Megoldott programozási feladatok standard C-ben

MÁRTON Gyöngyvér

Sapientia Erdélyi Magyar Tudományegyetem, Matematika-Informatika Tanszék
Marosvásárhely, Románia
mgyongyi@ms.sapientia.ro

Tartalomjegyzék

1. fejezet

Bevezető feladatok

1.1. Feladat. *Egysoros szöveg kiírása, képernyőre.*

```
#include <stdio.h>
int main()
{
    printf("Helo vilag\n");
    return 0;
}
```

1.2. Feladat. *Képernyőre való kiíratás, táblázatos formában.*

```
#include <stdio.h>
int main()
{
    printf("Januar\tFebruar\tMarcius\n");
    printf("2005\t2003\t2001\n");
    printf("1999\t2001\t1989\n\n");
    return 0;
}
```

1.3. Feladat. *Egysoros szöveg kiírása, állományba.*

```
#include <stdio.h>
int main()
{
    FILE *f;
    f = fopen("ki.txt","w");
    fprintf(f,"Helo vilag\n");
```

```
    fclose(f);
    return 0;
}
```

1.4. Feladat. *Állományba való kiíratás, táblázatos formában.*

```
#include <stdio.h>
int main()
{
    FILE *f;
    f = fopen("ki.txt","w");
    fprintf(f,"Januar\tFebruar\tMarcius\n");
    fprintf(f,"2005\t2003\t2001\n");
    fprintf(f,"1999\t2001\t1989\n");
    fclose(f);
    return 0;
}
```

1.5. Feladat. *Megjegyzések használata.*

```
#include <stdio.h>
int main()
{
    printf("Ez megjelenik a kepernyon\n");
    //printf("Ez nem jelenik meg a kepernyon\n");
    printf("Ez megint megjelenik a kepernyon\n");
    /*
    ez egy tobb sorbol allo megjegzes
    nem fog megjelenni a kepernyon
    a programozo utolagos eligazodasara szolgall
    */
    return 0;
}
```

1.6. Feladat. *Szorzás.*

```
#include <stdio.h>
int main()
{
    int x; //valtozo deklaralas
    x = 25; //valtozo ertekadasa
```

```
    printf("%i\n",x*x); //kepernyore valo kiiratas
    return 0;
}
```

1.7. Feladat. *Az osztási hányados meghatározása.*

```
#include <stdio.h>
int main()
{
    int x;
    x = 10;
    printf("%i\n",x/3); //az eredmeny az osztas egesz resze
    return 0;
}
```

1.8. Feladat. *Osztás, az eredmény egy valós szám*

```
#include <stdio.h>
int main()
{
    int x;
    x = 10;
    printf("%.3f\n",(float)x/3);
    return 0;
}
```

1.9. Feladat. *Az osztási maradék meghatározása.*

```
#include <stdio.h>
int main()
{
    int x;
    x = 10;
    //maradekos osztas
    printf("%i\n",x%3);
    //az eredmeny ugyanaz, csak mas formazasi jelet hasznalunk
    printf("%i\n",x%3);
    return 0;
}
```

1.10. Feladat. *Aritmetikai műveletek.*

```
#include <stdio.h>
int main()
{
    int i;
    i = 2;
    printf("Egesz tipusu valtozo: %i\n", i);
    // a valtozo erteket noveljuk 4-el
    i = i + 4;
    printf("az eredmeny: %i\n",i);
    // a valtozo erteket noveljuk 1-el
    i++;
    printf("az eredmeny: %i\n",i);
    // a valtozo erteket noveljuk 2-vel
    i += 2;
    printf("az eredmeny: %i\n",i);
    // a valtozo erteket szorozzuk 3-al
    i *= 3;
    printf("az eredmeny: %i\n",i);
    return 0;
}
```

1.11. Feladat. *Két változó értékének a felcserélése.*

```
#include <stdio.h>
int main()
{
    int a,b,seged;
    a = 12;
    b = -12;
    printf("Csere elott a szamok:\n\t%i %i\n",a,b);
    // elso modszer:
    seged = a;
    a = b;
    b = seged;
    printf("Elso csere utan a szamok:\n\t%i %i\n",a,b);
    // masodik modszer
    a = a - b;
    b = a + b;
    a = b - a;
    printf("Masodik csere utan a szamok:\n\t%i %i\n",a,b);
```

```
        return 0;
}
```

