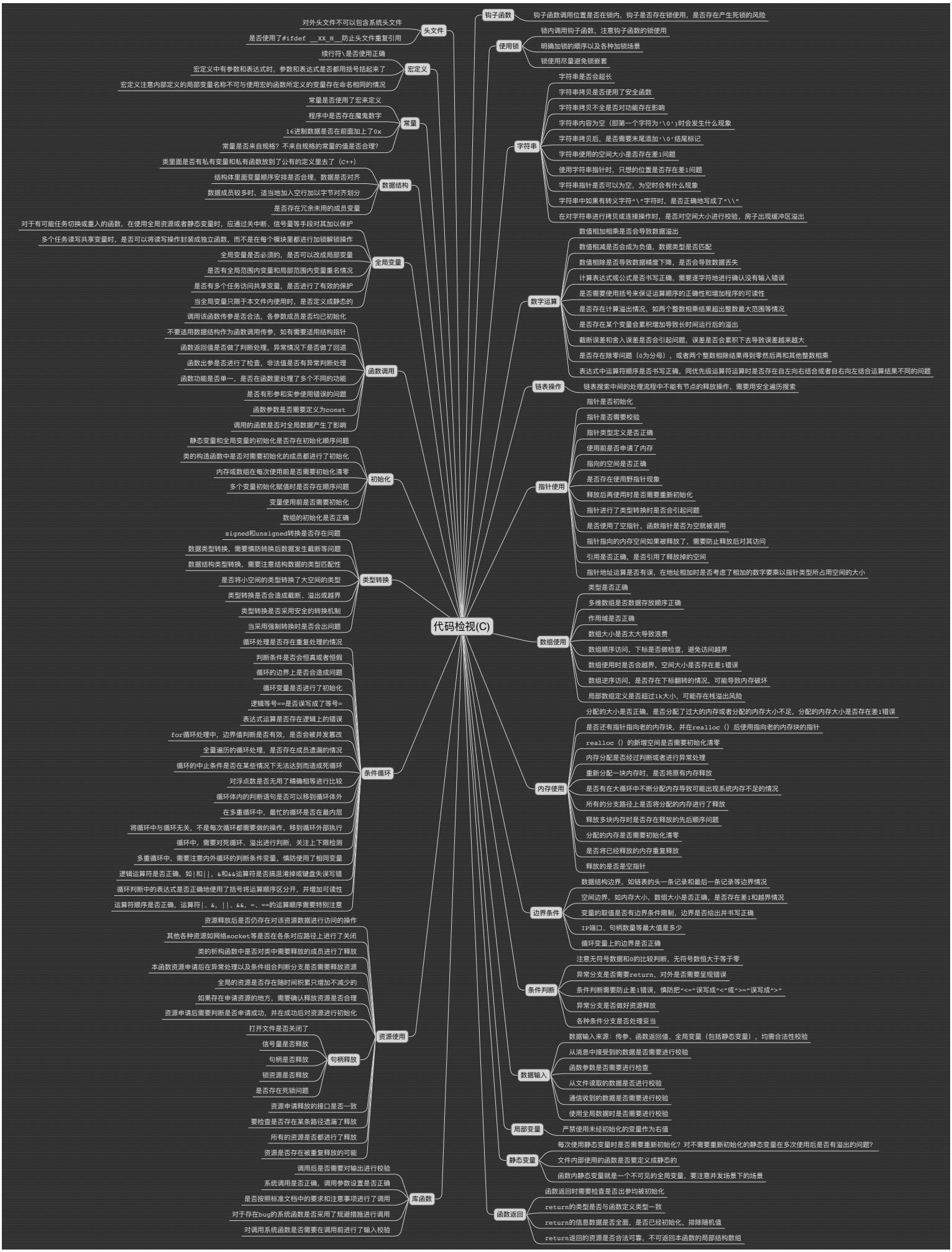
代码检视(C)
头文件
对外头文件不可以包含系统头文件
是否使用了#ifdef __XX_H__防止头文件重复引用
宏定义
续行符\是否使用正确
宏定义中有参数和表达式时，参数和表达式是否都用括号括起来了
宏定义注意内部定义的局部变量名称不可与使用宏的函数所定义的变量存在命名相同的情况
常量
常量是否使用了宏来定义
程序中是否存在魔鬼数字
16进制数据是否在前面加上了0x
常量是否来自规格？不来自规格的常量的值是否合理？
数据结构
类里面是否有私有变量和私有函数放到了公有的定义里去了（C++）
结构体里面变量顺序安排是否合理，数据是否对齐
数据成员较多时，适当地加入空行加以字节对齐划分
是否存在冗余未用的成员变量
全局变量
对于有可能任务切换或重入的函数，在使用全局资源或者静态变量时，应通过关中断、信号量等手段对其加以保护
多个任务读写共享变量时，是否可以将读写操作封装成独立函数，而不是在每个模块里都进行加锁解锁操作
全局变量是否必须的，是否可以改成局部变量
是否有全局范围内变量和局部范围内变量重名情况
是否有多个任务访问共享变量，是否进行了有效的保护
当全局变量只限于本文件内使用时，是否定义成静态的
函数调用
调用该函数传参是否合法，各参数成员是否均已初始化
不要适用数据结构作为函数调用传参，如有需要适用结构指针
函数返回值是否做了判断处理，异常情况下是否做了回退
函数出参是否进行了检查，非法值是否有异常判断处理
函数功能是否单一，是否在函数里处理了多个不同的功能
是否有形参和实参使用错误的问题
函数参数是否需要定义为const
调用的函数是否对全局数据产生了影响
初始化
静态变量和全局变量的初始化是否存在初始化顺序问题
类的构造函数中是否对需要初始化的成员都进行了初始化
内存或数组在每次使用前是否需要初始化清零
多个变量初始化赋值时是否存在顺序问题
变量使用前是否需要初始化
数组的初始化是否正确
类型转换
signed和unsigned转换是否存在问题
数据类型转换，需要慎防转换后数据发生截断等问题
数据结构类型转换，需要注意结构数据的类型匹配性
是否将小空间的类型转换了大空间的类型
类型转换是否会造成截断、溢出或越界
类型转换是否采用安全的转换机制
当采用强制转换时是否会出问题
条件循环
循环处理是否存在重复处理的情况
判断条件是否会恒真或者恒假
循环的边界上是否会造成问题
循环变量是否进行了初始化
逻辑等号==是否误写成了等号=
表达式运算是否存在逻辑上的错误
for循环处理中，边界值判断是否有效，是否会被并发篡改
全量遍历的循环处理，是否存在成员遗漏的情况
循环的中止条件是否在某些情况下无法达到而造成死循环
对浮点数是否无用了精确相等进行比较
循环体内的判断语句是否可以移到循环体外
在多重循环中，最忙的循环是否在最内层
将循环中与循环无关，不是每次循环都需要做的操作，移到循环外部执行
