Programozás
9. előadás

Tartalom

- Egymásba ágyazott programozási tételek
  (a 7. előadás folytatása)

- Tesztek előállítása
- Hibakeresés
- Hibajavítás
- Dokumentálás

- Programkészítési elvek

Egymásba ágyazott programozási tételek

- Független intervallum
  - Megszámolás ⊃ eldöntés – 1
  - Megszámolás ⊃ eldöntés – 2
  - Maximumkiválasztás ⊃ megszámolás
- Közös intervallum
  - Maximumkiválasztás ⊃ összegzés
  - Eldöntés ⊃ eldöntés

Egymásba ágyazott tételek
Független intervallum

Egy egész számokat tartalmazó mátrixban hány olyan sor van, ami csak páros számokat tartalmaz?

	1	2	3	m=4
1	1	2	4	3
2	2	6	4	2
3	5	3	2	1
4	2	8	2	8
n=5	5	3	2	3

Megszámolásban (optimista) eldöntés

Egymásba ágyazott tételek
Független intervallum

Egy egész számokat tartalmazó mátrixban hány olyan sor van, ami csak páros számokat tartalmaz?

Megszámolásban (optimista) eldöntés

Specifikáció:

- Bemenet: $n, m: Egész, x: Tömb(1..n, 1..m: Egész)$
- Kimenet: $db: Egész$
- Előfeltétel: $n = n' \text{ és } m = m' \text{ és } x = x' \text{ és } n, m \geq 0$
- Utófeltétel: Ef és $db = \sum_{i=1}^{n}{}_{csakpáros(i)} 1$
- Definíció: $csakpáros: Egész \rightarrow Logikai$
  $csakpáros(i) = \forall_{j=1}^{m} 2 | x[i,j]$

	1	2	3	m=4
1	1	2	4	3
2	2	6	4	2
3	5	3	2	1
4	2	8	2	8
n=5	5	3	2	3

Egymásba ágyazott tételek

Független intervallum

	1	2	3	m=4
1	1	2	4	3
2	2	6	4	2
3	5	3	2	1
4	2	8	2	8
n=5	5	3	2	3

Specifikáció:

- Utófeltétel: Ef és $db = \sum_{i=1}^{n} \underset{csakpáros(i)}{1}$
- Definíció: $csakpáros: Egész \rightarrow Logikai$

  $csakpáros(i) = \forall_{j=1}^{m} 2|x[i,j]$

$$db = \sum_{\substack{i=e \\ T(i)}}^{u} 1$$

$$mind = \forall_{i=e}^{u} T(i)$$

Megszámolás		
$e..u$	$\sim$	$1..n$
$T(i)$	$\sim$	$csakpáros(i)$

(Optimista) eldöntés		
$mind$	$\sim$	$csakpáros(i)$
i	$\sim$	j
$e..u$	$\sim$	$1..m$
$T(i)$	$\sim$	$2\|x[i,j]$

Egymásba ágyazott tételek
Független intervallum

	1	2	3	m=4
1	1	2	4	3
2	2	6	4	2
3	5	3	2	1
4	2	8	2	8
n=5	5	3	2	3

A belső tétel egy külön részfeladat

Specifikáció (csakpáros):

- Bemenet: $n, m: Egész, x: Tömb(1..n, 1..m: Egész), i: Egész$
- Kimenet: $csakpáros(i): Logikai$
- Előfeltétel: $n = n' \text{ és } m = m' \text{ és } x = x' \text{ és } i = i' \text{ és } n, m \geq 0$
- Utófeltétel: $Ef \text{ és } csakpáros(i) = \forall_{j=1}^{m} 2 | x[i, j]$

