C#编码规范

1、代码格式

1.1使所有的缩进为一个Tab，即4个空格

1.2、在代码中[垂直对齐](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=66636822&ss_c=ssc.citiao.link" \t "_blank)左号和右括号。

1.3、当一行被分为几行时，将串联运算符放在每一行的末尾而不是开头，清楚地表示没有后面的行是不完整的。

1.4、每一行上放置的语句避免超过一条

1.5、在大多数运算符[之前和之后](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=142255302&ss_c=ssc.citiao.link" \t "_blank)使用空格，这样做时不会改变代码的意图却可以使代码容易阅读。

3、命名规范

3.1、类变量命名:类变量命名一律用下划线开头，多个单词组成，第一个字母小写的，其他单词第一个字母大写。

3.2、类命名规范:

3.2.1、名字应该能够标识事物的特性；

3.2.2、首字母大写，缩略语可以全部字母大写；

3.2.3、名字可以有两个或三个单词组成，但通常不应多于三个；

3.2.5、使用名词或名词短语命名类；

3.2.6、少用缩写，除非是被广泛使用的；

3.2.7、不要使用下划线字符 (\_)。

3.3、局部变量命名：由小写字母开头，其他单词开头字母大写的各单词组成。

3.4、函数命名：一律由大写字母开头的各个单词组成。

3.5、类的实例命名规范：类的实例第一个字母小写，其他单词的第一个字母大写。

表B.1  数据类型简写规则

|  |  |  |  |
| --- | --- | --- | --- |
| 数 据 类 型 | 简    写 | 数 据 类 型 | 简    写 |
| 整型 | int | [长整型](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=10636914&ss_c=ssc.citiao.link) | lint |
| 字符串 | str | 单精度浮点型 | flt |
| [布尔型](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=54150574&ss_c=ssc.citiao.link) | bl | 双精度浮点型 | dbl |
| 短整型 | sint | 字节型 | bt |

B.1.7  控件命名规则

所有的控件名称都应该为自然名称的拼音简写，出现冲突可采用不同的简写规则。另外，在编码过程中涉及不到编码的控件，其名称可以取默认名称。控件命名规则如表B.3所示。

B.2.1  统一代码缩进

每个项目都应该根据实际情况建立统一的代码缩进标准，比如统一缩进两个字符，这样代码看起来非常美观，而且层次感很清楚，方便代码的阅读。

B.2.2  合理使用空行

在自动生成的命名空间和手动添加的命名空间之间要加空行分隔。

在每个类声明之后、每个方法声明之后都要加空行。

在一个方法体内，逻辑上密切相关的语句之间不加空行，其他地方应该加空行分隔。

B.2.3  代码换行

一行代码只做一件事情，如只定义一个变量，或只编写一条语句。语句独占一行，执行语句不能紧跟其后；而且，不论执行语句有多少都应该加{}。

代码行最大长度应该控制在70～80个字符以内，代码不要过长，否则不便于查看。

长表达式要在低优先级运算符处拆分成新行，运算符放在新行之首（以便突出运算符）。拆分的新行要适当缩进，以便使排版整齐。

B.2.4  空格的使用

关键字后面应该加空格，否则无法解析关键字。

if、for、while等控制语句关键字后面应该加空格，然后再跟左括号"("，以便突出关键字。

方法名后面不要加空格，而应该紧跟"("，以便与关键字进行区分。

"("向后紧跟，")"、","、";"向前紧跟，紧跟处不应该留空格。

[赋值运算符](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=55257294&ss_c=ssc.citiao.link)、[比较运算符](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=8011293&ss_c=ssc.citiao.link)、[算术运算符](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=56555579&ss_c=ssc.citiao.link)、[逻辑运算符](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=54086220&ss_c=ssc.citiao.link)和位运算符等二元运算符的前后应该加空格。

