**13. tétel**

**Adatbázis- és szoftverfejlesztés - Programozási alapismeretek (2.1.1.)**

**Programozási tételek és felhasználási lehetőségeik**

Ismertesse milyen alapvető programozási tételeket ismer, miért és mikor használjuk őket, mi az alapvető jellemzőjük, milyen elemekből épülnek fel és miért.

***Szempontok a tartalom értékeléséhez***

• Ismerteti az összegzés és az átlagszámítás tételét, mire jók, illetve mik a közös részek, valamint miben különböznek egymástól

• Bemutatja a minimum és maximumkeresés tételét, mire jók, illetve mik a közös részek, valamint miben különböznek egymástól

• Kitér a megszámlálás tételére, bemutatja a hasonlóságokat és a különbségeket. Mikor és miért használjuk őket?

• Halmazműveletekkel kapcsolatot tételekről említést tesz, elmagyarázza azok lényegét, és a közöttük lévő szerkezeti különbségeket illetve hasonlóságokat.

• A felsorolt tételek közül legalább kettő tételt folyamatábrával is bemutat, elmagyaráz.

**Összegzés tétele**

Az összegzés tételének kódjával képesek lehetünk egy adott tömb elemeit egyesével végigvizsgálni, és azokat összegét egy korábban létrehozott változóban letárolni annak értékének folyamatos növelésével.

**int osszeg = 0;** *//létrehozzuk az összeg változót, kezdőértéknek 0-t adunk meg*

**for (int i = 0; i < szamok.Length;i++)***//végigmegyünk a tömbön ciklussal*

**{**

**osszeg = osszeg + szamok[i];** *//növeljük az összeg értéket az aktuálisan vizsgált változó értékével*

**}**

**Console.WriteLine("\nA számok összege:{0}",osszeg);** *//kiíratjuk az összeget*

**Átlagszámítás tétele**

Átlagszámítás szinte teljesen megegyezik az összegzés tételével, azaz végigvizsgálunk egy tömböt, és egy változóba elkezdjük összeadni a tömb elemeinek értékét. A különbség csupán annyi, hogy az eredmény megjelenítésekor, ezt az összeget el kell osztani az elemszámmal, ami nem más, mint a tömb mérete.

**int osszeg = 0;** *//létrehozzuk az összeg változót, kezdőértéknek 0-t adunk meg*

**for (int i = 0; i < szamok.Length;i++)***//végigmegyünk a tömbön ciklussal*

**{**

**osszeg = osszeg + szamok[i];** *//növeljük az összeg értéket az aktuálisan vizsgált változó értékével*

**}**

**Console.WriteLine("\nA számok átlaga:{0}",osszeg/tomb.Lenght);** *//kiíratjuk az átlagot, ami nem más, mint az összeg elosztva a tömb méretével.*

**Megszámlálás tétele**

Megszámlálás tételével, mint a legtöbb programozási tétellel, végignézzük a tömb elemeit és megszámoljuk, hogy hány darab t tulajdonságú elem van benne. Ehhez nincs másra szükségünk, mint egy számlálóra, valamint a tömb bejárására használt ciklusban egy feltételre, ami tartalmazza a t tulajdonságot. (feltétel lehet pl.: a páros számok megszámlálása)

**int db = 0;**

**for (int i = 0; i < szamok.Length; i++)***//végigmegyünk a tömbön ciklussal*

**{**

**if (szamok[i] %2==0***)//A for cikluson belül ágyazott if-fel elvégezzük a feltétel vizsgálatot*

**{**

**db = db + 1***;//ha a feltétel megfelelő növeljük a db változó értékét eggyel*

**//db=db+1;** *helyett írhatunk "db++" -t is akár*

**}**

**}**

**Kiválogatás tétele**

Kiválogatás tétele esetén hasonlóan járunk el, mint megszámlálás tételekor, keressük az adott tömbben a t tulajdonsággal rendelkező elemeket, ám ahelyett, hogy ezeket egyszerűen csak megszámolnánk, kiírhatjuk a képernyőre, esetleg kimenthetjük egy másik összetett változóba azt. Ilyenkor csupán egy felteltére van szükségünk a vizsgálatot végző ciklusban. (feltétel lehet pl.: a páros számok megszámlálása)