(Optimista) eldöntés		
i	~	j
$e..u$	~	$1..m$
$T(i)$	~	$2 \mid x[i,j]$

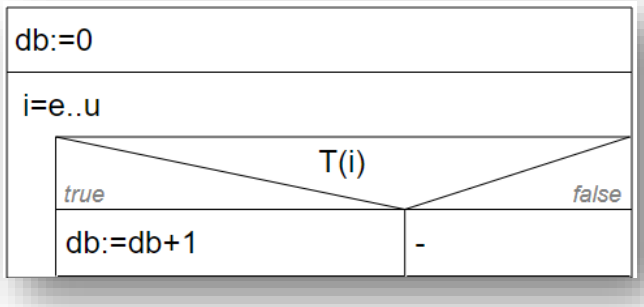
$$mind = \forall_{i=e}^{u} T(i)$$

Egymásba ágyazott tételek

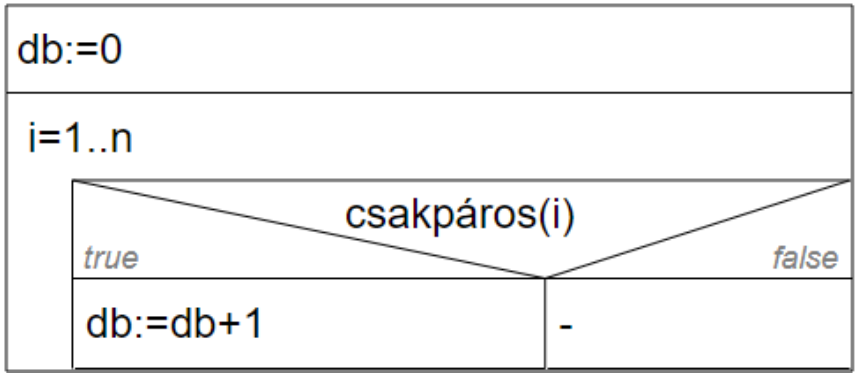
Független intervallum

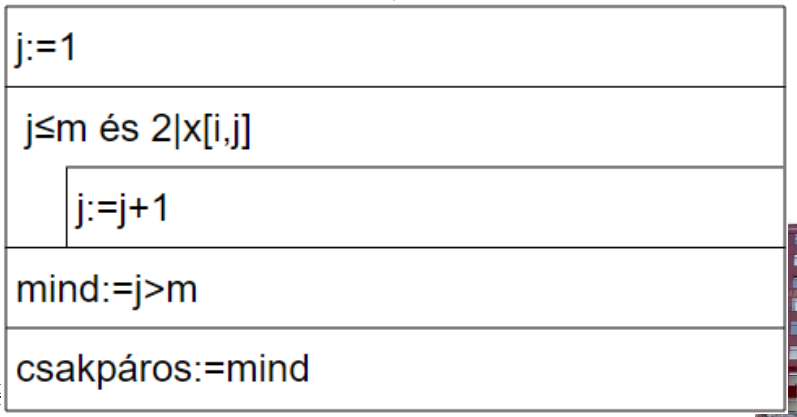
Megszámolás		
$e..u$	~	$1..n$
$T(i)$	~	$csakpáros(i)$

(Optimista) eldöntés		
i	~	j
$e..u$	~	$1..m$
$T(i)$	~	$2 \mid x[i,j]$



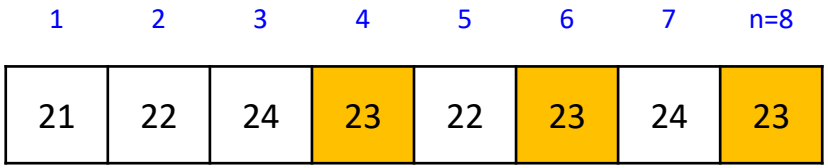
csakpáros(i:Egész): Logikai





Egymásba ágyazott tételek
Független intervallum

Egymást követő napokon délben megmértük a levegő hőmérsékletét. Állapítsuk meg, hogy melyik érték fordult elő leggyakrabban!



Maximumkiválasztásban megszámolás

Egymásba ágyazott tételek
Független intervallum

Egymást követő napokon délben megmértük a levegő hőmérsékletét. Állapítsuk meg, hogy melyik érték fordult elő leggyakrabban!

Maximumkiválasztásban megszámolás

Specifikáció:

- Bemenet: $n: Egész, x: Tömb(1..n: Egész)$
- Kimenet: $lgy: Egész$
- Előfeltétel: $n = n' \text{ és } x = x' \text{ és } n > 0$
- Utófeltétel: Ef és $(legtöbb, mxi) = Max_{i=1}^{n} hány(i)$ és $lgy = x[mxi]$
- Definíció: $hány: Egész \rightarrow Egész, hány(i) = \sum_{j=1}^{n} \begin{matrix} 1 \\ x[i]=x[j] \end{matrix}$

Egymásba ágyazott tételek

Független intervallum

Specifikáció:

$$(maxért, maxind) = Max_{i=e}^{u} f(i)$$

- Utófeltétel: Ef és $(legtöbb, mxi) = Max_{i=1}^{n} hány(i)$ és

  $lgy = x[mxi]$

- Definíció: $hány: Egész \rightarrow Egész$

  $hány(i) = \sum_{\substack{j=1 \\ x[i]=x[j]}}^{n} 1$

$$db = \sum_{\substack{i=e \\ T(i)}}^{u} 1$$

Maximumkiválasztás		
$maxért, maxind$	~	$legtöbb, mxi$
$e..u$	~	$1..n$
$f(i)$	~	$hány(i)$

Megszámolás		
i	~	j
$e..u$	~	$1..n$
$T(i)$	~	$x[i] = x[j]$

Egymásba ágyazott tételek

Független intervallum

Maximumkiválasztás		
$maxért, maxind$	~	$legtöbb, mxi$
$e..u$	~	$1..n$
$f(i)$	~	$hány(i)$

Megszámolás		
i	~	j
$e..u$	~	$1..n$
$T(i)$	~	$x[i] = x[j]$

```
maxért:=f(e);maxind:=e
i=e+1..u
    f(i)>maxért
    true                    false
    maxért:=f(i)            -
    maxind:=i
```

```
db:=0
i=e..u
    T(i)
    true                    false
    db:=db+1                -
```

hány(i:Egész): Logikai

```
legtöbb:=hány(1); mxi:=1
i=2..n
    hány(i)>legtöbb
    true                    false
    legtöbb:=hány(i); mxi:=i    -
lgy:=x[mxi]
```

```
db:=0
j=1..n
    x[i]=x[j]
    true                    false
    db:=db+1                -
hány:=db
```

Egymásba ágyazott tételek

Közös intervallum

Keressük meg a t négyzetes mátrixnak azt az oszlopát, amelyben a főátlóbeli és a feletti elemek összege a legnagyobb!

	1	2	3	4	m=5
1	1,1	2	4	3	1
2	2	6,2	4	2	1
3	5	3	2	1	0
4	2	8	2	8	2
n=5	5	3	2	3	0

Maximumkiválasztásban összegzés

Egymásba ágyazott tételek
Közös intervallum

Keressük meg a t négyzetes mátrixnak azt az oszlopát, amelyben a főátlóbeli és a feletti elemek összege a legnagyobb!

