C#编码规范

# 目的

1，计算机现代在测绘科学中使用广泛,测绘相关软件的平差计算问题简单但计算量大,需要掌握一门编程语言设计相应的程序解决实际问题。好的编码约定可使源代码严谨、可读性强且意义清楚，与其它语言约定相一致，并且尽可能的直观。

统一编码规范的主要原因，是使应用程序的结构和编码风格标准化，以便于阅 读和理解这段代码

1. 学会用C#窗体程序开发环境建立应用程序框架方法
2. 学会使用常见的Windows控件
3. 根据自身需要设计一个测量程序界面

# 二．适用范围

（1），本规范主要以c#为开发平差计算的规范

（2），适用于一切有关程序撰写的工作事项。

# 三．命名空间

命名空间应是用解决方案的名称，每个项目设置一个二级命名空间，并以学号命名

# 四．命名规范

**命名基本规范**

任何标识符的名字都应该可以简单、清楚、正确的表示出该标识符的作用。

同时我们要将尽可能多的信息装入到标识符的名字当中去，以便读代码的人可以快速的读懂代码。

1. 我们在命名标识符时（包括参数，常量，变量），应使用单词的首字母大小写来区分一个标识符中的多个单词，如UserName.
2. PasalCasing：标识符的第一个单词的字母大写；
3. camelCasing：标识符的第一个单词的字母小写。
4. 除了参数、变量、常量外，所有命名空间名称、类、函数、接口、属性、事件、枚举等名称的命名都含一到多个单词，每一个单词第一个字母大写，其余字母均小写。例如：HelloWorld、SetName等。
5. 参数与变量包含一到多个单词，第一个单词首字母小写，其余单词首字母大写。例如：name、productId等。

**变量命名规范**

变量一般指局部变量、成员变量、全局变量以及方法中的参数，这些命名要具有

相对应的含义，一般使用需要的英文单词来表示，要满足Camel规则。

Camel规则, 名称中的第一个单词的首字母不能大写。

注意, 一般在不知道返回的是什么类型的时候，一般会用弱类型(var)去接。

不能名称中的单词全部大写。

不能使用数字来区分变量。不能使用缩写来代替变量名(除了URL\IO这样达成共识的，在循环中的)。不给变量添加任何的前缀(除布尔类型(is/can)及私有和受保护的变量(“\_”前缀))。

变量要声明在类的顶部，静态变量放在最前面。常量所有单词大写，多个单词之间使用“\_”连接。

2

只能使用数字、字母、下划线

标识符开头第一个字符必须是字母或下划线；

不能把C#关键字用作标识符

当关键字用作标识符：前加@标识符

1. 要用名词、名词短语或形容词来命名属性
2. 要用描述集合中具体内容的短语的复数形式来命名属性集合，而不要用短语的单数形式加”List“、”Array”或”Collection“后缀

### 使用英语语序命名标识符

人在阅读代码时，能更快的理解符合其阅读习惯的命名。

如VerticalAlignment比AlignmentVertical能让人更快的知道该变量的含意

### 尽量不要使用单词缩写，除非它是众所周知的

### 名字一定要能够表达出标识符的含意

### 类和接口命名

类的名字**要**用名词；

避免使用单词的缩写，除非它的缩写已经广为人知，如HTTP。

接口的名字**要**以字母I开头。保证对接口的标准实现名字只相差一个“I”前缀，例如对IComponent的标准实现为Component；

泛型类型参数的命名：命名**要**为T或者以T开头的描述性名字，例如：

public class List<T>

public class MyClass<TSession>

对同一项目的不同命名空间中的类，命名**避免**重复。避免引用时的冲突和混淆；

### 方法命名

第一个单词一般是动词

如果方法返回一个成员变量的值，方法名一般为Get+成员变量名，如若返回的值 是bool变量，一般以Is作为前缀。另外，如果必要，考虑用属性来替代方法

**组件名称要缩写**

# 五．注释规范

**1注释的基本约定**

1注释应该增加代码的清晰度；

保持注释的简洁，不是任何代码都需要注释的，过多的注释反而会影响代码的可读性。

2注释**不要**包括其他的特殊字符。

**3建议**先写注释，后写代码，注释和代码一起完成

4如果语句块（比如循环和条件分枝的代码块）代码太长，嵌套太多，则在其结束“｝”**要**加上注释，标志对应的开始语句。如果分支条件逻辑比较复杂，也**要**加上注释。

#### 注释类型

##### **块注释**

1主要用来描述文件，类，方法，算法等，放在所描述对象的前边。具体格式以IDE编辑器输入“///”自动生成的格式为准，另外再附加我们自定义的格式，如下所列：

/// <Remark>作者，创建日期，修改日期</ Remark >

对类和接口的注释必须加上上述标记，对方法可以视情况考虑

##### **行注释**

##### 1主要用在方法内部，对代码，变量，流程等进行说明。整个注释占据一行。

##### **尾随注释**

1与行注释功能相似，放在代码的同行，但是要与代码之间有足够的空间，便于分清。例：

int m = 4 ; /／ 注释

2如果一个程序块内有多个尾随注释，每个注释的缩进**要**保持一致。

**程序修改注释**

新增代码行的前后**要**有注释行说明，对具体格式不作要求，但必须包含作者，新增时间，新增目的。在新增代码的最后必须加上结束标志；

1删除代码行的前后**要**用注释行说明，删除代码用注释原有代码的方法。注释方法和内容同新增；删除的代码行**建议**用#region XXX #endregion 代码段折叠，保持代码文件干净整洁