一元运算符如"!"、"++"、"--"、"&"等的前后不加空格。

类似"[]"、"."、"->"这类符号的前后不加空格。

对于表达式比较长的if语句和for语句，为了看上去更加紧凑，可以适当地去掉一些空格。

B.3  代码注释

B.3.1  注释的目的

文字说明代码的作用（即为什么要编写该代码，而不是如何编写）。

指出该代码的编写思路和逻辑方法。

代码中的重要转折点。

使代码的阅读者不必在他们的头脑中仿真运行代码的执行方法。

B.3.2  代码注释规范

方法体内使用类似"//"的形式进行注释，前面需要使用空格进行对齐时，使用Tab键替代空格键。避免对浅显易懂的语句进行注释说明。

逻辑性较强的地方加入注释，说明这段程序的逻辑是怎样的，以方便其他人理解，并且还可以在一定程度上排除Bug。

在编写代码前进行注释。

纯色字符注释行只用于主要注释。

增强注释的可读性。注释是供人阅读的，而不是让计算机阅读，所以应该使用完整的语句。

对注释进行缩进，使之与后随的语句对齐。注释通常位于它们要说明的代码的前面，为了从视觉上突出注释与代码之间的关系，可将注释缩进，使之与代码处于同一个层次上。

B.4.3  字段命名规范

字段一率采用英文单词或词组（可利用[翻译软件](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=8479185&ss_c=ssc.citiao.link" \t "_blank)）命名，如找不到专业的英文单词或词组，可以用相同意义的英文单词或词组代替。下面举例说明，如表B.6所示。

表B.6  字段命名

|  |  |
| --- | --- |
| 字 段 名 称 | 描    述 |
| name | 名字 |
| pwd | 密码 |

代码：

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

//角度转弧度

public double dmstorad(string s)

{

string[] ss = s.Split(new char[3] { '°', '′', '″' }, StringSplitOptions.RemoveEmptyEntries);

double[] d = new double[ss.Length];

for (int i = 0; i < d.Length; i++)

d[i] = Convert.ToDouble(ss[i]);

double sign = d[0] >= 0.0 ? 1.0 : -1.0;

double rad = 0;

if (d.Length == 1)

rad = Math.Abs(d[0]) \* Math.PI / 180;

else if (d.Length == 2)

rad = (Math.Abs(d[0]) + d[1] / 60) \* Math.PI / 180;

else

rad = (Math.Abs(d[0]) + d[1] / 60 + d[2] / 60 / 60) \* Math.PI / 180;

rad = sign \* rad;

return rad;

}

//弧度转角度

public string radtodms(double rad)

{

double sign = rad >= 0.0 ? 1.0 : -1.0;

rad = Math.Abs(rad) \* 180 / Math.PI;

double[] d = new double[3];

d[0] = (int)rad;

d[1] = (int)((rad - d[0]) \* 60);

d[2] = (rad - d[0] - d[1] / 60) \* 60 \* 60;

d[2] = Math.Round(d[2], 2);

if (d[2] == 60)

{

d[1] += 1;

d[2] -= 60;

if (d[1] == 60)

{

d[0] += 1;

d[1] -= 60;

}

}

d[0] = sign \* d[0];

string s = Convert.ToString(d[0]) + "°" + Convert.ToString(d[1]) + "′" + Convert.ToString(d[2]) + "″";

return s;

}

//坐标方位角计算

public double fangweijiao(double[] sdr, double[] cr)

{

double sum = 0;

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

{

cr[i] = cr[i - 1] + sdr[i] - Math.PI;

if (cr[i] >= Math.PI \* 2)

cr[i] -= Math.PI \* 2;

else if (cr[i] < 0.0)

cr[i] += Math.PI \* 2;

sum += sdr[i];

}

return sum;

}

private void excelToolStripMenuItem\_Click(object sender, EventArgs e)

{

dataGridView1.DataSource = null;

dataGridView1.Rows.Clear();

dataGridView1.Columns.Clear();

OpenFileDialog file = new OpenFileDialog();

file.Filter = "Excel文件|\*.xls|Excel文件|\*.xlsx";

if (file.ShowDialog() == DialogResult.OK)

{

string fname = file.FileName;

string strSource = "provider=Microsoft.ACE.OLEDB.12.0;" + "Data Source=" + fname + ";Extended Properties='Excel 8.0;HDR=Yes;IMEX=1'";

OleDbConnection conn = new OleDbConnection(strSource);

string sqlstring = "SELECT \* FROM [Sheet1$]";

