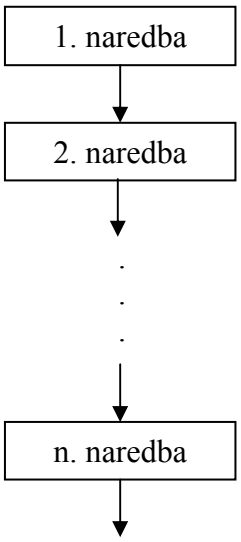
6. PROGRAMSKE STRUKTURE STRUKTUIRANOG PROGRAMIRANJA

U programiranju često postoji potreba da se redoslijed izvršavanja naredbi uslovi prethodno dobivenim međurezultatima u toku izvršavanja programa. Na ovaj način se mogu obrazovati različiti redoslijedi naredbi pri izvršavanju programa. Ovaj redoslijed naredbi, u kojem se program može izvršavati, čini programsku strukturu. Po ovoj osobini naredbe mogu obrazovati linijsku, razgranatu i cikličnu strukturu.

6.1 Linijska struktura

Naredbe koje se pri izvršavanju programa uvijek izvršavaju u navedenom redoslijedu obrazuju linijsku strukturu. Poslije izvršenja jedne, prelazi se na izvršenje slijedeće naredbe.



Sl 6.1 Linijska programska struktura

Primjer 6.1:

Sastaviti program koji za zadane vrijednosti x i y izračunava vrijednost funkcije:

$$F(x, y) = e^{-2x}\sin(3y - x)$$

```
‘PROGRAM P601

Print "Unesite vrijednosti za X i Y: "
Input "X = "; X
Input "Y = "; Y

F = Exp(-2*X) * Sin(3*Y-X)

Print F

End
```

Pokrenemo li program, na ekranu ćemo dobiti:

```
Unesite vrijednosti za X i Y:
X = 1.5
Y = -3.4
Rezultat je: F(x,y) = □0.37936929e-1
```

6.2 Razgranata struktura

Često je potrebno u programu predvidjeti različite tokove izvršavanja naredbi. Naredbe koje omogućavaju promjenu redoslijeda izvršavanja naredbi programa zovu se naredbe prelaska, koje se dijele na:

1. naredbe bezuslovnog prelaska i
2. naredbe uslovnog prelaska, koje omogućuju grananja u programu (razgranata struktura). Broj grana zavisi od naredbe prelaska kojom se ostvaruje grananje.

6.2.1 Naredba bezuslovnog prelaska

Naredba bezuslovnog prelaska omogućava prelazak na bilo koju izvršnu naredbu u istoj programskoj jedinici. Sintaksa naredbe je:

GOTO n

6.2.2 Naredbe uslovnog prelaska

Ove naredbe mogu biti:

1. blokovski uslovni prelazak,
2. logički uslovni prelazak,

Blokovski i logički uslovni prelazak kao uslov za prelazak imaju vrijednost logičkog izraza, kod koga argument izraza može biti relacijski izraz.

Relacijski izraz

Relacijski izraz je sastavljen od dva aritmetička ili dva znakovna izraza, između kojih se piše relacijski znak. Relacijski znaci su:

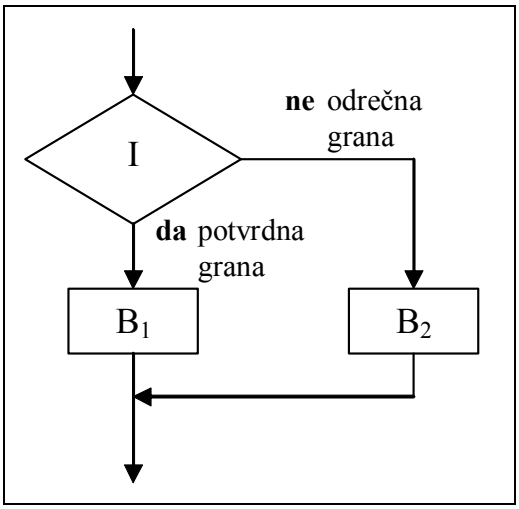
Tabela 6.1 Relacijski operatori **if** izraza

<	manje od
>	veće od
=	jednako
>=	veće ili jednako
<=	manje ili jednako
<>	nije jednako

Vrijednost relacijskog izraza može biti .TRUE. ako su aritmetički, odnosno znakovni izrazi u zadatoj relaciji, ili .FALSE. ako nisu.

