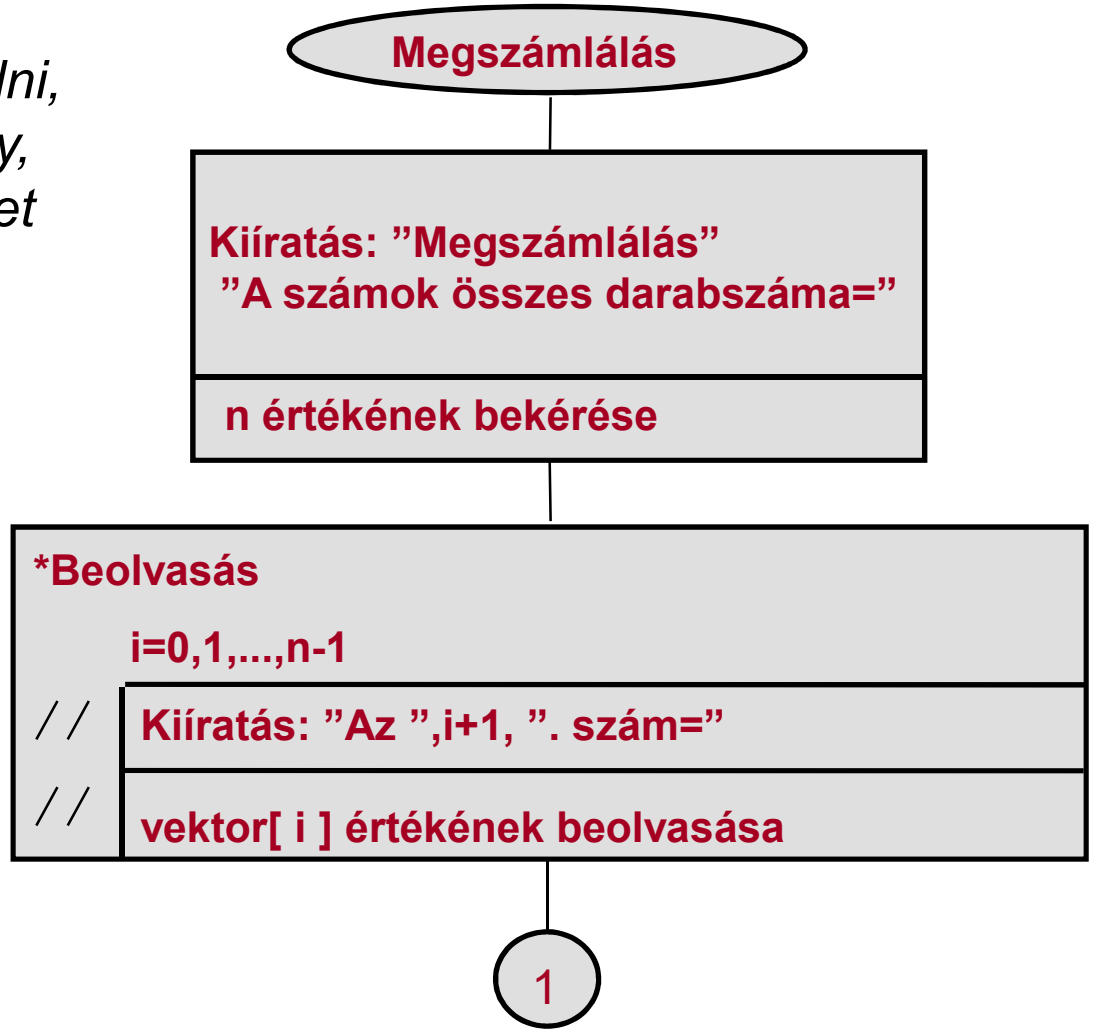
Megszámlálás, kiválasztás alapalgoritmusok

- *Vektoron értelmezett 2. alapalgoritmus: a megszámlálás algoritmusa*
- *Vektoron értelmezett 3. alapalgoritmus: a kiválasztás algoritmusa*
  - *Példa egyszerű logikai feltételnek eleget tevő elem kiválasztására*
  - *Elemkiválasztás összetett logikai feltétel alapján*