循环中，需要对死循环、溢出进行判断，关注上下限检测
多重循环中，需要注意内外循环的判断条件变量，慎防使用了相同变量
逻辑运算符是否正确，如|和||，&和&&运算符是否搞混淆掉或键盘失误写错
循环判断中的表达式是否正确地使用了括号将运算顺序区分开，并增加可读性
运算符顺序是否正确，运算符|、&，||、&&，=、==的运算顺序需要特别注意
资源使用
资源释放后是否仍存在对该资源数据进行访问的操作
其他各种资源如网络socket等是否在各条对应路径上进行了关闭
类的析构函数中是否对类中需要释放的成员进行了释放
本函数资源申请后在异常处理以及条件组合判断分支是否需要释放资源
全局的资源是否存在随时间积累只增加不减少的
如果存在申请资源的地方，需要确认释放资源是否合理
资源申请后需要判断是否申请成功，并在成功后对资源进行初始化
句柄释放
打开文件是否关闭了
信号量是否释放
句柄是否释放
锁资源是否释放
是否存在死锁问题
资源申请释放的接口是否一致
要检查是否存在某条路径遗漏了释放
所有的资源是否都进行了释放
资源是否存在被重复释放的可能
库函数
调用后是否需要对输出进行校验
系统调用是否正确，调用参数设置是否正确
是否按照标准文档中的要求和注意事项进行了调用
对于存在bug的系统函数是否采用了规避措施进行调用
对调用系统函数是否需要在调用前进行了输入校验
钩子函数
钩子函数调用位置是否在锁内，钩子是否存在锁使用，是否存在产生死锁的风险
使用锁
锁内调用钩子函数，注意钩子函数的锁使用
明确加锁的顺序以及各种加锁场景
锁使用尽量避免锁嵌套
字符串
字符串是否会超长
字符串拷贝是否使用了安全函数
字符串拷贝不全是否对功能存在影响
字符串内容为空（即第一个字符为'\0'）时会发生什么现象
字符串拷贝后，是否需要末尾添加'\0'结尾标记
字符串使用的空间大小是否存在差1问题
使用字符串指针时，只想的位置是否存在差1问题
字符串指针是否可以为空，为空时会有什么现象
字符串中如果有转义字符"\"字符时，是否正确地写成了"\\"
在对字符串进行拷贝或连接操作时，是否对空间大小进行校验，房子出现缓冲区溢出
数字运算
数值相加相乘是否会导致数据溢出
数值相减是否会成为负值，数据类型是否匹配
数值相除是否导致数据精度下降，是否会导致数据丢失
计算表达式或公式是否书写正确，需要逐字符地进行确认没有输入错误
是否需要使用括号来保证运算顺序的正确性和增加程序的可读性
是否存在计算溢出情况，如两个整数相乘结果超出整数最大范围等情况
是否存在某个变量会累积增加导致长时间运行后的溢出
截断误差和舍入误差是否会引起问题，误差是否会累积下去导致误差越来越大
是否存在除零问题（0为分母），或者两个整数相除结果得到零然后再和其他整数相乘
表达式中运算符顺序是否书写正确，同优先级运算符运算时是否存在自左向右结合或者自右向左结合运算结果不同的问题
链表操作
