方法(函数)参数的传递

- **值参数**

这种方式复制参数的实际值给函数的形式参数，实参和形参使用的是两个不同内存中的值。在这种情况下，当形参的值发生改变时，不会影响实参的值，从而保证了实参数据的安全。

```
static void Func(int a) {
    a++;
    Console.WriteLine($”Func a ={ a}”);
}

static void Main() {
    int a = 3;
    Func(a);
    Console.WriteLine(“Main a ={ a}”);
}

结果:
Func a=4
Main a=3
```

- **引用参数**

这种方式复制参数的内存位置的引用给形式参数。这意味着，当形参的值发生改变时，同时也改变实参的值。

```
public void swap(ref int x, ref int y) {
    int temp;
    temp = x; /* 保存 x 的值 */
    x = y;    /* 把 y 赋值给 x */
    y = temp; /* 把 temp 赋值给 y */
}

static void Main(string[] args) {
    NumberManipulator n = new NumberManipulator();
    /* 局部变量定义 */
    int a = 100;
    int b = 200;

    Console.WriteLine("在交换之前，a 的值：  {0}", a);
    Console.WriteLine("在交换之前，b 的值：  {0}", b);

    /* 调用函数来交换值 */
    n.swap(ref a, ref b);

    Console.WriteLine("在交换之后，a 的值：  {0}", a);
    Console.WriteLine("在交换之后，b 的值：  {0}", b);
```

```
    Console.ReadLine();
}

结果：
在交换之前，a的值：100
在交换之前，b的值：200

在交换之后，a的值：200
在交换之后，b的值：100
```

- **输出参数**

return 语句可用于只从函数中返回一个值。但是，可以使用 输出参数 来从函数中返回两个值。输出参数会把方法输出的数据赋给自己，这种方式可以返回多个值。

```
public void getValue(out int x) {
    int temp = 5;
    x = temp;
}

static void Main(string[] args) {
    NumberManipulator n = new NumberManipulator();
    /* 局部变量定义 */
    int a = 100;

    Console.WriteLine("在方法调用之前，a 的值： {0}", a);

    /* 调用函数来获取值 */
    n.getValue(out a);

    Console.WriteLine("在方法调用之后，a 的值： {0}", a);
    Console.ReadLine();

}

结果：
在方法调用前，a的值：100

在方法调用后，a的值：5
```