# Részletes tervek

## Osztályok és metódusok tervei

### Crate

#### Felelősség

Egy ládát szimbolizál. Nyilvántartja a láda pozícióját.

#### Ősosztályok

Entity

#### Interfészek

-

#### Attribútumok

* **-**

#### Metódusok

* **+ push(Worker, Entity, Direction): boolean:** A láda eltol egy entitást a megadott irányban lévő szomszédos mezőre. Igazzal tér vissza, ha sikerült eltolni, egyébként hamissal.
* **+ pushByWorker(Worker, Worker, Direction): boolean:** Egy dolgozó tolja a ládát a megadott irányban lévő szomszédos mezőre, így a láda oda lép. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
* **+ pushByCrate(Worker, Crate, Direction): boolean:** Egy láda tolja a ládát a megadott irányban lévő szomszédos mezőre, így a láda oda lép. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
* **+ remove(): void:** A láda kikerül a raktárból.
* **+ visit(Worker, IVisitable): boolean:** A láda egy vizitálható objektumra (mezőre) lép. Igazzal tér vissza amennyiben sikerült a lépés, egyébként hamissal.

### Direction

#### Felelősség

Az egyes lépési irányokat tartalmazza, mint fel (UP), le (DOWN), balra (LEFT) és jobbra (RIGHT).

### *Entity*

#### Felelősség

Absztrakt osztály, ami az entitásokat, avagy mozgatható dolgokat reprezentálja.

#### Ősosztályok

-

#### Interfészek

IVisitor

#### Attribútumok

* **- curField: Field:** A mező, amelyen jelenleg az entitás tartózkodik.
* **- weight: Double:** Az entitás tömege.

#### Metódusok

* ***+ push(Worker, Entity, Direction): boolean:*** Megpróbál eltolni egy megadott irányban lévő mezőn tartózkodó entitást, szintén a megadott irányban. Átveszi a dolgozót, aki a láncot tolja. Igazzal tér vissza, amennyiben sikerült eltolnia az entitást, egyébként hamissal.
* ***+ pushByWorker(Worker, Worker, Direction): boolean:*** Az entitást megpróbálja eltolni egy dolgozó a megadott irányban. Átveszi a dolgozót, aki a láncot tolja és az entitást tolni próbáló dolgozót. Igazzal tér vissza, amennyiben a tolás sikeres volt, egyébként hamissal.
* ***+ pushByCrate(Worker, Crate, Direction): boolean:*** Az entitást megpróbálja eltolni egy láda a megadott irányban. Átveszi a dolgozót, aki a láncot tolja és az entitást tolni próbáló ládát. Igazzal tér vissza, amennyiben a tolás sikeres volt, egyébként hamissal.
* **+ step(Worker, Direction): boolean:** Az entitás lép a megadott irányban lévő szomszédos mezőre. Ha léphet a következő mezőre, de ott tartózkodik egy entitás, akkor megpróbálja őt eltolni. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
* **+ setCurField(Field): void:** Beállítja az entitás mezőjét.
* **+ getCurField(): Field:** Lekérdezi az entitás mezőjét.

### *Field*

#### Felelősség

Absztrakt osztály, ami a játéktér (raktár) egy mezőjét szimbolizálja.

#### Ősosztályok

-

#### Interfészek

IVisitable

#### Attribútumok

* **- curEntity: Entity:** Az entitás, ami jelenleg ezen a mezőn tartózkodik.
* **- slipFactor: Double:** A mező tapadásának mértéke.
* **- warehouse: Warehouse:** A raktár, amelyhez a mező tartozik.

#### Metódusok

* **+ visitByLifeCrate(Worker, LifeCrate): boolean:** Egy szívecskés láda a mezőre lép. Ugyanaz történik, mint amikor egy láda lép a mezőre.
* **+ getNeighbourField(Direction): Field:** Az mező megadja az adott irányban lévő szomszédját.
* **+ setEntity(Entity): void:** A mezőre helyez egy entitást.
* **+ unsetEntity(): void:** A mezőn lévő entitás lekerül a mezőről, ami után a mező üres lesz.
* **+ isEmpty(): boolean:** Megadja, hogy van-e épp entitás a mezőn. Ha van, hamissal tér vissza, egyébként igazzal.
* **+ placeSlipFactor(double): boolean:** Egy entitás próbálja megváltoztatni egy mező tapadását. A művelet igazzal tér vissza, ha sikerült a változtatás, egyébként pedig hamissal.
* **+ setCurEntity(Entity): void:** Beállítja a mezőn lévő entitást.
* **+ getCurEntity(): Entity:** Lekérdezi a mezőn lévő entitást.

### Floor

#### Felelősség

A játéktér egy padlóját képviseli. Egy padlón tartózkodhat entitás, így rá is lehet lépni.

#### Ősosztályok

Field

#### Interfészek

-

#### Attribútumok

* **-**

#### Metódusok

* **+ visitByWorker(Worker, Worker): boolean:** Egy dolgozó a padlóra lép. Igazzal tér vissza, mivel egy munkás mindig ráléphet egy padlóra.
* **+ visitByCrate(Worker, Crate): boolean:** Egy láda a padlóra lép. Igazzal tér vissza, mivel egy láda mindig ráléphet egy padlóra.
* **+ placeSlipFactor(double): boolean:** Egy entitás próbálja megváltoztatni egy padló tapadását. Ez lehetséges, így a művelet igazzal tér vissza.

### Hole

#### Felelősség

A játéktér egy lyukas padlóját képviseli, ami lehet nyitott vagy zárt állapotú. Zárt állapotban ugyanúgy viselkedik, mint egy egyszerű padló. Nyitott állapotban mind ha dolgozó, mind ha láda lép rá, az beleesik.

#### Ősosztályok

Field ⇨ Floor

#### Interfészek

-

#### Attribútumok

* **- open: boolean:** Igaz, amennyiben a lyuk nyitott állapotban van, egyébként hamis.

