

Bevezetés a programozásba I

3. gyakorlat

PLanG: Programozási tételek

© 2011.09.27. Giachetta Roberto
groberto@inf.elte.hu
<http://people.inf.elte.hu/groberto>

Programozási tételek

Algoritmusok

- *Algoritmus*nak nevezzük azt a műveletsorozatot, amely a feladat megoldásához vezet
 - a program lényegi része, amely nem tartalmazza az adatok beolvasását és kiírását
 - egy programban több algoritmus is szerepelhet, amelyek valamilyen kombinációja oldja meg a feladatot
- A megoldandó feladatokban gyakorta fedezünk fel hasonlóságokat (pl. többen valamit kell keresni)
 - ennek köszönhetően a megoldó algoritmusok is teljesen hasonló, csupán néhány eltérést fedezhetünk fel közöttük
 - általában a megfelelő adat, illetve feltétel változtatásokkal megkapjuk az új feladat megoldását a korábbi alapján

Programozási tételek

Algoritmusok

- Az algoritmusokat ezért célszerű általánosan (absztraktnan) megfogalmazni, hogy a változtatások (transzformációk) könnyen véghezvihetők legyenek
 - amennyiben a feladatra találunk megoldó algoritmust, és azt átalakítjuk az aktuális feladatra, akkor azt mondjuk, hogy a feladatot *visszavezettük az algoritmusra*
 - az algoritmus lehet nagyon egyszerű (pl. szám szorzása), és nagyon összetett
- Az algoritmust két részre szeparáljuk:
 - *inicializálás*: változók kezdőértékeinek megadása
 - *feldolgozás* (mag): műveletvégzés a bemenő adatokkal és az inicializált változókkal

Programozási tételek

Algoritmusok

- Algoritmusokat azért célszerű használni, mert jó, bizonyított megoldást adják a feladatnak, nem kell újabb algoritmust kitalálni
 - már több ezer algoritmus létezik, amelyek mind nevesítettek
 - az algoritmusok bemenete általában egy adatsorozat, vagyis adatok egymásutánja (pl. tömbben)
- Az egyszerű, sorozatokra alkalmazott algoritmusokat nevezzük *programozási tételek*nek, ezek a következők:
 - összegzés, számlálás
 - lineáris keresés, bináris keresés
 - maximum keresés, feltételes maximumkeresés
 - elemenkénti feldolgozás

Programozási tételek

Összegzés

- Az *összegzés programozási tétele* lehetővé teszi tetszőleges sorozat (a_1, \dots, a_n) adott függvény (f) szerint vett értékének összesítését (*sum*)

$$sum = \sum_{i=1}^n f(a_i)$$

- az összegzés egy ciklusban történik, az összeget egy külön változóhoz adjuk hozzá minden lépésben, amelyet egy kezdeti értéken inicializálunk
- általában az összegző művelet az összeadás, ekkor az összeg változó 0-ról indul
- a függvény általában az identitás, de lehet nagyon összetett is

Programozási tételek

Összegzés

VÁLTOZÓK:

```
a : <forrás típusa>
sum : <eredmény típusa>
```

```
sum := <kezdőérték>
```

```
** általában: sum := 0
```

```
CIKLUS AMÍG <nincs vége a sorozatnak>
```

```
<a értéket kap> ** amennyiben szükséges
```

```
sum := sum <összegző művelet> <f(a)>
```

```
** az összegző művelettel hozzávesszük a
```

```
** függvény által adott értéket
```

```
** általában: sum := sum + a
```

```
CIKLUS_VÉGE
```

Programozási tételek	
Összegzés	
<p><i>Feladat:</i> Adjuk meg az első n természetes szám összegét.</p> <ul style="list-style-type: none"> • ehhez az összegzés tételére van szükségünk, a forrás és az eredmény típusa egész, a művelet az összegzés, a kezdőérték a 0 • az értéket nem kell külön beolvasnunk, de n értékét a felhasználótól kérjük be • egy számláló ciklusra lesz szükségünk, amely elmegy 1-től az n értékéig <p><i>Specifikáció:</i></p> <ul style="list-style-type: none"> • bemenet: n értéke. • kimenet: a számok összege n-ig a sum változóban 	
PPKE ITK, Bevezetés a programozásba I	3:7