Blokovski uslovni prelazak

Kod blokovskog uslovnog prelaska uslov po kojem se vrši prelazak predstavlja logički izraz (I) čija vrijednost može biti .TRUE. ili .FALSE.



Sl. 6.2 Grafički prikaz programske strukture koju definiše naredba IF ... THEN ... ELSE

Sintaksa ove strukture je:

```
IF (I) THEN
        B1
ELSE
        B2
END IF
```

Zbog bolje uočljivosti ove strukture u programu, dobro je primijeniti pomjeranje blokova udesno.

U blokovima B_1 i B_2 može biti jedan ili više izraza LBa.

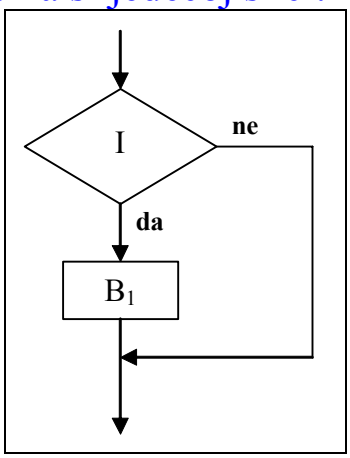
Ako je blok B_2 prazan, tada imamo:

```
IF (I) THEN
        B1
ELSE
END IF
```

odnosno:

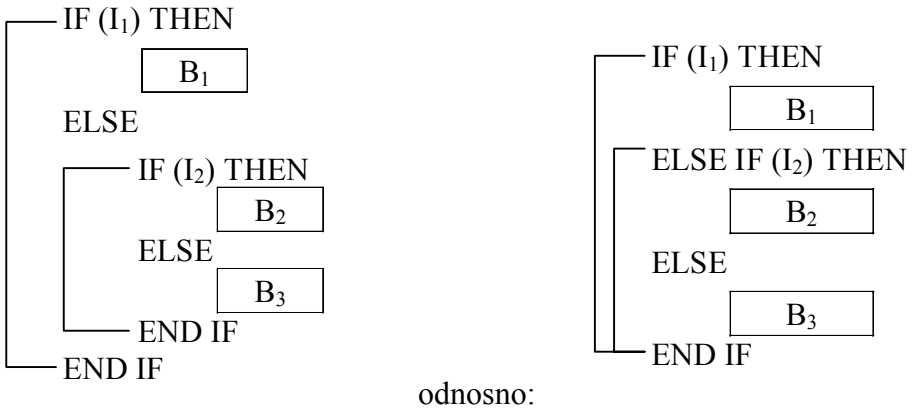
```
IF (I) THEN
        B1
END IF
```

jer se u ovom slučaju struktura može pisati i bez ELSE. Grafički prikaz ovog slučaja dat je na slijedećoj slici:



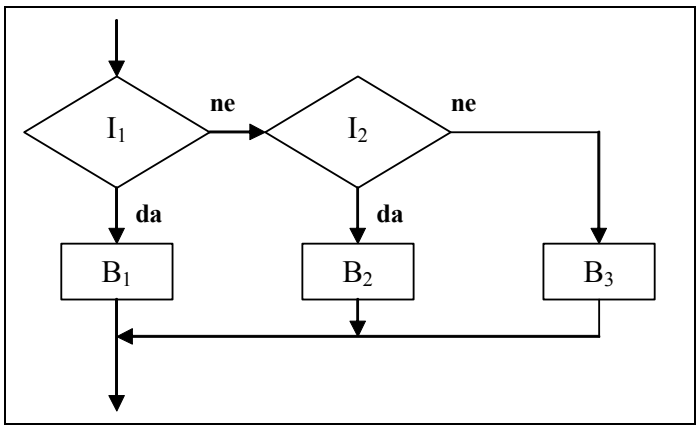
Sl. 6.3 Struktura IF ... THEN

Ugniježdene IF naredbe (kada se u određnoj grani IF ... THEN ... ELSE strukture nalazi druga IF ... THEN ... ELSE struktura):



