1. 避免将多个类放在一个文件里面，至于每个文件放多少个类，视具体情况而定。   
2. 一个文件应该只有一个命名空间，避免将多个命名空间放在同一个文件里面。   
3. 一个文件最好不要超过500行的代码(不包括机器产生的代码）。   
4. 一个方法的代码长度最好不要超过25行。   
5. 避免方法中有超过5个参数的情况。使用结构来传递多个参数（类和结构的不同前面已经叙述过）  
6. 每行代码不要超过80个字符。   
7. 不要手工的修改机器产生的代码。   
如果需要编辑机器产生的代码，编辑格式和风格要符合该编码标准。   
8. 避免利用注释解释显而易见的代码。   
a) 代码应该可以自解释。好的代码由可读的变量和方法命名因此不需要注释。   
10. 避免使用方法级的文档。   
a) 使用扩展的API文档说明。   
b) 只有在该方法需要被其他的开发者使用的时候才使用方法级的注释。（在C#中就是///）   
11. 不要硬编码数字的值，总是使用构造函数设定其值。   
12. 只有是自然结构才能直接使用const。   
13. 避免在只读的变量上使用const。如果想实现只读，可以直接使用readonly。   
14. 每个假设必须使用Assert检查   
a) 平均每15行要有一次检查(Assert)   
15. 代码的每一行都应该通过白盒方式的测试。   
16. 只抛出已经显示处理的异常。   
17. 在捕获(catch)语句的抛出异常子句中(throw)，总是抛出原始异常维护原始错误的堆栈分配（注意系统处理抛出异常是很耗费资源的）。   
18. 避免方法的返回值是错误代码。   
19. 尽量避免定义自定义异常类。   
20. 当需要定义自定义的异常时：   
a) 自定义异常要继承于ApplicationException。   
b) 提供自定义的序列化功能。   
21. 避免在单个程序集里使用多个Main方法。   
22. 只对外公布必要的操作，其他的则为internal。   
25. 使应用程序集尽量为最小化代码(EXE客户程序)。使用类库来替换包含的商务逻辑。   
26. 避免给枚举变量提供显式的值。   
27. 避免指定特殊类型的枚举变量。   
28. 即使if语句只有一句，也要将if语句的内容用大括号扩起来。    
29. 避免在条件语句中调用返回bool值的函数。可以使用局部变 量并检查这些局部变量。

30. 总是使用基于0开始的数组。

31. 在循环中总是显式的初始化引用类型的数组。

32. 在循环中总是显式的初始化引用类型的数组。   
33. 不要提供public 和 protected的成员变量，使用属性代替他们。   
34. 避免在继承中使用new而使用override替换。   
35. 在不是sealed的类中总是将public 和 protected的方法标记成virtual的。   
36. 除非使用interop(COM+ 或其他的dll)代码否则不要使用不安全的代码(unsafe code)。   
37. 避免显示的转换，使用as操作符进行兼容类型的转换。   
38. 当类成员包括委托的时候   
b) 在调用委托之前一定要检查它是否为null   
39. 不要提供公共的事件成员变量，使用事件访问器替换这些变量。   
40. 使用一个事件帮助类来公布事件的定义。   
41. 总是使用接口。   
42. 类和接口中的方法和属性至少为2:1的比例。   
43. 避免一个接口中只有一个成员。   
44. 尽量使每个接口中包含3－5个成员。   
45. 接口中的成员不应该超过20个。   
a) 实际情况可能限制为12个   
46. 避免接口成员中包含事件。   
47. 避免使用抽象方法而使用接口替换。   
48. 在类层次中显示接口。   
49. 推荐使用显式的接口实现。   
50. 从不假设一个类型兼容一个接口。  
51. 表现给最终用户的字符串不要使用硬编码而要使用资源文件替换之。   
52. 不要硬编码可能更改的基于配置的字符串，比如连接字符串。   
53. 当需要构建长的字符串的时候，使用StringBuilder不要使用string   
54. 避免在结构里面提供方法。   
a) 建议使用参数化构造函数   
b) 可以重裁操作符   
55. 总是要给静态变量提供静态构造函数。   
56. 能使用早期绑定就不要使用后期绑定。   
57. 使用应用程序的日志和跟踪。   
58. 除非在不完全的switch语句中否则不要使用goto语句。   
59. 在switch语句中总是要有default子句来显示信息(Assert)。

60. 除非在构造函数中调用其他构造函数否则不要使用this指针。

60. 除非在构造函数中调用其他构造函数否则不要使用this指针。   
61. 除非你想重写子类中存在名称冲突的成员或者调用基类的构造函数否则不要使用base来访问基类的成员。   
62. 基于模板的时候要实现Dispose()和Finalize()两个方法。   
63. 通常情况下避免有从System.Object转换来和由System.Object转换去的代码，而使用强制转换或者as操作符替换。  
64. 在一般情况下不要定影有限制符的接口。接口的限制级别通常可以用强类型来替换之  
65. 不确定在接口内的具体方法的限制条件。   
66. 总是选择使用C#内置（一般的generics）的数据结构。

67.Foreach比for语句具有更高效的执行效率,for写入时间大约是读取时间的10倍.

68.避免使用ArrayList,因为任何放入ArrayList中的对象均要进行装箱为object 对象。

69.实用Hashtable代替其他字典，存放少量数据时使用。

70.将固定字符串放在常量中定义。

71.不实用UpperCase和LowerCase进行字符串的比较，而改用Compare（）。

72.用StringBuilder来代替字符串的连接+操作符。

73.如果对于XML文件只涉及到读取，使用XPathDocument代替XDocument对象。

74.避免在循环中创建对象。

75.捕获指定异常，不要只使用通用的Exception对象。

76.不要使用Exception来控制程序流程。

77.使用using和Try/Catch来进行资源清理。

78.避免使用反射，反射会降低系统性能。在使用反射时，会进行校验参数，检查权限，可以使用动态构建类型来应用。

79.使用值类型的Tostring（）来避免装箱操作。

其实如果您经常使用IL进行反汇编来查看您写的代码的话，那么您就会发现，上面的这些规则其实就是微软在开发C#语言所做的一些内部规定优化。做不 了游戏规则的制定者，就好好遵守游戏规则吧！