

Programozás alapjai
9. gyakorlat
Függvényírás, top-down függvénytervezés

A saját függvénydefiníciókat is tartalmazó C program általános szerkezete:

előfordítónak szóló direktívák (#)
saját függvények prototípusa
main függvény definíciója
saját függvények definíciója

1. feladat: Írja meg egy egész szám ellenőrzött beolvasását megvalósító, valamint az egész szám faktoriálisát kiszámító függvényeket.

```
void int_beolvas(int* x);
int faktor(int x);

int main() {
    int szam;
    int_beolvas(&szam);
    printf("%d! = %d\n", szam, faktor(szam));
    return 0;
}

void int_beolvas(int* x) {
    char ch;
    int ok;
    do {
        ok = 1;
        printf("Adj meg egy számot: ");
        if(scanf("%d", x)!=1) {
            printf("Hibás input\n");
            while ((ch=getchar()) != '\n');
            ok = 0;
        }
    } while( !ok || *x<0 );
    return ;
}

int faktor(int x) {                                //pozitív egész szám vagy 0 faktoriálisát számítja ki
    int fakt=1;
    while(x>1) {
        fakt *= x;
        x--;
    }
    return fakt;
}
```

Önálló feladat: Írjon C programot az Euler szám ($e = 2,718$) kiszámítására az alábbi képlet alapján:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \dots$$

2. feladat: Két egész szám felcserélése függvény használatával.

```
void cserel(int *x, int *y);

int main() {
    int a = 5, b = 10;
    cserel(&a, &b);
    printf("a = %d, b = %d\n", a, b);
    return 0;
}

void cserel(int *x, int *y) {
    int tmp;
    tmp = *x;
    *x = *y;
    *y = tmp;
}
```

3. feladat: Írjon C programot, amely megmondja egy megadott számról h. tökéletes-e. Tökéletes szám az szám, amely megegyezik a nála kisebb osztóinak összegével (Pl.: $6=1+2+3$; 28, 496, 8128).

1. lépés:

```
int main() {
    int x;
    printf("Adj meg egy egész számot: ");
    scanf("%d", &x);
    if (tokeletes(x)) printf("\nA megadott szám tokeletes\n");
    else printf("\nA megadott szám nem tokeletes\n");
    //printf("%s", tokeletes(x) ? "Tokeletes szám" : "Nem tokeletes szám");
    return 0;
}
```

2. lépés:

```
int tokeletes(int szam) {
    if (szam==kisebbosztoszeg(szam))
        return 1;
    return 0;
}
```

3. lépés:

```
int kisebbosztoszeg(int szam) {
    int osszeg, osztos;
    osszeg=0;
    for (osztos=1; osztos<=szam/2; osztos+=1) {
        if (szam%osztos == 0)
            osszeg += osztos;
    }
    return osszeg;
}
```

4. feladat: Megadott intervallumban keresse meg a barátságos számpárokat. A barátságos számpároknál az egyik szám önmagánál kisebb osztóinak összege a másik számmal egyenlő és fordítva. Pl.: (220;284) (1184;1210) (2620;2924) (5020;5564) (6232;6368)

1. lépés:

```
int main() {
    int alsohatar, felsohatar, i, j;
    int_beolvas(&alsohatar);
    do {
        int_beolvas(&felsohatar);
    } while (felsohatar < alsohatar);
    for (i=alshatar; i<=felsohatar; i++){
        for (j=i+1; j<=felsohatar; j++) {
            if (baratsagos(i, j)==1)
                printf("(%d,%d)\n", i, j);
        }
    }
    return 0;
}
```

2. lépés:

```
void int_beolvas(int *hatar) {
    //lásd 1. feladat
}
```

3. lépés:

```
int baratsagos(int a, int b) {
    if (a==kisebbosztos_osszeg(b) && b==kisebbosztos_osszeg(a)) return 1;
    return 0;
}
```

4. lépés:

```
int kisebbosztos_osszeg(int szam) {
    int osszeg, osztos;
    osszeg=0;
    for (osztos=1; osztos<=szam/2; osztos+=1) {
        if (szam%osztos == 0)
            osszeg += osztos;
    }
    return osszeg;
}
```

5. feladat: Négyzetszámok keresése, kiírása.

a) Adott N-ig számolja meg a négyzetszámokat.

