Adatok beolvasása

Az alább leírtak a gyakorlatban, pl. a nagyházi készítésekor fontosak. ZH-ban nem kell foglalkoznunk az ilyen problémák kiküszöbölésével, ZH-ban mindig azt feltételezzük, hogy a felhasználó helyes adatokat adott meg, kivéve, ha a feladat másképpen nem szól. (Akkor sem azt kéri a feladat, hogy a szöveget és a számot különböztessük meg, hanem pl. azt, hogy a felhasználó pozitív számot adott-e meg.)

Ha számot akarunk beolvasni, a scanf("%d",&x); beolvasás nem tudja kezelni azt a helyzetet, ha a felhasználó pl. betűt ad meg: ilyenkor a scanf a visszatérési értékével jelzi, hogy nem sikerült a beolvasás (ha jó a beolvasás, a scanf 1-et ad vissza, ha nem jó, akkor 0-t vagy EOF-ot), és a betűt a bemeneten hagyja, így egy következő scanf ugyanazt a betűt kapja. A probléma orvosolható, ha szám helyett mindig szöveget olvasunk be, és aztán a szöveget alakítjuk át számmá a sscanf függvény segítségével.

Egy másik probléma, ha szöveget kell beolvasnunk. A char t[100]; scanf("%s", t); beolvasás csak a következő szóközig olvas. A másik probléma, hogy ha a felhasználó a tömb méreténél hosszabb egybefüggő szöveget ad meg, akkor a scanf a tömb vége utáni memóriahelyekre is írni fog, ami a program hibás működését eredményezi. Mindkét problémát megoldhatjuk a teljes sort beolvasó fgets függvény használatával.

A mintaprogram:

```c
#include <stdio.h>
#include <string.h>

int main(){
    char tarolo[100];
    int x;

    // int beolvasása:

    printf("Adj egy egesz szamot: ");
    do { fgets(tarolo,100,stdin); } while( sscanf( tarolo, "%d", &x) != 1 );
    printf("%d\n",x);

    // szöveg beolvasása

    printf("Adj egy szoveget!\n");
    fgets(tarolo,100,stdin);
    printf("Ebben van ujsor karakter: ***%s***\n",tarolo);
    char * enter = strchr(tarolo,'\n');
    if(enter!=NULL)
        *enter = '\0'; // az újsor jelet felülírjuk a lezáró 0-val
    printf("Ebben nincs ujsor karakter: ***%s***\n",tarolo);

    return 0;
}
```

Az egész szám beolvasását egy ciklusban végezzük. Először az fgets segítségével beolvasunk egy teljes sornyi szöveget, majd a beolvasott szöveget a sscanf-fel konvertáljuk egész számmá. (Ha két számot akarunk egyszerre beolvasni, akkor `while( sscanf( tarolo, "%d%d", &x, &y) != 2 );` lesz, mert a scanf függvények visszatérési értéke a sikeresen beolvasott változók száma, vagy hiba esetén EOF.) Ugyanez a kifejezés természetesen valós számok beolvasására is használható, ha pl. %lf-et írunk.

A fgets paraméterei: egy karaktertömb, a tömb mérete, és a fájl azonosító, ami jelen esetben az stdio.h-ban definiált stdin, ami a szabványos bemenetet jelenti. Az fgets maximum tömbméret-1 db karaktert olvas be, és a beolvasott szöveget a sztringet lezáró '\0'-val lezárja. Ha a felhasználó hosszabb szöveget adott meg, akkor a maradékot a következő beolvasáskor kapjuk meg.

A szöveg beolvasásakor az okoz problémát, hogy a fgets a sort lezáró újsor karaktert is beolvassa. Az újsor karakter mindig a beolvasott sor végén van, azaz ha kicseréljük egy lezáró nullára, akkor valóban csak az újsor karaktert töröljük a szöveg végéről, más kárt nem okozunk. Az újsor karaktert pl. ar strchr függvénnyel kereshetjük meg, a fenti módon. Elképzelhető, hogy a beolvasott szövegben nincs újsor karakter, ekkor az strchr NULL pointert ad vissza (ez akkor lehet, ha a felhasználó a tömb méreténél hosszabb sort adott meg, vagyis az fgets nem olvasta be az egész sort). Ezért nem szabad *strchr(tarolo,'\n')='\0'-t írni, hanem a pl. programban látható módon a NULL esetét kezelni kell.

Még egy dologra érdemes figyelni. A sima scanf("%d", &x) beolvasás beolvassa a számot, de az utána nyomott ENTER-t (újsor karaktert) a bemeneten hagyja. Ha a következő beolvasás is számot olvas, akkor nincs gond: a scanf tetszőleges számú whitespace karaktert (szóköz, tabulátor, újsor) átugrik ilyen esetben. Ha viszont a szám beolvasása után pl. fgets-se egy sornyi szöveget olvasunk, vagy scanf("%c", &ch)-val karaktert olvasunk be, akkor mindkettő az újsor karaktert tekinti a beolvasás értékének. Megoldás: vagy ne keverjük a scanf és a fgets beolvasásokat, vagy a fgets elé tegyünk egy plusz fges-t, ami az ENTER-t olvassa be.