**Console.WriteLine("Páros számok:");**

**for (int i = 0; i < szamok.Length; i++)** *//végigmegyünk a tömbön ciklussal*

**{**

**if(szamok[i]%2==2)** *//A for cikluson belül ágyazott if-fel elvégezzük a feltétel vizsgálatot*

**{**

**Console.Write("{0}, ", szamok[i]);** *//ha a feltétel megfelelő kiíratjuk a számot*

**}**

**}**

**Minimum kiválasztás tétele**

Minimum kiválasztás esetén keressük a tömbben megtalálható legkisebb elemet, arra viszont ügyelnünk kell, hogy a keresést segítő minimum értéket tároló változó értéke, sosem lehet kisebb, mint a tömb legkisebb eleme, mert így az sosem fog módosulni. Érdemes a használatához a tömb egyik elemének értékét kezdőértéknek meghatározni, majd ezután megvizsgálni a tömb többi elemét, és ha találunk az adott minimum értéknél kisebbet, módosítjuk azt az éppen vizsgált értékre. Esetleg tárolhatjuk a legkisebb értékű változónak a helyét is, az alábbi módon.

**int min = 0;** *//létrehozunk egy változót, amiben tároljuk a legkisebb változó helyét*

**for (int i = 1; i < szamok.Length; i++)** *//végigmegyünk a tömbön ciklussal*

**{**

**if (szamok[i] < szamok[min])** *//megvizsgáljuk kisebb-e az aktuális szám, mint az eddigi legkisebb*

**{**

**min = i;** *//kimentjük az aktuális legkisebb változó értékének helyét*

**}**

**else { }** *//különben nem csinálunk semmit*

**}**

**Console.WriteLine("\nA legkisebb szám az tömbben0}",szamok[min]);** *//kiíratjuk az eredményt*

**Maximum kiválasztás tétele**

Maximum kiválasztás esetén keressük a tömbben megtalálható legnagyobb elemet, arra viszont ügyelnünk kell, hogy a keresést segítő maximum értéket tároló változó értéke, sosem lehet nagyobb, mint a tömb legnagyobb eleme, mert így az sosem fog módosulni. Érdemes a használatához a tömb egyik elemének értékét kezdőértéknek meghatározni, majd ezután megvizsgálni a tömb többi elemét, és ha találunk az adott maximum értéknél nagyobbat, módosítjuk azt az éppen vizsgált értékre. Esetleg tárolhatjuk a legnagyobb értékű változónak a helyét is, az alábbi módon.

**int max = 0;** *//létrehozunk egy változót, amiben tároljuk a legnagyobb változó helyét*

**for (int i = 1; i < szamok.Length; i++)** *//végigmegyünk a tömbön ciklussal*

**{**

**if (szamok[i] > szamok[max])** *//megvizsgáljuk nagyobb-e az aktuális szám, mint az eddigi legkisebb*

**{**

**max = i;** *//kimentjük az aktuális legnagyobb változó értékének helyét*

**}**

**else { }** *//különben nem csinálunk semmit*

**}**

**Console.WriteLine("\nA legnagyobb szám az tömbben0}",szamok[max]);** *//kiíratjuk az eredményt*

**Halmazműveletek**

Említést kell tenni még a halmazműveletekről is, itt egy összetett adatszerkezet helyett többet hasonlítunk össze.(itt érdemes tömbök helyett listákat használni, mivel nem tudhatjuk, hány elem fog majd a végén belekerülni)

**UNIÓ** esetén két adott lista **(a, b)** elemeit mentjük egy harmadikba **(unio),** ügyelve arra, hogy amennyiben a listában vizsgált elem már szerepelt korábban az unió listában, azt ismételten ne írjuk bele.

**METSZET** létrehozása alkalmával, megvizsgáljuk, hogy az első **(a)** lista elemeit egyesével, hogy szerepelnek-e a második **(b)** listában. Amennyiben egyezést találunk, beleilleszthetjük az a metszet listába, hogyha még nem szerepelt korábban.

**KÜLÖNBSÉG** alkalmával pedig, hasonlóan járunk el mint a metszetnél, de míg ott a közös elemeket keressük, itt egy elem akkor kerül csak a különbség listába, ha az a másikban nem található meg.

|  |  |
| --- | --- |
| 1. ábra – Átlagszámítás tétele | C:\Users\Cyb\AppData\Local\Microsoft\Windows\INetCache\Content.Word\C# Összegzés tétele.png  2. ábra- Összegzés tétele |
| 3. ábra- Minimum kiválasztás tétele | 4. ábra - Maximum kiválasztás tétele |