Maximumkiválasztásban összegzés

Specifikáció:

- Bemenet: $n: Egész, t: Tömb(1..n, 1..n: Valós)$
- Kimenet: $maxoszlop: Egész$
- Előfeltétel: $n = n'$ és $t = t'$ és $n > 0$
- Utófeltétel: Ef és $(maxért, maxoszlop) = Max_{i=1}^{n} összeg(i)$
- Definíció: $összeg: Egész \rightarrow Valós$

  $összeg(i) = \sum_{j=1}^{i} t[j, i]$

Közös intervallum

Egymásba ágyazott tételek

Közös intervallum

Specifikáció:

$$(maxért, maxind) = Max_{i=e}^{u} f(i)$$

- Utófeltétel: Ef és $(maxért, maxoszlop) = Max_{i=1}^{n} összeg(i)$
- Definíció: $összeg: Egész \rightarrow Valós$

  $összeg(i) = \sum_{j=1}^{i} t[j,i]$

$$s = \sum_{i=e}^{u} f(i)$$

Maximumkiválasztás		
$maxind$	~	$maxoszlop$
$e..u$	~	$1..n$
$f(i)$	~	$összeg\ (i)$

Összegzés		
i	~	j
$e..u$	~	$1..i$
$f(i)$	~	$t[j,i]$

Egymásba ágyazott tételek

Közös intervallum

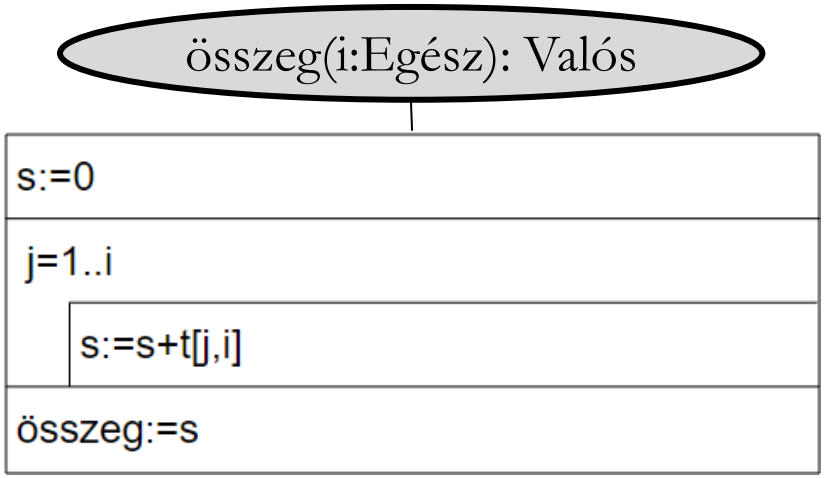
Maximumkiválasztás		
$maxind$	~	$maxoszlop$
$e..u$	~	$1..n$
$f(i)$	~	$összeg\ (i)$

Összegzés		
i	~	j
$e..u$	~	$1..i$
$f(i)$	~	$t[j,i]$

```
maxért:=f(e);maxind:=e
i=e+1..u
    f(i)>maxért
    true                    false
    maxért:=f(i)            -
    maxind:=i
```

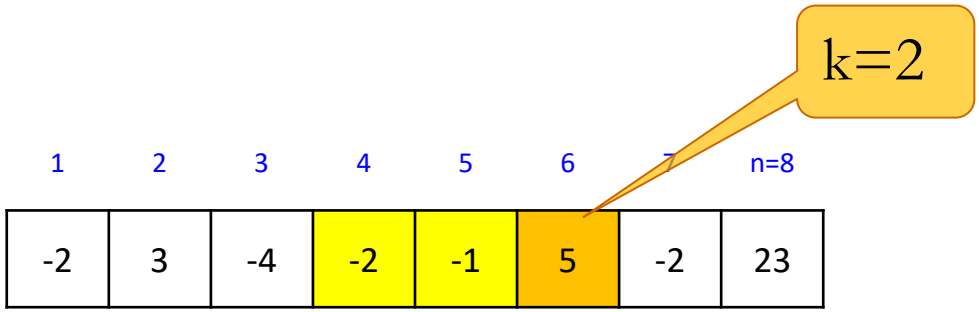
```
s:=0
i=e..u
    s:=s+f(i)
```

```
maxért:=összeg(1);maxoszlop:=1
i=2..n
    összeg(i)>maxért
    true                    false
    maxért:=összeg(i)       -
    maxoszlop:=i
```



Egymásba ágyazott tételek
Közös intervallum

Volt-e a lóversenyen olyan napunk, amikor úgy nyertünk, hogy a megelőző k napon mindig veszítettünk?



Eldöntésben (optimista) eldöntés

Egymásba ágyazott tételek
Közös intervallum

Volt-e a lóversenyen olyan napunk, amikor úgy nyertünk, hogy a megelőző k napon mindig veszítettünk?

Eldöntésben (optimista) eldöntés

Specifikáció:

- Bemenet: $n: Egész, x: Tömb(1..n: Valós), k: Egész$
- Kimenet: $van: Logikai$
- Előfeltétel: $n = n'$ és $x = x'$ és $n \geq 0$ és $k \geq 0$
- Utófeltétel: Ef és $van = \exists_{i=k+1}^{n} csupavesztés(i)$
- Definíció: $csupavesztés: Egész \rightarrow Logikai$

  $csupavesztés(i) = \forall_{j=i-k}^{i-1} x[i] > 0$ és $x[j] < 0$

Közös intervallum

Egymásba ágyazott tételek

Közös intervallum

$$van = \exists_{i=e}^{u} T(i)$$

Specifikáció:

- Utófeltétel: Ef és $van = \exists_{i=k+1}^{n} csupavesztés(i)$
- Definíció: $csupavesztés: Egész \rightarrow Logikai$

$$csupavesztés(i) = \forall_{j=i-k}^{i-1} x[i] > 0 \text{ és } x[j] < 0$$

$$mind = \forall_{i=e}^{u} T(i)$$

Eldöntés		
$e..u$	$\sim$	$k+1..n$
$T(i)$	$\sim$	$csupavesztés\ (i)$

(Optimista) eldöntés		
i	$\sim$	j
$e..u$	$\sim$	$i-k..i-1$
$T(i)$	$\sim$	$x[i] > 0$ és $x[j] < 0$

