# 方法的调用问题

* 在方法中定义的变量称为局部变量,其作用域从定义开始,到其所在的大括号结束为止.
* 在一个方法中想要访问另一个方法中的变量,怎么办?
* 两种解决方法:参数和返回值
* 我们在Main()函数中，调用Test()函数，我们管Main()函数称之为调用者，管Test()函数称之为被调用者。

## 如果被调用者想要得到调用者的值：

* 传递参数。
* 使用静态字段来模拟全局变量。

## 如果调用者想要得到被调用者的值：

* 返回值。（要有人接收）

方法中的return语句：

导致函数立即返回。在返回值为void的函数中return，在返回值非void的函数中return值

|  |
| --- |
| class Program  {  //字段 属于类的字段  public static int \_number = 10;  static void Main(string[] args)  {  //int b = 10;  int a = 3;  //Test(a); //传递参数 输出的值是3，因为没有返回值  //需要一个变量去接收返回的值  int res = Test(a);  Console.WriteLine(res);  //Console.WriteLine(\_number);  Console.ReadKey();    }  public static int Test(int a) //传递参数。  {  a = a + 5;  return a;  //Console.WriteLine(\_number);  }  public static void TestTwo()  {  //Console.WriteLine(\_number);  }  } |

示例：写一个方法,判断一个年份是否是润年，和最大值

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  #region 判断闰年  ////写一个方法,判断一个年份是否是润年。  //bool b = IsRun(2020);  //Console.WriteLine(b);  #endregion  #region 最大值  ////比较两个数字的大小，返回最大值  ////int a1 = 10;  ////int a2 = 20;  //int max = GetMax(10, 20);//实参  //Console.WriteLine(max);  //Console.ReadKey();  #endregion  }  /// <summary>  /// 判断一个年份是否是润年  /// </summary>  /// <param name="year">要判断的年份</param>  /// <returns>是否是润年</returns>  public static bool IsRun(int year)  {  bool b = (year % 400 == 0) || (year % 4 == 0 && year % 100 != 0);  return b;  }  /// <sum比较两个数字的大小，返回最大值mary>  ///  /// </summary>  /// <param name="n1">第一个数</param>  /// <param name="n2">第二个数</param>  /// <returns>返回的最大值</returns>  public static int GetMax(int n1,int n2)//形参  {  int max = n1 > n2 ? n1 : n2;  return max;  }  } |

**不管是实参还是形参，都是在内存中开辟了空间的。**

**方法的功能一定要单一。**

GetMax(int n1,int n2)

方法中最忌讳的就是出现提示用户输入的字眼。

练习：

## 读取输入的整数，定义成方法，多次调用(如果用户输入的是数字,则返回,否则提示用户重新输入)

## 还记得学循环时做的那道题吗?只允许用户输入y或n,请改成方法

## 计算输入数组的和：int Sum(int[] values)

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  #region 只返回输入的数字  //读取输入的整数，定义成方法，  //多次调用(如果用户输入的是数字,则返回,否则提示用户重新输入)  //NewMethod();  //Console.ReadKey();  Console.WriteLine("请输入一个数字");  string input = Console.ReadLine();  int number = GetNumber(input);  Console.WriteLine(number);  Console.ReadKey();  #endregion  #region 只返回yes或者no  //还记得学循环时做的那道题吗?只允许用户输入y或n,请改成方法  //这个方法做了什么？  //只能让用户输入yes或者no，只要不是，就重新输入（循环）  //要拿到输入的yes或者no  Console.WriteLine("请输入yes或者no");  string str = Console.ReadLine();  string result = IsYesOrNo(str);  Console.WriteLine(result);  Console.ReadKey();  #endregion  //计算输入数组的和：int Sum(int[] values)  int[] nums = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };  int sum = GetSum(nums);  Console.WriteLine(sum);  Console.ReadKey();  }  ////方法的功能一定要单一!!!!!这样写是错误的  private static void NewMethod()  {  while (true)  {  Console.WriteLine("请输入一个数字");  try  {  int number = Convert.ToInt32(Console.ReadLine());  Console.WriteLine(number);  break;  }  catch  {  Console.WriteLine("输入错误");  }  }  }  /// <summary>  /// 获取输入的数字  /// </summary>  /// <param name="s">任意输入的东西</param>  /// <returns>返回输入的数字</returns>  public static int GetNumber(string s)  {  while (true)  {  try  {  int number = Convert.ToInt32(s);  return number;  }  catch//这样不行 catch执行完成之后程序返回try 是死循环  {  Console.WriteLine("请重新输入");  s = Console.ReadLine();//把s重新赋值  }  }  }  /// <summary>  /// 只能让用户输入yes或者no,并且返回  /// </summary>  /// <param name="input">用户的输入</param>  /// <returns>返回yes或者no</returns>  public static string IsYesOrNo(string input)  {  while (true)  {  if (input == "yes" || input == "no")  {  return input;  }  else  {  Console.WriteLine("只能输入yes或者no，请重新输入");  input = Console.ReadLine();  }  }  }  /// <summary>  /// 整数类型数组的总和  /// </summary>  /// <param name="nums">要求的数组和</param>  /// <returns>返回这个数组的和</returns>  public static int GetSum(int[] nums)  {  int sum = 0;  for (int i = 0; i < nums.Length; i++)  {  sum += nums[i];  }  return sum;  }  } |

