ANSI C NYELVEN

22-23. ÓRA

Kiválasztás tétele
Adott egy N elemű sorozat ('A' tömb) és egy T tulajdonság. Tudjuk, hogy a sorozatnak van legalább egy T tulajdonságú eleme. Feladat ezen elem sorszámának meghatározása. *A megoldás szöveges algoritmus leíró módon a következő:*

```
ELJÁRÁS Kiválasztás

    i := 0
    CIKLUS AMÍG A[i] nem T tulajdonságú
           i := i+1
    CIKLUS VÉGE

    sorszam := i

ELJÁRÁS VÉGE
…
```

Keresés tétele
A tétel az **eldöntés** és a **kiválasztás** tételének egyesített algoritmusa. Azaz adott egy N elemű sorozat ('A' tömb) és egy T tulajdonság. Feladat annak meghatározása, hogy van-e a tömbben olyan elem, amely rendelkezik a T tulajdonsággal. Amennyiben van, akkor meg kell adnunk annak sorszámát.

A feladat többféle módon is elvégezhető. Az egyik lehetőség, amennyiben a tömb nem rendezett, hogy minden elemen végigmegyünk és megvizsgáljuk, hogy az adott elem T tulajdonságú-e (lásd eldöntés tétele). A másik lehetőség, ha a tömb rendezett, ebben az esetben gyorsabban is elvégezhető a keresés.

A keresés lehet
– lineáris (rendezetlen tömbök esetén)
– logaritmikus keresés (rendezett tömbök esetén) => **KÉSŐBB!**

Lineáris keresés
A keresésnek ebben a formájában, sorban egymás után nézzük végig a tömb elemeit, amíg meg nem találjuk a keresett elemet, vagy végig nem néztük a tömböt. Mivel sorban egymás után vizsgáljuk az elemeket, ezért nevezzük a keresést *lineáris* keresésnek.

A lineáris keresés megoldása szöveges algoritmus leíró módon a következő:

```
ELJÁRÁS LinearisKereses

    i := 0
    CIKLUS AMÍG i<N ÉS A[i] nem T tulajdonságú
           i := i+1
    CIKLUS VÉGE

    HA i<N AKKOR
           sorszam := i
           Ki: sorszam
    EGYEBKÉNT
           Ki: "Nincs ilyen elem"

ELJÁRÁS VÉGE
…
```

Keresés speciális esetei
– legkisebb érték keresése (minimum keresés)
– legnagyobb elem keresése (maximum keresés)

Ezt a két esetet együttesen szélsőértékek keresésnek is nevezik. A keresés megkezdésekor a tömb első elemét kijelöljük, mint a legkisebb, vagy legnagyobb elemet. Majd a *teljes tömb* minden elemére megvizsgáljuk, hogy nem kisebb-e (nagyobb-e), mint a kijelölt elem. Amennyiben kisebb (nagyobb), akkor megjegyezzük annak értékét, valamint indexét, mint legkisebb (legnagyobb) elem, és folytatjuk a vizsgálatot a tömb végéig.
Legkisebb elem keresésének megoldása szöveges algoritmus leíró módon a következő:

```
ELJÁRÁS LegkisebbKereses

    min = A[0]
    minindex = 0
    i := 1
    CIKLUS AMÍG i<N
           HA A[i]<min AKKOR
                  min = A[i]
                  minindex = i
           i++;
    CIKLUS VÉGE

ELJÁRÁS VÉGE
…
```

Az eljárás végén a **min** változó a legkisebb elem értékét, még a **minindex** annak 'sorszámát' tartalmazza!

Feladat – I.
Kérjen be a felhasználótól egy értéket (n), amely 1 és 10 között legyen. Az értéknek megfelelően olvassa be az 'n' elemű 'a' tömb értékeit, amelyek 10 és 99 között legyenek. A felhasználót tájékoztatnia kell arról, hogy az egyik elemnek KERESETT értékűnek kell lennie. Ezt követően vizsgálja meg, hogy a programban konstans értékként rögzített KERESETT értékű elem hányadikként került megadásra a számok között.

Megoldás ANSI C99 forráskódja

```
#include <stdio.h>

#define MAX_ELEM 10
#define KERESETT 55

int main(void) {
int n, a[MAX_ELEM];
int i, sorszam;

    // adatok bekerese felhasznalotol

    printf("\nElemek szama [1-10]:");
    scanf("%d", &n);

    printf("\nElemek kozott lennie kell %d erteku elemnek!\n", KERESETT);
    for (i=0; i<n; i++) {
           printf("- %2d. elem erteke [10-99]: ", i+1);
           scanf("%d", &a[i]);
    }

    // adott ertek keresese

    for (i=0; a[i]!=KERESETT; i++) { ; }
    sorszam = i;

    // eredmeny kijelzese

    printf("\nA tomb %d. eleme a(z) %d erteku elem!", i+1, KERESETT);
    return (0);
}
```

Házi feladat
Készítsük el a **legnagyobb** elem keresését biztosító algoritmust!