Egymásba ágyazott tételek

Közös intervallum

Eldöntés		
$e..u$	$\sim$	$k+1..n$
$T(i)$	$\sim$	$csupavesztés\ (i)$

```
i:=e
i≤u és nem T(i)
    i:=i+1
van:=i≤u
```

(Optimista) eldöntés		
i	$\sim$	j
$e..u$	$\sim$	$i-k..i-1$
$T(i)$	$\sim$	$x[i]>0$ és $x[j]<0$

```
i:=e
i≤u és T(i)
    i:=i+1
mind:=i>u
```

csupavesztés(i:Egész): Logikai

```
i:=k+1
i≤n és nem csupavesztés(i)
    i:=i+1
van:=i≤n
```

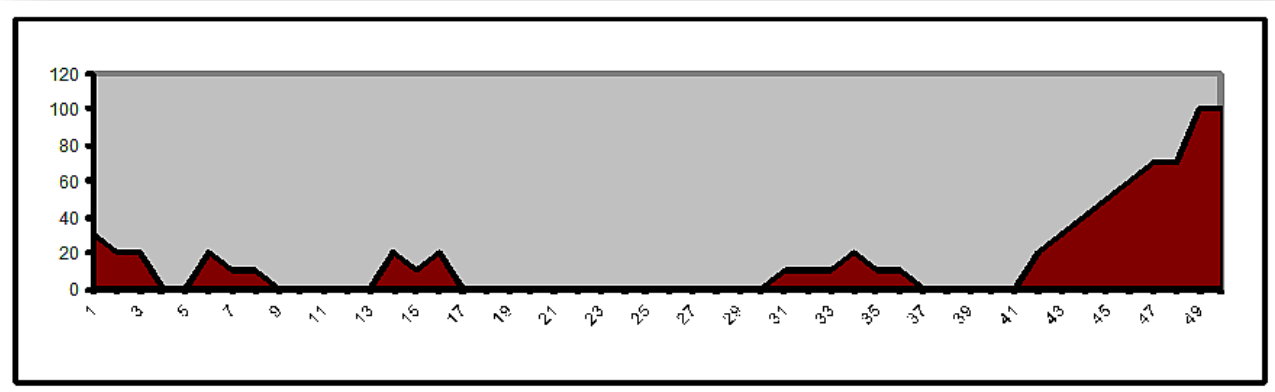
```
j:=i-k
j≤i-1 és (x[i]>0 és x[j]<0)
    j:=j+1
mind:=j>i-1
csupavesztés:=mind
```

Tesztelés
(folytatás)

Tesztek előállítása

Feladat (teszteléshez)**:**

Egy repülőgéppel Európából Amerikába repültünk. Az út során X kilométerenként mértük a felszín tengerszint feletti magasságát (≥0). 0 magasságot ott mértünk, ahol tenger van, >0-t pedig ott, ahol szárazföld. Adjuk meg a szigeteket!



Tesztek előállítása

Specifikáció:

- Bemenet: $N \in \mathbb{N}$, $Mag_{1..N} \in \mathbb{Z}^N$
- Kimenet: $Db \in \mathbb{N}$, $K_{1..N}, V_{1..N} \in \mathbb{N}^N$
- …

Tesztelés:

- **Kis** tesztek a tesztelési elveknek megfelelően, például:

  N=3, Mag=(1,0,1) → nincs sziget
  N=5, Mag=(1,0,1,0,1) → egy „rövid” sziget
  N=7, Mag=(1,0,1,0,1,0,1) → több „rövid” sziget
  N=7, Mag=(1,0,1,1,1,0,1) → hosszabb sziget

- Hogyan készítünk **nagy** (hatékonysági) teszteket?

Szabályos tesztek

Generálhatunk „**szabályos**” teszteket (egyszerű ciklusokkal).

Például így:

Változó
i:**Egész**

```
N:=1000
i=1..10
    Mag[i]:=11–i
i=11..900
    Mag[i]:=0
i=901..N
    Mag[i]:=i–900
```

⇐ Európa

⇐ tenger

⇐ Amerika

Kérdés:

hogyan illesszünk bele szigeteket sokféleképpen egyetlen programmal?

Válasz: véletlenszámokkal.

Véletlen tesztek
(alapok – véletlenszámok)

A **véletlenszám**okat a számítógép egy algoritmussal állítja elő egy kezdőszámból kiindulva.

$x_0 \rightarrow f(x_0)=x_1 \rightarrow f(x_1)=x_2 \rightarrow \ldots$

A „véletlenszerűséghez” megfelelő függvény és jó kezdőszám szükséges.

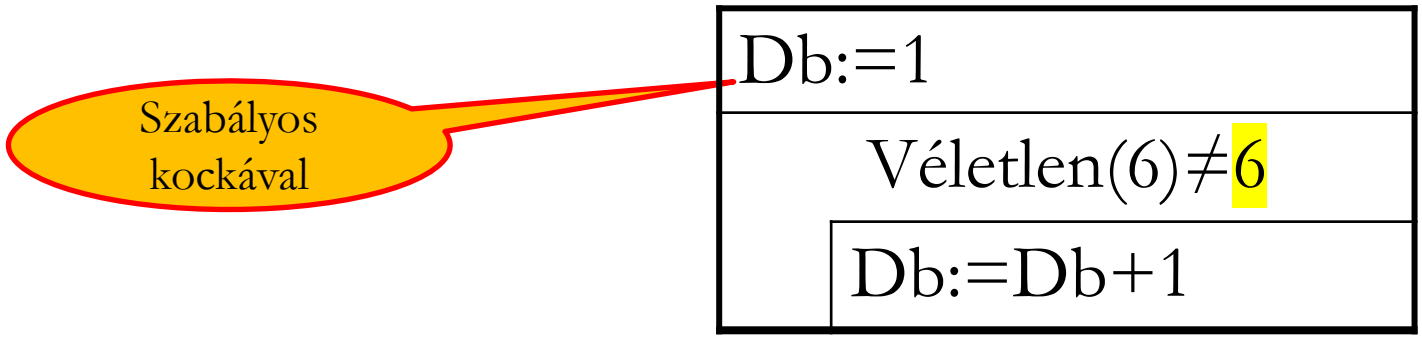
- **Kezdőszám**: (pl.) a belső órából vett érték.
- **Függvény** (az ún. lineáris kongruencia módszernél):
  $f(x) = (A*x+B) \text{ Mod } M,$
  ahol A, B és M a függvény belső konstansai.