Programozási tételek	
Összegzés	
<p><i>Megoldás:</i></p> <pre>PROGRAM osszegzes VÁLTOZÓK: i, n, sum : EGÉSZ i := 1 ** a ciklus 1-től indul BE: n sum := 0 CIKLUS AMÍG (i <= n) ** nincs szükség elembekérésre sum := sum + i ** elvégezzük az összegzést i := i + 1 ** léptetjük a ciklusváltozót CIKLUS_VÉGE KI: "Az első ", n, " szám összege: ", sum PROGRAM_VÉGE</pre>	
PPKE ITK, Bevezetés a programozásba I	3:8

Programozási tételek	
Összegzés	
<p><i>Feladat:</i> Szorozzuk össze egy 10 elemű valós vektor elemeit.</p> <ul style="list-style-type: none"> • ehhez az összegzés tételére van szükségünk a szorzás műveletével • ekkor az összegző változót 1-re kell inicializálnunk, mivel szorzásra az a neutrális elem • számláló ciklus kell a bejáráshoz <p><i>Specifikáció:</i></p> <ul style="list-style-type: none"> • bemenet: adatok egy valós tömbben (a) • kimenet: számok szorzata (sum) 	
PPKE ITK, Bevezetés a programozásba I	3:9

Programozási tételek	
Összegzés	
<p><i>Megoldás:</i></p> <pre>PROGRAM szorzat VÁLTOZÓK: a : VALÓS[10], sum : VALÓS, i : EGÉSZ i := 0 sum := 1 ** a szorzást egyről indítjuk CIKLUS AMÍG (i < 10) BE: a[i] ** beolvassuk az i. elemet sum := sum * a[i] ** elvégezzük az összegzést i := i + 1 CIKLUS_VÉGE KI: "Az elemek szorzata: ", sum PROGRAM_VÉGE</pre>	
PPKE ITK, Bevezetés a programozásba I	3:10

Programozási tételek	
Számolás	
<ul style="list-style-type: none"> • A <i>számlálás programozási tétele</i> lehetővé teszi, hogy tetszőleges (a_1, \dots, a_n) sorozatban megsámoljuk egy adott (logikai) felvételt (β) teljesítő elemek számát (c) $c = \sum_{i=1}^n 1_{\beta(a_i)}$ <ul style="list-style-type: none"> • a feltétel tetszőleges logikai értékű függvény • lényegében az összegzés egy speciális esetét végezzük, ahol vagy 1-t, vagy 0-t adunk hozzá az eddigi összeghez, függően a feltétel teljesülésétől • ezt a végrehajtásban elágazás segítségével valósítjuk meg 	
PPKE ITK, Bevezetés a programozásba I	3:11

Programozási tételek	
Számolás	
<p>VÁLTOZÓK:</p> <pre>a : <forrás típusa> c : EGÉSZ ** egész számba számlálunk</pre> <p>$c := 0$ ** a számláló mindig 0-ról indul</p> <pre>CIKLUS AMÍG <nincs vége a sorozatnak> <a értéket kap> ** amennyiben szükséges HA <a teljesíti a feltételt> AKKOR c := c + 1 ** mindig eggyel nő a számláló HA_VÉGE CIKLUS_VÉGE</pre>	
PPKE ITK, Bevezetés a programozásba I	3:12