1. lépés:

```
int main() {
    int N, i, db=0;
    scanf("%d", &N);
    for (i=1; i<=N; i++) {
        if (negyzetszam_e(i)==1)
            db++;
    }
    printf("%d", db);
    return 0;
}
```

2. lépés:

```
int negyzetszam_e(int szam) {  
    int gyok=sqrt((double)szam);    //a szám négyzetgyökét int-re kerekítjük  
    if (gyok*gyok==szam) return 1;  
    return 0;  
}
```

/* Algoritmikus megoldás: a szám osztói között ha találok olyat, amelyik önmagával megszorozva egyenlő a számmal, akkor a szám négyzetszám. */

```
int negyzetszam_e(int szam) {  
    int oszto;  
    for(oszto=1; oszto<=szam; oszto++) {    \\ az 1 négyzetszám  
        if(szam%oszto==0) {  
            if(oszto*oszto==szam) {  
                return 1;    \\ négyzetszám  
            }  
        }  
    }  
    return 0;    \\ nem négyzetszám  
}
```

b) Az első N db négyzetszám kiírása.

1. lépés:

```
int main() {  
    int N, i=1, db=0;  
    scanf("%d", &N);  
    while (db<N){  
        if (negyzetszam_e(i)==1) { printf("%d", i); db++; }  
        i++;  
    }  
    return 0;  
}
```

2. lépés: a *negyzetszam_e* függvény definíciója ugyanaz mint fent

c) Az összes N jegyű négyzetszám kiírása.

1. lépés:

```
int main() {  
    int N, i;  
    scanf("%d", &N);    //számjegyek száma  
    /* math.h hatványozó függvénye: double pow(double alap, double kitevo); */  
    for (i=pow(10.0, (double)(N-1)); i<pow(10.0, (double)N); i++) {  
        if (negyzetszam_e(i)==1)  
            printf("%d\n", i);  
    }  
    return 0;  
}
```

2. lépés: a *negyzetszam_e* függvény definíciója ugyanaz mint fent

Házi feladat:

1. Írjon programot, amely kiszámítja egy ellenőrzött beolvasott egész szám négyzetét. Az egyes részfeladatokat külön függvényben valósítsa meg!
2. Írjon C programot, ami meghatározza egy ellenőrzött beolvasott szám számjegyeinek az összegét.
3. Az 5. feladat mintájára írjon olyan C programot, amely N ellenőrzött beolvasását követően kiírja
 - N-ig az összes tükörszámot;
 - az első N db tükörszámot;
 - az N-jegyű tükörszámokat.Tükörszám az, amelyik megegyezik a fordítottjával.
4. Az összes N-jegyű Armstrong-szám kigyűjtése. Armstrong-számnak nevezünk egy n jegyű számot, ha minden számjegyét az n -edik hatványra emelve és összeadva, az eredeti számot kapjuk. Ilyenek az egyjegyű számok és 153; 370; 371; 407; 1634; 8208; 9474; stb.

```
beolvas n
db←0
amíg n<>0 végezd el
| db←db+1
| n←[n/10]
■
kiír db
```

Számjegyek száma

```
beolvas n
osszeg←0
amíg n<>0 végezd el
| osszeg←osszeg+n%10
| n←[n/10]
■
kiír osszeg
```

Számjegyek összege

```
beolvas n
ford←0
amíg n<>0 végezd el
| ford←ford*10+n%10
| n←[n/10]
■
kiír ford
```

Szám megfordítása

5. A 4. gyakorlat rajzoló algoritmusait valósítsa meg függvényként. Pl. az N magasságú háromszög kirajzolását megvalósító program kódja:

```
#include <stdio.h>
void haromszograjzolo(int magassag);
int main() {
    int m;
    printf("Adja meg a háromszög magasságát: ");
    scanf("%d", &m);
    haromszograjzolo(m);
    return 0;
}
void haromszograjzolo(int magassag) {
    int sor, i;
    for (sor=1; sor<=magassag; sor++) {
        for (i=1; i<=magassag-sor; i++) printf("%c", ' '); // space kiírása
        for (i=1; i<=sor+(sor-1); i++) printf("%c", '*'); // *-ok kiírása
        printf("\n"); // sortörés
    }
    return ;
}
```

6. Vegye elő a korábbi gyakorlatokon fejlesztett Kalkulátor programot. Módosítsa úgy, hogy a végrehajtandó aritmetikai műveletek eredményét külön függvényekben számítsa ki.
7. Írjon C programot, amely külön függvényben ellenőrzött módon beolvas egy hatványalapot, ill. egy kitevőt. Írja meg a hatványozást megvalósító függvényt.