#### Metódusok

* **+ visitByWorker(Worker, Worker): boolean:** Egy dolgozó a lyukra lép. Igazzal tér vissza, mivel egy munkás mindig ráléphet egy lyukra. Ha nyitva van a lyuk, a munkás leesik.
* **+ visitByCrate(Worker, Crate): boolean:** Egy láda a lyukra lép. Igazzal tér vissza, mivel egy láda mindig ráléphet egy lyukra. Ha nyitva van a lyuk, a láda leesik.
* **+ visitByLifeCrate(Worker, LifeCrate): boolean:** Egy szívecskés láda a lyukra lép. Igazzal tér vissza, mivel egy szívecskés láda mindig ráléphet egy lyukra. Ha nyitva van a lyuk, a szívecskés láda leesik és életet ad annak a dolgozónak, aki a lyukra tolta.
* **+ placeSlipFactor(double): boolean:** Egy entitás próbálja megváltoztatni egy lyuk tapadását. A művelet hamissal tér vissza, mert ez nem lehetséges.
* **+ setOpen(boolean): void:** Beállítja a lyuk állapotát. Ha paraméterként igaz logikai értéket kap, a lyukat kinyitja, egyébként becsukja.

### *IVisitable*

#### Felelősség

A Visitor minta egyik fele, ami a lépés sikerességét adja vissza. Ez a fél a sikeresség eldöntéséért felel.

#### Ősosztályok

-

#### Metódusok

* ***+ visitByWorker(Worker, Worker): boolean:*** Egy dolgozó lépésének sikerességét adja vissza.
* ***+ visitByCrate(Worker, Crate): boolean:*** Egy láda lépésének sikerességét adja vissza.
* ***+ visitByLifeCrate(Worker, LifeCrate): boolean:*** Egy szívecskés láda lépésének sikerességét adja vissza.

### *IVisitor*

#### Felelősség

A Visitor minta másik fele, ami a lépés sikerességét adja vissza. Ez a fél a dinamikus típussal való visszahívásért felel.

#### Ősosztályok

-

#### Metódusok

* ***+ visit(Worker, IVisitable): boolean:*** Dinamikus típussal való visszahívást tartalmazza, a visszatérési érték az átlépési lehetőséget fejezi ki.

### LifeCrate

#### Felelősség

Egy szívecske szimbólummal rendelkező ládát szimbolizál. Ez a láda egy hagyományos ládaként viselkedik, de ha leesik egy lyukba, akkor az őt leejtő dolgozó kap egy életet.

#### Ősosztályok

Entity ⇨ Crate

#### Interfészek

-

#### Attribútumok

* **-**

#### Metódusok

* **+ remove(Worker): void:** A szívecskés láda kikerül a raktárból, miközben a paraméterként átvett dolgozónak életet ad.

### PlaceableItem

#### Felelősség

Egy munkásnál lévő letehető dolgot szimbolizálja, ami lehet méz (HONEY) vagy olaj (OIL) de az is lehet, hogy nincs nála semmi (NOTHING).

### Spawn

#### Felelősség

A játéktér egy olyan padlóját képviseli, ami egy dolgozó kiindulási mezője. Minden dolgozó saját kiindulási mezővel rendelkezik, innen kezdik a játékot, és innen is folytatják életvesztés után, ha van még életük. Erre a mezőre kizárólag a hozzá tartozó dolgozó léphet, más entitás nem. Ha más dolgozó próbálna meg közvetetten rálépni (tehát úgy, hogy egy láda tolja őt), akkor a dolgozó életet veszít.

#### Ősosztályok

Field ⇨ Floor

#### Interfészek

-

#### Attribútumok

* **- owner: Worker:** A dolgozó, amelynek ez a kiindulási mezője.

#### Metódusok

* **+ visitByWorker(Worker, Worker): boolean:** Egy munkás kiindulási mezőre lép. Ez csak akkor lehetséges, ha a dolgozónak ez a kiindulási mezője vagy ha a dolgozót láncban tolják. Utóbbi esetben, ha a dolgozónak nem ez a kiindulási mezője, akkor ő nekiütközik a kiindulási mezőnek és életet veszít.
* **+ isEmpty(): boolean:** Minden esetben hamissal tér vissza, így biztosítható a kiindulási mezőre való lépés helyes működése.
* **+ placeSlipFactor(double): boolean:** Egy entitás próbálja megváltoztatni egy kiindulási hely tapadását. A művelet hamissal tér vissza, mert ez nem lehetséges.

### Switch

#### Felelősség

A játéktér egy olyan padlóját képviseli, melyen egy kapcsoló található. Amennyiben erre a kapcsolóra egy láda lép, egy vagy több lyuk nyitott állapotba kerül. Ha a láda lekerül a mezőről, a kapcsoló által kinyitott lyukak bezáródnak. Ha dolgozó lép a kapcsolóra, nem kapcsol.

#### Ősosztályok

Field ⇨ Floor

#### Interfészek

-

#### Attribútumok

* **- holes: List<Hole>:** Azon lyukak, melyek ehhez a kapcsolóhoz vannak rendelve.

#### Metódusok

* **+ visitByCrate(Worker, Crate): boolean:** Egy láda a kapcsolóra lép. Igazzal tér vissza, mivel egy láda mindig ráléphet egy kapcsolóra. Ilyenkor a kapcsolóhoz rendelt lyukak nyitott állapotba kerülnek.
* **+ unsetEntity(): void:** A mezőn lévő entitás lekerül a mezőről, ezek után a mező üres lesz. Ilyenkor a kapcsolóhoz rendelt lyukak zárt állapotba kerülnek.
* **+ placeSlipFactor(double): boolean:** Egy entitás próbálja megváltoztatni egy kapcsoló tapadását. A művelet hamissal tér vissza, mert ez nem lehetséges.

### Target

#### Felelősség

A játéktér egy olyan padlóját képviseli, ami egy előírt hely a ládák számára. Ha egy dolgozó erre a mezőre tol egy ládát, akkor pontot kap érte, azonban ha később valaki eltolja innen a ládát, az érte pontot kapó dolgozó elveszti a kapott pontot.

#### Ősosztályok

Field ⇨ Floor

#### Interfészek

-

#### Attribútumok

* **- whoPushed: Worker:** Az a dolgozó, amelyik pontot kapott egy láda ide tolásával.