1.12. Feladat. *Egész típus.*

```
#include <stdio.h>
int main()
{
    int a, b, ered;
    printf("Kerek egy egesz szamot:");
    scanf("%i",&a);
    printf("Kerek egy egesz szamot:");
    scanf("%i",&b);
    printf("\n");
    ered = a + b;
    printf("Az osszeg:");
    printf("%i", ered);
    printf("\n\n");
    return 0;
}
```

1.1. Megjegyzés. *Az* `int a;` *változó deklarálásakor*

- *az* `a` *azonosítóval a változó értékére tudunk hivatkozni,*
- *a* `&a` *azonosítóval a változó memória címére tudunk hivatkozni.*

1.2. Megjegyzés. *A* `scanf` *könyvtárfüggvény használatakor a változó memória címét használjuk, kivéve ha karakterlánc típusú változónak adunk értéket*

1.13. Feladat. *Karakter típus.*

```
#include <stdio.h>
int main()
{
    char a;
    printf("kerek egy karaktert:");
    scanf("%c",&a);
    printf("A karakter es az ASCII kodja\n");
    printf("%c\t%i",a,a);
    printf("\n");
    return 0;
}
```

1.14. Feladat. *Karakterlánc típus.*

```
#include <stdio.h>
int main()
{
    char szoveg[20];
    printf("Kerek egy szoveget");
    scanf("%s",szoveg);
    printf("%s",szoveg);
    printf("\n");
    return 0;
}
```

1.15. Feladat. *Valós típus, float.*

```
#include <stdio.h>
int main()
{
    float a, b;
    ered1 = 0;
    printf("Kerek egy valos szamot:");
    scanf("%f",&a);
    printf("Kerek egy valos szamot:");
    scanf("%f",&b);
    printf("\n");
    a = a * b; // a beolvasott erteket felulirjuk
    printf("%20s","A szorzat:");
    printf("%8.2f\n", a);
    return 0;
}
```

1.16. Feladat. *Valós típus, double.*

```
#include <stdio.h>
int main()
{
    double f1,f2,f3,f4;
    printf("\n\nKerek egy valos szamot:");
    scanf("%lf",&f1);
    printf("Kerek egy valos szamot:");
    scanf("%lf",&f2);
```

```
    printf("Kerek egy valos szamot:");
    scanf("%lf",&f3);
    printf("Kerek egy valos szamot:");
    scanf("%lf",&f4);
    printf("a szamok atlaga: ");
    printf("%6.3lf\n",(f1+f2+f3+f4)/4 );
    return 0;
}
```

1.17. Feladat. *A* `sizeof` *operátor alkalmazása*

```
#include <stdio.h>
int main()
{
    printf("karakter tipus -byte szama:: ");
    printf("%i\n",sizeof(char));
    printf("egesz tipus -byte szama:: ");
    printf("%i\n",sizeof(int));
    printf("valos tipus -byte szam:: ");
    printf("%i\n",sizeof(float));
    printf("hosszu valos -tipus byte szama:: ");
    printf("%i\n",sizeof(double));
    return 0;
}
```

1.18. Feladat. *Típuskonverzió*

```
#include <stdio.h>
int main()
{
    int a,b;
    printf("Kerek egy egesz szamot:");
    scanf("%i",&a);
    printf("Kerek egy egesz szamot:");
    scanf("%i",&b);
    printf("\n");
    printf("%20s%8d\n","Az egesz osztas:", a / b);
    /*
    az a valtozot valos szamma alakitom, csak igy kapom meg
    a valós osztás eredményét
    */
```

```
    printf("%20s%8.2f\n\n","A valos osztas:", (float)a / b);
    return 0;
}
```

1.19. Feladat. *Számrendszerek: tízes, nyolcas, tizenhatos.*

```
#include <stdio.h>
int main()
{
    char c;
    int i;
    float f;
    printf("Kerek egy karakter:");
    scanf("%c",&c);
    printf("A karkter erteke:\t\t%c\n",c);
    printf("A karakter kodja:\t\t%i\n",c);
    printf("A karakter kodja oktalisan:\t%o\n",c);
    printf("A karakter kodja hexaban:\t%x\n",c);
    printf("\n\nKerek egy egesz szamot:");
    scanf("%i",&i);
    printf("Az egesz szam tizedeskent:\t%i\n",i);
    printf("Az egesz szam tizedeskent:\t%i\n",i);
    printf("Az egesz szam oktalisan:\t%o\n",i);
    printf("Az egesz szam hexaban:\t\t%x\n",i);
    printf("\n\nKerek egy valos szamot:");
    scanf("%f",&f);
    printf("A valos szam erteke:\t\t%7.3f\n",f);
}
```

