

Idő	Óra menete	Leírás	Comment												
0-5	Köszönés, ismétlés	Gépek bekapcsolása. Mi volt múlt órán?	Válasz: Sorozatszámítás, függvény												
5-8	Struktúra	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.													
8-55	Struktúra létrehozása	<p>Feladat (2020 május érettségi): Olvassa be és tárolja el a <i>9. input.txt</i> 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):</p> <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> <p>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:</p> <pre>struct Egyadat{ public string telepules; public string ido; public string szeliranyerosseg; public int homerseklet; }</pre> <p>Ezek után a fő függvényben fogjuk megcsinálni a többit. Először is beolvassuk a filet és azt el is kell tárolni.</p> <p>Ehhez kell a System.IO használata. Majd egy StreamReader:</p> <pre>StreamReader olvas = new StreamReader("9. input.txt");</pre> <p>A sorok adatait is el kell tárolni, de 4 oszlopba vannak rendezve. Ehhez egy 4 elemű sor tömb elég nekünk:</p> <pre>string[] sor = new string[4];</pre> <p>Kell egy lista, amiben az adatokat eltároljuk, de itt nem string vagy int típust használunk, hanem azt a struktúrát, amit létrehoztunk:</p> <pre>List<Egyadat> adatok = new List<Egyadat>();</pre> <p>Ahhoz, hogy el választva tároljuk el az adatokat, kell egy segéd változó, ami ugyan úgy a struktúránkat használja:</p> <pre>Egyadat adat = new Egyadat();</pre> <p>majd beolvassuk az első sort, hogy később a do while ciklusban egyszerűbben meg tudjuk vizsgálni, mikor jön üres sor a file-ban. Ezt a sort a sor tömb 0. elemére olvassuk be így nem kell még egy változó.</p> <pre>sor[0] = olvas.ReadLine();</pre> <p>Majd a do while ciklusunk addig fusson, amíg üres sort nem talál:</p> <pre>do{ }while(sor[0] != null);</pre> <p>A ciklusban vágjuk fel a sorokat, hogy külön el tudjuk menteni az adatokat. Ezt a split függvény megcsinálja nekünk:</p>	település	szöveg (2 karakter)	A település kétbetűs kódja	idő	szöveg (óópp formátumban)	A mérés időpontja	szélirány és -erősség	szöveg (5 karakter) szélirány 3 karakter, -erősség 2 karakter	A szél iránya fokban vagy szöveggel és sebessége csomóban megadva	hőmérséklet	egész szám (2 karakter)	Mért hőmérséklet (nem negatív)	
település	szöveg (2 karakter)	A település kétbetűs kódja													
idő	szöveg (óópp formátumban)	A mérés időpontja													
szélirány és -erősség	szöveg (5 karakter) szélirány 3 karakter, -erősség 2 karakter	A szél iránya fokban vagy szöveggel és sebessége csomóban megadva													
hőmérséklet	egész szám (2 karakter)	Mért hőmérséklet (nem negatív)													

		<pre>sor = sor[0].Split(" ");</pre> <p>Itt a sor tömböt a 0. indextől feltölti az adatokkal, először fogja felválni 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 elmentenünk a segédváltozónkba:</p> <pre>adat.telepules = sor[0]; adat.ido = sor[1]; adat.szeliranyerosseg = sor[2]; adat.homerseklet = int.Parse(sor[3]);</pre> <p>Itt jól látszik, hogy használtuk a struktúránkat és úgy mentettük el, a hőmérsékletet meg átalakítottuk int típusú, mert az egész szám.</p> <p>Ezek után már csak a listánkhoz kell adni:</p> <pre>adatok.Add(adat);</pre> <p>Aztán beolvassunk még egy sort, hogy ellenőrizzük, van-e még a txt-ben sor:</p> <pre>sor[0] = olvas.ReadLine();</pre> <p>Ha ezzel megvagyunk, akkor elmentettük az összes adatot a txt-ből és neki állhatunk a feladatnak. Kérjünk be egy városkódot:</p> <pre>string bekertvaroskod = Console.ReadLine();</pre> <p>majd hozzunk létre egy segédváltozót, amibe az utolsó időt fogjuk tárolni:</p> <pre>string utolsoadat = "";</pre> <p>Mivel időrendben vannak az adatok megadva nekünk, így nem kell ellenőriznünk, hogy nagyobb érték kerül-e bele majd a változónkba, egyszerűen csak felülírjuk az előző benne levő értéket. Mivel struktúrát használtunk, ezért for ciklussal végig megyünk a listánkon:</p> <pre>for(int i = 0; i < adatok.Count(); i++){</pre> <p>Ebben a for ciklusban csak annyit kell ellenőriznünk, hogy a bekért város kódja megegyezik-e az aktuális város kódjával, ha igen akkor az utolsoadat változónkat csak felülírjuk:</p> <pre>if(adatok[i].telepules == bekertvaroskod){ utolsoadat = adatok[i].ido; }</pre> <p>Ezek után már csak két változóba szétszedjük az óra és a perc értékeket és irathatjuk is ki a képernyőre a megoldást:</p> <pre>string ora = utolsoadat[0].ToString(); ora += utolsoadat[1].ToString(); string perc = utolsoadat[2].ToString(); perc += utolsoadat[3].ToString();</pre> <p>A toString azért kell, mert ha a szöveget indexeljük, akkor karaktereket kapunk, mint a valóságban is és egy string változóba csak stringet tudunk tárolni. Így megvannak külön az óra és a perc értékek, amikkel nem kell számolnunk, így szöveggént mehet kiírásra:</p> <pre>Console.WriteLine(ora + ":" + perc);</pre>	
55-60	Elköszönés	<p>Mentsék el a projektet, nyugodtan vigyék haza (GitHub).</p> <p>Gépek kikapcsolása.</p> <p>Pozitív értékelés! + Jutalom: CUKORKA</p> <p>Elköszönés</p>	<p>Cukorka, matrica csak abban az esetben jár, ha megérdemlik!</p>

Az otthoni gyakorló feladatok a [9.hazi.cs](#) file-ban elérhetőek.