## Feladat

*Szimuláljuk különféle élőlények túlélési versenyét. A lények egy pályán haladnak végig, ahol váltakozó terep viszonyok vannak. Egy lénynek a terep fajtájától függően változik az életereje, miközben azon keresztül megy, és közben a terep fajtáját is átalakítja. Először az első lény próbál végigjutni a pályán. Egészen addig halad, amíg végig nem ér vagy el nem fogy az életereje és elpusztul. Egy terep akkor is átalakul, ha azon egyébként a lény elpusztul. Ezen az átalakított pályán indul a második lény, majd sorban a többi. Adjuk meg a pályán végig jutó, életben maradt lények neveit!*

*A pályán három fajta terep fordulhat elő: fű, homok, mocsár. A lények különböző fajokhoz tartoznak. Három féle fajt különböztetünk meg.*

***Zöldike**: kezdeti életereje 10; füvön az életerő eggyel nő, homokon kettővel csökken, mocsárban eggyel csökken; a mocsaras terepet fűvé alakítja, a másik két féle terepet nem változtatja meg.*

***Buckabogár**: kezdeti életereje 15; füvön az ereje kettővel csökken, homokon hárommal nő, mocsárban néggyel csökken; a füvet homokká, a mocsarat fűvé alakítja, de a homokot nem változtatja meg.*

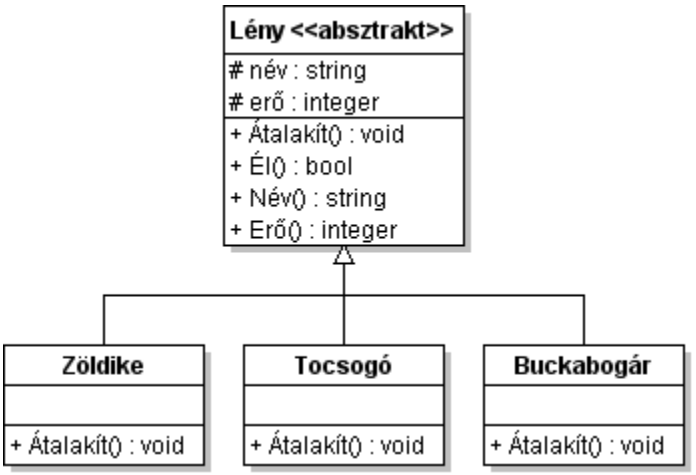
***Tocsogó**: kezdeti életereje 20; füvön az életerő kettővel csökken, homokon öttel csökken, mocsárban hattal nő; a füvet mocsárrá alakítja, a másik két féle terepet nem változtatja meg.*

*Minden lénynek van egy neve (sztring), ismert az aktuális életereje (egész szám) és a fajtája. Amíg az életerő pozitív, addig a lény életben van.*

*A verseny adatait egy szövegfájlból olvassuk be! A fájl első sora tartalmazza a lények számát, amelyet a lények adatai követnek. Egy karakter azonosítja a lény fajtáját, amit szóköz után a lény neve követ. Az azonosítók: Z – zöldike, B – buckabogár, T – tocsogó. A lények után következik a pálya leírása. Egy egész szám adja meg a pálya hosszát, amit pontosan ennyi szóközökkel elválasztott egész szám követ, amelyek a terepek fajtái adják meg. A fajták azonosítói: 0 – homok, 1 – fű, 2 – mocsár. Feltehetjük, hogy a fájl formátuma helyes.*

## Specifikáció

A lények leírásához bevezetünk négy osztályt. Attól függetlenül, hogy egy lény konkrétan kicsoda vagy mi a fajtája, számos közös tulajdonsággal rendelkeznek. Mindegyiknek van neve és életereje, meg lehet róla kérdezni, hogy hívják (*Név()*), él-e (*Él()*) még, azaz az életereje nagyobb-e nullánál, és szimulálni lehet a viselkedését a versenypálya egy bizonyos terepén. Ez utóbbi művelet (*Átalakít()*) egyrészt módosítja a lény életerejét, másrészt átalakítja a neki átadott terepet. Ennek a műveletnek a hatása attól függ, hogy egy lény milyen fajú, ezért ez a művelet a lények általános jellemzésének szintjén még nem implementálható. Az általános lény típusát leíró osztály absztrakt lesz, hiszen egyrészt az *Átalakít()* metódusa is absztrakt, másrészt úgysem akarunk ilyen objektumot létrehozni. Ebből származtatjuk a konkrét fajtájú lények, zöldikék, buckabogarak és tocsogók osztályait. A speciális osztályok konstruktorai meghívják az ősosztály konstruktorát, majd inicializálják az életerőt.




