Wprowadzenie do języka R (cz. 2.)

1) Funkcje - pierwsza wzmianka

1. Tworzymy plik `s21.R` i wpisujemy w nim:

```
sqr <- function(x) x * x # składnia
add <- function(a, b) a + b
```

2. Uruchamiamy skrypt (np. klikamy w `Source`)

3. Sprawdzamy działanie funkcji, wpisując w konsoli RStudio:

```
> sqr(5)
> add(2,3)
```

4. ***Zadania***:

    1. Napisz i przetestuj funkcję obliczającą pole prostokąta o bokach `a` i `b`

    2. Napisz i przetestuj funkcję obliczającą pole powierzchni bocznej prostopadłościanu o bokach podstawy a i b i wysokości h

    3. Napisz i przetestuj funkcję obliczającą pierwiastki równania kwadratowego $ax^2 + bx + c = 0$ dla przypadku gdy $\Delta = b^2 - 4ac > 0$; jako typ wyniku przyjmij wektor dwuelementowy

    4. Zmodyfikuj funkcje `sqr` i `add` w następujący sposób:

        - dodaj nawiasy klamrowe otaczające ciało funkcji
        - dodaj `return`

    5. [*opcjonalne*] Opisz składnię wyrażenia służącego do tworzenia funkcji (`function(___) ___`)

2) Wyrażenie warunkowe `if ... [else]` oraz `switch`

1. W pliku `s21.R` dodajemy funkcję `myAbs` :

```
myAbs <- function(x) {
  if (x >= 0) x
  else -x
}
```

   i testujemy jej działanie

2. W pliku `s21.R` dodajemy funkcję `f1` :

```
f1 <- function(x, y, z) {
  if (z > 0) {
    if (x %% y == 0) {
```

```
      y %/% z
    }
    else if (x > y) {
      x %/% z + 3
    } else {
      z + 2
    }
  } else {
    stop("z cannot be negative")
  }
}
```

i testujemy jej działanie

3. W pliku `s21.R` dodajemy funkcję `leadingActor`:

```
leadingActor <- function(title) {
  switch (title,
          "Blue"  = "Juliette Binoche",
          "White" = "Zbigniew Zamachowski",
          "Red"   = "Irene Jacob",
          stop("Unknown title!")
          )
}
```

i testujemy jej działanie

4. ***Zadania***:

    1. Przepisz funkcję `leadingActor` zamieniając `switch` na `if ... else`
    2. Przepisz funkcję `f1` w następujący sposób:

```
f1 <- function(x, y, z) {
  stopifnot(z > 0) # sprawdź opis funkcji "stopifnot" (?stopifnot)
  ___
}
```

    3. [*opcjonalne*] Usuń w definicji funkcji `f1` wszystkie zbędne nawiasy klamrowe

3) Funkcje - domniemane wartości argumentów i zasięg nazw

1. Tworzymy plik `s22.R` i wpisujemy w nim:

```
f2 <- function(x, y, safeMode = FALSE) {
  if (TRUE == safeMode) {
    stopifnot(c(is.numeric(x), is.numeric(y)))
  }

  if (x * y > 0) {
    2 * x + y
```

```
  } else {
    2 * y + 7
  }
}
```

2. W konsoli RStudio testujemy działanie funkcji f2, np.

```
> f2(1,2)
> f2("a", "b")
> f2("a", 2, safeMode = TRUE)
```

3. W pliku `s22.R` pod funkcją `f2` dodajemy następujący kod:

```
# koniec definicji funkcji f2
x <- 5 # zmienna (globalna) zdefiniowana poza funkcją
f3 <- function(a,b) {
  x <- (a^3 - b^3) / (a + 2 * b) # czy tu nadpisujemy wartość x?
  y <- 7 * a^2 + 8 * (a^2 + 2 * b * a)
  x + y
}
```

4. ***Zadania***:

    1. Uzasadnij zapis warunku logicznego w wyrażeniu logicznym w funkcji `f2` w formie `TRUE == safeMode` (zamiast `safeMode == TRUE`)
    2. Dodaj w funkcji `f2` kolejny parametr o wartości domniemanej, np. `logging = FALSE` , który ustala, czy funkcja powinna, czy też nie powinna wypisywać na ekranie argumenty wywołania
    3. [*opcjonalne*] Wyjaśnij zasady działania mechanizmu przesłaniania nazw w funkcjach. Czy każdy blok w języku R tworzy nowy zakres?