#### Metódusok

* **+ visitByCrate(Worker, Crate): boolean:** Egy láda az előírt helyre lép. Igazzal tér vissza, mivel egy láda mindig ráléphet egy előírt helyre. Ilyenkor a láncot toló munkás pontot kap.
* **+ unsetEntity(): void:** A mezőn lévő entitás lekerül a mezőről, ezek után a mező üres lesz. Ilyenkor a dolgozó, aki eredetileg a helyére tolta a ládát, pontot veszít.
* **+ placeSlipFactor(double): boolean:** Egy entitás próbálja megváltoztatni egy előírt hely tapadását. A művelet hamissal tér vissza, mert ez nem lehetséges.
* **+ unsetWhoPushed(): void:** Beállítja, hogy jelenleg nincs dolgozó aki ládát tolt volna az előírt helyre.
* **+ setWhoPushed(Worker): void:** Beállítja, hogy melyik dolgozó tolta a ládát az előírt helyre.

### Wall

#### Felelősség

A játéktér egy olyan mezőjét képviseli, amire nem lehet lépni. Ha dolgozó próbálna meg közvetetten rálépni (tehát úgy, hogy egy láda tolja őt), akkor a dolgozó életet veszít.

#### Ősosztályok

Field

#### Interfészek

-

#### Attribútumok

* **-**

#### Metódusok

* **+ visitByWorker(Worker, Worker): boolean:** Egy dolgozó a falra lép. Hamissal tér vissza, mivel egy munkás sosem léphet rá egy falra.
* **+ visitByCrate(Worker, Crate): boolean:** Egy láda a falra lép. Hamissal tér vissza, mivel egy láda sosem léphet rá egy falra.

### Warehouse

#### Felelősség

A játékteret szimbolizálja, tartalmazza a benne lévő mezőket és entitásokat.

#### Ősosztályok

Field

#### Interfészek

-

#### Attribútumok

* **- fields: List<Field>:** A raktárban található mezők.
* **- entities: List<Entity>:** A raktárban található entitások.

#### Metódusok

* **+ initialize(): void:** Inicializálja a játékteret, azzal felépíti a mezőket, és létrehozza az entitásokat.
* **+ setField(Field, int, int): void:** Beállítja a pálya adott helyére a paraméterként kapott mezőt.
* **+ addEntity(Entity, int, int): void:** Hozzáad egy entitást az adott pozíción levő mezőre.
* **- workerDie(Worker): void:** Amikor egy dolgozó véglegesen meghalt, a raktár kiregisztrálja őt.
* **- crateStuck(Crate): void:** Amikor egy láda beragad, azaz tovább már semmilyen irányban nem tolható el, a raktár regisztrálja neki ezt az állapotot.
* **- update(): void:** A raktár frissíti minden benne található mező és entitás állapotát, valamint figyeli a játék végét is.

### Worker

#### Felelősség

Egy dolgozót szimbolizál. Nyilvántartja a dolgozó pozícióját, életét, valamint pontszámát.

#### Ősosztályok

Entity

#### Interfészek

-

#### Attribútumok

* **- health: int:** A dolgozó élete. Ha eléri a nullát, a dolgozó meghal és kikerül a játékból.
* **- points: int:** A dolgozó pontszáma. Minden egyes előírt helyre tolt ládával egyel nő, ám amennyiben azok a ládák később elmozdulnak az előírt helyekről, a pontok elvesznek.
* **- power: double:** A munkás jelenlegi ereje, mely egy tolási kísérlet során entitásonként csökken.
* **- orgPower: double:** A munkás eredeti ereje.
* **- item: PlaceableItem:** A dolgozónál lévő dolog.
* **- spawnpoint: Spawn:** A dolgozóhoz rendelt kiindulási mező.

#### Metódusok

* **+ push(Worker, Entity, Direction): boolean:** A dolgozó eltol egy entitást a megadott irányban lévő szomszédos mezőre. Igazzal tér vissza, ha sikerült eltolni, egyébként hamissal.
* **+ pushByWorker(Worker, Worker, Direction): boolean:** Egy dolgozó tolja a dolgozót a megadott irányban lévő szomszédos mezőre, így a dolgozó oda lép. Hamissal tér vissza, mivel dolgozó nem tolhat közvetlenül másik dolgozót.
* **+ pushByCrate(Worker, Crate, Direction): boolean:** Egy láda tolja a dolgozót a megadott irányban lévő szomszédos mezőre, így a dolgozó oda lép. Igazzal tér vissza, ha sikerült lépni, egyébként hamissal.
* **+ move(Direction): void:** A játékos lépteti a dolgozót a megadott irányban.
* **+ loseHealth(): void:** A dolgozó egy életet veszít.
* **+ gainHealth(): void:** A dolgozó egy életet kap.
* **+ losePoint(): void:** A dolgozó egy pontot veszít.
* **+ gainPoint(): void:** A dolgozó egy pontot kap.
* **+ reSpawn(): void:** A dolgozó átkerül jelenlegi mezőjéről a számára kijelölt kiinduló mezőre.
* **+ die(): void:** A dolgozó meghal és kikerül a játékból.
* **+ consumePower(double): double:** A dolgozó próbál eltolni egy entitást. A művelet a munkás entitáshoz viszonyított erejét adja vissza, melynek negatív volta jelzi, ha a munkásnak nincs elég ereje a toláshoz.
* **+ placeItem(): void:** A dolgozó megpróbálja letenni a nála lévő dolgot arra a mezőre, amin jelenleg tartózkodik.
* **+ visit(Worker, IVisitable): boolean:** A dolgozó egy vizitálható objektumra (mezőre) lép. Igazzal tér vissza amennyiben sikerült a lépés, egyébként hamissal.

## A tesztek részletes tervei, leírásuk a teszt nyelvén

### A tesztek rövid leírása

Minden tesztesetünk egy-egy alap lépést próbál tesztelni, abból az elvből kiindulva, hogy ha minden elem jó akkor az egész is jó lesz.

Felépítést tekintve minden teszt eset a következő struktúrát követi:

Betöltünk egy előre elkészített pályát, amin végrehajtunk egy esemény szekvenciát, ahol az esemény vagy játékos inputot vagy belső eseményt (pl.: eszköz adást) vagy adatlekérést jelöl.

