

# ***SZAKDOLGOZAT***

*Fejes Judit*

***Debrecen***

***2007***

**Debreceni Egyetem**

**Informatikai kar**

***SZÁMELMÉLETI FELADATOK  
MEGOLDÁSA SZEMÉLYI  
SZÁMÍTÓGÉPPEL***

Témavezető:  
Dr. Papp Zoltán  
egyetemi adjunktus

Készítette:  
Fejes Judit  
Informatika tanárszak  
levelező tagozat

***Debrecen***

***2007***

## Tartalomjegyzék

|                                                                                                       |    |
|-------------------------------------------------------------------------------------------------------|----|
| Bevezetés.....                                                                                        | 2  |
| 1. Néhány szóban a C nyelv sajátosságairól .....                                                      | 3  |
| 2. Az oszthatóság .....                                                                               | 4  |
| 3. Prímszámok.....                                                                                    | 6  |
| 4. Eratoszthenész-szitája .....                                                                       | 7  |
| 5. A számelmélet alaptétele.....                                                                      | 9  |
| 6. Legnagyobb közös osztó .....                                                                       | 10 |
| 7. Euklidészi algoritmus.....                                                                         | 12 |
| 8. Az euklidészi algoritmus kibővített változata.....                                                 | 15 |
| 9. Relatív prím.....                                                                                  | 17 |
| 10. Rácsok .....                                                                                      | 21 |
| 11. Legkisebb közös többszörös .....                                                                  | 24 |
| 12. Néhány nevezetes tétel, sejtés és megállapítás a prímszámokkal és eloszlásukkal kapcsolatban..... | 26 |
| 13. Feladat .....                                                                                     | 28 |
| 14. Ikerprímek, prímhármasok, prímnégyesek .....                                                      | 31 |
| 15. Pitagoraszai számhármasok .....                                                                   | 38 |
| 16. Két négyzetszám tétel .....                                                                       | 40 |
| 17. Négy négyzetszám probléma .....                                                                   | 43 |
| 18. Fermat sejtés.....                                                                                | 44 |
| 19. Számelméleti függvények .....                                                                     | 44 |
| 20. Tökéletes számok .....                                                                            | 49 |
| 21. Nevezetes lineáris felbontási problémák.....                                                      | 50 |
| 21.1. Partíció probléma .....                                                                         | 50 |
| 21.2. Pénzváltási probléma .....                                                                      | 51 |
| 21.3. A Goldbach-sejtés .....                                                                         | 52 |
| 22. Kombinatorikai és egyéb feladatok.....                                                            | 54 |
| Irodalomjegyzék .....                                                                                 | 61 |

## Bevezetés

Szakkdolgozatomnak egy német nyelvű matematika könyv fordítását, feldolgozását választottam, melynek címe: „Arthur Engel: Mathematisches experimentieren mit dem PC” (Arthur Engel: Matematikai kísérletezés személyi számítógéppel). Az USA-ban is megjelent „New Mathemahical Library” címmel.

A szerző eredeti célja az volt, hogy matematikai példák által motiválja a tanulókat, a hallgatókat és matematikatanárokat, a PASCAL programnyelv elsajátítása. A könyv hét fejezetből áll, ezek közül a Zahlentheoretische Algorithmen (Számelméleti algoritmusok) címűt választottam feldolgozásra. Ennek a könyvnek ez a fejezete azért nyerte meg a tetszésemet, mert magam is matematikatanár vagyok, érdekelnek a számelméleti problémák, felvetések. Külön élményt jelent számomra, hogy ez a könyv PASCAL programnyelven írt matematikai programokat is tartalmaz, ezeknek a C programnyelvbe való átfordításával számot adhatok az informatikatanári szakon szerzett szaktudásomról. Igyekeztem a témát úgy feldolgozni, hogy egy magyar nyelvű lefordított, jól használható szakirodalmat alkossak vele. Mivel az eredeti könyv nem programozás könyv, ezért is tartottam fontosnak a magyar matematika oktatási sajátosságait figyelembe venni, ezáltal egészítettem ki a könyv egyes részeit a számelmélet témakörébe tartozó legfontosabb definíciókkal, tételekkel, példákkal és érdekességekkel, programokkal és programrészekkel.

## 1. Néhány szóban a C nyelv sajátosságairól

### A C típusrendszere

| aritmetikai típusok<br>(egyszerű típusok)         | származtatott típusok (összetett típusok) | void típus |
|---------------------------------------------------|-------------------------------------------|------------|
| <i>integrális típusok</i>                         | tömb                                      |            |
| egész (int, short[int], long[int])                | függvény                                  |            |
| karakter (char)                                   | mutató                                    |            |
| felsorolásos                                      | struktúra                                 |            |
| <i>valós típusok</i> (float, double, long double) | union                                     |            |

### A Pascal és a C nyelv adattípusainak összefoglalása

| Turbo Pascal |               |                           | Turbo C       |               |                           |
|--------------|---------------|---------------------------|---------------|---------------|---------------------------|
| Típus        | Hossza (byte) | Tartománya                | Típus         | Hossza (byte) | Tartománya                |
| char         | 1             | 0-255                     | char          | 1             | -128- 127                 |
| byte         | 2             | 0-255                     | unsigned char | 1             | 0-255                     |
| integer      | 2             | -32768-32767              | short         | 2             | -32768-32767              |
|              |               |                           | int           | 2             | -32768-32767              |
| word         | 2             | 0-65535                   | unsigned int  | 2             | 0-65535                   |
| longint      | 4             | -231- 231-1               | long          | 4             | -231- 231-1               |
|              |               |                           | unsigned long | 4             | 0-232-1                   |
| single       | 4             | $\pm 3.4e \pm 38$         | float         | 4             | $\pm 3.4e \pm 38$         |
| real         | 6             | $1e-38 - 1e \pm 38$       | -             |               |                           |
| double       | 8             | $\pm 1.7e \pm 308$        | double        | 8             | $\pm 1.7e \pm 308$        |
| extended     | 10            | $\pm 3.4e-4932-1.1e+4932$ | long double   | 10            | $\pm 3.4e-4932-1.1e+4932$ |
| boolean      | 1             | false, true               | -             |               |                           |

Logikai adattípus nincs a C nyelvben, hamisnak az int 0 felel meg, minden más értéket igaznak tekint a C. A C-ben a Pascal real típusának nincs pontos megfelelője, helyette a float, illetve a double használatos.

### A Pascal program struktúrája

```
program programnév;
var
{ változó_deklarációk }
```

### A C program struktúrája:

```
<preprocesszor parancsok>
<típusdefiníciók>
```

|                     |                         |
|---------------------|-------------------------|
| {egyéb deklarációk} | <függvény prototípusok> |
| begin               | <globális változók>     |
| {utasítások}        | <függvények>            |
| end.                |                         |

## 2. Az oszthatóság

A számelméleti algoritmusok alapja az oszthatóság.

Definíció: Azt mondjuk, hogy az  $a$  egész szám osztja a  $b$  egész számot, ha létezik  $c$  egész szám, hogy  $b=ac$ . Jelölés:  $a|b$ . Ekkor  $a$ -t  $b$  osztójának,  $b$ -t  $a$  többszörösének nevezzük.

Oszthatósági szempontból a nullát minden egész szám osztja. A nulla nullával való oszthatósága nem jelenti azt, hogy az osztás el is végezhető. Az egységek pedig minden számot osztanak. Egységeknek az 1 egységelem osztóit nevezzük, és  $e$ -vel jelöljük.

Tétel: Az oszthatóság tulajdonságai az egész számok körében az alábbiak:

$a|a$  (reflexív)

$a|b$  és  $b|c \Rightarrow a|c$  (transzitiv)

$a|b$  és  $a|c \Rightarrow a|(b \pm c)$  (additív)

$a|b+c$  és  $a|b \Rightarrow a|c$

$a|b \Rightarrow a|bd$

$a|b$  és  $a|b+c \Rightarrow a|c$

$a|b$  és  $c|d \Rightarrow ac|bd$  (multiplikatív)

$1|a, a|0$

$a|1 \Rightarrow a = \pm 1$

$0|a \Leftrightarrow a = 0$

$ac|bc$  és  $c \neq 0, \Rightarrow a|b$

$a|b$  és  $b|a \Rightarrow a = \pm b$

$d|a$  és  $d|b \Rightarrow d|ax+by$

nem antiszimmetrikus, mert  $a|-a$  és  $-a|a$ , de  $a \neq -a$  és nem is szimmetrikus.

Tetszőleges  $a, b, c, d$  egész számokra.

Természetes számok körében az oszthatóság antiszimmetrikus is,  $a|b$  és  $b|a \Rightarrow a=b$  ( $a, b \in N$ )

Tehát félig rendezési reláció.

Az alábbi program az osztandót és az osztót kéri, majd kiírja a hányadost és a maradékot. A program csak egész számokkal dolgozik.

```
#include <stdio.h>
#include <stdlib.h>
int a,b,c,e;
int main()
{
    printf("Osztandó="); scanf ("%d",&a);
    printf("Osztó="); scanf ("%d",&b);
    c==a/b;
    printf("Hányados: %d / %d=%d\n ", (int) a, (int) b, (int)
a / (int) b);
    e=a%b;
    printf("Maradék: %d\n", e);
    system("pause");
    return 0;
}
```

Definíció: Ha egy nemzérus  $a$  egész számot felírunk  $a=bc$  ( $b,c \in \mathbb{Z}$ ) alakban, akkor akkor  $a$  egy faktorizációját (szorzattá bontását) kapjuk. Ha  $b$  és  $c$  egyike sem egység, akkor  $bc$  valódi faktorizáció, amelyben  $b$  és  $c$  valódi osztói  $a$ -nak.

Definíció: Az  $a$  egész szám egységtől és  $\pm a$ -tól különböző osztóit valódi osztónak nevezzük.

Logikai jelekkel:  $b|a \wedge \neg(b \sim 1) \wedge \neg(b \sim a) \Rightarrow b$  valódi osztója  $a$ -nak.

Írjunk programot, amely kiírja egy 2-nél nagyobb szám összes valódi osztóját!

```
#include <stdio.h>
#include <stdlib.h>
main ()
{
    int szam, osztó;
    printf("\nA a szám összes valódi osztója\n");
    printf("Szám="); scanf ("%d",&szam);
```

```

printf("Osztók= \n");
osztó=2;
while (osztó<=szam/2)
{
    if(szam% osztó==0)
        printf("%d ",osztó);
    osztó++;
}
system("pause");
}

```

### 3. Prímszámok

A prímszámok vagy törzsszámok nagy jelentőséggel bírnak a matematika történetében. A prímszámok fogalmát már az egyiptomiak és a mezopotámiaiak is ismerték, ám első komoly tanulmányozói a püthagoreusok voltak és e fogalom pontos meghatározását Euklidésznél (Kr.e.300) találunk. Még ma is sok megoldatlan probléma kapcsolódik hozzájuk.

**Definíció:** A  $p$ -t prímszámnak nevezzük, ha  $plab$ -t ,akkor  $pla$  vagy  $plb$  ,( és  $p$  nem 0 és különbözik az egységektől). A zérus és az egységek nem törzsszámok, nem prímszámok és nem összetett számok.

**Lemma:** Minden összetett természetes szám legkisebb valódi osztója prímszám.

Egy  $p$  egész szám akkor és csakis akkor prímszám, ha törzsszám.

Ennek megfelelően, írjunk olyan programot, amely megállapítja egy természetes számról, hogy az prímszám, vagy összetett szám, írja ki a program, ha nem természetes számot írunk be, amennyiben 1-et vagy 0-a értéket adtunk a számnak, írja ki hogy nem prímszám és nem összetett szám.

```

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
main()
{

```

```

int szam, osztó;
printf ("Szám:"); scanf ("%d", &szam);
{
    if ((szam==1)|| (szam==0)) printf ("nem prímszám, és
        nem is összetett szám");
    else if (szam>1)
    {
        for (osztó=2; osztó<=sqrt(szam)&& szam%osztó!=0;
            ++osztó);
        if (osztó<=sqrt(szam)||szam==1)
            printf ("Összetett szám");
        else printf ("Prímszám");
    }
    else
        printf ("Nem természetes szám");
}
system ("pause");
}

```

#### 4. Eratoszthenész-szitája

Eratoszthenész, a neves ókori görög matematikus, aki kertjében kifeszített pergamenlapra írta föl a számokat, majd átszúrta a kieső számokat. Laikus szomszédai azt hitték, hogy valamilyen különös szitát készít, ezért nevezték el ezt az eszközt Eratoszthenész-szitájának. Eratoszthenész módszere, melynek segítségével egyszerű kizárásos algoritmussal megállapíthatjuk, egy adott felső határig meg tudjuk adni a prímszámokat.

