Tartalomjegyzék

Adattípusok, függvények

Egyszer? adattípusok

Eddig két egyszer? adattípussal ismerkedtünk meg: *int* (egész szám) és *float* (tört szám). Most lesz még néhány típusunk amiket egy C programban használhatunk:

- *uint* : unsigned int, vagyis el?jel nélküli egész.
- *short* : rövid egész
- *long* : hosszú egész
- *char* : karakter típus
- *double*: dupla pontosságú lebeg?pontos szám (mint a float, csak több byte-os, ezért szélesebb határok között tud számokat tárolni)
- *void* : ez a "http://wiki.math.bme.husemmi"http://wiki.math.bme.hu típus, függvények deklarációjánál kell pl használni, ha semmit nem ad vissza a függvény

A határok, hogy egy-egy adott típusú változó mekkora értékeket tud tárolni, a limits.h -ban vannak (ezt kell a program elején *include*-olni). A konkrét értékek architektúrafügg?k, vagyis egy másik típusú gépen mások lehetnek a határok.

Egy kis program, ami kiír néhányat a határok közül (limits.c):

```
#include <float.h>
#include <limits.h>
#include <stdio.h>

int main() {
    printf("SHORT max:\t %u \n", SHRT_MAX);
    printf("INT max:\t %d \n", INT_MAX);
    printf("UINT max:\t %u \n", UINT_MAX);
    printf("LONG max:\t %ld \n", LONG_MAX);
    printf("CHAR min:\t%d \n", CHAR_MIN);
    printf("CHAR max:\t %d \n", CHAR_MAX);
    printf("FLOAT min abs value:\t %1.45lf (kb. 10 ^ -37)\n", FLT_MIN);
```

```
    return 0;
}
```

A *limits* programunk kimenete egy 32 bites gépen:

```
   SHORT max:    32767
   INT max:      2147483647
   UINT max:     4294967295
   LONG max:     2147483647
   CHAR min:    -128
   CHAR max:     127
   FLOAT min abs value: 0.00000000000000000000000000000000000011754944 (kb 10 ^ -37)
```

A *limits* programunk kimenete egy 64 bites gépen (nézd meg a *long* típust):

```
   SHORT max:    32767
   INT max:      2147483647
   UINT max:     4294967295
   LONG max:     9223372036854775807
   CHAR min:    -128
   CHAR max:     127
   FLOAT min abs value: 0.00000000000000000000000000000000000011754944 (kb 10 ^ -37)
```

A fenti limits.c kódban az is látszik, hogy ha *printf()* vagy *scanf()* függvénnyel szeretnéd a típusokat kiírni/beolvasni, milyen kódokat kell használni. De azért foglaljuk össze:

- *uint* : %u
- *short* : %d
- *long* : %ld
- *char* : %c
- *double*: %lf

Túlcsordulás

Próbáljuk ki, mi történik ha feszegetjük a határokat: Legyen két *int* változónk, amiket összeadva már nagyobb értéket kapunk a tárolható maximálisnál:

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

int main() {
    int egyik = INT_MAX;
    printf("egyik: %d \n", egyik);
    int masik = 1;
    printf("masik: %d \n", masik);
    int osszeg = egyik + masik;
    printf("osszeg: %d \n", osszeg);
    return 0;
}
```

És a kimenet:

```
  egyik: 2147483647
  masik: 1
  osszeg: -2147483648
```

Egy jó nagy negatív számot kaptunk. Ezt hívják túlcsordulásnak. Ezért kell körülbelül el?re megbecsülni hogy a programunk mekkora számokkal fog dolgozni, és ha sejtjük hogy nagyon nagy (vagy nagyon kicsi) számok is el?fordulhatnak, akkor inkább *long* ill. *double* típusokat használjunk *int* és *float* helyett. Ennek az lesz az ára, hogy kicsit több memóriát fog foglalni a programunk a futása közben.

Nincs külön igazságérték-típus

A C-re épül? kés?bbi nyelvekben már szokott lenni *bool* (C++) vagy *boolean* (Java) típus ami csak két értéket vehet fel (Pythonban is léteztek el?re definiáltan a 'True' és 'False' értékek).

C-ben nincs ilyen, itt minden ami nem nulla, az "http://wiki.math.bme.huigaz"http://wiki.math.bme.hu, és ami nulla az "http://wiki.math.bme.huhamis"http://wiki.math.bme.hu.

Kiegészít? anyag

Kíváncsiaknak b?vebben (nem része a tananyagnak):

http://www.cplusplus.com/reference/clibrary/climits/
http://www.cplusplus.com/reference/clibrary/cfloat/
http://www.hit.bme.hu/~vitez/Progalap1/Progalap04.pdf

Tömbök vagy adatvektorok (array)

A tömbök segítségével azonos elemekb?l álló adathalmazt tárolhatunk. A C-beli tömbök kicsit hasonlítanak a Python-ban tanult listákra, de azért sok fontos különbség is lesz. (Pl az egyik hogy Python listában mindegy volt a típus, C-ban viszont csak azonos típusú elemek lehetnek egy tömbben).

Általános alakjuk:

típus tömbnév[elemszám];

Például, egy 11 elem?, alfa nev? tömb deklarációja, amiben hosszú egészek lehetnek, valamint egy béta tömb ami 23 karaktert tárolhat:

