

## Tartalomjegyzék

<b>Algoritmusok - pszeudókód</b> .....	<b>1–28</b>
Abszolút érték .....	1
Hányados ismételt kivonással .....	1
Legnagyobb közös osztó .....	1
Páros számok szűrése .....	1
Palindrom számok .....	2
Orosz szorzás .....	2
Minimum keresés .....	2
Maximum keresés .....	2–3
Eukleidész algoritmus .....	3
Prímszámok .....	3–4
Fibonacci-számok .....	4
Háromszög .....	4–5
Fordított szám .....	5
Törzstényezők .....	5–6
Prímszámvizsgálat .....	6
Konverzió – Számrendszer átalakítás .....	7
Gyors hatványozás .....	7–8
Szekvenciális (lineáris) keresés .....	8
Megszámlálás .....	8
Minimum- és maximumkiválasztás .....	8
A Maximum helye .....	8–9
Kiválogatás .....	9
Szétválogatás .....	9
Sorozat halmazra alakítása .....	9–10
Sorozatok keresztmetszete .....	10
Sorozatok egyesítése .....	11
Sorozatok összefésülése .....	11
Párok sorszáma egy sorozatban .....	12
Arány .....	12
Teljes négyzet .....	12
Osztályátlagok szétválasztása .....	12–13
Büvös négyzet .....	13–14
Polinom értéke adott pontban .....	14
Polinomok összege .....	14
Polinomok szorzata .....	14–15
Buborékrendezés (Bubble-sort) .....	15
Egyszerű felcseréléses rendezés .....	15
Válogatásos rendezés .....	16
Minimum/maximum kiválasztásra épülő rendezés .....	16
Beszűrő rendezés .....	16–17
Leszámláló rendezés .....	17
Összefésülésen alapuló rendezés .....	17–18
Gyorsrendezés (QuickSort) .....	18
Szavak sorrendjének megfordítása .....	18

Faktoriális.....	18–19
Számjegyösszeg .....	19
k elemű részhalmazok .....	19
Konverzió.....	19
Az { 1, 2, ..., n } halmaz minden részhalmaza.....	20
Kamatos kamatok kifrása .....	20
Általános backtracking .....	20–21
Általános rekurzív backtracking .....	21
Elhelyezni 8 királynőt a sakk táblán .....	21–22
Zárójelek .....	22
Játékok dobozva való elhelyezésének kifrása .....	22–23
X pénzüsszeg kifizetése n bankjegy segítségével .....	23–24
X Összeg kifizetése, minimum számú bankjeggyel.....	24
Általános Divide Et Impera .....	24
Szorzat (DivImp).....	25
Minimumszámolás (DivImp) .....	25
Hatványozás (DivImp) .....	25–26
Bináris keresés (DivImp).....	26
Általános mohó (Greedy) algoritmus .....	26
Összeg (Greedy) .....	26–27
Hátizsák probléma (Greedy).....	27
Összegkifizetés legkevesebb számú bankjeggyel (Greedy)..	27–28
<b>A Pascal nyelv elemei .....</b>	<b>28–35</b>
Azonosítók .....	28
Alapértelmezett egyszerű típusok.....	28
Változók .....	29–30
Konstansok.....	30–31
Egész típusú konstansok .....	30–31
Valós típusú konstansok .....	31
Karakter és karakterlánc típusú konstansok .....	31
Logikai konstansok .....	31
Kezdőértékkel rendelkező változók.....	32
Adatok beolvasása és kifrása .....	32
Szabványos függvények és eljárások.....	32
Matematikai függvények és eljárások.....	33–34
Sorszámozott típusú adatokra vonatkozó függv. és elj.....	34–35
<b>Egyszerű progr. készítése – Elementáris algoritmusok ...</b>	<b>36–37</b>
Kifejezések.....	36
Bitszintű műveletek.....	37
Relációs műveletek.....	37
<b>Pascal nyelvű programozás.....</b>	<b>37–62</b>
Döntések.....	37–41
Nagyobbik szám kifrása .....	37
Csökkenő sorrend létrehozása .....	38
Római szám felismerése .....	38–39
Római szám felismerése (case utasítással).....	39
Kis- nagybetű vagy más karakter .....	39–40