Ennek megfelelően a természetes számokat felírjuk 2-től  $n$ -ig, majd bekarikázzuk az első számot a 2-est, ezután kihúzzuk ennek többszöröseit (azaz minden másodikat). A megmaradó számok közül bekarikázzuk ismét az első, és kihúzzuk ennek többszöröseit (azaz minden harmadikat) s így tovább. Természetesen előfordulhat, hogy egy számot nemcsak egy alkalommal húzunk ki. Nyilván elegendő csupán az 1 és  $\sqrt{n}$  közötti  $p$  prímeikkel elvégezni a szitálást, mivel ha valamely  $a$  szám  $n$ -nél kisebb és összetett, akkor van  $\sqrt{n}$ -nél kisebb prím osztója. A bekarikázott, illetve a ki nem húzott számok lesznek  $n$ -ig az összes prímszámok. A

$\sqrt{n}$ -nél nem nagyobb, bekarikázott számok azért prímszámok, mert nincsen valódi osztójuk. A  $\sqrt{n}$ -nél nagyobb, ki nem húzott számok pedig azért prímszámok, mert ellenkező esetben lenne  $\sqrt{n}$ -nél nem nagyobb prímtényezőjük, ami nyilvánvalóan  $\sqrt{n}$ -nél sem nagyobb, s ezért kihúztuk volna őket.

|               |               |               |               |               |               |               |               |               |               |
|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
|               | (2)           | (3)           | <del>4</del>  | (5)           | <del>6</del>  | (7)           | <del>8</del>  | <del>9</del>  | <del>10</del> |
| (11)          | <del>12</del> | (13)          | <del>14</del> | <del>15</del> | <del>16</del> | (17)          | <del>18</del> | (19)          | <del>20</del> |
| <del>21</del> | <del>22</del> | (23)          | <del>24</del> | <del>25</del> | <del>26</del> | <del>27</del> | <del>28</del> | (29)          | <del>30</del> |
| (31)          | <del>32</del> | <del>33</del> | <del>34</del> | <del>35</del> | <del>36</del> | (37)          | <del>38</del> | <del>39</del> | <del>40</del> |
| (41)          | <del>42</del> | (43)          | <del>44</del> | <del>45</del> | <del>46</del> | (47)          | <del>48</del> | <del>49</del> | <del>50</del> |

Ezt az eljárást Eratoszthenészi-szitának nevezik.

A szitából látható, hogy a prímszámok rendkívül szabálytalanul helyezkednek el a természetes számok között. Azt is észrevehetjük, hogy a szita elején sokkal több prímszám van. Minél nagyobb számokból álló intervallumban keresünk, annál kevesebb számú prímét találunk.

A következő program egy természetes számot kér, majd kiírja eddig a számig a prímszámokat.

```
#include <stdio.h>
#include <stdlib.h>
int i, j, z[5000], N;
int main(void)
{
    printf("N="); scanf("%d", &N);
    for(i=2; i<=N; ++i)
        z[i]=i;
    for(i=2; i<N; i++)
    {
        for(j=i; j<=N; ++j)
```

```

    {
        if(z[j]!=0 && j!=i && z[j]%i ==0)
            z[j]=0;
    }
}
for(i=2;i<=N;++i)
    if(z[i]!=0)
        printf("%d ", z[i]);
    system("pause");
}

```

## 5. A számelmélet alaptétele

Gauss fogalmazta meg és bizonyította először 1801-ben, bár jóval előbb felhasználták a tételt.

Tétel: Minden nemzérus és nem egység egész szám sorrendtől és egységtényezőktől eltekintve bontható fel prímszámok szorzatára.

A számelmélet alaptétele azt mutatja, hogy a prímszámok az egész számok építőkövei.

Minden egynél nagyobb természetes szám felírható  $n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdots p_k^{\alpha_k} = \prod_{i=1}^k p_i^{\alpha_i}$  kanonikus alakban, ahol  $\alpha_i > 0$ . Ezt a műveletet törzstényező felbontásnak nevezzük.

Például:  $2880 = 2^6 \cdot 3^2 \cdot 5$

Ez a tétel az egyik oka annak, hogy az 1-et kihagyjuk a prímszámok halmazából. Ha az 1-et prímszámnak vennénk, a tételhez további megkötéseket kellene adnunk.

Írjuk fel a természetes számok törzstényező felbontását!

```

#include <stdio.h>
#include <stdlib.h>
int szam, osztó;
main()
{
    printf (" a vizsgálandó szám:"); scanf ("%d", &szam);
    printf ("törzstényezői: ");

```

```

    osztó=1;
do
{
    do { osztó=osztó+1;
    }
    while (szam%osztó!=0);
    do {
        szam=szam/osztó;
        if(szam==1) printf("%d",osztó);
        else printf ("%d*",osztó);
    }
    while (szam%osztó==0);
}
while (szam!=1);
printf("\n");
system ("pause");
}

```

## 6. Legnagyobb közös osztó

Ismerjük a számok prímtényezős felbontását, kiszámolhatjuk két egész szám legnagyobb közös osztóját.

Ha  $a$  és  $b$  legalább egyike 0-tól különböző egész. Az  $a$ ,  $b$  közös osztói legnagyobb közös osztójának nevezünk egy  $d$  elemet, ha

- (i)  $d$  közös osztó, azaz  $d|a \wedge d|b$ ;
- (ii)  $d$  az  $a$ ,  $b$  bármely közös osztójának többszöröse, azaz  $(d'|a \wedge d'|b) \Rightarrow d'|d$

Jelölés:  $d=(a,b)$  vagy  $d=\text{lnko}(a,b)$  vagy  $d=\text{lnko}\{a,b\}$ .

Tulajdonságai:

$$\text{lnko}(0,0)=0$$

$$\text{lnko}(a,b)=\text{lnko}(b,a)$$

$$\text{lnko}(a,1)=1$$

$$\text{lnko}(a,a)=|a| \quad (\mathbb{N}^+ \text{-ban van idempotencia})$$

$$\text{lnko}(a,0)=|a|$$

$$\text{lnko}(a,b)=\text{lnko}(-a,b)$$

$$\text{lnko}(a,b)=\text{lnko}(|a|,|b|)$$

$$\text{lnko}(a,ka)=|a| \text{ bármely } k \in \mathbb{Z}\text{-re}$$

$$\text{lnko}((a,b),c)=(a,(b,c))$$

$$\text{lnko}(a,b)c=(ac,bc) \quad (\forall c \in \mathbb{Q}, ac, bc \in \mathbb{Z} \text{ és } c \neq 0)$$

$$\text{lnko}(a,b)=(a+bc,b)$$

$$\text{lnko}(a,b)=a \Leftrightarrow a|b$$

$$\text{lnko}(ac,bc)=c \text{ lnko}(a,b)$$

$$\text{lnko}\left(\frac{a}{(a,b)}, \frac{b}{(a,b)}\right)=1$$

$$[(a,b)=1 \wedge (a,c)=1] \Rightarrow (a,bc)=1$$

$$n|ab \text{ és } \text{lnko}(a,n)=1, \text{ akkor } n|b$$

$$(a,b)=|a| \Leftrightarrow a|b$$

$a, b, c$  elemei az egész számok halmazának.

Írjunk ki két egész szám legnagyobb közös osztóját!

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    unsigned a, b;
    printf("a="); scanf("%u", &a);
    printf("b="); scanf("%u", &b);
    {
        while (a!=b)
            if (a>b)
                a-=b;
            else
                b-=a;
        printf("A legnagyobb közös osztó: %u.\n", a);
    }
}
```

```

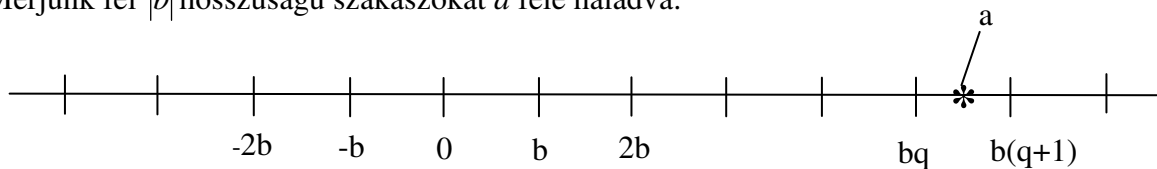
        system ("pause");
    }
}

```

## 7. Euklidészi algoritmus

Az euklidészi (maradékos) osztás tétele: Bármely  $a$  és  $b \neq 0$  egész számokhoz egyértelműen léteznek  $q$  és  $r$  egész számok, melyekre  $a=bq+r$  és  $0 \leq r < |b|$  vagy  $r=0$ . (A  $b|a$  oszthatóság  $b \neq 0$  esetén pontosan akkor teljesül, ha a maradék 0.)

Szemléltetése számegyenesen úgy történik, hogy feltesszük, hogy  $a$  is,  $b$  is pozitív és  $b < a$ . Mérjük fel  $|b|$  hosszúságú szakaszokat  $a$  felé haladva.



**Két esetet különböztetünk meg:**

1. eset szerint  $|b|$  valamely többszöröse egybeesik  $a$ -val, így van olyan  $q$  egész szám, melyre teljesül:  $a=bq$ , ekkor  $r=0$ .

2. eset szerint  $a$  két többszörös közé esik:  $a=bq+r$ ,  $0 < r < |b|$ . A két esetet együttesen tekintve jutunk el az imént említett tételhez.  $0 \leq r = a - bq < b$ , melyet felírhatunk úgy is, hogy  $bq \leq a < b(q+1)$ , ( ekkor  $a$  a  $[bq, b(q+1))$  intervallumba esik) így  $q \leq \frac{a}{b} < q+1$ . Ilyen  $q$  egész szám pontosan egy létezik.

$q = \left\lfloor \frac{a}{b} \right\rfloor$ , azaz  $a$  legnagyobb olyan egész szám, amely még kisebb, vagy egyenlő, mint  $\frac{a}{b}$ ,

$r = a - b \left\lfloor \frac{a}{b} \right\rfloor$ . A maradékos osztásnál kapott  $q$  számot hányadosnak, az  $r$ -et pedig (legkisebb nemnegatív) maradéknak nevezzük. C-ben ezeket a műveleteket  $q=a|b$ ,  $r=a \% b$  -vel jelöljük.

Az  $a$  és  $b$  egész számokon ( $b \neq 0$ ) végrehajtott euklidészi osztások következő sorozatát euklidészi algoritmusnak nevezzük. Az euklidészi algoritmust két egész szám legnagyobb közös osztójának meghatározására használják.

$$a = bq_0 + r_0, \quad 0 < r_0 < |b|,$$

$$b = r_0q_1 + r_1, \quad 0 < r_1 < r_0$$

$$r_0 = r_1q_2 + r_2, \quad 0 < r_2 < r_1$$

.....

.....

$$r_{n-2} = r_{n-1}q_n + r_n, \quad 0 < r_n < r_{n-1},$$

$$r_{n-1} = r_nq_{n+1} + 0, \quad (\text{azaz } r_{n+1} = 0)$$

Tétel: Az euklidészi algoritmus utolsó nem nulla maradéka  $r_n$  az  $a$  és  $b$  legnagyobb közös osztója.

Például: (360,225)

$$a = b \cdot q + r$$

$$360 = 225 \cdot 1 + 135$$

$$225 = 135 \cdot 1 + 90$$

$$135 = 90 \cdot 1 + 45$$

$$90 = 45 \cdot 2 + 0$$

$$(360, 225) = 45$$

A legnagyobb közös osztó rekurziós tétele: Tetszőleges  $a$  nemnegatív és  $b$  pozitív egész számokra  $\text{lko}(a, b) = \text{lko}(b, a \bmod b)$

Erre a tételre épülő rekurziós program:

EUKLIDESZ ( $a, b$ )

if  $b=0$

then return  $a$

else return EUKLIDESZ ( $b, a \bmod b$ )

Például:  $\text{lko}(30, 21)$

EUKLIDESZ (30,21)=

EUKLIDESZ (21,9)=

EUKLIDESZ (9,3)=

EUKLIDESZ (3,0)=

=3

A program háromszor is behívja az Euklideszi algoritmust. Az algoritmus helyességét az előző tétel garantálja. Ha az algoritmus a 2. sorban az  $a$  számmal tér vissza, akkor a  $b$  csakis 0 lehet, mert  $lnko(a,b)=lnko(a,0)=a$ . Az algoritmus hívása nem ismétlődhet végtelen sokszor, mivel a második argumentum szigorúan csökken minden újabb hívásnál, így az algoritmus mindig megadja a helyes választ.

Legyen  $a \geq b \geq 0$ . A tétel segítségével elő tudjuk állítani  $a$  és  $b$  egész számok legnagyobb közös osztóját.  $(b, a \bmod b)$  lépés ismétlésével, mindig kisebb számpárokat kapunk, míg  $(g,0)$ -t el nem érjük.

$lnko(a,b)=lnko(g,0)=g$

Az euklideszi algoritmust megvalósító program:

