

# INFORMATIKA

C NYELV



A 2016-OS ÉRETTSÉGI KÖVETELMÉNYEINEK  
MEGFELELŐ OKTATÁSI SEGÉDANYAG

A PUSKÁZÁS CSALÁSNAK  
MINŐSÜL.  
A PUSKÁK VIZSGÁN  
TÖRTÉNŐ HASZNÁLATÁT  
NEM AJÁNLJUK!

## Tartalomjegyzék

<b>Algoritmusok - pszeudókód</b> .....	<b>1–42</b>
Abszolút érték .....	1
Hányados ismételt kivonással .....	1
Legnagyobb közös osztó .....	1–2
Páros számok szűrése .....	2
Palindrom számok .....	2–3
Orosz szorzás .....	3
Minimum keresés .....	3
Maximum keresés .....	4
Eukleidész algoritmus .....	4
Prímszámok .....	5
Fibonacci-számok .....	5–6
Háromszög .....	6–7
Fordított szám .....	7–8
Törzstényezők .....	8
Prímszámvizsgálat .....	9–10
Konverzió – Számrendszer átalakítás .....	10–11
Gyors hatványozás .....	11–12
Szekvenciális (lineáris) keresés .....	12
Megszámlálás .....	12
Minimum- és maximumkiválasztás .....	13
A Maximum helye .....	13
Kiválogatás .....	13
Szétválogatás .....	14
Sorozat halmazá alakítása .....	14–15
Sorozatok keresztmetszete .....	15–16
Sorozatok egyesítése .....	16
Sorozatok összefésülése .....	17
Párok sorszáma egy sorozatban .....	18
Arány .....	18
Teljes négyzet .....	18–19
Osztályátlagok szétválasztása .....	19–20
Büvös négyzet .....	20–21
Polinom értéke adott pontban .....	21
Polinomok összege .....	22

Polinomok szorzata .....	22
Buborékrendezés (Bubble-sort) .....	23–24
Egyszerű felcseréléses rendezés .....	24
Válogatásos rendezés .....	24
Minimum/maximum kiválasztásra épülő rendezés .....	25
Beszűrő rendezés .....	25
Leszámláló rendezés .....	25–26
Összefésülésen alapuló rendezés .....	26–27
Gyorsrendezés (QuickSort) .....	27–28
Szavak sorrendjének megfordítása .....	28
Faktoriális .....	28
Számjegyösszeg .....	29
k elemű részhalmazok .....	29
Konverzió .....	30
Az $\{1, 2, \dots, n\}$ halmaz minden részhalmaza .....	30
Kamatos kamatok kifrása .....	30–31
Általános backtracking .....	31–32
Általános rekurzív backtracking .....	32
Elhelyezni 8 királynőt a sakkasztalán .....	32–33
Zárójelek .....	34
Játékok dobozva való elhelyezésének kifrása .....	34–35
X pénzösszeg kifizetése n bankjegy segítségével .....	35–36
X Összeg kifizetése, minimum számú bankjeggyel .....	36–37
Általános Divide Et Impera .....	37
Szorzat (DivImp) .....	38
Minimumszámolás (DivImp) .....	38–39
Hatványozás (DivImp) .....	39
Bináris keresés (DivImp) .....	40
Általános mohó (Greedy) algoritmus .....	40
Összeg (Greedy) .....	41
Hátizsák probléma (Greedy) .....	41–42
Összegkifizetés legkevesebb számú bankjeggyel (Greedy) .....	42
<b>A C nyelv elemei .....</b>	<b>43–53</b>
Azonosítók .....	43
A programok felépítése .....	44
Az alapértelmezett egyszerű típusok .....	45–47

Változók .....	47–49
Konstansok (állandók) .....	49–50
Adatok beolvasása és kiírása .....	50–53
<b>Egyszerű programok készítése .....</b>	<b>54–56</b>
Téglalap területe és kerülete .....	55
Inkrementáló és dekrementáló operátorok .....	55–56
Bitműveletek .....	56
<b>A C nyelv utasításai .....</b>	<b>57–74</b>
A kifejezés utasítás .....	57
Az összetett utasítás .....	57
Feltételes utasítások .....	57–60
Az if utasítás .....	58
Valós szám abszolút értéke .....	58
Páros – páratlan számok megállapítása .....	59
Nagyobbik szám kiválasztása .....	59–60
Nagybetű, kisbetű vagy szám .....	60
A switch utasítás .....	61–65
Aritmetikai műveletek .....	62–63
Az év hányadik napja .....	63–65
Ciklus utasítások .....	65–74
Az előtesztelő ciklus .....	65–66
A hátulatesztelő ciklus .....	66
A számlálós ciklus .....	66–67
Szám számjegyeinek száma .....	67–68
Törzstényezőkre való bontás .....	68
Legnagyobb közös osztó (Eukleidész algoritmus) .....	69
P számrendszerből q számrendszerbe .....	69–71
Faktoriális .....	71–72
Prímszám vizsgálat .....	72–73
Fibonacci sorozat n-dik eleme .....	73
Polinom értéke egy x pontban (Horner-séma) .....	73–74
<b>A tömbök .....</b>	<b>74–92</b>
A tömb fogalma .....	74
Egydimenziós tömbök .....	75–84
Számsorozat fordított sorrendben .....	75–76

