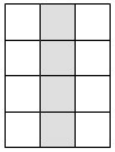
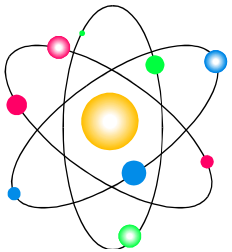
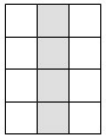
További vektor, mátrix algoritmusok



- *Egyindexes tömbökön értelmezett 5. alap-algoritmus: Osztályokba sorolás*
- *Mintaprogram*
- *Mátrix szorzása mátrixszal függvényben*
- *Mintaprogram*
- *A címszerinti és az értékszerinti adatátadás összehasonlítása*
- *Mintaprogram*



Egyindexes tömbökön értelmezett alap-algoritmusok:



[dr.Salánki József nyomán]

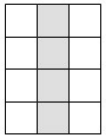
- ☑ *Összegezés*
- ☑ *Megszámlálás {hány adott tulajdonságú elem van?}*
- ☑ *Kiválogatás {nem a darabszám, az index érdekel}*
- ☑ *Rendezés*

Osztályokba sorolás

Pl.: egyszerű összegzés

```
main()
{
   float vektor[ ] = {1.2, 2.3, 3.4, 4.5} ;
   float szumma = 0 ;
   int k, elemszam;
   elemszam = sizeof (vektor) / sizeof (vektor [0]);
   for ( k = 0 ; k < elemszam; k++)
   {
        szumma += vektor [ k ];
   }
   printf("Szumma= %f", szumma);
}
```

Osztályokba sorolás



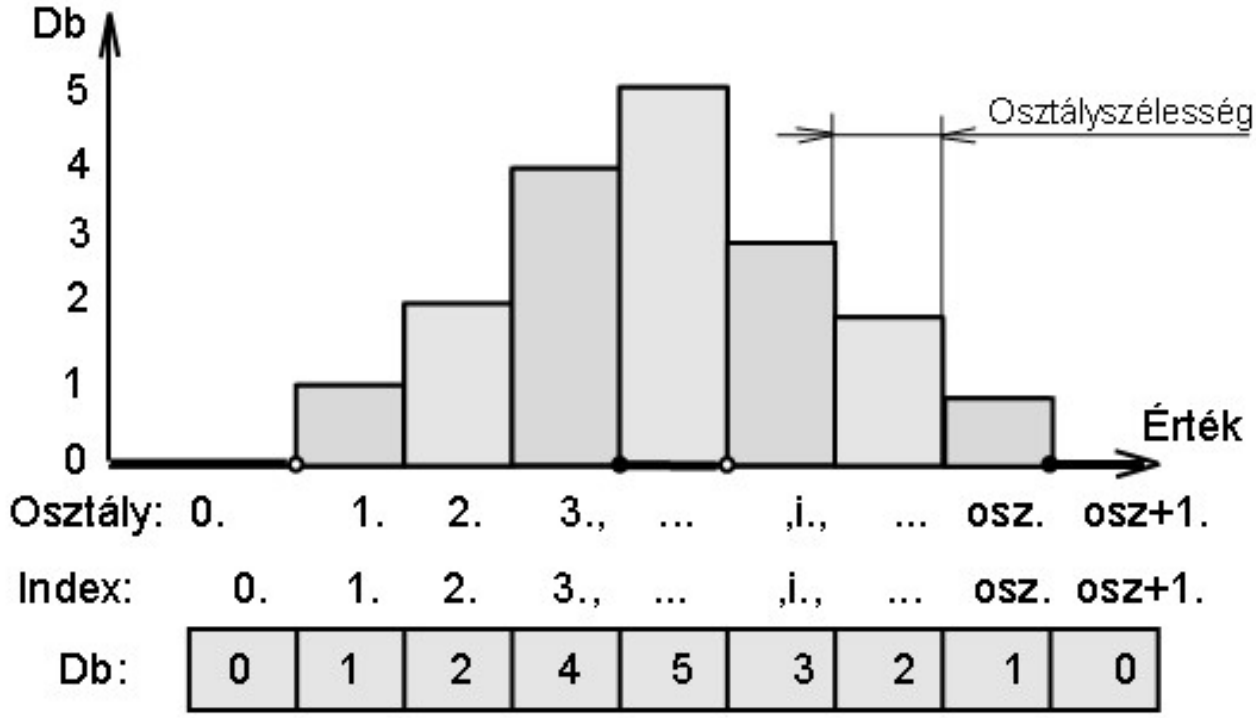
Feladat: Meg kell számlálni, hogy egy vektor elemei közül egy adott tulajdonságsorozat egyes tulajdonságainak hányan felelnek meg.