Jegyértékelés.....	40
Hónapok napjainak száma .....	40–41
Ciklusok .....	41
Legnagyobb közös osztó (Eukleidész algoritmus).....	41
Prímszámvizsgálat .....	41–42
Szám számjegyeinek száma .....	42
Törzstényezőkre való bontás.....	42–43
N szám összege .....	43
Fibonacci sorozat n-edik eleme.....	43–44
Egydimenziós tömbök.....	44–47
Átlagos jegynél kisebbek kiírása.....	44
Számsorozat kiírása fordítva.....	44–45
Sorozatból való teljes négyzetek összege.....	45
Karakterláncban felmerülő betűk száma.....	45–46
Két polinóm szorzata .....	46–47
Kétdimenziós tömbök.....	47–49
Két félévi átlagok.....	47
Négyzetes mátrix szimmetriája.....	47–48
Mátrix páros sorai elemeinek középarányosa .....	48–49
Karakterláncok .....	49–52
Relációs műveletek karakterláncokkal.....	50
Karakterlánc kezelő függvények.....	50–51
Karakterláncokat feldolgozó eljárások.....	51–52
Szóköz törlése egy karakterláncból.....	52
Karakterláncban folytatott szócsere.....	52
Halmazok .....	52–55
Műveletek halmazokkal.....	53
Halmazokra vonatkozó relációk és vizsgálatok .....	53–54
Egy halmaz elemeinek a kiírása.....	54
Karakterlánc betűinek kiírása ábécésorrendben.....	54
Erasztoténész algoritmus (prímszámok).....	54–55
Rekord vagy bejegyzés típusok.....	55–57
Alkalmazottak adatainak nyilvántartása.....	56–57
Kereső algoritmusok.....	57–58
Lineáris keresés.....	57
Bináris keresés.....	58
Rendező algoritmusok.....	58–60
Buborékrendezés.....	58–59
Minimumkiválasztásos rendezés.....	59–60
Beszűrős rendezés.....	60
Összefésülések .....	61–62
Összefésülés strázsa nélkül.....	61
Összefésülés strázssal (ütközővel).....	62
<b>Rekurzivitás .....</b>	<b>62–73</b>
Faktoriális kiszámítása .....	63
Fibonacci sorozat n-edik eleme .....	63
Két szám legnagyobb közös osztója.....	63–64

Hatványozás .....	64
10-es számrendszerből való alakítás .....	65
Első $n$ páratlan szám összege .....	65
Természetes szám számjegyeinek összege .....	66
Számsorozat negatív elemeinek száma .....	66–67
Szám előfordulása egy sorozatban .....	67
Vektor páros elemeinek összege .....	67–68
Sorozat legnagyobb prímszáma .....	68–69
$N$ hosszúságú $a$ és $b$ karakterű sorozatok .....	69–70
Természetes szám particiói .....	70–71
Labirintus-feladatok .....	71–73
<b>„Oszd meg és uralkodj” „Divide Et Impera” .....</b>	<b>74–78</b>
Sorozat legnagyobb száma .....	74
$N$ valós szám szorzata .....	74–75
Keresés számsorozatban .....	75–76
Hanoi tornyok .....	76
QuickSort .....	76–77
MergeSort .....	77–78
<b>Kombinatorikus feladatok .....</b>	<b>79–81</b>
Permutációk .....	79
Variációk .....	80
Kombinációk .....	81
<b>Listák .....</b>	<b>81–88</b>
Lista tartalmának kiírása .....	82
Listákat kezelő eljárások és függvények .....	82
Lista $k$ -adik elemének kiírása .....	82–83
Lista bővítése rendezetten .....	83–84
Két rendezett lista összefésülése .....	85–87
Verem .....	87–88
<b>Szöveges állományok .....</b>	<b>88–93</b>
Állomány megnyitása .....	89
Állomány bezárása .....	89
Írás állományba .....	89–90
Olvasás állományból .....	90–91
Szöveges állományok feldolgozása .....	91–93
Hibakezelés .....	91
A sor végének az ellenőrzése .....	91–92
Az állomány végének az ellenőrzése .....	92–93
<b>Gráfok .....</b>	<b>93–104</b>
Irányított és nem irányított gráfok .....	93
Definíciók .....	93–95
Dijkstra-algoritmus .....	95–97
Floyd-algoritmus .....	97–99
Minimális értékű Hamilton-kör bejárása .....	99–100
Szélességi bejárás .....	100–101
Keresési fa .....	101

