

# INFORMATIKA

C NYELV



A 2016-OS ÉRETTSÉGI KÖVETELMÉNYEINEK  
MEGFELELŐ OKTATÁSI SEGÉDANYAG

© 2016 PRESSTERN SOLUTIONS

# Tartalomjegyzék

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

Egyszerű felcseréléses rendezés.....	21
Válogatásos rendezés .....	21
Minimum/maximum kiválasztásra épülő rendezés.....	22
Beszűrő rendezés.....	22
Leszámláló rendezés .....	22
Összefésülésen alapuló rendezés .....	23
Gyorsrendezés (QuickSort).....	24
Szavak sorrendjének megfordítása .....	24
Faktoriális.....	25
Számjegyösszeg .....	25
k elemű részhalmazok.....	25
Konverzió.....	26
Az $\{1, 2, \dots, n\}$ halmaz minden részhalmaza.....	26
Kamatos kamatok kiírása.....	27
Általános backtracking.....	27
Általános rekurzív backtracking .....	28
Elhelyezni 8 királynőt a sakktablán.....	28
Zárójelek .....	29
Játékok dobozva való elhelyezésének kiírása .....	30
X pénzösszeg kifizetése n bankjegy segítségével .....	31
X Összeg kifizetése, minimum számú bankjeggyel .....	32
Általános Divide Et Impera.....	32
Szorzat (DivImp).....	33
Minimumszámolás (DivImp).....	33
Hatványozás (DivImp).....	34
Bináris keresés (DivImp) .....	34
Általános mohó (Greedy) algoritmus .....	35
Összeg (Greedy).....	35
Hátizsák probléma (Greedy).....	36
Összegkifizetés legkevesebb számú bankjeggyel (Greedy) .....	36

## A C nyelv elemei..... 38

Azonosítók.....	38
A programok felépítése .....	39
Az alapértelmezett egyszerű típusok .....	39
Egész jellegű típusok .....	40
Valós típus .....	40
A void típusnév.....	41
Változók.....	41
Változók deklarálása .....	42
Egész jellegű változók deklarálása .....	42

Valós típusú változók deklarálása.....	42
Konstansok (állandók).....	43
Egész konstansok.....	43
Karakter konstansok.....	43
Karakterlánc konstansok.....	43
Adatok beolvasása és kiírása.....	44
Karakter beolvasása a standard bemenetről.....	44
Karakter kiírása a standard kimenetre.....	44
Egy karakter beolvasása közvetlenül.....	45
Formázott kiírás a standard kimenetre.....	45
Formázott beolvasás a standard bemenetről.....	46
<b>Egyszerű programok készítése.....</b>	<b>47</b>
Téglalap területe és kerülete.....	47
Inkrementáló és dekrementáló operátorok.....	48
Bitműveletek.....	49
<b>A C nyelv utasításai.....</b>	<b>50</b>
A kifejezés utasítás.....	50
Az összetett utasítás.....	50
Feltételes utasítások.....	50
Az if utasítás.....	51
Valós szám abszolút értéke.....	51
Páros – páratlan számok megállapítása.....	51
Nagyobbik szám kiválasztása.....	52
Nagybetű, kisbetű vagy szám.....	53
A switch utasítás.....	53
Aritmetikai műveletek.....	54
Az év hányadik napja.....	55
Ciklus utasítások.....	57
Az előtesztelő ciklus.....	57
A hátulatesztelő ciklus.....	57
A számlálós ciklus.....	58
Szám számjegyeinek száma.....	58
Törzstényezőkre való bontás.....	59
Legnagyobb közös osztó (Eukleidész algoritmus).....	60
P számrendszerből q számrendszerbe.....	61
Faktoriális.....	62
Prímszám vizsgálat.....	63
Fibonacci sorozat n-dik eleme.....	63
Polinom értéke egy x pontban (Horner-séma).....	64

<b>A tömbök.....</b>	<b>66</b>
A tömb fogalma .....	66
Egydimenziós tömbök.....	66
Számsorozat fordított sorrendben.....	67
Két polinom szorzata.....	67
Kezdőértékkel rendelkező egydimenziós tömbök .....	68
Többdimenziós tömbök .....	68
Tömb szimmetriája .....	69
Mátrix szorzata vektorral .....	70
Két matrix szorzata .....	71
Mátrix inverze .....	72
Karaktertömbök .....	74
Karakterláncok beolvasása a standard bemenetről .....	75
Karakterláncok kiírása a standard kimenetre .....	76
Karakterlánc konverziós műveletek .....	76
Numerikus értékek karakterláncra alakítása .....	76
Karakterlánc numerikus értékké alakítása .....	76
Más konvertáló függvények .....	76
Karakterlánc kezelő függvények .....	77
Betűk frekvenciája egy sorban .....	78
Dinamikus tömbök.....	79
Vektor páros elemeinek összege .....	79
Mátrix páratlan elemeinek összege .....	80
<b>Az előfeldolgozó parancsok .....</b>	<b>81</b>
Állományok beillesztése .....	81
Makrók.....	81
Feltételes fordítás makró létezésétől függően.....	82
Feltételes fordítás makró nem-létezésétől függően .....	82
Feltételes fordítás.....	83
Fordítási hibáüzenet generálása .....	83
<b>Függvények .....</b>	<b>84</b>
A függvény fogalma .....	84
Függvény deklarációja.....	84
Függvény definíciója .....	85
Paraméterátadás .....	85
Tömb paraméterek.....	86
Faktoriális.....	87
Goldbach-feltétel.....	87