2. fejezet

Feltételes utasítások

2.1. Az if utasítás

2.1. Feladat. *Páros vagy páratlan számot olvastunk be?*

```
#include <stdio.h>
int main()
{
    int szam;
    printf("egesz szam:");
    scanf("%i",&szam);
    if( szam%2 == 0) printf("a szam paros\n");
    else printf("a szam paratlan\n");
    return 0;
}
```

2.2. Feladat. *Milyen karaktert olvastunk be?*

```
#include <stdio.h>
int main()
{
    char c1;
    int mas = 1;
    printf("karakter:");
    scanf("%c",&c1);
    if( c1 >= 'a' && c1 <= 'z')
    {
        printf("kisbetut olvastal be\n");
```

```
        return 0;
    }
    if( c1 >= 'A' && c1 <= 'Z')
    {
        printf("nagy betut olvastal be\n");
        return 0;
    }
    if( c1 >= '0' && c1 <= '9')
    {
        printf("szamjegyet olvastal be\n");
        return 0;
    }
    printf("Nem angol ABC-beli betut
    es nem is szamjegyet olvastal be\n");
    return 0;
}
```

2.3. Feladat. *Másodfokú egyenlet megoldása.*

```
#include <stdio.h>
#include <math.h>
int main()
{
    int a, b, c;
    float ered1,ered2,delta;
    printf("a:");
    scanf("%i",&a);
    printf("b:");
    scanf("%i",&b);
    printf("c:");
    scanf("%i",&c);
    if(a == 0)
    {
        //a sajatos esetek meghatarozasa
        if(b != 0 && c != 0) {
            ered1 = -(float)c/b;
            printf("az eredmeny:%5.2f\n",ered1);
        }
        if(c == 0 && b == 0) printf("Vegtelen sok megoldas\n");
        if(c != 0 && b == 0) printf("Nincs megoldas\n");
```

```
            if(c == 0 && b != 0) printf("az eredmeny:0\n");
        }
        else
        {
            /*
            a negyzetgyok meghatarozasara hasznaljuk
            az sqrt beepietett konyvtarfuggvenyt
            */
            delta = sqrt(b * b - 4 * a * c);
            if(delta < 0) printf("Nincs valos megoldas\n");
            else {
                ered1 = (-b + delta) / 2;
                ered2 = (-b - delta) / 2;
                //a kiiratast 2 tizedesnyi pontosaggal vegezzuk
                printf("az eredmeny:%5.2f\n",ered1);
                printf("az eredmeny:%5.2f\n",ered2);
            }
        }
        return 0;
    }
```

2.2. A switch utasítás

2.4. Feladat. *Milyen műveleteket végezzünk?*

```
    #include <stdio.h>
    int main()
    {
        int c;
        int szam1, szam2, szam3;
        int ered;
        printf("0 Kilpes\n");
        printf("1 Osszeadas\n");
        printf("2 Szorzas\n");
        printf("Milyen muveleteket vegzunk a szamokon?");
        scanf("%i",&c);
        if (c == 0) return 0;
        printf("Elso szam:");
        scanf("%i",&szam1);
        printf("Masodik szam:");
```

```
        scanf("%i",&szam2);
        printf("Harmadik szam:");
        scanf("%i",&szam3);
        switch(c)
        {
        case 1 :
            ered = 0;
            ered += szam1;
            ered += szam2;
            ered += szam3;
            printf("Osszeadas\n");
            printf("Az eredmeny:%i\n\n\n",ered);
            break;
        case 2:
            ered = 1;
            ered *= szam1;
            ered *= szam2;
            ered *= szam3;
            printf("Szorzas\n");
            printf("Az eredmeny:%i\n\n\n",ered);
            break;
        }
        return 0;
}
```

3. fejezet

Ciklus utasítások

3.1. A while utasítas

3.1. Feladat. *Melyik ciklus gyorsabb?*

```
}
    #include <stdio.h>
    #include <stdlib.h>
    #include <time.h>
    #define N1 10000000
    #define N2 100000000

    int main()
    {
        __int64 i;
        clock_t st;
        st = clock();
        for(i=0; i<N1; i++);

        printf("%I64i\t", i);
        printf("time:%.3lf\n",(clock()-st)/(double)CLOCKS_PER_SEC);

        st = clock();
        for(i=0; i<N2; ++i);

        printf("%I64i\t", i);
        printf("time:%.3lf\n",(clock()-st)/(double)CLOCKS_PER_SEC);
    }
```