# Algoritmusok - pszeudókód

## Abszolút érték

Határozzuk meg és írjuk ki adott valós szám abszolút értékét!

**Algoritmus** Abszolút\_érték( $x, mod$ ):

**Ha**  $x \geq 0$  **akkor** { bemeneti adat:  $x$ , kimeneti adat:  $mod$  }

$mod \leftarrow x$

**különben**

$mod \leftarrow -x$

**vége(ha)**

**Vége(algoritmus)**

## Hányados ismételt kivonással

Számítsuk ki két természetes szám egész hányadosát ismételt kivonásokkal!

**Algoritmus** Osztas( $a, b, hányados$ ):

$hányados \leftarrow 0$  { bemeneti adatok:  $a, b$ , kimeneti adat:  $hányados$  }

**Amíg**  $a \geq b$  **végezd el:**

$hányados \leftarrow hányados + 1$

$a \leftarrow a - b$

**vége(amíg)**

**Vége(algoritmus)**

## Legnagyobb közös osztó

Számítsuk ki két természetes szám legnagyobb közös osztóját!

**Algoritmus** Eukleidész( $a, b, lko$ ):

**Ismételd** { bemeneti adatok:  $a, b$ , kimeneti adat:  $lko$  }

$r \leftarrow \text{maradék}[a/b]$  { kiszámítjuk az aktuális maradékot }

$a \leftarrow b$  { az osztandót felülírjuk az osztóval }

$b \leftarrow r$  { az osztót felülírjuk a maradékkal }

ameddig  $r = 0$  { amikor a maradék 0, véget ér az algoritmus }

$lko \leftarrow a$  { lko egyenlő az utolsó osztó értékével }

**Vége(algoritmus)**

## Páros számok szűrése

Számoljuk meg  $n$  beolvasott szám közül a páros számokat!

{ bemenet  $n$  és a számok, kimenet:  $db$ , a páros számok száma }

**Algoritmus** Páros( $n, db$ ):

$db \leftarrow 0$

**Minden**  $i=1, n$  **végezd el:**

**Be:** szám

**Ha** szám *páros* **akkor**

$db \leftarrow db + 1$

**vége(ha)**

**vége(minden)**

**Vége(algoritmus)**

## Palindrom számok

Döntsük el egy adott számról, hogy palindromszám-e vagy sem!

**Algoritmus** Palindrom(szám,válasz):

másolat  $\leftarrow$  szám {bemeneti adat: *szám*, kimeneti adat: *válasz*}

újszám  $\leftarrow$  0

**Amíg** szám > 0 **végezd el:**

számjegy  $\leftarrow$  maradék[szám/10]

újszám  $\leftarrow$  újszám\*10 + számjegy

szám  $\leftarrow$  [szám/10]

**vége(amíg)**

válasz  $\leftarrow$  újszám = másolat

{ha újszám = másolat, akkor válasz értéke igaz}

{ha újszám  $\neq$  másolat, akkor válasz értéke hamis}

**Vége(algoritmus)**

## Orosz szorzás

Legyen  $a, b \in \mathbb{N}^*$ . Számítsuk ki  $a$  és  $b$  szorzatát!

