# C++编码风格指南

|  |  |  |
| --- | --- | --- |
| 版本 | 修改 | 修改人 |
| 1.0 | 完成初稿 | 周阳 |
| 1.1 | 修改函数、变量、结构体和枚举的命名规范和使用规范  添加不允许程序中间退出和使用英文注释的规则 | 周阳 |

## 一、前言

为形成统一的C++编码风格，以保障项目代码的可读性。本文档主要参考腾讯c++编码规范和谷歌C++编码规范。

## 二、格式

### 1、行长度

**总述：**

每一行代码字符数不超过120

**说明：**

以下情况应分多行书写：

1)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。

2)若函数或过程中的参数较长，则要进行适当的划分。

3)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。

### 2、制表符

**总述：**

只使用空格，每次缩进使用4个字符

**说明：**

使用空格代替制表符的最大优点是保证在所有电脑上缩进效果是一致的。如果习惯使用TAB键可以在设置中选择“插入空格”并设置大小为4。

在linux的C标准中推荐使用8个字符进行缩进。Linux给出的理由是8格的缩进更容易让开发者发现自己的嵌套层数过多需要进行函数拆分。

### 3、空行

**总述：**

相对独立的程序块之间、变量说明之后必须加空行。不允许把多个短语句写在一行中，即一行只写一条语句。

**说明：**

1)函数之间应该用空行分开；

2)用空行将代码按照逻辑片断划分；

3)每个类声明之后应该加入空格同其他代码分开。

一行代码只做一件事情，如只定义一个变量，或只写一条语句。这样的代码容易阅读，并且方便于写注释。

### 4、大括号缩进

**总述：**

程序块的分界符应各独占一行并且 位于同一列，同时与引用它们的语句左对齐。

**说明：**

有的编辑器在选择大括号时会自动标记相对应的括号，如果写在条件后会不太容易发现，容易出现多写或者少写的情况。

### 5、空格

**总述：**

代码行之内要留有适当的空格，采用这种松散方式编写代码的目的是使代码更加清晰。

**说明：**

1)关键字之后要留空格。象 const、virtual、inline、case 等关键字之后至少要留一个空格， 否则无法辨析关键字。象 if、for、while 等关键字之后应留一个空格再跟左括号‘（ ’， 以突出关键字。

2)函数名之后不要留空格， 紧跟左括号’(’ ， 以与关键字区别。

3)‘( ’ 向后紧跟，‘ )’、‘ ，’、‘ ;’ 向前紧跟， 紧跟处不留空格。

4）值操作符、比较操作符、算术操作符、逻辑操作符、位域操作符，如“ =”、“ +=” “ >=”、“ <=”、“ +”、“ \*”、“ %”、“ &&”、“ ||”、“ <<” 、“ ^” 等二元操作符的前后应当加空格。

5)一元操作符如“ !”、“ ~”、“ ++”、“ --”、“ &”（ 地址运算符） 等前后不加空格。

象“［ ］”、“ .”、“ ->” 这类操作符前后不加空格。

可以通过使用VS的自动格式代码调整代码中的空格。

## 三、命名约定

命名尽量使用英文单词 ：命名尽量使用英文单词，力求简单清楚 ，力求简单清楚，避免使用引起误解的词汇和模糊的 ，避免使用引起误解的词汇和模糊的 缩写，使人产生误解 ，使人产生误解。

对于一些过长的英文可以使用通用的缩写。如: btn tmp dst

### 1、类的命名

**总述：**

类命名使用大驼峰式命名。

**说明：**

对于继承于qt的类命名采用功能加类型的命名方式。例：MainWidget LoginDialog

对于一些功能类命名尽量保证可以从类名可以直观得了解类的功能。

### 2、函数的命名

**总述：**

函数的名称使用linux c式命名规范。即采用下划线分隔字符串。全字母小写。基本原则（函数名 = 动词+名词+形容词）

**说明：**

因为代码中和设备交互的函数会有许多名称差不多的函数。如果使用驼峰式命名在一堆函数中找关键词时没有下划线式方便。

对于函数中的参数使用宏IN OUT INOUT写在前面用于区分是入参还是出参。参数顺序为IN INOUT OUT。

例：

**int** load\_image\_gray(IN **const** **char** **\***pcFileName, INOUT **int** **\***piWidth, INOUT **int** **\***piHeight, OUT **char** **\***pcImg);

