Problémaosztályok algoritmusok

Ciklusok

Ciklus

- Valamely tevékenység sorozat ismételt végrehajtását jelenti.

- Az ismétlés lehet:
  - feltételhez kötött
  - előírt lépésszám szerint

- Három fajtája van:
  - Elöl tesztelő
  - Hátul tesztelő
  - Léptetéses (növekményes)

Elől tesztelő ciklus

Ciklus amíg Lefutási feltétel

Cikusmag utasításai

Ciklus vége

Működése

- A ciklusmag utasításait mindaddig kell végrehajtani. amíg a *Lefutási feltétel* igaz.
- Ha a lefutási feltétel hamissá válik, a végrehajtás a ciklus vége után folytatódik.
- Ha a lefutási feltétel már a ciklusba való belépéskor hamis, a ciklusmag utasításai egyszer sem hajtódnak végre.

Elöltesztelő ciklus: Példa

Ciklus amíg van nálam szórólap
elmegyek a köv. lakásig
bedobom a szórólapot a postaládába
Ciklus vége

Elől tesztelő ciklus, az előreolvasás fogalma

- Az elöl tesztelő ciklus megvalósítása mindig előreolvasással történik:
  - a feltételben vizsgált adatot előbb meg kell határozni
  - azután megvizsgáljuk, hogy a megadott érték megfelel-e a lefutási feltételnek
  - a ciklusmagban pedig biztosítanunk kell a feltétel hamissá válását, különben végtelen ciklust kapunk!

Elöltesztelő ciklus: C nyelv

```
while (kifejezés)     //lefutási feltétel
{
        ciklusmag utasításai
}
```

- A **while** ciklus mindaddig ismétli a hozzá tartozó utasításokat (a ciklusmag utasításait, más szóval a ciklus törzsét), amíg a vizsgált kifejezés (lefutási feltétel) értéke nem nulla (igaz).
- A vizsgálat mindig megelőzi a ciklusmag utasításainak végrehajtását, ezért van olyan eset, amikor a ciklusmag utasításai egyszer sem kerülnek végrehajtásra.

Elől tesztelő ciklus, példa

- **Feladat**: Vásároljunk könyveket a zsebpénzünkbőlkaptam egy összeget, amit könyvekre költhetek.
- **Tevékenység-sorozat**, amit ismétlünk:

  kiválasztok egy könyvet, kifizetem
- **Feltétel**:

  van még elég pénzem

- Pszeudokód-részlet:

```
Be: penzosszeg     //amit kaptam könyvekre …
Be: konyv_ara      //az első kiválasztott könyv ára          //előreolvasás
Ciklus amíg  konyv_ara < penzosszeg
        penzosszeg = penzosszeg – konyv_ara                  //feldolgozás
        Be: konyv_ara     // következő kiválasztott könyv ára
Ciklus vége
```

- **Kérdés: lehet, hogy egyetlen könyvet sem veszek meg ?**

Hátul tesztelő ciklus

Ciklus

Ciklusmag utasításai

mígnem *Kilépési feltétel*

Ciklus vége

Működése:

- A ciklusmag utasításait mindaddig kell végrehajtani, amíg a *Kilépési feltétel igaz.*
- Ha a *Kilépési feltétel* hamissá válik, a végrehajtás a ciklus vége után folytatódik.
- A ciklusmag utasításait **egyszer mindenképpen végrehajtja** a rendszer, mivel a Kilépési feltétel kiértékelése a ciklusmag utasításainak végrehajtása után történik.

Hátul tesztelő ciklus: példa

Felmászok a fára egy kosárral

Ismételd

    Leszedek egy almát a fáról

    Berakom a kosárba

Amíg van hely a kosárban

Ciklus vége

Hátul tesztelő ciklus a C-ben

- A C nyelvi programszerkezet:

```
do
{
    ciklusmag utasításai
}
while (kifejezés);    //kifejezés: lefutási feltétel
```

- A **do-while** ciklus futása során mindig a ciklusmag utasításait követi a kifejezés kiértékelése. Amíg a kifejezés értéke nem nulla (igaz), addig új iteráció kezdődik. Amikor a kifejezés értéke a vizsgálat során először bizonyul nulla értékűnek (hamis), a vezérlés a **while** utáni utasításra adódik, a ciklus befejezi a működését.

Példa

- Készítsünk programot, amely bekér egy dátumot!

