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

Tartalomjegyzék

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 bevezetõ tömbökhöz:

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

```
int[] t = new int[20];     // Itt hozzuk letre a tombot, elore megadjuk a meretet
t[5] = 5 * 5;             // Igy tudunk egy elemenek erteket adni
t[6] = t[5] + 11;         // Vagy felhasznalni ertekkent
for(int i = 0; i < 20; i++) { // Ez a ciklus vegzi el a munkankat
    t[i] = i * i;
}
```

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

Intervallumban

Írjatok függvényt, mely egy **float** tömböt és két **int**-et kap bemenetül. Egy **float** tömböt ad vissza, melyben a két **int** közé esõ elemek szerepelnek. Eleinte megírhatjuk a függvényt úgy, hogy feleslegesen nagy tömböt foglal le, de ha már mûködik, gondoljuk végig hogyan lehetne megoldani úgy, hogy pontosan akkora méretû 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 egyikõjük 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 mûködjön.

Lólépés

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

Ha ez megvan, akkor írjunk olyan függvényt, ami NxN, vagy akár NxM-es sakktáblára megcsinálja ugyanezt.