//string str1;

//str1 = "hhhu";

//string str2=String.cop

//Console.WriteLine(Object.ReferenceEquals(str3, str4));

//double d= 1.01D;

//int[,] arr1 = new int[5, 2];

//int[][,] arr2 = new int[5][,];

//int[] arrf = {1, 2};

//int i = 0;

//for(; ; )

//{

// if (i < 5)

// {

// Console.Write(i);

// }

// else

// break;

// i++;

//}

//var arr = new int[/\*6\*/] { 1, 2, 3, 4, 5, 6 };

//try

//{

// Console.WriteLine(arr[12]);

//}

//catch (Exception ex)

//{

// Console.WriteLine(ex.Message);

//}

//finally

//{

// Console.WriteLine("Vagy sikerült vagy nem!");

//}

//SortedList sortedList = new SortedList()

//{

// {2, true },

// {1, "one" }

//};

//foreach (DictionaryEntry sL in sortedList)

//{

// Console.Write(sL.Value);

//}

//Person pr = new Person();

//Console.WriteLine(pr.Id);

//pr.Introduction("Alen", 10);

//pr.Orokol();

//var x;

//x = 100;

//Console.WriteLine(x);

//while (DoSomething()) ;

//Console.Write(sizeof(int));

//Console.WriteLine();

//int?[] arr = new int?[5];

//Console.WriteLine(arr[0]);

//var i = 10, j = 2, l=3;

//Console.WriteLine(Convert.ToInt32('A'));

//IList nums = null;

//Console.WriteLine(nums?[0]);

int i = 2;

DD(ref i);

Console.WriteLine(i + "\n");

Console.Write(++i);

Console.Write(i++);

Console.Write(i);

Console.WriteLine();

static void DD(ref int i)

{

i = 1000;

Console.WriteLine(i);

Console.WriteLine();

}

//Console.WriteLine();

//int? n = 20;

//Nullable<int> m = null;

//Console.WriteLine(n.GetValueOrDefault());

//Console.WriteLine(m.HasValue);

//int op = m ?? 6;

//Console.WriteLine(op);

//test();

//int s = 0b0001\_000;

//Console.WriteLine(s);

DataStore<string> store = new DataStore<string>();

DataStore<int> st2 = new DataStore<int>();

Func<int, int, int> Eredmeny = Osszead;

static int Osszead(int x, int y)

{

return x + y;

}

var összeg = Osszead(40, 15);

Eredmeny = Kulonbseg;

static int Kulonbseg(int a, int b) => a - b;

//{

// return a - b;

//}

var különbség = Kulonbseg(200, 90);

Console.WriteLine($"Func eredmények: {összeg}, {különbség}");

int boksz = 2;

Object o = boksz;

int zut = (int)o;

}

static bool DoSomething() /\*=> false;\*/ //ezzel akkor nincs ford. hiba, ha a {} rész nincsen

{

Console.Write("working");

return false;

}

static void test() { Console.Write("test()"); }

}

public class Person : Bolygok, IKur

{

public int Id;

int IKur.Kuras(int r) => r + 1;

//{ int z = 1 + r; return z; }

//public override void Szopatas(string b)

//{

// base.Szopatas(b);

//}

public override void Keringes(int b) => throw new NotImplementedException();

public void Introduction(string name, int age=0) => Console.WriteLine($"My name is {name}, I am {age} years old.");

//{

// Console.WriteLine($"My name is {name}, I am {age} years old.");

//}

public override void BaszodjMeg() => throw new NotImplementedException();

//{

// throw new NotImplementedException();

//}

public override bool Igen() => 2 > 1 ? true : false;

}

public interface IKur

{

int Kuras(int z);

void MitUgatsz() { } // a {} nélkül ;-vel a végén ezt implementálni kellene

}

public abstract class Bolygok : IOrokles

{

public abstract void Keringes(int a) ;

public void Sugarzas() { }

public virtual void Szopatas(string t) { }

public abstract void BaszodjMeg(); //cannot be private, de internal, protected, public, protected internal, private protected igen

public abstract bool Igen();

public void Orokol() { }

}

public interface IOrokles

{

public void Orokol();

}

public static class Picsa { }

//public class Proba : Picsa { } nem lehet!

class DataStore<T>

{

public T Data { get; set; }

}

}