Programiranje v R-ju
2. Krmilni stavki

Vladimir Batagelj

Univerza v Ljubljani, FMF, Matematika

Finančna matematika
Ljubljana, februar 2009
15. oktober 2012

Kazalo

Sestavljeni izrazi

Več izrazov lahko združimo v en *sestavljeni* izraz (ali zaporedje izrazov) tako, da jih oklenemo z zavitimi oklepaji in ločimo s podpičji

{ $izraz_1$; $izraz_2$; . . . ; $izraz_k$ }

Izrazi v sestavljenem izrazu se izračunajo v istem vrstnem redu, kot stojijo v zaporedju. Vrednost sestavljenega izraza je enaka vrednosti zadnjega izraza v zaporedju.

```
> ({p <- 3; s <- 2*p; q <- s+5})
[1] 11
> {p <- 3; s <- 2*p; s+5}
[1] 11
```

Pogojni stavki

Krmilni stavki, `help(Control)`, so izrazi, ki nam omogočajo vejitve in ponavljanja v izračunih. *pogoj* je izraz, ki ima logično vrednost `TRUE` ali `FALSE`; *izraz* je enostaven ali sestavljeni izraz.

(Enovejni) pogojni stavek

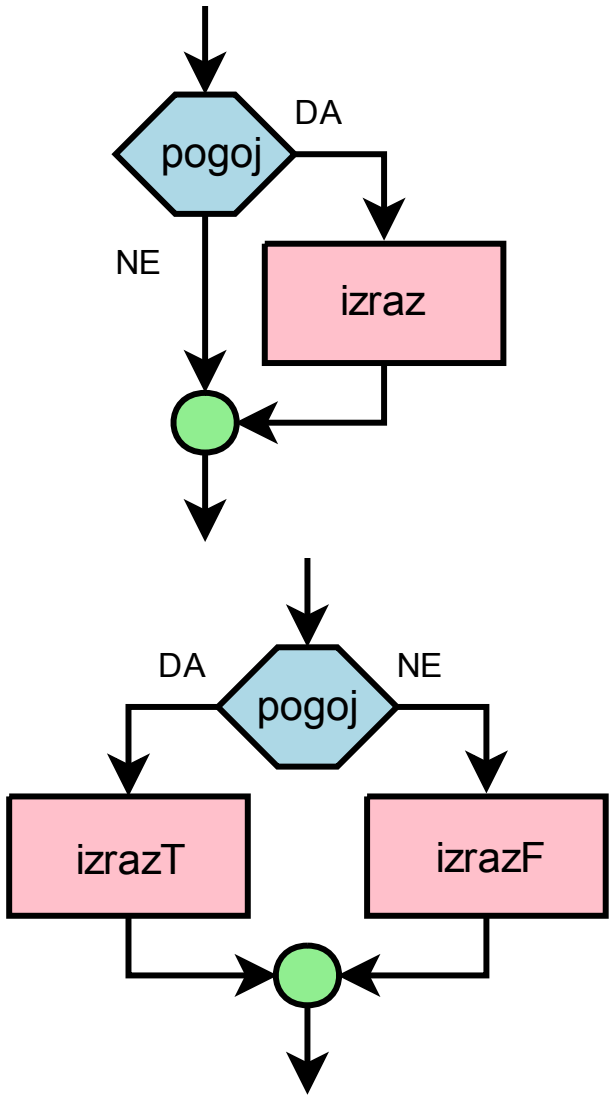
`if(`*pogoj*`)` *izraz*

če je *pogoj* resničen, izračuna in vrne vrednost *izraz*a; sicer vrne vrednost `NULL`.

Razvejitveni pogojni stavek

`if(`*pogoj*`)` *izrazT* `else` *izrazF*

če je *pogoj* resničen, izračuna in vrne vrednost *izrazT* a; sicer vrne izračuna in vrne vrednost *izrazF* a.



Zanke

Zanke omogočajo večkratni izračun nekega izraza.

Zanka `while`

`while(`*pogoj*`)` *izraz*

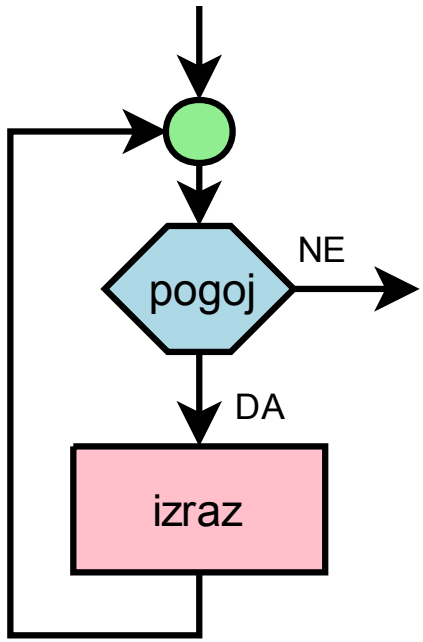
ponavlja naslednje: če je *pogoj* resničen izračuna *izraz*; sicer ponavljanje prekine in nadaljuje izračun na naslednjem izrazu.