- Pontosítsuk a feladatot:
  - A hónap 1-12 közt kell legyen
  - A nap pedig 1-[az adott hónapban található napok száma] közt van
    - Szükség lesz egy tömbre a napok számával
  - Mit csináljuk ha rossz adatot ad meg a felhasználó?
    - Kérjük be újra az adatot

Példa

Program
Változók: hónap, nap (pozitív egész)
napokszáma[12] (tömb)

Ciklus
BE: Hónap
amíg (hónap<1 vagy hónap>12)
ciklus vége

Ciklus
BE: nap
amíg (nap<1 vagy
nap>napokszáma[honap])
Ciklus vége

Ki: hónap, nap
Program vége

- C program (részlet)

```
do
{
    printf ("\nKérem a hónapot (1-12):");
    scanf ("%f", &honap); fflush (stdin);
}
while ((honap < 1) || (honap > 12));
```

Ellenőrzött adat-bevitel

- A hátultesztelő ciklust gyakran alkalmazzuk ellenőrzött adatbevitelre. Ez azt jelenti, hogy addig folytatjuk egy meghatározott adat bekérését, amíg az az előírt határok közé nem esik.

Hátul tesztelő ciklus

- A hátul tesztelő ciklusnál a ciklusmag utasításait egyszer mindenképpen végrehajtja a rendszer. Ez nem mindig alkalmas szerkezet a feladataink megoldására. (ld. könyv vásárlás)

Léptetéses (növekményes) ciklus

Gyakran előforduló feladat, hogy valamely **tevékenység(ek)et előre megadott számszor kell végrehajtani**. Erre a feladatra a programnyelvek egy külön ciklus-utasítást biztosítanak, a növekményes, számláló, vagy léptetéses ciklust.

Növekményes (számláló) ciklus

Ciklus *változó* = **A-tól B-ig C Lépésközzel**

Ciklusmag utasításai

Ciklus vége

Működése:

A ciklusmag utasításai a ciklusváltozó kezdés végértékének, valamint a lépésszámnak megfelelő alkalommal (számszor) kerülnek végrehajtásra

Léptetéses ciklus C-ben

- C nyelvben a léptetéses ciklusnak egy általánosított változatát használjuk:

```
for (init_kifejezés; feltétel_kifejezés; léptető_kifejezés)
    // inicializálás                          módosító rész
{
    ciklusmag utasításai
}
```

- A leállást és léptetést bármilyen kifejezéssel leírhatjuk
- Bármelyik kifejezést elhagyhatjuk

Példa

```
int i=10;
for(  ;i<20;  )
{
        printf("%d",i);
        i++;
}
```

Növekményes ciklus, 1. feladat

Készítsen programot, amely n darab * karaktert ír ki a képernyőre.

Megoldás:

Program

Változók: n - darabszám, pozitív egész
i - ciklusváltozó, pozitív egész

Ciklus

Be: n

amíg (n<= 0)

ciklus vége

Ciklus i= 1 **kezdőértéktől** n **végértékig** 1 **lépésközzel**

Ki: "*"

ciklus vége

Program vége

Növekményes ciklus a C-ben

```
void main (void)
//for ciklus: n darab * karakter kiírása
{
  int i, n;        //i a ciklusváltozó, n a darabszám
  do
  {
    printf ("\nHány csillag legyen a képernyőn ?");
    scanf ("%d", &n); fflush (stdin);
  }  //do
  while (n <= 0);
  for (i=1; i<= n; i++)
  {
     printf ("*");
  }
  getch ();
}    // program vége
```

Ciklusok egymásba ágyazása

- Leggyakrabban a for ciklusokat szokás egymásba ágyazni.

- Feladat: készítsen szorzótáblát !

	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	8	10	12
3	3	6	9	12	15	18
4	4	8	12	16	20	24
5	5	10	15	20	25	30

Ciklusok egymásba ágyazása

Program
Változók : szorzo1, szorzo2: pozitív egész
Ki: "SZORZÓTÁBLA"
//Következik a fejléc-sor kiírása
Ciklus szorzo1 = 1 **kezdőértéktől** 9 **végértékig** 1-es **lépésközzel**
Ki: szorzo1
Ciklus vége //szorzo1
// a szorzótábla elemei
Ciklus szorzo2 = 1 **kezdőértéktől** 9 **végértékig** 1-es **lépésközzel**
Ki: szorzo2 //első oszlop elemei…
Ciklus szorzo1 = 1 **kezdőértéktől** 9 **végértékig** 1-es **lépésközzel**
Ki: szorzo1 * szorzo2
Ciklus vége *//szorzo1*
Ciklus vége *//szorzo2*
Program vége

Kiugrás a ciklusból

Keresse meg programmal a 2009 szám legnagyobb osztóját !