**Algoritmus** Orosz\_szorzás(a,b,p):

$x \leftarrow a$

{bemeneti adatok: a, b}

$y \leftarrow b$

{kimeneti adat: p}

$p \leftarrow 0$

**Amíg**  $x > 0$  **végezd el:**

{ $xy + p = ab$  (\*)}

**Ha**  $x$  páratlan **akkor**

$p \leftarrow p + y$

**vége(ha)**

$x \leftarrow [x/2]$

$y \leftarrow y + y$

**vége(amíg)**

**Vége(algoritmus)**

## Minimum keresés

Határozzuk meg egy  $n$  elemű sorozat minimumát!

**Algoritmus** Minimum( $n, a, \min$ ):

$\min \leftarrow a_1$

**Minden**  $i=2, n$  **végezd el:**

**Ha**  $a_i < \min$  **akkor**

$\min \leftarrow a_i$

**vége(ha)**

**vége(minden)**

**Vége(algoritmus)**

## Maximum keresés

Írjuk ki három, páronként különböző valós szám közül a legnagyobbat!

**Algoritmus** Maximum(a,b,c):

**Há**  $(a > b)$  és  $(a > c)$  **akkor**

{bemeneti adatok: a, b, c}

**Ki:** 'A legnagyobb: ', a

**vége(ha)**

**Ha**  $(b > c)$  **és**  $(b > a)$  **akkor**

**Ki:** 'A legnagyobb: ', b

**vége(ha)**

**Ha**  $(c > a)$  **és**  $(c > b)$  **akkor**

**Ki:** 'A legnagyobb: ', c

**vége(ha)**

**Vége(algoritmus)**

## Eukleidész algoritmusa

Határozzuk meg két adott természetes szám legnagyobb közös osztóját (*lnko*) és legkisebb közös többszörösét (*lkkt*) Eukleidész algoritmusával!

**Algoritmus** Eukleidész(*a*,*b*,*lnko*,*lkkt*):

$x \leftarrow a$  {bemeneti adatok: a, b, kimeneti adatok: lnko, lkkt}

$y \leftarrow b$

**Amíg**  $a \neq b$  **végezd el:**

**Ha**  $a > b$  **akkor**

$a \leftarrow a - b$

**különb**

$b \leftarrow b - a$

**vége(ha)**

**vége(amíg)**

$lnko \leftarrow a$

$lkkt \leftarrow [x*y/lnko]$

**Vége(algoritmus)**

## Prímszámok

Adva van egy nullától különböző természetes szám (*n*). Írjunk algoritmust, amely eldönti, hogy az adott szám prímszám-e vagy sem!

**Algoritmus** Prímek:

**Be:** határ {határ-nál kisebb számokat fogunk vizsgálni}

**Ki:** 2

$n \leftarrow 3$

**Amíg**  $n < \text{határ}$  **végezd el:**

$\text{prím} \leftarrow \text{igaz}$

$\text{osztó} \leftarrow 3$

$\text{négyzetgyök} \leftarrow [ \sqrt{n} ]$

**Amíg**  $(\text{osztó} \leq \text{négyzetgyök})$  **és**  $\text{prím}$  **végezd el:**

**Ha**  $\text{maradék}[n/\text{osztó}] = 0$  **akkor**

$\text{prím} \leftarrow \text{hamis}$

**különb**

$\text{osztó} \leftarrow \text{osztó} + 2$

**vége(ha)**

**vége(amíg)**

**Ha**  $\text{prím}$  **akkor**

**Ki:** n

**vége(ha)**

$n \leftarrow n + 2$

Rendeld meg a teljes, nyomtatott verziót innen:  
[www.erettsegi-puskak.ro](http://www.erettsegi-puskak.ro)

[www.erettsegi-puskak.ro](http://www.erettsegi-puskak.ro)

```

ReadLn(a.nev);
Write('Fizetes: ');
ReadLn(a.fizetes);
alkalmazottak[szam] := a;
Write('Szemelyi kod: ');
ReadLn(a.szkod);
End;
WriteLn('A vallalkozas alkalmazottjai: ');
WriteLn('  kod  fizetes  nev');
For i := 1 To szam Do
  With alkalmazottak[i] Do
    WriteLn(szkod:10,' ',fizetes:10,' ',nev);
  WriteLn('Az alkalmazottak szama: ',szam);
ReadLn;
End.

