C#编码规范

1.规范目的

1. 方便代码的交流和维护，便于日后再次阅读；

2. 使代码更美观、阅读更方便；

3. 使代码的逻辑更清晰、更易于理解。

2. 适用范围

 本规范适用于C#源代码，为详细设计，代码编写和代码审核提供参考和依据。

3.文本

描述必须遵循的规范；

异常类要以 “Exception”后缀；

描述在一般情况下应该遵循的规范，但如果完全理解规范背后的道理，并有很好的理由不遵循它时，也不畏惧打破常规。

强制类型转换时，在类型和变量之间建议加一空格。

描述一些几乎绝对绝不应该违反的规范。

每个函数有效代码（不包括注释和空行）长度不要超过50行。

与建议相对，一般情况下应该遵循，但有很好的理由时也可以打破。

块内部的变量与它外部的变量名相同。

对一些规范内容一并提供了示例代码。

4.代码组织与风格

4.1 Tab

一个Tab为4个空格长。

4.2 缩进

一个代码块内的代码都统一缩进一个Tab长度。

4.3 空行

1.适当的增加空行，来增加代码的可读性。

2.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：

（1.方法之间；

（2.局部变量和它后边的语句之间；

（3.方法内的功能逻辑部分之间；

4.4 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。

4.5 {”，“}”

1.开括号“{”要放在块的所有者的下一行，单起一行；

2.闭括号“}”要单独放在代码块的最后一行，单起一行。

4.6 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。

4.7 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，

2.但是方法名和左括号之间不要有空格。

3.参数之间的逗号后要加一空格。

4.for语句里的表达式之间要加一空格。

5.二元操作符和操作数之间要用空格隔开。

6.强制类型转换时，在类型和变量之间要加一空格。

5. 注释

5.1 注释的基本约定

1.注释应该增加代码的清晰度；

2.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。

3. 注释不要包括其他的特殊字符。

4.先写注释，后写代码，注释和代码一起完成

5.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。

6.在VS2005环境中通过配置工程编译时输出XML文档文件可以检查注释的完整情况，如果注释不完整会报告编译警告；

5.2 注释类型

5.2.1 块注释

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。

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

5.2.2 行注释

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。

5.2.3 尾随注释

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。

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。

5.4 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。

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。

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。

6. 命名

要使用可以准确说明变量/字段/类的完整的英文描述符，如firstName。对一些作用显而易见的变量可以采用简单的命名，如在循环里的递增（减）变量就可以被命名为 ” i ”。

要尽量采用项目所涉及领域的术语。

要采用大小写混合，提高名字的可读性。为区分一个标识符中的多个单词，把标识符中的每个单词的首字母大写。不采用下划线作分隔字符的写法。有两种适合的书写方法，适应于不同类型的标识符：

PasalCasing：标识符的第一个单词的字母大写；

camelCasing：标识符的第一个单词的字母小写。

少使用缩写，如果一定要使用，就谨慎使用。同时，应该保留一个标准缩写的列表，并且在使用时保持一致；

对常见缩略词，两个字母的缩写要采用统一大小写的方式；多字母缩写采用首字母大写，其他字母小写的方式；

少使用长名字（最好不超过 15 个字母）；

少使用相似或者仅在大小写上有区别的名字；

不要用\_或&作为第一个字母；

尽量要使用短而且具有意义的单词；

如果变量是集合，则变量名要用复数；

避免使用单词的缩写，除非它的缩写已经广为人知；

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。

7. 声明

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;

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。

块内部的变量与它外部的变量名不要相同。

8．

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;

namespace \_20170624唐亮4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void excel文件ToolStripMenuItem\_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 menuStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

//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;

}

//坐标方位角推算

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 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]);

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

}