INFORMATIKA

C NYELV

A 2016-OS ÉRETTSÉGI KÖVETELMÉNYEINEK
MEGFELELŐ OKTATÁSI SEGÉDANYAG

Tartalomjegyzék

Algoritmusok – pszeudókód

Abszolút érték

Határozzuk meg és írjuk ki adott valós szám abszolút értékét!

```
Algoritmus Abszolút_érték(x,mod):
 Ha x ≥ 0 akkor                    {bemeneti adat: x, kimeneti adat: mod}
  mod ← x
 különben
  mod ← -x
 vége(ha)
Vége(algoritmus)
```

Hányados ismételt kivonással

Számítsuk ki két természetes szám egész hányadosát ismételt kivonásokkal!

```
Algoritmus Osztás(a,b,hányados):
 hányados ← 0                  {bemeneti adatok:a, b, kimeneti adat:hányados}
 Amíg a ≥ b végezd el:
  hányados ← hányados + 1
  a ← a - b
 vége(amíg)
Vége(algoritmus)
```

Legnagyobb közös osztó

Számítsuk ki két természetes szám legnagyobb közös osztóját!

```
Algoritmus Eukleidész(a,b,lnko):
 Ismételd                          {bemeneti adatok: a, b, kimeneti adat: lnko}
  r ← maradék[a/b]                          {kiszámítjuk az aktuális maradékot}
  a ← b                                     {az osztandót felülírjuk az osztóval}
  b ← r                                     {az osztót felülírjuk a maradékkal}
 ameddig r = 0                     {amikor a maradék 0, véget ér az algoritmus}
 lnko ← a                                   {lnko 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 ← 0
 Minden i=1,n végezd el:
  Be: szám
  Ha szám páros akkor
   db ← 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 ← szám                    {bemeneti adat: szám, kimeneti adat: válasz}
 újszám ← 0
 Amíg szám > 0 végezd el:
  számjegy ← maradék[szám/10]
  újszám ← újszám*10 + számjegy
  szám ← [szám/10]
 vége(amíg)
 válasz ← újszám = másolat
                    {ha újszám = másolat, akkor válasz értéke igaz}
                  {ha újszám ≠ 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 ← a                                  {bemeneti adatok: a, b}
 y ← b                                  {kimeneti adat: p}
 p ← 0
```

```
 Amíg x > 0 végezd el:                                   {xy + p = ab (*)}
  Ha x páratlan akkor
   p ← p + y
  vége(ha)
  x ← [x/2]
  y ← y + y
 vége(amíg)
Vége(algoritmus)
```

Minimum keresés

Határozzuk meg egy *n* elemű sorozat minimumát!

```
Algoritmus Minimum(n,a,min):
 min ← a1
 Minden i=2,n végezd el:
  Ha ai < min akkor
   min ← ai
  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                    {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 szamot:");
  scanf("%d", &x);
  if (x >= 0)
    abs = x;
  else
    abs = -x;
  printf("A szam abszolut erteke: %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 ∈ 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
```

```
  {
    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
```