```
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
unsigned szam1, szam2, maradek;
main()
{
    printf("szam1, szam2: " );
    scanf("%u,%u", &szam1, &szam2);
    if (szam2==0)
    {
        printf("%u\n", szam1);
        system("pause");
        return 0;
    }
    if (szam1==0)
    {
```

```

        printf("%u\n", szam2);
        system("pause");
        return 0;
    }
    if(szam1==szam2)
        printf("A legnagyobb kozos osztó: %u\n", szam1);
    else
    {
        do
        {
            maradék=szam1 % szam2;
            szam1=szam2;
            szam2=maradék;
        }
        while (maradék);
        printf("A legnagyobb kozos osztó %u\n", szam1);
    }
    system("pause");
    return 0;
}

```

## 8. Az euklideszi algoritmus kibővített változata

Azon  $x, y$  együtthatók kiszámítása a célunk, amelyekre

$$d = \text{lnko}(a, b) = ax + by. \quad (1)$$

$x$  és  $y$  lehet 0 és negatív is. A KIBŐVÍTETT- EUKLIDESZ algoritmus bemenő adata egy nemnegatív számpár, a kimenet pedig a  $(d, x, y)$  számhármasság, amely kielégíti a (1) egyenletet.

kibővített- euklidesz  $(a, b)$

if  $b=0$

then return  $(a, 1, 0)$

$(d', x', y') \leftarrow \text{KIBŐVÍTETT- EUKLIDESZ}(b, a \bmod b)$

$(d, x, y) \leftarrow (d', y', x' - \lfloor a/b \rfloor y')$

return ( $d, x, y$ )

| $a$ | $b$ | $\lfloor a/b \rfloor$ | $d$ | $x$ | $y$ |
|-----|-----|-----------------------|-----|-----|-----|
| 99  | 78  | 1                     | 3   | -11 | 14  |
| 78  | 21  | 3                     | 3   | 3   | -11 |
| 21  | 15  | 1                     | 3   | -2  | 3   |
| 15  | 6   | 2                     | 3   | 1   | -2  |
| 6   | 3   | 2                     | 3   | 0   | 1   |
| 3   | 0   | -                     | 3   | 1   | 0   |

A KIBŐVÍTETT- EUKLIDESZ algoritmus az EUKLIDESZ algoritmus egy változata.

Ahol  $a, b$  a bemenő adatok,  $\lfloor a/b \rfloor$  a kiszámított érték,  $d, x, y$  a visszatérő érték. Mindegyik sor a rekurzó egy újabb szintjét mutatja. A KIBŐVÍTETT- EUKLIDESZ (99, 78) algoritmus a (3, -11, 14) számhármast szolgáltatja, így  $\text{lnko}(99, 78) = 3 = 99 \cdot (-11) + 78 \cdot 14$

$b=0$  esetben a KIBŐVÍTETT- EUKLIDESZ algoritmus  $d=a$  eredményhez vezet,  $x=1$  és  $y=0$  együtthatókat is megadja, amelyekkel:  $d=a=ax+by$ .

$b \neq 0$  esetben az algoritmus először a  $(d', x', y')$  számhármast számítja ki, ahol  $d' = \text{lnko}(b, a \bmod b)$

Programozása C-ben:

```
#include <stdio.h>
#include <stdlib.h>
int a,b;
int q;
void dio(int a,int u,int v,int b,int x,int y)
{
    if (b==0)
        printf ("%d %d %d", a,u,v);
    else
```

```

        q=a/b;
        dio(b,x,y,a%b,u-q*x,v-q*y);
    }
main()
{
    printf ("a="); scanf ("%d", &a);
    printf ("b="); scanf ("%d", &b);
    dio (a,1,0,b,0,1);
    system ("pause");
}

```

## 9. Relatív prím

Definíció: Két vagy több 0-tól különböző egész számot relatív prímnek nevezzük, ha legnagyobb közös osztójuk 1.

Pl:  $(17, 61)=1$ ,  $(8, 21)=1, \dots$

Definíció: Az  $a_1, a_2, \dots, a_n$  ( $n \geq 2$ ) nem mind zérus számok relatív príme, ha legnagyobb közös osztójuk 1. Ha közülük bármely kettő különböző relatív prím, akkor páronként relatív príme.

A következő program megadja két egész szám legnagyobb közös osztóját és kiírja, hogy azok relatív príme-e vagy sem:

```

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
int szam1, szam2, maradek,i,n,relprim;
main()
{
    printf("szam1, szam2: " );
    scanf("%d,%d", &szam1, &szam2);
    for (i=2;i<n;i++)
        if (szam2==0)
        {
            printf("%d\n", szam1);

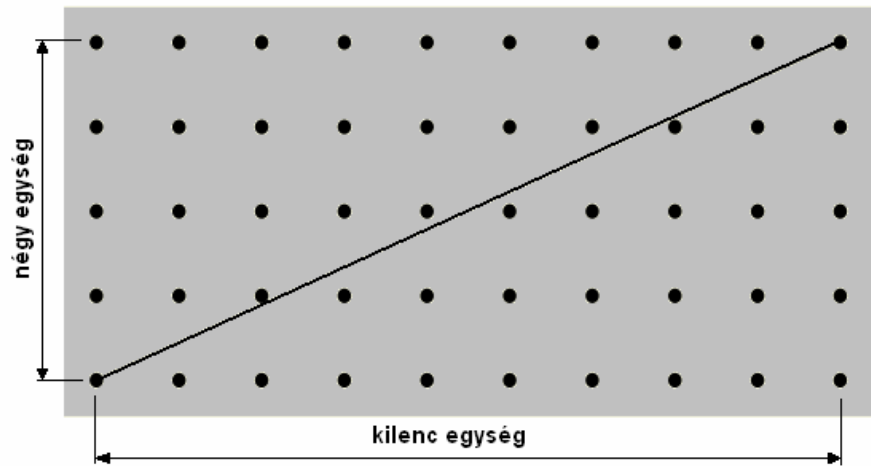
```

```

        system("pause");
        return 0;
    }
    if (szam1==0)
    {
        printf("%d\n", szam2);
        system("pause");
        return 0;
    }
    if(szam1==szam2)
    printf("A legnagyobb kozos osztó: %d\n", szam1);
    else
    {
        do {
            maradék=szam1 % szam2;
            szam1=szam2;
            szam2=maradék;
        }
        while (maradék);
        printf("A legnagyobb kozos osztó %d\n", szam1);
        if(szam1==1)
            printf("Relativ primek.\n");
        else printf("Nem relativ primek.\n");
        system("pause");
    }
}

```

Felvetődik bennünk az a kérdés, hogy mi annak a valószínűsége, hogy véletlenszerűen kiválasztva két egész számot, ezek egymáshoz relatív prímek.



a 4 és a 9 relatív prímek, mert az átló egyetlen rácsponton sem megy keresztül

Annak valószínűsége, hogy egy tetszőleges szám osztható a  $p$  prímmel  $1/p$ -vel egyenlő. Ebből következik, hogy annak a valószínűsége, hogy mindkét szám osztható ezzel a prímmel,  $1/p^2$ , annak a valószínűsége, hogy legalább egyikük nem osztható vele,  $1-1/p^2$ . Annak a valószínűségét, hogy két szám relatív prím, úgy kapjuk, hogy felírjuk annak az eseménynek a valószínűségét, hogy egyik prímszám sem osztja őket. Ekkor tehát minden prímszámmra felírjuk és összeszorozzuk az  $1-1/p^2$  valószínűséget:

$$p = \prod \left( 1 - \frac{1}{q^2} \right)$$

$$\frac{1}{p} = \frac{1}{1 - \frac{1}{2^2}} * \frac{1}{1 - \frac{1}{3^2}} * \frac{1}{1 - \frac{1}{5^2}} \dots$$

$$\frac{1}{p} = \left( \sum_{n \geq 0} \frac{1}{2^{2n}} \right) * \left( \sum_{n \geq 0} \frac{1}{3^{2n}} \right) * \left( \sum_{n \geq 0} \frac{1}{5^{2n}} \right) \dots = \sum_{n \geq 1} \frac{1}{n^2} = \frac{\pi^2}{6} \Rightarrow p = \frac{6}{\pi^2} \quad ( \pi \text{ közelítése Euler}$$

$$\text{módszere } \frac{\pi^2}{6} = 1 + \frac{1}{2^2} + \frac{1}{3^2} \dots )$$

Tétel: Két szám relatív prímiségének valószínűsége:  $\frac{6}{\pi^2}$

Annak a valószínűsége, hogy két véletlenül választott egész szám relatív prím, körülbelül 60%.

$$s = \sum_{n \geq 1} \frac{1}{n^2} = 1 + \frac{1}{4} + \frac{1}{9} + \dots + \frac{1}{n^2} + F(n)$$

$$F(n) = \frac{1}{(n+1)^2} + \frac{1}{(n+2)^2} + \dots$$

$$\int_{n+0.5}^{n+1.5} \frac{dx}{x^2} = \left[ -\frac{1}{x} \right]_{n+0.5}^{n+1.5} = \frac{1}{n+0.5} - \frac{1}{n+1.5}$$

$$F(n) \approx \int_{n+0.5}^{\infty} \frac{dx}{x^2} = \left[ -\frac{1}{x} \right]_{n+0.5}^{\infty} = \frac{1}{n+0.5}$$

Miután  $F(n)$ -nek jó becslést találtunk, számoljuk ki először  $s(n)$ -t, a kerekítési hibák minimalizálása végett, és adjuk hozzá  $\frac{1}{n+0.5}$ .

Teszteljük  $n=500, 1000, 1500\dots$

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int i,n,j;
double osszeg, x, y, v,z;
main ()
{
    printf ("n="); scanf ("%d", &n); osszeg=0;
    j=n;
    for (i=1; i<n+1; i++){
        x=(1/(double)j);
        y=(x/(double)j);
        osszeg=osszeg+y;
        j=j-1;
    }
    j=j+1;
    z=n+0.5;
    v=1/z;
    osszeg=osszeg+v;
```

```

printf("osszeg= %12.10lf\n",osszeg);
system ("pause");
}

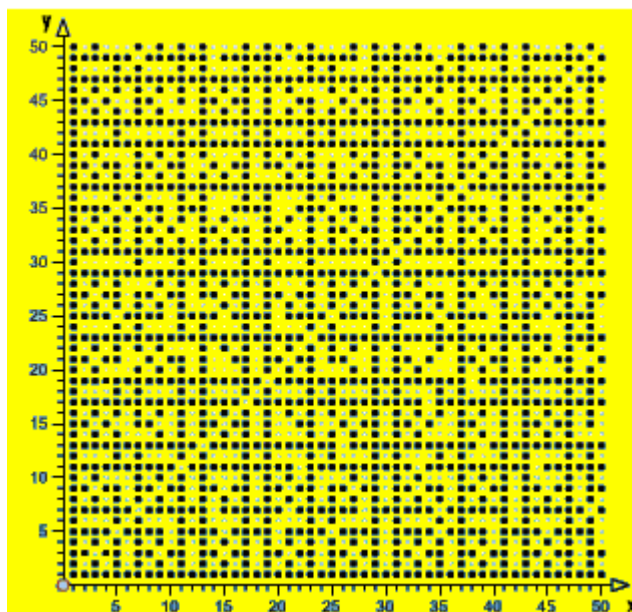
```

| $n$  | tseta        |
|------|--------------|
| 500  | 1.6449340675 |
| 1000 | 1.6449340669 |
| 1500 | 1.6449340669 |

## 10. Rácsok

Legyen  $Z^2$  a síkbeli rács, azon pontok halmaza, mely tartalmazza az  $(x, y)$  egész koordinátájú pontokat. Legyen  $s(n)$  azoknak az  $(x, y)$  koordinátájú pontok száma, amelyek relatív prímek, és amelyekre  $1 \leq x, y \leq n$  teljesül az  $n$  által meghatározott négyzetben.  $p(n) = \frac{s(n)}{n^2}$  a valószínűsége, hogy  $(x, y) = 1$ . Minden látható pont végtelen sok pontot elrejt  $(kx, ky)$   $k=2,3,4,..$  pontokat is. Ebből sejthetjük, hogy  $p(n) \rightarrow 0, n \rightarrow \infty$

Szemléletesen:  $a$  és  $b$  egész szám pontosan akkor relatív prímek, ha a Descartes-féle koordinátarendszerben az  $(a, b)$  koordinátájú pont „látszik” az origóból, azaz nincsen egész koordinátájú pont az origó és az  $(a, b)$  pont között.



Fekete pontok:  $x$  és  $y$  relatív prímek

Fehér pontok:  $x$  és  $y$  nem relatív prímek.

Pl:  $n=10$  esetben 63 pont fekete és 37 pont fehér, azaz  $p(n)=0.63$  (63%) a valószínűsége, hogy  $n \times n$ -es négyzetben az adott pont relatív prímek határoz meg.

