PROGRAMOVÁNÍ V R

- `if...else` podmínka
  - `ifelse`– alternativa
  - `Switch`
- `for` cyklus
- `apply` funkce
- funkce: `function() {}`

`if … else` podmínka

- `if (logická podmínka) {příkaz_1} else {příkaz_2}`
  - Pozn. složené závorky – pravý Alt + B/N; `{}`

- **logická podmínka** vrací **logickou hodnotu** `TRUE` nebo `FALSE`, tzn. je-li logická podmínka splněna, provede se `příkaz_1`, není-li logická podmínka splněna, je proveden `příkaz_2`

- POZOR! Příkaz `else` je volitelný, tzn. není nutný:
  - `if (logická podmínka) {příkaz_1}`
  - v případě, že logická podmínka vrátí hodnotu `FALSE` nebude proveden žádný příkaz

ifelse

- `if … else` bývá většinou součástí větších cyklů (zejména v případě programování)
  - ALE osamoceně pracuje s vektory délky 1
- `ifelse` umožňuje práci s vektory delšími než 1

```
> x <- c(0,1)
> if (x > 0) {"Kladné číslo"} else
+   if (x < 0) {"Záporné číslo"} else {"Nula"}
[1] "Nula"
Warning messages:
1: In if (x > 0) { :
  the condition has length > 1 and only the first element will be used
2: In if (x < 0) { :
  the condition has length > 1 and only the first element will be used
> x <- c(-1,0,1,5,-5)
> ifelse(x <= 0,"Podmínka splněna", "Podmínka nesplněna")
[1] "Podmínka splněna"   "Podmínka splněna"   "Podmínka nesplněna" "Podmínka nesplněna" "Podmínka splněna"
```

switch

- osamoceně pracuje s vektory délky 1

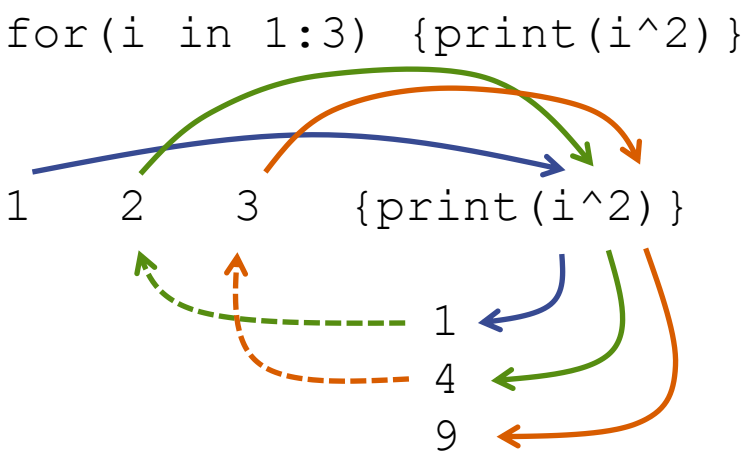
```
> switch(x, "child", "young adult", "adult")
Error in switch(x, "child", "young adult", "adult") :
  EXPR must be a length 1 vector
> switch(2, "child", "young adult", "adult")
[1] "young adult"
> switch(3, "child", "young adult", "adult")
[1] "adult"
> switch(4, "child", "young adult", "adult")
>
```

- v momentě, kdy je zadána hodnota, která neodpovídá textovému řetězci, nebude nic vypsáno

`for` cyklus

- v případě, že máme opakovat několik příkazů mnohokrát
- `for (objekt in vektor_hodnot) {příkaz 1}`
- Př.:

```
> for (i in 1:10) {print(i^2)}
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
[1] 49
[1] 64
[1] 81
[1] 100
> i
[1] 10
>
```