Elemezzük a feladatot:

Egy egész szám osztója legalább kétszer megvan a számban, tehát a legnagyobb osztó keresését elegendő a szám felénél kezdeni, majd egyesével haladni lefelé, amíg az első olyan számot megtaláljuk, amellyel osztva 2009-et, a maradék = 0

Pszeudokód:

Program

Változók : oszto pozitív egész

Ciklus oszto = 2009/2 **kezdőértéktől** 2 **végértékig** -1-es **lépésközzel**

Ha (2009 mod oszto = 0)

Ki: "A legnagyobb osztó:", oszto

kiugrás

ciklus vége

Ki: "A vizsgálat befejeződött."

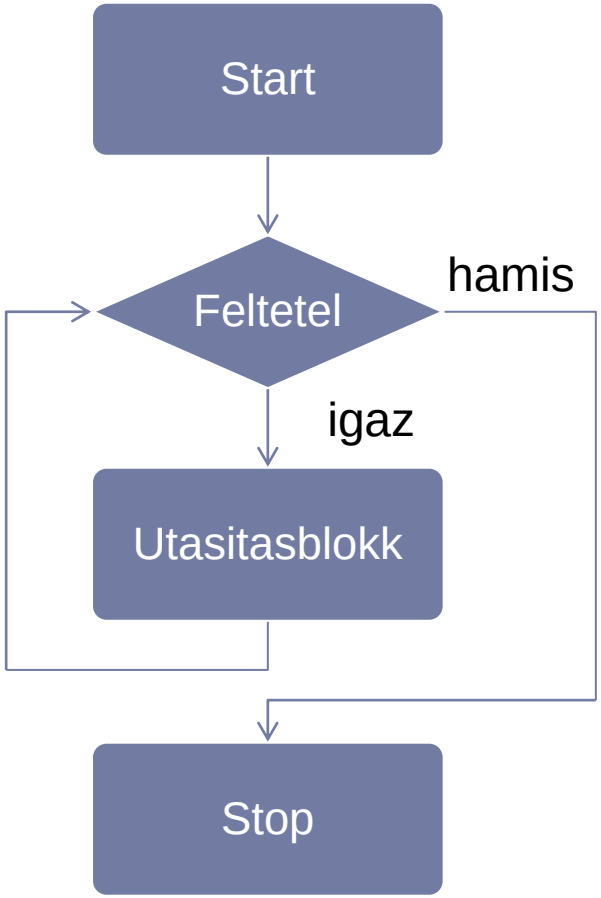
Program vége

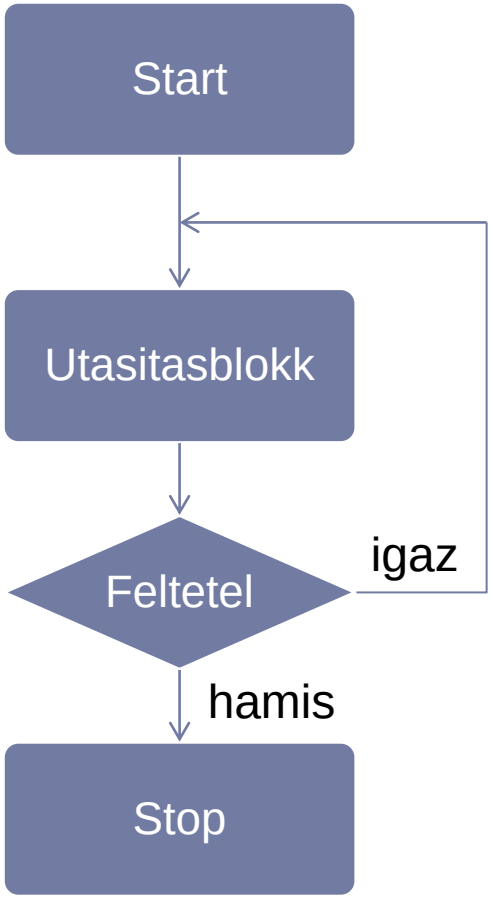
Kiugrás a ciklusból

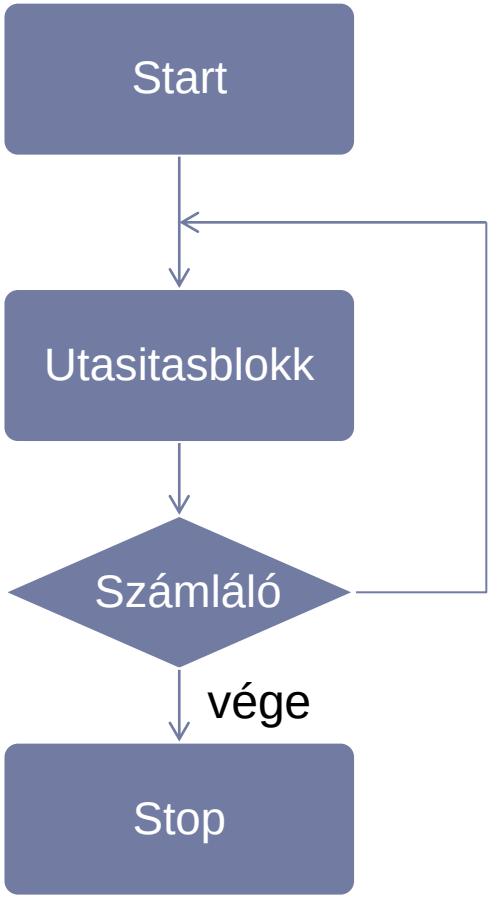
- Strukturált megoldás

Program
    Változók : oszto pozitív egész
    oszto = 2001/2
    Ciklus amíg 2001 mod oszto <> 0 ÉS oszto > 1
        oszto = oszto -1
    ciklus vége
     Ki: "A vizsgálat befejeződött.”
     Ha oszto > 1
            Ki: oszto
Program vége

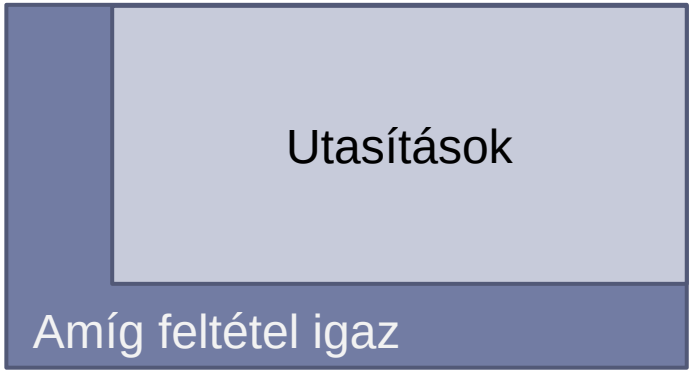
Ciklusok ábrázolása folyamatábrával

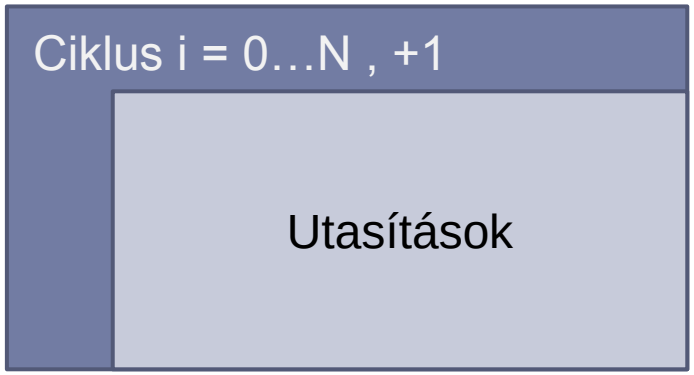






Ciklusok ábrázolása struktogrammal





Problémaosztályok, algoritmusok

Tömbök, rekordok

Összetett adattípusok

Két alapvető változat van:

- sok azonos jellegű elemből áll az összetett adattípus: **tömbök** (Pascalban: array)