Véletlen tesztek
(alapok – algoritmikus nyelv)

- Véletlenszám függvények:
  - `Véletlen(a..b)`$\in\{a,\ldots,b\}$, $a\leq b\in\mathbb{Z}$
  - `Véletlen(N)`$\in\{1,\ldots,N\}$, $N\in\mathbb{N}$
  - `Véletlenszám`$\in[0,1)\subset\mathbb{R}$

- Példa-feladat:

  Hányszor (Db) kell dobni kockát az első 6-osig?



Véletlen tesztek
(alapok – algoritmikus nyelv)

- Véletlenszám függvények:
  - `Véletlen(a..b)`$\in\{a,\ldots,b\}$, $a\leq b\in\mathbb{Z}$
  - `Véletlen(N)`$\in\{1,\ldots,N\}$, $N\in\mathbb{N}$
  - `Véletlenszám`$\in[0,1)\subset R$

Az i. intervallum:
az i bekövetkezése.
Hossza: valószínűsége.

- Példa-feladat:

  Hányszor (Db) kell dobni kockát az első 6-osig?

„Cinkelt” kockával

Val[i]: 1..i kijövetelének valószínűsége

```
Val[1..6]:=(0.1,0.3,0.5,0.7,0.9,1.0)
Db:=1
Véletlenszám≥Val[6-1]
    Db:=Db+1
```

Véletlen tesztek
(alapok – C#)

rnd egy Random osztálybeli objektum

`Random` egy véletlen értékeket szolgáltató osztály, amely `Next` függvényével lehet a következő véletlen értéket képezni.

```
Random rnd=new Random();
```

- `Véletlen(a..b)` $\in \{a,\ldots,b\}$, $a \leq b \in \mathbb{Z}$

```
v=rnd.Next(a,b);
```

