程序开发之

代 码 规 范

智能制造与信息工程研究所

2017年5月4日

目录

一、引例

- 先上一段代码

```
response.write "产品小图片已经成功上传！"
response.end
else%>
<script>
var img=null;
function sc(){
//判断浏览器及版本
var ua=navigator.userAgent;
var ie=false;
if(navigator.appName=="Microsoft Internet Explorer"){
ie=true;}
if(!ie){
form.file.outerHTML=form.file.outerHTML.replace(/value=\w/g,'');
alert("建议你使用IE6。0以上的浏览器,在非IE浏览器中上传功能不能正常使用。");
return;}
var IEversion=parseFloat(ua.substring(ua.indexOf("MSIE ")+5,ua.indexOf(";",ua.indexOf("MSIE ")+5)));
if(IEversion< 6.0){
form.file.outerHTML=form.file.outerHTML.replace(/value=\w/g,'');
alert("系统检测到你的浏览器的版本比较低,建议你使用IE6。0以上的浏览器,否则上传功能不用。\n你可以http://www.microsoft.com/china/免费获得IE的最新版本!");
return;}
//判断是否图片  inp为文本框id
```

一、引例

还有例子

- **名不副实的函数/变量/类**
  - EditorGUI：创建新对象
  - EditorObjectCreatorGUI：通过处理不同的对象进行导航
- **混乱的类**
  - 假设由于某种原因，某个GUI类需要分析什么样的纹理可行（可能是有按钮要用来选择纹理）。如果这个GUI类是 唯一需要这个分析结果的类，那么在GUI类中这样做是有意义的。然而，由于某种原因，一个完全无关的gameplay类也需要这些信息。所以你需要将这些 纹理查询的信息从GUI类传给gameplay类。这时候，其实这个GUI类已经变大了：因为它里面其实还包括了TextureAnalyser类。
- **过于庞大的类**
- **并行逻辑和代码重复**
- **代码注释**

