**Programozás I. – 25 tavasz**

**Forráskódok**

HelloMilton

namespace HelloMilton

// a 'szokásos' első program

{

internal class HelloMilton

{

static void Main(string[] args)

{

Console.WriteLine("Hello Milton!");

// szöveg kiírása az alapértelmezett kimentre

Console.ReadKey();

// egy billentyű leütésre vár (hogy megmaradjon az eredmény, ne zárja be azonnal a console-t

}

}

}

Osszegzes

namespace Osszegzes

// tömb elemek értékenk összegzése

{

internal class Osszegzes

{

static void Main(string[] args)

{

// tömb létrehozás - minden tételnél ua.

int[] Tomb = { 5,2,4,5,2,6,3,4,1,5,6,3,7,8,9,2,4,8,3,1};

// itt csak megadjuk hogy hány eleme van a tömbnek, később lesz más is

int Darab = 20;

// egy változó amelyben összehalmozzuk a tömb elemeinek értékét

int Osszeg = 0;

// ciklus mellyel végig járjuk a tömb minden elemét, és az osszeg változóban tároljuk el a tömbelemek összegét

for (int i = 0; i < Darab; i++)

{

Osszeg = Osszeg + Tomb[i];

}

// kiíratjuk az eredményt ... némi szöveggel

Console.WriteLine("Összeg: " + Osszeg);

// várunk egy beillentyű leütésre a kilépéshez

Console.ReadKey();

}

}

}

Megszamolas

namespace Megszamolas

// a megadott feltételnek megfelelő elemeket számolja meg

{

internal class Megszamolas

{

static void Main(string[] args)

{

// tömb létrehozás - minden tételnél ua.

int[] Tomb = { 5, 2, 4, 5, 2, 6, 3, 4, 1, 5, 6, 3, 7, 8, 9, 2, 4, 8, 3, 1 };

// a tömb típus Length metódusát használjuk a tömb méretének meghatározására

int Hossz = Tomb.Length;

// ebbe a változóban tároljuk a feltételnek megfelelő elemek számát

int Darab = 0;

// végig megyünk a tömb minden elemén és vizsgáljuk a feltétel teljesülését

for (int i = 0; i < Hossz; i++)

{

//legyen a feltétel hogy a 4-nél nagyobb elemket számoljuk össze

if (Tomb[i] > 4)

{

// a Darab változó értékét növeljük

Darab++;

}

}

// kiírjuk a tömb elemeit, hogy ellenőrizhessük az eredményt

for (int j = 0; j < Hossz; j++)

{

//kiírja a tömb j-edik elemét, és egy vesszőt, hogy formázza a listát

Console.Write(Tomb[j] + ",");

}

// soremelés

Console.WriteLine();

// kiírjuk az eredményt is

Console.WriteLine("A 4-nél nagyobb elemek száma: " + Darab);

// várunk egy beillentyű leütésre a kilépéshez

Console.ReadKey();

}

}

}

Vizsgalat

namespace Vizsgalat

// feltétel vizsgálat

{

internal class Vizsgalat

{

static void Main(string[] args)

{

// tömb létrehozás - minden tételnél ua.

int[] Tomb = { 5, 2, 4, 5, 2, 6, 3, 4, 1, 5, 6, 3, 7, 8, 9, 2, 4, 8, 3, 1 };

// a tömb típus Length metódusát használjuk a tömb méretének meghatározására

int Hossz = Tomb.Length;

// a vizsgálat eredményét tároló változó

string Eredmeny = "";

// ciklus a tömb vizsgálatára

for (int i = 0; i < Hossz; i++)

{

// a vizsgálat legyen az hogy van-e 7-es a tömbben

if (Tomb[i] == 7)

{

// igaz ág, van 7-es a tömbben, Eredmeny értékadás

Eredmeny = " Van 7-es a tömbben";

// 'elegánsan' kilépünk a ciklusból, nincs értelme a további vizsgálatnak

i = Hossz + 1;

}

else

{

// hamis ág, Eredmeny értékadás

Eredmeny = "Nincs 7-es a tömbben";

}

}

// kiírjuk a vizsgálat eredményét

Console.WriteLine("A vizsgálat eredménye: " + Eredmeny);

// várunk egy billentyű leütésre a kilépéshez

Console.ReadKey();

}

}

}

MaxMinKivalasztas

namespace MaxMinKivalasztas

// Maximum és minimum elem kiválasztása egy tömbben

// a programban mindkettőt megkeressük

{

internal class MaxMinKivalasztas

{

static void Main(string[] args)

{

// tömb létrehozás - minden tételnél ua.

int[] Tomb = { 5, 2, 4, 5, 2, 6, 3, 4, 1, 5, 6, 3, 7, 8, 9, 2, 4, 8, 3, 1 };

// a tömb típus Length metódusát használjuk a tömb méretének meghatározására

int Hossz = Tomb.Length;

