Cvetana Krstev

Programiranje – deo 2

Kontrolne strukture

Za pisanje smislenih i korisnih programa svaki programski jezik je snabdeven *kontrolnim strukturama* koje su sintaksička forma pomoću koje se iskazuje tok kontrole programa. Kao i u slučaju tipova podataka i operacija i relacija nad njima čiji je repertoar sličan u većini programskih jezika a način zapisivanja i korišćenja se može razlikovati, i većina programskih jezika poseduje slične kontrolne strukture čija sintaksa se može razlikovati od jednog do drugog jezika. Uobičajene kontrolne strukture su: **if-then-else**, **while-do**, **repeat-until** i **case**.

Razgranate strukture

if-then-else iskaz je bazični uslovni konstrukt u programskom jeziku koji dozvoljava izbor alternative – u zavisnosti od istinitosti datog uslova koji se zadaje iza ključne reči **if**, a u obliku logičkog ili relacijskog izraza. U sledećem primeru, posle izvršenja **if-then-else** iskaza promenljiva *k* će imati veću od dve vrednosti: one sadržane u promenljivoj *m* ili one sadržane u promenljivoj *n*.

VBA

```
Dim m As Integer, n As Integer, k As Integer
m = 5
n = 7
If m < n Then      ' true
    k = n          ' k <- 7
Else
    k = m          ' ne izvršava se!
End If
MsgBox k
```

Python

```
m = 5
n = 7
if m < n:      # true
    k = n       # k <- 7
else :
    k = m       # ne izvršava se!
print (k)
```

Opšti oblik razgranate strukture **if-then-else** je sledeći:

VBA

```
If uslov1 Then
    iskaz 'If-blok iskaza, izvršava se samo ako je uslov1=true
    iskaz
    …
    iskaz
ElseIf uslov2 Then
    iskaz ' Else If-blok iskaza, izvršava se samo ako je
    iskaz ' uslov1=false i uslov2=true
    …
    iskaz
ElseIf uslov3 Then
    iskaz ' Else If-blok iskaza, izvršava se samo ako je
    iskaz ' uslov1=false i uslov2=false i uslov3=true
    …
    iskaz
…
Else
    iskaz ' Else-blok iskaza, izvršava se samo ako su
    iskaz ' svi prethodni uslovi false
    …
    iskaz
End If
'uslov1, uslov2,... su logički izrazi čije su vrednosti True ili
False
```

Python

```
if uslov1:
   iskazi1
elif uslov2:
   iskazi2
elif uslov3:
   iskazi3
…
else:
  iskazi
```

Ova kontrolna struktura se izvršava na sledeći način: uvek se izvršava samo jedan blok iskaza, a posle izvršenja tog bloka se prelazi na izvršavanje prvog iskaza iza kontrolne strukture **If-Then-Else**. Ako je *uslov1* zadovoljen, onda se izvršava **if**-blok iskaza (*iskazi1*); ako *uslov1* nije zadovoljen, a *uslov2* jeste onda se izvršava prvi **elif** blok iskaza (*iskazi2*); ako *uslov1* i *uslov2* nisu zadovoljeni, a *uslov3* jeste onda se izvršava drugi **elif** blok iskaza *iskazi3*), i tako dalje; **Else**-blok iskaza se izvršava samo ako nijedan od uslova *uslov1*, *uslov2*, *uslov3*,... nije zadovoljen (*iskazi*).

Primer 1. Date su tri celobrojne vrednosti. Poređati ih od najmanje do najveće.
Prvo se na prvu poziciju smešta najmanja, a onda se na drugu poziciju smešta srednja vrednost – na trećoj poziciji nužno ostaje najveća vrednost.

VBA

```
Dim a As Integer, b As Integer, c As Integer, _
    p As Integer
```

	Dim s **As String**
	  a = 9
	  b = 6
	  c = 8
	  If b < a **Then** ' jeste
	  p = b ' p <- 6
	  b = a ' b <- 9
	  a = p ' a <- 6; a=6, b=9, c=8
	  End If
	  If c < a **Then** ' nije
	  p = c '
	  c = a '
	  a = p '
	  End If ' sada je u a najmanja vrednost
	  If c < b **Then** ' jeste
	  p = c ' p <- 8
	  c = b ' c <- 9
	  b = p ' b <- 8; a=6, b=8, c=9
	  End If
	  MsgBox a & " " & b & " " & c

Python	a = 9
	b = 6
	c = 8
	i**f** b < a : # jeste
	p = b # p <- 6
	b = a # b <- 9
	a = p # a <- 6; a=6, b=9, c=8
	if c < a : # nije (ovaj blok se ne
	p = c # izvršava)
	c = a # sada je u a najmanja vrednost
	a = p
	if c < b : # jeste
	p = c # p <- 8
	c = b # c <- 9
	b = p # b <- 8; a=6, b=8, c=9
	print (a, b, c)

Važna napomena: u pitanju su *tri odvojena* **if-then-else**, a ne jedan iskaz sa tri uslova. Zašto je to bitno? U principu se svaki od blokova u ova tri iskaza može izvršiti (preraspoređivanje vrednosti), a to ne bi bilo moguće sa jednim **if-then-else** iskazom.

Ako bismo želeli da program radi za bilo koje tri vrednosti *a*, *b* i *c*, morali bismo da dodamo i učitavanje.