链表搜索中间的处理流程中不能有节点的释放操作，需要用安全遍历搜索
指针使用
指针是否初始化
指针是否需要校验
指针类型定义是否正确
使用前是否申请了内存
指向的空间是否正确
是否存在使用野指针现象
释放后再使用时是否需要重新初始化
指针进行了类型转换时是否会引起问题
是否使用了空指针，函数指针是否为空就被调用
指针指向的内存空间如果被释放了，需要防止释放后对其访问
引用是否正确，是否引用了释放掉的空间
指针地址运算是否有误，在地址相加时是否考虑了相加的数字要乘以指针类型所占用空间的大小
数组使用
类型是否正确
多维数组是否数据存放顺序正确
作用域是否正确
数组大小是否太大导致浪费
数组顺序访问，下标是否做检查，避免访问越界
数组使用时是否会越界，空间大小是否存在差1错误
数组逆序访问，是否存在下标翻转的情况，可能导致内存破坏
局部数组定义是否超过1k大小，可能存在栈溢出风险
内存使用
分配的大小是否正确，是否分配了过大的内存或者分配的内存大小不足，分配的内存大小是否存在差1错误
是否还有指针指向老的内存块，并在realloc（）后使用指向老的内存块的指针
realloc（）的新增空间是否需要初始化清零
内存分配是否经过判断或者进行异常处理
重新分配一块内存时，是否将原有内存释放
是否有在大循环中不断分配内存导致可能出现系统内存不足的情况
所有的分支路径上是否将分配的内存进行了释放
释放多块内存时是否存在释放的先后顺序问题
分配的内存是否需要初始化清零
是否将已经释放的内存重复释放
释放的是否是空指针
边界条件
数据结构边界，如链表的头一条记录和最后一条记录等边界情况
空间边界，如内存大小，数组大小是否正确，是否存在差1和越界情况
变量的取值是否有边界条件限制，边界是否给出并书写正确
IP端口、句柄数量等最大值是多少
循环变量上的边界是否正确
条件判断
注意无符号数据和0的比较判断，无符号数恒大于等于零
异常分支是否需要return，对外是否需要呈现错误
条件判断需要防止差1错误，慎防把"<="误写成"<"或">="误写成">"
异常分支是否做好资源释放
各种条件分支是否处理妥当
数据输入
数据输入来源：传参、函数返回值、全局变量（包括静态变量），均需合法性校验
从消息中接受到的数据是否需要进行校验
函数参数是否需要进行检查
从文件读取的数据是否进行校验
通信收到的数据是否需要进行校验
使用全局数据时是否需要进行校验
局部变量
严禁使用未经初始化的变量作为右值
静态变量
每次使用静态变量时是否需要重新初始化？对不需要重新初始化的静态变量在多次使用后是否有溢出的问题？
文件内部使用的函数是否要定义成静态的
函数内静态变量就是一个不可见的全局变量，要注意并发场景下的场景
函数返回
函数返回时需要检查是否出参均被初始化
return的类型是否与函数定义类型一致
return的信息数据是否全面，是否已经初始化，排除随机值
return返回的资源是否合法可靠，不可返回本函数的局部结构数组