**备注：**

对于qt的信号和槽函数为了避免qt的自动绑定 (不用on开头，槽函数用slot开头，信号函数用sig开头)。使用connect不使用自动绑定在对于一个功能想对多个按钮绑定和功能关闭更为方便。为了方便区分信号和槽函数对函数命名有以下规范：

槽函数以slot开头，例：slot\_show\_login()。

信号函数以sig开头,例:sig\_show\_login()。

### 3、变量的命名

**总述：**

1. 变量使用匈牙利命名法；
2. 禁止使用过度简洁和抽象的缩写（如rn, js等）。
3. 命名规范遵守见名知义原则。

**说明：**

基本原则是：变量名=属性+类型+对象描述，其中每一[对象](https://baike.baidu.com/item/%E5%AF%B9%E8%B1%A1/17158" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)的名称都要求有明确[含义](https://baike.baidu.com/item/%E5%90%AB%E4%B9%89/316823" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)，可以取对象名字全称或名字的一部分。要基于容易记忆容易理解的原则。保证名字的连贯性是非常重要的。

**常用属性：**

g\_ [全局变量](https://baike.baidu.com/item/%E5%85%A8%E5%B1%80%E5%8F%98%E9%87%8F" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)

c\_ [常量](https://baike.baidu.com/item/%E5%B8%B8%E9%87%8F" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)

m\_ c++类[成员变量](https://baike.baidu.com/item/%E6%88%90%E5%91%98%E5%8F%98%E9%87%8F" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)

s\_ [静态变量](https://baike.baidu.com/item/%E9%9D%99%E6%80%81%E5%8F%98%E9%87%8F" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)

**常用类型：**

数组 a

[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)　p

函数　fn

无效　v

[句柄](https://baike.baidu.com/item/%E5%8F%A5%E6%9F%84" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)　h

[长整型](https://baike.baidu.com/item/%E9%95%BF%E6%95%B4%E5%9E%8B" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)　l

布尔　b

[浮点型](https://baike.baidu.com/item/%E6%B5%AE%E7%82%B9%E5%9E%8B" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)（有时也指文件）　f

双字 　dw

字符串　 sz

短整型　 n

双精度浮点　d

[计数](https://baike.baidu.com/item/%E8%AE%A1%E6%95%B0" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)　cnt

[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)　c

[整型](https://baike.baidu.com/item/%E6%95%B4%E5%9E%8B" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)　i

[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82" \t "https://baike.baidu.com/item/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95/_blank)　by

字　w

实型　r

无符号　u

**描述部分：**

最大　Max

最小　Min

初始化　Init

临时变量　T（或Temp）

源对象　Src

目的对象　Dest

**示例：**

int m\_iMaxWndHeignt; (含义：保存窗口最大高度的成员变量类型为int)。

### 4、文件的命名

**总述：**

文件名称和类全保持一致

**说明：**

文件和类名一致有助于找到相关类。一个文件中尽量保证只有一个类。

### 5、结构体和枚举的命名

**总述：**

结构体和枚举类型的命名规范和类名保持一致。对于枚举内的变量采用全大写的方式通过下划线进行分隔。

**说明：**

结构体和枚举的命名基本格式以（**模块名\_功能\_类型（STRUCT/ENUM）**）。枚举内命名规范以（**模块名\_功能\_具体类型**）

**结构体示例：**

**struct** ALC4A\_MATCH\_RESULT\_STRUCT

{

**double** dX;

**double** dY;

**double** dR;

};

**枚举示例：**

**enum** ALC4A\_STATE\_ENUM

**{**

ALC4A\_STATE\_MIN **= 0,**

ALC4A\_STATE\_TERM **= 1,**

ALC4A\_STATE\_IDLE **= 2,**

ALC4A\_STATE\_ERROR **= 3,**

ALC4A\_STATE\_MAX **= 4**

**};**

### 6、宏定义的命名

**总述：**

宏定义使用全大写命名。变量间通过下划线分隔。

**说明：**

因为作用域的原因对于一些常量并不一定要使用宏定义，可以使用const变量替代。

### 7、qt控件的命名

**总述：**

对于一些全局控件使用小驼峰式命名。名称中可以不携带控件的类型。

**说明：**

有的时候命名的时候会携带控件类型如LoginBtn可以方便了解控件是按钮，但有些控件名称过长且功能单一我觉得可以不强制要求。

## 注释

注释虽然写起来很痛苦, 但对保证代码可读性至关重要。下面的规则描述了如何注释以及在哪儿注释. 当然也要记住: 注释固然很重要, 但最好的代码应当本身就是文档.。有意义的类型名和变量名, 要远胜过要用注释解释的含糊不清的名字。