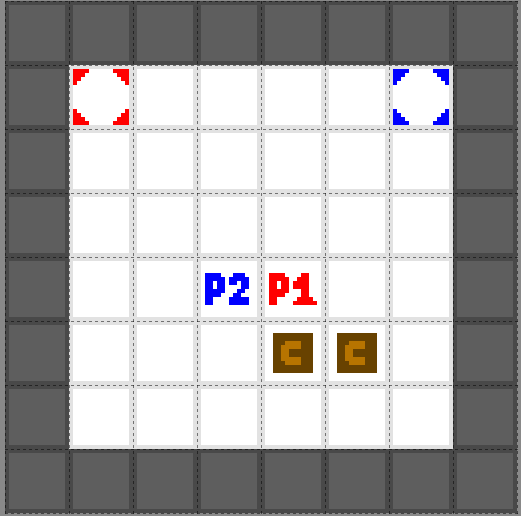
A modell alapján, megadjuk az elvárt viselkedést. Ehhez hasonlítjuk a tényleges eredményt, ha a kettő egyezik akkor sikeres a teszt.

Megjegyzés: LEVEL\_SUCCESS jelentés a sikeres pályabetöltés.

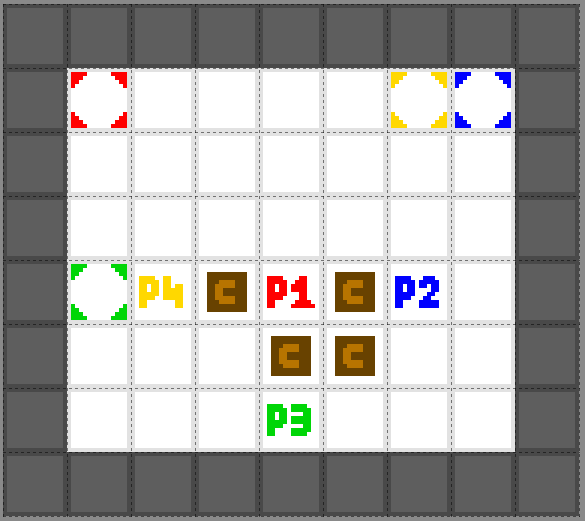
#### A tesztpályák felépítése

Az egyszerűség kedvéért felsoroljuk a tesztesetekben használt pályák grafikus képeit, ezáltal könnyedén el lehet igazodni az egyes tesztesetek lépéseiben.

