C语言技术管理规范文档

0 前言

为了提高产品代码质量，指导广大软件开发人员编写出简洁、可维护、可靠、可测试、高效、可移植的代码，参考业界编程规范成果，对容易遇到的一些编码问题进行梳理，编写了本规范。

1 代码风格

* 1. **强制**  
     1.1.1 程序块采用缩进风格编写，每级缩进为4个空格  
     1.1.2 相对独立的程序块间、变量说明后必须空一行  
     1.1.3 多个短语句（包括赋值语句）不允许写在同一行内，即一行仅能写一条语句  
     1.1.4 一条语句不能过长，不能拆分需要分行写

1.1.5 if、for、do、while、case、switch、default等语句独占一行

* 1. **推荐**  
     1.2.1 源程序中关系较为紧密的代码尽可能相邻  
     1.2.2 注释符（包括‘/\*’‘//’‘\*/’）与注释内容之间用一个空格进行分隔，例如：  
      /\* 注释内容 \*/
  2. **允许**  
     1.3.1 可以使用单行注释（//）或块注释（/\* \*/），但需保持团队一致。

2 代码结构

**2.1 强制**  
2.1.1 禁止头文件循环依赖  
2.1.2 .c/.h文件禁止包含用不到的头文件  
2.1.3 头文件应当自包含  
2.1.4 总是编写内部#include保护符(#define保护)，格式示例：

#ifndef MODULE\_NAME\_H

#define MODULE\_NAME\_H

// 头文件内容

#endif  
2.1.5 禁止在头文件中定义变量  
2.1.6 所有函数需在头文件中声明（静态函数除外）

**2.2 推荐**  
2.2.1 同一产品统一包含头文件排列方式  
2.2.2 函数不变参数使用const修饰  
2.2.3 函数的参数个数不超过5个

**2.3 允许**

2.3.1 性能关键路径可使用inline函数

3 内存与安全  
**3.1 强制**

3.1.1 禁止内存操作越界  
3.1.2 禁止内存泄漏  
3.1.3 禁止引用已经释放的内存空间  
3.1.4 对用户输入进行检查（如长度，合法性）  
3.1.5 禁止 strcpy/strcat，改用 strncpy/strncat

**3.2 推荐**3.2.1 函数中分配的内存，在函数退出前要释放

3.2.2 避免滥用goto语句  
3.2.3 使用工具（如 cppcheck、Coverity）检测内存泄漏

**3.3 允许**3.3.1 C99允许可变长数组（VLA），但嵌入式环境慎用（栈溢出风险）

4 性能与优化

**4.1 强制**4.1.1 嵌入式环境中避免使用浮点数

**4.2 推荐**4.2.1 创建资源库，以减少分配对象的开销  
4.2.2 对于多维大数组，避免来回跳跃式访问数组成员  
4.2.3 将不变条件的计算移到循环体外

**4.3 允许**4.3.1 在驱动或性能关键代码中允许使用内联汇编

5 测试与调试

**5.1 强制**5.1.1 在编写代码的同时，或者编写代码前，编写单元测试用例验证软件设计/编码的正确

5.1.2 单元测试关注单元的行为而非实现，避免针对函数的测试

5.1.3 关键路径添加调试日志

5.1.4 使用断言记录内部假设

5.1.5 不能用断言来检查运行时错误

**5.2 推荐**5.2.1 为单元测试和系统故障注入测试准备好方法和通道  
**5.3 允许**

5.3.1 允许使用宏来控制调试输出

6 其他

**6.1 强制**

6.1.1 禁止使用非标准扩展（如 gcc 特有语法），确保代码可移植性。

**6.2 推荐**

6.2.1 使用 clang-format 或 astyle 统一风格。

**6.3允许**

6.3.1 必要时可以使用 \_\_attribute\_\_((packed)) 等，但需注释原因。