```
eljárás kiválogat_2 (n, k: egész);
    i := 1;
    ciklus míg $A[i].s \neq k$
        i := i+1;
    c vége;
    ciklus míg $(i \leq n) \wedge (A[i].s = k)$;
        ki : A[i].mérés
        i := i+1;
    c vége;
e vége.
```

III.3.

4. előadás

Végrehajtási idő csökkentése - ciklusmag lefutásának száma

Ciklustranszformálás:

Keresés helyett egyetlen képlettel meghatározható:

- keresett elem pontos helye
- keresett elemhez közeli elem

F.: Adjuk meg n-ig az ikerprímeket! → (2 a távolságuk és prímek)

Vizsgáljuk meg a számokat 2-től, h. i és i+2 prímek-e?

```
Eljárás ikerprímek_1 (n: egész);
    ciklus i := 2 .. n
        ha prím(i) $\wedge$ prím(i+2) akkor ki: i, i+2;
    c vége;
e vége.
```

- látszik, hogy felesleges, mert az első és onnan minden második páros, és azok nem jók. A 2 sem ikerprím, mert a 4 megint nem felel meg
- ha nem lennének ikerprímek, felesleges lenne akkor is minden második

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21

páros számok

szóba kerülhető prímek

⇓

az i kezdőértéke lehet 3, mert minden második nem jöhet szóba.

```
Eljárás ikerprímek_2 (n: egész);
  i:= 3;
  ciklus míg i≤n
    ha prím (i) akkor
      Ha prím(i+2) akkor ki i, i+2;  i:= i+2;
      különben i:= i+4;
      h vége
    különben   i:= i+2;
    h vége
  c vége;
e vége;
```

- Ha az i prím, megnézzük, hogy az i+2 prím-e, és a következő i értéke legyen i+2, mert még az is lehet jó megoldás.
- a 7 esetében a következő i értékét néggyel nagyobbnak választjuk.
- ha prím: a rákövetkező (2-vel nagyobb) prím ⇒ 2-vel növeljük a ráköv. nem prím ⇒ 4-gyel növeljük

– ha i értéke 23, a rákövetkező nem prím, 27 sem prím, vegyük a rákövetkezőt.

7.: A megfigyelést végezve feljegyeztük, a különböző sorszámú buszokon hányan utaznak. Azonos számú buszokról több feljegyzés is készült. Határozzuk meg az egyes buszok átlagos utasszámát!

Arra kíváncsiak, hogy egy-egy buszon átlagosan mennyien utaznak.

A buszok sorszáma legyen 1-től s-ig folyamatos.

- Adat [1.. n] : megfigyelések
- Adat [i]. b : az i. megfigyelésben a busz sorszáma
- Adat [i]. u : –"– a buszon utazók száma
- m_db [1.. s] : az egyes buszokról hány feljegyzés készült
- u_db [1.. s] : az egyes buszokon utazók összesen (i buszon)

(kezdetben m_db és u_db minden eleme 0)

```
Eljárás utasszám_1_2 (n, s : egész)
    ciklus i := 1..s          → ki kell venni az adatokat a buszhoz kapcsolódóan
        ciklus j := 1.. n     → sorra kell venni minden megfigyelést
            ha adat [j]. b = i akkor
                m_db [i] := m_db[i]+1;          → összesen az i buszon hányan utaztak
                u_db [i] := u_db [i] + adat [j]. u   → mivel az u mezőben tárolják
            h vége;
        c vége;
    c vége;
```

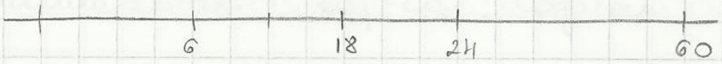
- Cseréljük meg a két ciklust! Ne a buszsorszámok szerint haladjunk, vegyük figyelembe a megfigyeléseket.

```
Eljárás utasszám_3(n,s : egész);
  ciklus j:=1..n
    i:=adat[j].b;        -> adatkérés szempontjából indokolt
    m_db[i]:=m_db[i]+1;
    u-db[i]:= u-db[i]+adat[j].u
  c vége;
e vége;
```

- 2 vezérlési szerkezetet (ciklus + feltételvizsgálat) kihagytuk.

