C#语言编码规范

1. 基本命名格式

Pascal

每个单词首字母大写

camel

第一个单词首字母小写，其余单词首字母大写

Hungarian

类型缩写+Pascal，其中类型缩写一般使用类型中关键辅音字母

\_camel

下划线 加 camel格式

ACL

所有字母大写，单词用下划线连接

其他规则

出字段外 尽量不使用下划线，如果需要使用，下划线应该表示语义上的分级

1. 一般规范

命名规则

一个好的名字应该要表达它的意义

变量，参数

名词短语

格式：camel

一般情况下应该取有意义的名字，但是计数变量当用在琐碎的技术循环式更适宜叫

I ，j，k，l，m，n

变量

名词，名词短语

格式：\_camel或m\_camel。如：private int \_itemCount; private int m\_itemCount;

这样的格式，可以在方法中把字段和参数、变量等区分开。

GUI控件字段的格式：Hungarian。

GUI控件字段不同于一般的字段，这些控件字段中控件类型是重要的信息，所以这里使用Hungarian格式。

除了常量及静态只读字段外，其它字段只能使用private、internal限定符。 必须使用限定符。

属性

名词、名词短语。

格式：Pascal。

必须使用限定符。

一般情况下，应该取有意义的名字。偶尔可以考虑使用与其类型相同的名字命名一个属性。如：

public TreeView TreeView

{

get

{

...

}

set

{

...

}

}

除非是十分简单的类或者纯数据类型（贫血数据模型），否则不要使用属性的缩写格式（public TreeView TreeView{get;set;}）。

这样可以在复杂类中只有字段才能表示数据，所以可以很清楚的看到当前有哪些数据构成该类。

方法（函数）

动词、动词短语。

格式：Pascal。

必须使用限定符。

常用的介词有：Of、In、By、DependOn等。如：

User GetByUserName(string username);

2.1.6 事件

考虑用一个动词命名事件。用现在和过去时态命名有前缀和复制概念的事件名字。

格式：Pascal。

必须使用限定符。

多使用EventHandler<T>进行定义。

如：public event EventHandler<MouseMoveEventArgs> MouseMove;

类型

一般情况下，类型都使用Pascal格式来进行命名。

必须使用限定符。

接口

使用I字母打头，然后接Pascal形式，即格式为：IPascal。如ICollection。

枚举

格式：Pascal。

对于位枚举(Flags)用复数名字。

异常类

格式：Pascal + Exception

自定义异常类以Exception结尾，并且在类名中能清楚的描述出该异常的原因。比如FileNotFoundException，描述出了某个实体（文件、内存区域等）无法被找到。

事件委托

格式：Pascal + EventHandler

如：比如public delegate void MouseEventHandler (object sender, MouseEventArgs e);    //用于处理与鼠标相关的事件或委托。

对于自定义的委托，其参数第一个建议仍然使用object sender，sender代表触发这个时间或委托的源对象。而第二个参数继承于EventArgs类，并且在派生类中实现自己的业务逻辑。

代码格式及逻辑

声明

类成员声明

所有类成员都应该明确写上限定符：private，protected，public，internal。

每行的声明数

每行只有一个声明，因为它可以方便注释。

int level; // indentation level

int size; // size of table

当声明变量时，不要把多个变量或不同类型的变量放在同一行，例如：

int a, b; //What is 'a'? What does 'b' stand for?

初始化

局部变量一旦被声明就要初始化。例如：

string name = myObject.Name;

或

int val = time.Hours;

方法声明

如果可以的话，方法的参数尽量不要使用out及ref。

语句

大括号的格式：大括号内表示一个语句块(block)，前大括号和后大括号可以只占一行。

每行都应该只包含一条语句。

If语句

可以使用单行if。

其余情况应该使用带括号的if语句。

如，使用：

If(argument <= 0) throw new ArgumentException(“argument必须是正整数！”);

或

If(value)

{

//do something

}

而不是

If(value)

This.Calculate();

