Practical No. 7

Title: Study and Implementation of Various Control Structures in R

Introduction:

Control structures in programming define the flow of execution of statements. In R, like other programming languages, control structures are essential for directing the execution of code based on logical conditions or repetitive tasks. The most common control structures in R include conditional statements (if-else), loops (for, while), and functions for controlling program execution. Understanding these structures is fundamental to writing efficient and functional code in R.

Objective:

- To understand and implement various control structures in R.
- To learn how to use conditional statements (`if`, `ifelse`) and loops (`for`, `while`, `repeat`) to control the flow of execution.
- To explore and practice the use of functions like `break`, `next`, and `return` in loops and conditional constructs.

Tools Required:

- **Software:** R (version 4.0 or higher)
- **IDE (Recommended):** RStudio
- **Libraries:** Base R (No additional libraries required)

Theory:

1. Conditional Statements:

Conditional statements are used to make decisions based on certain conditions. In R, these are implemented using the `if`, `else if`, and `else` constructs.

`if` Statement:

The `if` statement executes a block of code if the given condition is true.

Syntax:

```
if (condition) {
  # Code to be executed if condition is TRUE
}
```

Example:

```
x <- 10
if (x > 5) {
  print("x is greater than 5")
}
```

`ifelse()` Function:

`ifelse()` is a vectorized conditional function used to evaluate a condition and return one value if the condition is true, and another if false.

Syntax:

```
CopyEdit
ifelse(test, yes, no)
```

Example:

```
x <- 8
result <- ifelse(x > 5, "Greater", "Lesser")
print(result)  # Outputs: Greater
```

`else` and `else if` Statements:

These are used to handle multiple conditions.

Syntax:

```
if (condition1) {
  # Code for condition1
} else if (condition2) {
  # Code for condition2
} else {
  # Code for all other conditions
}
```

Example:

```
x <- 8
if (x > 10) {
  print("x is greater than 10")
} else if (x == 8) {
  print("x is equal to 8")
} else {
  print("x is less than 8")
}
```

2. Loops in R:

Loops are used for repeating a block of code multiple times. The main types of loops in R are the `for`, `while`, and `repeat` loops.

`for` Loop:

The `for` loop is used to iterate over a sequence (e.g., a vector or list) and execute a block of code for each element.

Syntax:

```
for (variable in sequence) {
  # Code to be executed
}
```

Example:

```
for (i in 1:5) {
```

```
  print(i)
}
```

`while` Loop:

The `while` loop executes a block of code as long as a given condition remains `TRUE`.

Syntax:

```
while (condition) {
  # Code to be executed as long as condition is TRUE
}
```

Example:

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

`repeat` Loop:

The `repeat` loop executes a block of code indefinitely unless it is explicitly terminated using `break`.

Syntax:

```
repeat {
  # Code to be executed
  if (condition) {
    break
  }
}
```

Example:

```
i <- 1
repeat {
  print(i)
  i <- i + 1
  if (i > 5) {
    break
  }
}
```

3. Loop Control:

Control flow inside loops can be modified using the `break`, `next`, and `return` statements.

`break` Statement:

The `break` statement is used to exit a loop prematurely.

Example:

```
for (i in 1:10) {
  if (i == 6) {
    break
  }
```

```
  print(i)
}
```

`next` Statement:

The `next` statement is used to skip the current iteration and proceed to the next iteration in a loop.

Example:

```
for (i in 1:5) {
  if (i == 3) {
    next  # Skip the iteration where i equals 3
  }
  print(i)
}
```

`return()` Function:

The `return()` function is used to exit a function and return a value. It can also be used within loops to exit the function early.

Example:

```
sum_numbers <- function(n) {
  total <- 0
  for (i in 1:n) {
    if (i > 5) {
      return(total)  # Exit the function if i > 5
    }
    total <- total + i
  }
  return(total)
}
print(sum_numbers(10))  # Outputs: 15
```

Conclusion:

In this practical, we have explored and implemented various control structures in R. We began by learning how to control the flow of the program using **conditional statements** (`if`, `ifelse`, `else`), and then moved on to using **loops** (`for`, `while`, `repeat`) for iterative tasks. We also saw how to control the flow within loops using `break`, `next`, and `return`. Mastery of these control structures is fundamental to writing effective R code, especially when dealing with large datasets, repetitive tasks, or complex logic.