* * *

korcsoportok:



- Azt akarjuk megszámolni, hogy az egyes korcsoportokban hányan vannak.

- Kijelölhető vmi lnko -> azzal kell felosztani osztópontokra, amik a korhatárokra is esnek

- bevezetünk egy számlálótömböt.

s[0] -> 6 éves kor alattiak létszáma

s[1] -> 6 - 12 éves kor közöttiek létszáma

s[2] -> 12 - 18 —"—

⋮

stb.

Az i-dik életkor := e.

s[e div 6]   0 adódik, ha 6-nál kisebb

↳ megindexelhetjük a vektort, és utána megnöveljük

INC (s[e div 6])

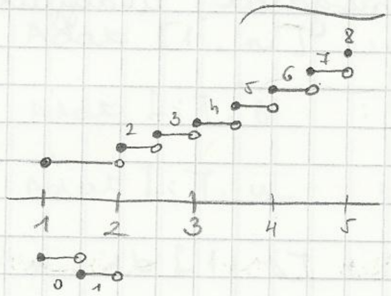
0 - 6 : s[0]

6 - 18 : s[1]+s[2]

18 - 24 : s[3]

24 - 60 : s[4] + ...

Indexé konvertáljuk az adott értéket



Jegyek!

- 9 tartományba esnek az átlagos értékei
- Szorozzuk meg az átlagot 2-vel. 2·a
- az egyenes ki fog tolódni 10-ig. (a szorzás miatt)
- Vegyük a 2a-nak az egészrészét [2a]
- Vonjuk ki 2-t. [2a] - 2

$\text{INC}\,(\underset{\downarrow}{\text{db}}\,[\overset{\text{trunc}}{\text{round}}\,(2a)-2])$

Végrehajtási idő csökkentése – ciklusmag lefutásának száma

Iterált típus megfelelő finomítása

Ha a feladat során alkalmazott adatstruktúra egy sokaság (halmaz, sorozat, hierarchikus szerkezet, hálós szerkezet), a hatékonyságot annak finomítása erősen befolyásolja.
A finomítás célszerű módja a feladattól függ.

A halmaz különböző megvalósításai

- minden halmaz elemét tároljuk egy sorozatban
  - rendezetten
  - rendezetlenül (nem rendezetten)
- csak az alaphalmaz elemeit tároljuk egy sorozatban, az egyes halmazokat egy-egy logikai sorozat reprezentálja.

Feladat: Adott egy vasútvonal esetében az egymást követő vasútállomások távolságának sorozata. Adjuk meg az i. és a j. állomás távolságát!

n vasútállomás esetén a távolságok $\{a_{n-1}\}$ sorozatát A[1..n-1] vektor tárolja, ahol az i. és az i+1. állomás távolságát A[i] tárolja.

```
Eljárás távolság_1(i,j,t: egész);
  t = 0;
  ciklus k := i .. j-1
      t := t + A[k];
  c vége;
e vége;
```

Ha előfeldolgozással előállítjuk a $\{d_{n-1}\}$ sorozatot, aminek i. eleme az első és az i. állomás távolságát tartalmazza.

```
Eljárás távolság_2 (i, j; t: egész);
    t := A[j] – A[i];
e vége.
```

További példák:

- hierarchikus adatszerkezet alkalmazása (keresőfa)
  - szekvenciális v.
  - láncolt ábrázolás

Végrehajtási idő csökkentése – ciklusmag lefutásának száma.

Sorozat elemeinek rekurzív előállítása:

pl.: $$\frac{1}{e} = \sum_{i=0}^{\infty} (-1)^i \frac{1}{i!} \frac{1}{(k-1)(k-1)!} = \sum_{i=1}^{\infty} \frac{1}{\prod_{k=i}^{i+k-1} k}$$

$$(-1)^0 \frac{1}{0!} = \textcircled{1}$$

$$-1 \frac{1}{1} = \textcircled{-1}$$

$$1 \cdot \frac{1}{2} = \textcircled{\tfrac{1}{2}}$$

elemek

⋮

stb.