- `Véletlen(N)` $\in \{1,\ldots,N\}$, $N \in \mathbb{N}$

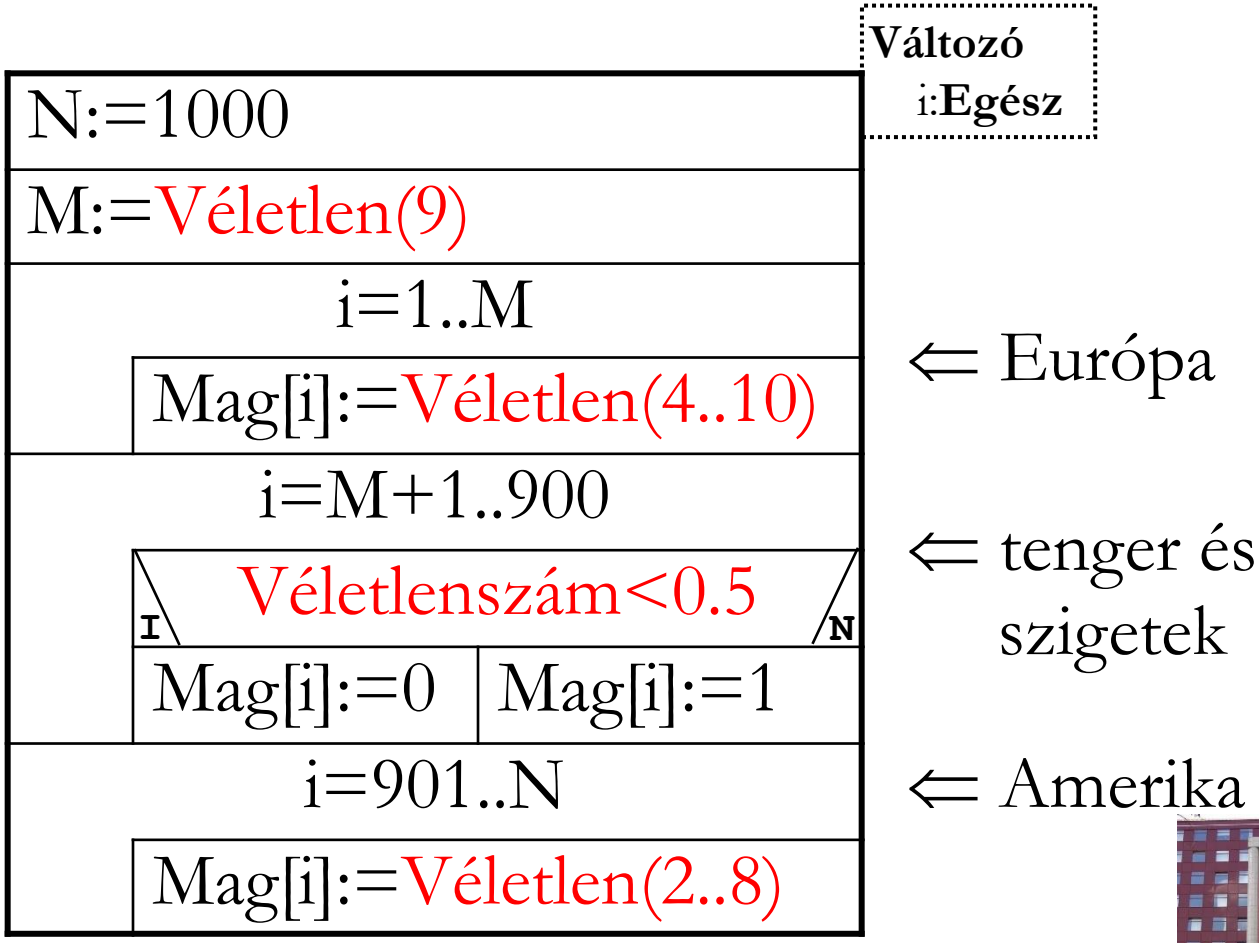
```
v=rnd.Next(N)+1;
```

- `Véletlenszám` $\in [0,1) \subset \mathbb{R}$

```
v=rnd.NextDouble();
```

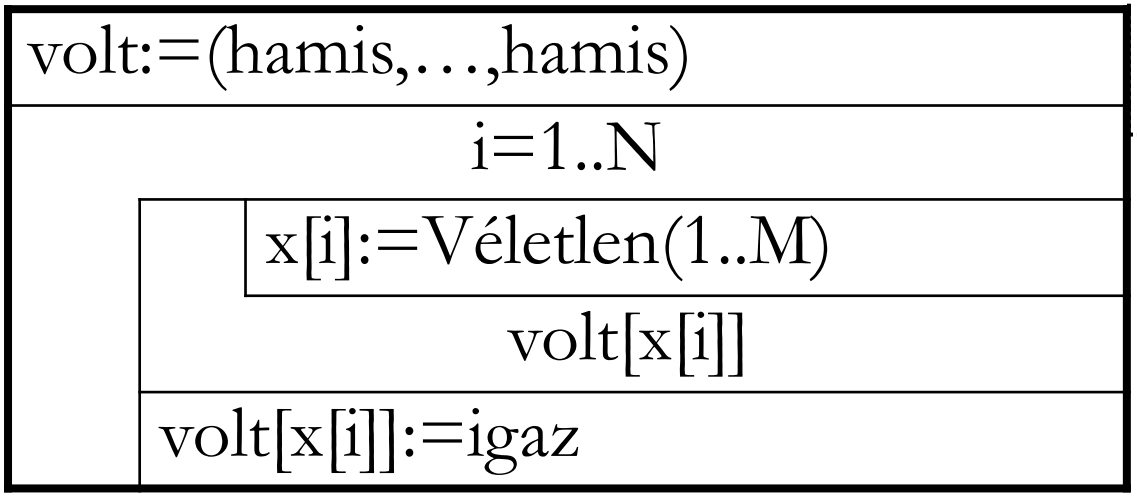
Véletlen tesztek

Véletlen tesztekhez használjunk véletlenszámokat! Például így:



Véletlen tesztek

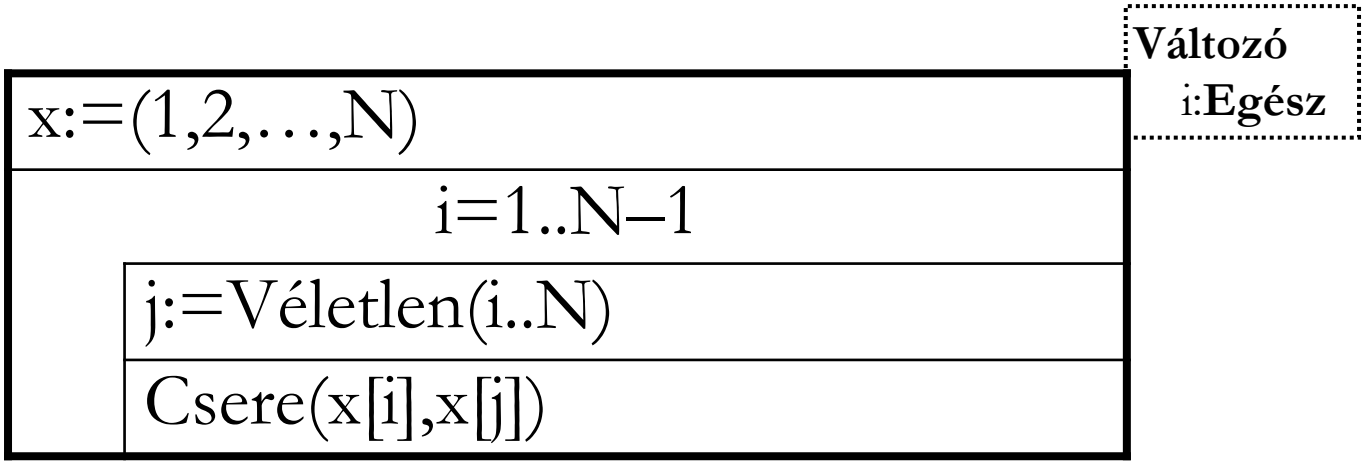
Rendezést is tartalmazó feladatoknál gyakori egy **rendezetlen bemenet előállítása** (1 és M közötti különböző N darab érték). Ha M „elég nagy” N-hez (M»N):



Változó
i:**Egész**
volt:**Tömb[**1..M**:**
Logikai]

Véletlen tesztek

Rendezést is tartalmazó feladatoknál gyakori egy **rendezetlen bemenet előállítása** (1 és M közötti különböző N darab érték). Az M=N eset (keverés):



Véletlen tesztek

Rendezést is tartalmazó feladatoknál gyakori egy **rendezetlen** bemenet előállítása (1 és M közötti különböző N darab érték).