Sl. 6.4 Ugniježdene IF naredbe

Desno je prikazano ugnježdenje sa ELSE ... IF naredbom u jednostavnijem i pregledijem obliku, a grafički prikaz ove programske strukture dat je na slijedećoj slici:



Sl. 6.5 Grafički prikaz ugnježdenja sa IF ... THEN naredbom

Primjer P602

Za zadano x_1 i x_2 treba izračunati y po formuli:

$$y = \begin{cases} x_1 + x_2 & \text{ako je } x_1 < x_2 \\ x_1 \cdot x_2 & \text{ako je } x_1 \geq x_2 \end{cases}$$

```
‘Program P602

Print "Unesite dva broja:"
Input "X1 =□"; X1
Input "X2 =□"; X2

If (X1 < X2) Then
    Y = X1+X2
Else
    Y = X1*X2
End if

Print "Rezultat je Y =□"; Y
End
```

Kada se u blokovima nalazi samo po jedna naredba, tada se može pisati samo jedna naredba:

```
If (X1 < X2) Then Y = X1+X2 Else Y = X1*X2
```

Primjer P603

Isti zadatak kao i prethodni, riješen korištenjem naredbe IF ... THEN:

```
‘Program P603

Print "Unesite dva broja:"
Input "X1 =□"; X1
Input "X2 =□"; X2

If (X1 < X2) Then
    Y = X1+X2
End if
If (X1 >= X2) Then
    Y = X1*X2
End if

Print "Rezultat je Y = "; Y
End
```

Primjer P604

Da bi se ilustrovalo korištenje IF-naredbe u okviru druge IF-naredbe, treba napisati program za izračunavanje:

$$y = \begin{cases} x_1 + x_2 & \text{ako je } x_1 < x_2 \\ x_1 \cdot x_2 & \text{ako je } x_1 = x_2 \\ x_1 - x_2 & \text{ako je } x_1 > x_2 \end{cases}$$

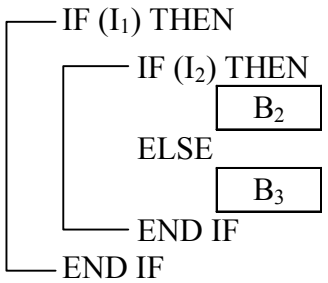
```
'Program P604

Print "Unesite dva broja:"
Input "X1 =□"; X1
Input "X2 =□"; X2

If (X1 < X2) Then
    Y = X1+X2
Else
    If (X1 = X2) Then
        Y = X1*X2
    Else
        Y = X1-X2
    End If
End If

Print "Rezultat je Y = "; Y
End
```

Na slici 6.6 prikazane su ugniježdene If naredbe za slučaj kada je blok B_1 prazan (prema slici 6.4).



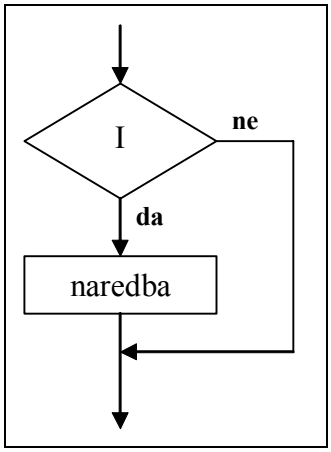
Sl. 6.6 Ugniježdene IF naredbe za slučaj kada je blok B_1 prazan (prema slici 6.4)

Logički uslovni prelazak

Naredba logičkog uslovnog prelaska omogućuje zapis jedne izvršne naredbe u potvrdnoj grani i bez naredbi u odrečnoj grani strukture. Sintaksa ove naredbe je:

IF (I) THEN naredba

gdje je (I) logički izraz čija se vrijednost izračunava. Ako je ova vrijednost istinita izvršava se navedena naredba, u protivnom ova naredba se preskače i prelazi se na naredbu koja slijedi iza IF naredbe.