Programozási tételek	
Számolás	
<p><i>Feladat:</i> Generáljuk 100 véletlen számot 1 és 10 között, és számoljuk meg, hány páratlan van közöttük.</p> <ul style="list-style-type: none"> • használjuk a számolás tételét, ahol a páratlanság biztosítja a feltételt (oszthatóságot a moduló segítségével ellenőrizhetünk, 2-vel osztva 1-t ad maradékul) • a számot 0-9 közé generáljuk, majd hozzáadunk egyet • szükségünk lesz egy számláló ciklusra <p><i>Specifikáció:</i></p> <ul style="list-style-type: none"> • bemenet: véletlenszerűen generált számok 1 és 10 között (<i>a</i>) • kimenet: a páratlan számok száma (<i>c</i>) 	
PPKE ITK, Bevezetés a programozásba I	3:13

Programozási tételek	
Számolás	
<p><i>Megoldás:</i></p> <pre>PROGRAM paratlan_szamok VÁLTOZÓK: i, a, c : EGÉSZ c := 0 i := 0 CIKLUS AMÍG (i < 100) a := RND 10 + 1 ** 1 és 10 közötti szám generálása HA (a MOD 2 = 1) AKKOR c := c + 1 HA_VÉGE</pre>	
PPKE ITK, Bevezetés a programozásba I	3:14

Programozási tételek	
Számolás	
<p><i>Megoldás:</i></p> <pre>i := i + 1 CIKLUS_VÉGE KI: "Összesen ", c, " páratlan szám volt." PROGRAM_VÉGE</pre>	
PPKE ITK, Bevezetés a programozásba I	3:15

Programozási tételek	
Lineáris keresés	
<ul style="list-style-type: none"> • A <i>lineáris keresés</i> programozási tétele segítségével megállapíthatjuk, hogy van-e egy sorozatban egy adott feltételt teljesítő elem, és amennyiben több van, melyik az első ilyen elem. • a teljesüléshez szükségünk van egy logikai változóra (<i>l</i>), és megadhatjuk magát az értéket, vagy a helyét • amennyiben már teljesült a feltétel, akkor nincs értelme végignézni a további értékeket, hiszen a teljesülést nem befolyásolják • ezért a ciklust korábban is terminálhatjuk úgy, hogy a logikai változót is bevesszük a ciklusfeltételbe 	
PPKE ITK, Bevezetés a programozásba I	3:16

Programozási tételek	
Lineáris keresés	
<pre>VÁLTOZÓK: a : <forrás típusa> l : LOGIKAI ** van-e keresett elem a sorozatban ind : <forrás típusa, vagy EGÉSZ> ** a teljesítő elem, vagy annak helye l := HAMIS ** kezdetben feltételezzük, hogy nincs CIKLUS AMÍG (NEM l ÉS <nincs vége a sorozatnak>) ** ha l teljesül, nem megyünk végig <a értéket kap> ** amennyiben szükséges HA <a teljesíti a feltételt> AKKOR l := IGAZ ** ezután kilép a ciklusból ind := <a, vagy a helye> HA_VÉGE CIKLUS_VÉGE</pre>	
PPKE ITK, Bevezetés a programozásba I	3:17

Programozási tételek	
Lineáris keresés	
<p><i>Feladat:</i> 100 napon át megmértük a hőmérsékletet, mértünk-e fagypon alatti értéket, és ha igen, hányadik napon először.</p> <ul style="list-style-type: none"> • valós értékekkel dolgozunk, ugyanakkor az érték helye egész lesz • számláló ciklust alkalmazunk, amely 100 lépést tesz meg • lineáris keresést alkalmazunk, a feltétel a negatív érték <p><i>Specifikáció:</i></p> <ul style="list-style-type: none"> • bemenet: 100 valós szám (<i>h</i>) • kimenet: van-e negatív érték (<i>l</i>), és ha igen, hol (<i>ind</i>) 	
PPKE ITK, Bevezetés a programozásba I	3:18