Vektoron értelmezett 2. alapalgoritmus: a megszámlálás algoritmusa

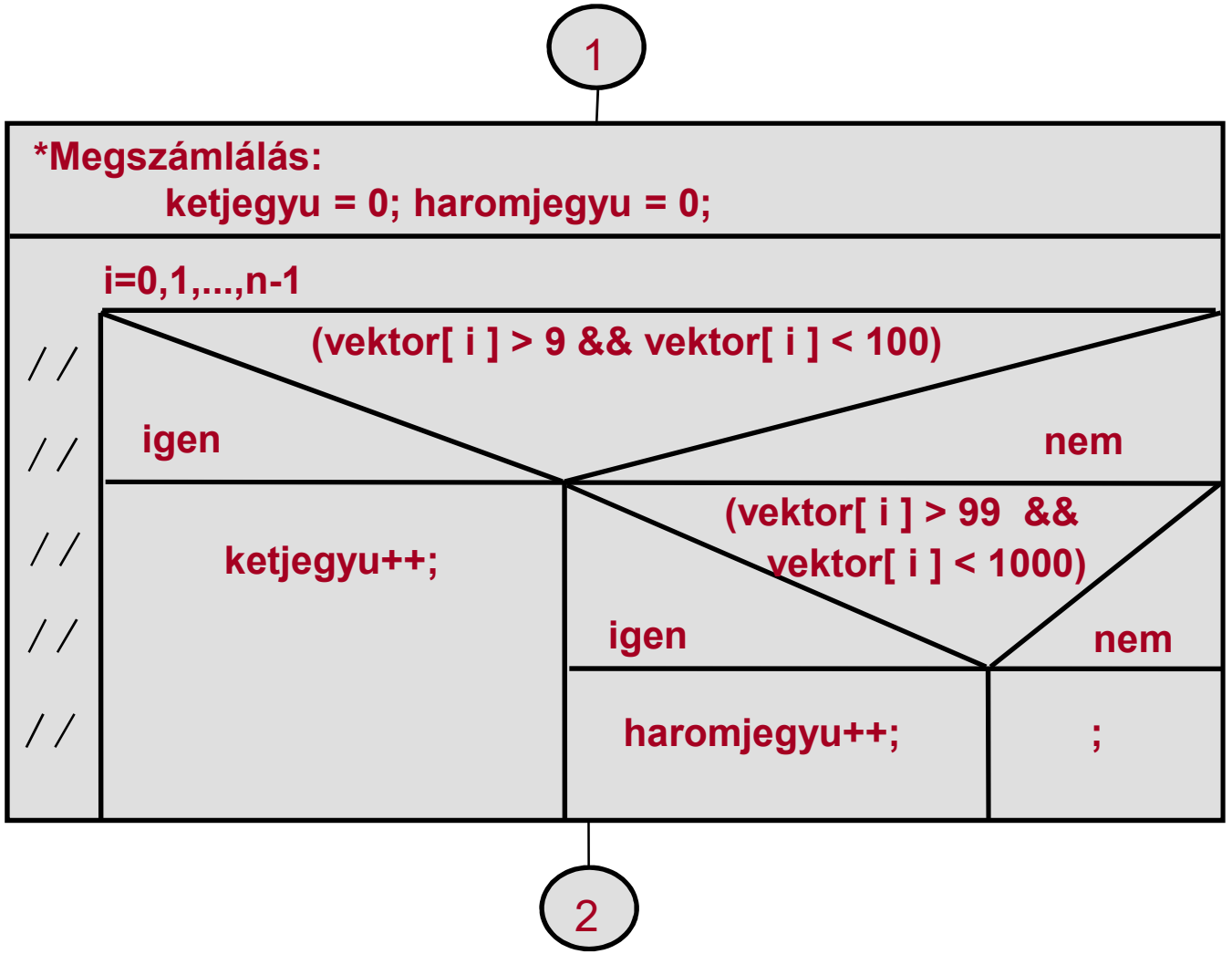
Feladat: *meg kell számlálni, hogy egy sorozatban hány, megadott feltételnek eleget tevő elem van.*

Példa: *Olvassunk be n darab pozitív egész számot egy vektorba, majd számláljuk meg, hogy hány kétjegyű és hány háromjegyű van közöttük !*