Sl. 6.7 Logički uslovni prelazak

Primjer P605

```
'Program P605

Print "Unesite dva broja:"
Input "X1 = "; X1
Input "X2 = "; X2

If (X1 < X2) Then Y = X1+X2
If (X1 = X2) Then Y = X1*X2
If (X1 > X2) Then Y = X1-X2

Print "Rezultat je Y = "; Y
End
```

6.3 Ciklična struktura

Ako se niz naredbi u programu može izvršiti više puta u toku izvršavanja programa, kaže se da ove naredbe obrazuju cikličnu strukturu. Među ovim naredbama barem jedna mora omogućavati izlazak iz ciklusa. **Uslov** pod kojim se izlazi iz ciklusa zove se **izlazni kriterijum ciklusa** (brojački ciklus – broj ponavljanja ciklusa i iterativni ciklus – dostignuta tačnost u procesu računanja).

Programski ciklus u kojem izlazni kriterijum ne obezbjeđuje konačan broj ponavljanja ciklusa, već se ciklus potencijalno može ponavljati neograničen broj puta zove se beskonačan ciklus.

Programska struktura zove se:

1. linijska kompozicija ciklusa ako više programskih ciklusa slijede jedan iza drugog u programu, ili
2. koncentrična kompozicija ciklusa da se programski ciklusi nalaze jedan unutar drugog.

6.3.1 Beskonačan ciklus

Vrlo rijetko se koristi u programiranju. To može često biti rezultat greške u izlaznom kriterijumu ciklusa koji ne može biti zadovoljen. Ipak, u nekim slučajevima koriste se i beskonačni ciklusi.

Primjer P606 (primjer korištenja beskonačnog ciklusa):

Ciklički se ponavlja množenje broja x i izračunava

$y = x^2$ i $z = x^3$

Prekid programa postiže se zatvaranjem radnog prozora.

```
'Program P606

10 Print "Unesite jedan broj:"
Input "X =□"; X

Y = X^2
Z = X^3

Print "Rezultati su:"
Print "Y = "; Y
Print "Z = "; Z

Goto 10
End
```

6.3.2 Iterativni ciklus

Ovdje je kriterijum za izlaz iz ciklusa dostignuta tačnost u procesu računanja. Najčešća primjena je kod iterativnih numeričkih postupaka. Jedan prolazak u programu odgovara jednoj iteraciji u numeričkoj metodi. **Izlazni kriterijum** može biti uslov da su razlike vrijednosti izračunatih u tekućem i prethodnom iterativnom koraku dovoljno male.

Primjer P607

Odrediti najmanji pozitivan korijen transcendentne jednačine:

$$\sin(x) = \frac{A}{x}$$

za zadanu vrijednost parametra A iz intervala [0,1;1,0]. Za izračunavanje korijena primijeniti iterativni postupak:

$$x_{i+1} = \frac{A}{\sin(x_i)}, \quad i = 0,1,2,...$$

gdje je $x_0 = \frac{\pi}{2}$. Iterativni postupak prekinuti kada bude ispunjen uslov:

$$\left|x_{i+1} - x_i\right| \le \varepsilon$$

gdje je $\varepsilon = 10^{-4}$ zadana tačnost.

Za ovakve cikluse ne možemo unaprijed znati broj ponavljanja ciklusa.

```
'Program P607

EPS = 1.E-4

10 Input "Unesite vrijednost za A:"; A
If (A < 0.1) Then Goto 10
If (A > 1) Then Goto 10

X = 3.141592*0.5
20 X1 = A/Sin(X)
   IF (Abs(X1-X) < EPS) Then Goto 30
   X = X1
   GOTO 20
```

```
30 Print "Korijen jednacine sin(x) = A/x je:"
   Print "A =□"; A
   Print "X =□"; X
End
```

6.3.3 Brojački ciklus

Ovdje je kriterijum za izlazak iz ciklusa broj ponavljanja ciklusa.