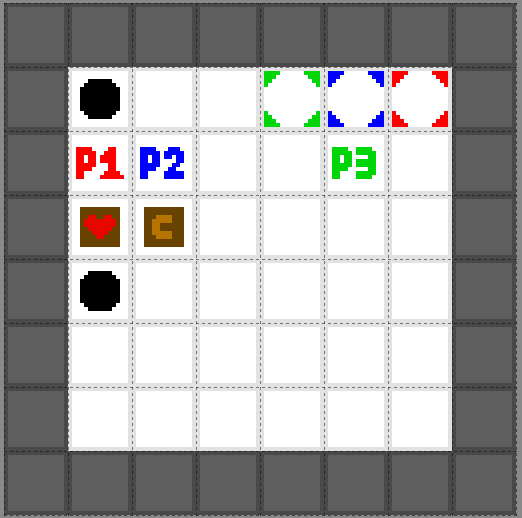
#### testlevel1



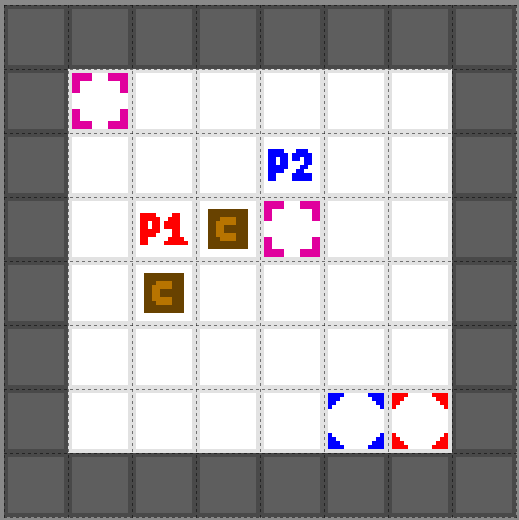
#### testlevel2



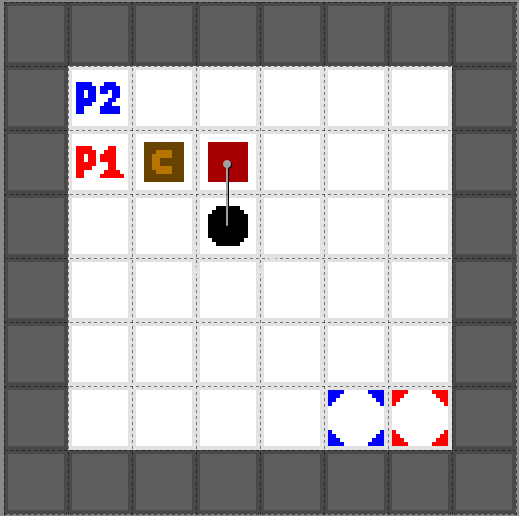
#### testlevel3



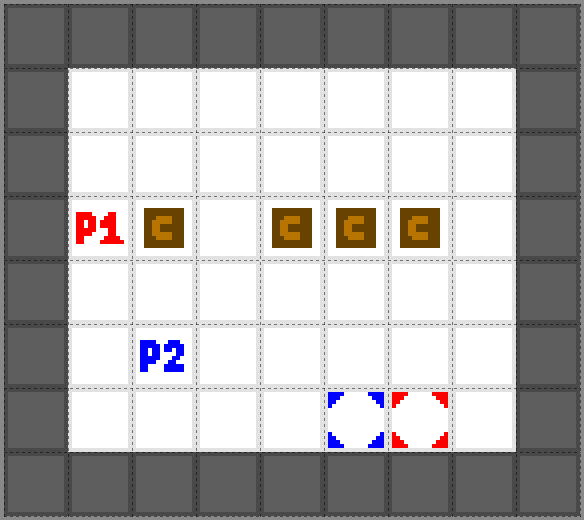
#### testlevel4



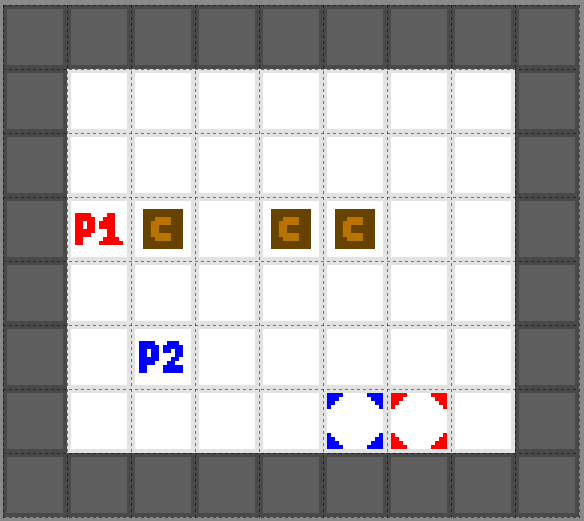
#### testlevel5



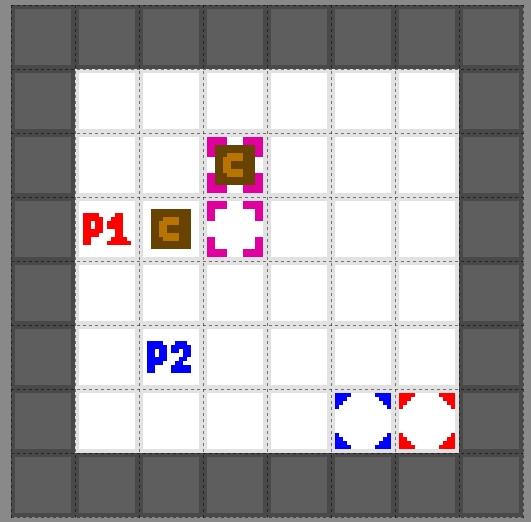
#### testlevel6



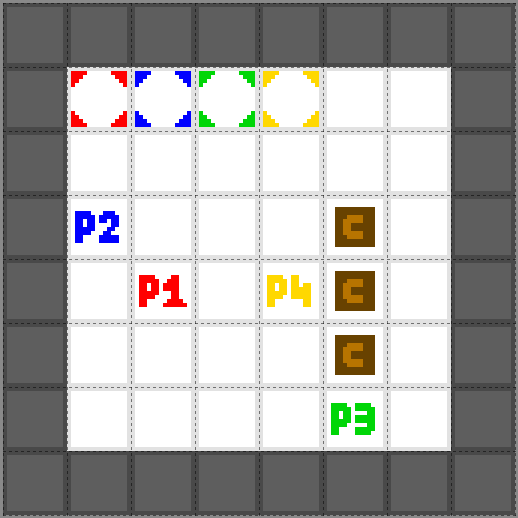
#### testlevel7



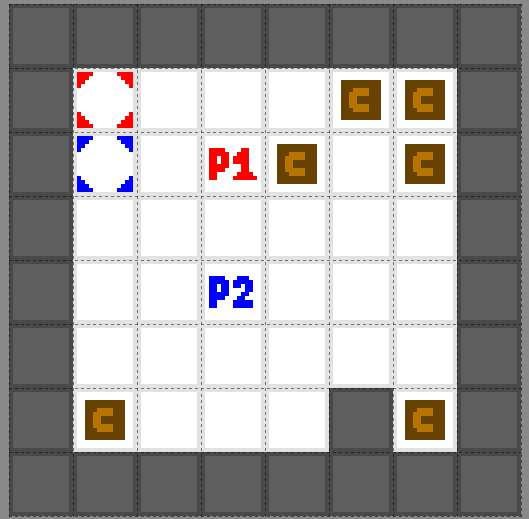
#### testlevel8



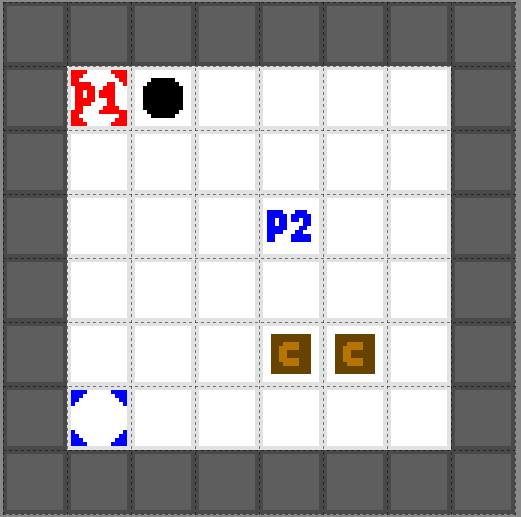
#### testlevel9

****

#### testlevel10

****

#### testlevel11

****

### Move worker on empty field

* **Leírás**

Egy munkás léptetésének tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Az első lépés a teszthez szükséges pálya betöltése, melyen az objektumok előre beállított - a tesztnek kedvező - helyen tartózkodnak. Az 1-es azonosítóval rendelkező dolgozó mellett jobb oldalt egy üres padló van, melyre minden esetben rá kell tudnia lépni. A teszt rálépteti a dolgozót erre a padlóra, majd ellenőrzi a lépés sikerességét.

* **Bemenet**

LEVEL "testlevel1.json"  
STEP 1 R

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel1.json"  
STEP\_OK 1

### Worker pushes crate to empty field

* **Leírás**

Egy láda eltolásának tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A pálya betöltését követően az 1-es számú dolgozóval megpróbálunk jobbra lépni. Ebben az irányban egy láda van, így a dolgozó eltolja azt. A kimenet mutatja a tolás sikerességét.

* **Bemenet**

LEVEL "testlevel1.json"

STEP 1 D

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel1.json"

STEP\_OK 1

### Worker pushes other worker directly

* **Leírás**

Egy dolgozó dolgozót közvetlen tolásának tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A pálya betöltését követően az 1-es számú dolgozóval megpróbálunk balra lépni. Mivel ezen a mezőn egy másik dolgozó tartózkodik, akit közvetlenül nem tud eltolni, ezért a lépés sikertelen.

* **Bemenet**

LEVEL "testlevel1.json"

STEP 1 L

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel1.json"

STEP\_FAIL 1

### Worker pushes other worker in chain

* **Leírás**

Egy dolgozó dolgozót láncban tolásának tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A pálya betöltését követően az 1-es számú dolgozóval megpróbálunk jobbra lépni. Ebben az irányban egy lánc van, a dolgozó melletti végén egy ládával, így egy dolgozó, amely a láncban van, közvetetten eltolhatóvá válik. A kimenet mutatja a tolás sikerességét.