```

## Kereső algoritmusok

### Lineáris keresés

```

Program linearis_kereses;
Var
  v: array[1..50] of Integer;
  x: Integer;
  n, i: Byte;
  megvan: Boolean;
Begin
  WriteLn;
  Write('Az elemek szama: ');
  ReadLn(n);
  For i := 1 To n Do
    Begin
      Write('Az ',i,' elem: ');
      ReadLn(v[i]);
    End;
  Write('A keresett szam: ');
  ReadLn(x);
  i := 0;
  megvan := False;
  While (i <= n) and not megvan Do
    Begin
      If x = v[i] Then
        megvan := True;
        i := i + 1;
      End;
    If megvan Then
      WriteLn('A szam megtalalhato a sorozatban.')
    Else
      WriteLn('A szam nem talalhato meg a sorozatban.');
```

```

  ReadLn;
End.

```

### Bináris keresés

Olvassunk be egy rendezett számsorozatot, és keressünk meg benne egy beolvasott számot! Alkalmazzuk a bináris keresés módszerét!

```
Program binaris_kereses; { binker.pas }
Var
  v: array[1..50] of Integer;
  x: Integer;
  n, i, eleje, kozepe, vege: Byte;
  megvan: Boolean;
Begin
  WriteLn;
  Write('Az elemek szama: ');
  ReadLn(n);
  WriteLn('A rendezett sorozat elemei:');
  For i := 1 To n Do
  Begin
    Write('Az ',i,' elem: ');
    ReadLn(v[i]);
  End;
  Write('A keresett szam: ');
  ReadLn(x);
  eleje := 1;
  vege := n;
  megvan := False;
  While (eleje <= vege) and not megvan Do
  Begin
    kozepe := (eleje+vege) div 2;
    If v[kozepe] = x Then
      megvan := True
    Else
      If x < v[kozepe] Then
        vege := kozepe - 1
      Else
        eleje := kozepe + 1
  End;
  If megvan Then
    WriteLn(x, ' a ', kozepe, ' helyen található.')
  Else
    WriteLn('A keresett szam nincs a sorozatban.');
```

ReadLn;

End.

## Rendező algoritmusok

### Buborékrendezés

Olvassunk be egy természetes számokból álló sorozatot, majd a buborékrendezés segítségével írjuk ki őket növekvő sorrendben!

```
Program buborek_rendezes;
Var
  v: array[1..50] of Word;
```

```

seged: Word;
n, i, j: Byte;
rendezett: Boolean;
Begin
  Write('Az elemek szama: ');
  ReadLn(n);
  For i := 1 To n Do
    Begin
      Write('v[' ,i,'] = ');
      ReadLn(v[i]);
    End;
  i := n - 1;
  rendezett := False;
  While (i >= 1) and not rendezett Do
    Begin
      rendezett := True;
      For j := 1 To i Do
        If v[j] > v[j+1] Then
          Begin
            seged := v[j];
            v[j] := v[j+1];
            v[j+1] := seged;
            rendezett := False;
          End;
        i := i - 1;
      End;
      WriteLn('A rendezett szamsorozat:');
      For i := 1 To n Do
        Write(v[i], ' ');
      End.

```

### Minimumkiválasztásos rendezés

Felhasználva a minimumkiválasztásos rendezés módszerét, rendezzünk egy egész számokból álló sorozatot!

```

Program minimumkivalasztasos_rendezes;
Var
  v: array[1..50] of Integer;
  seged: Integer;
  n, i, j, minindex: Byte;
Begin
  WriteLn;
  Write('Az elemek szama: ');
  ReadLn(n);
  WriteLn('A szamsorozat elemei:');
  For i := 1 To n Do
    Begin
      Write('v[' ,i,'] = ');
      ReadLn(v[i]);
    End;
  For i := 1 To n - 1 Do
    Begin
      minindex := i;
      For j := i + 1 To n Do
        If v[j] < v[minindex] Then

```

```

    minindex := j;
  If i <> minindex Then
  Begin
    seged := v[i];
    v[i] := v[minindex];
    v[minindex] := seged;
  End;
End;
WriteLn('A rendezett szamsorozat:');
For i := 1 To n Do
  Write(v[i], ' ');
End.

