Parameters Demo

Using different types of parameters

# Objectives

* 1. Demonstrate passing parameters by value vs reference.
  2. Demonstrate ref and out parameters.
  3. Demonstrate Named Arguments.
  4. Demonstrate Optional Arguments.
  5. Demonstrate using overloaded methods.

# Passing by value

* 1. Create a new console project. Call it “ParametersDemo”.
  2. Add the following method:

public static void Test(int aValue)

{

aValue = 111;

Console.WriteLine("In Test Value is {0}", aValue);

}

* 1. In main add:

//Pass by value

Console.WriteLine("Pass by value test:");

int testVal1 = 1;

Console.WriteLine("Original value: {0}", testVal1);

Test(testVal1);

Console.WriteLine("Returned value: {0}", testVal1);

* 1. Try it with Ctrl-F5
  2. What changed or did not change?

# Pass in a reference type

* 1. Add the following method:

public static void TestArray(int[] anArray)

{

anArray[0] = 111;

Console.WriteLine("In Test Value is {0}", anArray[0]);

}

* 1. In main add:

//Pass in a reference type

Console.WriteLine("\nPass in a reference type test:");

int [] testArray = { 1, 1, 1 };

Console.WriteLine("Original value: {0}", testArray[0]);

TestArray(testArray);

Console.WriteLine("Returned value: {0}", testArray[0]);

* 1. Try it with Ctrl-F5
  2. What changed or did not change?

# Out parameter test

* 1. Add the following method:

public static void TestOut(out int aValue)

{

aValue = 222;

Console.WriteLine("In TestOut Value is {0}", aValue);

}

* 1. In main add:

//Out parameter test

Console.WriteLine("\nOut parameter test:");

int testVal2 = 2;

Console.WriteLine("Original value: {0}", testVal2);

TestOut(out testVal2);

Console.WriteLine("Returned value: {0}", testVal2);

* 1. Try it with Ctrl-F5
  2. What changed or did not change?
  3. Change code to:

int testVal2;

//Console.WriteLine("Original Value: {0}", testValue2);

* 1. Try it with Ctrl-F5. Notice that testValue2 is initialized by the method. You can do this with an out parameter.

# Ref Parameter Test

* 1. Add the following method:

public static void TestRef(ref int aValue)

{

aValue = 333;

Console.WriteLine("In TestRef Value is {0}", aValue);

}

* 1. In main add:

//Ref parameter test

Console.WriteLine("\nRef parameter test:");

int testVal3 = 3;

Console.WriteLine("Original value: {0}", testVal3);

TestRef(ref testVal3);

Console.WriteLine("Returned value: {0}", testVal3);

* 1. Try it with Ctrl-F5. Notice that the method modifies the value in main.
  2. What changed or did not change?

# Optional parameters

* 1. Add the following method:

public static void TestOptional(int aValue = 444)

{

Console.WriteLine("In TestOptional Value is {0}", aValue);

}

* 1. In main add:

//Optional parameter passed by position

Console.WriteLine("\nCalling TestOptional with a parameter");

TestOptional(4);

* 1. Try it with Ctrl-F5. Notice that aValue takes the default.
  2. In Main Add:

//Optional parameter passed with no parameters

Console.WriteLine("\nCalling TestOptional with no parameter");

TestOptional();

* 1. Try it with Ctrl-F5. Notice that all values takes the default.

# Multiple and Named Parameters

* 1. Add the following method:

public static void TestMultiple(int aValue, int bValue = 222, int cValue = 333)

{

Console.WriteLine("Inside TestMultiple -"

+ "Values: "

+ "{0}, {1}, {2}", aValue, bValue, cValue);

}

* 1. In main add:

//Multiple parameters

Console.WriteLine("\nTestMultiple all three set");

TestMultiple(1, 2, 3);

* 1. Try it with Ctrl-F5. Notice all three values set normally.
  2. In main add:

//Multiple parameters, not all set

Console.WriteLine("\nTestMultiple set just first 2");

TestMultiple(1, 2);

* 1. Try it with Ctrl-F5. Notice that the last parameter derives its value from the default
  2. In main add:

//Multiple parameters with named parameter

Console.WriteLine("\nTestMultiple set just first 2");

TestMultiple(1, cValue: 3);

* 1. Try it with Ctrl-F5. Notice the second parameter gets its value from the default and the third is set as a named parameter.

# Overloaded methods

* 1. Add the following methods:

public static void TestOverloaded(string strParam)

{

Console.WriteLine("String overload called with a value of {0}", strParam);

}

public static void TestOverloaded(int intParam, double dblParam)

{

Console.WriteLine("Numeric overload called with values of {0} and {1}", intParam, dblParam);

}

* 1. In main add:

//Overload test string method

Console.WriteLine("\nTest overloaded method with string");

TestOverloaded("Test with string");

//Overload test numeric method

Console.WriteLine("\nTest overloaded method with numbers");

TestOverloaded(5,5.5);

* 1. Try it with Ctrl-F5. Notice that the correct method is called based on the types of arguments that are passed in.