```
int alfa[11];
char beta[23];
```

A deklarációkor már lefoglalódik a tömböknek a hely a memóriában, ezért el?re meg kell mondanunk hogy hány elemet szeretnénk (legfeljebb) tárolni.

Egy pici kódrészlet, ami feltölti a *beta* karaktertömbünk els? 4 elemét (0-tól sorszámozódnak az elemek, az indexeket szögletes zárójelbe kell tenni):

```
beta[0] = 'a';
beta[1] = 'l';
beta[2] = 'm';
beta[3] = 'a';
```

Általában valamilyen ciklussal célszer? feltölteni vagy végigolvasni egy tömböt. Praktikus ha a tömb mérete megvan egy változóban. Feltöltés (és rögtön kiírás is) for ciklussal:

```
int tomb_merete = 23;
char karakterek[tomb_merete];
int i;
for (i=0; i<tomb_merete; i++) {
    karakterek[i] = 'a';
    printf("%c\n", karakterek[i]);
}
```

Sajnos (a Python-os listákkal ellentétben) itt nincs mód arra hogy egy tömb méretét elkérjük, így ha egy függvénynek átadsz egy tömböt, akkor add át vele a tömb méretét is, ha arra a függvénynek szüksége van.

Függvények

Ahogy egyre bonyolultabb programokat írunk, célszer? függvényekbe szervezni a m?ködést. Számos el?nye van a függvények használatának:

- a program logikai strukturálása
- áttekinthet?bb, olvashatóbb lesz a kód
- elkerüljük a kódismétlést: kevesebbet kell gépelni, ill. csak egy helyen kell javítani ha hiba van (a copy-paste stílusú programozás hosszútávon nem m?ködik)
- minél általánosabban megírt egy függvény, annál valószín?bb hogy máshol (másik programban) is használható lesz

C nyelven egy függvény deklarációja általánosan így néz ki:

visszatérési_típus függvénynév(param_típus1 param_név1, param_típus2 param_név2, ...);

Nézzünk három példát függvény deklarálására. A második nem ad vissza értéket, ezért *void* a visszatérési típusa. Az utolsónak pedig nincs bemen? paramétere de a zárójelek akkor is kellenek:

```
int maximum_ertek(int tomb[], int meret);
void irj_ki_egy_szamot(int szam);
int random_szam();
```

És most "http://wiki.math.bme.hutöltsük is ki"http://wiki.math.bme.hu az els?t, vagyis definiáljuk:

```
int maximum_ertek(int tomb[], int meret) {
    int max = INT_MIN;  // include limits.h !
    int i;
    for (i=0; i<meret; i++) {
        if (tomb[i] > max) {
            max = tomb[i];
        }
    }
    return max;  // figyelni kell mindig hogya fv visszatérési típusával megegyez? típusú értéket
}
```

A másodikat:

```
void irj_ki_egy_szamot(int szam) {
    printf("%d\n", szam);
    // ez nem ad vissza értéket, ekkor a "return;" parancs el is hagyható
}
```

Paraméterek, argumentumok

C-ben csak érték szerinti paraméter-átadás van (minden ellenkez? híreszteléssel szemben), ez azt jelenti, hogy ha egy függvényt meghívunk bizonyos argumentumokkal, akkor azok értékei átmásolódnak a függvény paramétereibe. Magyarul nem tudjuk a függvény törzséb?l az aktuális argumentumokat megváltoztatni, mivel maga a függvény csak a másolaton dolgozik.

A fentiek értelmezéséhez segítség: A "http://wiki.math.bme.huparaméter"http://wiki.math.bme.hu a hely és az "http://wiki.math.bme.huargumentum"http://wiki.math.bme.hu a bele helyettesített érték. Egy példán keresztül:

```
void Foo(int i, float f) {
    // Do things
}

void Bar() {
    int anInt = 1;
    Foo(anInt, 2.0);
}
```

Itt *i* és *f* paraméterek, "http://wiki.math.bme.huanInt"http://wiki.math.bme.hu és "http://wiki.math.bme.hu2.0"http://wiki.math.bme.hu pedig argumentumok.

Olvasnivaló függvényekhez:

http://progtut.net/index.php?p=Article&id=113

Ellen?rz? kérdések

- Sorold fel a C nyelv legalább 4 egyszer? típusát!
- Deklarálj egy 20 elem? tömböt, ami karatereket tartalmazhat!
- Mit jelent ha egy függvénynek *void* a visszatérési típusa?
- Mit jelent hogy C-ben "http://wiki.math.bme.huérték szerint"http://wiki.math.bme.hu adódnak át a paraméterek a függvényeknek?