VBA	**Dim** a **As Integer**, b **As Integer**, c **As Integer**, _
	  p **As Integer**
	Dim s **As String**
	  s = **InputBo**x("Unesi prvi broj")
	  a = s
	  s = **InputBox**("Unesi drugi broj")

```
b = s
s = InputBox("Unesi treci broj")
c = s
If b < a Then       '
    p = b           '
    b = a           '
    a = p           '
End If              ' a sadrzi manju od prve dve
If c < a Then       '
    p = c           '
    c = a           '
    a = p           '
End If              ' a sadrzi najmanju od tri
If c < b Then       '
    p = c           '
    c = b           '
    b = p           '
End If        ' b sadrzi manju od preostale dve
MsgBox a & " " & b & " " & c
```

Python

```
a = int(input())
b = int(input())
c = int(input())
if b < a :            #   postavlja manji od prva dva
   p = b              #   na prvo mesto
   b = a              #
   a = p              #
if c < a :            #   postavlja najmanji na prvo
   p = c              #   mesto
   c = a
   a = p
if c < b :            #   postavlja srednji na drugo
   p = c              #   mesto
   c = b              #
   b = p              #
print (a, b, c)
```

☺

Iskaz *case* je uslovni konstrukt u programskom jeziku koji dozvoljava izbor jedne od više mogućnosti — u zavisnosti od vrednosti datog izraza. On predstavlja uopštenje **if-then-else** iskaza, jer se za svaku moguću vrednost izraza koji se testira, može primeniti samo jedna od opcija. Sa određenim sintaksičkim varijacijama, javlja se u većini programskih jezika, ali ne postoji u programskom jeziku Python. U programskom jeziku VBA ovaj iskaz ima sledeći opšti oblik:

```
Select Case TestIzraz
    Case SpisakSlučajeva1
        iskazi
    Case SpisakSlučajeva2
        iskazi
    Case SpisakSlučajeva3
```

```
        iskazi
    ...
    Case Else
        iskazi 'default
End Select
```

U gornjem iskazu `TestIzraz` je celobrojnog ili tekstuelnog tipa (`Integer, Long, String`). `SpisakSlučajeva` je niz:

- vrednosti (istog tipa kao i `TestIzraz),`
- opsega vrednosti (npr. `1 To 12`) ili
- uslova (npr. `Is > 10`)

međusobno razdvojenih zapetom. Na primer,

```
Case 4, 20 To 30, Is > 100
```

obuhvata celobrojne vrednosti 4, 20, 21, 22,...30 i vrednosti veće od 100.

Primer 2. Sledeći programski kôd određuje broj dana u mesecu čiji redni broj je vrednost promenljive *m*. Posle izvršenja ovog koda, promenljiva *BrDana* sadrži broj dana u tom mesecu. U februaru broj dana zavisi, osim od meseca, i od godine koja je vrednost promenljive *g*. Za sve vrednosti izraza koje nisu eksplicitno pomenute i za koje nije pripremljen programski kôd, može se predvideti kôd (u programskom jeziku VBA označen kao **Case Else**) koji se u tim slučajevima izvršava. U programskom jeziku Python Case iskaz se simulira (zapisuje) korišćenjem **If-Then-Else** iskaza.

VBA

```
Sub BrDanaUMesecu()
Dim m, g, BrDana As Integer
' m je dati mesec
' g je data godina
' BrDana je broj dana u mesecu m godine g
g = InputBox("Godina je:")
m = InputBox("Mesec je:")
Select Case m
    Case 1, 3, 5, 7, 8, 10, 12
        BrDana = 31
    Case 4, 6, 9, 11
        BrDana = 30
    Case 2
        If (g Mod 400 = 0) _
        Or _
        ((g Mod 4 = 0) And (g Mod 100 <> 0)) Then
            BrDana = 29
        Else
            BrDana = 28
        End If
```

	Case Else BrDana = 0 **MsgBox** "Nepostojeci mesec " & m **End Select** **MsgBox** "Ovaj mesec ima " & BrDana & " dana" **End Sub**

Python	# m je dati mesec
	# g je data godina
	# BrDana je broj dana u mesecu m godine g
	g = int(input())
	m = int(input())
	if m in [1, 3, 5, 7, 8, 10, 12]:
	BrDana = 31
	elif m in [4, 6, 9, 11]:
	BrDana = 30
	elif m == 2:
	if (g % 400 == 0) or ((g % 4 == 0) and (g % 100 !=0)):
	BrDana = 29
	else:
	BrDana = 28
	else:
	BrDana = 0
	print ("Nepostojeci mesec")
	print ("Ovaj mesec ima " + str(BrDana) + "dana")

☺

Cikličke strukture

Petlje su programske kontrolne strukture koje čine sekvencije instrukcija koje se ponavljaju dok se ne zadovolji propisani uslov, kao što je saglasnost određenih podataka ili završetak odbrojavanja. U petlje spadaju **while**, **do-until** i **for** konstrukti.

Iskaz **while** je oblik programske petlje u kome se uslov za nastavak ponavljanja izračunava pri svakom prolasku kroz petlju. Ova petlja spada u takozvane *nulte petlje* jer telo petlje uopšte neće biti izvršeno ukoliko je uslov lažan, to jest nije zadovoljen, pri prvom prolasku. Ovaj oblik petlje se nalazi u većini programskih jezika, ali sa dosta sintaksičkih varijacija. U programskim jezicima ovaj iskaz ima sledeći opšti oblik:

VBA	**While** uslov iskazi **Wend**
Python	while uslov: iskazi

Njegovo značenje je: *Sve dok je ispunjen* `uslov` *ponavljaj* `iskaze`. Izračunavanje najvećeg zajedničkog delioca dva cela broja bismo na sledeći način zapisali u programskim jezicima korišćenjem **while** iskaza.