3.2. Feladat. *Páros számok kiíratása adott n értékig.*

```
#include <stdio.h>
int main()
{
    int i, k, n;
    printf("Meddig::");
    scanf("%i",&n);
    i = 0;
    k = 0;
    while(i < n)
    {
        printf("%i ",k);
        k = k + 2;
        i++;
    }
    printf("\n");
    return 0;
}
```

3.3. Feladat. *Adott szám számjegyeinek fordított sorrendjének a meghatározása.*

```
#include <stdio.h>
int main()
{
    unsigned int szam, szamj;
    //elojel nelkuli egesz tipus deklaralasa
    printf("Kerek egy szamot::");
    scanf("%i",&szam);
    printf("\nA szam szamjegyei forditott sorrendben:");
    while(szam != 0)
    {
        szamj = szam % 10;
        printf("%i ", szamj);
        szam = szam / 10;
    }
    return 0;
}
```

3.4. Feladat. *Két szám legnagyobb közös osztójának a meghatározása, Euklideszi algoritmussal.*

```
#include <stdio.h>
int main()
{
    unsigned int a, b, r;
    printf("Kerek egy egesz szamot:");
    scanf("%i",&a);
    printf("Kerek meg egy egesz szamot");
    scanf("%i",&b);
    while ( b != 0 )
    {
        r = a % b;
        a = b;
        b = r;
    }
    printf("Az lnko:%i\n",a);
    return 0;
}
```

3.5. Feladat. *Határozzuk meg egy adott állományban levő számokra a számok négyzetgyökét.*

```
#include <stdio.h>
double negyzetgy(int x);

int main() {
    FILE *f;
    double y = 1;
    int x;
    f = fopen("szamok.txt", "r");
    while(1)
    {
        fscanf(f,"%i", &x);
        if(feof(f)) break;
        y = negyzetgy(1, x);
        printf("%10i%10.2lf\n", x, y);
    }
    fclose(f);
    return 0;
}
double negyzetgy(int x) {
```

```
    double y = 1;
    while (y*y - x > 0.00001 || x-y*y > 0.00001)
        y = (y + x/y) /2;
    return y;
    }
```

3.6. Feladat. *Határozzuk meg egy adott állományban levő* $\boldsymbol{x}$ *és* $\boldsymbol{y}$ *számpárokra az* x^y *értékét, felhasználva a* ***pow*** *könyvtárfüggvényt.*

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

int main()
{
    FILE *f;
    double e;
    int x, y;
    f = fopen("szamok.txt", "r");
    while(1)
    {
        fscanf(f,"%i%i", &x, &y);
        if(feof(f)) break;
        e = pow(x, y);
        printf("%10i%10i%10.0lf\n", x, y, e);
    }
    fclose(f);
    return 0;
}
```

3.2. A for utasítas

3.7. Feladat. *Páros számok kiíratása adott* n *értékig.*

```
#include <stdio.h>
int main()
{
    int i, n;
    printf("n:");
    scanf("%i",&n);
    for(i = 0; i < n; i++)
```

```
        printf("%i\n",2*i);
    return 0;
}
```

3.8. Feladat. *Billentyűzetről beolvasott számok átlagértékének a meghatározása.*

```
#include <stdio.h>
int main()
{
    int i, n, ossz, szam;
    printf("n=");
    scanf("%i",&n);
    ossz = 0;
    for(i = 0; i < n; i++)
    {
        printf("kerek egy szamot:");
        scanf("%i",&szam);
        ossz += szam;
    }
    printf("A szamok atlaga::%.2f\n",(float)ossz/n);
    return 0;
}
```

3.9. Feladat. *Határozzuk meg a billentyűzetről beolvasott számok közül a pozitív számok számát,* ***n*** *szám esetében*

```
#include <stdio.h>

int main()
{
    int i, n, db, szam;
    printf("n=");
    scanf("%d",&n);
    db = 0;
    for(i = 0; i < n; i++)
    {
        printf("kerek egy szamot:");
        scanf("%d",&szam);
        if (szam>0) db++;
    }
    printf("A pozitiv szamok szama: %i\n",db);
```

```
    return 0;
}
```

3.10. Feladat. *Válasszuk ki a billentyűzetről beolvasott számok közül a legnagyobbat.*