Az *Él()* és *Név()* metódusok az ősosztály szintjén implementálhatók, de az *Átalakít()* csak a konkrét osztályok szintjén. Az *Él()* metódus akkor ad *igaz* értéket, ha az életerő pozitív. A *Név()* metódus a *név* adattag értékét adja vissza. A származtatott osztályokban felüldefiniáljuk az *Átalakít()*metódust az alább definiált táblázatok alapján módosítja az életerőt és megváltoztatja az adott pályamezőt.

Zöldikék esetében a kezdő életerő: 10, amit a konstruktor állít be. Az *Átalakít()* művelet hatását az alábbi táblázat foglalja össze. Ez megkapja bemenetként az aktuális terep fajtáját, és a táblázat megfelelő sora alapján megváltoztatja az aktuális lény életerejét és visszaadja a terep új fajtáját.

| terep | életerő változás | terepváltozás |
|---|---|---|
| homok | -2 | - |
| fű | +1 | - |
| mocsár | -1 | fű |

Buckabogarak esetében a kezdő életerő: 15, és az *Átalakít()* művelet hatása:

| terep | életerő változás | terepváltozás |
|---|---|---|
| homok | +3 | - |
| fű | -2 | homok |
| mocsár | -4 | fű |

Tocsogók esetében a kezdő életerő: 20, és az *Átalakít()* művelet hatása:

| terep | életerő változás | terepváltozás |
|---|---|---|
| homok | -5 | - |
| fű | -2 | mocsár |
| mocsár | +6 | - |

Most pedig specifikálhatjuk a teljes feladatot. A specifikációban meg kell különböztetni egy-egy lény áthaladta utáni pálya állapotokat. A nulladik változat a kezdőpálya, az *i*-edik az *i*-edik lény mozgása utáni pálya.

Állapottér: $kezdőpálya: \mathbb{N}^m$, $lények: Lény^n$, $túlélők: String^*$

Előfeltétel: $lények = lények' \wedge kezdőpálya = kezdőpálya'$

Utófeltétel: $lények = lények' \wedge kezdőpálya = kezdőpálya' \wedge$

$pálya:(\mathbb{N}^m)^n \wedge pálya[0] = kezdőpálya \wedge$

$\forall i \in [1..n]: pálya[i] = Eredm(lények[i], pálya[i-1])_2 \wedge$

$$túlélők = \bigoplus_{\substack{i=1 \\ Eredm(lények[i], pálya[i-1])_1.Él()}}^{n} < lények[i].név >$$

ahol az $Eredm: Lény \times \mathbb{N}^m \rightarrow Lény \times \mathbb{N}^m$ függvény kiszámolja, hogy egy lény az adott pályán áthaladva hogyan változik (életben marad-e) és hogyan változik a pálya. Ezt a számítást egy rekurzív definíciójú függvénnyel lehet leírni úgy, hogy egy kezdeti életerővel rendelkező *lény* és egy *m* hosszúságú *pálya* esetén $Eredm(lény, pálya) = r(m)$, ahol