VBA

```
Dim m As Integer, n As Integer, r As Integer
      m = InputBox("prvi broj je")
      n = InputBox("drugi broj je")
      r = m Mod n
      While r > 0
            m = n
            n = r
            r = m Mod n
      Wend
      MsgBox "Najveci zajednicki delilac je " & n
```

Python

```
# m prvi broj
# n drugi broj
m = int(input())
n = int(input())
r = m % n
while r != 0:
    m = n
    n = r
    r = m % n
print "Najveci zajednicki delilac je ", n
```

Primer: *m*=252, *n*=105

m	252	105	42
n	105	42	21
r	42	21	0
r != 0?	true	true	false

```
Najveci zajednicki delilac je 21
```

☺

Primer 3. Dat je ceo broj za koji unapred ne znamo koliko ima cifara. Napisati program koristeći *while* iskaz koji izračunava zbir svih cifara broja (koriste se samo aritmetičke operacije).

VBA

```
Dim m As Integer, d As Integer, c As Integer
' m je dati ceo broj
' d je pomocna promenljiva, kolicnik pri deljenju
' c je izdvojena cifra
m = InputBox("Zadaj neki broj")
c = m Mod 10
d = m \ 10
MsgBox c    ' ispisi cifru c (najmanje tezine)
While d <> 0
    c = d Mod 10
    d = d \ 10
    MsgBox c    ' sledeca cifra
Wend
```

Python

```
# m je dati ceo broj
# d je pomocna promenljiva, kolicnik pri deljenju
# c je izdvojena cifra
# i zbir cifara broja
print ("Unesi neki ceo broj")
m = int(input())
c = m % 10
d = m // 10
i = c
while d != 0:
    c = d % 10
    d = d // 10
    i = i + c
print ("Zbir cifara broja ", str(m) , " je ", str(i))
```

Sledećom tabelom je prikazano kako teče izvršavanje ovog programa ako je na početku učitan broj 2734.

Primer: 2734

c	4	3	7	2
d	273	27	2	0
i	4	7	14	16
d!=0?	true	true	true	false

```
Zbir cifara broja 2734 je 16
```

☺

I iskaz **do-until** je oblik programske petlje u kome se uslov za završetak ponavljanja izračunava pri svakom prolasku kroz petlju, ali za razliku od **while** petlje telo ove petlje se izvršava najmanje jedanput. Ovo je s toga što se kod **do-until** petlje tačnost uslova proverava na kraju tela petlje, a ne na početku kao kod **while** petlje. Ova vrsta petlje se ređe koristi i ne postoji u svim programskim jezicima (U programskom jeziku Python ne postoji). U programskom jeziku VBA ovaj iskaz ima sledeći opšti oblik:

```
Do
     iskazi
Loop Until uslov
```

Njegovo značenje je: *Ponavljaj* `iskaze` *Sve dok ne bude ispunjen* `uslov`. Znači, izvršavanje tela petlje se ponavlja sve dok je `uslov` lažan (tj. nije ispunjen). Izračunavanje najvećeg zajedničkog delioca dva cela broja bismo na sledeći način zapisali u programskom jeziku VBA korišćenjem **do-until** iskaza, i simulirali u programskom jeziku Python:

VBA

```
Dim m As Integer, n As Integer, r As Integer
m = InputBox("prvi broj je")
n = InputBox("drugi broj je")
    Do
        r = m Mod n
        m = n
        n = r
    Loop Until r = 0
MsgBox "Najveci zajednicki delilac je " & m
```

Python

```
# m je prvi broj
# n je drugi broj
print("Unesi prvi broj")
m = int(input())
print("Unesi drugi broj")
n = int(input())
while True:
    r = m % n
    m = n
    n = r
    if r == 0:
        break
print ("Najveci zajednicki delilac je ", str(m))
```

Primer: *m*=252, *n*=105

r	42	21	0
m	105	42	21
n	42	21	0
r = 0?	false	false	true

```
Najveci zajednicki delilac je 21
```

Primer 4. Dat je ceo broj za koji unapred ne znamo koliko ima cifara. Napisati program koristeći **do-until** iskaz koji izračunava zbir svih cifara broja (koriste se samo aritmetičke operacije).

VBA

```
Sub CifreDoUntil()
Dim m As Integer, c As Integer
' m je dati ceo broj
' c je izdvojena cifra
m = InputBox("Zadaj neki broj")
Do
    c = m Mod 10
    MsgBox c    ' sledeca cifra
    m = m \ 10
Loop Until m = 0
```

Python

```
# m je dati ceo broj
# d je pomocna promenljiva, kolicnik pri deljenju
# c je izdvojena cifra
# i je zbir cifara
print ("Unesi neki ceo broj")
```

```
m = int(input())
d = m
i = 0
while True:
    c = d % 10
    d = d // 10
    i = i + c
    if d == 0:
        break
print ("Zbir cifara broja ", str(m) , " je ", str(i))
```

Sledećom tabelom je prikazano kako teče izvršavanje ovog programa ako je na početku učitan broj 51423.

Primer: 51423

c		3	2	4	1	5
d	51423	5142	514	51	5	0
i	0	3	5	9	10	15
d==0?		false	false	false	false	true

Program ispisuje:

```
Zbir cifara broja 51423 je 15
```

☺

Petlja **do** je programska petlja sa prebrojavanjem, u kojoj se sekcija koda izvršava sa ponavljanjem, pri čemu brojač uzima sukcesivne vrednosti. Uslov za završetak ponavljanja ovakve petlje se ostvaruje kada vrednost promenljive-brojača dostigne neku postavljenu vrednost. Tekuća vrednost promenljive-brojača se često koristi unutar petlje. U programskim jezicima VBA i Python ovaj iskaz ima sledeći opšti oblik:

VBA	`For brojač = početna To krajnja [Step korak]`
	`    iskaz 'For-blok iskaza`
	`    …`
	`    iskaz`
	`Next brojač`

	Ponavljaj `iskaze` *redom za sve vrednosti* `brojača` *od* `početne` *do* `krajnje` *koje se razlikuju za* `korak`.
Python	`for brojač in range([početna,] krajnja [,korak]):` `        iskazi`
	Ponavljaj `iskaze` *redom za sve vrednosti* `brojača` *od* `početne` (ili 0 ako nije zadata) *do* `krajnje` *koje se razlikuju za* `korak` (ili se razlikuju za 1 ako nije zadata).