OleDbDataAdapter adapter = new OleDbDataAdapter(sqlstring, conn);

DataSet da = new DataSet();

adapter.Fill(da);

dataGridView1.DataSource = da.Tables[0];

}

else

return;

}

private void Button1\_Click(object sender, EventArgs e)

{

{

string[] sd = new string[dataGridView1.RowCount - 5];

double[] sdr = new double[sd.Length];

double[] cr = new double[sd.Length];

double sum = 0;

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

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

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

{

sd[i] = Convert.ToString(dataGridView1.Rows[i].Cells[1].Value);

sdr[i] = dmstorad(sd[i]);

}

sum = fangweijiao(sdr, cr);

dataGridView1.Rows[dataGridView1.RowCount - 4].Cells[1].Value = radtodms(sum);

double fd, fdx;

fd = cr[cr.Length - 1] - acd;

fdx = 60 \* Math.Sqrt(sd.Length - 1);

dataGridView1.Rows[dataGridView1.RowCount - 3].Cells[1].Value = Convert.ToString(Math.Round(fd\*180/Math.PI\*3600,2))+"″"=dataGridView1.Rows[dataGridView1.RowCount - 2].Cells[1].Value =

= Convert.ToString(Math.Round(fdx, 2)) + "″";

=if (Math.Abs(fd \* 180 / Math.PI \* 3600) > fdx) MessageBox.Show("角度闭合差超限！");

else

{

double vd = -fd / (sd.Length - 1);

double sumvd = 0;

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

{

sdr[i] += vd;

sumvd += vd;

dataGridView1.Rows[i].Cells[2].Value = Convert.ToString(Math.Round(vd \* 180 / Math.PI \* 3600, 2)) + "″";

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

}

if (Math.Round(sumvd, 8) != Math.Round(-fd, 8))

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)) + "″";

sum = fangweijiao(sdr, cr);

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);

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

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

}

}

double[] jl = new double[sd.Length - 1];//新建一个数组存放距离

double[] x = new double[jl.Length];//新建一个数组存放Δx

double[] y = new double[jl.Length];//新建一个数组存放Δy

double[] zb = new double[sd.Length];//新建一个数组存放计算出来的坐标

double suml = 0, sumx = 0, sumy = 0;

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

{

jl[i] = Convert.ToDouble(dataGridView1.Rows[i].Cells[5].Value);//将距离放入表格中

suml += jl[i]; //计算距离总和

x[i] = jl[i] \* Math.Cos(cr[i]); //利用距离和坐标方位角计算坐标增量

y[i] = jl[i] \* Math.Sin(cr[i]);

sumx += x[i]; //计算坐标增量总和

sumy += y[i];

}

double[] X = new double[sd.Length - 1];

double[] Y = new double[sd.Length - 1];//新建一个数组存放坐标

double[] xx = new double[jl.Length];

double[] yy = new double[jl.Length];//新建一个数组存放改正数及其和

double sumxx = 0, sumyy = 0;

double[] gx = new double[jl.Length];//

double[] gy = new double[jl.Length];//新建一个数组存放改正后的坐标增量及其和

double sumgx = 0, sumgy = 0;

double X1, Y1, X2, Y2;//两个已知坐标

X1 = Convert.ToDouble(dataGridView1.Rows[1].Cells[12].Value);

Y1 = Convert.ToDouble(dataGridView1.Rows[1].Cells[13].Value);

X2 = Convert.ToDouble(dataGridView1.Rows[sd.Length - 1].Cells[12].Value);

Y2 = Convert.ToDouble(dataGridView1.Rows[sd.Length - 1].Cells[13].Value);

X[1] = X1;

Y[1] = Y1;

double fx, fy, fxy, k1;

fx = sumx - (X2 - X1);//计算坐标增量闭合差

fy = sumy - (Y2 - Y1);

fxy = Math.Sqrt(fx \* fx + fy \* fy);//计算导线全长闭合差

k1 = suml / fxy;//计算导线全长相对闭合差分母

if (k1 < 2000) //判断导线全长相对闭合差是否超限

MessageBox.Show("导线全长相对闭合差超限！");

else

{

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

{

xx[i] = -fx \* jl[i] / suml; //计算坐标增量改正数

yy[i] = -fy \* jl[i] / suml;

sumxx += xx[i]; //计算坐标增量改正数总和

sumyy += yy[i];

