# HÁZI FELADAT

Programozás alapjai 2.

Tervezés

Nagy Zalán

2022. április 21.

---

## TARTALOM

# 1. Feladat

## Töltött részecskék

Készítsen objektummodellt töltött részecskék kölcsönhatásának modellezésére! Ezt követően az egymással kommunikáló objektumok határozzák meg a rájuk ható erőket, és helyváltoztatással határozzák meg a nyugalmi pozíciójukat!
Az objektumok kommunikációját szervezzük gyűrűbe. A gyűrűben egy N elemű vektort körbeküldve mindenki megismerheti a többi objektum helyzetét és töltését, így miden objektum ki tudja számolni a rá ható erőket, amiből számítható az elmozdulás. Az iteráció addig folytatódik, míg be nem áll az egyensúlyi helyzet, vagy az iterációs ciklusszám el nem ér egy előre megadott értéket. Olyan modellt tervezzen, hogy tetszőleges számú részecske is modellezhető legyen!

# 2. Feladatspecifikáció

A program NEM realisztikus modellezést valósít meg.
A programban négy fajta töltés kölcsönhatásai modellezhetők, ezek a következők:

- elektron
- proton
- neutron
- bozon

Az elemi töltések nagyjából a rájuk jellemző fizikai jellemzőkkel rendelkeznek (töltés és tömeg).

Mindegyik részecske saját tulajdonságokkal bír.

- elektron: A hullám tulajdonságai miatt egyes körökben kimarad a kölcsönhatásból.
- proton: Egy körben több lépést is megtehet. (Számára az „idő múlása” (lépészámok) nagyságrendje más, ennek a következménye, hogy gyorsabbnak tűnik a mozgása)
- neutron: Mivel nincsen töltése, ezért másképpen hat a többi részecskére, gravitációs vonzóerővel, tömege lényegesen nagyobb, mint a valóságban.
- bozon: Véletlenszerűségének köszönhetően minden körben más a töltése.

A töltések rögzítésére is van lehetőség a megadott pozícióban.
A töltések ütközésekor, azaz amikor egy pozícióba kerülnek, akkor rögzített töltésekké válnak, onnan „nem tudnak megegyezni” hová haladjanak tovább.

A program a modellezéshez szükséges adatokat egy fájlból olvassa be.
Ennek a fájlnak az első sorában a részecskék darabszáma szerepel(N).
Ezt követő N sorban pedig a töltés típusa (e, p, n, b), majd a helyvektorának három koordinátája x y z sorrendben.
Ezután pedig, hogy rögzített-e a töltés. (1-rögzített, 0-nem rögzített).
Ezeken kívül tartalmazza az időintervallumot (lépésszámot), ameddig a modellezés tartson.

