MDCS R — ćwiczenia 1

17 styczeń 2014

1 Zrzut rzeczy które były na beamerze

```
# HW1, Z1

pesel <- c(4, 4, 0, 5, 1, 4, 0, 1, 3, 5, 8)
pesel10 <- c(4, 4, 0, 5, 1, 4, 0, 1, 3, 5)
pesel11 <- c(8)

pesel10 * c(1, 3, 7, 9, 1, 3, 7, 9, 1, 3)

##  [1]  4 12  0 45  1 12  0  9  3 15

sum(pesel10 * c(1, 3, 7, 9, 1, 3, 7, 9, 1, 3))

## [1] 101

10 - sum(pesel10 * c(1, 3, 7, 9, 1, 3, 7, 9, 1, 3))%%10

## [1] 9

pesel10 <- c(4, 4, 0, 5, 2, 4, 0, 1, 3, 5)
10 - sum(pesel10 * c(1, 3, 7, 9, 1, 3, 7, 9, 1, 3))%%10

## [1] 8

10 - sum(pesel10 * c(1, 3, 7, 9))%%10

## Warning: długość dłuszego obiektu nie jest wielokrotnością długości
krótszego obiektu

## [1] 8

10 - sum(c(pesel10, 0, 0) * c(1, 3, 7, 9))%%10

## [1] 8
```

```r
# Rozkład na cyfry
123%%(10^(3:1))%/%(10^(2:0))
```

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

```r
floor(log(12345, 10) + 1)
```

```
## [1] 5
```

```r
x <- 44051401358
x%%(10^(floor(log(x, 10) + 1):1))%/%(10^(floor(log(x, 10)):0))
```

```
##  [1] 4 4 0 5 1 4 0 1 3 5 8
```

```r
# HW1, Z2

jazdy <- c(52, 23, 20, 73, 1, 21, 21, 19, 278, 44, 38, 1, 101, 21, 44, 71, 43,
    13, 35, 76, 14, 67, 6, 75, 54, 93, 17, 84, 135, 16, 134, 126, 55, 87, 33,
    31, 4, 103, 11, 69, 61, 20, 25, 55, 9, 85, 12, 23, 50, 8, 7, 3, 61, 13,
    128, 12, 7, 49, 33, 7, 10, 48, 1, 20, 97, 19, 3, 24, 1, 41, 0, 37, 7, 28,
    14, 70, 174, 58, 12, 38, 16, 9, 33, 1, 40, 16, 25, 39, 5, 5, 24, 83, 33,
    59, 4, 13, 18, 113, 36, 14, 39, 43, 16, 70, 17, 16, 17, 1, 164, 12, 159,
    48, 19, 4, 8, 10, 70, 37, 11, 143, 43, 158, 61, 19, 41, 6, 22, 9, 44, 30,
    13, 33, 36, 14, 7, 11, 54, 14, 57, 23, 1, 101, 65, 7, 13, 11, 8, 23, 135,
    6, 36, 15, 4, 57, 45, 12, 54, 16, 13, 48, 46, 22, 32, 8, 149, 11, 13, 8,
    6, 16, 36, 104, 54, 107, 19, 153, 9, 24, 2, 3, 0, 56, 3, 37, 26, 11, 46,
    157, 11, 4, 49, 46, 10, 17, 49, 18, 13, 28, 50, 17)

sum(sort(jazdy, decreasing = TRUE) >= 1:length(jazdy))
```

```
## [1] 51
```

```r
sum(sort(jazdy) >= length(jazdy):1)
```

```
## [1] 51
```

```r
# HW1, Z3

n <- 10^(0:7)
(1 + 1/n)^n
```

```
## [1] 2.000 2.594 2.705 2.717 2.718 2.718 2.718 2.718
```

```r
(1 + 1/n)^n - exp(1)
```

```
## [1] -7.183e-01 -1.245e-01 -1.347e-02 -1.358e-03 -1.359e-04 -1.359e-05
## [7] -1.359e-06 -1.343e-07
```

```r
cumsum(1/factorial(0:20))
```

```
##  [1] 1.000 2.000 2.500 2.667 2.708 2.717 2.718 2.718 2.718 2.718 2.718
## [12] 2.718 2.718 2.718 2.718 2.718 2.718 2.718 2.718 2.718 2.718
```

```r
cumsum(1/factorial(0:20)) - exp(1)
```

```
##  [1] -1.718e+00 -7.183e-01 -2.183e-01 -5.162e-02 -9.948e-03 -1.615e-03
##  [7] -2.263e-04 -2.786e-05 -3.059e-06 -3.029e-07 -2.731e-08 -2.261e-09
## [13] -1.729e-10 -1.229e-11 -8.153e-13 -5.063e-14 -2.665e-15  0.000e+00
## [19]  0.000e+00  0.000e+00  0.000e+00
```

```r
# Funkcje
kwadrat <- function(x) {
    return(x^2)
}
kwadrat(2)
```