Az alábbi program egy számot kér, majd az  $n \times n$ -es Descartes-koordináta-rendszerbeli pontokat megvizsgálva kiírja, hogy hány pont tud relatív prímeket meghatározni, és megadja, hogy mekkora valószínűséggel.

```
#include <stdio.h>
#include <stdlib.h>
int i, j, n, s=0;
double p, x;
int lkko(int a, int b)
{
    if (b==0)
        return a;
    else return lkko(b, a%b);
}
main()
{
    printf ("n="); scanf ("%d", &n);
    for (i=1; i<n+1; i++)
        for (j=1; j<n+1; j++)
            if (lkko(i, j) == 1)
                s=s+1;
    x=n;
    p=s/x;
    printf("s= %d\n", s);
    printf("p= %lf\n", p);
    system ("pause");
}
```

A program futásának eredménye:

| $n$ | $s(n)$ | $p(n)$  |
|-----|--------|---------|
| 10  | 63     | 0.63000 |
| 20  | 255    | 0.63750 |
| 40  | 979    | 0.61187 |
| 50  | 1547   | 0.61880 |
| 100 | 6087   | 0.60870 |
| 120 | 8771   | 0.60910 |
| 150 | 13715  | 0.60956 |
| 200 | 24463  | 0.61157 |

$n=30000$ -et esetén  $n^2 = 9E + 08$  pontot kellene tesztelni, ez az előbbi számokhoz képest igen nagy szám, ezért elégedjünk meg szűrőpróbaszerűen 10000 ponttal. A programban a *rand()* függvényt használjuk, mely a számok véletlenszerű előállítását végzi. Az  $m=10000$ ,  $n=30000$  konstansokat a változók előtt deklaráljuk.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
const m=10000;
const n=30000;
int a,b,i,szam;
int lkko(int a,int b)
{
    if (b==0)
        return a;
    else return lkko(b,a%b);
}
main()
{
    srand( time(0) );
    szam=0;
    for (i=1; i<m+1; i++)
```

```

{
a=1+rand()%(n)/100;
b=1+rand()%(n)/100;

if (lkk(a,b)==1)
szam=szam+1;
}
printf("szam= %d\n",szam);
system ("pause");
}

```

## 11. Legkisebb közös többszörös

A legkisebb közös többszörös: A 0-tól különböző  $a$ ,  $b$  elemek legkisebb közös többszörösének nevezünk egy  $m$  elemet, ha

- (i)  $m$  közös többszörös, azaz  $a|m \wedge b|m$
- (ii)  $m$  az  $a$ ,  $b$  bármely közös többszörösének osztója, azaz  $(a|m' \wedge b|m') \Rightarrow m|m'$

Jele:  $[a,b]$ .

Tétel: Ha zérustól különböző elemeknek van legnagyobb közös osztója, akkor van legkisebb közös többszöröse is, és fennáll:

$$(a,b) \cdot [a,b] = ab$$

$$[a,b] = \frac{ab}{(a,b)}, \text{ speciálisan } [a,b] = ab, \text{ ha } a \text{ és } b \text{ relatív prímek.}$$

Az egész számok halmazán a legkisebb közös többszörös az alábbi tulajdonságokkal rendelkezik még minden  $a, b, c \in \mathbb{Z}$ -re :

$$[a,a] = |a| \quad (\mathbb{Z}^+ \text{-ban van idempotencia})$$

$$[a,b] = [b,a]$$

$$[a,b]c = [ac,bc] \quad (\forall c \in \mathbb{Q}, ac, bc \in \mathbb{Z} \setminus \{0\})$$

$$[[a,b]c] = [a[b,c]]$$

Írjunk olyan programot, amely meghatározza két pozitív egész szám legnagyobb közös osztóját és legkisebb közös többszörösét.

```
#include <stdio.h>
#include <stdlib.h>
int a,b,x,y,u,v,lnko,lkkt;
main ()
{
    printf ("a="); scanf ("%d", &a);
    printf ("b="); scanf ("%d", &b);
    x=a; y=b; u=a; v=b;
    while (x!=y)
    {
        if (x<y)
        {
            y=y-x; v=u+v;
        }
        else
        {
            x=x-y; u=u+v;
        }
    }
    lnko=x; lkkt=(u+v) / 2;
    printf("lnko= %d\n",lnko);
    printf("lkkt= %d\n",lkkt);
    system ("pause");
}
```

## 12. Néhány nevezetes tétel, sejtés és megállapítás a prímszámokkal és eloszlásukkal kapcsolatban

A prímek eloszlásával számos matematikus foglalkozott, köztük Pierre de Fermat, Bernhard Riemann, George Hardy és Erdős Pál.

Tétel: A prímszámok száma végtelen. Ezt a tételt Euklidesz bizonyította be az Elemek IX. könyvében.

Minden 3-nál nagyobb  $p$  prímszám  $6k+1$  vagy  $6k-1$  alakú, ahol  $k=1, 2, 3, \dots$  természetes szám.

Tétel: Végtelen sok  $4k-1$  alakú prímszám van.

Tétel: Végtelen sok  $4k+1$  alakú prímszám van.

Viszont nem minden  $n$ -re adnak a  $4k+1$ , illetve a  $4k-1$  képletek prímszámokat.

Tétel:  $4n+3$  alakú prímek száma végtelen.

Csebisev igazolta Joseph Bertrand sejtését, mely szerint minden, 1-nél nagyobb természetes szám és kétszerese között van prímszám. Szintén Csebisev nevéhez fűződik az alábbi tétel: A  $[2, x]$  intervallumbeli prímszámok sűrűsége asszimptotikusan két 1 körüli korlát között marad:

$$c_1 < \frac{\pi(x)}{\frac{x}{\log x}} < c_2, \text{ ahol } 0.922 < c_1 < 1 < c_2 < 1.105$$

Fermat sejtése: A  $2^{2^n} + 1$  alakú számok prímek. A sejtés csak  $n \leq 4$ -re bizonyított, bizonyos  $n$ -ekre cáfolt, a legtöbb  $n$ -re nyitott.

Fermat-féle prímek:

$$F_0 = 2^1 + 1 = 3$$

$$F_1 = 2^2 + 1 = 5$$

$$F_2 = 2^4 + 1 = 17$$

$$F_3 = 2^8 + 1 = 257$$

$$F_4 = 2^{16} + 1 = 65537$$

...

Euler 1750-ben megállapította, hogy  $2^{31} - 1$  prímszám. 1876-ban Lucas bebizonyította, hogy  $2^{127} - 1$  prímszám.

Mersenne-prímek: Egy  $2^n - 1$  alakú szám csak akkor lehet prím, ha  $n$  prím. Néhány Mersenne-prím: 3,7,31,127,8191. A Mersenne-képlet szerint összeállított számok között nagyon sok összetett szám van és nem mindegyik állítható elő ilyen alakban. Nyitott kérdés, hogy hány van belőlük. A számítógépek megjelenésével előkerültek a nagy Mersenne-prímek is. Például: 1999-ben Nayan Hajratwala 2098960 számjegyű Mersenne prímet talált. A program egy Pentium 350 MHz-es számítógépen futott 111 napig. A prímszám:  $2^{6972593} - 1$ . 2001-ben Michael Cameron találta meg a Nagy Internetes Mersenne-féle Prímkutatás (GIMPS) keretében 4053946 számjegyű  $2^{13466917} - 1$ . Cameron egy egyszerű otthoni számítógépet használt, mely a programot 45 napig futtatta. 2003-ban Michael Shafer megtalálta a  $2^{20996011} - 1$  prímszámot, mely 6320430 számjegyből áll. A keresőprogram az egyetem 2 GHz-es Pentium 4-es számítógépén futott 19 napig. 2006. szeptember 4-én fedezték fel a 44-edik Mersenne-prímet, ez a  $2^{32\ 582\ 657} - 1$  szám, amely 9 808 358 számjegyű. Ez egyben a jelenleg ismert legnagyobb prímszám és ez a GIMPS projekt tizedik prímrekordja.

Dirichlet tétele: Egy olyan számtani sorozatban, amelynek első tagja és különbsége relatív prím, végtelen sok prímszám van. L. Dirichlet nevezetes tétele azt állítja, hogy minden  $a, a+q, a+2q, a+3q, \dots$

számtani sorozatban végtelen sok prím van, feltéve, hogy  $a$  és  $q > 0$  relatív prímelek.

Euler felfedezte, hogy az  $x^2 + x + 41$  polinom csupa prímszámot ad  $x=0,1,2,\dots,39$  esetén. Később azt is felfedezték, hogy az  $x^2 + 79x + 1601$  polinom az  $x=0,1,2,\dots,79$  értékek mindegyikére prímszámot ad.

A prímszámtételt Gauss 15 évesen megsejtette, hogy a prímszámok száma fordítottan arányos a logaritmusával, de igazolni nem tudta sejtését. A tétel analitikus bizonyítását két francia matematikus Hadamard és De la Vallée Poussin nevéhez fűződik, akik 1896-ban egymástól függetlenül, egyidejűleg bizonyította be. Elemi bizonyítást adott Erdős Pál és A. Selberg.

$$\text{Jelölje } \pi(n) \sim \frac{n}{\ln n}, \frac{\pi(n)}{n} \sim \ln n, \lim_{n \rightarrow \infty} \frac{\frac{\pi(n)}{n}}{\ln n} = 1$$

Felvetődött az a kérdés, hogy miként oszlanak el a prímszámok a természetes számok között. 10-ig 4 darab, 100-ig 25 darab, 1000-ig 168 darab, 10000-ig 1239 darab prímszám van.

Tétel: a szomszédos prímszámok közötti hézagok közt előfordulnak tetszés szerinti nagyok.

Faktoriális prímek: Olyan prímszámok, amelyek egy pozitív egész szám faktoriálisánál eggyel nagyobbak, vagy kisebbek.

Sophie-Germain prímek: Olyan prímszámok, amelyekre igaz, hogy a kétszeresüknél eggyel nagyobb szám is prím.

Napjainkban is tart az a verseny, amelyben számítógépes intézetek és azok tudósai vesznek részt, a céljuk az, hogy minél nagyobb prímszámokat találjanak. A fent említett prímszaládok rekordjának birtoklásáért is folyik a küzdelem.

### 13. Feladat

Feladat az elemek szitálására:

Megadhatunk-e 1983 különböző természetes számot, melyek  $\leq 10000$ , és ahol semelyik három nem alkot számtani sorozatot?

$a < b < c$  illetve számtani sorozatot alkotnak, ha  $2b = a + c$  vagy  $c = 2b - a$ . A 2 és 10000 közötti számok közül szitáljuk ki azokat az elemeket, amelyek egy számtani sorozatot alkotnak.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int i, j, k, v, z, s;
int y;
int x[10000];
int a[10000];
main()
{
    for (i=0; i<10000; i++)
        x[i]=1;
    a[0]=0; a[1]=1; i=1; j=1;
```

```

do
{
    for (k=0; k<j;k++)
    {
        v=a[k];
        z=2*i;
        x[z-v]=0;
    }
do
{
    i=i+1;
    y=x[i];
}
while (y!=1);
j=j+1; a[j]=i;
}
while (j!=256);
for (i=0; i<257; i++)
printf("%5d",a[i]);
system("pause");
}

```

A megoldás:

|                                                     |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|-----------------------------------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 0                                                   | 1    | 3    | 4    | 9    | 10   | 12   | 13   | 27   | 28   | 30   | 31   | 36   | 37   | 39   | 40   |
| 81                                                  | 82   | 84   | 85   | 90   | 91   | 93   | 94   | 108  | 109  | 111  | 112  | 117  | 118  | 120  | 121  |
| 243                                                 | 244  | 246  | 247  | 252  | 253  | 255  | 256  | 270  | 271  | 273  | 274  | 279  | 280  | 282  | 283  |
| 324                                                 | 325  | 327  | 328  | 333  | 334  | 336  | 337  | 351  | 352  | 354  | 355  | 360  | 361  | 363  | 364  |
| 729                                                 | 730  | 732  | 733  | 738  | 739  | 741  | 742  | 756  | 757  | 759  | 760  | 765  | 766  | 768  | 769  |
| 810                                                 | 811  | 813  | 814  | 819  | 820  | 822  | 823  | 837  | 838  | 840  | 841  | 846  | 847  | 849  | 850  |
| 972                                                 | 973  | 975  | 976  | 981  | 982  | 984  | 985  | 999  | 1000 | 1002 | 1003 | 1008 | 1009 | 1011 | 1012 |
| 1053                                                | 1054 | 1056 | 1057 | 1062 | 1063 | 1065 | 1066 | 1080 | 1081 | 1083 | 1084 | 1089 | 1090 | 1092 | 1093 |
| 2187                                                | 2188 | 2190 | 2191 | 2196 | 2197 | 2199 | 2200 | 2214 | 2215 | 2217 | 2218 | 2223 | 2224 | 2226 | 2227 |
| 2268                                                | 2269 | 2271 | 2272 | 2277 | 2278 | 2280 | 2281 | 2295 | 2296 | 2298 | 2299 | 2304 | 2305 | 2307 | 2308 |
| 2430                                                | 2431 | 2433 | 2434 | 2439 | 2440 | 2442 | 2443 | 2457 | 2458 | 2460 | 2461 | 2466 | 2467 | 2469 | 2470 |
| 2511                                                | 2512 | 2514 | 2515 | 2520 | 2521 | 2523 | 2524 | 2538 | 2539 | 2541 | 2542 | 2547 | 2548 | 2550 | 2551 |
| 2916                                                | 2917 | 2919 | 2920 | 2925 | 2926 | 2928 | 2929 | 2943 | 2944 | 2946 | 2947 | 2952 | 2953 | 2955 | 2956 |
| 2997                                                | 2998 | 3000 | 3001 | 3006 | 3007 | 3009 | 3010 | 3024 | 3025 | 3027 | 3028 | 3033 | 3034 | 3036 | 3037 |
| 3159                                                | 3160 | 3162 | 3163 | 3168 | 3169 | 3171 | 3172 | 3186 | 3187 | 3189 | 3190 | 3195 | 3196 | 3198 | 3199 |
| 3240                                                | 3241 | 3243 | 3244 | 3249 | 3250 | 3252 | 3253 | 3267 | 3268 | 3270 | 3271 | 3276 | 3277 | 3279 | 3280 |
| 6561A folytatáshoz nyomjon meg egy billentyűt . . . |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |

### Feladat egy adott intervallumban lévő prímszámok vizsgálatára

Egy olyan programot szeretnénk írni, amely azt vizsgálja, hogy egy  $n$  szám prímszám.