Programozási tételek	
Lineáris keresés	
<i>Megoldás:</i>	
<pre>PROGRAM negativ_ertek_kereses VÁLTOZÓK: i, ind : EGÉSZ, h : VALÓS, l : LOGIKAI i := 0 l := HAMIS CIKLUS AMÍG (NEM l ÉS i < 100) BE: h HA (h < 0) AKKOR l := IGAZ ind := i</pre>	
PPKE ITK, Bevezetés a programozásba I	3:19

Programozási tételek	
Lineáris keresés	
<i>Megoldás:</i>	
<pre>HA_VÉGE i := i + 1 CIKLUS_VÉGE ** a kiírás függ attól, találtunk-e egyáltalán ** negatív értéket HA l AKKOR KI: "A(z) ", ind, ". helyen volt negatív." KÜLÖNBEN KI: "Nem volt negatív érték." HA_VÉGE PROGRAM_VÉGE</pre>	
PPKE ITK, Bevezetés a programozásba I	3:20

Programozási tételek	
Maximum keresés	
<ul style="list-style-type: none"> Egy sorozat valamilyen szempont szerinti szélső értékét a <i>maximumkeresés</i> programozási tételével állapíthatjuk meg. <ul style="list-style-type: none"> a szélső érték lehet maximum, vagy minimum (ehhez csak a relációt kell megcserélnünk) megadja a maximum értékét, illetve helyét mindig összehasonlítjuk az új elemet a sorozat eddigi részének maximumával, és ha nagyobb nála, akkor ő lesz az új maximum ehhez a maximum értéket kezdetben valamilyen extrémális értéktől indíthatjuk el (pl. ha csak pozitív számok vannak, akkor 0-tól), vagy rögtön ráállítjuk a sorozat első elemére, így a sorozat második elemétől indul a ciklus 	
PPKE ITK, Bevezetés a programozásba I	3:21

Programozási tételek	
Maximum keresés	
<pre>VÁLTOZÓK: a, ind : <forrás típusa> max : <eredmény típusa> <első x értéket kap> ind := a ** érték és hely az első elemre max := <f(a)> CIKLUS AMÍG <nincs vége a sorozatnak> <a értéket kap> ** amennyiben szükséges HA <f(a) szélsőbb, mint max> AKKOR ind := a ** felvesszük értékét és helyét max := <f(a)> HA_VÉGE CIKLUS_VÉGE</pre>	
PPKE ITK, Bevezetés a programozásba I	3:22

Programozási tételek	
Maximum keresés	
<p><i>Feladat:</i> Minden nap megmértük a hőmérséklet egészen addig, amíg fagypont alá nem estek. Keressük meg az addig tartó tartomány legnagyobb értékét, és hogy hányadik.</p> <ul style="list-style-type: none"> maximumkeresést alkalmazunk előtesztelő ciklusban dolgozunk, amely 0-nál kisebb érték esetén megáll bár nem számláló ciklust használunk, számolnunk kell a napokat, hogy megállíthassuk a maximum helyét 	
<i>Specifikáció:</i>	
<ul style="list-style-type: none"> bemenet: valós számok (<i>h</i>) sorozata kimenet: a számok legnagyobb értéke (<i>max</i>), és helye (<i>ind</i>) 	
PPKE ITK, Bevezetés a programozásba I	3:23

Programozási tételek	
Maximum keresés	
<i>Megoldás:</i>	
<pre>PROGRAM maximum_ertek VÁLTOZÓK: i, ind : EGÉSZ, ** i segítségével számoljuk a napokat h, max : VALÓS BE: h ** első elem bekérése i := 1 ** ez az első nap mérése ind := 1 ** érték és hely az első elemre max := h CIKLUS AMÍG (h >= 0) BE: h i := i + 1 ** a napokat is számoljuk közben</pre>	
PPKE ITK, Bevezetés a programozásba I	3:24

