2. feladat

Részecske adatok feldolgozása struktúrák segítségével

Egy kísérletben mérjük a részecskék impulzusának p_x és p_y komponensét és ezt egy fájlban egymás mellé feljegyezzük. A részecskéknek van még egy szabadsági fokuk, amelyet színnek nevezünk és ezt is feljegyezzük az impulzus komponensek után. Három különböző szín lehet, az egyes színeket egy karakterrel jelöljük: p (piros), f (fehér) és z (zöld).

A feladatban a részecskék adatait kell különbözú szempontok szerint feldolgozni.

A program a következő parancssori argumentumokat olvassa be, pontosan ebben a sorrendben:

adatfájl_neve keresendő energiaérték

Az adatfájlok megtalálhatóak a halnum.public/feladat2 könyvtárban.

A program végezze el egymás után az a)-e) pontokban felsorolt műveleteket.

Figyeljünk a következőkre:

- a program fejlesztése közben kis adatfájllal dolgozzunk, végén pedig próbáljuk ki egy nagy fájllal is! Ezekre példa található a k8plex-en a halnum.public mappában: particle_small.dat és particle_large.dat. A k8plex-en való futtatáshoz ezeket a fájlokat ne másoljuk be a feladat2 mappába, hanem használjuk a /v/courses/halnum.public/feladat2/ elérési útvonalat!
- a programot strukturáltan írjuk meg: a logikailag különálló műveletek külön függvényekben legyenek!

a)

Definiáljunk egy struktúra adattípust, amely az egyes részecskékre vonatkozó adatokat tárolja! Ezután a fájlból olvassuk be az adatokat és tároljuk őket egy, a fentebb definiált struktúrákat tartalmazó vektorban. Próbáljuk úgy megoldani a beolvasást, hogy ne kelljen előre megadni a vektor méretét (tehát azt, hogy hány részecske adatait fogja tartalmazni)! Ha ez esetleg gondot okoz, akkor kellően nagy, fix vektorméretet is alkalmazhatunk.

Ezután a program keresse meg a legnagyobb energiájú részecskét! Ultrarelativisztikus részecskékről lévén szó, az energiájukat az impulzus függvényében a $\sqrt{p_x^2 + p_y^2}$ képlet adja meg. A program írja ki a legnagyobb energia értéket és a hozzá tartozó részecske színét is!

Az előadáson kétféle *for* ciklust is tárgyaltunk. Gondoljuk át, melyiket lehet/célszerű az a) és b) rész megoldásához használni! (A)

b)

Írjunk egy-egy függvényt, amely kiszámolja a részecske energiák átlagát és a szórását, és a program írja ki ezeket! (A)

c)

Írjunk egy függvényt, amelyik ellenőrzi, hogy a parancssori paraméterként megadott energiaérték megegyezik-e valamelyik részecske energiájával? Ha igen, akkor írja ki, hogy melyik részecskéről van szó, és azt is, hogy hány összehasonlításra volt szükség a részecske megtalálásához! (A)

d)

Rendezze át a program a beolvasott adatokat olyan sorrendbe, amely a számolt energia értékek csökkenő sorrendjének felel meg! Ehhez használjuk a buborékos rendezést! Ha legfeljebb 20 eleme van a vektornak, akkor a program írja ki a sorrendezett energia értékeket és a hozzájuk tartozó részecskék színét is! (T)

e)

Ha legfeljebb 20 eleme van a vektornak, akkor a program írja ki energia szerint sorbarendezve előbb a piros részecskék energiáját, majd a fehérekét és végül a zöldekét! (T)

f) Szorgalmi:

Tekintsük a d) feladatrészben sorbarendezett vektort! Keresse meg a program ebben a vektorban a parancssori bemenetként megadott energiaértéket és írja ki az indexét, valamint az összehasonlítások számát! Kihasználva, hogy a c) résszel ellentétben az adatvektor már energia szerint sorrendezett, találjunk olyan algoritmust, hogy minél kevesebb összehasonlításra legyen szükség! Tudunk-e olyan algoritmust találni, amely kevesebb összehasonlítást igényel, mint a buborékos rendezés?

g) Szorgalmi

Ismerkedjünk meg a *quick sort* algoritmussal és írjunk egy saját implementációt. Futassuk le a részecskék adatait tartalmazó, nem sorrendezett adatvektoron és rendezzük nagyság szerint sorba az energia értékek szerint! Írjuk ki az elvégzett összehasonlítások számát és hasonlítsuk össze a d) feladatrészeben használt buborékos rendezéssel!