 때 따라 명령문을 수행
- ifesle(조건식, 참일 때 실행할 내용, 거짓일때 실행할 내용)

```
x = 10
ifelse(x > 5, x+5, x-5)
```

```
## [1] 15
```

3.조건문과 반복문 함께 사용하기

- x의 데이터 중 짝수만 출력하도록 함

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

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

##Pr6 ### 문제 1

```
for (i in 1:9) {
  print(paste0("2 * ", i , " = ", 2*i))
}
```

```
## [1] "2 * 1 = 2"
## [1] "2 * 2 = 4"
## [1] "2 * 3 = 6"
## [1] "2 * 4 = 8"
## [1] "2 * 5 = 10"
## [1] "2 * 6 = 12"
## [1] "2 * 7 = 14"
## [1] "2 * 8 = 16"
## [1] "2 * 9 = 18"
```

문제 2

```
n = 24
find_minimum_boxes <- function(n) {
  if (n < 3) {
    return(-1)
  }

  if (n %% 7 == 0) {
    return(n %/% 7)
  }

  max_seven_boxes <- n %/% 7
  remaining_weight <- n - max_seven_boxes * 7

  while (max_seven_boxes >=0 ) {
    if (remaining_weight %%3 ==0) {
      return(max_seven_boxes + remaining_weight %/%3)
    }

    max_seven_boxes <- max_seven_boxes -1
    remaining_weight <- n - max_seven_boxes *7
   }

   return(-1)
}


minimum_boxes <- find_minimum_boxes(n)
print(minimum_boxes)
```

```
## [1] 4
```