Gyakori feladat egy sorozat elemeinek lépésenként történő előállítása (Σ, Π) (A algoritmus szerkezete ekkor általában két egymásba ágyazott ciklus.)

Feladat: Adott $\{a_n\}$ számsorozat esetén számítsuk ki a k hosszúságú részsorozatainak átlagát (mozgó átlag) $k \leq n$

$$b_i = \sum_{j=0}^{k-1} a_{i+j}$$

$$a_1 \;\; \underbrace{a_2 \;\; \rlap{$\overbrace{}^{s_3}$}a_3 \;\; a_4}_{s_2} \;\; a_5 \;\; a_6 \;\; a_7 \;\; a_8$$

```
Eljárás mozgóátlag_1 (n, k : egész, A, B : sorozat);
ciklus i := 1 .. n-k+1;
        ↳ hány ilyen részsorozat van
  s := 0;
    ciklus j := 0 .. k-1;   → összegzi az i. sorozatban a részsorozatokat
      s := s + A[i+j];  → összegzés
    c vége;
  B[i] := s/k;
c vége;
e vége;
```

$s_3 = s_2 - a_2 + a_5$

elegendő az első részsorozat összegét kiszámolni, csak kivonni ami lemarad, és hozzáadni az új elemet

$$b_{i+1} = b_i - \frac{a_i}{k} + \frac{a_{i+k}}{k}$$

```
Eljárás mozgóátlag_2 (n, k: egész, A, B: sorozat);
  s:=0;
  ciklus i:=1..k
    s:=s+A[i];
  c vége;
  B[1]:=s/k;
  ciklus j:=1..n-k
      B[j+1]:=B[j]+(A[j+k]-A[j])/k;
  c vége;
evége;
```

III. 10.

5. előadás

Eratoszthenészi szita: olyan logikai sorozatot ad, ahol a prímedik
helyen 1 érték van, a többin H.

$\{a_n\}$ → prímeknél 1 érték van, nem prímnél H érték van.

1 logikai elemek indexeit kell összehasonlítani.

(→ persze 1 értékűeket)

Ha növekvő sorrendbe gyűjtöttük ki ⇒ csak a szomszédos elemeket kell megnézni, hogy a különbségük 2-e.

$|_3 \quad |_5 \quad |_7 \quad |_{11} \quad |_{13} \quad \ldots$

$2 \quad 2 \quad 4 \quad 2$

Végrehajtási idő csökkentése - ciklusmag végrehajtási idejének csökkentése

Elágazások transzformálása

Sokszor a ciklusmag végrehajtásának száma nem csökkenthető, de az egyszeri végrehajtási idő igen.

$T = n(t + \Delta t)$ $T' = nt$

„megtakarítás": $n \cdot \Delta t$ $(T > T')$

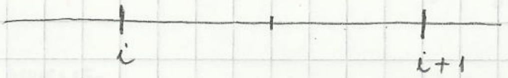
T' pontosan $n \cdot \Delta t$-vel kevesebb a T-nél.

F: Végezzük el az első n db szám kerekítését!

(i) Ha $(A - [A]) < 0,5$ akkor $B := [A]$,

A ↓ szám, $[A]$ ↓ szám egészrésze

különben $B := [A] + 1$



kerekítés szabályai szerint: $a < i + \frac{1}{2} \Rightarrow [a]$ kapjuk

$(i+1) > a \geq i + \frac{1}{2} \Rightarrow [a]+1$ kapjuk

$a < i + \frac{1}{2}$ $/+\frac{1}{2}$	$a \geq i + \frac{1}{2}$ $/+\frac{1}{2}$
$a + \frac{1}{2} < i+1$	$a + \frac{1}{2} \geq i+1$
i-t kapunk	i+1-et kapunk

(ii) Mondhatjuk, hogy:

$B := [A + 0,5]$;