- több különböző jellegû összetevõbõl áll az összetett adattípus: **struktúrák** (Pascalban: **record**) –

  C nyelven: struct, C++-ban class-nak fogjuk hívni (ez a struct általánosítása)

Információtárolás tömbökben

***Definíció* :**

- A tömb összetett adatszerkezet, véges számú, azonos típusú elem összessége, melyek a memóriában folytonosan helyezkednek el.

Fajtái :

- Vektor : egydimenziós tömb - egy elem megadásához egy index kell
- Mátrix : kétdimenziós tömb - egy elem megadásához két index kell
- Többdimenziós tömb

Egydimenziós tömbök

- Deklaráció:

  típus tömbnév[méret] ;

  méret: C-ben a fordító által kiszámítható

  Pl. int vektor[10] , v [20] ;

- Egy vektorelemre hivatkozás :

  tömbnév [index]

*az elemek indexelése C-ben **0**-tól **méret-1**-ig történik*

Pl. vektor[1], v[1] , v[5] ,
vektor[i], v[j]

Egydimenziós tömbök C-ben

- ***int v [10];***
- *A tömbindexek általában 1-től indulnak,*
  - *DE a C-ben az indexek mindig 0-tól indulnak !!!!!*
- *A vektor első elemére hivatkozás C-ben : v [0] !!!*
- *A vektor utolsó elemének indexe C-ben: elemszám-1*
- *Pl. v [0], v[1], v[2], ... , v[9]*
- *Nincs ellenőrzés az indexhatár túllépésére!!!!*

Egydimenzios tömbök

- **Tárolás a memóriában** :
  egy adott címtől kezdve folytonosan

- **Inicializálás (kezdőértékadás):**

  A deklarációban { } között felsoroljuk vesszővel elválasztva az elemek értékét is.