```

### Beszúrásos rendezés

Olvassunk be egy egész számokból álló szamsorozatot, majd írjuk ki ezt a sorozatot növekvő sorrendben, felhasználva a beszúrásos alapuló rendezés módszerét!

```

Program beszurasos_rendezes;
Var
  v: array[1..50] of Integer;
  ment: Integer;
  n, i, j: Byte;
Begin
  Writeln;
  Write('Az elemek szama: ');
  ReadLn(n);
  For i := 1 To n Do
  Begin
    Write('v['i,'] = ');
    ReadLn(v[i]);
  End;
  For i := 2 To n Do
  If v[i] < v[i-1] Then
  Begin
    ment := v[i];
    j := i;
    Repeat
      j := j - 1;
      v[j+1] := v[j];
    Until (j = 1) or (v[j-1] <= ment);
    v[j] := ment;
  End;
  Writeln('A rendezett szamsorozat:');
  For i := 1 To n Do
    Write(v[i], ' ');
  Writeln;
End.

```

Rendeld meg a teljes, nyomtatott verziót innen:  
[www.erettsegi-puskak.ro](http://www.erettsegi-puskak.ro)

[www.erettsegi-puskak.ro](http://www.erettsegi-puskak.ro)

# „Oszd meg és uralkodj” „Divide Et Impera”

## Sorozat legnagyobb száma

```
Program maximumszamias_oszd_meg_es_uralkodjal;
```

```
Var
```

```
  x: array[1..20] of Integer;
```

```
  i, n: Byte;
```

```
Function maximum(bal, jobb: Byte): Integer;
```

```
Var
```

```
  max1, max2: Integer;
```

```
  kozepe: Byte;
```

```
Begin
```

```
  If bal = jobb Then
```

```
    maximum := x[bal]
```

```
  Else
```

```
    Begin
```

```
      kozepe := (bal + jobb) div 2;
```

```
      max1 := maximum(bal, kozepe);
```

```
      max2 := maximum(kozepe + 1, jobb);
```

```
      If max1 < max2 Then
```

```
        maximum := max2
```

```
      Else
```

```
        maximum := max1;
```

```
      End;
```

```
    End;
```

```
Begin
```

```
  Write('n = ');
```

```
  ReadLn(n);
```

```
  Write('A számok: ');
```

```
  For i := 1 To n Do
```

```
    Read(x[i]);
```

```
  ReadLn;
```

```
  WriteLn('Maximum: ', maximum(1,n));
```

```
  ReadLn;
```

```
End.
```

## N valós szám szorzata

Számítsuk ki n valós szám szorzatát oszd meg és uralkodj módszerrel!

```
Program szorzas_oszd_meg_es_uralkodjal;
```

```
Var
```

```
  x: array[1..20] of Integer;
```

```
  i, n: Byte;
```

```
Function szorzas(bal, jobb: Byte): Integer;
```

```
Var
```

```
  sz1, sz2: Integer;
```

```
  kozepe: Byte;
```

```

Begin
  If bal = jobb Then
    szorzas := x[bal]
  Else
    Begin
      kozepe := (bal + jobb) div 2;
      sz1 := szorzas(bal, kozepe);
      sz2 := szorzas(kozepe + 1, jobb);
      szorzas := sz1 * sz2;
    End;
  End;
End;

Begin
  Write('n = ');
  ReadLn(n);
  Write('A szamok: ');
  For i := 1 To n Do
    Read(x[i]);
  ReadLn;
  WriteLn('A szamok szorzata: ', szorzas(1,n));
  ReadLn;
End.

