# Podstawy programowania w R - część 2

## Funkcje sterujące

1. Przeanalizuj działanie poniższej pętli for.

```
v <- c("a", "b", "c", "d", "e", "f", "g", "h", "i", "j")
for(i in v) {
      print(i)
}

for(i in 1:10) {
      print(v[i])
}

for(i in seq_along(v)) {
      print(v[i])
}
```

2. Przeanalizuj działanie poniższej pętli while

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

3. Przeanalizuj działanie poniższej pętli repeat. Zwróć uwagę na strukturę if-else, next oraz break.

```
i <- 1
repeat {
      i <- i + 1
      if(i == 11) {
            break;
      } else if(i == 5) {
            next;
      } else {
```

```
            print(i)
      }
}
```

## Funkcje

1. Stwórz funkcje obliczającą wartość funkcji liniowej w punkcie x o podanych parametrach a,b, a następnie wyświetl wprowadzoną funkcje.

```
y <- function(x, a=1, b=0) {
      a*x + b
}
y
```

2. Wywołaj funkcje bez parametrów. Jak z pewnością zaobserwujesz jest to niemożliwe.

```
y()
```

3. Wywołaj funkcje z domyślnymi parametrami wprowadzając tylko argument x.

```
y(1)
```

4. Wywołaj funkcje z parametrami x=1, a=2, b=3. Zwróć uwagę na różne sposoby wywoływania funkcji – w jaki sposób odbywa się dopasowanie argumentów

```
y(1,2,3)
y(b=3, a=2, x=1)
y(b=3, 1, a=2)
```

5. Zmodyfikuj funkcje f wprowadzając dodatkowy argument c bez domyślnej wartości. Czy wywołanie funkcji wyłącznie z parametrem x będzie możliwe: y(1)?

```
y <- function(x, a=1, b=0, c) {
      a*x + b
}
```

6. Stwórz funkcje z argumentem „…”.

```
s <- function(...) {
      arg <- c(...)
      sum <- 0
      for(i in arg) {
            sum <- sum + i
      }
      sum
}
```

```
s(1,2,3,4,5)
```

## Lexical scoping

1. Wyświetl listę pakietów, w kolejności w jakiej są przeszukiwane w poszukiwaniu obiektów. Wykorzystaj funkcje search(). Zwróć uwagę, że w przestrzeni nazw global environment jest pierwsze, a pakiet base ostatni.

```
search()
```

2. Wczytaj pakiet dplyr. Zwróć uwagę, że nowo wczytywane pakiety pojawiają się na 2 miejscu. Jeśli pakiet nie jest zainstalowany użyj install.packages(„dplyr”)

```
library(dplyr)
```

```
search()
```

3. Wywołaj funkcje sum dla wektora 1:5. Następnie zdefiniuj funkcje o tej samej nazwie, bez żadnych argumentów i kodu do wykonania. Zwróć uwagę, że przy próbie ponownego wywołania funkcji zostanie wywołana funkcja znajdująca się w global environment, a nie pakiecie base. Ponowne wywołanie funkcji powinno zwrócić błąd. W celu wywołania funkcji z pakietu base użyj składni nazwa_pakietu::funkcja.

```
sum(1:5)
```

```
sum <- function(){}
```

```
sum(1:5)
```

```
base::sum(1:5)
```

4. Usuń stworzoną funkcje poleceniem rm(sum), a następnie ponownie wywołaj funkcje. Czy została wywołana funkcja z pakietu base?

```
rm(sum)
```

```
sum(1:5)
```

5. Zdefiniuj funkcje z dwoma „free variable”: a oraz b. Następnie zdefiniuj wartości a i b oraz wywołaj funkcje.

```
y <- function(x) {
    a*x + b
}
a <- 1
b <- 0
y(1)
```

6. Zdefiniuj funkcje w poniższy sposób, a następnie wywołaj funkcje. Zwróć uwagę, że „free variable” są szukane w przestrzeni nazw, w której funkcja została zdefiniowana.

```
y <- function(x) {
```

```
      a <- 2
      b <- 1
      f <- function() {
            a*x + b
      }
      f()
}
a <- 1
b <- 0
y(1)
```

7. Zdefiniuj funkcje w poniższy sposób. Przed jej uruchomieniem zastanów się jaki będzie wynik.

```
a <- 1
b <- 2

m <- function(x){
      a <- 3
      b <- 4
      b*s(x)
}

s <- function(x){
      a+x
}
s(5)
```

## Wywoływanie funkcji na elementach wektorów, macierzy, list

1. Stwórz listę składającą się z wektora liczb od 1 do 10 oraz pierwszych 5 potęg liczby 2. Dla każdego elementu listy oblicz średnią.

```
l <- list(1:10, 2^(1:5))
lapply(l, mean)
```

2. Wykorzystując funkcje sapply ponownie policz średnią. Zwróć uwagę, że wynik został zwrócony w postaci wektora. Warto podkreślić, że wynikiem funkcji sapply w zależności od ilości elementów zwracanych przez wywoływaną funkcje może być wektor, macierz lub lista.

```
sapply(l, mean)
```

3. Stwórz funkcję, która podnosi x do potęgi n, a następnie wykorzystując stworzoną funkcję podnieś elementy wektora v do potęgi 5.

```
v <- 1:10

f <- function(x,n) {

     x^n

}

sapply(v, f, 5)
```

4. Wykorzystując funkcje tapply policz średnią dla parzystych i nieparzystych elementów wektora v.

```
v <- c(15,73,64,25,67,32,67,23,56,67)

g <- c(rep(1:2,length(v)/2))

tapply(v,g,mean)
```

5. Stwórz macierz o wartościach od 1 do 10 o 2 kolumnach i 5 wierszach. Następnie wykorzystując funkcje apply policz średnią liczb znajdujących się w każdym wierszu i kolumnie.

```
m <- matrix(1:10, nrow=5, ncol=2)

apply(m,1,mean)

apply(m,2,mean)
```

6. Wykorzystując funkcje mapply() podnieś kolejne elementy wektora v1 do potęgi zapisanej w wektorze v2.

```
v1 <- 1:5

v2 <- 5:1

mapply(f,v1,v2)
```