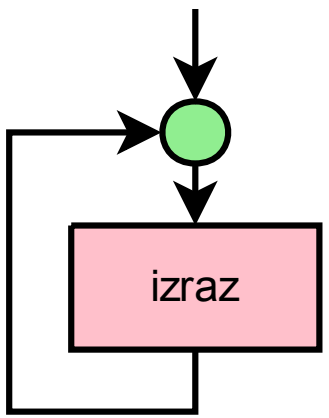
Zanka `repeat`

`repeat` *izraz*

ponavlja izračun *izraz*a, dokler v samem *izraz*u ni zahtevana prekinitev.

Vrednost zanke je vrednost zadnjega v njej izračunanega izraza.





Prehodi

Izstop iz zanke

`break`

zahteva izstop iz (prve) zanke, ki oklepa ta stavek.

Začetek zanke

`next`

zahteva nadaljevanje na začetku (prve) zanke, ki oklepa ta stavek.

Izraza `break` in `next` vrneta vrednost `NULL`.

Izvajanje programa lahko prekinemo tudi z ukazom `stop(`*sporočilo*`)`.

```
> n <- 0; repeat{n <- n+1; if(n>10) stop("Konec"); cat(n,',',sep='')}
1,2,3,4,5,6,7,8,9,10,Error: Konec
```

Zapis programov v R-ju

V krmilnih stavkih so oklepaji okrog pogoja **obvezni**!!!
V razvejitvenem pogojnem stavku mora biti `else` v isti vrstici kot zaključek *izrazT* a.

```
if(pogoj) {
   izraziT
} else {
   izraziF
}
```

Kadar so v krmilnih stavkih podrejeni izrazi sestavljeni, njihovo podrejenost v zapisu nakažemo z *zamikanjem*.

Krmilni stavki – primeri


```
> a <- 3
> (if (a > 2) "dober")
[1] "dober"
> a <- 0
> (if (a > 2) "dober")
NULL
> a <- 3
> (b <- if (a<0) "N" else "P")
[1] "P"
> a <- -3
> (b <- if (a<0) "N" else "P")
[1] "N"
```

```
> q <- 1; adelta <- 1
> while (q < 1000){
+   p <- round(pi*q)
+   delta <- abs(p/q-pi)
+   if (delta < adelta){
+     adelta <- delta
+     ap <- p; aq <- q
+     cat(p,"/",q,"=",p/q,
+       " d=",delta,"\n",sep="")
+   }
+   q <- q+1
+ }
3/1=3 d=0.1415927
13/4=3.25 d=0.1084073
16/5=3.2 d=0.05840735
19/6=3.166667 d=0.02507401
22/7=3.142857 d=0.001264489
179/57=3.140351 d=0.001241776
201/64=3.140625 d=0.0009676536
223/71=3.140845 d=0.0007475832
245/78=3.141026 d=0.0005670126
267/85=3.141176 d=0.000416183
289/92=3.141304 d=0.0002883058
311/99=3.141414 d=0.0001785122
333/106=3.141509 d=8.321963e-05
355/113=3.141593 d=2.667642e-07
```

Funkcije

Izraz oblike

`function(`*argumenti*`)` *telo*

ustvari funcijo. Če jo priredimo nekemu imenu, jo z njim poimenujemo. *argumenti* vsebujejo seznam argumentov funkcije ločenih z vejico. Posamezni argument ima ali obliko *ime* ali *ime* = *izraz*. Argument oblike *ime* = *izraz* določa privzeto vrednost, ki velja, če tega argumenta pri klicu funkcije ne navedemo. *telo* je (sestavljeni) izraz, ki določa, kako izračunamo vrednost funkcije. V telesu lahko izračun zaključimo z izrazom `return(`*vrednost*`)`, ki zahteva prekinitev izračuna funkcije in vrne *vrednost* kot vrednost funkcije.