```
#include <stdio.h>
int main()
{
    int i, n, max, szam;
    printf("n=");
    scanf("%i",&n);
    max = szam; printf("kerek egy szamot:");
    scanf("%i",&szam);
    max = szam; //inicializalas
    for(i = 1; i < n; i++)
    {
        printf("kerek egy szamot:");
        scanf("%i",&szam);
        if (max < szam) max = szam;
    }
    printf("A legnagyobb szam::%i\n",max);
    return 0;
}
```

3.11. Feladat. *Határozzuk meg a billentyűzetről beolvasott számok átlagértékét,* ***n*** *szám esetében*

```
#include <stdio.h>
int main()
{
    int i, n, ossz, szam;
    printf("n=");
    scanf("%d",&n);
    ossz = 0;
    for(i = 0; i < n; i++)
    {
        printf("kerek egy szamot:");
        scanf("%d",&szam);
        ossz += szam;
    }
```

```
    printf("A szamok atlaga::%.2f\n",(float)ossz/n);
    return 0;
}
```

3.12. Feladat. *Határozzuk meg a billentyűzetről beolvasott számok szorzatát,* ***n*** *szám esetében*

```
#include <stdio.h>
int main()
{
    int i, n, szorzat, szam;
    printf("n=");
    scanf("%d",&n);
    szorzat = 0;
    for(i = 0; i < n; i++)
    {
        printf("kerek egy szamot:");
        scanf("%d",&szam);
        szorzat *= szam;
    }
    printf("A szamok szorzata: %i\n",szorzat);
    return 0;
}
```

3.13. Feladat. *Írjuk ki az ASCII kódtáblát.*

```
#include <stdio.h>
int main()
{
    int i;
    for(i = 0; i < 256; i++)
        printf("%c: %i\t", i, i);
    printf("\n\n");
    return 0;
}
```

3.14. Feladat. *Vizsgáljuk meg egy számról, hogy prím szám-e, vagy sem.*

```
#include <stdio.h>
int main()
{
```

```
        int szam, ok, i;
        printf("Kerek egy szamot:");
        scanf("%i",&szam);
        ok = 1;
        if( szam ==2 ) printf("A szam prim\n");
        else
            if( szam % 2 == 0) printf("A szam nem prim\n");
            else
            {
                for(i=3; i*i<=szam && ok ; i+=2)
                //eleg a paratlan osztokat vizsgalni
                    if(szam % i == 0)
                    {
                        printf("A szam nem prim\n");
                        ok = 0;
                    }
                 if (ok) printf("A szam prim\n");
            }
            return 0;
    }
```

3.15. Feladat. *Írjunk egy függvényt mely megvizsgálja hogy egy szám teljes négyzet-e vagy sem.*

```
    #include <stdio.h>
    int negyzetszam(int szam);

    main() {
        int p, sz = 141;
        p = negyzetszam(sz);
        if(p == 1) printf("teljes negyzet\n");
        else printf("nem teljes negyzet\n");
        return 0;
}

    int negyzetszam(int szam) {
        int i;
        for (i=1; i*i<=szam; i++)
            if (i*i == szam) return 1;
```

```
    return 0;
}
```

4. fejezet

Bit műveletek

4.1. Feladat. *A 0-ik bit 1-re állítása.*

```
#include <stdio.h>
int main()
{
    unsigned int a=28; // 0x1c;
    printf("Az OR muvelet bitteken:\n%x\n",a);
    a = a | 0x1;
    printf("%x\n",a);
    return 0;
}
```

4.2. Feladat. *Az 5-ik bitet 0-ra állítása.*

```
#include <stdio.h>
int main()
{
    unsigned int a = 58;
    printf("\nAz END muvelet bitteken:\n%x\n",a);
    a = a & 0x1f;
    printf("%x\n",a);
    return 0;
}
```

4.3. Feladat. *Adott számérték titkosítása.*

```
#include <stdio.h>
int main()
```

```
{
    unsigned int a = 1994;
    printf("\nTitkositas kovetkezik, a XOR muvelettel:\n");
    printf("A unsigned int:\t\t%x\n",a);
    a = a ^ 0xaaaa;
    printf("Kodolva:\t%x\n",a);
    a = a ^ 0xaaaa;
    printf("Dekodolva:\t%x\n",a);
    return 0;
}
```

4.4. Feladat. *Két változó értékének a felcserélése*