```

## Keresés számsorozatban

Olvassunk be egy rendezett számsorozatot, és keressünk meg benne egy beolvasott számot! Ha megtaláltuk írjuk ki a pozícióját. Alkalmazzuk a bináris keresés módszerét!

```

Program binaris_kereses;
Var
  x: array[1..20] of Integer;
  ker: Integer;
  i, n: Byte;

Procedure bin_kereses(bal, jobb: Byte);
Var
  kozepe: Byte;
Begin
  If bal > jobb Then
    Begin
      WriteLn(ker, ' nincs az adott sorozatban');
      Exit;
    End;
  kozepe := (bal + jobb) div 2;
  If ker = x[kozepe] Then
    WriteLn(ker, ' a(z) ', kozepe, 'h. talalhato')
  Else
    If ker < x[kozepe] Then
      bin_kereses(bal, kozepe - 1)
    Else
      bin_kereses(kozepe + 1, jobb);
  End;
End;

Begin

```

```

Write('n = ');
ReadLn(n);
Write('A novekvő sorozat: ');
For i:=1 To n Do
  Read(x[i]);
Write('Keresett szám: ');
ReadLn(ker);
bin_kereses(1,n);
ReadLn;
End.

```

### Hanoi tornyok

Adva van három rúd: A, B és C. Az A rúdon induláskor n különböző átmérőjű lyukas korong található, az átmérők szerint csökkenő sorrendben. Írjuk ki az összes lehetséges módját annak, ahogyan a korongokat át lehet helyezni az A rúdról a B-re, ugyanolyan sorrendben, ahogyan az A-n helyezkedtek el! Az áthelyezés közben fel lehet használni a C rudat. Egy mozzgatás alkalmával csak egy korongot lehet áthelyezni, és csak kisebb átmérőjű korongot helyezhetünk egy nagyobb átmérőjű korongra.

```

Program hanoi_tornyok;
Var
  n: Byte;

Procedure hanoi(n: Byte; a, b, c: Char);
Begin
  If n = 1 Then
    WriteLn(a, '-', b)
  Else
    Begin
      hanoi(n - 1, a, c, b);
      WriteLn(a, '-', b);
      hanoi(n - 1, c, b, a);
    End;
End;

Begin
  Write('n = '); ReadLn(n);
  WriteLn('A mozzgatások: ');
  hanoi(n, 'A', 'B', 'C');
  ReadLn;
End.

```

### QuickSort

Rendezzünk növekvő sorrendbe n egész számot, a QuickSort algoritmussal!

```

Program gyorsrendezés;
Var
  x: array[1..20] of Integer;
  i, n: Byte;

Procedure feloszt(bal, jobb:Byte; var m:Byte);

```

```

Var
  seged: Integer;
  i, j, elvalaszto: Byte;
Begin
  elvalaszto := x[bal];
  i := bal - 1;
  j := jobb + 1;
  Repeat
    While x[j] > elvalaszto Do
      Dec(j);
    While x[i] < elvalaszto Do
      Inc(i);
    If i < j Then
      Begin
        seged := x[i];
        x[i] := x[j];
        x[j] := seged;
      End;
  Until i >= j;
  m := j;
End;

Procedure QuickSort(bal, jobb:Byte);
Var
  m: Byte;
Begin
  If bal < jobb Then
    Begin
      feloszt(bal, jobb, m);
      QuickSort(bal, m);
      QuickSort(m + 1, jobb);
    End;
End;

Begin
  Write('n = ');
  ReadLn(n);
  Write('A sorozat: ');
  For i := 1 To n Do
    Read(x[i]);
  QuickSort(1, n);
  WriteLn('A rendezett sorozat:');
  For i := 1 To n Do
    Write(x[i], ' ');
  ReadLn;
End.

```

### MergeSort

Rendezzünk növekvő sorrendbe  $n$  egész számot, a MergeSort algoritmussal!

```

Program osszefesuleses_rendezes;
Var
  x: array[1..20] of Integer;

```

Rendeld meg a teljes, nyomtatott verziót innen:  
[www.erettsegi-puskak.ro](http://www.erettsegi-puskak.ro)

[www.erettsegi-puskak.ro](http://www.erettsegi-puskak.ro)