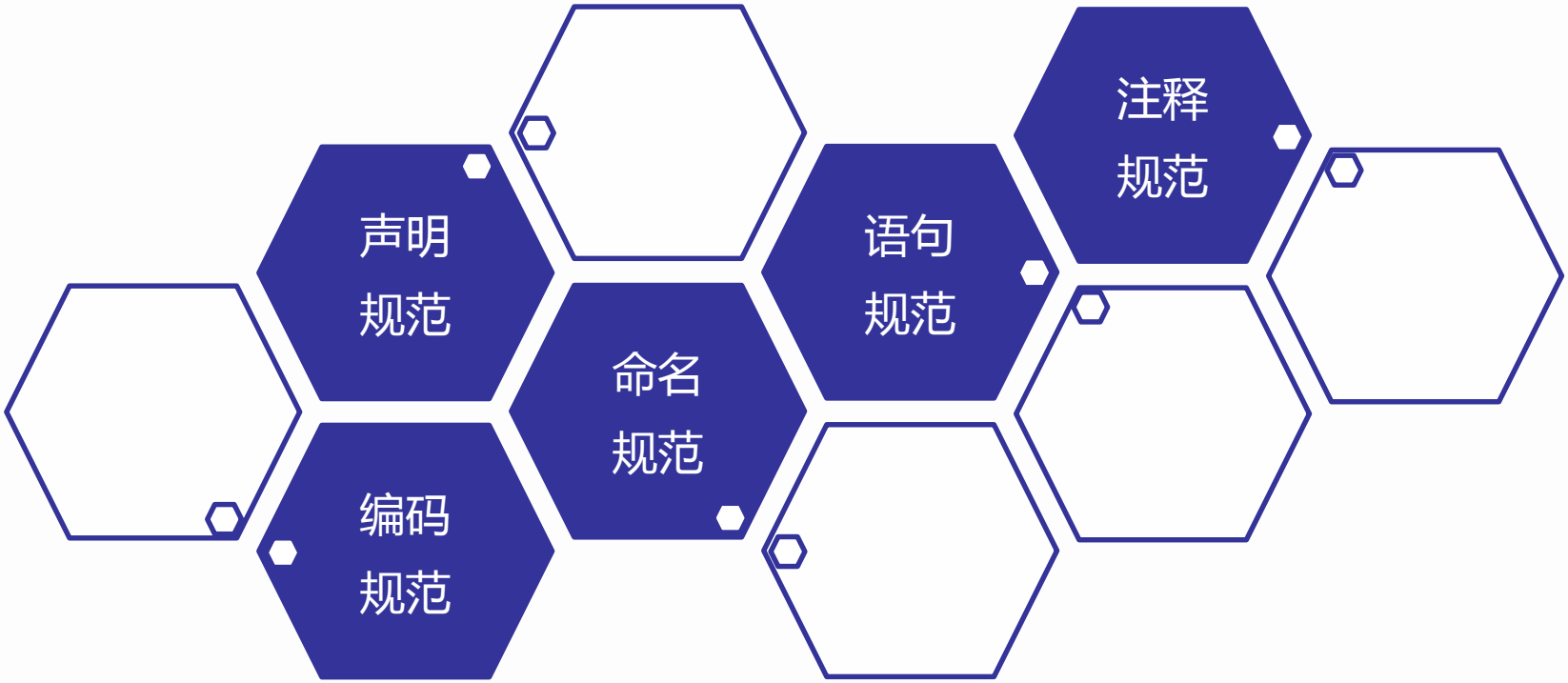
一、引例

怎么样才能成为一名优秀的程序员？

代码应该清晰易懂易于维护

- 尽可能地写出最简洁的代码；
- 如果代码后期会因为改动而变得凌乱不堪就得重构；
- 尽量删除没用的代码，并添加注释
- 不能一旦代码“起效了”就立马置之脑后
- 将功能代码改进为简洁代码可能在短期内是看不到回报的

一、引例



二、编码规范

1、缩进排版

- 4个空格为缩进排版的一个单位

```
#include<stdio.h>
#include<memory.h>
#define INF 100000000
int map[500][500];
int visit[500];
int prim(int num)
{
    int sum=0;
    memset(visit,0,sizeof(visit));
    visit[0]=1;
    for(int k=0;k<num-1;k++)
    {
        int min=INF,minnum=0;
        for(int n=0;n<num;n++)
            if(visit[n])
                for(int m=0;m<num;m++)
                    if(!visit[m]&&min>map[n][m])
                        minnum=m,min=map[n][m];
        visit[minnum]=1;
        sum=sum>min?sum:min;
    }
    return sum;
}
int main()
{
    int sum;
```

二、编码规范

2、行长度

- 尽量避免一行的长度超过80个字符

3、断行规则

- 当一个表达式无法容纳在一行内时，可以根据如下一般规则断开：
  - 在一个逗号“，”后面断开；
  - 在一个操作符前面断开；
  - 宁可选择较高级别（higher-level）的断开，而非较低级别（lower-level）的断开（见下面的例子）；
  - 新的一行应该在上一行同一级别表达式的开头处缩进4个空格；
  - 如果以上规则导致代码混乱或者使代码都堆挤在右边，则代之以缩进8个空格。

二、编码规范

3、断行规则

- 以下是两个断开算术表达式的例子。前者属于更高级别的断开，因为断开处位于括号表达式的外边。

```
longName1 = longName2 * (longName3 + longName4 – longName5)
+ 4 * longName6;
longName1 = longName2 * (longName3 + longName4
 – longName5) + 4 * longName6;
```

- 以下是两个缩进方法声明的例子。前者是常规情形，后者若使用常规的缩进方式将会使第二行和第三行移得很靠右，所以代之以缩进8个空格。

```
//规范的缩进
someMethod ( int anArg, Object anotherArg, string yetAnotherArg,
                    Object andStillAnother )
//以8个空格来缩进，以避免非常纵深的缩进。
private static synchronized horkingLongMethodName ( int anArg,
        Object anotherArg, string yetAnotherArg,
        Object andStillAnother )
```

二、编码规范

4、空行

- 空行可将逻辑相关的代码段分隔开，以提高可读性。
- 下列情况应该总是使用一个空行：
  - 一个源文件的两个片段（section）之间；
  - 类声明和接口声明之间；
  - 两个方法之间；
  - 方法内的局部变量和方法的第一条语句之间；
  - 块注释或单行注释之前；
  - 一个方法内的两个逻辑段之间，用以提高可读性

三、命名规范

1、类（Classes）

- 类名是一个名词，采用大小写混合的方式，每个单词的首字母大写。如：UserInfo

2、接口（Interfaces）

- 接口名是一个名词，大小写规则与类名相似。在接口名前加I来标识。如：IUserInfo

3、方法（Methods）

- 方法名是一个动词，大小写规则与类名相似。如：GetUserByLoginID（int LoginID）

三、命名规范

4、属性（Properties）

- 属性是类里面的成员，其名称是一个名词，大小写规则与类名相似。如：UserInfo.LoginID

5、常量（Constants）

- 常量名是一个名词，应全部大写，单词间用下划线隔开。如：const int TASK_ID = 1

三、命名规范

6、变量（Variables）

- 变量名是一个名词，采用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写。
- 变量名不应以下划线“_”或美元符号“$”开头，尽管这在语法上是允许的。
- 变量名应简短且富于描述。变量名的选用应易于记忆，即能够指出其用途。
- 尽量避免单个字符的变量名，除非是一次性的临时变量。
- 整型临时变量通常取名为i，j，k，m和n；字符型临时变量通常取名为c，d，e。如string userName