$r:[0 .. m] \rightarrow Lény \times \mathbb{N}^m$

$$r(0) = (lény, pálya)$$

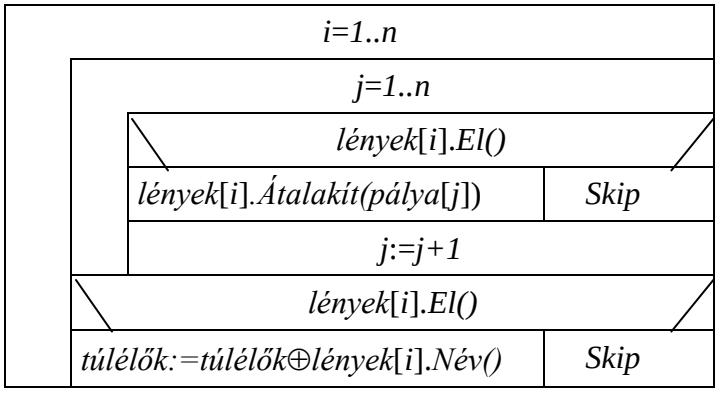
$$\forall j \in [1..m]: \quad r(j) = \begin{cases} Lépés(r(j-1)) & ha \quad r(j-1)_1.Él() \\ r(j-1) & ha \quad \neg r(j-1)_1.Él() \end{cases}$$

A $Lépés(r(j-1))$ állítja elő a *j*-edik lépés utáni $r(j)_1$ lényt (csak az életereje változik az előző lépéshez képest) valamint az $r(j)_2$ pályát (ennek a *j*-edik mezője változhat meg ekkor). Ezeket a változásokat a korábban definiált *Átalakít()* függvény alapján számíthatjuk ki. Ez ugyanis az aktuális $r(j-1)_2[j]$ pályamező alapján megadja, hogy mennyivel változik az $r(j-1)_1$ lény életereje és mivé alakul a az $r(j-1)_2[j]$ pályamező.

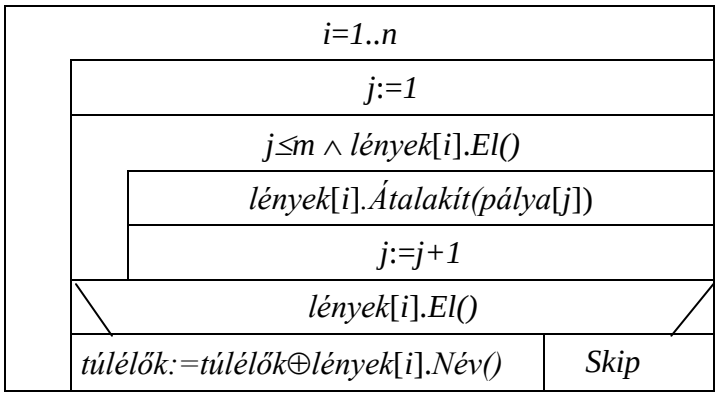
## Absztrakt program

A megoldó programnak két szintje lesz. A felső szinten a fenti specifikációnak megfelelő algoritmus található, az alsó szinten a lények osztályai.

A külső ciklus a versenyző lényeken iterál végig, és egyszerre implementálja a pálya-átalakítást és a túlélő lények listájának előállítását. Ezen belül a specifikáció rekurzív függvényének kiszámolása történik, amely több lépésben átalakítja a *lények*[*i*]-t és a *pálya*-t.




A belső ciklus egyszerűsíthető, ha észrevesszük, hogy a rekurzív függvény értéke attól kezdve már nem változik, ha a vizsgált lény életereje elfogyott.



## Megvalósítás

Az absztrakt algoritmust a `main.cpp` állományban elhelyezett *main* függvényben találjuk. Az osztályok definíciói a `creature.h` fejállományba, az `Transmute()` metódusok implementációi a `creature.cpp` forrásállományba kerülnek.

A specifikációban és a programkódban használt elnevezések kapcsolata az alábbi:

| | |
|---|---|
| lény | creature |
| zöldike | greenfinch |
| buckabogár | sandbug |
| tocsogó | squelchy |
| erő | power |
| név | name |
| él | alive |

| | |
|---|---|
| átalakít | transmute |
| pálya | field |
| terep | ground |
| fű | grass |
| homok | sand |
| mocsár | swamp |

### *A főprogram*

A *main* függvény az absztrakt program kódján kívül a lények és a pálya beolvasását is tartalmazza.