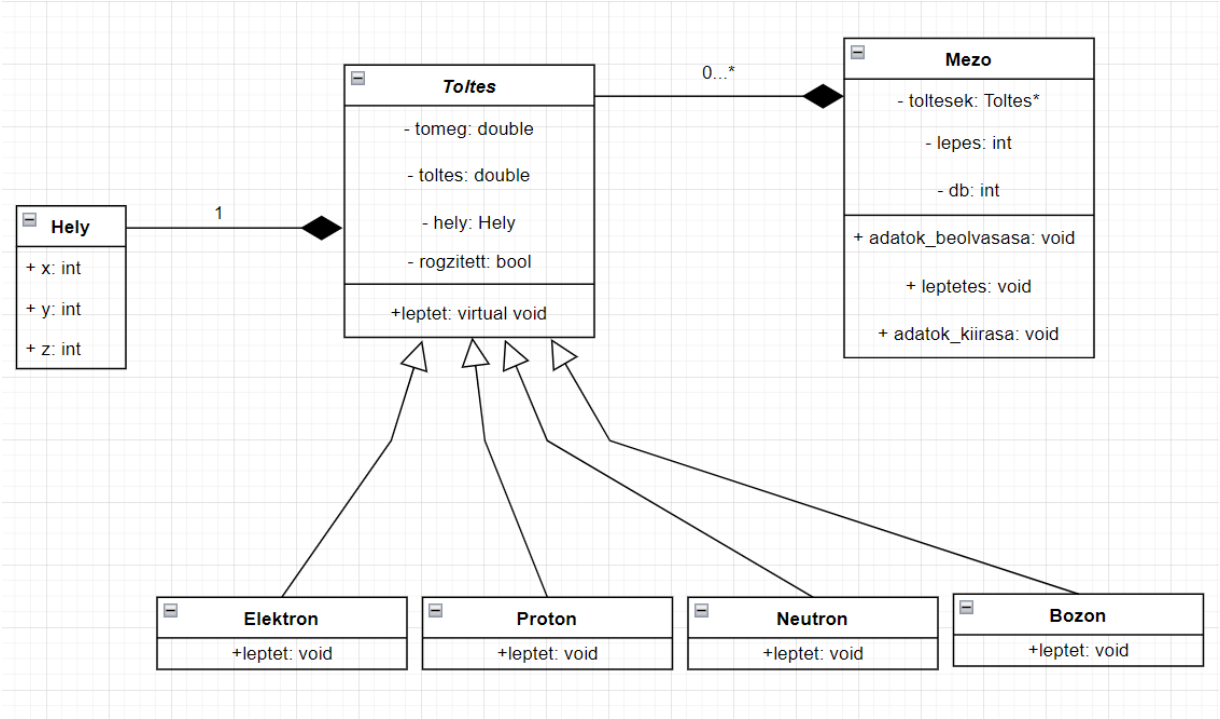
A program a kimenetet egy fájlba írja, amelyben az eltelt idő (lépésszám) utáni pozíció jelenik meg minden részecskére a bemeneti sorrendben, valamint, hogy kialakult-e egyensúlyi helyzet.

# 3. Terv

## 3.1. Szerkezet

A feladat egy olyan tároló létrehozása, mely egy fix méretű dinamikusan allokált memóriaterületen tárolja az töltéseket, azoknak is több fajtáját. Így egy heterogén kollekciót valósítok meg. A *Toltes* osztályból négy további osztály származik, melyeknek jelentőségét a 2.bekezdés taglalja. Egy töltés pozícióját egy osztályban tárolom el, mely a tér három koordinátájával jellemzi a pozíciót.

## 3.2. Osztálydiagram




## 3.3. Algoritmusok

### 3.3.1. Adatok beolvasása

A bemeneti fájl első sorából a töltések darabszámát olvassa be. Ezután dinamikusan lefoglalja a memóriában a szükséges helyet és feltölti a további sorok adataival. Az utolsó sorból a szükséges lépésszámot olvassa be.

### 3.3.2. Léptetés

A *Mezo* osztály egy objektumára meghívódik a *leptetes* függvény, ami minden *Toltes* objektumra meghívja a megfelelő függvény, így tudnak a részecskék a saját tulajdonságaik

szerint viselkedni. A kezdetben megadott lépésszámig minden lépésben kiszámítjuk a töltésekre ható erőket és ennek megfelelően új pozícióba kerülnek, de az is előfordulhat, hogy maradnak az eredeti helyükön. A töltések között egy vektort küldünk körbe, melynek segítségével meg tudják határozni a rá ható erőket és az új pozíciót. Ezután egyszerre kerülnek a töltések az új helyükre a léptetés végén.

### 3.3.3. Adatok kiírása

Az elmozgatott töltések új pozíciói a lépésszám letelte után vagy pedig az egyensúlyi helyzet beálltakor a bemeneti sorrendben kiíródnak egy kimeneti fájlba, majd a program futásának befejezésének oka is kiíródik. (Beállt-e az egyensúlyi helyzet vagy sem)