4) Pętle: `while` , `repeat` i `for`

1. Tworzymy plik `s23.R` i wpisujemy w nim:

```
avg <- function(v) {
  stopifnot(c(is.vector(v), is.numeric(v)))

  sum <- 0
  i <- 0
  lengthOfV <- length(v)

  while (i <= lengthOfV) {
    sum <- sum + v[i]
    i <- i + 1
  }

  sum / lengthOfV
}
```

2. Zaznaczamy myszką całą funkcję i klikamy przycisk `Run`
3. Testujemy działanie funkcji, np.

```
> avg(1:5)  # ?
> avg(c(1,3)) # ?
```

4. Ustawiamy "pułapkę" klikając w obszarze edytora skryptów, po lewej stronie (numerów linii), w linii `sum <- 0` (pojawia się czerwony okrąg)
5. Analizujemy ostrzeżenie/komunikat, który pojawił się w górnej części edytora skryptów
6. Zgodnie z sugestią klikamy przycisk `Source` (okrąg zmienia się w kółko :)
7. W konsoli RStudio wpisujemy:

```
> avg(1:5) # naciskamy [ENTER] (wygląd okna zmienia się; zaczął działać "debugger")
```

8. W lewym dolnym panelu (konsoli RStudio) pojawiły się nowe przyciski, klikamy kilkukrotnie `Next` i obserwujemy zawartość panelu `Environment` - jaką wartość ma `sum` ?
9. Kontynuujemy klikanie w `Next` aż do końca pętli
10. Zmieniamy kod w następujący sposób:

```
avg <- function(v) {
  stopifnot(c(is.vector(v), is.numeric(v)))

  sum <- 0
  i <- 1
  lengthOfV <- length(v)

  while (i <= lengthOfV) {
    sum <- sum + v[i]
    i <- i + 1
  }

  sum / lengthOfV
}
```

11. W pliku `s23.R` dodajemy funkcję `avg2` :

```
avg2 <- function(v) {
  stopifnot(c(is.vector(v), is.numeric(v)))

  sum <- 0
  i <- 1
  lengthOfV <- length(v)
```

```
  repeat {
    sum <- sum + v[i]
    i <- i + 1

    if (i == lengthOfV) {
      return (sum / lengthOfV)
    }
  }
}
```

i w konsoli RStudio testujemy jej działanie, np.

```
> avg2(1:5) # ?
> avg2(1:5) == avg(1:5) # ?
```

12. Rozpoczynamy sesję "debuggowania"
13. Zmieniamy kod funkcji `avg2` na:

```
avg2 <- function(v) {
  stopifnot(c(is.vector(v), is.numeric(v)))

  sum <- 0
  i <- 1
  lengthOfV <- length(v)

  repeat {
    sum <- sum + v[i]

    if (i == lengthOfV) {
      return (sum / lengthOfV)
    }

    i <- i + 1
  }
}
```

14. W pliku `s23.R` dodajemy funkcję `avg2` :

```
avg3 <- function(v) {
  stopifnot(c(is.vector(v), is.numeric(v)))

  sum <- 0

  for (e in v) {
    sum <- sum + e
  }

  sum / length(v)
}
```

i testujemy jej działanie

15. W menu `Tools` wybieramy `Install Packages...` , a potem w polu edycyjnym `Packages` wpisujemy `microbenchmark` i naciskamy `Install`

16. W pliku `s23.R` dodajemy następujący kod:

```
library(microbenchmark)

testVec <- 1:1000

print(microbenchmark(
    aver_while = avg(testVec),
    aver_repeat = avg2(testVec),
    aver_for = avg3(testVec),
    aver_lib = mean(testVec) # funkcja biblioteczna
), signif = 4)
```

klikamy w przycisk `Source` i analizujemy wyświetlone wyniki

17. ***Zadania***:

    1. Wyjaśnij na czym polega błąd w pierwotnej wersji funkcji `avg2`
    2. Wyjaśnij powód tak dużej różnicy w czasie wykonania testowanych funkcji (`avg` , `avg2` , `avg3` i `mean`)
    3. Napisz i przetestuj funkcję `sDevVec` obliczającą odchylenie standardowe (dla) wektora `v`
    4. Napisz i przetestuj funkcję `medianVec` obliczającą medianę dla wektora `v`
    5. [*opcjonalne*] Napisz i przetestuj funkcję `modeVec` obliczającą modę/dominantę dla wektora `v`
    6. [*opcjonalne*] Napisz i przetestuj funkcję `pearsonCorCoef` obliczającą współczynnik korelacji liniowej Pearsona