2修改代码行**建议**以删除代码行后再新增代码行的方式进行（针对别人的代码进行修改时，必须标明，对于自己的代码进行修改时，酌情进行）。注释方法和内容同新增；

# 六.代码编写规范

#### Tab

使一个Tab为4个空格长。

#### 缩进

1**要**使一个代码块内的代码都统一缩进一个Tab长度。

#### 空行

1**建议**适当的增加空行，来增加代码的可读性。

2在在类，接口以及彼此之间**要**有两行空行：

3 在下列情况之间**要**有一行空行：

方法之间；

局部变量和它后边的语句之间；

方法内的功能逻辑部分之间；

#### 函数长度

´每个函数有效代码（不包括注释和空行）长度**不要**超过50行。

*{”，“}”*

2.开括号“{”**要**放在块的所有者的下一行，单起一行；

1.闭括号“}”**要**单独放在代码块的最后一行，单起一行。

#### 行宽

´ 每行代码和注释**不要**超过70个字符或屏幕的宽度，如超过则应换行，换行后的代码应该缩进一个Tab。

#### 空格

1．括号和它里面的字符之间**不要**出现空格。括号应该和它前边的关键词留有空格，如：while (true) {};

但是方法名和左括号之间**不要**有空格。

2．参数之间的逗号后**要**加一空格。如：method1(int i1, int i2)

3．for语句里的表达式之间**要**加一空格。如：for (expr1; expr2; expr3)

4．二元操作符和操作数之间**要**用空格隔开。如：i + c;

5. 强制类型转换时，在类型和变量之间**要**加一空格。如：(int) i ;

# 七。声明

Ö 每行**要**只有一个声明，如果是声明i,j,k之类的简单变量可以放在一行;

Ö 除了for循环外，声明**要**放在块的最开始部分。for循环中的变量声明可以放在for语句中。如：for(int i = 0; I < 10; i++) 。

´ **避免**块内部的变量与它外部的变量名相同。

# 八.表达式和语句

Ö 每行建议只有一条语句。

**Ö if-else,if-elseif语句**，任何情况下，都应该有“{”，“}”，格式如下：

if (condition)

{

statements;

}

else if (condition)

{

statements;

}

else

{

statements;

}

**Ö for语句格式如下：**

for (initialization; condition; update)

{

statements;

}

如果语句为空：

for (initialization; condition; update) ;

**Ö while语句格式如下**：

while (condition)

{

statements;

}

如果语句为空:

while (condition);

Ö do-while语句格式如下：

do

{

statements;

}

while (condition);

# 九.参考文档

1，《.NET设计规范》，本规范很多内容都参考了这本书，书中对规范背后的背景和原则做了深入讨论；

2, 《测量程序设计》，本规范大多内容都是从这学期所学中参考。

示例代码：

private void button1\_Click(object sender, EventArgs e) {

string[] sd = new string[dataGridView1.RowCount-5]; //新建一个数组存放观测角度的 原始值 double[] sdr = new double[sd.Length]; //新建一个数组存放观测角度的弧度值

double[] cr = new double[sd.Length ]; //新建一个数组存放计算的坐标方位角 double sum = 0;

cr[0] = dmstorad(Convert.ToString(dataGridView1.Rows[0].Cells[4].Value));

//获取第一个坐标方位角，并将其转换成弧度，放入cr[]数组第一个元素中

Double acd =

dmstorad(Convert.ToString(dataGridView1.Rows[dataGridView1.RowCount-6].Cells[4].Value));

//获取终边坐标方位角，并将其转换成弧度，放入放入acd中用于计算和检核

if (Math.Abs(fd \* 180 / Math.PI \* 3600) > fdx)

//检查角度闭合差是否满足要求 MessageBox.Show("角度闭合差超限！");

else { double vd = -fd / (sd.Length - 1); //分配角度闭合差（观测左角）

double sumvd = 0;

for (int i = 1; i < sdr.Length; i++)

{ sdr[i] += vd; //计算改正后的观测角度，并存入sdr数组中

sumvd += vd; dataGridView1.Rows[i].Cells[2].Value = Convert.ToString(Math.Round(vd \* 180 / Math.PI \* 3600, 2))+"″"; //将角度改正数存入表格中

dataGridView1.Rows[i].Cells[3].Value = radtodms(sdr[i]); }

if (Math.Round(sumvd, 8) != Math.Round(-fd , 8)) //秒保留2位对应弧度是8位 MessageBox.Show("角度改正数分配有误！");

else dataGridView1.Rows[dataGridView1.RowCount-4].Cells[2].Value=Convert.ToString(Math.Round(sumvd \* 180 / Math.PI \* 3600, 2)) + "″"; //将角度改正数总和存入表格中

sum = fangweijiao(sdr, cr); //推算改正后的坐标方位角

if (Math.Round(cr[cr.Length - 1], 8) != Math.Round(acd, 8)) MessageBox.Show("坐标方位角推算有误！");

else { dataGridView1.Rows[dataGridView1.RowCount - 4].Cells[3].Value = radtodms(sum);

//将改正后观测角度总和放入表格中 for (int i = 1; i < cr.Length-1; i++)//将改正后坐标方位角存入表格

dataGridView1.Rows[i].Cells[4].Value = radtodms(cr[i]);

}

}

//至此角度调整和计算完