Név: Gyóni Ákos Neptun kód: PKK65Q

Reversi

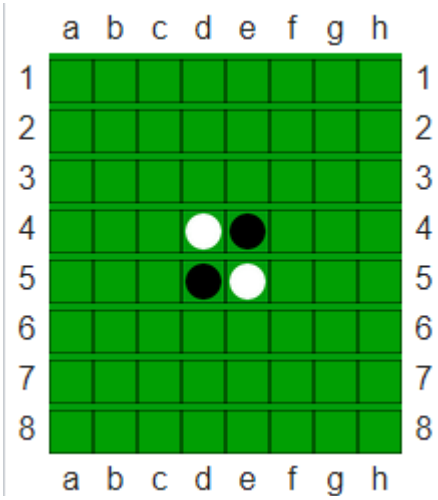
Készítsen objektumot a reversi (Othello) nevű játék megvalósításához! Az objektum tárolja a játék állását és "ismerje" a szabályokat, azaz automatikusan "forgassa" át a megfelelő korongokat!

Demonstrálja a működést külön modulként fordított tesztprogrammal! A játék állását nem kell grafikusan megjeleníteni, elegendő csak karakteresen, a legegyszerűbb formában! A megoldáshoz ne használjon STL tárolót!

Feladat

A program a reversi játékot valósítja meg két játékos számára. A konzolon megjelenik a 8x8-as tábla (karakterekkel ábrázolva) és az utolsó sorba az, hogy most melyik játékosnak kell lépnie. Aki jelenleg következik az beír a szöveges bemenetre egy betű szám kombinációt a pálya sorait és oszlopait reprezentálva (pl A1), ahova rakni szeretne és ha az lehetséges akkor a program megvalósítja azt a lépést.

Szabályok



Alapállás

A játék az eredeti Reversi szabályok alapján működik, azaz:

A játék klasszikus alapállása szerint a két játékos két-két figurát helyez a tábla közepére.

A két játékos felváltva lép úgy, hogy lerak egy korongot valamelyik üres mezőre. Csak olyan helyre rakhat, ahol meg tudja fordítani az ellenfél legalább egy korongját; ez úgy lehetséges, hogy az ellenfél korongja az éppen letett korong és a játékos másik korongja között van, egyenes vonalban vízszintesen, függőlegesen vagy átlósan.

Előfordul, hogy a soron levő játékos nem tud lépni, mert nincs olyan üres mező, ahova rakva megfordíthatja ellenfele legalább egy korongját. Ez nem jelenti a játék végét. A lépni nem tudó játékos passzol, s ellenfele lép. Akárhányszor lehet passzolni sorozatban is, ha az ellenfélnek sikerül úgy megfordítgatni

Forrás a szabályokhoz: https://mek.oszk.hu/00000/00056/html/138.htm
Forrás a képhez: https://en.wikipedia.org/wiki/Reversi

korongjainkat, hogy nekünk ne adjon lépéslehetőséget Passzolni azonban csak akkor szabad (és kell), ha nem tudunk lépni. Ha tudunk, akkor lépnünk kell.

A játék a következő esetekben ér véget:

-ha betelt a tábla;

- ha az egyik játékosnak elfogytak a korongjai (hiszen a szabályos lépéshez egy már táblán levő korong is kell);

- ha egyik játékos sem tud lépni.

Mindegyik esetben az győz, akinek több korongja van a táblán; ha egyenlő számúak, az eredmény döntetlen. A végeredményt és a pillanatnyi állást kiírja a program a tábla mellé.

Fontosabb algoritmusok leírása

Tábla

A LephetIranyba függvény ellenőrzi, hogy üres-e a mező, majd megnézi a megadott mezőn a megadott irányba, hogy arra átfordulna-e Korong. Ha átfordulna Korong, akkor igaz értéket ad vissza azaz lehetséges ez a lépés. Ha forditE igaz akkor meg is fordítja a korongokat abba az irányba.

Az IranybaKeres függvény megnézi a megadott mezőn a megadott irányba a korongot, ha sz színű akkor megfordít minden előzőt (ha forditE is igaz) és igazzal tér vissza, ha masik színű akkor tobább megy. (ha üres akkor hamis értékkel tér vissza), ez a függvény rekurzívan működik.

A TudLepni függvény megnézi hogy az sz játékosnak van-e lehetséges lépése a jelenlegi táblán.

A Lepes függvény ellenőrzi, hogy üres-e a megadott mező, végignézi a megadott mezőn a lehetséges nyolc irányba, hogy arra átfordulna-e Korong, ha átfordulna korong akkor igaz értéket ad vissza, azaz lehetséges ez a lépés, ha forditE igaz akkor meg is fordítja a korongokat és lerakja a lépést

Játék

A BemenetEllenorzessel függvény a megadott bemenetről beolvas egy sor és oszlop értéket, ezeket a megadott size_t változókba rakja, ha nem jól vannak megadva a változók, akkor jelzi a kimenetre és addig kérdez újra, amíg jót nem ad a felhasználó.

Forrás a szabályokhoz: https://mek.oszk.hu/00000/00056/html/138.htm
Forrás a képhez: https://en.wikipedia.org/wiki/Reversi

A Start függvényben megy a játék fő loopja, meghívása elindít egy játékot üres táblával és a Jatek kimenetere ábrázolja is, a játékosok lépéseit a konzolról olvassa be a BemenetEllenorzessel függvénnyel.

Megoldási vázlat

A feladatot először a Jatek osztály, és azon belül a fő loop(Start függvény) megírásával kezdtem, ami addig megy, amíg betelik a pálya, vagy az egyik játékos sem tud rakni.

