PROGRAMOZÁSI TÉTELEK

Most nyolc, a szakirodalomban jól ismert programozási tételt mutatunk be. Ezek kellően általános formájúak ugyan, de mindegyiküknél az egész számok egy tartományán értelmezett függvény (vagy függvények) értékeinek feldolgozása a cél. Ez a tartomány az egész számoknak egy zárt intervalluma, ezért a tételeinket ***intervallumos programozási tételeknek*** is szoktuk nevezni.

Az összegzés egy függvénynek az egész számok egy intervallumán felvett értékeit adja össze. A számlálás esetében ez a függvény logikai értékeket vesz fel az intervallumon, és azt kell meghatároznunk, hogy ezen értékek között hány igaz van. A lineáris keresés sorban, egymás után vizsgálja meg egy intervallum elemeit, és az első olyan elemnél állnak meg, amelyre a feladatnál definiált logikai függvény igaz értéket mutat. A maximum kiválasztás egy függvénynek az egész számok egy intervallumán felvett értékei között keresi meg az egyik legnagyobbat. Az egész számokon értelmezett rekurzív függvény helyettesítési értékét egy adott egész számra úgy tudjuk meghatározni, hogy előbb a megelőző néhány egész számra kiszámoljuk a függvény értékeit. A „legelső” függvényérték meghatározásától a kívánt függvényérték meghatározásáig az egész számok egy intervallumát kell befutnunk. Nem kötődik intervallumhoz ellenben a lineáris kiválasztás. Ezt a lineáris keresésekhez hasonló olyan esetben használhatjuk, ha biztosan tudjuk, hogy a keresett elem létezik. Ez a keresés is az egész számegyenesen folyik, de elég megadni a keresés kezdőpontját, a keresés addig tart, amíg a keresési feltétel (egész számokon értelmezett logikai függvény) igaz nem lesz, de erről tudjuk, hogy előbb-utóbb bekövetkezik. Itt tehát a keresés során egy olyan intervallumot járunk be, amelynek a végpontja csak implicit módon van megadva.

Az intervallumokra a feladatosztályok különféle előfeltételeket fogalmaznak meg. Nem lehet üres az intervallum a rekurzív függvény helyettesítési értékének meghatározásakor, illetve a közönséges maximum kiválasztásnál. A maximum kiválasztásnak van egy általánosabb változata is (feltételes maximum keresés), amely csak egy logikai feltételnek megfelelő függvényértékek között keresi a legnagyobbat, és itt az is megengedett, ha egyetlen egy függvényérték sem elégíti ki ezt a feltételt, vagy az intervallum üres. Egy logikai változó jelzi, hogy egyáltalán volt-e vizsgált függvényérték.

Az algoritmusminták $f(i)$, $\beta(i)$, $h(i, y, y_1, \ldots, y_{k-1})$ kifejezései az f, β, h függvények adott argumentumú helyettesítési értékeit jelölik. Feltesszük, hogy ezek a helyettesítési értékek kiszámíthatóa

1. Összegzés

Az összegzés programozási tételét az előző alfejezetekben alaposan elemeztük. Most csak a teljesség igénye miatt ismételjük meg.

Feladat: Adott az egész számok egy [*m..n*] intervalluma és egy *f*:[*m..n*]→*H* függvény. A *H* halmaz elemein értelmezett egy asszociatív, baloldali nulla elemmel rendelkező művelet (nevezzük összeadásnak és jelölje ezt a +). Határozzuk meg az *f* függvény [*m..n*]-en felvett értékeinek az összegét, azaz a $\sum_{k=m}^{n} f(k)$ kifejezés értékét! (*m*>*n* esetén ennek az értéke definíció szerint a nulla elem).

Specifikáció:

$A = (m{:}\mathbb{Z}, n{:}\mathbb{Z}, s{:}H)$

$Ef = (m{=}m' \wedge n{=}n')$

$Uf = (Ef \wedge s = \sum_{i=m}^{n} f(i))$

Algoritmus:

$s, i := 0, m$	
$i \le n$	
	$s := s + f(i)$
	$i := i + 1$

2. Számlálás

Példaként gondoljunk arra a feladatra, amikor egy *20* fős osztályban azon diákok számát kell megadnunk, akik ötöst kaptak történelemből. A diákokat *1*-től *20*-ig megsorszámozzuk, és a történelem jegyeiket egy *1*-től *20*-ig indexelt *t* tömbben tároljuk: az *i*-edik diák osztályzata a $t[i]$. Definiálunk egy $\beta:[1..20]\to\mathbb{L}$ logikai függvényt, amely azokhoz a diákokhoz rendel *igaz* értéket, akik ötöst kaptak, más szóval $\beta(i) = (t[i]=5)$. Azt kell meghatároznunk, hogy a β által felvett értékek között hány *igaz* érték szerepel. Ilyenkor valójában *0* illetve *1* értékeket összegzünk attól függően, hogy a logikai kifejezés hamis vagy igaz. A számlálás tehát az összegzés speciális eseteként fogható fel, amelyben az $s:=s+1$ értékadást csak a $\beta(i)$ feltétel teljesülése esetén kell végrehajtani.

Feladat: Adott az egész számok egy $[m..n]$ intervalluma és egy $\beta:[m..n]\to\mathbb{L}$ feltétel. Határozzuk meg, hogy az $[m..n]$ intervallumon a β feltétel hányszor veszi fel az *igaz* értéket! (Ha $m>n$, akkor egyszer sem.)

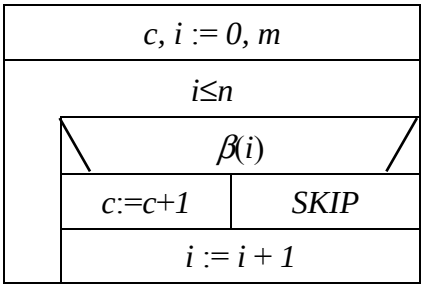
Specifikáció:

$A = (m:\mathbb{Z}, n:\mathbb{Z}, c:\mathbb{N})$

$Ef = (m=m' \wedge n=n')$

$$Uf = \left(Ef \wedge c = \sum_{\substack{i=m \\ \beta(i)}}^{n} 1 \right)$$

Algoritmus:



3. Maximum kiválasztás

Amikor olyan elemeket adunk meg (természetesen egy intervallumon értelmezett függvény által felvett értékekként), amelyeket valamilyen szempont szerint össze tudunk hasonlítani, akkor megkérdezhetjük, melyik elem a legnagyobb. A kérdés csak akkor válaszolható meg, ha egyfelől van legalább egy ilyen elem, másfelől bármelyik két elem összehasonlítható, és ez az összehasonlítás egy úgynevezett teljes rendezési reláció. Például arra vagyunk kíváncsiak, hogy egy 20 fős osztályban kinek van a legjobb jegye történelemből. A diákokat *1*-től *20*-ig megsorszámozzuk, és a történelem jegyeiket egy *1*-től *20*-ig indexelt *t* tömbben tároljuk: az *i*-edik diák osztályzata a *t*[*i*]. Ez a tömb tekinthető egy $f:[1..20]\to\mathbb{N}$ függvénynek ($f(i)=t[i]$), amely értékei között keressük a legnagyobbat.

Feladat: Adott az egész számok egy [*m..n*] intervalluma és egy $f:[m..n]\to H$ függvény. A *H* halmaz elemein értelmezett egy teljes rendezési reláció. Határozzuk meg, hogy az *f* függvény hol veszi fel az [*m..n*] nem üres intervallumon a legnagyobb értéket, és mondjuk meg, mekkora ez a maximális érték!

Specifikáció:

$A = (m:\mathbb{Z}, n:\mathbb{Z}, ind:\mathbb{Z}, max:H)$

$Ef = \ (m=m' \wedge n=n' \wedge n\geq m)$

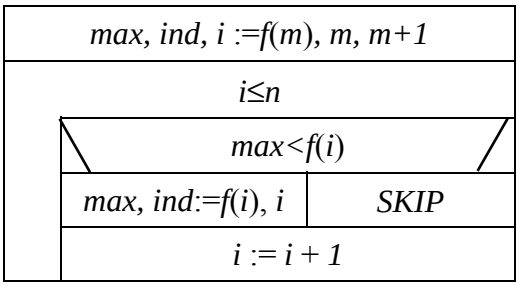
$Uf = \ (Ef \wedge ind\in[m..n] \wedge \ max=f(ind) \ \wedge \forall i\in[m..n]: max\geq f(i)) =$

másképpen jelölve, illetve rövidített jelölést használva

$$= (Ef \wedge ind\in[m..n] \wedge max = f(ind) = \max_{i=m}^{n} f(i)) =$$

$$= (Ef \wedge max= \max_{i=m}^{n} f(i) \wedge ind= \max_{i=m}^{n}{}_{f}\, i) = (Ef \wedge max,ind= \max_{i=m}^{n} f(i))$$

