软件编程规范总则 CHECKLIST

检　查　人：_____

检查日期：_____年_____月_____日

审查内容：_____

审查结果：通过□　　不通过□

说　　明：_____

序号	总 则 条 款	执行情况	说明
	1 排版		
1	¹1-1：程序块要采用缩进风格编写，缩进的空格数为4个。	是[ ] 否[ ] 免[ ]	
2	¹1-2：相对独立的程序块之间、变量说明之后必须加空行。	是[ ] 否[ ] 免[ ]	
3	¹1-3：较长的语句（>80字符）要分成多行书写，长表达式要在低优先级操作符处划分新行，操作符放在新行之首，划分出的新行要进行适当的缩进，使排版整齐，语句可读。	是[ ] 否[ ] 免[ ]	
4	¹1-4：循环、判断等语句中若有较长的表达式或语句，则要进行适应的划分，长表达式要在低优先级操作符处划分新行，操作符放在新行之首。	是[ ] 否[ ] 免[ ]	
5	¹1-5：若函数或过程中的参数较长，则要进行适当的划分。	是[ ] 否[ ] 免[ ]	
6	¹1-6：不允许把多个短语句写在一行中，即一行只写一条语句。	是[ ] 否[ ] 免[ ]	
7	¹1-7：if、while、for、default、do等语句自占一行。	是[ ] 否[ ] 免[ ]	
8	¹1-8：对齐只使用空格键，不使用TAB键。	是[ ] 否[ ] 免[ ]	
9	¹1-9：函数或过程的开始、结构的定义及循环、判断等语句中的代码都要采用缩进风格，case语句下的情况处理语句也要遵从语句缩进要求。	是[ ] 否[ ] 免[ ]	
10	¹1-10：程序块的分界符（如C/C++语言的大括号‘{’和‘}’）应各独占一行并且位于同一列，同时与引用它们的语句左对齐。在函数体的开	是[ ] 否[ ] 免[ ]	

序号	总 则 条 款	执行情况	说明
	始、类的定义、结构的定义、枚举的定义以及if、for、do、while0、switch、case语句中的程序都要采用如上的缩进方式。		
11	¹1-11：在两个以上的关键字、变量、常量进行对等操作时，它们之间的操作符之前、之后或者前后要加空格；进行非对等操作时，如果是关系密切的立即操作符（如－>），后不应加空格。	是[ ] 否[ ] 免[ ]	
	2 注释		
	¹2-1：一般情况下，源程序有效注释量必须在20%以上。	是[ ] 否[ ] 免[ ]	
	¹2-2：说明性文件（如头文件.h文件、.inc文件、.def文件、编译说明文件.cfg等）头部应进行注释，注释必须列出：版权说明、版本号、生成日期、作者、内容、功能、与其它文件的关系、修改日志等，头文件的注释中还应有函数功能简要说明。	是[ ] 否[ ] 免[ ]	
	¹2-3：源文件头部应进行注释，列出：版权说明、版本号、生成日期、作者、模块目的/功能、主要函数及其功能、修改日志等。	是[ ] 否[ ] 免[ ]	
	¹2-4：函数头部应进行注释，列出：函数的目的/功能、输入参数、输出参数、返回值、调用关系（函数、表）等。	是[ ] 否[ ] 免[ ]	
	¹2-5：边写代码边注释，修改代码同时修改相应的注释，以保证注释与代码的一致性。不再有用的注释要删除。	是[ ] 否[ ] 免[ ]	
	¹2-6：注释的内容要清楚、明了，含义准确，防止注释二义性。	是[ ] 否[ ] 免[ ]	
	¹2-7：避免在注释中使用缩写，特别是非常用缩写。	是[ ] 否[ ] 免[ ]	
	¹2-8：注释应与其描述的代码相近，对代码的注释应放在其上方或右方（对单条语句的注释）相邻位置，不可放在下面，如放于上方则需与其上面的代码用空行隔开。	是[ ] 否[ ] 免[ ]	
	¹2-9：对于所有有物理含义的变量、常量，如果其命名不是充分自注释的，在声明时都必须加以注释，说明其物理含义。变量、常量、宏的注释应放在其上方相邻位置或右方。	是[ ] 否[ ] 免[ ]	