```
#include <stdio.h>
int main()
{
    unsigned int a, b;
    printf("\nszam1:");
    scanf("%d",&a);
    printf("szam2:");
    scanf("%d",&b);
    b = a ^ b;
    a = a ^ b;
    b = a ^ b;
    printf("\nszam1:%d",a);
    printf("\nszam2:%d",b);
    return 0;
}
```

4.5. Feladat. *Egy szám kettes számrendszerbeli alakja.*

```
#include <stdio.h>
int main()
{
    unsigned int a, b, i = 7;
    printf("Kerem a szamot:");
    scanf("%d",&a);
    while(i >= 0)
    {
        b = (a >> i) & 1;
        printf("%i",b);
```

```
        i--;
    }
    return 0;
}
```

4.6. Feladat. *Az unsigned int típus belső ábrázolásához szükséges bitek száma.*

```
#include <stdio.h>
int main()
{
    unsigned int a = 1, sz = 0;
    //a szám belso abrazolasanak a negaltja
    a = ~a;
    while(a != 0)
    {
        a = a >> 1;
        sz++;
    }
    printf("A gepszo hosszusaga:%d", sz);
}
```

5. fejezet

Tömbök

5.1. Egydimenziós tömbök

5.1. Feladat. *Egy egydimenziós tömb elemeit inicializáljuk véletlenszerűen generált elemekkel, majd határozzuk meg az elemek összegét.*

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    int n, tomb[50];
    int i, ossz;
    srand(time(NULL));
    printf("n:");
    scanf("%d",&n);
    for(i = 0; i < n; i ++)
    {
        tomb[i] = ( rand() % 100 ) - 50;
        printf("%d\t",tomb[i]);
    }
    printf("\n");
    ossz = 0;
    for(i = 0; i < n; i++)
        ossz += tomb[i];
    printf("Az osszeg: %d\n",ossz);
    return 0;
}
```

5.2. Feladat. *Egy egydimenziós tömb elemeit inicializáljuk véletlenszerűen generált elemekkel, majd határozzuk meg a páros elemek indexét.*

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    int n, tomb[50];
    int i;
    srand(time(NULL));
    printf("n:");
    scanf("%d",&n);
    for(i = 0; i < n; i++)
        printf("%3d\t",i);
    printf("\n");
    for(i = 0; i < n; i++)
    {
        tomb[i] = ( rand() % 100 ) - 50;
        printf("%3d\t",tomb[i]);
    }
    printf("\n\n");
    printf("A paros elemek pozicioi:");
    for( i = 0; i < n; i++)
        if(tomb[i]%2 == 0) printf("%d\t", i);
    printf("\n\n");
    return 0;
}
```

5.3. Feladat. *Határozzuk meg egy tömb elemeinek a szorzatát, ahol a tömb elemeit konstansként inicializáljuk.*

```
#include <stdio.h>

int main() {
    int n, tomb[] = {10, 8, 7, 6, 3};
    int i, szorzat;
    n = sizeof(tomb)/sizeof(tomb[0]);
    szorzat = 1;
    for(i=1; i<n; i++)
        szorzat *= tomb[i];
```

```
    printf("A szorzat: %i", szorzat);
    return 0;
}
```

5.4. Feladat. *Határozzuk meg egy tömb elemeinek az átlagértékét, ahol a tömb elemeit konstansként inicializáljuk.*

```
#include <stdio.h>

int main() {
    int n, tomb[] = {10, 8, 7, 6, 3};
    int i, osszeg;
    n = sizeof(tomb)/sizeof(tomb[0]);
    osszeg = 0;
    for(i=0; i<n; i++)
        osszeg += tomb[i];
    printf("Az atlag: %.2f\n",(float)osszeg/n);
    return 0;
}
```

5.5. Feladat. *Határozzuk meg egy tömb elemei közül a maximum elemet.*

```
#include <stdio.h>

int main() {
    int n, tomb[] = {10, 8, 7, 6, 3};
    int i, max;
    n = sizeof(tomb)/sizeof(tomb[0]);
    max = tomb[0];
    for(i=0; i<n; i++)
        if(tomb[i]>max) max =tomb[i];
    printf("Maximum: %d\n",max);
    return 0;
}
```

5.6. Feladat. *Vizsgáljuk meg, hogy egy tömb csak páros számokat tartalmaze vag sem.*

