# 11 Műveletek tömbökkel

- legkisebb, legnagyobb elem megkeresése
- tömb elemeinek összeadása
- tömb tükrözése
- tömbök rendezése
  - egyszerű cserés rendezés
  - minimumkiválasztásos rendezés
- két rendezett tömb összefésülése

## 11.1 Legkisebb, legnagyobb elem megkeresése

Készítsünk programot, amely egy 20 elemű tömbbe kigenerál 1 és 150 közötti véletlen számokat, majd az elemek kiírása után megkeresi a tömbben található legkisebb és legnagyobb elemet.

Ehhez bevezetünk két változót: **min** és **max**. Kezdetben beállítjuk mindkét változó értékét a tömb első elemének értékére - feltételezve hogy ez a legkisebb és legnagyobb elem is a tömbben, majd a második elemtől a tömb végéig végignézzük az elemeket (van-e a beállított első elemnél nagyobb vagy kisebb). Ha bármelyik elem kisebb mint a **min** változó értéke, akkor az új elem értékét megjegyezzük a **min** változóban. Hasonlóan, ha olyan elemet találunk, amely nagyobb mint a **max** értéke, akkor azt megjegyezzük a **max** változóban. Így a ciklus lefutása (tömb elemeinek átnézése) után a **min** és a **max** változók a tömb legkisebb ill. a legnagyobb elemét fogja tartalmazni. Programunk így néz ki:

```
program Pelda27a;
uses crt;
const n=20; {a tömb elemeinek a száma}
var a: array[1..n] of integer;
    i,min,max: integer;
begin
 clrscr;
 randomize;
 for i:=1 to n do a[i]:=random(150)+1;
 for i:=1 to n do write(a[i],', ');
 writeln;
 {a tömb legkisebb és legnagyobb elemének keresése...}
 min:=a[1];
 max:=a[1];
 for i:=2 to n do begin
                  if a[i]<min then min:=a[i];
                  if a[i]>max then max:=a[i];
                  end;
 writeln('Legkisebb: ',min);
 writeln('Legnagyobb: ',max);
readln;
end.
```

## 11.2 Tömb elemeinek összeadása

Bővítsük tovább az előző programunkat. Egészítsük ki egy olyan programrésszel, amely összeadja a tömb elemeit, majd kiírja az összeget.

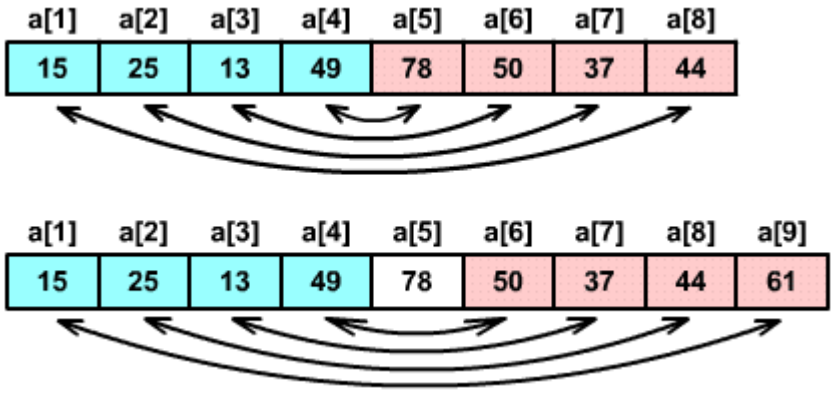
Ehhez bevezetünk egy újabb változót - **osszeg**, melynek kezdeti értékét beállítjuk 0-ra, majd a ciklus segítségével (amit módosítottunk, hogy 1-től menjen) végigmegyünk a tömb elemein és ehhez a változóhoz sorban hozzáadjuk a tömb első, második, ... utolsó elemét. Így a ciklus lefutása után az **osszeg** változó a tömb eleminek összegét fogja tartalmazni. Módosított programunk így néz ki:

```
program Pelda27b;
uses crt;
const n=20; {a tömb elemeinek a száma}
var a: array[1..n] of integer;
    i,min,max,osszeg: integer;
begin
clrscr;
 randomize;
 for i:=1 to n do a[i]:=random(100)+1;
 for i:=1 to n do write(a[i],', ');
 writeln;
 {a tömb legkisebb és legnagyobb elemének keresése }
 {az összeadással kibővítve... }
 min:=a[1];
 max:=a[1];
 osszeg:=0;
 for i:=1 to n do begin
                  if a[i]<min then min:=a[i];
                  if a[i]>max then max:=a[i];
                  osszeg:=osszeg+a[i];
                  end;
 writeln('Legkisebb: ',min);
 writeln('Legnagyobb: ',max);
 writeln('Az elemek osszege: ',osszeg);
 readln;
end.
```