Az M>N eset:

Változó
i,db,
kell,
van:**Egész**

```
db:=0; kell:=N
van:=M
i=1..M
    Véletlenszám<kell/van
    I                    N
    db:=db+1; x[db]:=i; kell:=kell-1 | —
    van:=van-1
```

Hibakeresés

Hibakeresés

Hibajelenségek a tesztelés során…

- hibás az eredmény,
- futási hiba keletkezett,
- nincs eredmény,
- részleges eredményt kaptunk,
- olyat is kiír, amit nem vártunk,
- túl sokat (sokszor) ír,
- nem áll le a program,
- …

Hibakeresés

Célja:

a felfedett hibajelenség okának, helyének megtalálása.

Elvek:

- Eszközök használata előtt alapos végiggondolás.
- Egy megtalált hiba a program más részeiben is okozhat hibát.
- A hibák száma, súlyossága a program méretével nemlineárisan (annál gyorsabban!) nő.
- Egyformán fontos, hogy *miért* nem *csinálja* a program, amit *várunk*, illetve, hogy *miért csinál* olyat, amit nem *várunk*.
- Csak akkor javítani, ha megtaláltuk a hibát!

Hibakeresés

Hibakeresési eszközök (folytatás):

- Változó-, memória-kiírás (feltételes fordítás)
- Töréspont elhelyezése
- Lépésenkénti végrehajtás
- Adat-nyomkövetés
- Állapot-nyomkövetés (pl. paraméterekre vonatkozó előfeltételek, ciklus-invariánsok)
- Postmortem nyomkövetés: hibától visszafelé

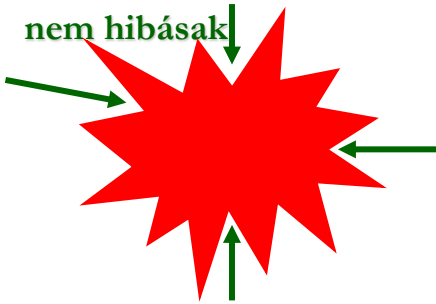
Hibakeresési módszerek

Célja:

- A **bemenetnek mely** része, amelyre hibásan működik a program?
- **Hol** található a **programban** a hibát okozó utasítás?

Módszerfajták:

1. Indukciós módszer (hibásak körének bővítése)
2. Dedukciós módszer (hibásak körének szűkítése)
3. Hibakeresés hibától visszafelé
4. Teszteléssel segített hibakeresés (olyan teszteset kell, amely az ismert hiba helyét fedi fel)



Hibakeresési módszerek

Példa az indukciós módszerre:

Feladat: *1 és 99 közötti N szám kiírása betűkkel.*

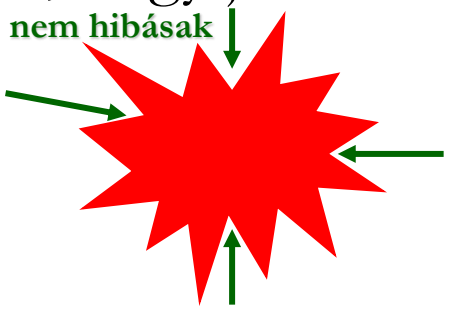
- Tesztesetek: N=8 ⇒ jó, N=17 ⇒ jó, N=30 ⇒ hibás.
- Próbáljunk a hibásakból általánosítani: tegyük fel, hogy minden 30-cal kezdődőre rossz!
- Ha beláttuk (teszteléssel), akkor próbáljuk tovább általánosítani, pl. tegyük fel, hogy minden 30 felettire rossz!
- Ha nem lehet tovább általánosítani, akkor tudjuk mit kell keresni a hibás programban.
- Ha nem ment az általánosítás, próbáljuk másképp: hibás-e minden 0-ra végződő számra!
- …

Hibakeresési módszerek

Példa a dedukciós módszerre:

Feladat: *1 és 99 közötti N szám kiírása betűkkel.*

- Tesztesetek: N=8 ⇒ jó, N=17 ⇒ jó, N=30 ⇒ hibás.
- Tegyük fel, hogy minden nem jóra hibás!
- Próbáljunk a hibás esetek alapján szűkíteni:
  tegyük fel, hogy a 20-nál kisebbekre jó!
- Ha beláttuk (teszteléssel), akkor szűkítsünk tovább, jó-e minden 39-nél nagyobbra?
- Ha nem szűkíthető tovább, akkor megtaláltuk, mit kell keresni a hibás programunkban.
- Ha nem, szűkítsünk másképp: tegyük fel, hogy jó minden nem 0-ra végződő számra!
- …



Hibajavítás

Hibajavítás

Célja:

a megtalált hiba kijavítása.

Elvek:

- A hibát kell javítani és **nem a tüneteit**.
- A hiba kijavítása a program **más** részében hibát okozhat (rosszul javítunk, illetve korábban elfedett más hibát).
- Javítás után a **tesztelés megismételendő**!
- A jó javítás valószínűsége a program méretével **fordítva arányos**.
- A hibajavítás a **tervezési** fázisba is visszanyúlhat (a módszertan célja: lehetőleg ne nyúljon vissza).

Dokumentálás

Dokumentációk

Fajtái:

- *Programismertető*
- Felhasználói dokumentáció
- Fejlesztői dokumentáció
- …

Dőlten szedve, ami a beadandó komplex program estén a dokumentációból elhagyható.

Felhasználói dokumentáció

Tartalma:

E nélkül be sem adható!