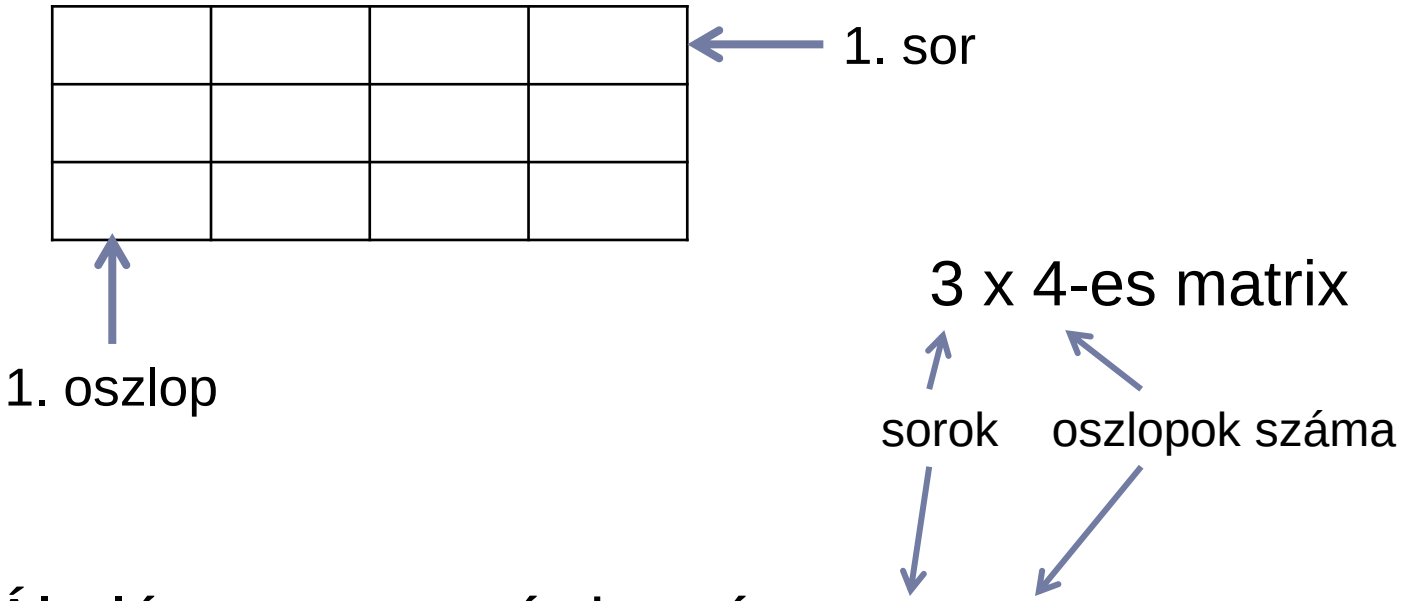
  Pl. int v1[5] = { 1, 2, 3, 4, 5 } ;

Többdimenziós tömbök

- *Példa: egy autó helyzete és sebessége:*
  - *közút száma*
  - *km szelvény*
  - *sebesség*
  - *időpont*
- Ebben a fejezetben csak a kétdimenziós tömbökkel foglalkozunk
- A többdimenziós tömbök hasonlóképpen kezelhetőek

Mátrixok

- Kétdimenzios tömbök



- Általánosan: a mátrix mérete n x m

Mátrixok

- Egy mátrixot kétféle módon tárolhatunk a memóriában :

- Sorfolytonos tárolás :

  Az elemek tárolása soronként történik,

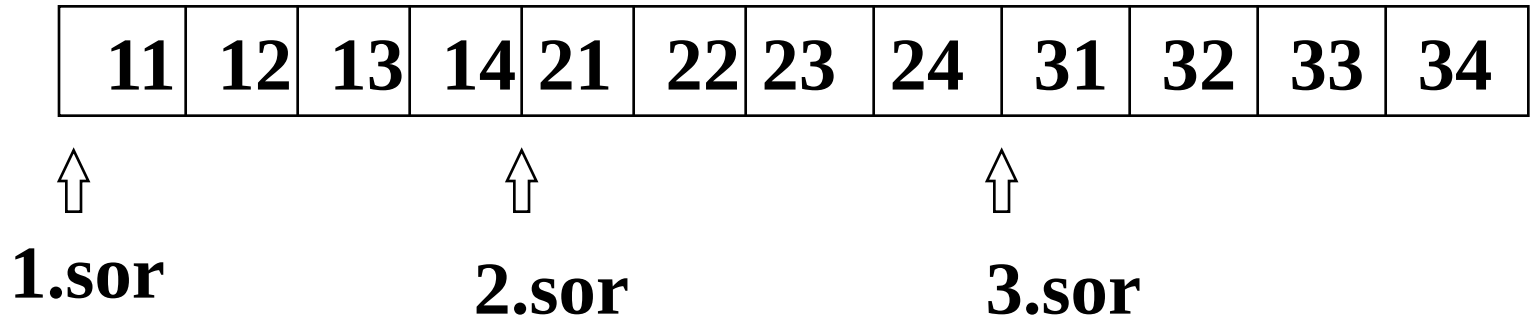
  azaz egy adott címtől kezdődően (mátrix kezdőcíme) először az 1. sor elemei, majd utánuk a 2. sor elemei , stb. kerülnek tárolásra.