Programozási tételek	
Maximum keresés	
<i>Megoldás:</i>	
<pre> HA (h > max) AKKOR ** ha nagyobb az új érték ind := i ** felvesszük értékét és helyét max := h HA_VÉGE CIKLUS_VÉGE KI: "A legnagyobb hőmérséklet: ", max, ", a(z)", ind, ". napon mértük. " PROGRAM_VÉGE </pre>	
PPKE ITK, Bevezetés a programozásba I	3:25

Programozási tételek	
Elemenkénti feldolgozás	
<ul style="list-style-type: none"> Az eddigi feladatok mind olyanok voltak, hogy bekértük a sorozat egy értékét, és azt valamilyen módon feldolgoztuk (összegeztük, kerestük, ...), az ilyen jellegű feladatokat nevezzük elemenkénti feldolgozásoknak Az <i>elemenkénti feldolgozás programozási tétele</i> általánosítása az eddigi tételeknek, mely csak azt mondja meg, hogy a sorozatot elemenként kezeljük 	
VÁLTOZÓK: <pre> x : <forrás típusa> CIKLUS AMÍG <nincs vége a sorozatnak> <x elem bekérése> <x elem feldolgozása> CIKLUS_VÉGE </pre>	
PPKE ITK, Bevezetés a programozásba I	3:26

Programozási tételek	
Egymásba ágyazott ciklusok	
<ul style="list-style-type: none"> Egy ciklusmagba bármilyen utasítást tehetünk, így akár másik ciklust is, ekkor ezeket <i>egymásba ágyazott ciklusoknak</i> nevezzük <ul style="list-style-type: none"> az első ciklus lesz a külső, a másik a belső ciklus a belső ciklus működhet teljesen függetlenül a külső ciklustól, más ciklusfeltétellel, vagy ciklusszámlálóval amennyiben több számláló ciklust ágyazunk egymásba, ügyeljünk arra, hogy a külső ciklus számlálóját a külső ciklusban, a belső ciklus számlálóját a belső ciklusban változtassuk (de a külsőben inicializáljuk), persze a külső ciklus számlálóját a belsőben is el tudjuk érni, és fel tudjuk használni 	
PPKE ITK, Bevezetés a programozásba I	4:27

Programozási tételek	
Elemenkénti feldolgozás	
<i>Feladat:</i> Olvassunk be a bemeneten kapott karaktereket, amíg 'q'-t nem találunk, és minden karaktert írjunk ki 10-szer a kimenetre. <ul style="list-style-type: none"> használjunk elemenkénti feldolgozást egy ciklusban olvasnunk kell a bemenetet q-ig, ez lehet utántesztelő ciklus, ekkor persze a q karaktert is kiírjuk tízszer egy másik, belső ciklusban ki kell írunk minden karaktert 10-szer, ezt számláló ciklussal oldjuk meg, ehhez kell egy ciklusszámláló 	
PPKE ITK, Bevezetés a programozásba I	3:28

Programozási tételek	
Elemenkénti feldolgozás	
<i>Specifikáció:</i>	
<ul style="list-style-type: none"> bemenet: karakterek (<i>kar</i>) sorozata kimenet: minden karakterből 10 példány 	
<i>Megoldás:</i>	
<pre> PROGRAM tiz_karakter VÁLTOZÓK: kar: KARAKTER, i: EGÉSZ </pre>	
PPKE ITK, Bevezetés a programozásba I	4:29

Programozási tételek	
Elemenkénti feldolgozás	
<i>Megoldás:</i>	
<pre> CIKLUS ** külső ciklus BE: kar ** beolvasás a külső ciklusban ** kar feldolgozása: i := 0 ** számláló nullázása a külső ciklusban CIKLUS AMÍG (i < 10) ** belső ciklus KI: kar ** kiíratás a belsőben i := i + 1 CIKLUS_VÉGE ** belső ciklus vége AMÍG (kar /= 'q') ** külső ciklus vége PROGRAM_VÉGE </pre>	
PPKE ITK, Bevezetés a programozásba I	4:30