Funkcijo uporabimo z izrazom oblike *fun*`(`*dejanski_argumenti*`)`, kjer je *fun* funkcija ali njeno ime, *dejanski_argumenti* pa določajo vrednosti argumentov: imenskim argumentom ustrezajo istoležni dejanski argumenti; morebitni argumenti oblike *ime* = *izraz* jim slede v poljubnem vrstnem redu. Pri uporabah funkcij v telesih drugih funkcij lahko uporabimo na repu argumentov tudi `...`, kar pomeni, da funkcija 'podeduje' imenovane argumente nadrejene funkcije.

```
args, body, formals, environment, alist, debug, invisible
help("function")
```

Funkcije – primeri


```
> (function(x) x^2-x+41)(3)
[1] 47
> ascii <- function(char) {
+ a <- as.integer(charToRaw(char))
+ if (length(a)>1) NA else a[1] }
> ascii("A")
[1] 65
> ascii("a")
[1] 97
> ascii("\u263A")
[1] NA
> gcd <- function(a,b)
+ if (b==0) abs(a) else gcd(b,a%%b)
> gcd(12,21)
[1] 3
> gcd(624,918)
[1] 6
> "%m%" <- function(a,b) min(a,b)
> "%M%" <- function(a,b) max(a,b)
> 4 %m% 3 %M% 5
[1] 5
```

Funkcije

Funkcije imajo v programiranju zelo pomembno vlogo. Omogočajo:

- "izpostavljanje" ponavljajočih se delov programa: tak del zapakiramo v funkcijo, ustrezne dele programa pa nadomestimo s klici funkcije;
- ustvarjanje ravni abstrakcije: za izbrano tematiko definiramo ustrezne objekte in sestavimo funkcije za delo z njimi; shranimo jih v posebno knjižnico. Odslej lahko razmišljamo naj tej ravni – pristop črnih škatelj.
- povečanje berljivosti in razumljivosti programov

R ima zelo bogato zbirko knjižnic. Za uporabo moramo knjižnico najprej namestiti na svoj računalnik `Packages/Install Package(s)...`. Pred vsako uporabo pa jo še vključimo z ukazom `library(`*kn*`)` ali `require(`*kn*`)`.

Za preprostejše uporabe lahko funkcije shranimo na znakovni datoteki *fun*`.R`, ki jo, ko jo potrebujemo, vnesemo z ukazom `source(`*fun*`.R)`.

Ugibanje števil


```
# ugibanje števil
ugani <- function(m=50){
  a <- 1+trunc(m*runif(1))
  s <- 0
  cat("Ugani število med 1 in",m,"\n")
  repeat{
    s <- s+1
    g <- as.integer(readline(paste(s,". ugibek = ",sep="")))
    if(a < g) {cat("manjše\n"); next}
    if(a > g) {cat("večje\n"); next}
    break
  }
  cat("število",a,"si uganil v",s,"poskusih\n")
}
```

Funkcijo shranimo na datoteko `ugani.R`.

```
> source("C:\\Users\\Batagelj\\test\\R\\ugani.R")
> ugani()
Ugani število med 1 in 50
1. ugibek = 25
manjše
2. ugibek = 13
manjše
3. ugibek = 6
večje
4. ugibek = 10
manjše
5. ugibek = 8
manjše
6. ugibek = 7
število 7 si uganil v 6 poskusih
```

Vidnost spremenljivk


```
> test1 <- function(a){
+   cat('b1=',b); b <- 3; cat('   b2=',b,'\n')
+   cat('a1=',a); a <- 7; cat('   a2=',a,'\n') }
> test2 <- function(a){
+   cat('b1=',b); b <<- 3; cat('   b2=',b,'\n')
+   cat('a1=',a); a <<- 7; cat('   a2=',a,'\n') }
> b <- 5; a <- 1; cat('a=',a,'  b=',b,'\n')
a= 1   b= 5
> test1(9)
b1= 5   b2= 3
a1= 9   a2= 7
> cat('a=',a,'  b=',b,'\n')
a= 1   b= 5
> test2(9)
b1= 5   b2= 3
a1= 9   a2= 9
> cat('a=',a,'  b=',b,'\n')
a= 7   b= 3
```

Pri klicu funkcije se ustvari klicu pripadajoče okolje, v katerem se ustvarijo spremenljivke za vsak parameter in spremenljivke na levi strani prireditvenih stavkov iz telesa funkcije. Parametrske spremenljivke dobijo vrednosti ustreznih dejanskih parametrov.

Ob izteku funkcije to okolje izgine. Če R ne najde dane spremenljivke v okolju funkcije, jo poišče po verigi v nadrejenih okoljih. S prireditvijo <<- priredimo vrednost spremenljivki zunaj okolja funkcije.