Primjer P608

Programom se izračunava suma od 6 brojeva.

Izlazni kriterijum je ispitivanje da li je vrijednost projenljive N jednaka nuli. Ciklička struktura je realizirana logičkom uslovnom naredbom IF, kombinovano sa GOTO naredbom. Brojevi se unose jedan po jedan.

```
'Program P608

N = 6
Y = 0
Print "Unesite brojeve za sumiranje:"
10 Input X
   Y = Y+X
   N = N-1
   If (N <> 0) Goto 10

Print "Suma unesenih brojeva iznosi:□"; Y
End
```

6.3.4 Petlje ili standardne naredbe programskog cilusa

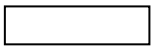
Za programiranje brojačkih programskih ciklusa, koji se u programiranju najčešće javljaju, u LB jeziku postoje dva oblika standardnih naredbi za petlje, odnosno za programske cikluse:

- For ... Next i
- While ... Wend petlja.

For ... Next petlja

```
For i=m1 To m2 [Step m3]
    
Next i
```

gdje je:

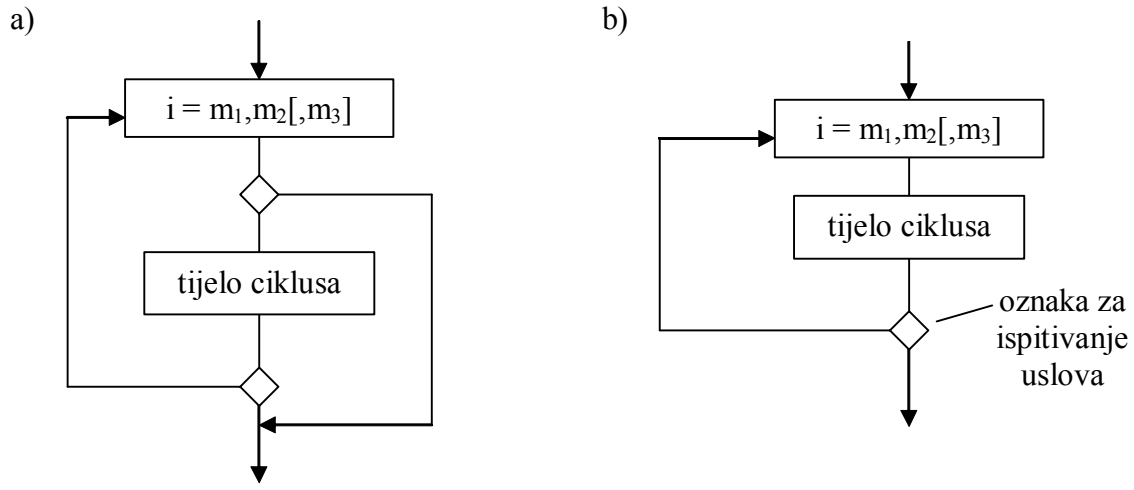
`For, To, Step` – službene riječi,
i – cjelobrojna promjenljiva – ciklusna promjenljiva (brojač ciklusa, a često i pokazivač),
m_1, m_2, m_3 – cjelobrojne veličine,.
 – tijelo ciklusa.

Ako se `Step` m_3 ne navede, podrazumijeva se da je $m_3 = 1$. Značenje gornje naredbe je sljedeće:

ako je $m_1 \le m_2$ i $m_3 > 0$ ili $m_1 \ge m_2$ i $m_3 < 0$, onda pri prvom prolazu kroz tijelo ciklusa ciklusna promjenljiva ima vrijednost m_1, pri drugom m_1+m_3, pri trećem m_1+2m_3, itd., dok ne dostigne vrijednost m_2. Dakle, ciklus se ponavlja k puta, gdje je:

$$k = \text{int}\left(\frac{m_2 - m_1}{m_3} + 1 \right)$$

Ako gore navedeni uslov između m_1, m_2 i m_3 nije ispunjen, tj. $k \le 0$, tada se ne prolazi kroz tijelo ciklusa.