A versenyen résztvevő lényekre történő hivatkozásokat, azaz `Creature*` típusú elemeket egy tömbben (`vector<Creature*>`) tároljuk. Így lényegében egy olyan tömböt használunk, amelyik vegyesen tárolhat különböző, de a `Creature` osztályból származtatott osztályú (típusú) elemeket: a tömb egy elemének tehát alternatív szerkezetű típusa van.

```
ifstream f("input.txt");

int n;
f >> n;
vector<Creature*> creatures(n);

for(int i=0; i<n; ++i){
  char l;
  string a;
  f >> l >> a;
  switch(l){
      case 'T' : creatures[i] = new Squelchy(a); break;
      case 'Z' : creatures[i] = new Greenfinch(a); break;
      case 'B' : creatures[i] = new Sandbug(a); break;
  }
}
```

A pálya a pályamezők számkódjait tartalmazó tömbbe (`vector<int>`) kerül.

```
int m;
f >> m;
vector<int> field(m);
for(int j=0; j<m; ++j) f >> palya[j];
```

A feldolgozás kódja némileg átalakult a struktogrammhoz képest, de azzal azonos hatású.

```
    for(int i=0; i<n; ++i){
        bool l = true;
        for(int j=0; j<m; ++j){
            if (l = creatures[i]->Alive())
                creatures[i]->Transmute(palya[j]);
            else break;
        }
        if (l) cout << creatures[i]->Name() << endl;
    }
```

A program végén felszabadítjuk a saját memóriafoglalásainkat.

```
      for(int i=0; i<n; ++i){
        delete creatures[i];
      }
```

### *Creature osztály*

```
class Creature {
  protected:
    std::string name;
    int power;
    Creature(std::string a):name(a) {}
  public:
    std::string Name() const { return name;}
    bool Alive() const { return power > 0;}
    virtual void Transmute(int &gound) = 0;
    virtual ~Creature(){}
};
```

### *Speciális lények osztályai*

```
class Greenfinch : public Creature {
  public:
    Greenfinch(std::string a):Creature(a){ power = 10;}
    void Transmute(int &gound);

};

class Sandbug : public Creature {
  public:
    Sandbug(std::string a):Creature(a){ power = 15;}
    void Transmute(int &gound);

};

class Squelchy : public Creature {
  public:
    Squelchy(std::string a):Creature(a){ power = 20;}
    void Transmute(int &gound);
};
```

A `Transmute()` metódus felüldefiniálásai:.

```
void Greenfinch::Transmute(int &gound)
{
    switch(gound){
        case 1: power+=1; break;
        case 0: power-=2; break;
        case 2: power-=1; gound = 1; break;
    }
}
void Sandbug::Transmute(int &gound)
{
    switch(gound){
        case 1: power-=2; gound = 0; break;
        case 0: power+=3; break;
        case 2: power-=4; gound = 1; break;
    }
}
void Squelchy::Transmute(int &gound)
{
    switch(gound){
        case 1: power-=2; gound = 2; break;
        case 0: power-=5; break;
        case 2: power+=6; break;
    }
}
```

## Tesztelési terv

Érvényes fekete doboz tesztesetek:

1. Nincsenek lények.
2. Nulla hosszúságú a pálya (minden lény életben marad).
3. Egy speciális lény kipróbálása (mindhárom fajtára külön-külön) olyan pályán, ahol egymás után mindhárom fajta terep előfordul, és ezeken a lény végig megy (életben marad). Ehhez a teszthez érdemes kiíratni a megváltozott pályát.
4. Egy speciális lény kipróbálása (mindhárom fajtára külön-külön) olyan pályán ahol a lény életereje elfogy.
5. Külön vizsgálva azt, amikor az utolsó mezőre lépve fogy el egy lény életereje.
6. Általános eset sok lénnyel.

Érvénytelen adatokra nincs felkészítve a fenti program. Nem létező állomány vagy hibás formátumú állomány esetén a program elromlik.

Fehér doboz tesztesetek: A fenti esetek tesztelik a program minden utasítását. Dinamikus helyfoglalások miatt viszont tesztelni kellene még a memóriaszivárgást.

A tesztelésnek részét képezik a modulonkénti tesztek (unit tesztek), amelyek egy-egy osztály szolgáltatásait vizsgálják.