Algoritmus:



Konkrét esetekben nincs mindig szükség a maximális érték indexére. Ilyenkor az erre vonatkozó értékadásokat elhagyhatjuk. Tipikusan ilyen eset az, amikor az *f* függvény identitás, azaz egy index és annak értéke megegyezik, mert ekkor a maximális index és maximális érték is ugyanaz. A *max* változó elhagyása hatékonysági ok miatt nem javasolt még akkor sem, ha a feladat nem kíváncsi a maximális értékre.

4. Feltételes maximum keresés

A maximum kiválasztást sokszor úgy kellene elvégezni, hogy a vizsgált elemek közül csak azokat vesszük figyelembe, amelyek eleget tesznek egy adott feltételnek. Például, ha ismerjük néhány diák történelem jegyeit, és a nem jeles diákok között keressük azt, akinek a legtöbb ötöse van. Az ilyen esetben előfordulhat, hogy egyetlen elem sem elégíti ki a feltételt, azaz a vizsgálat tárgyát képező elemek halmaza üres. Ekkor azonban a maximum kiválasztás nem alkalmazható. Ezért adunk meg erre az esetre egy általánosabb maximum keresést.

Feladat: Adott az egész számok egy $[m..n]$ intervalluma, egy $f{:}[m..n]\to H$ függvény és egy $\beta{:}[m..n]\to\mathbb{L}$ feltétel. A H halmaz elemein értelmezett egy teljes rendezési reláció. Határozzuk meg, hogy az $[m..n]$ intervallum β feltételt kielégítő elemei közül az f függvény hol veszi fel a legnagyobb értéket, és mondjuk meg, mekkora ez az érték! (Lehet, hogy egyáltalán nincs β feltételt kielégítő elem az $[m..n]$ intervallumban vagy $m>n$.)

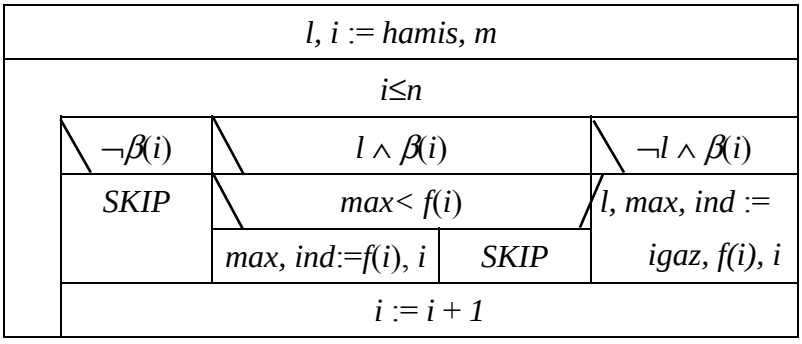
Specifikáció:

$$A = (m{:}\mathbb{Z},\ n{:}\mathbb{Z},\ l{:}\mathbb{L},\ ind{:}\mathbb{Z},\ max{:}H)$$
$$Ef = (\ m{=}m' \wedge n{=}n'\ )$$
$$Uf = (\ Ef \wedge (\ l = \exists i{\in}[m..n]{:}\beta(i)\ )\ \wedge$$
$$(\ l \to\ ind{\in}[m..n] \wedge\ max{=}f(ind)\ \wedge \beta(ind) \wedge (\forall i{\in}[m..n]{:}\beta(i) \to max{\geq}f(i)\ )\ )\ )$$

Itt is bevezetünk egy rövid jelölést. Ebben úgy tekintünk a feltételes maximumkeresés feladatára, mint egy olyan függvényre, amelyik három eredményt is ad: az első a logikai érték (van-e egyáltalán adott tulajdonságú elem az intervallumban), a második az adott tulajdonságú elemekre (f függvény által) adott értékek között a legnagyobb érték, a harmadik egy olyan adott tulajdonságú elem, amelyhez a legnagyobb érték tartozik

$$Uf = (\ Ef \wedge l,max,ind = \mathop{\boldsymbol{max}}_{\substack{i=m \\ \beta(i)}}^{n} f(i)\ )$$

Algoritmus:



A maximum kiválasztásnál említett egyszerűsítés itt is alkalmazható.