Sl. 6.8 Grafički prikaz ciklusa sa For .. Next naredbom
a) slučaj kada se može desiti da se ne prolazi kroz tijelo ciklusa,
b) slučaj kada se ciklus uvijek izvršava najmanje jedanput

Primjer P609

Izračunava se suma 6 unešenih brojeva korištenjem `For ... Next` petlje. Odbrojavanje prolazaka kroz ciklus vrši se pomoću ciklusne promjenljive i.

```
'Program P609

Y = 0
Print "Unesite brojeve za sumiranje:"
For i = 1 To 6
    Input X
    Y = Y+X
Next i

Print "Suma unesenih brojeva iznosi:□"; Y
End
```

Primjer P610

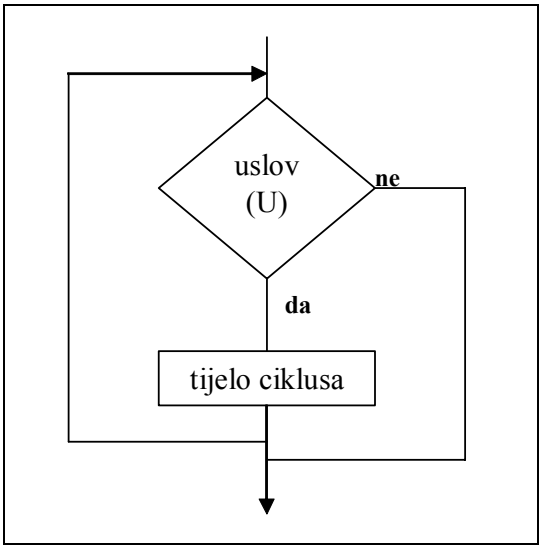
```
'Program P610

Y = 0
Print "Unesite koliko brojeva zelite sumirati:"
Input "N = "; N
Print "Unesite brojeve za sumiranje:"
For i = 1 To N
    Input X
    Y = Y+X
Next i

Print "Suma unesenih brojeva iznosi:□"; Y
End
```

While ... Wend petlja

Ova petlja omogućava kontrolu petlje preko realacionog testa, koji se koristi istim relacijskim operatorima kao i kod `If` izraza (Tabela 6.1).



Sl. 6.9 Grafički prikaz While ... Wend petlje

Sintaksa:	Djejstvo:
`WHILE (U)`	Radi dok je (U)
tijelo ciklusa	
`WEND`	Kraj ciklusa

Primjer:

```
WHILE (NPROL.LT.NMAX)
    
WEND
```

Uslov (U) za izlazak iz ciklusa može biti:

1. unaprijed zadani broj ponavljanja ciklusa,
2. zadana tačnost računanja određene veličine. U tom slučaju brojač ciklusa mora biti izostavljen,
3. maksimalno dozvoljeni broj iteracija, kombinovan sa traženom tačnošću.

Primjer 6.11

```
'count to ten
x = 0
while x < 10
    x = x + 1
    print x
wend
end
```

Do Loop petlja

Liberty BASIC also provides a DO LOOP structure for cases when you want a loop that always executes once and then only loops back as long as a condition is met. It will continue looping back and executing the code as long as the booleanExpr evaluates to true. Here is the routine that counts to 10 using DO LOOP with while and DO LOOP with until.

Primjer 6.12

'count to 10 using "loop while" and "loop until"

```
do
    print a
    a = a + 1
loop while a < 11
```

```
do
    print b
    b = b + 1
loop until b = 11
```

6.4 Složene algoritamske strukture

Složene algoritamske strukture su kombinacija pomenutih struktura i često sadrže veći broj povezanih razgranatih i cikličkih struktura, kao i algoritamskih (programskih) cjelina, realizovanih potprogramima.

