.NET FEJLESZTÉS C# NYELVEN

3. forduló

RENDELKEZÉSRE ÁLLÓ IDŐ:

16:00

Ismertető a feladathoz

Fontos információk

> Ezután a forduló után automatikusan jár a kitartóknak szóló garantált ajándékunk, érdemes kitöltened a feladatlapot! :)
>
> Ha kifutsz az adott feladatlap kitöltésére rendelkezésre álló időből, a felület **automatikusan megpróbálja beküldeni** az addig megadott válaszokat.
>
> A kérdésekre **mindig van helyes válasz,** olyan kérdés viszont nincs, amelyre az összes válasz helyes!

Egyéb információkat a versenyszabályzatban találsz!

Harmadik forduló

Az ajtót kinyitva a folyosó végén egy árnyat látsz elsuhanni, utána szaladsz, és a karja után nyúlsz. Rátaláltál Vikire. Viki egy Junior fejlesztő, aki szintén a hajón utazik. Megtudod, hogy a hajót fenntartó elsődleges microservice alapú rendszer részlegesen meghibásodott. A rendszer próbálja helyreállítani az elsődleges szolgáltatásokat több kevesebb sikerrel. A hajó egy aszteroida mező felé tart, viszont a pajzsok modulációért felelős szolgáltatás leállt. Egy egyszerű mock service segítségével átadható lenne a helyes pajzs konfiguráció a rendszer számára. A parancsnoki híd felé indultok.

Felhasznált idő: 00:16/16:00 **Elért pontszám:** 0/8

1. feladat 0/2 pont

Rengeteg elemen végre kell hajtanod egy összetett számítást.

Milyen technikát használnál erre?

Válasz

- ○ Minden elemet egy külön Task-ban végzi a saját számítását és megvárod az összes Task-ot.
- ◯ Parallel.For-t használsz hogy egyidejüleg végezz el minden feladatot.
- ○ A dotnet keretrendszer magától kioptimalizálja a számításokat.
- ○ Minden elemet egy BlockingCollection-be helyezel és egy külön szálon dolgozod fel őket.

Magyarázat

A külön Taskok létrehozása nem szükséges. A Parallel osztály gondoskodik a létrehozásról és az optimális beállításokról.

Az async/await nem végez konkurens feldolgozást.

A Parallel.For az ideális választás.

The Blockingcollection a szálak között osztja meg az adatokat. Egy producer és egy consumer szál, esetén nem javítja a skálázhatóságot.

2. feladat 0/3 pont

Viki a kódbázis böngészése közben talált egy furcsa kódrészletet.

A kódrészlet egy 75000 elemű ügyféllistával dolgozik.

Szerinted melyik futása tart a legtovább?

```
IEnumerable<(string N, int U)> list = /*.....*/

var list2 = list.Where(x => x.N.Length == list.Max(l => l.N.Length));

var list3 = list2.SelectMany(t => t.N)
  .Where(x => x.ToString().Length == list.Max(l => l.N.Length));

var list4 = list2.ToList().AsParallel()
  .ForAll(n => Console.WriteLine(n);
```

Válaszok

- [ ] list2
- [x] list3
- [x] list4

Magyarázat

A második és a harmadik sor esetén nem hajtja végre a kiértékelést kizárólag a negyedik sor esetén.

Frissítve (2021.11.08. 14:15): a kódban van egy elírás (hiányzik egy bezáró zárójel a pontosvessző elől), illetve, ha be is lett volna zárva, akkor is void a visszatérési érték, így nem rakható listába az eredmény. Ezért a list3 megoldás is elfogadott, mint a lefuttathatók közül a legidőigényesebb.

3. feladat 0/3 pont

Sikerült megszerezni a meteorzápport esetén használt pajzs modulációért felelős kódrészletet.

Sajnos a tabletet nem rendelkezik elengedő erőforrással a kód futtatásához.

Mit ír ki az alábbi kódrészlet?

```
for (int i = 0; i <= 2; ++i) { Console.Write(i); }
var l = Enumerable.Range(1, 10);
l.Where(predicate: i => { Console.Write(i); return i % 5 == 0; });
Console.Write(l.Last());
```

Válaszok

A helyes válasz:
01210
0,1,2,10
0 1 2 10

Magyarázat

A for (int i = 0; i <= 2; ++i) { Console.Write(i); } kiírja a 012 karaktereket.

{ Console.Write(i); return i % 5 == 0; }; nem kerül kiértékelésre.

A Console.Write(l.Last()); kimenetele pedig 10.

Így kapjuk 01210 kimenetet.