Legnagyobb közös osztó.....	89
<b>Bonyolultabb adatszerkezetek.....</b>	<b>90</b>
Struktúrák .....	90
Kezdőértékkel rendelkező struktúrák .....	91
Tanuló adatai .....	91
Saját típusok definiálása .....	93
<b>Állományok .....</b>	<b>95</b>
Állomány megnyitása.....	95
Állomány bezárása .....	96
Írás állományba.....	96
Karakter írása állományba.....	96
Karakterlánc írása állományba .....	96
Formázott írás állományba.....	97
Olvasás állományból .....	97
Karakter olvasása állományól.....	97
Karakterlánc olvasása állományból.....	97
Formázott olvasás állományból.....	98
Állomány végének az ellenőrzése .....	98
Pozicionálás az állományban.....	98
Szöveges állomány feldolgozása (feladat1).....	99
Szöveges állomány feldolgozása (feladat2) .....	100
Szöveges állomány feldolgozása (feladat3) .....	101
<b>Klasszikus algoritmusok .....</b>	<b>103</b>
Kereső algoritmusok .....	103
Lineáris keresés.....	103
Bináris keresés .....	104
Rendező algoritmusok .....	105
Buborékrendezés.....	105
Minimumkiválasztásos rendezés.....	106
Beszúrásos rendezés .....	106
Összefűsölések.....	107
Összefűsülés strázsa (ütköző) nélkül .....	107
Összefűsülés strázásával (ütközővel) .....	110
<b>Rekurzió .....</b>	<b>112</b>
Közvetlen rekurzió .....	112
Szó betűi fordított sorrendben.....	112

Faktoriális.....	113
Fibonacci sorozat n-edik eleme .....	113
Legnagyobb közös osztó.....	114
Ackermann függvény.....	115
$X^n$ -diken kiszámítása.....	115
Tizes számrendszerből való átírás p számrendszerbe.....	116
Az első n páratlan természetes szám összege .....	117
Természetes szám számjegyeinek összege .....	117
N természetes szám összege.....	118
Tömbök rekurzívan .....	119
Számsorozat negatív elemeinek száma.....	119
Szám előfordulása egy tömbben .....	120
Páros elemek összege .....	121
Rekurzív szerkezetű eredményt igénylő feladatok .....	122
N hosszúságú megadott karakterlánc.....	122
Természetes szám partíciói.....	123
<b>Az „Oszd meg és uralkodj” módszer.....</b>	<b>124</b>
Sorozat legnagyobb elem .....	124
N szám szorzata.....	125
Bináris keresés.....	126
Hanoi tornyok .....	127
QuickSort.....	128
MergeSort.....	130
<b>Kombinatorikai feladatok.....</b>	<b>132</b>
Permutációk .....	132
Variációk.....	133
Kombinációk .....	134
<b>A dinamikus adatszerkezetek .....</b>	<b>136</b>
Szimplán láncolt lista.....	136
Lista létrehozása.....	137
Lista elemeinek a kiírása.....	137
Egy új elem beszúrása a listába.....	137
Egy elem törlése a listából.....	138
Megoldott listás feladat.....	138

# 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** 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( $a, b, loko$ ):

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

$r \leftarrow maradék[a/b]$  {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}

$loko \leftarrow a$  {loko 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  $\leftarrow$  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  $a, b \in \mathbf{N}^*$ . 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)**

## Maximum keresés

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

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

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

$\{\text{bemeneti adatok: } a, b, c\}$

**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)**

# 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 számot:");
    scanf("%d", &x);
    if (x >= 0)
        abs = x;
    else
        abs = -x;
    printf("A szám abszolút értéke: %d",abs);
    getch();
}
```

## Páros – páratlan számok megállapítása

Állapítsuk meg egy számról, hogy páros vagy páratlan!

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

main()
{
    int szam;
```

```

printf("\nKerek egy egesz szamot:");
scanf("%d", &szam);
if (szam % 2 != 0)
    printf("\nParatlan!");
else
    printf("\nParos!");
getch();
}

```

## Nagyobbik szám kiválasztása

Olvassunk be három egész számot, majd megfelelő szöveg kíséretében írjuk ki a nagyobbikat!

```

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

main()
{
    double szam1, szam2, szam3, max;
    printf("\n Az elso szam: "); scanf("%lf",&szam1);
    printf("\n A masodik szam: "); scanf("%lf",&szam2);
    printf("\n A harmadik szam : "); scanf("%lf",&szam3);
    if (szam1 > szam2)
    {
        if (szam1 > szam3)
            max = szam1;
        else
            max = szam3;
    }
    else
    {
        if (szam2 > szam3)
            max = szam2;
        else
            max = szam3;
    }
    printf("A nagyobbik szam: %lf\n", max);
    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 végrehajtódnak az utasítások a folytatási / leállási feltételig;
- 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, végrehajtódik 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;
}
```

## Fibonacci sorozat n-edik eleme

Határozzuk meg a Fibonacci sorozat  $n$ -edik elemét!

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

long int fibo(int n)
{
    if (n == 1 || n == 2)
        return 1;
    else
```

```

    return fibo(n - 1) + fibo(n - 2);
}

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

```

### Legnagyobb közös osztó

Olvassunk be két természetes számot, majd számítsuk ki rekurzívan a két szám legnagyobb közös osztóját ( $a, b \in \mathbb{N}^*$ )!

```

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

int lnko(int a, int b)
{
    if (b % a == 0)
        return a;
    else
        return lnko(a, b % a);
}

void main()
{
    int a, b;
    do
    {
        printf("a, b = ");
        scanf("%d%d", &a, &b);
    } while (a < 0 || b < 0);
    printf("Lnko(%d, %d) = %d ", a, b, lnko(a, b));
    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
            return x[kozepe];
}
```

```

{
    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 számokat!\n");
    for (i = 0; i < n; i++)
    {
        printf("Az %d .elem: ", i);
        scanf("%d", &x[i]);
    }
    printf("A legnagyobb szám : %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

```