- Így tárolja a mátrixokat a Pascal, C/C++,C#, Java...

- Pl. ha a mátrixunk a következő :

11	12	13	14
21	22	23	24
31	32	33	34

- Tárolása sorfolytonosan :



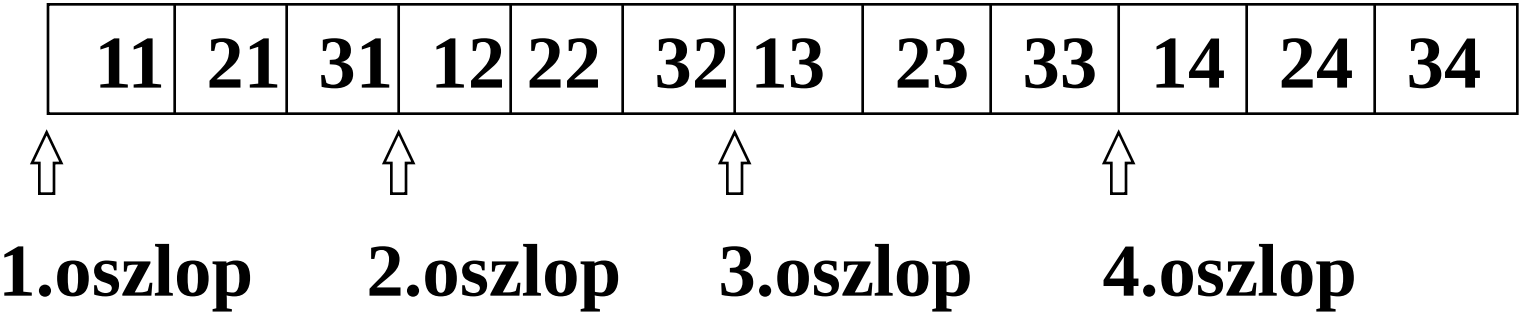
- **<u>Oszlopfolytonos tárolás</u>** :

  Az elemek tárolása oszloponként történik,

  azaz egy adott címtől kezdődően először az 1. oszlop elemei, majd a 2. oszlop elemei, stb. kerülnek tárolásra.

- Így tárolja a mátrixokat a FORTRAN.

Az előbbi mátrix oszlopfolytonosan tárolva :



Mátrixok

- Kétféle deklarálást engednek meg a programnyelvek :
- Egy n x m -es mátrix n*m darab elem összessége.
- A deklaráláskor a következő formában adjuk meg a mátrix méretét :
- Pl. Egy 3 x 4-es mátrix esetében:

  típus nev [3, 4]
- Egy elemre hivatkozás : nev [i , j]

Mátrixok

- Másik megadási mód:

- Egy n x m -es mátrixot úgy deklarálunk,
  mint egy n elemű vektort,
  melynek minden eleme egy m elemű vektor,
  azaz vektorok vektoraként.

- Pl. A 3 x 4 -es mátrix esetén :

  típus nev [3] [4]

- Egy elemre hivatkozás : nev [i] [j]

Mátrix

- A C-ben vektorok vektoraként kell megadni a mátrixot:
  nev [sor][oszlop]

- Pascalban mind a két féle deklarálást lehet használni:
  nev [sor, oszlop]
  nev [sor][oszlop]

Mátrixok

Deklaráció C-ben:

típus azonosító[sorok száma][oszlopok száma];

Pl. Egy 3 x 4 -es , integer típusú elemekből álló mátrix deklarációja :

```
int mx [ 3 ] [ 4 ] ;
```

Mátrixok

- *Mind a sor- , mind az oszlopindex 0 - tól indul !!*

- *Az utolsó sorindex értéke : sorok száma -1,*
- *Az utolsó oszlopindex értéke: oszlopok száma -1*

- *Egy elemre való hivatkozás :*

mx [0][0] , mx [0][1], ..., mx [2][3] ,

vagy mx[i][j]

Mátrixok: feladat

- **Feladat**:
- Rajzolj fel egy 3 sorból és 4 oszlopból álló táblázatot.
- Hogyan hivatkozhatunk a tömb egyes elemeire ?

