

El?z? - Fel - Következ?

A múlt heti gyakorlathoz hasonlóan ismét a **main** függvényben és esetleg még más *static* függvényekben fogunk dolgozni.

Tartalomjegyzék

- 1 Vezérlési szerkezetek
 - ◆ 1.1 Sakktábla
 - ◆ 1.2 Fibonacci
 - ◆ 1.3 Intervallumban
- 2 Összetettebb feladatok
 - ◆ 2.1 Csoki osztás
 - ◆ 2.2 Lólépés

Vezérlési szerkezetek

Sakktábla

Rajzolj ki egy NxN-es sakktábla mintát, ahol X-szel jelöljük a fekete mez?ket, és üresen hagyjuk (egy szóköz) a fehéreket. Nem kell keretet adni a táblának.

Segítség for ciklushoz:

```
for(int i = 0; i < 10; i++)
{
    ...
}
```

Módosítsd a kódot, hogy ha eddig a **main**-ben dolgoztál, akkor emeld ki a programrészt függvénybe, hogy lehessen bármilyen NxN-es sakktáblát kirajzolni. (Refactor -> Extract new method)

Fibonacci

Egy kis bevezető tömbökhöz:

A tömbökre lehet úgy gondolni, mint a listákra python-ból, de a kettő közt rengeteg különbség van. Jelenleg csak fix méretű tömbökkel fogunk dolgozni, változó méretűekre később lesz példa. Egy n hosszú tömböt 0-tól n-1-ig indexelünk. Az elemeit egyenként lehet módosítani, vagy lekérdezni. A következő kódrészlet belerakja egy tömbbe 0-tól 19-ig a számok négyzetét:

```
int[] t = new int[20]; // Itt hozzuk létre a tömböt, előre megadjuk a méretet
t[5] = 5 * 5; // Így tudunk egy elemnek értéket adni
t[6] = t[5] + 11; // Vagy felhasználni értékként
for(int i = 0; i < 20; i++) { // Ez a ciklus végzi el a munkánkat
    t[i] = i * i;
}
```

Most írjunk egy programot, amely az első 100 fibonacci számot beteszi egy tömbbe.

Intervallumban

Írjatok függvényt, mely egy **float** tömböt és két **float** számot kap bemenetül. Egy **float** tömböt ad vissza, melyben a két másik szám közé eső elemek szerepelnek. Eleinte megírhatjuk a függvényt úgy, hogy feleslegesen nagy tömböt foglal le, de ha már működik, gondoljuk végig hogyan lehetne megoldani úgy, hogy pontosan akkora méretű tömböt adjunk vissza, mint amennyi elem esik majd az intervallumba.

Összetettebb feladatok

Csoki osztás

Van 3 matematikus, találnak valahány tábla csokit. Úgy döntenek, hogy majd másnap reggel osztoznak testvériesen, viszont egyikőjük se bízik a másikban. Egymást követve mindegyik felébred éjszaka, elosztja a csokit 3 felé, konstatálja, hogy igazságos osztoszkodás után marad 1 tábla, ezt odaadja a padló alatt lapuló fizikusnak, majd a saját részét elviszi, a maradékot ott hagyja. Ezt eljátsza egymás után mindegyik matematikus és mind ugyanazt tapasztalja, 1 csoki maradt meg mindegyiknél, ha igazságosan osztotta el és mind a fizikusnak adta. Legkevesebb hány tábla csokiból kezdtek?

Ha ezzel megvagyunk, akkor írjuk át a programot, hogy n matematikussal is működjön.

Lólépés

Egy 8x8-as sakktabla bal felső sarkában áll egy ló, innen indulva számoljuk ki, hogy legkevesebb hány lépésben érne el a többi mezőre.

Ha ez megvan, akkor írjunk olyan függvényt, ami $N \times N$, vagy akár $N \times M$ -es sakktablára megcsinálja ugyanezt.

Előző? - Fel - Következő?