// az 'aktuális' legnagyobb/legkisebb elemet tartalmazó változók

// maximum értéknek 0-át, minimum értéknek 1000-et állítunk be

int Maximum = 0;

int Minimum = 1000;

// feldolgozzuk a tömb valamennyi elemét

for (int i = 0; i < Hossz; i++)

{

// a maximum értékhez szükséges vizsgálat

if (Tomb[i] > Maximum)

{

//igaz ág, az aktuális tömb elem nagyob mint a Maximum változó értéke tehát csere

Maximum = Tomb[i];

}

// a minimum értékhez szükséges vizsgálat

if (Tomb[i] < Minimum)

{

//igaz ág, az aktuális tömb elem kisebb mint a Minimum változó értéke tehát csere

Minimum = Tomb[i];

}

}

// kiírjuk az eredményt

Console.WriteLine("A tömb legnagyobb eleme: " + Maximum);

Console.WriteLine("A tömb legkisebb eleme: " + Minimum);

// várunk egy billentyű leütésre a kilépéshez

Console.ReadKey();

}

}

}

Kereses

namespace Kereses

// egy megadott elemet keresünk a tömbben - most ez legyen a 7-es

{

internal class Kereses

{

static void Main(string[] args)

{

// tömb létrehozás - minden tételnél ua.

int[] Tomb = { 5, 2, 4, 5, 2, 6, 3, 4, 1, 5, 6, 3, 7, 8, 9, 2, 4, 8, 3, 1 };

// a tömb típus Length metódusát használjuk a tömb méretének meghatározására

int Hossz = Tomb.Length;

// kell egy Keresett változó, hogy mit keresünk

int Keresett = 7;

// tároljuk el, hogy hol találtuk meg a keresettet

int Hely = 0;

// megvizsgáljuk a tömböt, hogy van-e benne a keresett érték

for (int i = 0; i < Hossz; i++)

{

// vizsgálat, hogy az aktuális elem azonos-e a keresettel

if (Tomb[i] == Keresett)

{

// igaz ág, az aktuális elem megegyezik a keresettel

// tároljuk el a helyét

Hely = i;

// kilépünk a ciklusból

i = Hossz + 10;

}

}

// nézzük meg hogy megtaláltuk-e, ha a Hely = 0, vagyis nem változott, akkor nincs találat

if (Hely == 0)

{

//a ciklusnak nem azért lett vége, mert megtaláltuk, hanem elfogytak az elemek

Console.WriteLine("Nincs a keresett elem a tömbben!");

Console.ReadKey();

}

else

{

//azért léptünk ki, mert megtaláltuk a keresettet

Console.WriteLine("Megvan a keresett elem!");

Console.WriteLine("A tömb "+ Hely +". poziciójában.");

Console.ReadKey();

}

}

}

}

FileRead

namespace FileRead

{

using System;

using System.IO;

internal class FileRead

{

static void Main(string[] args)

{ // ebben a változóban tároljuk el a beolvasott sort

string sor = "";

//a Nevsor változóba olvassuk be a szöveg file aktuális sorát

// Figyelem!! Ez az útvonal az én gépemen helyes, módosítani kell a saját gépnek megfelelően!!

StreamReader Nevsor = new StreamReader("D:\\MFE\\2025 tavasz\\Kurzusok\\Prog I\\Kódok\\FileRead\\Nevsor.txt");

// a beolvasott sort eltároljük a sor változóba

sor = Nevsor.ReadLine();

// a ciklus a szöveges file végéig fut

while (!Nevsor.EndOfStream)

{

// kiírjuk a sor változó tartalmát a konzolra

Console.WriteLine(sor);

// beolvassuk a következő sort

sor = Nevsor.ReadLine();

}

// mivel az utolsó sor beolvasása után kilép a ciklusból, ezért az utolsó sort is ki kell íratni

Console.WriteLine(sor);

// lezárjuk a szöveges file-t

Nevsor.Close();

// várunk egy billentyű leütésre ...

Console.ReadKey();

}

}

}

FileWrite

namespace FileWrite

{

internal class FileWrite

{

static void Main(string[] args)

{

// létrehozunk egy string tömböt, és egyszerű értékadaással négy elemet helyezünk el benne

// természetesen máshogy is létre hozható, tárolható a kiírandó szöveg

string[] sorok = { "Első sor", "Második sor", "Harmadik sor", "Negyedik sor" };

//az Allomany változót használjuk a szöveg file létrehozására, írására

//Figyelem! Ez az útvonal az én gépemen helyes, módosítani kell a saját gépnek megfelelően!!

using StreamWriter Allomany = new StreamWriter("d:\\MFE\\2025 tavasz\\Kurzusok\\Prog I\\Kódok\\FileWrite\\AtolGabor.txt");

{

// a ciklussal addig írjuk az elemeket a szöveges állományba, amíg el nem fogy a kiírandó szöveg

foreach (string line in sorok)

// az állomány változóba írjuk a string tömb aktuális sorát

Allomany.WriteLine(line);

}

}

}

}