

## Az algoritmusok leírása

Az előző fejezetben igyekeztünk a korábban már tanult eljárásokat vektorok segítségével, lehetőleg nyelvfüggetlenül leírni. Az algoritmusok leírására többféle eljárás alakult ki, ezek egyike az eddig is használt *mondatszerű* vagy *verbális* leírás. Egy másik - sokkal szigorúbb - leírási módszer a *struktogram*. A struktogram esetében lényegében egy tömör táblázatot készítünk.

Az alábbiakban összefoglaljuk a verbális leírás (bal oldal) és a struktogram (jobb oldal) jelölésrendszerét.

### Egymást követő utasítások

Az egymást követő (szekvenciális) utasítások esetében az utasításokat a verbális leírásban és a struktogram táblázatában is egyszerűen egymás alá írjuk:

Be: A C = A + 2	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Be: A</td> </tr> <tr> <td style="padding: 2px;">C:= A + 2</td> </tr> </table>	Be: A	C:= A + 2
Be: A			
C:= A + 2			

Általában nem okoz gondot, de érdemes odafigyelni az  $A = B$  jelsorozat jelentésére. Visual Basicben ez két dolgot is jelenthet. Legyen az  $A$  értéke egyenlő a  $B$  értékével (értékadás), vagy egyenlő-e  $A$  értéke  $B$  értékével (ekkor a jelsorozat egy logikai kifejezés, melynek értéke igaz vagy hamis).

A legtöbb programozási nyelv ezt a két lehetőséget megkülönbözteti, ezért ezt többnyire az algoritmusokban is megteszik, például úgy, hogy az értékadást a  $:=$  jellel (legyen egyenlő) jelölik. A struktogramoknál mi is ezt használjuk.

### Elágazás

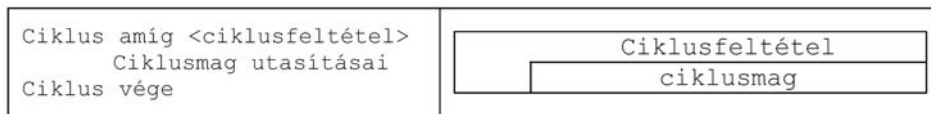
Elágazás esetében egy adott logikai kifejezés értékétől függően kell végrehajtani az utasításokat.

Ha <logikai kifejezés> akkor Utasítások1 Egyébként Utasítások2 Elágazás vége	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td style="border: none; padding: 2px;">/</td> <td style="border: none; padding: 2px;">logikai kifejezés</td> <td style="border: none; padding: 2px;">\</td> </tr> <tr> <td style="border: none; padding: 2px;">/</td> <td style="border: none; padding: 2px;">Utasítások1</td> <td style="border: none; padding: 2px;">\</td> </tr> <tr> <td style="border: none; padding: 2px;">/</td> <td style="border: none; padding: 2px;">Utasítások2</td> <td style="border: none; padding: 2px;">\</td> </tr> </table>	/	logikai kifejezés	\	/	Utasítások1	\	/	Utasítások2	\
/	logikai kifejezés	\								
/	Utasítások1	\								
/	Utasítások2	\								

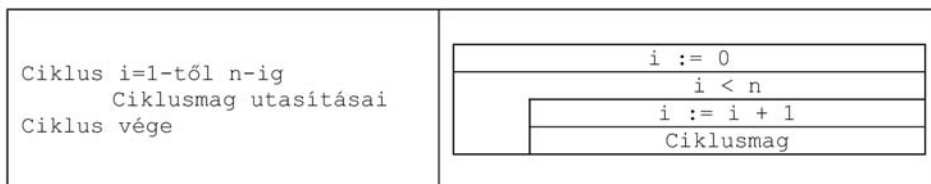
A struktogram esetében a kifejezés alatt az  $i$ -vel jelölt oldalon álló utasításokat akkor kell végrehajtani, ha az állítás igaz, a  $h$ -val jelölt oldalon pedig akkor, ha hamis. Természetesen elképzelhető hogy a hamis ág (amit a VB-ben az Else utasítás vezet be) hiányzik, ebben az esetben az adott oldalt át kell húzni.

A ciklus

A ciklus esetében a ciklusmagban lévő utasításokat mindaddig végre kell hajtani, amíg egy adott feltétel, a ciklusfeltétel teljesül.

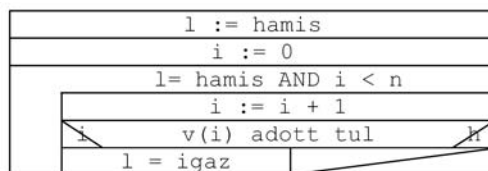


Visual Basicben ez a szerkezet a Do Loop ciklussal kódolható le közvetlen módon. Az ugyancsak gyakran használt For Next ciklus esetében a struktogramban a ciklusváltozó kezdőértékét a ciklus előtt szerepeltetnünk kell, és a változó növelését a ciklusmagban meg kell oldanunk:



Példa: A lineáris keresés algoritmus

A lineáris keresés algoritmusát az előző fejezetben már verbális leírással megadtuk. A következő példánk struktogrammal mutatja be:



Általában is jellemző, hogy a struktogram sokkal tömörebb, és a használt (emberi) nyelvtől távolabb áll, mint a verbális leírás. Hátránya viszont, hogy áttekintéséhez nagyobb gyakorlat szükséges, és nehezebben módosítható.