### 11.3 Tömb tükrözése

Most készítsünk egy újabb programot. A program elején egy N=20 elemű tömbbe (N egy konstans legyen) generáljunk 10 és 99 közötti véletlen számokat, majd írjuk ki a tömböt. Ezek után tükrözzük a tömböt, tehát az első elemet cseréljük ki az utolsóval, a másodikat az utolsó előttivel, stb. Az így módosított tömböt írjuk ki újból a képernyőre.

A tömb elemeinek cseréjét (a tükrözést) egy 8 és egy 9 elemű tömbön az alábbi ábra szemlélteni:




Ebben is láthatjuk, hogy a tükrözéshez valójában egy olyan ciklusra van szükségünk, amely páros számú elemek esetében egytől a tömb feléig, páratlan számú elemek esetében a felénél eggyel kevesebbig megy (az ábrán a kék elemek) és elvégzik a megfelelő cserét. Ezt egyszerűen egy olyan for ciklussal tudjuk elvégezni, melynek ciklusváltozója **1**-től **(n div 2)**-ig megy.

A cikluson belül valójában melyik elemeket kell cserélni melyikkel? Az **1.** elemet az **n**-dik elemmel, a **2.** elemet az **(n-1)**-dik elemmel, a **3.** elemet az **(n-2)**-dik elemmel, stb. Tehát ha vesszünk az említett for ciklus ciklusváltozóját (**i**), akkor az **i**-dik elemet mindig az **(n-i+1)**-dik elemmel kell kicserélnünk (a cserét egy **x** segédváltozó segítségével végezzük el). Programunk tehát így néz ki:

```
program Pelda28;
uses crt;
const n=20;
var a: array[1..n] of integer;
    i,x: integer;
begin
clrscr;
 randomize;
 for i:=1 to n do a[i]:=random(90)+10;
 {a tömb elemeinek kiírása...}
 for i:=1 to n do write(a[i]:3);
 writeln;
 {a tömb tükrözése...}
 for i:=1 to n div 2 do begin
                        x:=a[i];
                        a[i]:=a[n-i+1];
                        a[n-i+1]:=x;
                        end;
 {a tömb elemeinek kiírása...}
 for i:=1 to n do write(a[i]:3);
 readln;
end.
```

## 11.4 Tömbök rendezése

Egy N elemű sorozatot nagyság szerint sorba kell rendezni. Nagyon sok rendezési algoritmus létezik. Az alábbiakban ezek közül kettőt fogunk megvizsgálni. Közös vonásuk az lesz, hogy az eredmény, a rendezett sorozat, helyben keletkezik, így az eredeti sorrend elvész. Az a rendezési algoritmus jó, amelynek kicsi a tárigénye és nagyon gyorsan végrehajtja a rendezést, és persze egyszerű, könnyen megérthető a működése.

Tökéletes, minden igényt kielégítő rendezési eljárás nem létezik. Az, hogy mikor melyik rendező algoritmust használjuk, sok összetevő függvénye. Tisztázni kell egy-egy konkrét eljárás kiválasztásakor, hogy az algoritmus elsősorban gyors legyen vagy inkább helytakarékos. Az alábbiakban tárgyalt algoritmusokbanaz N=5 elemű A() vektort fogjuk növekvő sorrendbe rendezni.

| | 1. | 2. | 3. | 4. | 5. |
|---|---|---|---|---|---|
| A() | **5** | **3** | **9** | **1** | **7** |

rendezve

| | 1. | 2. | 3. | 4. | 5. |
|---|---|---|---|---|---|
| A() | **1** | **3** | **5** | **7** | **9** |

### 11.4.1 Egyszerű cserés rendezés

Hasonlítsuk össze a() tömb első elemét a sorozat összes többi mögötte lévő elemével, s ha valamelyik kisebb nála, akkor cseréljük meg azzal! Ezzel elérhetjük, hogy a sorozat első helyére a legkisebb elem kerül. Folytassuk ugyanezen elven a sorozat második elemével, utoljára pedig az utolsó előttivel.

| | i ↓ | j ↓ | | | | |
|---|---|---|---|---|---|---|
| | 1. | 2. | 3. | 4. | 5. | |
| A() | **5** | **3** | **9** | **1** | **7** | ➔ csere |