Primer 5. Sledeći program izračunava funkciju faktorijel za prvih 12 celih brojeva. Funkcija faktorijel se definiše kao $i! = 1*2*3*\ldots*(i-1)*i$.

VBA	

```
Dim i As Integer, f As Long
' i je pomocna promenljiva, brojac petlje
' f je izracunati faktorijel
f = 1
For i = 1 To 12
    f = f * i                        ' f sadrzi i!
    MsgBox "Faktorijel od " & i & " je " & f
Next i
```

Python	

```
# i je brojač petlje
# f izračunati faktorijel
f = 1
for i in range(12):
    f = f * (i+1)
    print("Faktorijel od " + str(i+1) + " je " + str(f))
```

Program će izračunati i prikazati sledeće vrednosti:

```
i     f
1     1
2     2
3     6
4     24
5     120
6     720
7     5040
8     40320
9     362880
10    3628800
11    39916800
12    479001600
13    6227020800
```

☺

VBA: Na osnovu ovako izračunatih brojeva je jasno zašto je promenljiva *f* u kojoj je izračunata vrednost funkcije faktorijel deklarisana kao `Long` (4 bajta) i zašto se računa samo faktorijel do broja 12. Faktorijel broja 13 je broj koji je veći od 2^{32} pa se ne može zapisati u 4 bajta. U Python-u veličina celog broja nije ograničenje, čim vrednost broja prevaziđe najveću vrednost koja se može zapisati u 4 bajta, Python prelazi na **long integer** i tu nema ograničenja.

Zadatak. . (a) Šta radi sledeći deo programa (ispisati šta program ispisuje):

VBA	

```
i = 1
s = 0
While i <= 6
     s = s + i*i
```

	MsgBox "za i=" & i " s=" & s i = i + 1 **Wend**
Python	i = 1 s = 0 while i <= 6 : s = s + i*i print("Za i=" + str(i) + " s=" + str(s)) i = i+1

i	s	i <= 6	s=s+i*i	ispis
1	0	true	1	za i=1 s=1
2	1	true	5	za i=2 s=5
3	5	true	14	za i=3 s=14
4	14	true	30	za i=4 s=30
5	30	true	55	za i=5 s=55
6	55	true	91	za i=6 s=91

Program ispisuje zbir (*s*) kvadarata (*i*i*) prirodnih brojeva od 1 do 6.

(b) Zapisati u deo programa koji ima isti učinak kao deo pod (a), a koristi **for**-iskaz:

VBA	s = 0 **For** i = 1 **To** 6 s = s + i * i MsgBox "za i=" & i " s=" & s **Next** i
Python	s = 0 for i in range(1,7): s = s + i * i print("Za i=" + str(i) + " s=" + str(s))

(c) Zapisati u deo programa koji ima isti učinak kao deo pod (a), a koristi **Do-Loop Until** (**repeat-until**) iskaz:

VBA	**i = 1** **s = 0** **Do** **s = s + i*i** MsgBox "za i=" & i " s=" & s i = i + 1 **Loop Until** i > 6
Python	s = 0 i = 1 while True: s = s + i*i

```
    print("Za i=" + str(i) + " se=" + str(s))
    i = i + 1
    if i > 6:
        break
```

Primer 6. Pronalažnje *n*-tog Fibonačijevog broja. Fibonačijev niz se definiše na sledeći način:

$$f_1 = 1, f_2 = 1, f_n = f_{n-1} + f_{n-2}, n > 2$$

VBA

```
Dim i As Integer, n As Integer, f As Integer, _
    f1 As Integer, f2 As Integer
' n je dati ceo broj (n > 0)
' i je pomocna promenljiva, brojac petlje
' f je izracunati Fibonacijev broj
' f1 je prethodni izracunati Fibonacijev broj
' f2 je pred-prethodni izracunati Fibonacijev broj
n = InputBox("Koji Fibonacijev broj zelis da izracunas")
If n <= 2 Then
    f = 1
Else
    f1 = 1
    f2 = 1
      For i = 3 To n
            f = f1 + f2 ' f sadrzi i-ti Fibonacijev broj
            f2 = f1
            f1 = f
      Next i
End If
MsgBox n & "-ti Fibonacijev broj je " & f
```

Python

```
# n redni broj Fibonačijevog broja
# f je n-ti Fibonačijev broj
# i brojač petlje
# f1 prethodno izračunat Fibonačijev broj
# f2 Fibonačijev broj koji prethodi prethodno
izračunatom
print("Koji Fibonačijev broj želiš da izračunaš?")
n = int(input())
if n <= 2:
    f = 1
else:
    f1 = 1
    f2 = 1
    for i in range(2,n):
        f = f1+f2
        f2 = f1
        f1 = f
print(str(n) + ". Fibonačijev broj je " + str(f))
```

Sledećom tabelom je prikazano kako teče izvršavanje ovog programa ako se traži izračunavanje osmog Fibonačijevog broja.

Primer: n←8
n <=2 ? Nije

i		2	3	4	5	6	7
f		2	3	5	8	13	**21**
f2	1	1	2	3	5	8	13
f1	1	2	3	5	8	13	21

8. Fibonačijev broj je 21
☺

Procedure

Komponente koje se razvijaju u toku profinjavanja algoritama su često nezavisne od glavnog algoritma u smislu da se mogu dizajnirati nezavisno od konteksta u kome će se koristiti. Takvu komponentu može da razvija drugi programer, a ne onaj isti koji je dizajnirao i glavni algoritam, a ona se takođe može uključiti i kao komponenta nekog drugog programa. Takav je, na primer, slučaj sa algoritmom za utvrđivanje da li je neki broj prost koji koristi Euklidov algoritam za određivanje najvećeg zajedničkog delioca.