Programozási tételek Egymásba ágyazása	
<ul style="list-style-type: none"> Mint hogy ciklusokat is tudunk egymásba ágyazni, ez algoritmusokkal is megtehető, így lehetőségünk van egy programozási tétel feldolgozása közben egy másik programozási tételt teljes egészében lefuttatni <ul style="list-style-type: none"> a külső ciklus a külső programozási tételé, a belső a belsőé a külső programozási tételben kihasználhatjuk a belső által megadott értékeket tipikus eset, amikor a belső tételben összegzést, vagy számlálást végzünk, míg a külsőben lineáris keresést, vagy maximumkeresést 	3:31
PPKE ITK, Bevezetés a programozásba I	

Programozási tételek Egymásba ágyazása	
<p><i>Feladat:</i> Egy bankszámla pénzmozgásait (betétek és kivétel) egy 100 elemű tömbben tartjuk nyilván. Hányadik forgalom után volt a legmagasabb a számlaegyenleg, és mennyi volt az?</p> <ul style="list-style-type: none"> a feladat megoldásához maximumkeresést kell használnunk, de előtte összegeznünk kell a számla addigi forgalmait ezért be kell ágyaznunk a maximum keresésbe az összegzést <p><i>Specifikáció:</i></p> <ul style="list-style-type: none"> bemenet: valós számok tömbje (<i>t</i>) kimenet: a legmagasabb számlaegyenleg ideje (<i>ind</i>) és helye (<i>max</i>) 	3:32
PPKE ITK, Bevezetés a programozásba I	

Programozási tételek Egymásba ágyazása	
<p><i>Megoldás:</i></p> <pre> PROGRAM legnagyobb_szamla_egyenleg VÁLTOZÓK: forgalmak : EGÉSZ[100] i, j, ind, max, sum : EGÉSZ ** adatok beolvasása: i := 0 CIKLUS AMÍG (i < 100) BE : forgalmak[i] i := i + 1 CIKLUS_VÉGE </pre>	3:33
PPKE ITK, Bevezetés a programozásba I	

Programozási tételek Egymásba ágyazása	
<p><i>Megoldás:</i></p> <pre> max := 0 ** maximum keresés inicializálás ind := -1 i := 0 CIKLUS AMÍG (i < 100) j := 0 ** belső ciklus: j index sum := 0 ** összegzés inicializálás CIKLUS AMÍG (j < i) ** j csak i-ig megy, így mindig a megfelelő ** időpontig kapjuk meg a számlaegyenleget sum := sum + forgalmak[j] j := j + 1 CIKLUS_VÉGE </pre>	3:34
PPKE ITK, Bevezetés a programozásba I	

Programozási tételek Egymásba ágyazása	
<p><i>Megoldás:</i></p> <pre> HA (sum > max) ** maximum keresés max := sum ind := i HA_VÉGE i := i + 1 CIKLUS_VÉGE KI: "A legnagyobb egyenleg: ", max, ", a(z) ", ind, ". forgalom után volt." PROGRAM_VÉGE </pre>	3:35
PPKE ITK, Bevezetés a programozásba I	

Programozási tételek Feladatok	
<p><i>III. Programozási tételek:</i></p> <ol style="list-style-type: none"> Számítsd ki egy szám faktoriálisát. Add meg egy tetszőleges egész szám valódi osztóinak a számát. (*) Add meg, hogy az A és B közötti egész számok közül melyiknek van a legtöbb valódi osztója. 15. a) Egy szigorúan növekvő egész számsorban add meg a legnagyobb ugrást (szomszédos elemek közötti legnagyobb előforduló különbséget). 21. Egy tetszőleges szövegről add meg, hány kis "a" betű van benne. 	3:36
PPKE ITK, Bevezetés a programozásba I	

Programozási tételek

Feladatok

III. Programozási tételek:

23. (*) Add meg egy tetszőleges szövegben, hogy melyik karakter fordul elő benne a legtöbbször.
27. Add meg egy tetszőleges szöveg leghosszabb szavát.