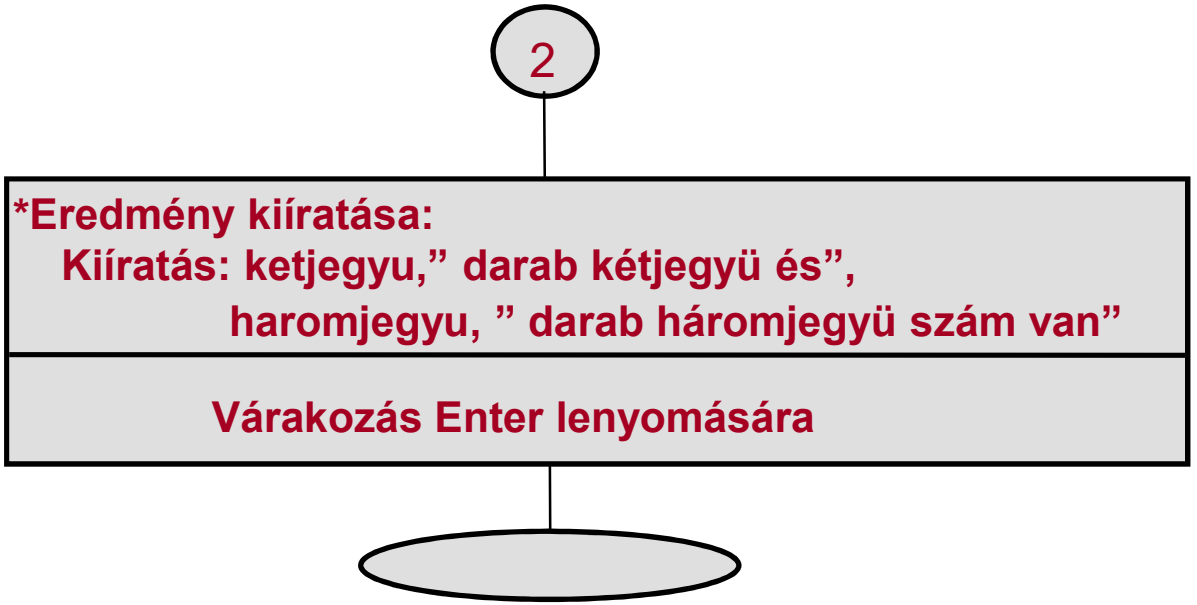


Chapin diagram == struktogram

A megszámlálás algoritmusa . .



A megszámlálás algoritmusa . .



A feladat struktogramja után lássuk a programját!

A megszámlálás algoritmusa . .

```
#include <stdio.h>
int vektor[100];
void main(void)
{
   int i, n, ketjegyu = 0, haromjegyu = 0 ;
   printf("  Megszamlalas \n");
   printf("Hany szamot olvassunk be?  ");
   scanf("%d", &n);
   // Beolvasás ciklussal
   for (i = 0; i < n ; i++)
   {
      printf("A %u. szám = ", i +1);
      scanf("%d", &vektor[ i ]);        // vagy scanf("%u", vektor+i);
   }
```