* **Bemenet**

LEVEL "testlevel2.json"

STEP 1 R

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel2.json"

STEP\_OK 1

### Worker smashed by wall

* **Leírás**

Egy dolgozó fal általi összenyomódásának tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A pályát betöltjük, lekérjük a 3-as sorszámú játékos életeinek a számát, majd a dolgozóját az 1-es sorszámú dolgozóval egy falba toljuk. Az életei számának ismételt lekérdezésével megbizonyosodhatunk arról, hogy a játékos elveszített egy életet.

* **Bemenet**

LEVEL "testlevel2.json"

PLAYERINFO 3 HP

!STEP 1 D

PLAYERINFO 3 HP

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel2.json"

PLAYER\_HP 3 3

STEP\_OK 1

PLAYER\_HP 3 2

### Worker smashed by spawn

* **Leírás**

Egy dolgozó kiindulási hely általi összenyomódásának tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A pályát betöltjük, lekérjük a 4-es sorszámú játékos életeinek a számát, majd a dolgozóját az 1-es sorszámú dolgozóval egy idegen kiindulási mezőre toljuk. Az életei számának ismételt lekérdezésével megbizonyosodhatunk arról, hogy a játékos elveszített egy életet.

* **Bemenet**

LEVEL "testlevel2.json"

PLAYERINFO 4 HP

!STEP 1 L

PLAYERINFO 4 HP

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel2.json"

PLAYER\_HP 4 3

STEP\_OK 1

PLAYER\_HP 4 2

### Worker falls into hole

* **Leírás**

Egy dolgozó lyukba esésének tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Betöltjük az előkészített tesztpályát, lekérdezzük az 1-es sorszámú játékos életeinek a számát illetve a mellette található lyuk állapotát, majd a lyukra léptetjük a játékoshoz tartozó dolgozót. A léptetést követően ismét lekérdezzük az életek számát, amely eggyel csökkent, mivel a dolgozó beleesett a nyitott állapotú lyukba.

* **Bemenet**

LEVEL "testlevel3.json"

PLAYERINFO 1 HP

HOLEINFO 1,1

!STEP 1 U

PLAYERINFO 1 HP

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel3.json"

PLAYER\_HP 1 3

OPEN 1,1

STEP\_OK 1

PLAYER\_HP 1 2

### Worker pushes lifecrate into hole

* **Leírás**

Egy szívecskés ládát lyukba tolásának tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Első lépésben betöltjük a pályát. Ezt követően lekérdezzük az 1-es sorszámú játékos életeinek a számát, továbbá a dolgozójától két mezőre található lyuk állapotát. A lyuk és a játékos között egy szívecskés láda van, amit a dolgozó léptetésével a lyukba tolunk. A kimenet mutatja, hogy a lyuk nyitott állapotban volt, a tolást követően pedig a játékos egy életet kapott, tehát a láda lyukba tolása és annak leesése sikeres volt.

* **Bemenet**

LEVEL "testlevel3.json"

PLAYERINFO 1 HP

HOLEINFO 1,4

!STEP 1 D

PLAYERINFO 1 HP

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel3.json"

PLAYER\_HP 1 3

OPEN 1,4

STEP\_OK 1

PLAYER\_HP 1 4

### Target activation test

* **Leírás**

Egy láda előírt helyre tolásának ellenőrzése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Szokásos módon betöltjük a tesztpályát, majd lekérjük az előírt helyen lévő ládák számát és az 1-es sorszámú játékos pontjait. A játékos dolgozójával jobbra lépünk, amellyel a mellette lévő ládát egy előírt helyre toljuk. Az ismételt lekérdezéssel megbizonyosodhatunk a teszt sikerességéről, végezetül pedig fájlba mentjük a teszt állapotát, amely egy későbbi tesztesetben kerül felhasználásra.

* **Bemenet**

LEVEL "testlevel4.json"

PLAYERINFO 1 PTS

CNT CR

!STEP 1 R

PLAYERINFO 1 PTS

CNT CR

SAVE "TEST9\_P1\_SCORED.ser"

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel4.json"

PLAYER\_PTS 1 0

CNT\_CR 0

STEP\_OK 1

PLAYER\_PTS 1 1

CNT\_CR 1

SAVE\_SUCCESS "TEST9\_P1\_SCORED.ser"

### Target deactivation test

* **Leírás**

Egy láda előírt helyről eltolásának ellenőrzése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Betöltjük a 8.2.9-es tesztesetben elmentett állapotot és lekérdezzük az előírt helyen lévő ládák számát illetve az 1-es sorszámú játékos pontjait. Ezt követően a 2-es sorszámú dolgozót lefele léptetjük, aminek keretében az 1-es sorszámú játékos dolgozója által odatolt láda lekerül az előírt helyről, így a játékos elveszíti a tolásért kapott pontot. Az újbóli lekérdezés mutatja a teszt sikerességét a kimeneten.

* **Bemenet**

LOAD "TEST9\_P1\_SCORED.ser"

PLAYERINFO 1 PTS

CNT CR

!STEP 2 D

PLAYERINFO 1 PTS

CNT CR

* **Elvárt kimenet**

LOAD\_SUCCESS "TEST9\_P1\_SCORED.ser"

PLAYER\_PTS 1 1

CNT\_CR 1

STEP\_OK 2

PLAYER\_PTS 1 0

CNT\_CR 0

### Switch activation test

* **Leírás**

Egy kapcsoló bekapcsolásának tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Az 5-ös számú tesztpálya betöltését követően lekérdezzük a 3,3-as pozíciójú lyuk állapotát, majd az 1-es sorszámú dolgozót jobbra léptetjük, ahol egy láda található. A dolgozó a ládát a láda mellett található kapcsolóra tolja, ezzel aktiválva azt. A kapcsoló kinyitja a hozzárendelt lyukakat, beleértve a 3,3-ast, aminek kinyílásáról a státuszának ismételt lekérdezésével megbizonyosodhatunk. Végezetül pedig fájlba mentjük a teszt állapotát, amely egy későbbi tesztesetben kerül felhasználásra.