Két polinom szorzata .....	76–77
Kezdőértékkel rendelkező egydimenziós tömbök .....	77
Többdimenziós tömbök .....	78
Tömb szimmetriája .....	78–79
Mátrix szorzata vektorral .....	80–81
Két matrix szorzata .....	81–82
Mátrix inverze .....	83–84
Karaktertömbök .....	85–86
Karakterláncok beolvasása a standard bemenetről .....	86
Karakterláncok kiírása a standard kimenetre .....	86
Karakterlánc konverziós műveletek .....	87–90
Numerikus értékek karakterláncná alakítása .....	87
Karakterlánc numerikus értékekké alakítása .....	87
Más konvertáló függvények .....	87
Karakterlánc kezelő függvények .....	87–89
Betűk frekvenciája egy sorban .....	89–90
Dinamikus tömbök .....	90–92
Vektor páros elemeinek összege .....	90–91
Mátrix páratlan elemeinek összege .....	91–92
<b>Az előfeldolgozó parancsok .....</b>	<b>92–96</b>
Állományok beillesztése .....	92–93
Makrók .....	93–94
Feltételes fordítás makró létezésétől függően .....	94
Feltételes fordítás makró nem-létezésétől függően .....	95
Feltételes fordítás .....	95–96
Fordítási hibaüzenet generálása .....	96
<b>Függvények .....</b>	<b>96–103</b>
A függvény fogalma .....	96–97
Függvény deklarációja .....	97
Függvény definíciója .....	97–98
Paraméterátadás .....	98–99
Tömb paraméterek .....	99–100
Faktoriális .....	100–101
Goldbach-feltétel .....	101–102
Legnagyobb közös osztó .....	103

<b>Bonyolultabb adatszerkezetek</b> .....	<b>104–109</b>
Struktúrák.....	104–105
Kezdőértékkel rendelkező struktúrák.....	106
Tanuló adatai.....	106–108
Saját típusok definiálása.....	108–109
<b>Állományok</b> .....	<b>109–118</b>
Állomány megnyitása.....	110
Állomány bezárása.....	111
Írás állományba.....	111–112
Karakter írása állományba.....	111
Karakterlánc írása állományba.....	111–112
Formázott írás állományba.....	112
Olvasás állományból.....	112–118
Karakter olvasása állományól.....	112
Karakterlánc olvasása állományból.....	113
Formázott olvasás állományból.....	113
Állomány végének az ellenőrzése.....	113–114
Pozicionálás az állományban.....	114
Szöveges állomány feldolgozása (feladat1).....	115–116
Szöveges állomány feldolgozása (feladat2).....	116–117
Szöveges állomány feldolgozása (feladat3).....	117–118
<b>Klasszikus algoritmusok</b> .....	<b>119–128</b>
Kereső algoritmusok.....	119–121
Lineáris keresés.....	119
Bináris keresés.....	120–121
Rendező algoritmusok.....	121–124
Buborékrendezés.....	121–122
Minimumkiválasztásos rendezés.....	122–123
Beszúrásos rendezés.....	123–124
Összefésztések.....	124–128
Összefésztés strázsa nélkül.....	124–126
Összefésztés strázssával (ütközővel).....	127–128
<b>Rekurzió</b> .....	<b>129–142</b>
Közvetlen rekurzió.....	129–137