Naka je, na primer, potrebno da se izračuna najveći zajednički delilac tri broja. To se može izračunati računanjem NZD prva dva broja, a zatim NZD trećeg broja i prethodno izračunatog NZD. Za programiranje takvog zadatka izgleda da bismo morali dva puta da ispisujemo programske iskaze koji izračunavaju NZD. To je nepraktično, kako sa stanovišta memorijskog prostora računara – dva puta se generišu slične programske instrukcije – tako i sa stanovišta čitljivosti programa. Složeniji programi bi uskoro postali sasvim nerazumljivi i onima koji ih pišu.

Deo programa koji se ponavlja se izdvaja u zaseban modul koji se može uključiti u drugi program. Modul je samostalan algoritam koji se dizajnira nezavisno od konteksta u kome će se koristiti. Moduli se u programskim jezicima nazivaju *procedure*, *rutine*, *podprogrami* ili *funkcije*. Za algoritam koji koristi modul kažemo da ga *poziva*.

Primer 7. Sastaviti program koji izračunava najveći zajednički delilac tri broja. Program treba da koristi zasebnu funkciju za računanje najvećeg zajedničkog delioca.

VBA	`Function Euklid(ByVal m As Integer, ByVal n As Integer) _`
	`As Integer`
	`    Dim r As Integer`
	`    r = m Mod n`
	`    While r > 0`
	`        m = n`
	`        n = r`

```
        r = m Mod n
    Wend
    Euklid = n
End Function
Sub EuklidFunction()
Dim a As Integer, b As Integer, c As Integer
Dim nzd As Integer
    a = InputBox("Prvi broj")
    b = InputBox("Drugi broj")
    c = InputBox("Treci broj")
    nzd = Euklid(a, b)
    nzd = Euklid(nzd, c)
    MsgBox "Najveci zajednicki delilac je " & nzd
End Sub
```

Python

```
# m prvi broj
# n drugi broj
def Euklid(m,n):
    r = m % n
    while r > 0:
        m = n
        n = r
        r = m % n
    return n

# a, b, c - tri cela broja
print("Unesi tri cela broja")
a = int(input())
b = int(input())
c = int(input())
nzd = Euklid(a,b)
nzd = Euklid(nzd,c)
print ("Najveci zajednicki delilac je ", str(nzd))
```

Primer:

```
a <- 44
b <- 66
c <- 55
```

Prvi poziv funkcije `Euklid`:

```
Euklid(44,66)  m<-44     66   44
               n<-66     44   22
               r<-44     22   0
               r > 0 da  da   ne   Euklid <- 22
nzd <- 22
```

Drugi poziv funkcije `Euklid`:

```
Euklid(22,55)  m<-22     55   22
               n<-55     22   11
               r<-22     11   0
               r > 0 da  da   ne   Euklid <- 11
nzd <- 11
```

Porcedura `EuklidFunction` ispisuje:

```
Najveci zajednicki delilac je 11
☺
```

U ovom programu izračunavanje najvećeg zajedničkog delioca izdvojeno je u zaseban modul – funkciju *Euklid*. Iskazi tela funkcije se ne izvršavaju sve dok funkcija ne bude pozvana. U definiciji funkcije zadati su *formalni parametri* – u gornjoj funkciji to su promenljive *m* i *n* koje se koriste samo u telu funkcije. Kada se funkcija poziva, u pozivu funkcije koriste se *stvarni parametri* – u gornjem primeru to su promenljive *a*, *b* i *nzd*, *c*. Prilikom poziva funkcije formalni parametri dobijaju vrednost stvarnih parametara ukoliko se koristi *prenos parametara po vrednosti* (by value), što je slučaj sa gornjom funkcijom. I telo funkcije se izvršava za te vrednosti. U gornjem primeru je formalni parametar *m* za prvi poziv funkcije *Euklid* dobio vrednost promenljive *a*, a za drugi poziv vrednost promenljive *nzd*. Drugi način prenosa formalnih parametara je *po referenci* (by reference). U tom slučaju formalni parametri postaju samo druga imena za memorijske lokacije stvarnih parametara s kojima su povezani pozivom funkcije. Rezultat toga je da sve izmene formalnih parametara prenetih po vrednosti ostaju lokalne i ne menjaju stvarne parametri s kojima su povezani, dok se izmene formalnih parametara pozvanih po referenci prosleđuju i na stvarne parametre s kojima su povezani pozivom funkcije.

(VBA)Funkcija može (i treba) da vrati vrednost, a to je vrednost koja je dodeljena imenu funkcije u telu funkcije. Prema tome, u programu se mora deklarisati i kog je tipa vrednost koju funkcija vraća. (Python) Vrednost koju funkcija vraća mora se pojaviti u iskazu ***return*** koji istovremeno označava i povratak u glavnu proceduru. U gornjem primeru vrednost koju funkcija vraća, a to je najveći zajednički delilac dva broja, koristi se u iskazu dodele.

VBA

Funkcije

```
Function imeFunkcije(deklaracije argumenata) As PovratniTip
      iskazi
      ...
      imeFunkcije = povratnaVrednost
End Function

Function Kvadrat(n As Integer) As Integer
      Kvadrat = n * n
End Function
```

Funkcije se prema tipu vrednosti koje vraćaju dele na:

- celobrojne (Integer, Long)
- realne (Single, Double)
- logičke (Boolean)

Procedure

```
Sub imeProcedure(deklaracije argumenata)
      iskazi
End Sub
```

Prenos argumenata potprograma

- po referenci (podrazumevan). Može se specificirati navođenjem ključene reči **ByRef** ispred deklaracije argumenta potprograma.
- po vrednosti. Može se specificirati navođenjem ključene reči **ByValue** ispred deklaracije parametara potprograma.