* **Bemenet**

LEVEL "testlevel5.json"

HOLEINFO 3,3

!STEP 1 R

HOLEINFO 3,3

SAVE "TEST11\_HOLE\_ACT.ser"

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel5.json"

CLOSED 3,3

STEP\_OK 1

OPEN 3,3

SAVE\_SUCCESS "TEST11\_HOLE\_ACT.ser"

### Switch deactivation test

* **Leírás**

Egy kapcsoló kikapcsolásának tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

Betöltjük a 8.2.11-es tesztesetben elmentett állapotot, majd lekérdezzük a 3,3-as pozíciójú lyuk állapotát. Az 1-es sorszámú dolgozót jobbra léptetjük, ahol egy kapcsoló, rajta pedig egy láda található, amely eltolható. Azáltal, hogy a ládát letoljuk a kapcsolóról és helyére a dolgozó kerül, a kapcsoló kikapcsol és bezárja a hozzárendelt lyukakat, beleértve a 3,3-ast, aminek bezáródásáról a státuszának ismételt lekérdezésével megbizonyosodhatunk.

* **Bemenet**

LOAD "TEST11\_HOLE\_ACT.ser"

HOLEINFO 3,3

!STEP 1 R

HOLEINFO 3,3

* **Elvárt kimenet**

LOAD\_SUCCESS "TEST11\_HOLE\_ACT.ser"

OPEN 3,3

STEP\_OK 1

CLOSED 3,3

### Field modificator test - oil

* **Leírás**

Egy dolgozó olajat lehelyezésének tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A hatos számú tesztpálya, úgy van előkészítve, hogy az 1-es számú játékostól jobbra egy láda, üres padló, majd egy lánc található. A lánc és az említett láda súlya túl nagy alap esetben, hogy egy munkás eltolja, de ha a kiegészítő láda eltolását olaj segítségével megkönnyítjük, akkor már eltolható a teljes lánc.

Ennek megfelelően a pályát betöltjük majd jobbra léptetjük a dolgozó, ami nem sikerül, majd újra betöltjük a pályát elhelyezzük az olajat a láda és a lánc közti üres padlóra, ami után a kezdeti helyre visszalépve végrehajtjuk a tolást, ami most sikeres.

* **Bemenet**

LEVEL "testlevel6.json"

!STEP 1 R

!STEP 1 R

LEVEL "testlevel6.json"

!STEP 1 U

!STEP 1 R

!STEP 1 R

!STEP 1 D

GIVE 1 O

!PLACE 1

!STEP 1 U

!STEP 1 L

!STEP 1 L

!STEP 1 D

!STEP 1 R

!STEP 1 R

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel6.json"

STEP\_OK 1

STEP\_FAIL 1

LEVEL\_SUCCESS "testlevel6.json"

STEP\_OK 1

STEP\_OK 1

STEP\_OK 1

STEP\_OK 1

PLACE\_OK 1

STEP\_OK 1

STEP\_OK 1

STEP\_OK 1

STEP\_OK 1

STEP\_OK 1

STEP\_OK 1

### Field modificator test - honey

* **Leírás**

Egy dolgozó méz lehelyezésének tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A hetes számú tesztpálya, úgy van előkészítve, hogy az 1-es számú játékostól jobbra egy láda, üres padló, majd egy lánc található. A lánc és az említett láda súlya pont akkora alap esetben, hogy egy munkás eltolja, de ha a kiegészítő láda eltolását méz segítségével megnehezítjük, akkor már nem eltolható a teljes lánc.

Ennek megfelelően a pályát betöltjük majd jobbra léptetjük a dolgozó, ami pont sikerül, majd újra betöltjük a pályát elhelyezzük a mézet a láda és a lánc közti üres padlóra, ami után a kezdeti helyre visszalépve végrehajtjuk a tolást, ami most sikertelen.

* **Bemenet**

LEVEL "testlevel7.json"

!STEP 1 R

!STEP 1 R

LEVEL "testlevel7.json"

!STEP 1 U

!STEP 1 R

!STEP 1 R

!STEP 1 D

GIVE 1 H

!PLACE 1

!STEP 1 U

!STEP 1 L

!STEP 1 L

!STEP 1 D

!STEP 1 R

!STEP 1 R

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel7.json"

STEP\_OK 1

STEP\_OK 1

LEVEL\_SUCCESS "testlevel7.json"

STEP\_OK 1

STEP\_OK 1

STEP\_OK 1

STEP\_OK 1

PLACE\_OK 1

STEP\_OK 1

STEP\_OK 1

STEP\_OK 1

STEP\_OK 1

STEP\_OK 1

STEP\_FAIL 1

### Locking mechanism of simultaneously pushed conflicting chains

* **Leírás**

Egyszerre történő, egymást kizáró láncok viselkedésének tesztje.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A tesztpálya két független ütközési helyet tartalmaz. Az 1-es és 2-es dolgozók azonos helyre próbálnak lépni, illetve ettől függetlenül a 3-as és 4-es játékos pedig olyan láncot szeretnének tolni, melynek van közös tagja. A test során azonos időperiódusban? mind a 4 játékos a hozzá tartozó ütközési hely irányába lép, ekkor a 2-2 lépés közül 1-1 sikerül (az első) és 1-1 nem (a második).

* **Bemenet**

LEVEL "testlevel9.json"

STEP 1 U

STEP 2 R

STEP 3 U

!STEP 4 R

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel9.json"

STEP\_OK 1

STEP\_FAIL 2

STEP\_OK 3

STEP\_FAIL 4

### Game ends because there is a crate on all target

* **Leírás**

A célterületek feltöltése miatti játék vég észlelésének tesztje.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A tesztpálya olyan, hogy 1 kivételével minden célterületén láda található, más játék végét jelentő feltétel sem teljesül. Betöljük a tesztpályát, és az 1-es munkással eltolunk egy ládát úgy, hogy az utolsó célterületre kerüljön, várjuk hogy jó okkal legyen vége.

* **Bemenet**

LEVEL "testlevel8.json"

!STEP 1 R

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel8.json"

STEP\_OK 1

LEVEL\_END TR

### Game ends because there are no moveable crates

* **Leírás**

Beragadás miatti játék vég észlelésének tesztje.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A tesztpálya olyan, hogy egy kivételével az rajta található ládák nem mozgatható, más játék végét jelentő feltétel sem teljesül.

Betöljük a tesztpályát, és az 1-es munkással eltoljuk a ládát úgy, hogy beragadjon, várjuk hogy jó okkal legyen vége.

A ládák különféle edge-case konfigurációkban vannak, amelyek lefogják az összes esetet ami miatt egy láda többé nem mozgathatóvá válik.