序号	总 则 条 款	执行情况	说明
	¹2-10：数据结构声明(包括数组、结构、类、枚举等)，如果其命名不是充分自注释的，必须加以注释。对数据结构的注释应放在其上方相邻位置，不可放在下面；对结构中的每个域的注释放在此域的右方。	是[ ] 否[ ] 免[ ]	
	¹2-11：全局变量要有较详细的注释，包括对其功能、取值范围、哪些函数或过程存取它以及存取时注意事项等的说明。	是[ ] 否[ ] 免[ ]	
	¹2-12：注释与所描述内容进行同样的缩排。	是[ ] 否[ ] 免[ ]	
	¹2-13：将注释与其上面的代码用空行隔开。	是[ ] 否[ ] 免[ ]	
	¹2-14：对变量的定义和分支语句（条件分支、循环语句等）必须编写注释。	是[ ] 否[ ] 免[ ]	
	¹2-15：对于switch语句下的case语句，如果因为特殊情况需要处理完一个case后进入下一个case处理，必须在该case语句处理完、下一个case语句前加上明确的注释。	是[ ] 否[ ] 免[ ]	
3 标识符命名			
	¹3-1：标识符的命名要清晰、明了，有明确含义，同时使用完整的单词或大家基本可以理解的缩写，避免使人产生误解。	是[ ] 否[ ] 免[ ]	
	¹3-2：命名中若使用特殊约定或缩写，则要有注释说明。	是[ ] 否[ ] 免[ ]	
	¹3-3：自己特有的命名风格，要自始至终保持一致，不可来回变化。	是[ ] 否[ ] 免[ ]	
	¹3-4：对于变量命名，禁止取单个字符（如i、j、k...），建议除了要有具体含义外，还能表明其变量类型、数据类型等，但i、j、k作局部循环变量是允许的。	是[ ] 否[ ] 免[ ]	
	¹3-5：命名规范必须与所使用的系统风格保持一致，并在同一项目中统一，比如采用UNIX的全小写加下划线的风格或大小写混排的方式，不要使用大小写与下划线混排的方式。	是[ ] 否[ ] 免[ ]	
4 可读性			
	¹4-1：注意运算符的优先级，并用括号明确表达式的操作顺序，避免使用默认优先级。	是[ ] 否[ ] 免[ ]	

序号	总 则 条 款	执行情况	说明
	¹4-2：避免使用不易理解的数字，用有意义的标识来替代。涉及物理状态或者含有物理意义的常量，不应直接使用数字，必须用有意义的枚举或宏来代替。	是[ ] 否[ ] 免[ ]	
	5 变量		
	¹5-1：去掉没必要的公共变量。	是[ ] 否[ ] 免[ ]	
	¹5-2：仔细定义并明确公共变量的含义、作用、取值范围及公共变量间的关系。		
	¹5-3：明确公共变量与操作此公共变量的函数或过程的关系，如访问、修改及创建等。		
	¹5-4：当向公共变量传递数据时，要十分小心，防止赋与不合理的值或越界等现象发生。		
	¹5-5：防止局部变量与公共变量同名。		
	¹5-6：严禁使用未经初始化的变量作为右值。		
	6 函数、过程		
	¹6-1：对所调用函数的错误返回码要仔细、全面地处理。	是[ ] 否[ ] 免[ ]	
	¹6-2：明确函数功能，精确（而不是近似）地实现函数设计。	是[ ] 否[ ] 免[ ]	
	¹6-3：编写可重入函数时，应注意局部变量的使用(如编写C/C++语言的可重入函数时，应使用auto即缺省态局部变量或寄存器变量)。	是[ ] 否[ ] 免[ ]	
	¹6-4：编写可重入函数时，若使用全局变量，则应通过关中断、信号量（即P、V操作）等手段对其加以保护。	是[ ] 否[ ] 免[ ]	
	7 可测性		
	¹7-1：在同一项目组或产品组内，要有一套统一的为集成测试与系统联调准备的调测开关及相应打印函数，并且要有详细的说明。	是[ ] 否[ ] 免[ ]	
	¹7-2：在同一项目组或产品组内，调测打印出的信息串的格式要有统一的形式。信息串中至少要有所在模块名（或源文件名）及行号。	是[ ] 否[ ] 免[ ]	
	¹7-3：编程的同时要为单元测试选择恰当的测试点，并仔细构造测试代码、测试用例，同时给出	是[ ] 否[ ] 免[ ]	

序号	总 则 条 款	执行情况	说明
	明确的注释说明。测试代码部分应作为（模块中的）一个子模块，以方便测试代码在模块中的安装与拆卸（通过调测开关）。		
	17-4：在进行集成测试/系统联调之前，要构造好测试环境、测试项目及测试用例，同时仔细分析并优化测试用例，以提高测试效率。	是[ ] 否[ ] 免[ ]	
	17-5：使用断言来发现软件问题，提高代码可测性。	是[ ] 否[ ] 免[ ]	
	17-6：用断言来检查程序正常运行时不应发生但在调测时有可能发生的非法情况。	是[ ] 否[ ] 免[ ]	
	17-7：不能用断言来检查最终产品肯定会出现且必须处理的错误情况。	是[ ] 否[ ] 免[ ]	
	17-8：对较复杂的断言加上明确的注释。	是[ ] 否[ ] 免[ ]	
	17-9：用断言确认函数的参数。	是[ ] 否[ ] 免[ ]	
	17-10：用断言保证没有定义的特性或功能不被使用。	是[ ] 否[ ] 免[ ]	
	17-11：用断言对程序开发环境（OS/Compiler/Hardware）的假设进行检查。	是[ ] 否[ ] 免[ ]	
	17-12：正式软件产品中应把断言及其它调测代码去掉（即把有关的调测开关关掉）。	是[ ] 否[ ] 免[ ]	
	17-13：在软件系统中设置与取消有关测试手段，不能对软件实现的功能等产生影响。	是[ ] 否[ ] 免[ ]	
	17-14：用调测开关来切换软件的DEBUG版和正式版，而不要同时存在正式版本和DEBUG版本的不同源文件，以减少维护的难度。	是[ ] 否[ ] 免[ ]	
	17-15：软件的DEBUG版本和发行版本应该统一维护，不允许分家，并且要时刻注意保证两个版本在实现功能上的一致性。	是[ ] 否[ ] 免[ ]	
8 程序效率			
	18-1：编程时要经常注意代码的效率。	是[ ] 否[ ] 免[ ]	
	18-2：在保证软件系统的正确性、稳定性、可读性及可测性的前提下，提高代码效率。	是[ ] 否[ ] 免[ ]	
	18-3：局部效率应为全局效率服务，不能因为提高局部效率而对全局效率造成影响。	是[ ] 否[ ] 免[ ]	
	18-4：通过对系统数据结构的划分与组织的改进，	是[ ] 否[ ] 免[ ]	