Szó betűi fordított sorrendben.....	129–130
Faktoriális .....	130
Fibonacci sorozat n-edik eleme.....	131
Legnagyobb közös osztó.....	131–132
Ackermann függvény.....	132–133
$X^n$ -diken kiszámítása .....	133–134
Tíz-es számrendszerből való átírás p számrendszerbe...	134–135
Az első n páratlan természetes szám összege.....	135
Természetes szám számjegyeinek összege.....	135–136
N természetes szám összege .....	136–137
Tömbök rekurzívan .....	137–140
Számsorozat negatív elemeinek száma .....	137–138
Szám előfordulása egy tömbben .....	138–139
Páros elemek összege.....	139–140
Rekurzív szerkezetű eredményt igénylő feladatok .....	140–142
N hosszúságú megadott karakterlánc .....	140–141
Természetes szám partíciói .....	141–142
<b>Az „Oszd meg és uralkodj” módszer .....</b>	<b>143–151</b>
Sorozat legnagyobb elem.....	143–144
N szám szorzata .....	144–145
Bináris keresés .....	146–147
Hanoi tornyok .....	147–148
QuickSort.....	148–150
MergeSort .....	150–151
<b>Kombinatorikai feladatok.....</b>	<b>152–155</b>
Permutációk.....	152–153
Variációk.....	153–154
Kombinációk .....	154–155
<b>A dinamikus adatszerkezetek .....</b>	<b>156–161</b>
Szimplán láncolt lista .....	156–161
Lista létrehozása .....	157
Lista elemeinek a kifrása.....	157
Egy új elem beszúrása a listába.....	158
Egy elem törlése a listából .....	158
Megoldott listás feladat.....	158–161

# 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:  $mod$ }

$mod \leftarrow x$

**különb**

$mod \leftarrow -x$

**vége(ha)**

**Vége(algoritmus)**

## Hányados ismételt kivonással

Számítsuk ki két természetes szám egész hányadosát ismételt kivonásokkal!

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

$hányados \leftarrow 0$  {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( $a, b, loko$ ):

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

$r \leftarrow \text{maradék}[a/b]$  {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 }  
 $l\text{inko} \leftarrow a$  { línko egyenlő az utolsó osztó értékével }  
**Vége(algoritmus)**

## Páros számok szűrése

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

{ bemeneti adat:  $n$  és a számok, kimeneti adat:  $db$ , 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:**

$számjegy \leftarrow \text{maradék}[szám/10]$

$újszám \leftarrow újszám * 10 + számjegy$

$szám \leftarrow [szám/10]$

**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)**

$x \leftarrow \lfloor x/2 \rfloor$

$y \leftarrow y + y$

**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)**

# A C nyelv utasításai

## A kifejezés utasítás

A kifejezés utasítás végrehajtása a kifejezésnek a megfelelő szabályok szerinti kiértékelését jelenti. Mielőtt a következő utasításra adódik a vezérlés, a teljes kiértékelés (a mellékhatásokkal együtt) végbemegy.

Példák:

```
kerulet = x + y + b;           //értékadás
j++;                          //j növelése 1-gyel
a = b = c = 3;                //többszörös értékadás
z = cos (alfa) + 1.35         //függvényt hívó kifejezés
```

## Az összetett utasítás

Programjainkban nagyon gyakran használjuk az összetett utasítást. Az összetett utasítás a (és az) között megadott utasítások sorozatából áll. Ezt az utasítást olyan helyen használjuk, ahol egynél több utasítás végrehajtására van szükség, de a szintaxis csak egy utasítás használatát engedélyezi. Az összetett utasítás általános szintaxisa:

```
{
lokális definíciók és deklarációk;
utasítás1;
utasítás2;
...
utasításn;
}
```

## Feltételes utasítások

A programok végrehajtása során sokszor szükségünk van arra, hogy bizonyos feltételektől függően a számítógép a

program különböző részeit (ágait) hajtsa végre. Tehát ha a feltétel teljesül, a program egy bizonyos műveletsort végez, különben egy másikat. Ezt a feltételes vagy döntéshozó utasításokkal (válogatással, szelekcióval) valósítjuk meg.

### **Az if utasítás**

A legegyszerűbb feltételes utasítás az if, amely a kétágú döntésnek felel meg. Az utasítás általános formája:

```
if (kifejezés)
    utasítás1;
else
    utasítás2;
```

Az if utasítás végrehajtása a kifejezés kiértékelésével kezdődik. A kifejezés általában egy logikai kifejezés, amelynek ha értéke 1 vagy ennél nagyobb egész szám, az utasítás1 hajtódik végre, különben (ha a kifejezés értéke 0) az utasítás2 kerül végrehajtásra.

### **Valós szám abszolút értéke**

```
#include <stdio.h>
#include <conio.h>

main()
{
    int x, abs;
    printf("\n Kerem a szamot:");
    scanf("%d", &x);
    if (x >= 0)
        abs = x;
    else
        abs = -x;
    printf("A szam abszolut erteke: %d",abs);
    getch();
}
```

# Rekurzió

## Közvetlen rekurzió

A közvetlen rekurzió a leggyakoribb. Ebben az esetben az adott függvény blokkjában explicit módon meghívjuk az illető függvényt. Egy rekurzív függvény végrehajtása azonos módon történik, mint bármely nem rekurzív függvényé.