四、声明规范

1、每行声明变量的数量

- 推荐一行声明一个变量，因为这样有利于注释。如：

```
string loginName;          //用户登录名
string userName;           //用户姓名
```

要优于：

```
string loginName, userName;
```

2、初始化

- 尽量在声明局部变量的同时初始化，除非变量的初始值依赖于某些先前发生的操作

四、声明规范

3、布局

- 只在代码块的开始处声明变量（一个块是指任何被包含在大括号“{”和“}”之间的代码）。不要在首次用到该变量时才声明之。如：

```
Void MyMethod( )
{
        int int1 = 0;                    //方法（代码块）开始处声明变量

        if(condition)
        {
                int int2 = 0;            //if代码块开始处声明变量
                ……
        }
}
```

- 该规则的一个例外是for循环的索引变量：

```
for (int I = 0; I < maxLoops; i++) { …… }
```

四、声明规范

4、类和接口的声明

- 在编写类与接口时，遵循以下格式规范：
  - 在方法名与其参数列表之前的左括号“(”间不要有空格；
  - 左右大括号“{”和“}”均另起一行，与相应的声明语句对齐；
  - 方法与方法之间以空行分隔。
  - 如：

```
public class UserInfo()
{
        int userID;
        string userName;

        public string GetUserName()
        {
                return userName;
        }
}
```

五、语句与注释规范

1、简单语句

- 每行至多包含一条语句，例如：

```
argv++;                    //推荐
argv--;                    //推荐
argv++; argv--;            //避免
```

2、复合语句

- 复合语句是包含在大括号中，形如“{ 语句 }”的语句序列。遵循原则如下：
  - 被括其中的语句应该较左大括号“{”前的语句缩进一个层次；
  - 左右大括号“{”和“}”均另起一行，与相应的声明语句对齐；
  - 大括号可以适用于所有语句，包括单个语句，只要这些语句是诸如if - else或for控制结构的一部分。

五、语句与注释规范

3、行末注释

- 行末注释与它们所要描述的代码位于同一行，使用注释符“//”标记。如：

```
argv++;                    //这是行末注释
```

- 当注释太长，导致一行字符数超过80个时，可将注释放于所要描述的代码的上一行。如：

```
//这是过长时的行注释
argv++;
```

五、语句与注释规范

4、块注释

- 块注释通常用于提供对文件、方法、数据结构和算法的描述，使用注释符“/*注释内容*/”标记。
- 块注释被置于类和文件的开始处及每个方法之前。它们也可以用于其它地方，如方法内部。在功能和方法内部的块注释应该和它们所描述的代码具有一样的缩进格式。
- 块注释的前后都应留有空行，用以分隔块注释和代码。在文件开始处的块注释，注释上方的空行可以取消。如：

```
/*
*这是块注释
*/

public class Example()
{
        ……
}
```

六、一些例子

```
namespace ConsoleApp {
    /// <summary>
    /// 产品售光时被调用的委托
    /// </summary>
    public delegate void SalesOutEventHandler();

    /// <summary>
    /// 产品类，描述产品的基本信息
    /// </summary>
    public class Product {

        /// <summary>
        /// 定义产品被售光时的处理逻辑
        /// </summary>
        public event SalesOutEventHandler OnSalesOut;

        /// <summary>
        /// 根据产品Id查找产品
        /// </summary>
        /// <param name="id">产品的Id</param>
        /// <returns>符合此Id的产品实例，当不存在该产品时，返回null</returns>
        public Product GetProductById(int id) {
            return new Product();
        }

        /// <summary>
        /// 产品类型，描述产品种类，参考《需求说明》
        /// </summary>
        public enum ProductType {

        }
    }
}
```

六、一些例子

```
namespace ConsoleApp {

    public delegate void SalesOutEventHandler();

    public class Product {

        public event SalesOutEventHandler OnSalesOut;

        public Product GetProductById(int id) {
            return null;
        }

       private enum ProductType { }
    }
}
```

Pascal风格（单词首字母大写）

```
public class Product {
    public float Price { get; set; }

    public float GetProductPrice(int productId) {
        Product item = this.GetProductById(productId);
        float price = item.Price;

        if (price < 100) {
            return price;
        } else {
            return price * 0.95f;
        }
    }

    public Product GetProductById(int id) {
        return new Product();
    }
}
```