推荐不要把逻辑上不是一类的判断组合在一起。

条件尽量简单，不宜过长。

如果条件过长，可采用两种方式：

1.可以先把条件的值计算出，并存入一个临时的变量。

2.可对条件进行折行，折行的方式见：2.4.5.2换行。

2.2.2.2 For语句

带上两个括号，即采用以下格式：

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

{

//do sth

}

λ 当判定条件比较耗时时，应该采用缓存变量的形式，即以下格式：

for( int i = 0, c = list.Count; i < c; i++)

{

//do sth

}

While语句

采用以下格式：

while(value)

{

}

或

do

{

}

while(value)

Switch语句

switch应该处理所有的可能分支。如果不处理时，应该抛出异常。如：

switch(value)//value 是string类型

{

case “1”:

//case 1;

break;

case “2”:

//case 2;

break;

default:

   throw new NotSupportedException();

}

Try语句

格式如下：

try

{

}

catch(NotSupportedException ex)//子异常

{

}

catch(Exception e)//基异常

{

}

finally//finally语句可以省略

{

}

小技巧

以上所有语句，在VS环境下，都已经默认携带了snippet方便快捷输入。如在书写if语句时，可输入“if”，然后按两下tab键即可。

另外，它的用处在于它可以进行自定义你经常输入的代码段。具体内容见：http://www.cnblogs.com/WilsonWu/archive/2010/02/25/1673745.html

属性

不要连续地对同一属性进行调用。

如：

var prop2 = this.Prop1.Prop2;

var prop3 = this.Prop1.Prop3;

而是应该用变量进行缓存，如：

var prop1 = this.Prop1;

var prop2 = prop1.Prop2;

var prop3 = prop1.Prop3;

这是因为属性其实就是方法。所以就算是最简单的属性，对它进行多次调用都会浪费一定的资源。

属性的内部代码，不要设计得太过复杂。方便调试。

最好不要访问易中断的外部资源。因为很可能会导致无法对其调试。

事件

监听的事件处理方法取名按照“一定规则”，如btnSubmit\_Click。监听的事件处理方法里面，不应该写比较复杂的代码。如果比较复杂，应该提取函数。

方法

原则上单个方法的长度不能太长，如不超过50行、不超过一屏。

程序编码力求简洁，结构清晰，避免太多的分支结构及太过于技巧性的程序。

程序代码要严格，要把所有情况都考虑到。抛出的异常，应该都是在意料范围内的。如，以下代码是不严格的：

int[] array = this.GetIntArray();//这个方法可能会返回一个空数组。

int first = array[0];//由于数组可能没有元素，所以这里可能会抛出异常。

好的代码应该是：

if(array.Length > 0)

{

    int first = array[0];

}

或断言：

if(array.Length == 0) throw new InvalidException(“返回的数组没有元素。”)

如果是未完成的方法，必须抛出NotImplementException。

参数的检测，使用正常的代码进行检测。而调试期代码则使用Debug.Assert方法。

方法逻辑：

按照以下顺序编写：

1.参数检查。（private方法在参数传入时已经肯定参数正确时，可以忽略此步。）

2.参数转换。（private方法在参数传入时已经肯定参数正确时，可以忽略此步。）

3.核心逻辑。

4.返回值转换。

5.返回。

注释

注释尽量使用中文。

添加注释的目的：使代码易读、易写、易维护 。

如何添加注释：

    a. 代码、数据、算法的解释

    b. 做标记（时间、所做的改动等）

    c. 标识代码的功能和目的

    d. 代码如何调用

避免

    a. 对显而易见的内容进行注释

    b. 添加大段注释

    c. 注释的拼写错误

文件头

为了清晰的说明代码功能，跟踪版本纪录，应该在源代码文件中加入相应的描述信息，本文将列举相应的文件头格式。

需要说明的是，文件头中的历史纪录位置应该详细记录每次所作修改的内容。每次修改版本号都应该作相应的变更。

如对版本号无特殊要求，请遵照如下通用原则：

版本号形如 X.Y，其中X为大版本号，Y为小版本号。

bug修正，变更小版本号。

新增功能或重大bug修正，变更大版本号

2，实例代码

Application.Exit();

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;

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;

}

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;

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]);

}

}