A következő program két páratlan  $a$  és  $b$  számot kér, amelyek  $1 < a < b$  feltételnek megfelelnek, majd kiírja azt az számot, ahány prímszám található az  $[a, b]$  intervallumban.

```
#include <stdio.h>
#include <stdlib.h>
#define TRUE 1
#define FALSE 0
int a,b,x,szam;
int prim(int n,int d)
{
    if (d*d>n) return (TRUE);
    else if (n%d==0) return (FALSE);
    else return prim(n,d+2);
}
main()
{
    printf ("Két páratlan számot kérek, ahol a<b\n a,b:");
    scanf ("%d,%d",&a,&b);
    x=a;
    szam=0;
    do
    {
        if (prim(x,3))
            szam=szam+1;
        x=x+2;
    }
    while (x<=b);
    printf ("primek száma az [a,b] intervallumban: %d\n",szam);
    system ("pause");
}
```

Ezzel a programmal megvizsgálhatjuk azt is, hogy hány prímszám található 10000, 30000, 100000, 1000000 után. Nézzük meg, százasával a prímek darabszámát.

Pl: [10001,10099] intervallumban 11 darab prímszám található, [10101,10199] intervallumban 12 darab prímszám található. Ennek megfelelően:

10000 után: 11, 12, 10, 12, 10, 8, 12, 11, 10, 10. Átlag: 10.6

30000 után: 9, 12, 9, 9, 9, 9, 9, 8, 13, 8. Átlag: 9.5

100000 után: 6, 9, 8, 9, 8, 10, 8, 7, 6, 10. Átlag: 8.1

1000000 után: 6, 10, 8, 8, 7, 7, 10, 5, 6, 8. Átlag: 7.5

Egy  $a, b$  intervallumban  $\int_a^b \frac{1}{\ln x} dx = \frac{b-a}{\ln \frac{a+b}{2}}$  prímszám várható.

A feladatnál  $b-a=98$ . Ebből következően a várható értékek:

$$\frac{98}{\ln 10500} = 10.6, \quad \frac{98}{\ln 30500} = 9.5, \quad \frac{98}{\ln 100500} = 8.5, \quad \frac{98}{\ln 1000500} = 7.1$$

## 14. Ikerprímek, prímhármasok, prímnégyesek

Már az ókorban megfigyelték, hogy a prímek gyakran párokban fordulnak elő, mint például a 11 és a 13, a 29 és a 31 vagy az 59 és a 61, ezeket ikerprímeknek nevezik. A prímek gyakran csoportosan is felbukkannak, mint például a 101, 103, 107, 109, 113 sorozat esetén.

Régóta gyanították azt is, hogy ez a jellegzetesség az egész számok körében folyamatosan ismétlődik, azonban ezt eddig nem sikerült bizonyítani. Bizonyíték hiányában viszont feltételezhető volt az is, hogy kellően nagy számok esetén már nem fordulnak elő sem ikerprímek, sem prímcsoportok. Ha két prímszám különbsége 2, akkor azokat ikerprímszámoknak nevezzük. Általános formája:  $(p, p+2)$

### *Ikerprímek 2000-ig*

(3, 5), (5, 7), (11, 13), (17, 19), (29, 31), (41, 43), (59, 61), (71, 73), (101, 103), (107, 109), (137, 139), (149, 151), (179, 181), (191, 193), (197, 199), (227, 229), (239, 241), (269, 271), (281, 283), (311, 313), (347, 349), (419, 421), (431, 433), (461, 463), (521, 523), (569, 571), (599, 601), (617, 619), (641, 643), (659, 661), (809, 811), (821, 823), (827, 829), (857, 859), (881, 883), (1019, 1021), (1031, 1033), (1049, 1051), (1061, 1063), (1091, 1093), (1151,

1153), (1229, 1231), (1277, 1279), (1289, 1291), (1301, 1303), (1319, 1321), (1427, 1429), (1451, 1453), (1481, 1483), (1487, 1489), (1607, 1609), (1619, 1621), (1667, 1669), (1697, 1699), (1721, 1723), (1787, 1789), (1871, 1873), (1877, 1879), (1931, 1933), (1949, 1951), (1997, 1999)

***A legnagyobb ismert ikerprímek:***

$100314512544015 \cdot 2^{171960} \pm 1$  (2006. június 20. – Csajbók Tímea, dr. Farkas Gábor, dr. Járai Antal, Járai Zoltán, Kasza János)

$2003663613 \cdot 2^{195000} \pm 1$  (2007. január 15. – Eric Vautier, Franciaország, a Twin Prime Search (TPS) projekt és a PrimeGrid (BOINC platform) segítségével)

A prímszámok sorozatában elég messze menve, két szomszédos prímszám közötti hézag tetszés szerinti nagy lehet, így ritkán helyezkednek el a prímek között. A számelméletnek nem megoldott problémája, hogy az ikerprímszámok száma véges-e vagy végtelen, ebből következően verseny folyik az aktuálisan legnagyobb ikerprím számpár megtalálásáért, és találnak is elég gyakran elég nagy ikerprímeket. A prímek reciprokösszege divergens, ami azt jelenti, hogy a prímek reciprocai „lassan” fogynak, azaz maguk a prímek lassan növekednek, vagyis a prímek viszonylag sűrűn helyezkednek el a pozitív egészek között. Az ikerprímszámok reciprokaiból képzett sor konvergens, azaz véges összeghez tart. Ez azt mutatja, hogy az ikerprímszámok, ha egyáltalán van belőlük végtelen sok, viszonylag ritkán fordulnak elő.

A következő program két számot kér, és kiírja a közöttük levő ikerprímeket:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define TRUE 1
#define FALSE 0
int a,b,x;
int prim(int n,int d)
{
    if (d*d>n) return (TRUE);
    else if (n%d==0) return (FALSE);
```

```

        else return prim(n,d+2);
    }
main()
{
    printf ("a,b="); scanf ("%d, %d",&a, &b);
    x=6*trunc(a/6)+5;
    do
    {
        if (prim(x,5))
            if (prim(x+2,5))
                printf ("%d  %d\n ", x, x+2);
        x=x+6;
    }
    while (x<=b-2);
    system ("pause");
}

```

Nézzük meg, hogy a=1000000 és b=1001000 között hány ikerprím található.

Figyeljük, meg, hogyan oszlanak el az ikerprímek.

|                 |                                    |
|-----------------|------------------------------------|
| 1000000-1000100 | 1000037,1000039                    |
| 1000101-1000200 | -                                  |
| 1000201-1000300 | 1000211, 1000213; 1000289, 1000291 |
| 1000301-1000400 | -                                  |
| 1000401-1000500 | 1000427,1000429                    |
| 1000501-1000600 | 1000577,1000579                    |
| 1000601-1000700 | 1000619,1000621;1000667,1000669    |
| 1000701-1000800 | 1000721,1000723                    |
| 1000801-1000900 | 1000847,1000849;1000859,1000861    |
| 1000901-1001000 | 1000919,1000921                    |

Láthatjuk, hogy a második és a negyedik rész üres. Megfogalmazhatunk ezzel kapcsolatban egy másik feladatot is, így:

Dobjunk 11 golyót 10 rekeszbe. Hány rekesz marad üresen? Valamelyik cella  $0.9^{11}$  valószínűséggel marad üresen. Az összesre vonatkoztatva  $10 \cdot 0.9^{11} = 3.138$  cella várható üresnek.

### ***Prímhármaskok 2000-ig***

5,7,11 ; 7,11,13 ; 11,13,17 ; 13,17,19 ; 17,19,23 ; 37,41,43 ; 41,43,47 ; 67,71,73 ; 97,101,103 ;  
101,103,107 ; 103,107,109 ; 107,109,113 ; 191,193,197 ; 193,197,199 ; 223,227,229 ;  
227,229,233 ; 277,281,283 ; 307,311,313 ; 311,313,317 ; 347,349,353 ; 457,461,463 ;  
461,463,467 ; 613,617,619 ; 641,643,647 ; 821,823,827 ; 823,827,829 ; 853,857,859 ;  
857,859,863 ; 877,881,883 ; 881,883,887 ; 1087,1091,1093 ; 1091,1093,1097 ;  
1277,1279,1283 ; 1297,1301,1303 ; 1301,1303,1307 ; 1423,1427,1429 ; 1427,1429,1433 ;  
1447,1451,1453 ; 1481,1483,1487 ; 1483,1487,1489 ; 1487,1489,1493 ; 1607,1609,1613 ;  
1663,1667,1669 ; 1693,1697,1699 ; 1783,1787,1789 ; 1867,1871,1873 ; 1871,1873,1877 ;  
1873,1877,1879 ; 1993,1997,1999 ;

A prímhármaskok felírásának általános formája: (p, p+2, p+6) illetve (p, p+4, p+6)

Prímhármaskok (p, p+2, p+6) alapján:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define TRUE 1
#define FALSE 0
int a,b,p;
int prim(int n,int d)
{
    if (d*d>n) return (TRUE);
    else if (n%d==0) return (FALSE);
    else return prim(n,d+2);
}
main()
{
    printf ("a,b="); scanf ("%d, %d",&a, &b);
```

```

p=6*trunc(a/6)+5;
do {
    if (prim(p,3))
    if (prim(p+2,3))
    if (prim(p+6,3))
    printf("%d      %d      %d\n",p,p+2,p+6);
    p=p+6;
}
while (p<=b);
system ("pause");
}

```

**A prímhármasok felírásának általános formája: (p, p+4, p+6)**

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define TRUE 1
#define FALSE 0
int a,b,p;
int prim(int n,int d)
{
    if (d*d>n) return (TRUE);
    else if (n%d==0) return (FALSE);
    else return prim(n,d+2);
}
main()
{
    printf ("a,b="); scanf ("%d, %d",&a, &b);
    p=6*trunc(a/6)+5;
    do {
        if (prim(p,3))
        if (prim(p+4,3))
        if (prim(p+6,3))

```

```

        printf ("%d      %d      %d\n", p, p+4, p+6);
        p=p+4;
    }
    while (p<=b);
system ("pause");
}

```

**A megmaradt prímhármasok kiíratása a következő programmal történik:**

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define TRUE 1
#define FALSE 0
int a,b,p;
int prim(int n,int d)
{
    if (d*d>n) return (TRUE);
    else if (n%d==0) return (FALSE);
    else return prim(n,d+2);
}
main()
{
    printf ("a,b="); scanf ("%d, %d",&a, &b);
    p=3*trunc(a/2)+0;
    do {
        if (prim(p,3))
            if (prim(p+4,3))
                if (prim(p+6,3))
                    printf ("%d      %d      %d\n", p, p+4, p+6);
        p=p+4;
    }
    while (p<=b);
    system ("pause");
}

```

```
}
```

### ***Prímnégyesek 2000-ig***

5,7,11,13;    11,13,17,19;    101,103,107,109;    191,193,197,199;    821,823,827,829;  
1481,1483,1487,1489;    1871,1873,1877,1879;

Észrevehető, hogy a prímnégyesek 11-től kezdve 1, 3, 7, 9-re végződnek. Írjunk programot ami előállítja:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define TRUE 1
#define FALSE 0
int a,b,p;
int prim(int n,int d)
{
    if (d*d>n) return (TRUE);
    else if (n%d==0) return (FALSE);
    else return prim(n,d+2);
}
main()
{
    printf ("a,b="); scanf ("%d, %d",&a, &b);
    p=6*trunc(a/6)+5;
    do {
        if (prim(p,3))
            if (prim(p+2,3))
                if (prim(p+6,3))
                    if (prim(p+8,3))
                        printf("%d      %d      %d      %d\n",p,p+2,p+6,p+8);
        p=p+6;
    }
    while (p<=b);
}
```