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

```r
kwadrat(7)
```

```
## [1] 49
```

```r
kwadrat
```

```
## function(x) {
##     return(x^2)
## }
```

```r
kwadrat <- function(x) {
    x^2
}
kwadrat <- function(x) x^2
kwadratPlus1 <- function(x) {
    x <- x + 1
    x^2
}
kwadratPlus1(6)
```

```
## [1] 49
```

```r
x <- 10
kwadratPlus1(x)
```

```
## [1] 121
```

```r
x
```

```
## [1] 10
```

```
y <- 7
kwadratPlusY <- function(x) {
    x <- x + y
    x^2
}
kwadratPlusY(0)
```

```
## [1] 49
```

```
kwadratPlusYPlus1 <- function(x) {
    y <- y + 1
    x <- x + y
    x^2
}
y
```

```
## [1] 7
```

```
kwadratPlusY(0)
```

```
## [1] 49
```

```
y
```

```
## [1] 7
```

```
kwadratPlus <- function(x, y) (x + y)^2
kwadratPlus(1, 2)
```

```
## [1] 9
```

```
kwadratPlus <- function(x, y = 4) (x + y)^2
kwadratPlus(1)
```

```
## [1] 25
```

```
kwadratPlus(1, 5)
```

```
## [1] 36
```

```
kwadratPlus(x = 1, y = 5)
```

```
## [1] 36
```

```
kwadratPlus(y = 1, x = 5)
```

```
## [1] 36
```

```r
kwadratDziel <- function(x, y = 4) (x/y)^2
kwadratPlus(y = 1, x = 5)
```

```
## [1] 36
```

```r
kwadratPlus(x = 1, y = 5)
```

```
## [1] 36
```

```r
kwadratDziel(x = 1, y = 5)
```

```
## [1] 0.04
```

```r
kwadratDziel(y = 1, x = 5)
```

```
## [1] 25
```

```r
kwadratDziel(y = 1)
```

```
## Error: brakuje 'x'
```

```r
leniwaFunkcja <- function(x) 3
leniwaFunkcja(224)
```

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

```r
kwadratDziel(y = 1)
```

```
## Error: brakuje 'x'
```

```r
leniwaFunkcja(kwadratDziel(y = 1))
```

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

```r
# Dygresja dzieleniu przez zero dla numerów rzeczywistych IEEE
1/0
```

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

```r
0/0
```

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

```r
1/0 + 3
```

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

```r
1/0 - 3
```

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

```r
1/0 - 1/0
```

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

```r
# Analityczne rozwiązanie równania kwadratowego ax^2+bx+c=0

kwPir <- function(a, b, c) {
    delta <- b^2 - 4 * a * c
    (-b + sqrt(delta) * c(-1, 1))/(2 * a)
}
kwPir(1, 0, -3)
```

```
## [1] -1.732  1.732
```

```r
kwPir(1, 0, -4)
```

```
## [1] -2  2
```

```r
# Listy

l <- list(1, 2, 3)
l
```

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

```r
l <- list(1, 2:5, "kotek", function(x) x^2)
l
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 2 3 4 5
##
## [[3]]
## [1] "kotek"
##
## [[4]]
## function (x)
## x^2
```

```r
# Sapply

function(x) x^2

## function(x) x^2

(function(x) x^2)(3)

## [1] 9

sapply(1:3, function(x) x^2)

## [1] 1 4 9

lapply(1:3, function(x) x^2)

## [[1]]
## [1] 1
##
## [[2]]
## [1] 4
##
## [[3]]
## [1] 9

lapply(1:3, function(x) 1:x)

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

sapply(lapply(1:3, function(x) 1:x), sum)

## [1] 1 3 6

sum(sapply(lapply(1:3, function(x) 1:x), sum))

## [1] 10
```

2 Co powinno się wiedzieć/umieć

- Umieć zrobić pierwszą pracę domową.
- Umieć zdefiniować funkcję.
- Wiedzieć że wartości przekazane funkcji ulegają skopiowaniu tak żeby można było nimi manipulować bez psucia oryginałów. Kopie te zostają wymazane gdy wykonanie funkcji się kończy.
- Wiedzieć kiedy można pominąć `return()` i {}.
- Wiedzieć jak definiować argumenty i argumenty z domyślnymi wartościami.
- Wiedzieć jak używać nazw argumentów do przeciążenia kolejności argumentów.
- Wiedzieć że w R funkcjonuje leniwe wartościowanie argumentów.
- Wiedzieć że funkcje w R są takimi samymi wartościami jak wektory i że nawet wbudowane funkcje to zwykłe zmienne zawierające kod. Rozumieć interpretację `()` jako operatora wykonania kodu.
- Umieć stworzyć i rozpoznać listę.
- Umieć zastosować `sapply` i `lapply`.