4. gyakorlat

Ezen a gyakorlaton az időkezelésről, és a véletlenszám generálásról esett szó.

Elméleti alapok az eheti gyakorlathoz

Idő, `<time.h>`

Az idő számítógépes reprezentálása nem egyszerű feladat. Ha a szokásos év-hónap-nap-óra-perc-másodperc alakot használjuk, bonyolult lesz az idő számlálása, mert minden 60. másodperc után percet kell váltani, stb. Nem is beszélve a szökőévekről, szökőmásodpercekről, időzónákról, téli/nyári időről. Ha egyszerűbb alakot használunk, akkor viszont mi nem tudjuk könnyedén értelmezni a tárolt időt.

A C-ben úgy oldották meg a kérdést, hogy mindkét alakot alkalmazzák. A számítógép ún. **UNIX timestamp** formában tárolja az adatokat, viszont egy függvény készséggel átalakítja ez olvasható adatstruktúrává vagy szöveggé.

UNIX timestamp

A számítógépek számokkal szeretnek a leginkább dolgozni, ezért az időt is ebben mérik. Egyszerűen az 1970 január 1. 0:00:00 óta eltelt másodpercek számát tárolják, integer vagy double számként. (A C integert használ). Ennek az adatnak van egy saját adattípusa, a `time_t`. Ez valójában egy 32 vagy 64 bites egész szám, de időtárolás esetén ezt a típust kell használnunk.

`struct tm`

Ha szükségünk van arra az információra, hogy milyen év/hónap/nap/óra/perc/másodperc van éppen, az időt lekérhetjük egy `struct tm` típusú adatstruktúrában is. Ez a fenti adattagokat külön tartalmazza.

```
struct tm
{
    int tm_sec;
    int tm_min;
    int tm_hour;
    int tm_mday;
    int tm_mon;
    int tm_year;
    int tm_wday;
    int tm_yday;
    int tm_isdst;
}
```

Az adatstruktúrák használatát később fogjuk tanulni.

> ***Megjegyzés*** Programjainkban amikor csak lehetséges, használjuk az ISO 8601 szabvány szerinti időformátumokat. A szabványos formátum a YYYY-MM-DD hh:mm:dd, például 2019-09-01 13:07:19.

Random

A számítógépek determinisztikus gépek, önmagukban képtelenek valószínűséget bevinni a számításba. A véletlenszám-generáláshoz ezért ún. pszeudorandom elemeket használnak, amelyek ugyan nem ténylegesen valószínűségi változók, de eléggé konvuláltak ahhoz hogy ne lehessen őket reprodukálni.

A véletlen elemet **seednek** hívjuk. Sok nyelv alapból az időt használja seedként, a C-nél ezt nekünk kell beállítanunk. Ugyanazon seed esetén ugyanazokat a számokat fogja kiköpni a generátorunk, ezért fontos az egyedi seed használata.

A random függvények a `<stdlib.h>` könyvtárban vannak.

Modulo operátor (%)

A modulo a maradékos osztás maradékát számolja ki.

```
15 % 3  // 0
16 % 3  // 1
17 % 3  // 2
```

Gyakran használjuk arra, hogy oszthatóságot vizsgáljunk. Ha `a % b == 0`, akkor `a` osztható `b`-vel.

Az elmélet után következzenek a ténylegesen használható függvények.

`time()`

A `time()` függvény a pillanatnyi időt adja vissza `time_t` típussal, valamint az argumentumként átadott `time_t` változóba bele is írja.

Kétféleképpen lehet használni:

```
// 1
time_t ido;
ido = time(NULL);

// 2
time_t ido;
time(&ido);
```

Az első módszer az ajánlott. Ne felejtsük el a NULL-t átadni argumentumként. (Ez azt jelenti hogy ne írja sehova az eredményt, csak adja át visszatérési értékként.)

A változó tartalma valami ilyesmi lesz: 1565937898. Ne lepődjünk meg, hisz ez egy UNIX timestamp.

Ezzel a számmal végezhetünk műveleteket (ha akarunk), de leggyakrabban csak kiírni szoktuk. A kiírás végezhető számként is:

```
printf("A pontos idő %lli.\n", ido);
```

Viszont így a kimenet nem túl barátságos.

```
A pontos idő 1565937898.
```

`ctime()`

A `<time.h>` tartalmazza a `ctime()` idő-kiíró függvényt. A címével átadott `time_t` típusú időből készít olvasható szöveget, melynek a végére még soremelést is tesz.

```
printf("A pontos idő %s", ctime(&ido));
```

```
A pontos idő Fri Aug 16 08:44:58 2019
```

> ***Megjegyzés*** A szövegek kiírásnál a `"%s"` formátumot használjuk.

`rand()`

A `rand()` egy véletlen egész számot generál `0` és `RAND_MAX` között. (A `RAND_MAX` egy konstans, értékét a fordító szabja meg).

```
int random_szam = rand();
printf("A random szám: %i\n", random_szam);
```

Ha fenti kódot lefuttatod párszor, feltűnik, hogy mindig ugyanazt a számot adja ki. Ez azért van, mert az alapértelmezett seed `1`.

`srand()`

A változatosabb seed elérése érdekében állítsuk be az aktuális időt seedként. A seed beállítására a `srand()` függvény szolgál, melynek egy argumentuma az `unsigned int` típusú seed. Ezután a `rand()` már tényleg véletlen számokat ad.

```
unsigned int seed = time(NULL);
srand(seed);

printf("A random szám: %i\n", rand());
```

A `time_t` típus automatikusan konvertálódik `unsigned int` típusra.

Random egész számok bizonyos intervallumon

Ha egész számokat akarunk kapni a és b között, akkor egyszerű megoldásként használhatjuk a modulo operátort.

Az általános formula

```
int szam = rand() % (b - a + 1) + a
```

Például 1 és 5 között így kapunk véletlen számokat:

```
int dobas = rand() % 5 + 1
```

A `rand()` egész szám öttel való maradéka lehet 0, 1, 2, 3 vagy 4. Ehhez egyet hozzáadva kapunk 1 és 5 közti számokat.

Random valós számok

A `rand()` maximális eredményét a `RAND_MAX` konstans tartalmazza. Ezzel tudjuk normálni a véletlen számot.

```
int veletlen = rand();
double normalt_veletlen = (double)veletlen / RAND_MAX;
```

Az eredmény egy 0 és 1 közötti szám lesz.

Feladatok

1. Készíts Twister pörgető programot!

Válasszon egy random testrészt (jobb kéz, jobb láb, bal kéz, bal láb) és egy random színt (piros, sárga, kék, zöld)!

A testrészeket és a színeket kódold számokkal (1, 2, 3, 4)! A program két véletlen számot választ 1-4 között, az ennek megfelelő testrészt és színt kiírja.

Várt kimenet:

```
jobb láb a kékre!
```