Camel风格（首字母小写，其后每个单词的首字母大写）

六、一些例子

常见集合类型后缀命名

说明	后缀	示例
数组	Array	int[] productArray
列表	List	List<Product> productList
DataTable/HashTable	Table	HashTable productTable
字典	Dictionary	Dictionay<string,string> productDictionary
EF中的DbSet /DataSet	Set	DbSet<Product> productSet

六、一些例子

常见后缀命名——集合

说明	后缀	示例	示例说明
费用相关	Cost	ShipCost	运输费
价格相关	Price	ProductUnitPrice	产品单价
消息相关	Message（弃用Note）	SuccessMessage	成功消息
日期相关	Date（弃用Time）	OrderDate	下单日期
计数、数量相关	Count（弃用Time）	LoginCount	登录次数
链接地址相关	Url	BlogUrl	博客链接
图片相关	Image	SignImage	签名图片
金额相关	Amount	PrepaidAmount	预付款
点数、积分相关	Point	MemberPoint	会员积分
记录、日志相关	Record（弃用Log）	ErrorRecord	错误记录
配置相关	Config	DataBaseConfig	数据库配置
状态相关	Status	OrderStatus	订单状态

说明	后缀	示例	示例说明
模式、方式相关	Mode	OpenMode	打开方式
种类相关	Category / Type 二选一	UserCategory	用户种类
工厂类相关	Factory	ConnectionFactory	连接工厂
启用相关	Enabled	ExportEnabled	开启导出
流相关	Stream	UploadStream	上传流
读取器相关	Reader	ExcelReader	Excel读取器
写入器相关	Writer	ExcelWriter	Excel写入器
适配器相关	Adapter	IntroOPAdapter	IntroOP适配器
提供器相关	Provider	MemebershipProvider	会员信息提供器
包装器相关	Wrapper	ProductWrapper	Product包装器
连接相关	Connection	ExcelConnection	Excel连接

六、一些例子

常见后缀命名——局部变量、方法参数、字段、属性

说明	后缀	示例	示例说明
费用相关	Cost	ShipCost	运输费
价格相关	Price	ProductUnitPrice	产品单价
消息相关	Message（弃用Note）	SuccessMessage	成功消息
日期相关	Date（弃用Time）	OrderDate	下单日期
计数、数量相关	Count（弃用Time）	LoginCount	登录次数
链接地址相关	Url	BlogUrl	博客链接
图片相关	Image	SignImage	签名图片
金额相关	Amount	PrepaidAmount	预付款
点数、积分相关	Point	MemberPoint	会员积分
记录、日志相关	Record（弃用Log）	ErrorRecord	错误记录
配置相关	Config	DataBaseConfig	数据库配置

说明	后缀	示例	示例说明
状态相关	Status	OrderStatus	订单状态
模式、方式相关	Mode	OpenMode	打开方式
种类相关	Category / Type 二选一	UserCategory	用户种类
工厂类相关	Factory	ConnectionFactory	连接工厂
启用相关	Enabled	ExportEnabled	开启导出
流相关	Stream	UploadStream	上传流
读取器相关	Reader	ExcelReader	Excel读取器
写入器相关	Writer	ExcelWriter	Excel写入器
适配器相关	Adapter	IntroOPAdapter	IntroOP适配器
提供器相关	Provider	MemebershipProvider	会员信息提供器
包装器相关	Wrapper	ProductWrapper	Product包装器
连接相关	Connection	ExcelConnection	Excel连接

六、一些例子

常见类型命名

类型	命名	类型	命名
客户	Customer	分销商	Reseller
零售商	Retailer	经销商/批发商	Dealer
用户	UserInfo（User为数据库关键字）	订单	OrderInfo（Order为数据库关键字）
供应商	Supplier	管理员	Admin
密码	Password	会员	Member
评论	Remark（弃用Comment）	文章	Article
新闻	News	发票	Invoice
导入	Import	导出	Export
公司、企业	Company（弃用Enterprise）	产品	Product
省份	Province	城市	City
区县	District	地址	Address
角色	Role（弃用Group）	权限	Authority（弃用Permission）
仓库	Warehouse	工厂	Plant
登录	Login（弃用SignIn）	登出	LogOut（弃用SignOut）
创建	Create（弃用Add）	编辑	Edit
更新	Update	删除	Remove（弃用Delete）
照片	Photo	图片	Image

六、一些例子

常见字段、属性命名

类型	名称	类型	名称
Id（int型）	Id（“d”小写，弃用ID）	GuidId（Guid型）	Id
Name	名称	Title	标题
Remark	备注、描述（弃用Memo、Description）	Category	种类（弃用Class、Type）
Linkman	联系人		

谢 谢！

Q & A