```
#include <stdio.h>

int csak_paros(int t [], int i);
```

```
main() {
    int t[]= {10,21,0,32,4,52};
    int n, v;
    n = sizeof(t)/sizeof(t[0]);
    v = csak_paros(t, n-1);
    if (v==1) printf("csak paros elemeket tartalmaz\n");
    else printf("nem csak paros elemeket tartalmaz\n");
    return 0;
}

int csak_paros(int t[], int n) {
    int i;
    for(i=0; i<n; i++)
        if (t[i]%2 == 1) return 0;
    return 1;
}
```

5.7. Feladat. *Határozzuk meg egy szám kettes számrendszerbeli alakját, egy tömbbe.*

```
#include <stdio.h>
main()
 {
    int i, k, tomb[100];
    int szam = 18;
    k = 0;
    while(szam>0)
    {
        tomb[k++] = szam%2;
        szam /=2;
    }
    for(i=k-1; i>=0; i--)
        printf("%3i", tomb[i]);
    printf("\n");
    return 0;
}
```

5.8. Feladat. *Határozzuk meg egy szám valódi osztóit, előállítva őket egy tömbbe.*

```
   #include <stdio.h>

    main() {
        int szam, k, i, tomb[100];
        printf("szam:");
        scanf("%i", &szam);
        k = 0;
        for(i = 2; i<=szam/2; i++)
            if(szam%i == 0) tomb[k++] = i;
        for(i=0; i<k; i++)
            printf("%4i", tomb[i]);
        printf("\n");
        return 0;
}
```

5.9. Feladat. *Hozzunk létre egy n elemű tömböt, mely tartalmazza az első n prím számot.*

```
    #include <stdio.h>
    #include <math.h>
    int main()
    {
        int n, tomb[10000];
        int i, szam, prim, j;
        printf("n:");
        scanf("%d",&n);
        tomb[0] = 2;
        i = 1; szam = 3;
        while( i<n )
        {
            prim = 1;
            for( j = 3; j <= sqrt(szam) && prim; j += 2)
                if ( szam % j == 0 ) prim = 0;
                if ( prim ) {
                    tomb[i] = szam;
                    i++;
                }
            szam += 2;
        }
        for(i = 0; i < n; i++)
```

```
        printf("%d\t",tomb[i]);
    printf("\n\n");
    return 0;
}
```

5.10. Feladat. *Adott egy egész szám. Töltsünk fel egy tömböt a szám számjegyeivel.*

```
#include <stdio.h>
int main()
{
    //64 bitten tarolt szam
    __int64 szam;
    int i, n, tomb[100];
    printf("szam:");
    scanf("%I64i", &szam);
    n = 0;
    while (szam != 0 )
    {
        tomb[n] = szam % 10 ;
        n++;
        szam = szam / 10;
    }
    //a tomb elemeit forditott sorrendbe irjuk ki
    for ( i = n-1; i >= 0; i--)
        printf("%d ", tomb[i]);
    printf("\n\n");
    return 0;
}
```

5.11. Feladat. *Olvassunk be egy karakterláncot a billentyűzetről és készítsünk karakter előfordulási statisztikát.*

```
#include <stdio.h>
int main()
{
    char tomb[20];
    int i, stat[256];
    printf("karakterlanc:");
    scanf("%s", tomb);
    for(i = 0; i < 256; i++)
        stat[i] = 0;
```

```
    for(i = 0; tomb[i] != '\0'; i++)
        stat[ tomb[i] ]++;
    for(i = 0; i < 256; i++)
        if( stat[i] != 0) printf("%c:: %d\n", i, stat[i]);
    printf("\n\n");
    return 0;
}
```

5.2. Kétdimenziós tömbök

5.12. Feladat. *Adott egy n*n-es mátrix, melynek elemeit véletlenszerű egész számokkal töltsük fel. Írjuk ki a mátrixot táblázatos formában majd határozzuk meg a mátrix legkisebb elemét és ezen elem sor és oszlop értékét*

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    int n, mat[10][10];
    int i, j, max, m, mi, mj;
    printf("n: ");
    scanf("%d", &n);
    printf("m: ");
    scanf("%d", &m);
    if (n >= 10 || m >= 10 ) //hibakezeles
    {
        printf("Error:: beolvasott ertek >= 10 \n");
        exit(1);
    }
    srand(time(NULL));
    for(i = 0; i < n; i++)
    {
        for(j = 0; j < m; j++) //vegig megyunk az osszes oszlopbeli elemen
        {
            mat[i][j] = rand() % 100; //ertekadas
            printf("%3d", mat[i][j]); //kiiratas
        }
        printf("\n"); //ujsorba megyunk ha kiirtuk a matrix egy adott sorat
    }
```

```
    max = mat[0][0];
    mi = 0;
    mj = 0;
    for(i = 0; i < n; i++)
        for(j = 0; j < m; j++)
            if( mat[i][j] > max ) {
                max = mat[i][j];
                mi = i;
                mj = j;
            }
    printf("\nA maximum elem: %d", max);
    printf("\nA maximum elem pozicioi: %d, %d", mi + 1, mj + 1);
    printf("\n\n");
    return 0;
}
```