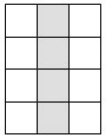
Pl.: *Mérési eredmények ábrázolása oszlopdiagrammal. Adott egy mérési sorozat eredménye egy n elemű valós vektorban.*

Készítsünk programot,
mely adott alsó és felső határok és az ezek közötti osztályok számának ismeretében meghatározza, hogy az egyes osztályokba hány mérési eredmény esik!

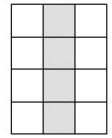
Osztályhatárra eső értéket soroljuk a nagyobb értékeket tartalmazó szomszédos osztályba. Számláljuk meg az alsó határ alatti és a felső határtól nem kisebb értékeket is!

Osztályokba sorolás ..

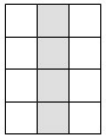




```
#include <stdio.h>
#include <math.h>
#define N 20          // mérési eredmények száma
#define OSZ 7         // osztályok száma
float ertekek[ N ];
unsigned int db[OSZ + 2];      // darabszámok vektora
float min, max;
unsigned int i;
// Itt következik a függvénydefiníció:
unsigned int index(float x, float xmin, float xmax, unsigned int
oszt_szama)
{
 float oszt_szelesseg = (xmax - xmin) / oszt_szama;

 if (x < xmin)  return 0;

 if (x >= xmax)  return oszt_szama + 1;

 return ((unsigned int) floor((x - xmin) / oszt_szelesseg)) +1;


}
```



```
void main()    // Itt következik a főfüggvény:
{   printf("Mérési eredmények beolvasása:");
    for (i = 0; i < N; i++)
    {
          printf("A %u. eredmény= ",i+1);
          scanf("%f",&ertekek[i]);
    }
   printf("Alsó határ= ");   scanf("%f",&min);
   printf("Felső határ= ");   scanf("%f",&max);
   for (i =0; i <= OSZ+1; i++)
   {
          db[ i ] = 0;    // Osztályok nullázása, inicializálása
   }
   for (i = 0; i < N; i++)
   {
          db[index(ertekek[ i ], min, max, OSZ)]++;
   }
   printf("\nDarabszámok kiíratása:");
   for ( i = 0; i <= OSZ+1; i++)
        printf("A %u. osztályba %u érték esik.\n", i, db[ i ]);
}    // Határokon számábrázolási korlátok miatt nem mindig pontos
```

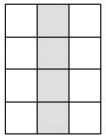


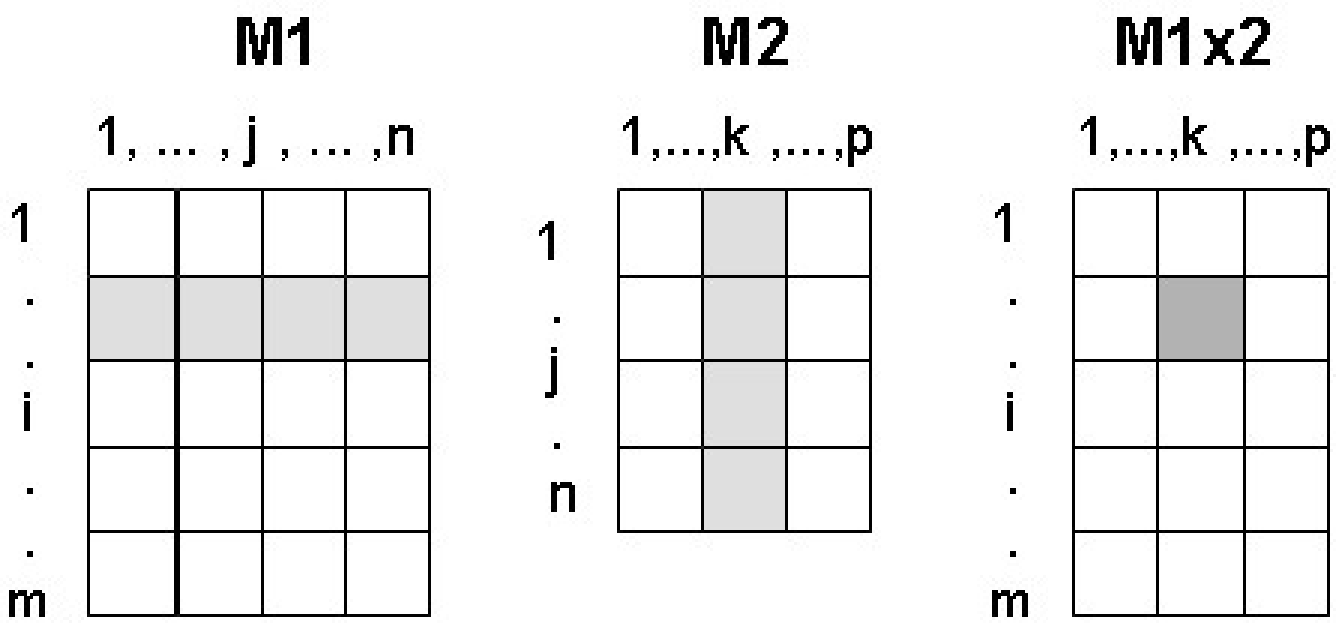
Megjegyzés:

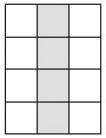
A floor( ) függvény a valós argumentumértéket a legközelebbi kisebb, vagy egyenlő egészre csonkítja.