- feladatszöveg (rövidített és *részletes* is)
- futási környezet (szg.+or.+*hw/sw-elvárások*)
- használat leírása (*telepítés,* kérdések + lehetséges válaszok,...)
- bemenő adatok, eredmények, *szolgáltatások*
- mintaalkalmazások – példafutások
- hibaüzenetek és a hibák lehetséges okai

Dőlten szedve, ami a beadandó komplex program estén a dokumentációból elhagyható.

Fejlesztői dokumentáció

E nélkül be sem adható!

Tartalma:

- feladatszöveg, specifikáció
- fejlesztői környezet (or.+fordító program, …)
- adatleírás (feladatparaméterek reprezentálása)
- algoritmusok leírása, döntések (pl. tételekre utalás), *más alternatívák*, *érvek*, magyarázatok
- kód, *implementációs szabványok*, ~ *döntések*
- tesztesetek
- *hatékonysági mérések*
- fejlesztési lehetőségek
- szerző(k)

Dőlten szedve, ami a beadandó komplex program estén a dokumentációból elhagyható.

Szerző

Név: **Szabó Emerencia**
ETR-azonosító: **SZEKAAT.ELTE**
Neptun-azonosító: **ESZ98A**
Drótposta-cím: **sze@elte.hu**
Kurzuskód: **IP-08PAEG/77**
Gyakorlatvezető neve: **Kiss-József Alfréd**
Feladatsorszám: **18**

E nélkül be sem adható!

Értelmesen strukturálva.

Programkészítési elvek

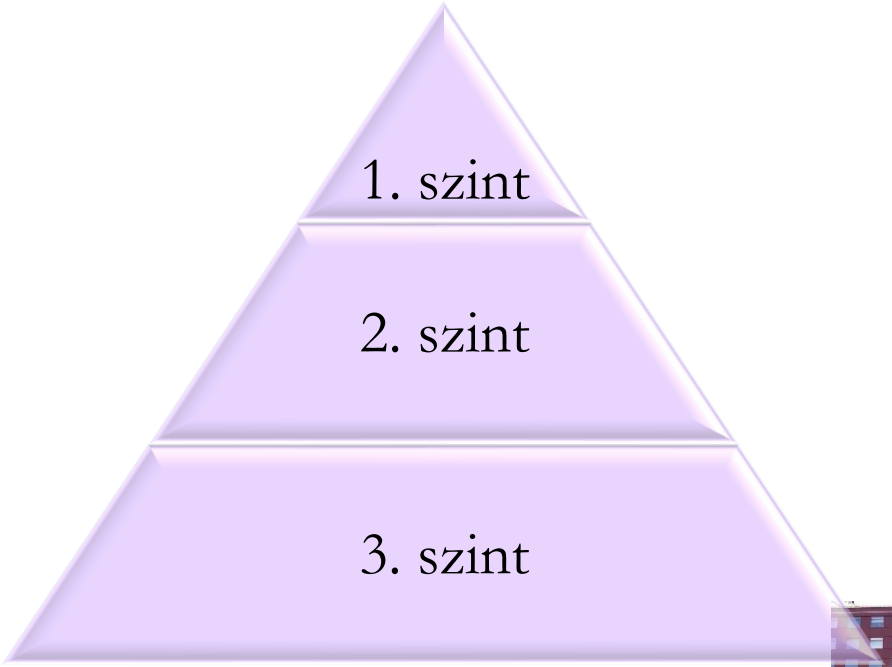
Programkészítési elvek

- **Stratégiai elv**: a problémamegoldás logikája – a lépésenkénti finomítás.
- **Taktikai elvek**: az algoritmuskészítés gondolati elvei a felülről lefelé kifejtéshez.
- **Technológiai elvek**: algoritmus és kód módszertani kívánalmai.
- **Technikai elvek**: kódolási technika.
- **Esztétikai, ergonómiai elvek**: emberközelség.

Stratégiai elv:
lépésenkénti finomítás

- Felülről–lefelé (top–down) = probléma–dekomponálás, –analizálás.
- Alulról–felfelé (bottom–up) = probléma–szintézis.

Nem alternatívák!



Taktikai elvek

- Párhuzamos finomítás
- Döntések elhalasztása
- Döntések nyilvántartása
- Vissza az ősökhöz
- Nyílt rendszer felépítés (általánosítás)
- Adatok elszigetelése (pl. alprogramokba helyezéssel + paraméterezéssel + lokális adatok deklarálásával)
- Párhuzamos ágak függetlensége
- Szintenkénti teljes kifejtés

Technológiai elvek
az algoritmus készítéshez

- Struktúrák zárójelezése
- Bekezdéses struktúrák

Struktogram esetén ezek nyilvánvalóan teljesülnek

- Értelmes utasítás-csoportosítás
- Kevés algoritmusleíró szabály definiálása, de azok szigorú betartása (pl. tétel → algoritmus)
- Beszédes azonosítók, kifejező névkonvenciók (pl. Hungarian Notation)

Technikai elvek
a kódoláshoz

- Barátságosság (pl. kérdések, címek)
- Biztonságosság (pl. I/O-ellenőrzések)
- Kevés kódolási szabály definiálása, de azok következetes betartása (algoritmus és kód koherenciája; továbbá pl. amígos ciklusokhoz, I/O-hoz)
- Jól olvashatóság

Esztétikai/ergonómiai elvek

- Lapokra tagolás, kiemelés, elkülönítés
- Menütechnika
- Ikontechnika, választás egérrel
- Következetesség (beolvasás, kiírás, ...)
- Hibafigyelés, hibajelzés, javíthatóság
- Súgó, tájékoztató
- Ablakkezelés
- Értelmezési tartomány kijelzése
- Naplózás

Visszetekintés

- Egymásba ágyazott programozási tételek
  (a 7. előadás folytatása)

- Tesztek előállítása
- Hibakeresés
- Hibajavítás
- Dokumentálás

- Programkészítési elvek