序号	总 则 条 款	执行情况	说明
	以及对程序算法的优化来提高空间效率。		
	[18]8-5：循环体内工作量最小化。	是[ ] 否[ ] 免[ ]	
	9 质量保证		
	[19]9-1：在软件设计过程中构筑软件质量。	是[ ] 否[ ] 免[ ]	
	[19]9-2：代码质量保证优先原则	是[ ] 否[ ] 免[ ]	
	[19]9-3：只引用属于自己的存贮空间。	是[ ] 否[ ] 免[ ]	
	[19]9-4：防止引用已经释放的内存空间。	是[ ] 否[ ] 免[ ]	
	[19]9-5：过程/函数中分配的内存，在过程/函数退出之前要释放。	是[ ] 否[ ] 免[ ]	
	[19]9-6:过程/函数中申请的(为打开文件而使用的)文件句柄，在过程/函数退出之前要关闭。	是[ ] 否[ ] 免[ ]	
	[19]9-7：防止内存操作越界。	是[ ] 否[ ] 免[ ]	
	[19]9-8：认真处理程序所能遇到的各种出错情况。	是[ ] 否[ ] 免[ ]	
	[19]9-9：系统运行之初，要初始化有关变量及运行环境，防止未经初始化的变量被引用。	是[ ] 否[ ] 免[ ]	
	[19]9-10：系统运行之初，要对加载到系统中的数据进行一致性检查。	是[ ] 否[ ] 免[ ]	
	[19]9-11:严禁随意更改其它模块或系统的有关设置和配置。	是[ ] 否[ ] 免[ ]	
	[19]9-12：不能随意改变与其它模块的接口。	是[ ] 否[ ] 免[ ]	
	[19]9-13：充分了解系统的接口之后，再使用系统提供的功能。	是[ ] 否[ ] 免[ ]	
	[19]9-14：编程时，要防止差1错误。	是[ ] 否[ ] 免[ ]	
	[19]9-15：要时刻注意易混淆的操作符。当编完程序后，应从头至尾检查一遍这些操作符，以防止拼写错误。	是[ ] 否[ ] 免[ ]	
	[19]9-16：有可能的话，if语句尽量加上else分支，对没有else分支的语句要小心对待；switch语句必须有default分支。	是[ ] 否[ ] 免[ ]	
	10 代码编辑、编译、审查		
	[1]10-1:打开编译器的所有告警开关对程序进行编译。	是[ ] 否[ ] 免[ ]	
	[1]10-2：在产品软件（项目组）中，要统一编译开	是[ ] 否[ ] 免[ ]	

序号	总 则 条 款	执行情况	说明
	关选项。		
	¹10-3：通过代码走读及审查方式对代码进行检查。	是[ ] 否[ ] 免[ ]	
	¹10-4：测试部测试产品之前，应对代码进行抽查及评审。	是[ ] 否[ ] 免[ ]	
	11 代码测试、维护		
	¹11-1：单元测试要求至少达到语句覆盖。	是[ ] 否[ ] 免[ ]	
	¹11-2：单元测试开始要跟踪每一条语句，并观察数据流及变量的变化。	是[ ] 否[ ] 免[ ]	
	¹11-3：清理、整理或优化后的代码要经过审查及测试。	是[ ] 否[ ] 免[ ]	
	¹11-4：代码版本升级要经过严格测试。	是[ ] 否[ ] 免[ ]	
	¹11-5：使用工具软件对代码版本进行维护。	是[ ] 否[ ] 免[ ]	
	¹11-6：正式版本上软件的任何修改都应有详细的文档记录。	是[ ] 否[ ] 免[ ]	
	12 宏		
	¹12-1：用宏定义表达式时，要使用完备的括号。	是[ ] 否[ ] 免[ ]	
	¹12-2：将宏所定义的多条表达式放在大括号中。	是[ ] 否[ ] 免[ ]	
	¹12-3：使用宏时，不允许参数发生变化。	是[ ] 否[ ] 免[ ]	