# out、ref、params

## out参数。

* 如果你在一个方法中，返回多个相同类型的值的时候，可以考虑返回一个数组。但是，如果返回多个不同类型的值的时候，返回数组就不行了，那么这个时候，我们可以考虑使用out参数。。
* out参数就侧重于在一个方法中可以返回多个不同类型的值。

示例：

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  #region 一般方法  //写一个方法 求一个数组中的最大值、最小值、总和、平均值  int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };  //将要返回的4个值，放到一个数组中返回  int[] res = GetMaxMinSumAvg(numbers);  Console.WriteLine("最大值是{0}，最小值是{1}，总和是{2}，平均值是{3}", res[0], res[1], res[2], res[3]);  Console.ReadKey();  #endregion  #region out方法  int max = 0;//可以不一样max1，可以不用赋初值因为下面返回了  int min = 0;  int sum = 0;  int avg = 0;  bool b;  string s;  double d;  Test(numbers, out max, out min, out sum, out avg,out b,out s,out d);  Console.WriteLine("最大值是{0}，最小值是{1}，总和是{2}，平均值是{3}", max, min, sum, avg);  Console.WriteLine(b);  Console.WriteLine(s);  Console.WriteLine(d);  Console.ReadKey();  #endregion  }  /// <summary>  /// 求一个数组中的最大值、最小值、总和、平均值  /// </summary>  /// <param name="nums">所求数组</param>  /// <returns>返回最大值、最小值、总和、平均值</returns>  public static int[] GetMaxMinSumAvg(int[] nums)  {  int[] res = new int[4];  //假设res[0] 最大值 res[1] 最小值 res[2] 总和 res[3]平均值  res[0] = nums[0];//amx  res[1] = nums[0];//min  res[2] = 0;  //但是当返回的不是同一类的数据时，这时需要out  string name = "孙权";  bool b = true;  for (int i = 0; i < nums.Length; i++)  {  //当前循环到的值比我假定的的最大值还要大 赋值下一个  if (nums[i] > res[0])  {  //把循环到的元素赋值给我的最大值  res[0] = nums[i];  }  if (nums[i] < res[1])  {  res[1] = nums[i];  }  res[2] += nums[i];  }  //平均值  res[3] = res[2] / nums.Length;  return res;  }  /// <summary>  /// 求一个数组中的最大值、最小值、总和、平均值  /// </summary>  /// <param name="nums">要求的数组</param>  /// <param name="max">多余返回的最大值</param>  /// <param name="min">多余返回的最小值</param>  /// <param name="sum">多余返回的总和</param>  /// <param name="avg">多余返回的平均值</param>  public static void Test(int[] nums,out int max,out int min,out int sum,out int avg,out bool b,out string s,out double d)  {  //out参数要求在方法的内部必须为其赋值  max = nums[0];  min = nums[1];  sum = 0;  for (int i = 0; i < nums.Length; i++)  {  if (nums[i] > max)  {  max = nums[i];  }  if (nums[i] < min)  {  min = nums[i];  }  sum += nums[i];  }  avg = sum / nums.Length;  b = true;  s = "小刚";  d = 3.14;  }  } |

2)、ref参数

能够将一个变量带入一个方法中进行改变，改变完成后，再讲改变后的值带出方法。

ref参数要求在方法外必须为其赋值，而方法内可以不赋值。

3)、params可变参数

将实参列表中跟可变参数数组类型一致的元素都当做数组的元素去处理。

params可变参数必须是形参列表中的最后一个元素。

5、方法的重载

概念：方法的重载指的就是方法的名称相同给，但是参数不同。

参数不同，分为两种情况

1)、如果参数的个数相同，那么参数的类型就不能相同。

2)、如果参数的类型相同，那么参数的个数就不能相同。

\*\*\*方法的重载跟返回值没有关系。

6、方法的递归

方法自己调用自己。

找出一个文件夹中所有的文件。