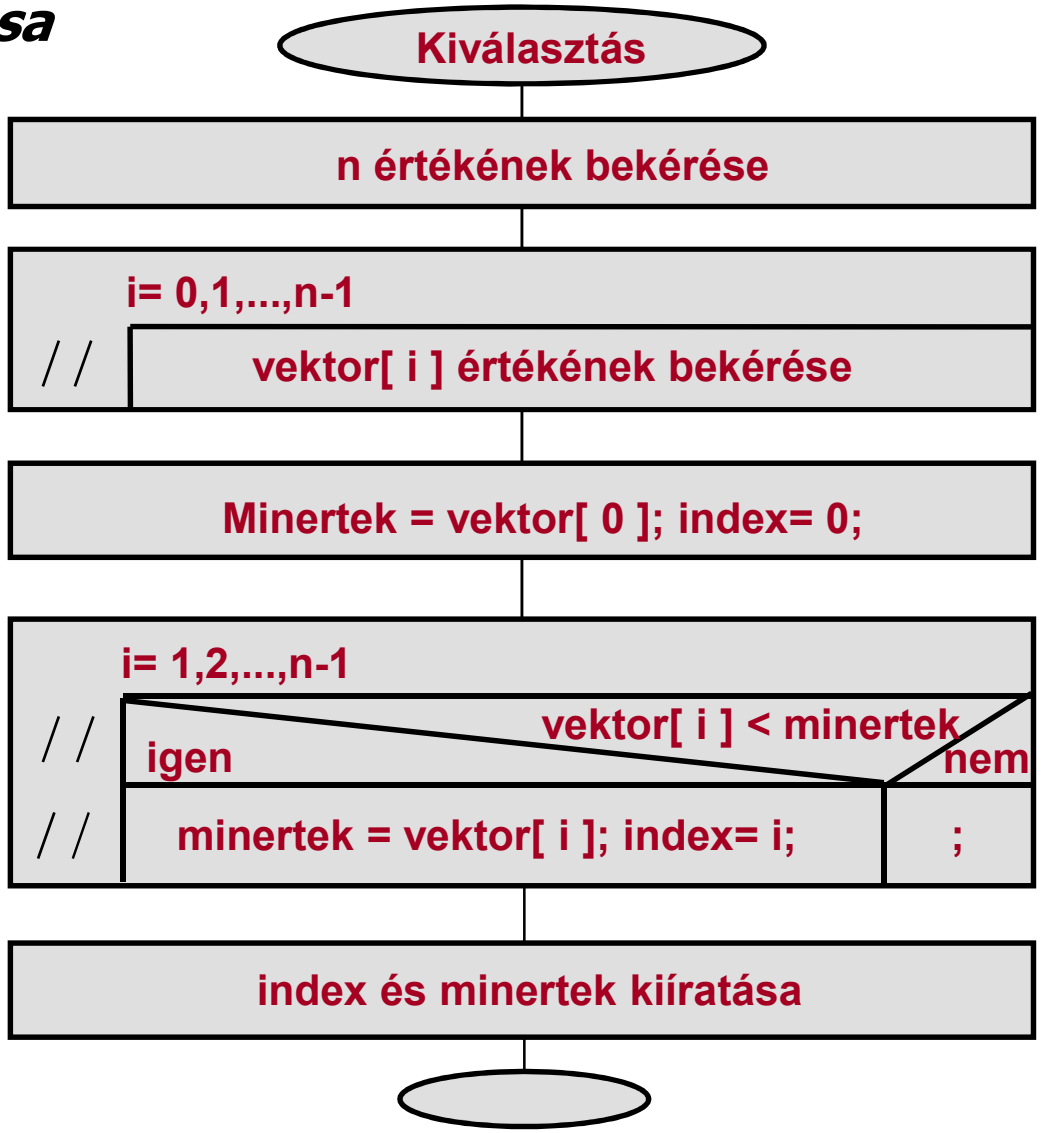
A megszámlálás algoritmusa . .

```
/* Megszámlálás: */
for (ketjegyu = 0, haromjegyu = 0, i = 0; i < n; i++)
{
    if (vektor[ i ] > 9 && vektor[ i ] < 100)
    {
        ketjegyu++;
    }
    else if (vektor[ i ] > 99 && vektor[ i ] < 1000)
    {
        haromjegyu++;
    }
}
// Eredmény kiíratása:
printf("\n %d darab kétjegyű és %d darab háromjegyű \
        szám van.\n", ketjegyu, haromjegyu);
getch();
}
```

Vektoron értelmezett 3. alapalgoritmus: a kiválasztás algoritmusa

Feladat: egy n elemű sorozatban megadott tulajdonsággal bíró elem sorszámának (indexének) meghatározása.

Példa egyszerű logikai feltételnek eleget tevő elem kiválasztására: a minimális elem kiválasztása.



A minimális elem kiválasztás programja

```
#include <stdio.h>
float vektor[100];
void main(void)
{
   int n, i, index ;
   float minertek ;
   printf("  Kiválasztás \n");
   printf("Elemek száma= "); scanf("%u", &n);
   // Beolvasás a vektorba
   for (i = 0; i < n; i++)
   {
      printf("Vektor[%d]= ", i +1);
      scanf("%f ", &vektor[ i ]);          /* vagy scanf(%f",vektor+ i ); */
   }
```