Deklaracija argumenta

[ByValue | ByRef] imeArgumenta As tipArgumenta (npr. ByValue n As Integer)

Pozivanje funkcija i procedura

- Funkcije koje vraćaju vrednost se pozivaju uključivanjem u neki izraz – brojčani ili logički već prema tipu funkcije.
- Procedure **Sub** se pozivaju navođenjem imena procedure i stvarnih argumenata posle ključne reči **Call**.

Primer 8. Za dati učitani ceo broj odrediti da li je prost broj. Koristiti postupak izložen u odeljku o algoritmima. Donji program koristi funkciju `Euklid` koja je korišćena i u **Primer 7**. Utvrđivanje da li je broj prost je takođe izdvojeno u zasebnu funkciju `DaLiProst` koja vraća bulovsku vrednost `true` ako broj jeste prost, a vrednost `false` ako nije.

VBA	`Function DaLiProst(ByVal m As Integer) As Boolean`

```
Dim f As Integer

If Euklid(m, 2) <> 1 Then     ' broj je deljiv sa 2
    DaLiProst = False
Else
    f = 3
    While f <= Sqr(m) And Euklid(m, f) = 1
        f = f + 2
    Wend
    If Euklid(m, f) <> 1 Then
        DaLiProst = False
    Else
        DaLiProst = True
    End If
End If
End Function

Sub ProstBroj()
Dim broj As Integer
broj = InputBox("unesi broj")
If DaLiProst(broj) Then
    MsgBox "broj " & broj & " je prost"
Else
    MsgBox "broj " & broj & " nije prost"
End If
End Sub
```

Python

```
import math
# m prvi broj
# n drugi broj
def Euklid(m,n):
    r = m % n
    while r > 0:
        m = n
        n = r
        r = m % n
    return n

def DaLiProst(x):
    if Euklid(x,2) != 1:
        return False
    else:
        f = 3
        while (f <= math.sqrt(x)) and (Euklid(x, f)== 1):
            f = f + 2
        if Euklid(x, f) != 1:
            print("Ima činilac", f)
            return False
        else:
            return True

print("unesi broj")
```

```
a = int(input())
dali = DaLiProst(a)
if dali:
    print("Broj",a," je prost")
else:
    print("Broj",a," nije prost")
```

Primer: 143

```
Glavni – učitavanje broj ← 143
     DaLiProst x ← broj=143
          Euklid    m ← broj=143   2    1
                    n ← 2          1
                    r ← 1          0
     Euklid(143,2)!=1    false     (nastavak)
     f ← 3
     f < √143=11.96 true
          Euklid    m ← broj=143   3    2    1
                    n ← 3          2    1
                    r ← 2          1    0
     Euklid(143,3) ==1   true
     While uslov true → continue
     f ← 5
     f < √143=11.96 true
          Euklid    m ← broj=143   5    3    2    1
                    n ← 5          3    2    1
                    r ← 3          2    1    0
     Euklid(143,5) ==1   true
     While uslov true → continue
     f ← 7
     f < √143=11.96 true
          Euklid    m ← broj=143   7    3    1
                    n ← 7          3    1
                    r ← 3          1    0
     Euklid(143,7) ==1   true
     While uslov true → continue
     f ← 9
     f < √143=11.96 true
          Euklid    m ← broj=143   9    8    1
                    n ← 9          8    1
                    r ← 8          1    0
     Euklid(143,9) ==1   true
     While uslov true → continue
     f ← 11
     f < √143=11.96 true
          Euklid    m ← broj=143   11
                    n ← 11
```

```
                r ← 0
Euklid(143,11) ==1  false
while uslov false → stop
Euklid(143,11) != 1 →
```
DaLiProst ispisuje:
```
Ima činilac 11
DaLiProst return false
```
Program ispisuje:
```
Broj 143 nije prost
```
☺

Primer 9. Sastaviti proceduru `Sub` funkciju koja od datog datuma pravi datum sledećeg dana. Ona koristi funkciju za određivanje broja dana u mesecu koja se zasniva na istom algoritmu koji je predstavljen u **Primer 2**.

VBA

```
Function BrDana(ByVal m As Integer, ByVal g As Integer) _
      As Integer
      ' ulaz: m – mesec; g – godina
      ' izlaz: BrDana – broj dana u mesecu m godine g
Select Case m
    Case 1, 3, 5, 7, 8, 10, 12
        BrDana = 31
    Case 4, 6, 9, 11
        BrDana = 30
    Case 2
        If (g Mod 400 = 0) _
        Or _
        ((g Mod 4 = 0) And (g Mod 100 <> 0)) Then
            BrDana = 29
        Else
            BrDana = 28
        End If
    Case Else
        BrDana = 0
        MsgBox "Nepostojeci mesec " & m
End Select
End Function

Sub SledeciDatum(ByRef d As Integer, _
                ByRef mes As Integer, _
                ByRef god As Integer)
   ' ulaz – tekući datum: d – dan, mes – mesec, god – godina
   ' izlaz – sutrašnji datum: d – dan, mes – mesec, god – godina
    Dim n As Integer
    n = BrDana(mes, god)
    If n = 0 Then       ’ mes nije broj od 1 do 12
        d = 0
        mes = 0
        god = 0
    ElseIf d = n Then   ’ tekući dan poslednji u mesecu
        d = 1
        If mes = 12 Then      ’ mesec je poslednji mesec u god
```

```
            mes = 1
            god = god + 1
        Else
            mes = mes + 1
        End If
    Else                ' tekući dan nije poslednji u mesecu
        d = d + 1
    End If
End Sub

Sub Primer9()
Dim dan As Integer, mesec As Integer, godina As Integer
godina = InputBox("Unesi broj tekuce godine")
mesec = InputBox("Unesi broj tekuceg meseca u godini")
dan = InputBox("Unesi broj dana u tekucem mesecu")
Call SledeciDatum(dan, mesec, godina)
MsgBox "Sutra je " & dan & "." & mesec & "." & godina & "."
End Sub
```