}

if (Math.Round(sumxx, 4) != Math.Round(-fx, 4) || Math.Round(sumyy, 4) != Math.Round(-fy, 4)) MessageBox.Show("坐标增量分配有误！");

else

{

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

{

gx[i] = x[i] + xx[i]; //计算改正后坐标增量

gy[i] = y[i] + yy[i];

sumgx += gx[i]; //计算改正后坐标增量总和

sumgy += gy[i];

}

if (Math.Round(sumgx, 4) != Math.Round(X2 - X1, 4) || Math.Round(sumgy, 4) != Math.Round(Y2 - Y1, 4)) MessageBox.Show("改正后的坐标增量计算有误!");

else

{

for (int i = 2; i < X.Length; i++)

{

X[i] = X[i - 1] + gx[i - 1];//计算x,y坐标

Y[i] = X[i - 1] + gy[i - 1];

}

if (Math.Round(X[X.Length - 1], 4) != Math.Round(X2, 4) || Math.Round(Y[Y.Length - 1], 4) != Math.Round(Y2, 4)) MessageBox.Show("坐标计算有误！");

else

{

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

{

dataGridView1.Rows[i].Cells[6].Value = Convert.ToString(Math.Round(x[i], 4)); //将坐标增量放入表格

dataGridView1.Rows[i].Cells[7].Value = Convert.ToString(Math.Round(y[i], 4));

dataGridView1.Rows[i].Cells[8].Value = Convert.ToString(Math.Round(xx[i], 4)); //将坐标增量改正数放入表格

dataGridView1.Rows[i].Cells[9].Value = Convert.ToString(Math.Round(yy[i], 4));

dataGridView1.Rows[i].Cells[10].Value = Convert.ToString(Math.Round(gx[i], 4));

dataGridView1.Rows[i].Cells[11].Value = Convert.ToString(Math.Round(gy[i], 4)); //将改正后坐标增量放入表格

dataGridView1.Rows[i].Cells[12].Value = Convert.ToString(Math.Round(X[i], 3));

dataGridView1.Rows[i].Cells[13].Value = Convert.ToString(Math.Round(Y[i], 3)); //将x,y坐标放入表格

}

dataGridView1.Rows[dataGridView1.RowCount - 4].Cells[5].Value = Convert.ToString(Math.Round(suml, 4));

dataGridView1.Rows[dataGridView1.RowCount - 4].Cells[6].Value = Convert.ToString(Math.Round(sumx, 4));

dataGridView1.Rows[dataGridView1.RowCount - 4].Cells[7].Value = Convert.ToString(Math.Round(sumy, 4)); //将距离总和、坐标增量总和放入表格中

dataGridView1.Rows[dataGridView1.RowCount - 4].Cells[8].Value = Convert.ToString(Math.Round(sumxx, 4));

dataGridView1.Rows[dataGridView1.RowCount - 4].Cells[9].Value = Convert.ToString(Math.Round(sumyy, 4)); //将坐标增量改正数总和、改正后坐标增量总和放入表格中

dataGridView1.Rows[dataGridView1.RowCount - 4].Cells[10].Value = Convert.ToString(Math.Round(sumgx, 4));

dataGridView1.Rows[dataGridView1.RowCount - 4].Cells[11].Value = Convert.ToString(Math.Round(sumgy, 4));

dataGridView1.Rows[dataGridView1.RowCount - 3].Cells[7].Value = Convert.ToString(Math.Round(fx, 4));

dataGridView1.Rows[dataGridView1.RowCount - 2].Cells[7].Value = Convert.ToString(Math.Round(fy, 4));

dataGridView1.Rows[dataGridView1.RowCount - 3].Cells[10].Value = Convert.ToString(Math.Round(fxy, 4));

dataGridView1.Rows[dataGridView1.RowCount - 2].Cells[11].Value = Convert.ToString((int)k1); //导线全长相对闭合差分母取整

}

}

}

}