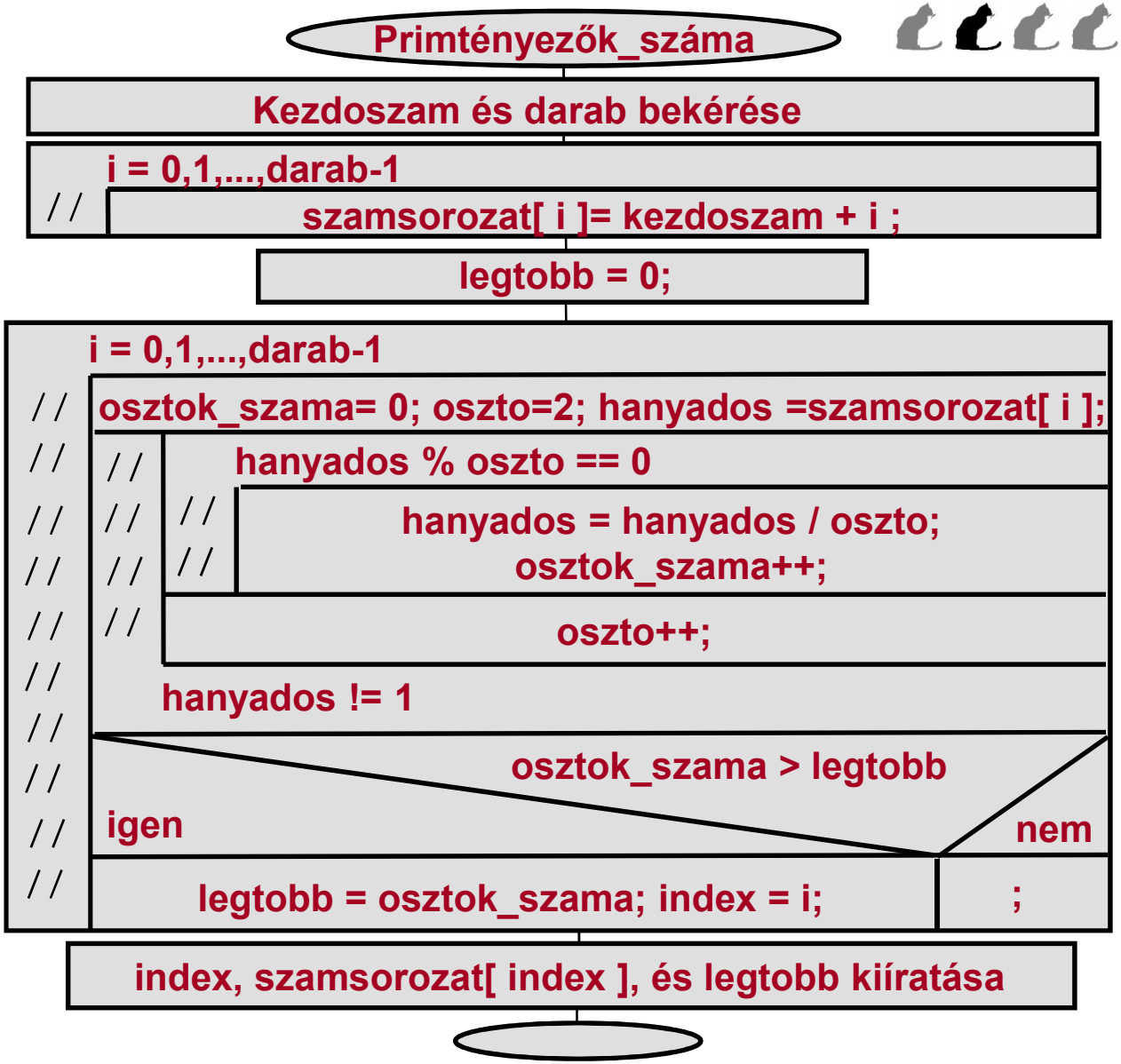
A minimális elem kiválasztás programja . .

```
// A legkisebb elem megkeresese
for (minertek =vektor [0], index =0, i =1; i < n; i++)
{
    if (vektor[ i ] < minertek)
    {
        minertek = vektor [ i ]; index = i ;
    }
}
// Az eredmény kiíratása
printf("\nIndex= %d, minimális érték =%f", index, minertek);
}
```

- ***Elemkiválasztás összetett logikai feltétel alapján***

Példa:

Adott egy egyesével növekvő számsorozat a kezdőszámmal és az elemek számával. Meghatározandó annak a számnak a sorszáma, amely a legtöbb prímtényezőre bontható fel.



Elemkiválasztás összetett logikai feltétel alapján . .

A primtényezők számát megadó program:

```
#include <stdio.h>
int szamsorozat [100];
void main(void)
{
    int kezdoszam, darab, oszto, i, index,
                osztok_szama, legtobb, hanyados;
    printf("Legtobb primtenyezo \n");
    printf("Kezdo szam=");
    scanf("%d",&kezdoszam);
    printf("\nSzamok darabszama="); scanf("%d", &darab);
    for (i = 0; i < darab; i++)
    {
        szamsorozat [ i ] = kezdoszam+i ;
    }
    legtobb = 0;
```

kezdoszam= 11; darab= 2; szamsorozat ☞ { 11, 12 }

Elemkiválasztás összetett logikai feltétel alapján . .

```
for (i = 0; i < darab; i++)
{
        osztok_szama = 0; oszto = 2;  hanyados = szamsorozat [ i ];
        do
        {
            while (hanyados % oszto == 0)
            {
                hanyados /= oszto;                  // egesz osztas!
                osztok_szama++;
            }
            oszto++;
        } while (hanyados != 1);
        if (osztok_szama > legtobb)
        {
            legtobb = osztok_szama; index= i ;
        }
    }
    printf("\nIndex= %d, a szám= %d, tenyezok szama= %u\n",
            index, szamsorozat[index], legtobb);
}
```