Pl. : floor(6.25) értéke 6.0 lesz,
floor(27.0) értéke 27.0 lesz,
floor(-42.28) értéke –43.0 lesz.

Mátrix szorzása mátrixszal függvényben

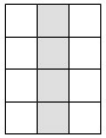




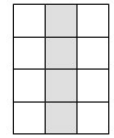


```
/* Mátrixszorzás */
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define M 5
#define N 4
#define P 3
float M1[M][N];
float M2[N][P];
float M1x2[M][P];
//A függvény deklarációja:
void mszorzas(int mm, int nn,  int pp, float M1p[][N],
                                  float M2p[][P], float M1x2p[][P]);
```

*Megj.: M2p[j][k] elem elérése: *((float*)M2p+(j*3)+k)*
látható, hogy szükség van az egy sorban levő elemek számát megadó P==3 értékre is.
Az első index megadható, de a fordító nem használja.
*float M2p[ ][P] jelent egy float** M2p mutatót és egy float* M2p[ ] mutatóvektort is, melynek elemei által mutatott vektorelemek mérete P*sizeof(float). P hiányában az M2p[j][k] == *(M2p[j] + k)==*(*(M2p+j)+k) elem nem érhető el pointeraritmetikával, azaz sorméretnyi (és elemméretnyi) léptetésekkel.*

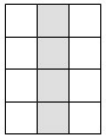


```
void main()
{
 int k, l;
 srand(time(NULL));
 for (k = 0; k < M; k++)
  for (l = 0; l < N; l++)
   M1[k][ l ] = rand() % 15;
 for (k = 0; k < N; k++)
     for (l = 0; l < P; l++)    M2[k][ l ] = rand() % 15;
 mszorzas(M,N,P,M1,M2,M1x2);
 // Kihasználtuk, hogy a tömb neve egy mutató konstans
 for (k = 0; k < M; k++)
  for (l = 0; l < P; l++)
   printf("M1x2[%u,%u]= %8.2f\n",k,l,M1x2[k][ l ]);
}
```



```
//A függvény definiálása:
void mszorzas(int mm, int nn, int pp, float M1p[ ][N],
                                      float M2p[ ][P], float M1x2p[ ][P])
{
   int i,j,k;
   for (i = 0; i < mm; i++)
     for (k = 0; k < pp; k++)
        for ( M1x2p[ i ][ k ] = 0, j = 0; j < nn; j++)
           M1x2p[ i ][ k ] += M1p[ i ][ j ] * M2p[ j ][ k ];
}
```

A címszerinti és az értékszerinti adatátadás összehasonlítása



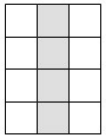
A címszerinti átadás (& operátorral)

- *szükséges, ha a függvény a paraméterén keresztül értéket ad vissza a meghívó programkörnyezetnek,*
- *előnyös, ha nagyméretű az adat, pl. többdimenziós nagy tömböknél, mert nem kell az értékeket átmásolni,*
- *előnytelen, ha a paraméter csak bemenő paraméter, mert nem adhatunk meg kifejezést aktuális értékként, csak változót.*

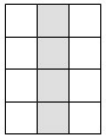
Az értékszerinti átadás

- *előnyös, ha a bemenő paraméter helyén aktuális paraméterként kifejezést is szeretnénk megadni,*
- *előnytelen, ha csak bemenő paraméterként nagyméretű adat átadására használjuk,(pl. struct)*
- *hibás, ha aktuális paraméterként változót megadva abban érték visszaadására számítunk.*

Példa:



```
 /* Értékadás */
#include <stdio.h>
#include <conio.h>
float elhossz, terulet;

void jo(float elhossz, float* teruletp)
{ *teruletp = elhossz * elhossz; }

void elonytelen(float* elhosszp, float* teruletp)
{ *teruletp = *elhosszp * *elhosszp; }

void hibas(float elhossz, float terulet)
{ terulet = elhossz * elhossz; }
```



```
void main()
{
 clrscr();
 jo( 4.7+7.3, &terulet );
 printf("Terület= %f\n",terulet);
 elhossz = 6.9 + 4.1;
 elonytelen( &elhossz, &terulet );
 printf("Terület= %f\n",terulet);
 hibas( 4.8+0.2, terulet );
 printf("Terület= %f ?? nem 25 ?\n",terulet);
 getch();
}
```