任何查询的类型都必须为 [IEnumerable](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerable.aspx) 或 [IEnumerable<T>](https://msdn.microsoft.com/zh-cn/library/9eekhta0.aspx)，或一种派生类型（如 [IQueryable<T>](https://msdn.microsoft.com/zh-cn/library/bb351562.aspx)）。因此，返回查询的方法的任何返回值或 **out** 参数也必须具有该类型。如果某个方法将查询具体化为具体的 [List<T>](https://msdn.microsoft.com/zh-cn/library/6sh2ey19.aspx) 或 [Array](https://msdn.microsoft.com/zh-cn/library/system.array.aspx) 类型，则认为该方法在返回查询结果（而不是查询本身）。仍然能够编写或修改从方法返回的查询变量。

在下面的示例中，第一个方法以返回值的形式返回查询，第二个方法以 **out** 参数的形式返回查询。请注意，在这两种情况下，返回的都是查询，而不是查询结果。

class MQ

{

// QueryMethhod1 returns a query as its value.

IEnumerable<string> QueryMethod1(ref int[] ints)

{

var intsToStrings = from i in ints

where i > 4

select i.ToString();

return intsToStrings;

}

// QueryMethod2 returns a query as the value of parameter returnQ.

void QueryMethod2(ref int[] ints, out IEnumerable<string> returnQ)

{

var intsToStrings = from i in ints

where i < 4

select i.ToString();

returnQ = intsToStrings;

}

static void Main()

{

MQ app = new MQ();

int[] nums = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

// QueryMethod1 returns a query as the value of the method.

var myQuery1 = app.QueryMethod1(ref nums);

// Query myQuery1 is executed in the following foreach loop.

Console.WriteLine("Results of executing myQuery1:");

// Rest the mouse pointer over myQuery1 to see its type.

foreach (string s in myQuery1)

{

Console.WriteLine(s);

}

// You also can execute the query returned from QueryMethod1

// directly, without using myQuery1.

Console.WriteLine("\nResults of executing myQuery1 directly:");

// Rest the mouse pointer over the call to QueryMethod1 to see its

// return type.

foreach (string s in app.QueryMethod1(ref nums))

{

Console.WriteLine(s);

}

IEnumerable<string> myQuery2;

// QueryMethod2 returns a query as the value of its out parameter.

app.QueryMethod2(ref nums, out myQuery2);

// Execute the returned query.

Console.WriteLine("\nResults of executing myQuery2:");

foreach (string s in myQuery2)

{

Console.WriteLine(s);

}

// You can modify a query by using query composition. A saved query

// is nested inside a new query definition that revises the results

// of the first query.

myQuery1 = from item in myQuery1

orderby item descending

select item;

// Execute the modified query.

Console.WriteLine("\nResults of executing modified myQuery1:");

foreach (string s in myQuery1)

{

Console.WriteLine(s);

}

// Keep console window open in debug mode.

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

}

}