```

    system ("pause");
}

```

## 15. Pitagoraszai számhármások

Definíció: Az  $x^2 + y^2 = z^2$  egyenlet pozitív egész megoldásait pitagoraszai számhármásoknak nevezzük. Ha még  $(x, y, z)=1$  is teljesül, akkor a számhármast primitív pitagoraszai számhármásoknak nevezzük. Keressük az  $x^2 + y^2 = z^2$  másodfokú diofantoszi egyenlet megoldásait. Diofantoszi egyenleten olyan egész együtthatós egyenletet értünk, melynek megoldásait is az egész számok körében keressük.

Azonnal látszik, hogy az egyenlet megoldható (például 3, 4, 5 számhármast megoldás), sőt egy olyan  $x, y, z$  megoldást, ahol  $(x, y, z)=1$  tetszőleges  $d$  pozitív egésszel beszorozva a kapott  $dx, dy, dz$  számhármast nyilván megoldás.

Megmutatjuk, hogy az alapmegoldásból is végtelen sok van, sőt elő tudjuk állítani az összes alapmegoldást és így az összes megoldást is alkalmas paraméterek segítségével:

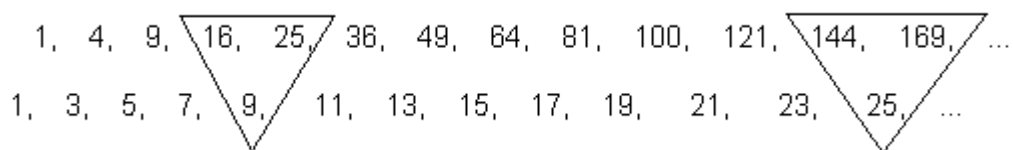
Tétel:  $x^2 + y^2 = z^2$  ( $x, y, z \in \mathbb{N}^+$ ) diofantoszi egyenlet összes megoldását az úgynevezett pitagoraszai számhármások adják. A számhármások alakja:

$x = m^2 - n^2, y = 2mn, z = m^2 + n^2$ , vagy  $x = 2mn, y = m^2 - n^2$  és  $z = m^2 + n^2$ , ahol  $(m, n) = 1$  és  $m > n$ .

Az alapmegoldások többszörösei pedig:  $x = (m^2 - n^2)d, y = 2mnd, z = (m^2 + n^2)d$ , vagy  $x = 2mnd, y = (m^2 - n^2)d, z = (m^2 + n^2)d$ , ahol  $d$  tetszőleges pozitív egész, az  $m$  és  $n$  pozitív egész.

Lemma: Ha két természetes szám felírható két négyzetszám összegeként, akkor a szorzatuk is.

A pitagoreusok találták meg a pitagoraszai számhármások korlátlan mennyiségben való előállításának egy módját. Ha egy sorba a négyzetszámokat és azok alá a páratlan számokat írjuk, akkor az alsó sor minden négyzetszámahoz a felső sor két száma csatlakozik. Ezek hárman pitagoraszai számhármast alkotnak:



Példák ( $n$  tetszőleges pozitív egész):

|     |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |     |     |
|-----|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|-----|
| $y$ | 3 | 8  | 7  | 20 | 12 | 9  | 28 | 11 | 33 | 16 | 48 | 36 | 13 | 39 | 65 | 20  |     |
| $x$ | 4 | 15 | 24 | 21 | 35 | 40 | 45 | 60 | 56 | 63 | 55 | 77 | 84 | 80 | 72 | 99  |     |
| $z$ | 5 | 17 | 25 | 29 | 37 | 41 | 53 | 61 | 65 | 65 | 73 | 85 | 85 | 89 | 97 | 101 | ... |

A Pitagorasz-tétel értelmében  $x, y, z$  egy derékszögű háromszög oldalai.

Pitagorasz és követői végtelen sok primitív pitagorasz számhármast adtak meg a következő módon:

$$x=n, y=\frac{n^2-1}{2} \text{ és } z=\frac{n^2+1}{2}, \text{ ahol } n=3,5,7,9,\dots$$

Platon a pitagorasz számhármast a következő módon állította elő:

$$x=n, y=\left(\frac{n}{2}\right)^2 - 1 \text{ és } z=\left(\frac{n}{2}\right)^2 + 1, \text{ ahol } n=4,6,8,10,\dots$$

Írjunk olyan programot, amely egy maximum értéket kér, és a  $max=z$  értékig kilistázza azokat az  $(x, y, z)$  számhármastokat amelyek eleget tesznek az  $x^2 + y^2 = z^2$  egyenletnek.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int ggt(int a, int b)
{
    if (b==0)
        return a;
    else return ggt(b, a%b);
}
int z, n, x, y, max;
main()
{
    printf ("max="); scanf ("%d", &max);
    z=1;
    while (z<=max)
```

```

{
    z=z+1; n=z*z;
    x=trunc(sqrt(n/2)); y=x;
    do
    {
        while (x*x+y*y>n)
            y=y-1;
        if (x*x+y*y<n) x=x+1;
        else
        {
            if (ggt(x,y)==1)
                printf("y=%4d x=%4d z=%5d\n", y, x, z);
            x=x+1; y=y+1;
        }
    }
    while (x*x<=n);
}
system ("pause");
}

```

## 16. Két négyzetszám tétel

Az  $x^2 + y^2 = n$  ( $n$  természetes szám) egyenlet megoldhatósága azt jelenti, hogy  $n$  természetes szám felbomlik két négyzetszám összegére.

Tétel:  $p=4k+1$  és  $p$  prím, úgy  $p$  előáll két négyzetszám összegeként.

A  $4n-1$  alakúak viszont soha nem állíthatók elő két négyzetszám összegeként.

Eulertől származik a következő eredmény: pontosan azok a számok bonthatók fel két négyzetszám összegére, amelyek prímfelbontásában  $4k-1$  alakú prímszám nem fordul elő páratlan hatványon.

Mely számok állíthatók elő két négyzetszám összegeként?

Példa:

$$1 = 1^2 + 0^2$$

$$2 = 1^2 + 1^2$$

3 nem állítható elő

$$4 = 2^2 + 0^2$$

$$5 = 1^2 + 2^2$$

6 nem állítható elő

7 nem állítható elő

$$8 = 2^2 + 2^2$$

$$9 = 3^2 + 0^2$$

$$10 = 3^2 + 1^2$$

11 nem állítható elő

...

A következő program egy természetes számot kér, majd kiírja, hogy mely két szám négyzetének összege. A második megoldás felhasználja a *trunc()* és az *sqrt()* függvényeket.

```
#include <stdio.h>
#include <stdlib.h>
int n,x,y;
main()
{
    printf ("n="); scanf ("%d", &n);
    x=0; y=0;
    do
    {
        x=x+1; y=y+1;
    }
    while (x*x+y*y<n);
    do
    {
```

```

while (x*x+y*y>n)
    y=y-1;
if (x*x+y*y<n) x=x+1;
else
{
    printf("x= %d\n",x);
    printf("y= %d\n",y);
    x=x+1;
}
}
while (x*x<=n);
system ("pause");
}

```

**Második megoldás:**

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int n,x,y;
main()
{
    printf ("n="); scanf ("%d", &n);
    x=trunc(sqrt(n/2)); y=x;
    do
    {
        while (x*x+y*y>n)
            y=y-1;
        if (x*x+y*y<n) x=x+1;
        else
        {
            printf("x= %d\n",x);
            printf("y= %d\n",y);
            x=x+1;y=y-1;
        }
    }
}

```

```

    }
}
while (x*x<=n);
system ("pause");
}

```

## 17. Négy négyzetszám probléma

A tételt 1621-ben Bachet sejtette meg, és  $n=1\dots 325$ -ig ellenőrizte, majd Lagrange igazolta 1770-ben. Ezek szerint, az  $x^2 + y^2 + z^2 + t^2 = n$  diofantoszi egyenlet mindig megoldható. Ez más szóval azt jelenti, hogy minden természetes szám előáll négy négyzetszám összegeként.

Például:  $31=5^2 + 2^2 + 1^2 + 1^2 = 3^2 + 3^2 + 3^2 + 2^2$

$37=6^2 + 1^2 (+0^2 + 0^2) = 5^2 + 2^2 + 2^2 + 2^2 = 4^2 + 4^2 + 2^2 + 1^2$

$67=8^2 + 1^2 + 1^2 + 1^2 = 7^2 + 4^2 + 1^2 + 1^2 = 7^2 + 3^2 + 3^2 (+0) = 5^2 + 5^2 + 4^2 + 1^2$

$103=10^2 + 1^2 + 1^2 + 1^2 = 9^2 + 3^2 + 3^2 + 2^2 = 7^2 + 7^2 + 2^2 + 1^2 = 7^2 + 6^2 + 3^2 + 3^2 =$   
 $= 7^2 + 5^2 + 5^2 + 2^2$

Írjunk olyan programot, amely egy természetes számot kér, majd megadja az összes előállítását annak, hogy mely négy szám négyzetének összegeként áll elő.

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int x,y,z,u,n,r;
main ()
{
    printf ("n="); scanf ("%d", &n);
    r=trunc(sqrt(n));
    for (x=r; x>=r/2; x--)
        for (y=x; y>=0; y--)
            for (z=y; z>=0; z--)
                for (u=z; u>=0; u--)

```

```

if (x*x+y*y+z*z+u*u==n)
printf("%d      %d      %d      %d\n", x, y, z, u);
system ("pause");
}

```

Ha  $x^2 + y^2 + z^2 + t^2 = n$  egyenletben szereplő 2-es kitevő helyett 4-es kitevőt tekintünk, akkor a következő tétel igaz: minden természetes szám előállítható legfeljebb 53 negyedik hatvány összegeként.

## 18. Fermat-sejtés

Fermat a 17. század első felében élő francia matematikus az  $x^2 + y^2 = z^2$  diofantoszi egyenlet általánosításával, az  $x^n + y^n = z^n$  diofantoszi egyenlettel kapcsolatban egy könyv szélére a következőt írta: „Az  $n > 2$  estben nincsenek olyan zérustól különböző számok, amelyek ezt az egyenletet kielégítenék. Ennek egy csodálatosan szép bizonyítását találtam, azonban a lap széle kevésnek bizonyult, hogy azt befogadja.” Ez a „bizonyítás” azonban nem maradt fenn. Azóta igen sokan fáradoztak azon, hogy Fermatnak ezt a sejtését bebizonyítsák.

Fermat igazolta, hogy  $n=4$  esetén igaz a sejtés, később  $n=3$ -ra Euler adta meg a megoldást. A sejtést végül Andrew Wiles amerikai matematikus 1995-ben bizonyította, hogy  $p < 2 \cdot 10^6$  és  $p$  prím, akkor ilyen  $p$  –k esetén igaz a sejtés, majd ugyancsak ő bizonyította az állítást tetszőleges  $n$ -re 1996-ban.

## 19. Számelméleti függvények

Definíció: A pozitív egészek halmazán értelmezett valós vagy komplex értékű függvényt számelméleti függvénynek nevezzük.

$N(n)$ :  $n$  pozitív osztóinak száma. Pl: 6 pozitív osztói 1, 2, 3, 6, így  $N(6)=4$ , 120 pozitív osztói 1, 2, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60, 120.  $N(120)=15$

$S(n)$ :  $n$  pozitív osztóinak összege. Pl: 6 pozitív osztóinak összege  $1+2+3+6=12$ , így  $S(6)=12$

$S(20)=1+2+4+5+10+20=42$

$\pi(x)$ -szel az  $x$  valós számnál nem nagyobb prímszámok számát jelöli.

Például:  $\pi(20) = 8$ , mert 20-ig 8 darab prímszám található, ezek pedig: 2,3,5,7,11,13,17,19.

$\lim_{x \rightarrow \infty} \frac{\pi(x)}{\frac{x}{\ln x}}$  határérték létezik, s ez 1, azaz  $\pi(x)$  és  $\frac{x}{\ln x}$  aszimptotikusan egyenlők.

Csebisev bizonyította be először a következő tételt:

$c_1 \frac{x}{\ln x} < \pi(x) < c_2 \frac{x}{\ln x}$ . A tétel azt fejezi ki, hogy  $\pi(x)$  körülbelül úgy nő, mint  $\frac{x}{\ln x}$ .

### Az Euler-féle $\varphi$ függvény

$\varphi(n)$ : az Euler-féle  $\varphi$ -függvény jelöli az  $1, 2, 3, \dots, n$  számok közül az  $n$ -hez relatív prímek számát. Pl: 12-höz a nála kisebb pozitív egészek közül relatív prímek az 1, 5, 7, 11  $\varphi(12) = 4$ ,  $\varphi(11) = 10$ , mivel az 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 számok mindegyike relatív prím a 11-hez.