Egy rekurzív függvény a következőképpen hajtódik végre:

- a függvény első aktiválása során végrehajtnak az utasítások a *folytatási / leállási feltétel*ig;
- kiértékelődik a feltétel és amíg ennek a logikai értéke azt jelenti, hogy a függvénynek meg kell hívja önmagát, akkor a függvény aktiválásainak sora következik, ami azt jelenti, hogy végre lesz hajtva a függvénynek a feltételig tartó része, valamint az újrahívás;
- amikor a feltétel azt jelenti, hogy a függvénynek nem kell többé önmagát meghívja, akkor, ismételten, a hívásokhoz viszonyítva fordított sorrendben, végrehajtnak a függvény hívása / önmeghívása utáni rész.

### **Szó betűi fordított sorrendben**

Írjuk ki egy szó betűit fordított sorrendben! A szó végét egy szóközzel jelöljük. Ne használjunk a megoldásban tömböt és karakterláncot!

```
#include <stdio.h>
#include <conio.h>

void fordit()
{
    char betu;
    scanf("%c", &betu);
    if (betu != ' ')
        fordit();
}
```

```

printf("%c", betu);
}

void main()
{
printf("A szo: ");
fordit();
getch();
return;
}

```

### Faktoriális

Írjunk rekurzív függvényt, amely kiszámítja az  $n!$  -t!

```

#include <stdio.h>
#include <conio.h>

long int fakt(int n)
{
if (n == 0)
return 1;
else
return n * fakt(n - 1);
}

void main()
{
int n;
printf("n = ");
scanf("%d", &n);
printf("n != %ld", fakt(n));
getch();
return;
}

```

# Az „Oszd meg és uralkodj” módszer

Ezt a módszert sikeresen lehet alkalmazni az informatikában. A módszer lényege az, hogy a feladatot *egymástól független* részfeladatokra bontjuk, amelyeket az eredeti feladathoz hasonlóan oldunk meg, de kisebb méretű adatok esetében.

A módszer lépéseit a következőképpen foglalhatjuk össze:

- A feladatot kettő vagy több, hasonló egymástól független jellegű részfeladatra bontjuk. A részfeladatok lehetnek *elemi* vagy *nem elemi* részfeladatok.
- Az elemi részfeladatokat megoldjuk, a nem elemieket pedig újabb elemi, vagy nem elemi részfeladatokra bontjuk. Ezt a műveletet addig folytatjuk, ameddig elemi részfeladatokhoz jutunk.
- Az eredményt úgy kapjuk, hogy az egyes részfeladatokat a felosztás fordított sorrendjében összerakjuk, összekombináljuk.

## Sorozat legnagyobb elem

Határozzuk meg  $n$  egész szám közül a legnagyobbat az oszd meg és uralkodj módszerrel!

```
#include <stdio.h>
#include <conio.h>

int x[100], n, i;

int maximum(int bal, int jobb)
{
    int max1, max2, kozepe;
    if (bal == jobb)
        return x[bal];
    else
        if (jobb - bal == 1)
            if (x[bal] < x[jobb])
                return x[jobb];
```

```

    else
        return x[bal];
else
{
    kozepe = (bal + jobb) / 2;
    max1 = maximum(bal, kozepe);
    max2 = maximum(kozepe + 1, jobb);
    if (max1 < max2)
        return max2;
    else
        return max1;
}
}

void main()
{
    printf("n = ");
    scanf("%d", &n);
    printf("Adjuk meg a szamokat!\n");
    for (i = 0; i < n; i++)
    {
        printf("Az %d .elem: ", i);
        scanf("%d", &x[i]);
    }
    printf("A legnagyobb szam : %d", maximum(0, n - 1));
    getch();
    return;
}

```

### N szám szorzata

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

```

#include <stdio.h>
#include <conio.h>

int x[100], n, i;

```



```

int szorzas(int bal, int jobb)
{
    int sz1, sz2, kozepe;
    if (bal == jobb)
        return x[bal];
    else
        if (jobb - bal == 1)
            return x[bal] * x[jobb];
        else
            {
                kozepe = (bal + jobb) / 2;
                sz1 = szorzas(bal, kozepe);
                sz2 = szorzas(kozepe + 1, jobb);
                return sz1 * sz2;
            }
}

void main()
{
    printf("n = ");
    scanf("%d", &n);
    printf("Adjuk meg a szamokat!\n");
    for (i = 0; i < n; i++)
    {
        printf("Az %d .elem: ", i);
        scanf("%d", &x[i]);
    }
    printf("A szamok szorzata : %d", szorzas(0, n - 1));
    getch();
    return;
}

```