5. Kiválasztás (szekvenciális vagy lineáris kiválasztás)

Azokra a feladatokra gondoljunk, ahol egymás után sorba állított elemek között keressük az első adott tulajdonságút úgy, hogy tudjuk, hogy ilyen elem biztosan van, és így a vizsgálat során csak véges sok elemet kell ellenőrizni. Mivel a vizsgált elemeket mindig meg lehet sorszámozni, ez a feladat megfogalmazható úgy is, hogy az egész számok egy félegyenesén elindulva kell az első olyan elemet megtalálnunk, amelyikre egy adott feltétel teljesül. Például meg kell keresni az első *200*-nál nagyobb prím számot. Tudjuk, hogy ilyen biztosan van, így a 201-től elindulva egyenként megvizsgáljuk az egész számokat, amíg csak prímszámot nem találunk. Másik példa az, amikor egy osztályban keressük az első ötös történelem jeggyel rendelkező diákot, ha tudjuk, hogy ilyen biztosan van. A diákok történelem jegyeit ilyenkor egy *1*-től *20*-ig indexelt *t* tömbben tároljuk (az *i*-edik diák osztályzata a $t[i]$), ha a diákokat *1*-től *20*-ig sorszámoztuk meg. A keresés szempontjából azonban közömbös, hogy egy *20* fős osztállyal van-e dolgunk.

Felfogható az alapfeladat úgy is, hogy egy intervallum felső határát kell megkeresni, amelyik csak közvetve adott egy logikai állítás segítségével: az első olyan számot keressük, amelyikre az állítás igazat ad.

Feladat: Adott egy *m* egész szám és egy *m*-től jobbra értelmezett $\beta:\mathbb{Z}\to\mathbb{L}$ feltétel. Határozzuk meg, hogy az *m*-től jobbra eső első olyan számot, amely kielégíti a β feltételt, ha tudjuk, hogy ilyen szám biztosan van!

Specifikáció:

$A = (m{:}\mathbb{Z}, ind{:}\mathbb{Z})$
$Ef = (m{=}m' \wedge \exists k{\geq}m{:}\ \beta(k))$
$Uf = (Ef \wedge ind{\geq}m \wedge \beta(i) \wedge \forall k{\in}[m..ind{-}1]{:}\neg\beta(k))$

vagy rövidített jelölést alkalmazva:

$$Uf = (Ef \wedge ind = \underset{ind\geq m}{\textbf{select}}\ \beta(ind))$$

Algoritmus:

$ind{:=}m$
$\neg\beta(ind)$
$ind := ind{+}1$

6. Keresés (szekvenciális vagy lineáris keresés)

Példaként tekintsünk véges számú, sorba állított elemet, amelyek között az első adott tulajdonságút keressük. Mivel a vizsgált elemeket mindig meg lehet sorszámozni, ez a feladat megfogalmazható úgy is, hogy az egész számok egy intervallumán kell balról az első olyan elemet megtalálnunk, amelyikre az adott feltétel teljesül. Keressük meg például egy *20* fős osztályban az első olyan diákot, akinek a történelem jegye ötös. A diákokat *1*-től *20*-ig megsorszámozzuk, és a történelem jegyeiket egy *1*-től *20*-ig indexelt *t* tömbben tároljuk: az *i*-edik diák osztályzata a $t[i]$. Definiálunk egy $\beta:[1..20]\to\mathbb{L}$ logikai függvényt, amely azokhoz a diákokhoz rendel *igaz* értéket, akik ötöst kaptak, más szóval $\beta(i) = (t[i]=5)$.

Feladat: Adott az egész számok egy $[m..n]$ intervalluma és egy $\beta:[m..n]\to\mathbb{L}$ feltétel. Határozzuk meg az $[m..n]$ intervallumban balról az első olyan számot, amely kielégíti a β feltételt!