6. fejezet

Rekurzió

6.1. Feladat. *Határozzuk meg egy adott* ***n*** *szám faktoriálisát.*

```
#include <stdio.h>

int fakt(int szam);

main() {
    int szam = 5;
    int m;
    m = fakt(szam);
    printf("Faktorialis: %i\n", m);
    return 0;
    }

int fakt(int n)
{
    int t;
    if (n == 0) return 1;
    t = fakt(n-1);
    return n * t;
}
```

6.2. Feladat. *Határozzuk meg egy adott szám számjegyeinek az összegét.*

```
#include <stdio.h>

int szamj(int szam);
```

```
main() {
    int szam = 12345;
    int m;
    m = szamj(szam);
    printf("Osszeg: %i\n", m);
    return 0;
}

int szamj(int szam) {
    int t;
    if (szam <= 0) return 0;
    t = szamj1(szam/10);
    return (szam%10) + t;
}
```

6.3. Feladat. *Határozzuk meg egy tömb elemének a maximum elemét.*

```
#include <stdio.h>

int mmax(int t[], int n);

main() {
    int t [] = {1034, 6, 912, 356, 11, 8, 99};
    int n = sizeof(t)/sizeof(t[0]);
    int m;
    m = mmax (t,n-1);
    printf("Maximum: %i\n", m);
    return 0;
}

int mmax(int t[], int n) {
    int m;
    if (n == 0) return t[0];
    m = mmax(t, n-1);
    if (m < t[n]) return t[n];
    else return m;
}
```

6.4. Feladat. *Vizsgáljuk meg, hogy egy tömb elemei csak párosak-e vagy sem.*

```
#include <stdio.h>

int csak_paros(int t [], int i);

main() {
    int t[]= {10,2,0,32,4,52};
    int n, v;
    n = sizeof(t)/sizeof(t[0]);
    v = csak_paros(t, n-1);
    if (v == 1) printf("csak paros elemeket tartalmaz\n");
    else printf("nem csak paros elemeket tartalmaz\n");
    return 0;
}

int csak_paros (int t[], int n) {
    if (n < 0) return 1;
    if (t[n]%2 == 1) return 0;
    csak_paros(t, n-1);}
```

6.5. Feladat. *Határozzuk meg két szám legnagyobb közös osztóját, Euklideszi algoritmussal, rekurzívan.*

```
#include <stdio.h>

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

main() {
    printf("%i", lnko(48,102));
}
```

6.6. Feladat. *Írjuk ki 2 hatványait egy megadott* ***n*** *számig.*

```
#include <stdio.h>

int rec(int sz, int n);
int main() {
    int n = 10;
```

```
    rec(1, n);
    return 0;
}

int rec(int sz, int n) {
    if(n<0) return 0;
    printf("%i\n", sz);
    rec(sz*2, n-1);
}
```

6.7. Feladat. *Határozzuk meg egy szám kettes számrendszerbeli alakját, rekurzívan.*

```
#include <stdio.h>

int kettes(int szam); int main() {
    int sz = 18;
    kettes(sz);
    printf("\n");
    return 0;
}

int kettes(int szam) {
    if(szam<=0)
        return 0;
    kettes(szam/2);
    printf("%3i",szam%2);
}
```

6.8. Feladat. *Határozzuk meg egy szám négyzetgyökét.*

```
#include <stdio.h>

double negyzetgy(double y, int x);

int main() {
    double y;
    int x = 16;
    y = negyzetgy(y, x);
    printf("%10i%10.2lf\n", x, y);
```

```
    return 0;
}

double negyzetgy(double y, int x) {
    y = (y+ x/y)/2;
    if(y*y-x < 0.00001 && x-y*y < 0.00001) return y;
    return negyzetgy(y,x);
}
```

Tárgymutató