* **Bemenet**

LEVEL "testlevel10.json"

!STEP 1 R

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel10.json"

STEP\_OK 1

LEVEL\_END CR

### Game ends because all the workers died except one

* **Leírás**

Játékos elimináció miatti játék vég észlelésének tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A tesztpálya olyan, hogy 2 játékos található rajta, melyek közül az 1-esnek már csak két élete van, játék végét jelentő feltétel nem teljesül.

Betöljük a tesztpályát, és az 1-es munkással kétszer lyukba lépünk, várjuk hogy jó okkal legyen vége.

* **Bemenet**

LEVEL "testlevel11.json"

HOLEINFO 2,1

PLAYERINFO 1 HP

!STEP 1 R

PLAYERINFO 1 HP

!STEP 1 R

PLAYERINFO 1 HP

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel11.json"

PLAYER\_HP 1 2

OPEN X,X

STEP\_OK 1

PLAYER\_HP 1 1

STEP\_OK 1

PLAYER\_HP 1 0

LEVEL\_END P

### Player gains an item

* **Leírás**

Tárgy (pl.: olaj, méz) szerzésének tesztelése.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszteset során egy dolgozónak oda adjuk sorban az összes lehetséges itemet, majd ellenőrizzük, hogy megkapta-e.

* **Bemenet**

LEVEL "testlevel11.json"

GIVE 1 N

PLAYERINFO 1 ITEM

GIVE 1 O

PLAYERINFO 1 ITEM

GIVE 1 H

PLAYERINFO 1 ITEM

GIVE 1 N

PLAYERINFO 1 ITEM

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel11.json"

PLAYER\_ITEM 1 N

PLAYER\_ITEM 1 O

PLAYER\_ITEM 1 H

PLAYER\_ITEM 1 N

### Worker puts down item

* **Leírás**

Tárgy elhelyezésének tesztje.

* **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszteset során egy dolgozó minden lehetséges item mellett kísérletet tesz a tárgy lehelyezésére.

* **Bemenet**

LEVEL "testlevel11.json"

GIVE 1 O

!PLACE 1

GIVE 1 H

!PLACE 1

GIVE 1 N

!PLACE 1

* **Elvárt kimenet**

LEVEL\_SUCCESS "testlevel11.json"

PLACE\_OK 1

PLACE\_OK 1

PLACE\_FAIL 1

## A tesztelést támogató programok tervei

### Tesztesetek elkészítése

Az előírt szintaxist használva szövegesen, külön *.txt* fájlokba kell elkészíteni az egyes tesztesetekhez tartozó bemenetet és elvárt kimenetet. A teszteset nevét a fájok neve határozza meg, a program is erre hivatkozik a későbbiek során.

Egy tesztesethez tehát két fájl tartozik, az egyik a bemeneti parancsokat tartalmazza soronként és *tst\_in<name>.txt* a neve, a másik pedig az elvárt kimenetet, és *tst\_out<name>.txt* a neve. A *<name>* tag a teszteset nevét tartalmazza, és kizárólag a standard 128 karakteres ASCII betűiből állhat. A program később ezen a néven fog hivatkozni a tesztesetre. Amennyiben valamely követelmény nem teljesül (pl. bemeneti fájlhoz nincsen kimeneti) az adott teszteset automatikusan ignorálásra kerül.

Egy fájlon belül a bemeneti parancsok vagy kimeneti eredmények soronként szerepelnek. Az üres sor ignorálásra kerül, a pontosvesszővel pedig kommentezés lehetséges (pontosvessző után az adott tartalom a sor végéig ignorálásra kerül).

A tesztekhez tartozó pályá(ka)t a Tiled nevű programmal kell elkészíteni a mellékelt segédlet alapján.

### A tesztek kiértékelése

A program karakteres menüben jeleníti meg az elkészített teszteseteket, a nevükkel hivatkozva rájuk. A program a menüben a tesztekhez immár egy azonosítót (egész szám) is társít, melynek megadásával lehet egy tesztesetet lefuttatni. A program a bemeneti parancsok lefutása után összehasonlítja az eredményt az elvárt kimenettel, majd szövegesen jelzi a teszteset sikerességét vagy bukását, utóbbi esetben megadja a bukást okozó hibás kimenetet, illetve a hozzá tartozó elvárt eredményt is. Lehetőség van egy speciális menüpontban az összes teszteset egymás után történő futtatására. Ezek után a konzolon szintén szöveges formában jelennek meg a tesztek eredményei, többek között az összes teszt és a sikeres tesztek aránya.

Előfordulhatnak olyan tesztesetek, melyek egymásra hivatkoznak, például egy teszteset egy másik által kimentet állapotot használ. Ilyen esetben a tesztesetek ezt figyelmen kívül hagyó sorrendben történő futtatása automatikusan hibához vezet. Továbbá szintén automatikus hibát okoz egy nem létező pálya vagy állapot betöltésének kísérlete, általánosabban a *LEVEL/LOAD* parancsokkal hívott műveletek hibája.

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2018.04.07. 10:00 | 1,5 óra | SZAKÁLLAS | Osztályok terveinek (8.1) elkészítése |
| 2018.04.07. 12:00 | 1 óra | LAKATOS | Osztályok terveinek elkészítése, kiegészítése (8.1) |
| 2018.04.07. 19:00 | 1,5 óra | SZAKÁLLAS | Tesztesetek (8.2) leírásainak elkészítése |
| 2018.04.07. 20:00 | 2 óra | LENKEFI | Tesztesetek (8.2) leírásainak elkészítése, tesztesetekhez tartozó pályák elkészítése |
| 2018.04.08. 16:00 | 3 óra | JANI | Tesztesetek (8.2) leírásainak elkészítése, kiegészítése, felülvizsgálata |
| 2018.04.08. 16:00 | 2 óra | CSANÁDY | Tesztesetek (8.2) leírásainak elkészítése, kiegészítése |
| 2018.04.08. 20:00 | 0,5 óra | LAKATOS | A tesztelést támogató programok terveinek elkészítése (8.3) |
| 2018.04.08. 22:00 | 1 óra | JANI | Tesztesetek kiegészítése (8.2) |
| 2018.04.08. 22:00 | 1 óra | LENKEFI | Tesztesetek kiegészítése (8.2) |
| 2018.04.08. 23:00 | 1 óra | CSANÁDY | Dokumentum átnézése, konzisztencia vizsgálata |
| 2018.04.09. 00:30 | 0,5 óra | LAKATOS | Dokumentum véglegesítése |