Specifikáció:

$A = (m:\mathbb{Z}, n:\mathbb{Z}, l:\mathbb{L}, ind:\mathbb{Z})$
$Ef = (m=m' \wedge n=n')$
$Uf = (Ef \wedge (l=\exists i\in[m..n]: \beta(i)) \wedge (l\to ind\in[m..n]\wedge\beta(ind)\wedge\forall i\in[m..ind-1]:\neg\beta(i)))$

vagy rövidített jelölést alkalmazva

$$Uf = (Ef \wedge l, ind = \overset{n}{\underset{i=m}{\textbf{search}}}\,\beta(i))$$

Algoritmus[1]:

l, i:= *hamis, m*	
$\neg l \wedge i\le n$	
	$l := \beta(i)$
	ind:=i
	$i := i+1$

[1] A lineáris keresés algoritmusmintájának többféle változata is ismert. A mi céljainkhoz legjobban a fent bevezetett változat illeszkedik.

l, ind := *hamis, m–1*	
$\neg l \wedge ind<n$	
	$ind := ind+1$
	$l := \beta(ind)$

l, ind := *hamis, m*		
$\neg l \wedge ind\le n$		
	$\beta(ind)$	
	l:=*igaz*	$ind := ind+1$

ind := *m*	
$ind\le n \wedge \neg\beta(ind)$	
	$ind := ind+1$
$l := ind\le n$	

7. Logaritmikus keresés

Az alábbi mintafeladatot meg lehet oldani a korábban bemutatott lineáris keresés segítségével is, mert ez a mintafeladat az ott látottnak speciális esete. De ha kihasználjuk a speciális tulajdonságokat, akkor a lineáris keresésnél látott algoritmusnál jóval gyorsabb kereső algoritmust tudunk előállítani. Amíg a lineáris keresés egy h hosszú intarvallumot legrosszabb esetben h lépésben tud átvizsgálni (a lépések száma a hosszúság lineáris függvénye), addig a most bemutatásra kerülő keresésnek legrosszabb esetben ehhez csak $log_2 h$ lépésre van szüksége.

A keresés ötlete az, hogy ha monoton növekedően rendezett értékek között keresünk egy adott értéket, akkor a keresési intervallumot felezzük el, és a felező pontnál levő értéket vessük össze a keresettel. Ha megegyeznek, akkor megtaláltuk a keresett értéket, ha a vizsgált érték nagyobb a keresett értéknél, akkor a keresett érték (ha egyáltalán ott van az értékek között) a keresési intervallum első felében fordulhat csak elő (a rendezettség miatt), ha kisebb, akkor a második felében.

Feladat: Adott az egész számok egy [*m..n*] intervalluma és egy $f:\mathbb{Z}\rightarrow H$ függvény, amelyik az [*m..n*] intervallumon monoton növekvő. (A H halmaz elemei között értelmezett egy rendezési reláció.) Keressünk meg a függvény [*m..n*] intervallumon felvett értékei között egy adott értéket!

Specifikáció:

$A = (m:\mathbb{Z}, n:\mathbb{Z}, h:\mathbb{Z}, l:\mathbb{L}, ind:\mathbb{Z})$
$Ef = (m=m' \wedge n=n' \wedge h=h' \wedge \forall j \in [m..n-1]: f(j) \leq f(j+1))$
$Uf = (Ef \wedge l=(\exists j \in [m..n]: f(j)=h) \wedge l \rightarrow (ind \in [m..n] \wedge f(ind)=h))$

Algoritmus:

<table>
<tr><td colspan="4">ah,fh,l :=m,n,hamis</td></tr>
<tr><td colspan="4">¬l ∧ ah≤fh</td></tr>
<tr><td></td><td colspan="3">ind :=(ah+fh) div 2</td></tr>
<tr><td></td><td>f(ind)>h</td><td>f(ind)<h</td><td>f(ind)=h</td></tr>
<tr><td></td><td>fh := ind-1</td><td>ah := ind+1</td><td>l := igaz</td></tr>
</table>

8. Rekurzív függvény kiszámítása

Tekintsük a Fibonacci számokat: *1, 1, 2, 3, 5, 8, 12,….* Ezeket a $Fib:\mathbb{N}\rightarrow\mathbb{N}$ függvény állítja elő. Az első két Fibonacci szám definíció szerint az *1*, azaz $Fib(1)=1$ és $Fib(2)=1$. A további Fibonacci számokat a megelőző két Fibonacci szám összegeként kapjuk meg: $Fib(i)=Fib(i-1)+Fib(i-2)$. Ha az *n*-edik Fibonacci számhoz szeretnénk eljutni ($n>2$), akkor először a harmadik, utána a negyedik és így tovább Fibonacci számot kell előállítani, azaz végig kell mennünk a *3..n* intervallumon. Másik példa az *n* faktoriálisának meghatározása. A $Fact:\mathbb{N}\rightarrow\mathbb{N}$ függvény az *1*-hez definíció szerint az *1*-et rendel. Az *1*-nél nagyobb számok faktoriálisa a megelőző faktoriális segítségével állítható elő: $Fact(i)=i*Fact(i-1)$. Az *n* szám faktoriálisának meghatározásához ($n>1$) sorban egymás után meg kell határoznunk a *2..n* intervallum elemeinek faktoriálisát.