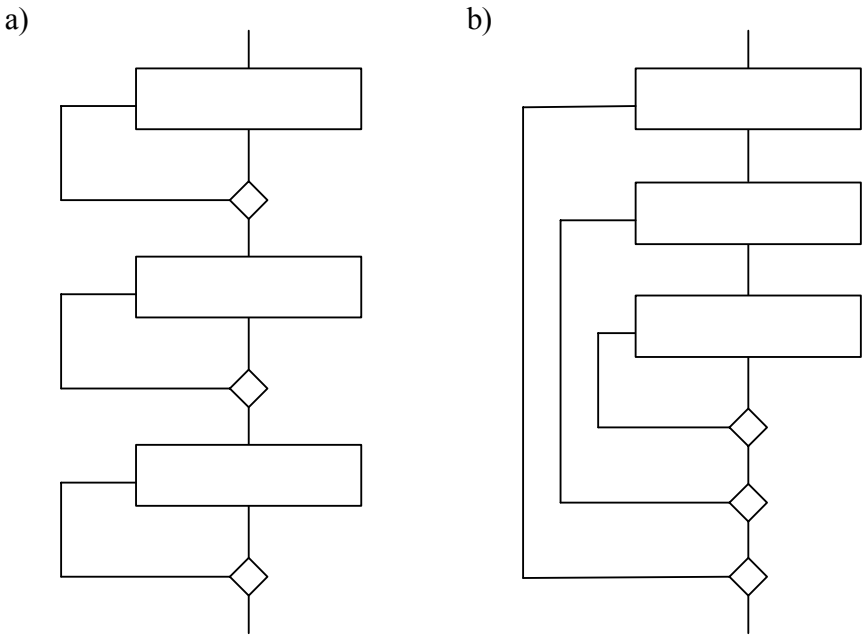
6.4.1 Kompozicije ciklusa

Složeniji programi sadrže često veći broj ciklusa. Ako su ciklusi pisani For ... Next naredbama, tada se u složenim programima mogu javiti kao:

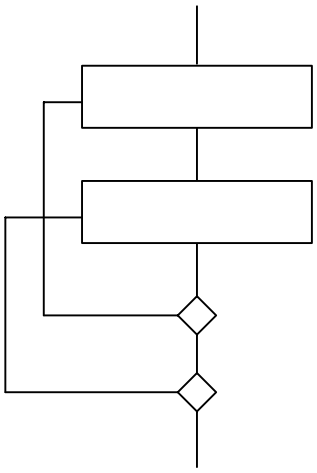
1. linijska kompozicija ciklusa (ciklusni idu jedan za drugim) i/ili
2. koncentrična kompozicija ciklusa (ciklusi se nalaze jedan unutar drugog). Zove se još i spregnuta ciklička struktura.

Ovakve kompozicije se mogu kombinovati, ali nije dozvoljeno presijecanje ciklusa, kako je to pokazano na slici 6.11

Ako jedan ciklus potpuno obuhvata drugi, onda se za prvi ciklus, koji ima višu hijerarhiju, kaže da je spoljni, a za drugi, s nižom hijerarhijom, da je unutrašnji. Naime, za svaku promjenu indeksa spoljnjeg ciklusa, mora se unutrašnji ciklus u potpunosti završiti. Tako na primjer, ako spoljni ciklus dvije spregnute cikličke strukture a ima k_1 prolazaka, a unutrašnji ciklus ima k_2 prolazaka, onda kod spregnutog rada ova dva ciklusa, unutrašnji ciklus se izvršava $k_1 \cdot k_2$ puta.



Sl. 6.10 Kompozicije ciklusa
a) linijska kompozicija ciklusa , b) koncentrična kompozicija ciklusa



Sl. 6.11 Nepravilno spregnute cikličke strukture

Prenos upravljanja u koncentričnim kompozicijama ciklusa