Megoldott programozási feladatok standard C-ben

Márton Gyöngyvér

Műszaki és Humántudományok Kar
EMTE-Sapientia, Marosvásárhely

2008. december 14.

1. fejezet

Megoldott feladatok

1.1. Alap algoritmusok

1.1. 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;
}
```

1.2. 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;
}
```

1.3. 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;
}
```

1.4. feladat. *Határozzuk meg a billentyüzetről beolvasott számok közül a legnagyobbat,* ***n*** *szám esetében*

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

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

1.5. feladat. *Határozzuk meg két szám legnagyobb közös osztóját, Euklideszi algoritmussal*

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

1.6. 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;
    }
```

1.7. 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;
    }
```

1.8. feladat. *Írjunk egy függvényt mely megvizsgálja hogy egy szám prímszám-e vagy sem.*

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

main() {
    int p, sz = 91;
    p = prim(sz);
    if(p == 1) printf("prim szam\n");
    else printf("nem prim szam\n");
    return 0;
}

int prim(int szam) {
    int i;
    if(szam == 2) return 1;
    if(szam%2 == 0) return 0;
    for(i=3; i*i<=szam; i++)
        if(szam%i == 0) return 0;
    return 1;
}
```

1.9. 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;
    }
```

1.2. Tömbös feladatok

1.10. feladat. *Határozzuk meg egy tömb elemei közül a pozitív elemek számát,* ***n*** *szám esetében*

```
   #include <stdio.h>

    main() {
        int t[]= {10,-21,0,32,4,52};
        int n, db, i;
        n = sizeof(t)/sizeof(t[0]);
        db = 0;
        for (i=0; i<n; i++)
            if (t[i]>0==1) db++;
        printf("a pozitiv elemek szama: %i\n", db);
        return 0;
}
```

1.11. feladat. *Határozzuk meg egy tömb elemeinek az átlagértékét.*

```
    #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;
}
```

1.12. feladat. *Határozzuk meg egy tömb elemeinek a szorzatát.*

```
#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;
}
```

1.13. feladat. *Határozzuk meg egy tömb elemének a maximum elemét.*

```
#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;
}
```

1.14. feladat. *Határozzuk meg egy tömb elemei közül a páros elemek pozícióját, ahol a tömb elemei véletlenszerűen generált számok.*

```
#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;
        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;
}
```

1.15. feladat. *Vizsgáljuk meg, hogy egy tömb csak páros számokat tartalmaz-e vagy 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;
}
```

1.16. 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;
}
```

1.17. 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;
}
```

1.18. feladat. *Határozzuk meg egy szám kettes számrendszerbeli alakjában az 1-ek számát.*

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

1.19. feladat. *Állítsuk elő a következő számsorozatot egy tömbbe, $n = 5$ esetén:* $1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5$

```
#include <stdio.h>

 int main(){
     int i, n, j, k, tomb[100];
     printf("n:");
     scanf("%i", &n);
     k = 0;
     for(i=1; i<=n; i++)
         for(j=1; j<=i; j++)
             tomb[k++] = i;
     for (i=0; i<k; i++)
```

```
            printf("%4i", tomb[i]);
        printf("\n");
        return 0;
    }
```

1.3. Rekurzió

1.20. feladat. *Határozzuk meg egy adott* ***n*** *szám faktoriálisát*

```
#include <stdio.h>
#include <stdlib.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;
}
```

1.21. feladat. *Határozzuk meg egy adott szám számjegyeinek az összegét*

```
#include <stdio.h>
#include <stdlib.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;
}
```

1.22. feladat. *Határozzuk meg egy tömb elemének a maximum elemét*

```
#include <stdio.h>
#include <stdlib.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;
}
```

1.23. 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);}
```

1.24. feladat. *Határozzuk meg két szám legnagyobb közös osztóját, Euklideszi algoritmussal.*

```
#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));
}
```

1.25. feladat. *Írjuk ki 2 hatványait egy megadott* $\boldsymbol{n}$ *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);
}
```

1.26. feladat. *Határozzuk meg egy szám kettes számrendszerbeli alakját.*

```
#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);
}
```

1.27. feladat. *Határozzuk meg egy szám négyzetgyökét.*

```
#include <stdio.h>

double negyzetgy(double y, int x);

int main() {
    int x = 16;
    y = negyzetgy(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 negyzetgy2(y,x);
}
```