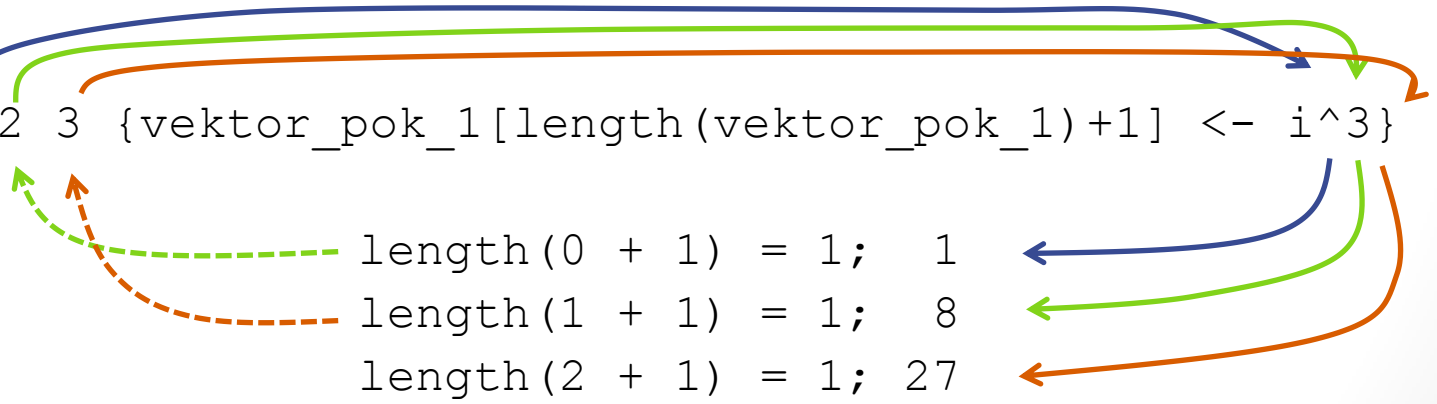
- objekt `i` v rámci `for` cyklu upravujeme, ale cyklus to neovlivní

Uložení výsledku `for` cyklu

- `vektor_pok_1 = c()`
- `for(i in 1:3)`
  `{vektor_pok_1[length(vektor_pok_1)+1] <- i}`
- `vektor_pok_1`

- `vektor_pok_1 = c()` <- délka vektoru je 0

- `1 2 3 {vektor_pok_1[length(vektor_pok_1)+1] <- i^3}`



`vektor_pok_1` <- délka vektoru je nyní 3

- **POZOR:**
- `for(i in 1:3) {vysledek[length(vysledek)+3] <- c(i^3,i-2,i)}`
- v tomto případě, se do jedné dimenze snažíme vložit tři hodnoty najednou, resp.:
- `vysledek[length(vysledek)+3] = 0 + 3 = 3`
- `vysledek[3] <- c(1, -1, 1)` což **NELZE**!

- *Alternativa:*
- `for(i in 1:3) {`
- `  vysledek[length(vysledek)+1] <- i^3`
- `  vysledek[length(vysledek)+1] <- i-2`
- `  vysledek[length(vysledek)+1] <- i`
- `  }`

- Nebo přímo „slučováním“ vektorů, bez `length()`:
- `vysledek_2 = c()`
- `for(i in 1:3){vysledek_2 = c(vysledek_2,i^3,i-2,i)}`

apply

- Funkce aplikuje funkci zadanou parametrem `FUN`

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

- Např.
- `apply(X = ma, MARGIN = 1, FUN = mean)`
- `apply(ma,1,mean)` průměr po řádcích
- `apply(ma,2,mean)` průměr po sloupcích

- Jako parametr `FUN` lze vkládat i námi vytvořené funkce

Funkce

- `nazev_funkce <- function (argument/y) {výraz/skript}`

- ```
  podil <- function(citatel, jmenovatel)
   { result <- citatel/jmenovatel
   return(result)
   }
  ```

- `return()` vrátí hodnotu/výstup funkce
- `print()` zobrazí hodnotu/výstup funkce
- Note: v případě více výsledků:
  - `return(list(objekt1, objekt2, … , objektN))`
  ```

- **EUKLIDOVSKÁ VZDÁLENOST DVOU BODŮ**

```
x1 <- c(0,0)
x2 <- c(1,2)
euc_dist <- function(x1, x2) {
  sqrt(sum((x1 - x2)^2))
  }
euc_dist(x1,x2)

A <- c(0,0,0)
B <- c(1,2,3)

euc_dist(A,B)
```