9. óra

<table>
<tr><th>Idő</th><th>Óra menete</th><th>Leírás</th><th>Comment</th></tr>
<tr><td>0-5</td><td>Köszönés, ismétlés</td><td>Gépek bekapcsolása.
Mi volt múlt órán?</td><td>Válasz: Sorozatszámítás, függvény</td></tr>
<tr><td>5-8</td><td>Struktúra</td><td>A struktúrákat azért használjuk, mert így a kapcsolódó adatokat nem kell külön tömbökben tárolni (lásd múlt óra), hanem egy tömbben ugyan azon az indexen elérhető például az útdíja, a megtett távja is a múlt órai futárnak.</td><td></td></tr>
<tr><td>8-55</td><td>Struktúra létrehozása</td><td>
Feladat (2020 május érettségi): Olvassa be és tárolja el a 9. input.txt állomány adatait! Majd kérje be a felhasználótól egy város kódját! Adja meg, hogy az adott városból mikor érkezett az utolsó mérési adat! A kiírásban az időpontot óó:pp formátumban jelenítse meg! Forrásfájl felépítése(max 500 sor, időrendben van):
<table>
<tr><td>település</td><td>szöveg (2 karakter)</td><td>A település kétbetűs kódja</td></tr>
<tr><td>idő</td><td>szöveg (óópp formátumban)</td><td>A mérés időpontja</td></tr>
<tr><td>szélirány
és -erősség</td><td>szöveg (5 karakter) szélirány
3 karakter, -erősség 2 karakter</td><td>A szél iránya fokban vagy szöveggel
és sebessége csomóban megadva</td></tr>
<tr><td>hőmérséklet</td><td>egész szám (2 karakter)</td><td>Mért hőmérséklet (nem negatív)</td></tr>
</table>
A struktúrákat hasonlóan külön kell létrehozni, mint a függvényeket a fő függvényen kívül. Alapból a benne levő változók private típusúak, azaz csak a struktúrán belül érhetőek el, ezért kell elé a public előtag:
<pre>
struct Egyadat{
    public string telepules;
    public string ido;
    public string szeliranyerosseg;
    public int homerseklet;
}
</pre>
Ezek után a fő függvényben fogjátok megcsinálni a többit. Először is beolvassátok a filet és azt el is kell tárolni.
Ehhez kell a System.IO használata. Majd egy StreamReader:
<pre>
StreamReader olvas = new StreamReader("9. input.txt");
</pre>
A sorok adatait is el kell tárolni, de 4 oszlopba vannak rendezve. Ehhez egy 4 elemű sor tömb elég nekünk:
<pre>
string[] sor = new string[4];
</pre>
Kell egy lista, amiben az adatokat eltároljátokk, de itt nem string vagy int típust használtok, hanem azt a struktúrát, amit létrehoztatok:
<pre>
List<Egyadat> adatok = new List<Egyadat>();
</pre>
Ahhoz, hogy el választva tároljátok el az adatokat, kell egy segéd változó, ami ugyan úgy a struktúrát használja:
<pre>
Egyadat adat = new Egyadat();
</pre>
majd beolvassa az első sort, hogy később a do while ciklusban egyszerűbben meg tudjátok vizsgálni, mikor jön üres sor a file-ban. Ezt a sort a sor tömb 0. elemére olvassuk be így nem kell még egy változó.
<pre>
sor[0] = olvas.ReadLine();
</pre>
Majd a do while ciklus addig fusson, amíg üres sort nem talál:
<pre>
do{
}while(sor[0] != null);
</pre>
A ciklusban vágjuk fel a sorokat, hogy külön el tudjátok menteni az adatokat. Ezt a split függvény megcsinálja:
<pre>
            sor = sor[0].Split(" ");
</pre>
</td><td></td></tr>
</table>

		Itt a sor tömböt a 0. indextől feltölti az adatokkal, először fogja felvágni és csinál egy 4 elemű tömböt és utána menti el. Így a 0. elemen a település lesz látható, az 1. indexen az idő stb… Ezt kell elmenteni a segédváltozóba:
		`adat.telepules = sor[0];`
		`adat.ido = sor[1];`
		`adat.szeliranyerosseg = sor[2];`
		`adat.homerseklet = int.Parse(sor[3]);`
		Itt jól látszik, hogy használja a struktúránkat és úgy mentette el, a hőmérsékletet meg átalakítottjátok int típussá, mert az egész szám.
		Ezek után már csak a listához kell adni:
		`adatok.Add(adat);`
		Aztán beolvastok még egy sort, hogy ellenőrizzétek, van-e még a txt-ben sor:
		`sor[0] = olvas.ReadLine();`
		Ha ezzel megvagytok, akkor elmentettitek az összes adatot a txt-ből és neki állhattok a feladatnak. Kérjetek be egy városkódot:
		`string bekertvaroskod = Console.ReadLine();`
		majd hozzatok létre egy segédváltozót, amibe az utolsó időt fogjátok tárolni:
		`string utolsoadat = "";`
		Mivel időrendben vannak az adatok megadva nekünk, így nem kell ellenőrizni, hogy nagyobb érték kerül-e bele majd a változóba, egyszerűen csak felülírja az előző benne levő értéket. Mivel struktúrát használtatok, ezért for ciklussal végig mentek a listán:
		`for(int i = 0; i < adatok.Count(); i++){`
		`}`
		Ebben a for ciklusban csak annyit kell ellenőrizni, hogy a bekért város kódja megegyezik-e az aktuális város kódjával, ha igen akkor az utolsoadat változót csak felülírjátok:
		`if(adatok[i].telepules == bekertvaroskod){`
		`utolsoadat = adatok[i].ido;`
		`}`
		Ezek után már csak két változóba szétszedjük az óra és a perc értékeket és irathatjuk is ki a képernyőre a megoldást:
		`string ora = utolsoadat[0].ToString();`
		`ora += utolsoadat[1].ToString();`
		`string perc = utolsoadat[2].ToString();`
		`perc += utolsoadat[3].ToString();`
		A toString azért kell, mert ha a szöveget indexeltek, akkor karaktereket kaptok, mint a valóságban is és egy string változóba csak stringet tudtok tárolni.
		Így megvannak külön az óra és a perc értékek, amikkel nem kell számolni, így szövegként mehet kiírásra:
		`Console.WriteLine(ora + ":" + perc);`	

55-60	Elköszönés	Mentsék el a projektet, nyugodtan vigyék haza (GitHub). Gépek kikapcsolása. Pozitív értékelés! + Jutalom: CUKORKA Elköszönés	Cukorka, matrica csak abban az esetben jár, ha megérdemlik!

Az otthoni gyakorló feladatok a 9.hazi.cs file-ban elérhetőek.