Ezután megvalósítottam a Tabla osztályt, ami maga végzi az ellenörzést a lépéseken. A függvényei részben egymásra épülnek. Az IranybaKeres függvény rekurzívan megnézi egy irányba, hogy lehet-e arra lépni és ezt használja fel a LephetIranyba függvény, hogy leellenőrizze a tényleges lépést. Mindkettő függvény tud vagy fordítani vagy csak ellenőrizni is, a forditE változó értékétől függően. A Lepes függvény egy mezőről minden irányt megnéz, az előző két függvénnyel, a TudLepni pedig egy játékos számára az összes mezőt ellenőrzi a Lepes függvény nem fordító változatával.

A megjelenítést a Kimenet virtuális osztály valósítja meg, ezért lehet más fajta kimeneti módszereket(például: grafikus) is implementálni az alap program megváltoztatása nékül. Én jelenleg csak konzolos megjelenítést csináltam a Konzol osztállyal, ami a Kimenetből származik.

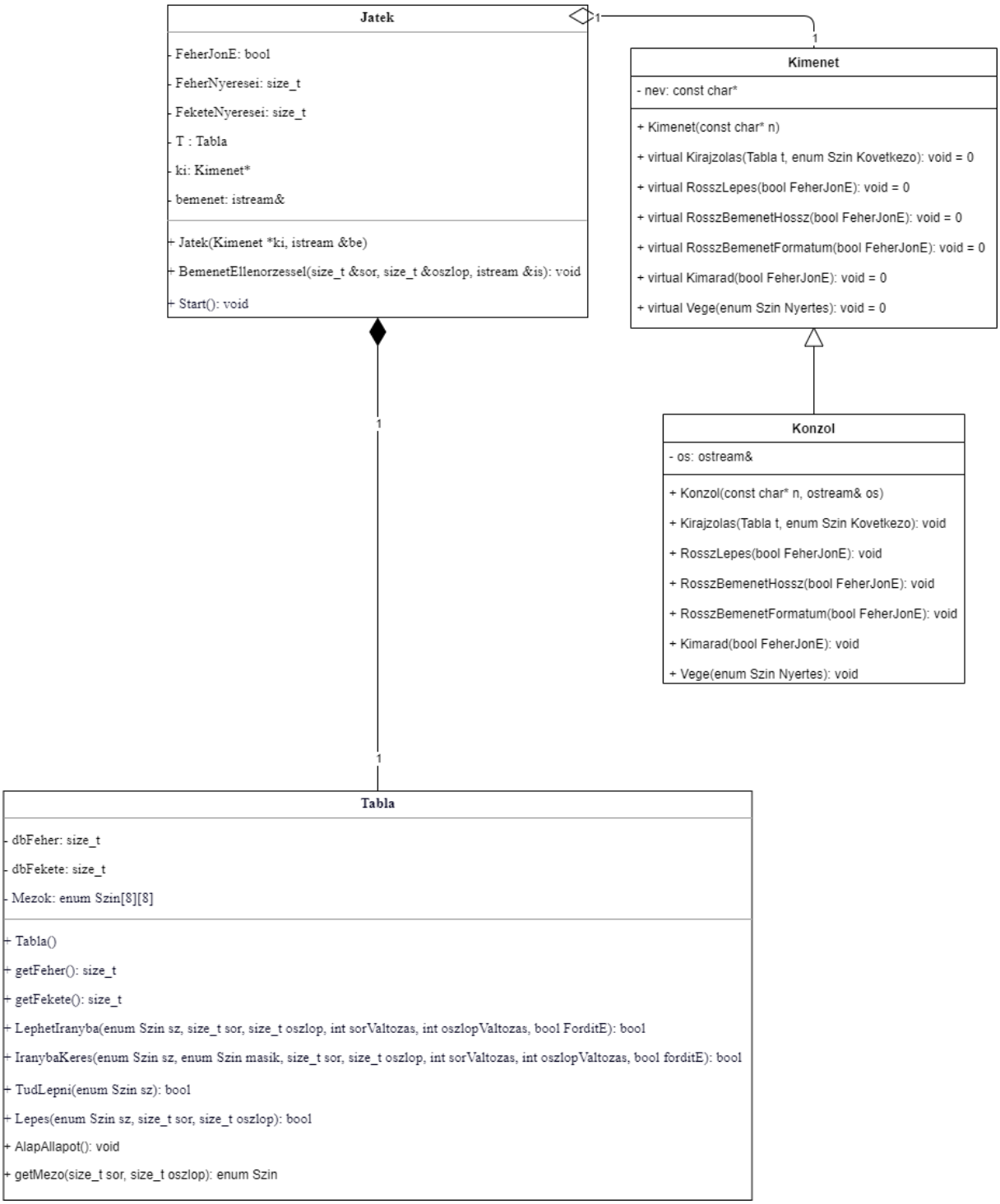
Tesztelési dokumentáció

A programot az 1. esetben, az egyik legrövidebb nyerési lehetőséggel tesztelem, ahol a fekete színű játékos nyer, mert az összes korong fekete lesz és így a fehér színű játékos nem tud rakni, ehhez a bemeneteket egy string streamből olvassa ki a program, amit úgy kezel mint a konzol bemenetet. A 2. esetben simán elindul a játék és a két játékos felváltva tud lépni, addig megy amíg véget nem ér a meccs, utána új tesztesetet lehet indítani vagy ki is lehet lépni 0-val. Memória szivárgás nincsen a programban.

Forrás a szabályokhoz: https://mek.oszk.hu/00000/00056/html/138.htm
Forrás a képhez: https://en.wikipedia.org/wiki/Reversi

Név: Gyóni Ákos Neptun kód: PKK65Q

UML diagram



Forrás a szabályokhoz: https://mek.oszk.hu/00000/00056/html/138.htm
Forrás a képhez: https://en.wikipedia.org/wiki/Reversi