Python

```
# ulaz: m – mesec; g – godina
# izlaz: broj dana u mesecu m godine g
def BrDana(m, g):
    if m in [1, 3, 5, 7, 8, 10, 12]:
        return 31
    elif m in [4, 6, 9, 11]:
        return 30
    elif m == 2:
        if (g % 400==0) or ((g % 4==0) and (g % 100!=0)):
            return 29
        else:
            return 28
    else:
        return 0

# ulaz – tekući datum: x = [d, mes, god]
# izlaz – sutrašnji datum
def SledeciDatum(x):
    d = x[0]
    mes = x[1]
    god = x[2]
    n = BrDana(mes, god)
    if n == 0:
        d = 0   # mes nije broj od 1 do 12
        mes = 0
        god = 0
    elif d == n:
        d = 1  # tekući dan poslednji u mesecu
        if mes == 12:
            mes = 1  # mesec je poslednji mesec u god
            god = god + 1
```

```
        else:
            mes = mes + 1
    else:
        d = d + 1  # tekući dan nije poslednji u mesecu
    return [d,mes,god]

# dan, mesec, godina – tri cela broja
print("Unesi datum (dan, mesec, godina)")
dan = int(input())
mesec = int(input())
godina = int(input())
datum = [dan,mesec,godina]
print("Mesec datuma ", datum, " ima ",
BrDana(mesec,godina), " dana")
sledeci_datum = SledeciDatum(datum)
print("Sutra je ", sledeci_datum)
```

Primer:

```
godina ← 2010
mesec ← 11
dan ← 30
poziv SledeciDatum
     d ← 30
     mes ← 11
     god ← 2010
     poziv BrDana
          m ← 11
          g ← 2010
          Case 11 → BrDana ← 30
     n ← 30
     n = 0? False
     n = d? True
     d ← 1
     mes = 12? False
     mes ← 12
```

Program ispisuje:

```
Mesec datuma[30,11,2010] ima 30 dana
Sutra je [1,12,2010]
```

☺

Rekurzivne procedure

Algoritam koji se izražava preko samog sebe naziva se rekurzivnim algoritmom. Programske funkcije koje pozivaju same sebe su rekurzivne funkcije. Na primer, funkcija *faktorijel* (vidi Primer 5) se računa kao $n! = 1*2*3*\ldots*(n-1)*n$. Ovaj izraz bi se na drugi na drugi način mogao zapisati kao: $n! = n \cdot (n-1)!$ Drugim rečima, računanje faktorijela celog broja *n* izražava se preko izračunavanja

faktorijela broja *n*-1, a računanje faktorijela celog broja *n*-1 izražava se preko izračunavanja faktorijela broja *n*-2, i tako dalje sve dok se ne dođe do računanja faktorijela broja 1 koji nam je unapred poznat (broj 1). Sledi rekurzivna funkcija za računanje funkcije *faktorijel* i program koji izračunava vrednost funkcije za zadati broj s ulaza.

VBA

```
Function faktRek(ByVal n As Integer) As Integer
    If n = 1 Then
        faktRek = 1
    Else
        faktRek = n * faktRek(n - 1)
    End If
End Function

Sub ProcFaktRek()
Dim broj As Integer
broj = InputBox("Unesi broj <=12")
If broj <= 12 Then
    MsgBox "Faktorijel broja " & broj & " je " & _
              faktRek(broj)
Else
    MsgBox "prekoračenje"
End If
```

Python

```
def rfaktorijel(n):
    if n == 1:
        return 1
    else:
        return n * rfaktorijel(n - 1)

print("unesi prirodan broj")
broj = int(input())
fakt = rfaktorijel(broj)
print("Faktorijel broja ", broj, " je ", fakt)
```

Kako radi ova procedura za slučaj broja 4 je prikazano u donjoj tabeli (poslednja vrsta u tabeli sa sivom pozadinom se izvršava u obrnutom redosledu).

broj ← 4

I poziv	II poziv	III poziv	IV poziv
n ← 4	n ← 3	n ← 2	n ← 1
n == 1? false	n == 1? false	n == 1? false	n == 1? true
(4 == 1?)	(3 == 1?)	(2 == 1?)	(1 == 1?)
← 4*→ II	← 3*→ III	← 2*→ IV	← 1
← 4*6=24	← 3*2=6 ⇐	← 2*1=2 ⇐	 ⇐

Program ispisuje:

Faktorijel broja 4 je 24

☺

Kod rekurzivnih funkcija je važno da se obezbedi da funkcija ne poziva samu sebe beskonačno. Kod svakog poziva funkcija bi trebalo da rešava „jednostavniji“ slučaj, pa treba da postoji granični slučaj kada je ulaz tako jednostavan da se izlaz može direktno dati, bez ponovnog poziva funkcije. U gornjem primeru taj granični slučaj je $n = 1$, kada bez poziva funkcije znamo da je 1!=1. Taj granični slučaj će svakako biti dostignut jer je pri svakom rekurzivnom pozivu funkcije argument n sve manji (za 1).

Primer 10. Sastaviti rekurzivnu funkciju za računanje najvećeg zajedničkog delioca.