$\varphi(n) = n - 1 \Leftrightarrow n$  prím. Az is belátható, hogy  $\varphi(p^k) = p^k \left(1 - \frac{1}{p}\right)$ .

A  $\varphi(n)$  számelméleti függvény, és az előbb említett  $N(n)$ ,  $S(n)$  számelméleti függvények mind multiplikatívak. Definíció szerint ez azt jelenti, hogy ha  $(a, b) = 1$ , mindannyiszor  $\varphi(a \cdot b) = \varphi(a)\varphi(b)$ .

Ha  $n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \dots \cdot p_k^{\alpha_k} = \prod_{i=1}^k p_i^{\alpha_i}$ , ahol  $\alpha_i > 0$ , akkor

$$\varphi(n) = n \prod_{i=1}^k \left(1 - \frac{1}{p_i}\right) = n \prod_{p|n} \left(1 - \frac{1}{p}\right)$$

Tétel: Ha  $n = p_1^{\alpha_1} \dots p_k^{\alpha_k}$ , akkor

$$N(n) = (a_1 + 1)(a_2 + 1) \dots (a_k + 1)$$

$$S(n) = \frac{p_1^{\alpha_1+1} - 1}{p_1 - 1} \cdot \frac{p_2^{\alpha_2+1} - 1}{p_2 - 1} \cdot \dots \cdot \frac{p_k^{\alpha_k+1} - 1}{p_k - 1}$$

$$\varphi(n) = n \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \dots \left(1 - \frac{1}{p_k}\right)$$

A következő program felsorolja az általunk megadott számhoz a relatív prímeket, és megadja  $\varphi$  értékét.

```

#include <stdio.h>
#include <stdlib.h>
int lkko(int a, int b)
{
    if (b==0)
        return a; else return lkko(b, a%b);
}
main()
{
    int i, n, fi;
    printf ("n="); scanf ("%d", &n);
    for (i=1; i<n; i++)
        if (lkko(i, n) == 1)
            printf("%d\n", i);
    fi=0;
    for (i=1; i<n; i++)
        if (lkko(i, n) == 1)
            fi=fi+1;
    printf("fi: %d\n", fi);
    system ("pause");
}

```

### Möbius-függvény

$\mu(n)$ , a számok „négyzetmentességét” mérő függvény, minden pozitív egészre definiált. Értéke az  $\{-1, 0, 1\}$  halmazból kerül ki, mely  $n$   $p$  prímfelbontásától függ.

$\mu(n)=1$  ha  $n$  négyzetmentes, és a prímtényezők száma páros.

$\mu(n)=-1$  ha  $n$  négyzetmentes, és a prímtényezők száma páratlan.

$\mu(n)=0$ , ha  $n$  nem négyzetmentes

$$\mu(1)=1$$

Egy számot négyzetmentesnek nevezünk, ha a prímtényező felbontásban minden prím kitevője 1, vagyis a szám nem osztható négyzetszámmal.

Például:

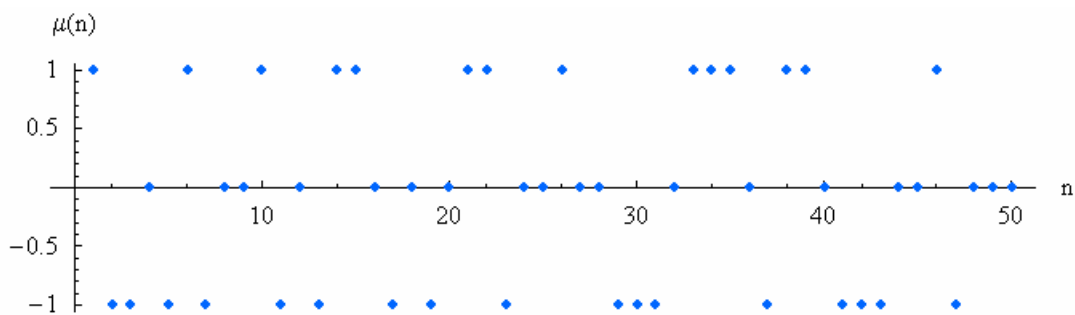
$$6 = 2 \cdot 3$$

$$30 = 2 \cdot 3 \cdot 5$$

$$154 = 2 \cdot 7 \cdot 11$$

Ellenpélda:  $3^2 \nmid 18$

|               |                                                  |
|---------------|--------------------------------------------------|
| $\mu(n) = 1$  | 1, 6, 10, 14, 15, 21, 22, 26, 33, 34, 35, ...    |
| $\mu(n) = -1$ | 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 30, 31, ...  |
| $\mu(n) = 0$  | 4, 8, 9, 12, 16, 18, 20, 24, 25, 27, 28, 32, ... |



$$\sum_{d|n} \mu(d) = 1, \text{ ha } n=1$$

$$\sum_{d|n} \mu(d) = 0, \text{ ha } n>1$$

A program 500-tól 15000-ig ötszázásával kér egy számot, majd az általunk megadott számig kiírja, hogy milyen gyakorisággal találhatók meg azok a számok, melyek négyzetmentesek.

```
#include <stdlib.h>
#include <math.h>
int i,k,m,n,s;
int x[15000];
main()
{
    printf ("n="); scanf ("%d", &n);
```

```

m=trunc(sqrt(n)); s=0;
for (i=1; i<n+1; i++)
x[i]=1;
for (i=2; i<m+1; i++)
{
    k=0;
    do
    {
        k=k+i*i;
        if (k<=n)
            x[k]=0;
    }
    while (k<n);
}
for (i=1; i<n+1; i++)
{
    s=s+x[i];
    if ((i%500)==0)
        printf ("i= %5d s/i= %10.5lf\n", i, (double) s/i);
}
system ("pause");
}

```

A program futásának eredménye:

|      |         |      |         |       |         |
|------|---------|------|---------|-------|---------|
| 500  | 0.61200 | 5500 | 0.60764 | 10500 | 0.60810 |
| 1000 | 0.60800 | 6000 | 0.60767 | 11000 | 0.60791 |
| 1500 | 0.61000 | 6500 | 0.60800 | 11500 | 0.60783 |
| 2000 | 0.60750 | 7000 | 0.60786 | 12000 | 0.60775 |
| 2500 | 0.60920 | 7500 | 0.60813 | 12500 | 0.60768 |
| 3000 | 0.60800 | 8000 | 0.60812 | 13000 | 0.60777 |

|      |         |       |         |       |         |
|------|---------|-------|---------|-------|---------|
| 3500 | 0.60914 | 8500  | 0.60812 | 13500 | 0.60770 |
| 4000 | 0.60825 | 9000  | 0.60811 | 14000 | 0.60779 |
| 4500 | 0.60800 | 9500  | 0.60842 | 14500 | 0.60772 |
| 5000 | 0.60840 | 10000 | 0.60830 | 15000 | 0.60800 |

## 20. Tökéletes számok

Definíció: Tökéletes számnak nevezzük azt a természetes számot, amelynek a nála kisebb pozitív osztóinak összege maga a szám.

Már a görögök is ismertek néhány tökéletes számot, és ezeket a harmónia jelképének tekintették. Az általuk ismert tökéletes számok:

$$6=1+2+3$$

$$28=1+2+3+4+7+14$$

$$496=1+2+4+8+16+31+62+124+248$$

$$8128=1+2+4+8+16+32+64+127+254+508+1016+2032+4064$$

Euklidesz is említi az Elemek című könyvében, hogy ha egy  $n$  természetes szám felírható  $n = 2^{p-1}(2^p - 1)$  alakban, ahol  $p$  és  $2^p - 1$  prímek, akkor  $n$  tökéletes szám.

$$\text{Példa: } 6=2^{2-1}(2^2-1)=2 \cdot 3$$

$$28=2^{3-1}(2^3-1)=4 \cdot 7$$

$$496=2^{5-1}(2^5-1)=16 \cdot 31$$

$$8128=2^{7-1}(2^7-1)=64 \cdot 127$$

Eukleidész Kr.e. 300 körül tudta, hogy  $k$  természetes szám esetén, ha  $2^{k+1} - 1$  törzsszám, akkor  $2^k (2^{k+1} - 1)$  tökéletes szám. Euler (1707-1783) kimutatta, hogy fordítva is így van, azaz az összes páros tökéletes szám,  $2^k (2^{k+1} - 1)$  alakú.

Válasszunk egy páros tökéletes számot és adjuk össze osztóinak reciprokait.

$$6 \text{ osztói: } 1, 2, 3, 6. \text{ A reiprokok összege: } 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{6} = 2$$

28 osztói: 1,2,4,7,14,28. A reciprok összege:

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{7} + \frac{1}{14} + \frac{1}{28} = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{7} \cdot \left(1 + \frac{1}{2} + \frac{1}{4}\right) = \frac{7}{4} + \frac{1}{7} \cdot \frac{7}{4} = \frac{7}{4} + \frac{1}{4} = 2$$

Általánosítva:

$$p = 1 + 2 + 2^2 + \dots + 2^{n-1} = 2^n - 1$$

$$\begin{aligned} 1 + \frac{1}{2} + \frac{1}{2^2} + \dots + \frac{1}{2^{n-1}} + \frac{1}{p} \cdot \left(1 + \frac{1}{2} + \frac{1}{2^2} + \dots + \frac{1}{2^{n-1}}\right) &= \frac{2^{n-1} + 2^{n-2} + \dots + 2 + 1}{2^{n-1}} \cdot \left(1 + \frac{1}{p}\right) = \\ &= \frac{2^n - 1}{2^{n-1}} \cdot \frac{p + 1}{p} = \frac{2^n - 1}{2^{n-1}} \cdot \frac{2^n - 1 + 1}{2^n - 1} = 2 \end{aligned}$$

Tehát a tökéletes szám osztóinak reciprokát összeadva mindig 2-t kapunk. Modern számítógépekkel a XX. században már egyre nagyobb tökéletes számokat sikerült előállítani. Azonban nyitott kérdés, hogy vannak-e páratlan tökéletes számok.

## 21. Nevezetes lineáris felbontási problémák

### 21.1. Partíció probléma

Hányféleképpen bontható fel egy természetes szám más természetes számok összegére?

$$m_1 + m_2 + \dots + m_k = m \quad (m, m_i \in \mathbb{N})$$

Legyen  $p(n)$  egy adott természetes szám különböző felbontásainak összege. Például:

$$p(3) = 3, \text{ mert } 3 = 2 + 1 = 1 + 1 + 1$$

$$p(5) = 7, \text{ mert } 5 = 4 + 1 = 3 + 2 = 3 + 1 + 1 = 2 + 2 + 1 = 2 + 1 + 1 + 1 = 1 + 1 + 1 + 1 + 1,$$

$$p(127) = 3913864295$$

$$p(138) = 12292341831$$

$$p(149) = 37027355200$$

$$p(159) = 97662728555$$

Programozása C-ben:

```
#include <stdio.h>
#include <math.h>
#define N 200
```

```

int i,j,n;
double p[N];
main()
{
    printf ("n="); scanf ("%d", &n);
    for (i=0; i<n+1; i++)
        p[i]=1;
    for (i=2; i<n+1; i++)
        for (j=i; j<n+1; j++)
            (double) p[j]=(double) p[j]+ (double) p[j-i];
    printf("p(%d)= %lf\n",n, (double) p[n]);

    system("pause");
}

```

## 21.2. Pénzváltási probléma

Általánosított partíció probléma. Hányféle módon tudunk címletezni? A rendelkezésre álló pénzérték legyenek 1, 2, 5, 10, 20, 50, 100, amit így is jelölhetnénk: és  $d[1]=\{1\}$ ,  $d[2]=\{1,2\}$ ,  $d[3]=\{1,2,5\}$  ...

Pl: 8 Ft-ot akarunk 1, 2, 5 forintosokkal kifizetni. Hányféleképpen tehetjük meg?

$8=1+1+1+1+1+1+1+1=2+1+1+1+1+1+1=2+2+1+1+1+1=2+2+2+1+1=2+2+2+2=5+2+1=5+1+1+1$

A programban:  $n=8$ ,  $k=3$ . A lehetőségek száma: 7.