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]

- *int a [3][4];*

Feladat

Egy háromfős társaság minden tagja 2-2 lottószelvényt vásárol minden héten. 5-ös lottón, állandó számokkal játszanak. Készítsen programot, amely tárolja a szelvényeken megjátszott tippeket, bekéri a nyerőszámokat és megmondja, hogy volt-e ötös a szelvények között.

Megoldás:

Elemezzük a feladatot

Összesen 6 szelvényünk (tipp-sorunk) van, egy-egy tipphez öt darab egész szám tartozik.

Mivel a társaság állandó számokkal játszik, a tippeket előre megadhatjuk a programban.

Bekérni csak a nyerőszámokat kell.

Feladat

Egy háromfős társaság minden tagja 2-2 lottószelvényt vásárol minden héten. 5-ös lottón, állandó számokkal játszanak. ...

A 6 szelvényen szereplő tippsor tárolása egy kétdimenziós tömbben történik, melynek egy-egy sora tartalmaz egy tippsort:

1. tipp:	11, 34, 45, 66, 89
2. tipp:	3, 13, 43, 52, 78
3. tipp:	25, 31, 72, 86, 90
4. tipp:	8, 15, 26, 48, 81
5. tipp:	19, 29, 39, 49, 69
6. tipp:	21, 32, 43, 54, 65

A C++-ban az alábbi tömböt fogjuk deklarálni :

```
int tippek [6][5];
```

Pszeudokód:

Program

Változók: tippek [6][5] a tippeket tartalmazó tömb, elemei pozitív egész számok, melyeket a deklaráláskor kell megadni

nyeroszamok [5] a heti nyerőszámok

i , j ciklusváltozók, poz. egész

egyezik segédváltozó, egész típusú

vanotos 0 ha nincs ötös, 1, ha van ötös a tippek között

Ciklus i = 0 **kezdőértéktől** i = 4 **végértékig**

Be: nyeroszamok [i]

Ciklus vége

```
vanotos = 0
 Ciklus i = 0 kezdőértéktől  i = 5 végértékig
     egyezik = 0
     Ciklus j = 0 kezdőértéktől  j = 4 végértékig
          egyezik = egyezik + (tippek [i][j] == nyeroszamok [j])
     Ciklus vége //  j
      Ha egyezik == 5
           kiugrás a ciklusból
 Ciklus vége //i

 Ha egyezik == 5
      Ki: "Van ötös !"
 egyébként
       Ki: "Nincs ötös !"
program vége
```

Feladat

- Készítsen programot, amely egy maximum 20 fős tanulócsoport zh eredményeit tárolja egy tárgyból. A csoport öt zh-t ír.
- Az adatbevitel úgy történik, hogy megadjuk a nevet, majd pontszámokat.
- A program minden zh-ra kiszámítja az átlagot és le is tárolja azt.
- A programnak tárolnia kell a hallgatók neveit is.

Pszeudokód:

Program

Változók: i, j, letszam *egész számok*

nevek [20][31] *char típ. tömb, soronként egy nevet tartalmaz*

pontok [20][5] *int típusú tömb, egy sora egy hallgató zh-pontszámait tartalmazza*