Fenti példákban szereplő függvények *i*-dik helyen felvett helyettesítési értékét a megelőző néhány függvényérték és az *i* szám alapján, azaz rekurzív módon számolhatjuk ki. Azt, hogy hány megelőző függvényértékre van szükség, a rekurzió rendjének nevezzük. Láthatjuk, hogy minden számításhoz tartozik egy *m..n* intervallum. Ahhoz, hogy a függvény értékét az *n* egész számnál meghatározzuk, sorban egymás után ki kell számolnunk a függvény értékeit az *m..n* intervallum összes elemére. Az *m* számra viszont csak úgy tudjuk kiszámolni, ha közvetlenül ismerjük a függvény *m*-nél kisebb helyen felvett néhány értékét, szám szerint annyit, amennyi a rekurzió rendje. Az *m* szám a rekurzió bázisa.

Feladat: Legyen az $f:\mathbb{Z}\rightarrow H$ egy *k*-ad rendű *m* bázisú rekurzív függvény (*m* egész szám, *k* pozitív egész szám), azaz $f(i) = h(i, f(i-1), \ldots, f(i-k))$ ahol $i\geq m$ és a *h* egy $\mathbb{Z}\times H^k\rightarrow H$ függvény. Ezen kívül $f(m-1)=e_{m-1}, \ldots, f(m-k)= e_{m-k}$, ahol $e_{m-1}, \ldots, e_{m-k}$ *H*-beli értékek. Számítsuk ki az *f* függvény adott *n* ($n\geq m$) helyen felvett értékét!

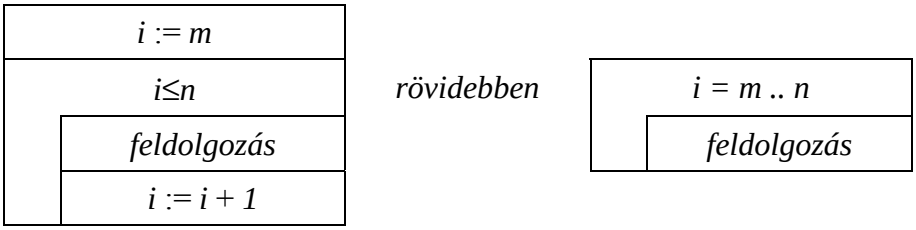
Specifikáció:

$A = (n:\mathbb{Z},\ y:H)$
$\quad n \quad y$
$Ef = (n=n' \wedge n\geq m)$
$Uf = (Ef \wedge y=f(n))$

Algoritmus:

$y, y_1, \ldots, y_{k-1}, i := e_{m-1}, e_{m-2}, \ldots, e_{m-k}, m$
$i\leq n$
$y, y_1, y_2, \ldots, y_{k-1} := h(i, y, y_1, \ldots, y_{k-1}), y, y_1, \ldots, y_{k-2}$
$i := i + 1$

A kiválasztás, lineáris keresés és a logaritmikus keresés kivételével a fenti algoritmus minták ciklusszervezése megegyezik: egy indexváltozót vezetnek végig az *m..n* intervallumon. Az ilyen esetekben alkalmazhatunk egy rövidített jelölést az algoritmus felírására. Ezt a változatot szokták számlálós ciklusnak nevezni. Az elnevezés kicsit félrevezető, hiszen, mint látjuk, ez egy kezdeti értékadásnak és egy ciklusnak a szekvenciája. Jelentőségét egyrészt az adja, hogy ennél a ciklusnál pontosan meg lehet mondani, hány iteráció után fog leállni a ciklus, másrészt a különféle programozási nyelvek külön nyelvi elemet (például „for”) szoktak biztosítani a kódolásukhoz.



A továbbiakban ott, ahol lehetőség van rá, mi is ezt a rövidebb változatot fogjuk használni.