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
const int d[7]={1,2,5,10,20,50,100};
int a(int n, int k)
{

```

```

    if (n<0) return (0);
    else if (k==1) return (1);
    else return (a(n,k-1)+a(n-d[k-1],k));
}
main()
{
    int i,k,n;
    printf ("n,k:"); scanf ("%d, %d", &n, &k);
    printf ("Lehetőségek= %d\n", a(n,k));
    system("pause");
}

```

Az alábbi szabályszerűségeket vehetjük észre:

1.  $a(n,1)=1$
2.  $a(n,k)=0$  ha  $n<0$
3.  $a(n,k)=a(n, k-1)+a(n-d[k], k)$

Például:

1.  $(10,1)=1$
2.  $(-8,2)=0$
3.  $(8,3)=(8,2)+(3,3)$  ami nem más, mint  $7=5+2$

$$(15,3)=(15,2)+(10,3) \text{ az } 18=8+10$$

### 21.3. A Goldbach-sejtés

$4=2+2$ ,  $6=3+3$ ,  $8=5+3$ ,  $10=7+3$ ,  $12=7+5$ ,... A kérdés az, hogy felírható-e 4-től kezdve minden páros szám két prímszám összegeként?

A Goldbach-sejtés azt mondja ki, hogy

(I.) Minden 2-nél nagyobb páros szám előáll két prímszám összegeként.

(II.) Minden 5-nél nagyobb páratlan szám előáll három prímszám összegeként.

Ezt a sejtést a német Goldbach fogalmazta meg először 1742-ben Leonhard EULER svájci matematikushoz írt levelében, melyben azt állította, hogy minden páros szám előállítható két prímszám összegeként, és minden 6-nál nem kisebb természetes szám kifejezhető három prímszám összegeként. A válaszában Euler megjegyezte, hogy az állítás igazolásához elég

lenne belátni azt, hogy minden páros szám felbontható két prímszám összegére. Ez utóbbi állítás azonnal következik a „páros” Goldbach-sejtésből. Később részeredmények is születtek, így 1937-ben Vinogradov orosz matematikusként sikerült részben bebizonyítani a „páratlan” Goldbach-sejtést.

A következő programban azt a két prímszámot keressük, melyek összegéből előáll az általunk megadott  $g > 6$  páros szám. Arra is kíváncsiak vagyunk, hogy hányféleképpen állíthatjuk elő a  $g$  számot. Ezt az összeget a program „Goldbachszám” néven adja meg.

```
#include <stdio.h>
#include <stdlib.h>
#define TRUE 1
#define FALSE 0
int g,x,szam;
int prim(int n,int d)
{
    if (g<6)
        return 0;
    if (g%2!=0) return 0;
    if (d*d>n) return (TRUE);
    else if (n%d==0) return (FALSE);
    else return prim(n,d+2);
}
main()
{
    printf ("g="); scanf ("%d",&g);
    x=3; szam=0;
    do
    {
        if (prim(x,3))
            if (prim(g-x,3))
            {
                printf("%d + %d\n ",x,g-x);
                szam=szam+1;
            }
        x=x+2;
    } while (x<g);
    printf("Goldbachszám: %d\n",szam);
}
```

```

    }
    x=x+2;
}
while (x<=g-x);
printf (" \n");
printf ("Goldbachszam= %d\n",szam);
system ("pause");
}

```

Például:

| <i>g</i> | 200 | 202 | 206 | 208 | 210 | 212 | 214 | 216 | 218 | 220 | 222 | 224 | 226 | 228 |
|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| <i>G</i> | 8   | 9   | 7   | 7   | 19  | 6   | 8   | 13  | 7   | 9   | 11  | 7   | 7   | 12  |

$G$ =Goldbach-szám

A feladat által több kérdés is felvetődhet bennünk. Mely számok tekinthetők elég kicsi, illetve elég nagy Goldbach számoknak? Vannak-e törvényszerűségek? A háromjegyű számoknál 36-tól számítják nagy Goldbach-számnak a számokat.

A kis prímszámokkal osztható számok felírásánál a sor közepén növekszik a Goldbach-szám.

Az előző példában 210 Goldbach száma a legnagyobb, mely a következőképpen állítható elő:  $n=3*5*7=105$ , mely kettővel megszorozva  $2n=210$ , így egy nagy Goldbach-szám áll elő.

## 22. Kombinatorikai és egyéb feladatok

### 1. feladat:

Számoljuk ki  $n$  elem 2-od osztályú kombinációinak számát:

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int n,k;
int f(int n)
{

```

```

srand( time(0) );
if (n<3) return n-1;
else
{
    k=1+rand()%(n-1)/100;
    return (k*(n-k)+f(k)+f(n-k));
}
}
main()
{
    printf ("n="); scanf ("%d",&n);
    printf ("f(n)= %d\n",f(n));
    system("pause");
}

```

|                    |   |   |   |    |    |    |    |    |    |
|--------------------|---|---|---|----|----|----|----|----|----|
| <b><i>n</i></b>    | 2 | 3 | 4 | 5  | 6  | 7  | 8  | 9  | 10 |
| <b><i>f(n)</i></b> | 1 | 3 | 6 | 10 | 15 | 21 | 28 | 36 | 45 |

$$f(n) = \binom{n}{2}$$

Kombináció alatt azt értjük, ha  $n$  különböző lehetséges módon, de sorrendre való tekintet nélkül, kiválasztunk  $k$  elemet, ekkor megkapjuk  $n$  elem  $k$ -ad osztályú kombinációit.

Képletben:  $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ , ahol  $k \leq n$   $\binom{n}{0} = \binom{n}{n} = 1$

Ezt felhasználva könnyen igazolható, hogy  $\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$ , illetve  $\binom{n}{k} = \binom{n}{n-k}$

## 2. feladat: Pascal-háromszög

A binomiális tételből kiindulva  $(a+b)^n = \binom{n}{0}a^n + \binom{n}{1}a^{n-1}b + \binom{n}{2}a^{n-2}b^2 + \dots + \binom{n}{n}b^n$   $n \in N$

Ez a Newtontól származó tétel a kéttagú kifejezések pozitív egész kitevőre való hatványozási szabályát rögzíti.

A binomiális együtthatók egy háromszög alakú táblázatban, az úgynevezett Pascal-féle háromszögben helyezhetők el az alábbi módon:

[illegible]

### A Pascal-háromszögre vonatkozó néhány tulajdonság:

- A Pascal-háromszögben lévő számok a középvonalra nézve szimmetrikusan helyezkednek el.
- Bármelyik, nem a peremen levő szám a fölötte balra és jobbra álló számok összege.
- A bal oldali peremmel párhuzamosan elhelyezkedő, egymás utáni számok összege az utolsó tag alatt jobbra lévő szám.

A Pascal-háromszög  $n$ -edik sorában álló számok összege  $2^n$ .

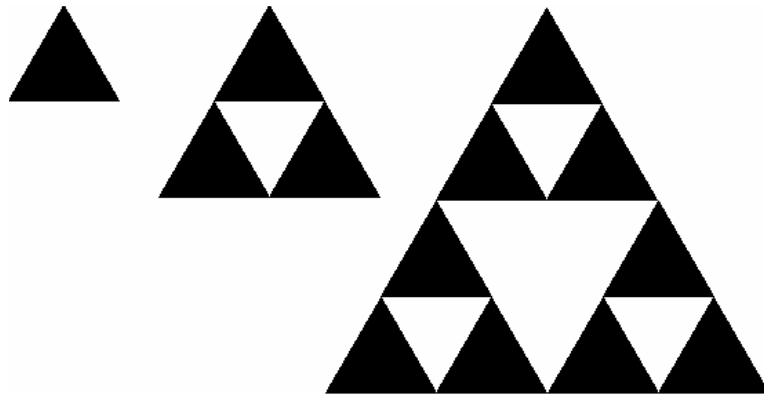
A Pascal-háromszöget leíró program:

```
#include <stdio.h>
#include <stdlib.h>
#define G 100
int a,b,i,j,n;
int x[G];
main()
{
    printf("n= "); scanf("%d",&n);
    for(i=0; i<=n; i++)
        x[i]=0;
    for(j=0; j<=n; j++)
```

}

üresen állnak.

[illegible]



### 3. feladat: elemek permutációja

Hogyan tudnánk egy elemet az egyik állásból a másik állásba tenni?

Például:  $n=10$ ,  $m=3$  0123456789-et a 3456789012 állásba transzformálja. A rotáció két tömb felcserélését jelenti  $AB \rightarrow BA$ . Alkalmazzuk REVERSE eljárást, amely egy blokkot felforgat,  $A \rightarrow A^R$ . Utána képezzük  $(A^R B^R)^R$  ez pedig  $BA$ .

Például:  $n=10$ ,  $m=3$  0123456789  $\rightarrow$  2109876543  $\rightarrow$  3456789012.

A REVERSE eljárást háromszor kell meghívni:

REVERSE (1,  $m$ ):210 3456789

REVERSE ( $m+1$ ,  $n$ ):210 9876543

REVERSE (1,  $n$ ):3456789 012

C-ben megvalósítva:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int k,m,n,i,j,t;
int v[200];
void reverse (int l,int r)
{
    i=l; j=r;
    while (j-i>0)
    {
        t=v[i]; v[i]=v[j]; v[j]=t;
```

```

        i=i+1; j=j-1;
    }
}
main()
{
    printf ("m="); scanf ("%d", &m);
    printf ("n="); scanf ("%d", &n);
    for (k=1; k<=n; k++)
        v[k]=k;
    reverse(1,m); reverse(m+1,n);
    reverse(1,n);
    for (k=1; k<=n; k++)
        printf ("%d", v[k]);
    system ("pause");
}

```

#### 4. feladat:

Definiáljunk egy  $f$  függvényt  $a \geq 0$  egész számok halmazára.  $f(0)=0$ ,  $f(1)=1$ ,  $f(2n)=f(n)$ ,  $f(2n+1)=f(n)+f(n+1)$   $n \geq 0$ .

$$\begin{aligned}
 f(85) &= f(42) + f(43) \\
 &= f(21) + f(21) + f(22) \\
 &= 2 * f(21) + f(11) \\
 &= 2 * f(10) + 3 * f(11) \\
 &= 5 * f(5) + 3 * f(6) \\
 &= 8 * f(3) + 5 * f(2) \\
 &= 13 * f(1) + 8 * f(3) \\
 &= 13 * f(0) + 21 * f(1) \\
 &= 21
 \end{aligned}$$

Ennek megfelelően a képlet:

$$g(n, i, j) = i * f(n) + j * f(n+1)$$

$$g(2n, i, j) = i * f(2n) + j * f(2n+1) = (i+j) * f(n) + j * f(n+1) = g(n, i+j, j)$$

$$g(2n+1, i, j) = i * f(2n+1) + j * f(2n+2) = i * f(n) + (i+j) * f(n+1) = g(n, i, i+j)$$

$g(2n,i,j)=g(n,i+j,j)$

$g(2n+1,i,j)=g(n,i,i+j)$

$f(n)=g(n,1,0)$

$g(0,i,j)=j$   $f(n)=j$

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
double n;
double g(double n,int i,double j)
{
    if (n==0) return j;
    else if (n/2!=((int)(n/2)))
        return (g((n-1)/2,i,i+j));
    else return (g(n/2,i+j,j));
}
main()
{
    printf("n="); scanf("%lf",&n);
    printf ("f(n)= %lf\n",  g(n,1,0));
    system ("pause");
}
```

## Irodalomjegyzék

*Arthur Engel:* Mathematisches experimentieren mit dem PC. Ernst Klett Schulverlag, Stuttgart, 1991.

*Ágotai László:* Ez az optimum. Szalay Könyvkiadó és Kereskedőház Kft., Kisújszállás, 1999.

*Benkő Tiborné, Benkő László, Tóth Bertalan, Varga Balázs:* Programozzunk Turbo Pascal nyelven. Computerbooks, Budapest, 1996.

*Benkő Tiborné, Benkő László, Tóth Bertalan:* Programozzunk C nyelven! Computerbooks, Budapest, 2004.

*Dr. Szendrei János:* Algebra és számelmélet. Nemzeti Tankönyvkiadó, Budapest, 1996.

*Dusza Árpád:* Algoritmusok Pascal nyelven. Dusza Bt., Miskolc, 2005.

*Freud Róbert-Gyarmati Edit:* Számelmélet. Nemzeti Tankönyvkiadó, Budapest, 2000.

*Juhász István, Kósa Márk, Pánovics János:* C példatár. Panem, Budapest, 2005.

*Kovács Ákos-Csorba Kristóf:* Ismerkedés a programozással. Pedellus, Debrecen, 2000.

URL: [http://hu.wikipedia.org/wiki/A\\_sz%C3%A1melm%C3%A9let\\_alapt%C3%A9tele](http://hu.wikipedia.org/wiki/A_sz%C3%A1melm%C3%A9let_alapt%C3%A9tele)

URL: <http://www.bke.hu/~sg/szekciok/matekszek/lexikon.html>

URL: <http://www.jgytf.u-szeged.hu/tanszek/matematika/speckoll/1999/algebra/algebra.html#Goldbach>

URL: [http://www.math.klte.hu/~turjanyi/algebra\\_2006-07-1/a0605.doc](http://www.math.klte.hu/~turjanyi/algebra_2006-07-1/a0605.doc)

URL: [http://matek.fazekas.hu/portal/eloadas/2005/eloadas\\_2005\\_11\\_22\\_freud.pdf](http://matek.fazekas.hu/portal/eloadas/2005/eloadas_2005_11_22_freud.pdf)

URL: <http://www.ttk.pte.hu/matek/ltoth/>