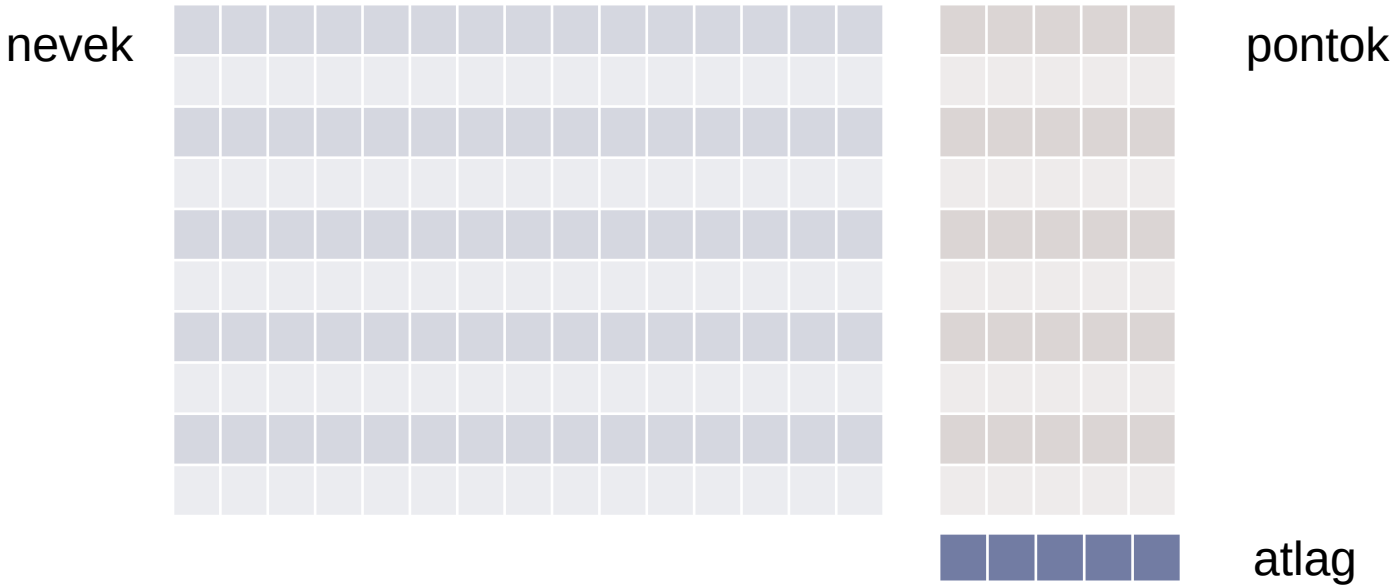
atlag [5] = [0,0,0,0,0] *valós típ. tömb, elemei a zh átlagok*

nevek [20][31] *char típ. tömb, soronként egy nevet tartalmaz*

pontok [20][5] *int típusú tömb, egy sora egy hallgató zh-pontszámait tartalmazza*

atlag [5] = [0,0,0,0,0] *valós típ. tömb, elemei a zh átlagok*

- Rajzoljuk fel a három tömböt



Be: letszam

//Névsor beolvasása

Ciklus i=0 **kezdőértéktől** i=letszam-1 **végértékig**

    Be: nevek [i]

Ciklus vége

//Pontszámok beolvasása

Ciklus i=0 **kezdőértéktől** i=letszam-1 **végértékig**

    Ki: nevek [i]

    Ciklus j=0 **kezdőértéktől** j=5-1 **végértékig**

        Be: pontszam [i][j]

    Ciklus vége

Ciklus vége

```
//Altag számítás
Ciklus i=0 kezdőértéktől i=letszam-1 végértékig
        Ciklus j=0 kezdőértéktől j=5-1 végértékig
        atlag [j] = atlag [j] + pontszam [i][j]
        Ciklus vége
Ciklus vége

//Átlagok kiírása
Ciklus j=0 kezdőértéktől j=5-1 végértékig
        Ki: j+1, atlag [j] / letszam
Ciklus vége

Program vége
```

- Tömb: azonos típusú objektumok összessége
- Feladat: árukészlet nyilvántartása

Adat neve	Típus
Megnevezés	char[30]
EAN Kód	long
Mennyiség	double
Mennyiségi egység	char[10]
Beszerzési ár nettó	int
Eladási ár nettó	int
ÁFA kulcs	int

- Tárolás: tömbökben ? memóriában ?

Struktúra típus

- Több, tetszőleges típusú (kivéve a void és a függvény típust) objektum együttese.
- Ezek az objektumok önálló, a rekordon (struktúrán) belül érvényes nevekkel rendelkeznek.
- Az objektumok szokásos elnevezése: mező, struktúraelem, vagy adattag

Struktúra definálása C-ben

```
struct aru_leiro_struct {                  //Struktúra deklaráció
        char megnevezes [30];              //az áru megnevezése
        long ean_kod;                      //az áru EAN kódja
        char me [10];                      //mennyiségi egység
        double mennyiseg;                  //mennyiség
        float beszerz_ar;                  //nettó beszerzési ár
        float elad_ar;                     //eladási ár nettó
        int afa_kulcs;                     //ÁFA kulcs %-os értéke
  } aru_leiro;                             //Típus deklaráció
```

- A típusdeklaráció után definiálhatunk az új típusnak megfelelő változókat:

  aru_leiro tej, vaj, sajt;
  aru_leiro tesco[1000];

Hivatkozás a struktúra elemeire

- Adjunk értékeket a tej nevű változó adattagjainak:

```
strcpy (tej.megnevezes,  ”Pécsi tej”);
tej.ean_kod = 9789635451784;
strcpy (tej.me, ”liter”);
tej.mennyiseg = 1000;
tej.besz_ar = 120;
tej.elad_ar = 180;
tej.afa_kulcs = 27;
```

- A struktúra adattagjaira való hivatkozásnál a pont operátort használjuk