VBA

```
Function EuklidRek(ByVal a As Integer, ByVal b As Integer) As
Integer
    If (a Mod b) = 0 Then
        EuklidRek = b
    Else
        EuklidRek = EuklidRek(b, a Mod b)
    End If
End Function

Sub Primer10()
Dim a As Integer, b As Integer
a = InputBox("unesi prvi broj")
b = InputBox("unesi drugi broj")
MsgBox "NZD je " & EuklidRek(a, b)
End Sub
```

Python

```
def rEuklid(m, n):
    if (m % n) == 0:
        return n
    else:
        return rEuklid(n, m % n)

# a, b - dva cela broja
print("Unesi dva cela broja")
a = int(input())
b = int(input())
nzd = rEuklid(a,b)
print("NZD je ", nzd)
```

Primer: `nzd = rEuklid (25, 15)`

I poziv	II poziv	III poziv
m ← 25	m ← 15	m ← 10
n ← 15	n ← 10	n ← 5
m Mod n == 0? NE	m Mod n == 0? NE	m Mod n == 0? DA
(10 == 0?)	(5 == 0?)	(0 == 0?)
← →II	← →III	← 5
←5	← 5 ⇐	⇐

Program ispisuje:

NZD je 5 ☺

Primer jednog testa

1. Povezati promenljive i kostantne vrednosti sa načinom zapisivanja u memoriji računara - (a) potpuni komplement, (b) kodirani karakteri (c) pokretni zarez):

VBA		**Python**	
7.23	c	7.23	c
"7.23"	b	"7.23"	b
7	a	7	a
7.23E-2	c	7.23E-2	c
"!«#$%"	b	"!«#$%"	b
Dim x As String	b		
Dim x As Single	a		
Dim x As Long	c		

2. Sastaviti logički izraz koji utvrđuje da li je godina koja je sadržaj promenljive `god` prestupna (`true` – jeste, `false` – nije). Prikazati kako se računa vrednost tog izraza za godinu 1900.

VBA	(god Mod 400 = 0) Or ((god Mod 4 = 0) And (god Mod 100 <> 0)) (1900 Mod 400 = 0) Or ((1900 Mod 4 = 0) And (1900 Mod 100 <> 0)) (300 = 0) Or ((1900 Mod 4 = 0) And (1900 Mod 100 <> 0)) false Or ((1900 Mod 4 = 0) And (1900 Mod 100 <> 0)) false Or ((0 = 0) And (1900 Mod 100 <> 0)) false Or (true And (1900 Mod 100 <> 0)) false Or (true And (0 <> 0)) false Or (true And false) false Or false false
Python	(god % 400 == 0) or ((god % 4 == 0) and (god % 100 != 0)) (1900 % 400 == 0) or ((1900 % 4 == 0) and (1900 % 100 != 0)) (300 == 0) or ((1900 % 4 == 0) and (1900 % 100 != 0)) False or ((1900 % 4 == 0) and (1900 % 100 != 0)) False or ((0 == 0) and (1900 % 100 != 0)) False or (True and (1900 % 100 != 0)) False or (True and (0 != 0)) False or (True and False) False or False False

3. Neka je X celobrojna promenljiva čija je vrednost pozitivan broj od tačno 5 cifara. Napisati u VBA deo programa koji određuje broj Y kod koga je izbačena treća cifra po redu (cifra stotina). Na primer, za broj X=12468

program treba da odredi Y=1268. Ovaj deo programa može da koristi pomoćne promenljive (ne treba ispisivati deklaracije promenljivih).

VBA	`p = x \ 1000  ' p <- 12`
	`k = x Mod 100  ' k <- 68`
	`y = p*100+k  ' y <- 12*100+68=1268`

Python	`p = x // 1000  # p <- 12`
	`k = x % 100  # k <- 68`
	`y = p*100+k  # y <- 12*100+68=1268`

```
p = x \ 1000        ' p <- 12
k = x Mod 100       ' k <- 68
y = p*100+k         ' y <- 12*100+68=1268
```

4. (a) Šta radi sledeći deo VBA programa (ispisati šta program ispisuje):

VBA	`s = 29`
	`n = ""`
	`MsgBox " s=" & s & ", n=" & n`
	`While` `s > 0`
	`  n = (s Mod 2) & n`
	`  s = s \ 2`
	`  MsgBox " s=" & s & ", n=" & n`
	`Wend`

Python	`s = 29`
	`n = ""`
	`print(" s=", s, ", n=", n)`
	`while s > 0 :`
	`  n = str(s % 2) + n`
	`  s = s // 2`
	`  print(" s=", s, ", n=", n)`

```
s=29, n=
s=14, n=1
s=7, n=01
s=3, n=101
s=1, n=1101
s=0, n=11101
```

(b) Zapisati u VBA deo programa koji ima isti učinak kao deo pod (a), a koristi **Do-Loop Until** iskaz (tj. Simulira korišćenje **Do-Loop Until** iskaza):

VBA	`s = 29`
	`n = ""`
	`MsgBox " s=" & s & ", n=" & n`
	`Do`
	`  n = (s Mod 2) & n`
	`  s = s \ 2`
	`  MsgBox " s=" & s & ", n=" & n`
	`Loop Until s = 0`

Python	`s = int(input())`
	`n = ""`
	`print(" s=", s, ", n=", n)`
	`while True:`
	`    n = str(s % 2) + n`
	`    s = s // 2`
	`    print(" s=", s, ", n=", n)`
	`    if s == 0:`
	`        break`

5. Sastaviti funkciju koja vraća vrednost **True** ili **False** u zavisnosti od toga da li je vrednost celobrojne promenljive *z* jednaka zbiru celobrojnih promenljivih *x* i *y*. Celobrojne promenljive *x*, *y* i *z* su argumenti funkcije.

VBA	`Function Dali(ByVal x As Integer, ByVal y As Integer, _`
	`            By Val z As Integer) As Boolean`
	`    Dali = (z = x + y)`
	`End Function`

Python	`def Dali(x, y, z):` `    return (z == x + y)`