程序中的注释因为编码的原因，程序中的注释以**英文注释**为主。

除了类和函数注释外不推荐使用/\*\*/注释的方式，尽量以//为主。每个注释在单独一行。不可以写在语句的后面

错误示例：

QPushButton\* m\_loginBtn; // login button

正确示例：

// login button

QPushButton\* m\_loginBtn;

注释应该和代码同时更新：注释应该和代码同时更新，不再有用的注释要删除 ，不再有用的注释要删除。

### 1、文件的注释

**总述：**

文件的注释可以更好的了解文件中类的功能。

**说明：**

在文件的注释中需要表达清楚文件是做何用途。文件的修改日志也可以体现在注释中

示例：

/\*

\* 这个类主要实现了用户登录窗口

\* 开发者：A

\* 版本：1.0

\* 添加了手机登录的功能

\* 开发者：B

\* 版本：1.1

\*/

### 2、结构体、枚举的注释

**总述：**

结构体和枚举使用 // 式注释。

**说明：**

对于结构体和枚举中的变量最好每个都提供相应的注释。语义很清楚的情况下可以在整体的注释中说明。

### 3、函数的注释

**总述：**

函数采用/\*\*/式注释。在注释中要表明函数的功能和每个变量的含义

**说明：**

如果函数为重载函数且父函数添加过注释的情况下可以不添加注释。注释格式可以不相同，但对于函数和变量的功能要注释明确

示例：

/\*

函数的主要功能

@param [in] a 输入变量

@param [out] b 输出变量

@return 函数返回值

\*/

### 4、变量的注释

**总述：**

变量采用//式注释

**说明：**

对于成员变量和全局变量最好每个都添加相应的注释。局部变量如果是一些特殊情况下使用的变量可以考虑加注释，正常局部变量不需要加相应注释。

### 5、函数中的注释

**总述：**

函数中的注释采用//方案。根据个人喜好自行添加

**说明：**

函数中的注释主要是为了说明一些特殊情况。不需要加特别细。对于一些引导思路的注释也可以添加。

## 五、代码建议

### 1、Qt的宏的使用

**总述：**

使用qt的宏不直接使用qt的信号函数

**说明：**

使用Q\_EMIT 替换 emit

使用Q\_SLOTS 替换 slots

使用 Q\_SIGNALS 替换 signals

使用宏的话可以防止第三方库的关键字冲突

### 2、代码尽量不使用数字常量

**总述：**

涉及物理状态或者含有物理意义的常量，避免直接使用数字，必须用有意义的枚举或const常量来代替。

**说明：**

使用枚举或常量第一是提高可读性，第二是如果有多个地方使用修改更为方便。

### 3、函数的大小限制

**总述：**

一个函数的大小最好在80行以内

**说明：**

一个函数如果过大在看到下文的时候再去找上文会不太方便。设计上是希望控制在一屏内。并且大函数意味着功能多，不利于后期的测试和维护。

### 4、使用前检查变量的有效性

**总述：**

在使用变量的时候最好判断变量是否为空，判断变量是否符合使用条件。

**说明：**

对指针变量的判空和对一些函数返回值的判断可以有效处理程序的崩溃和异常情况下的处理。

### 5、函数传参

**总述：**

对于函数中不应该修改的变量在声明时将变量声明为const，可以的话使用引用的方式传递参数。

**说明：**

使用const可以防止错误的修改。使用引用传递可以减少运算和内存的分配。

### 6、程序编译开到最高

**总述：**

在程序编译的时候将警告开到最高，尽量保证没有异常

**说明：**

对于一个程序最好是0 error 0 warning。有时warning在某些情况下就会导致异常。如果warning无法解决最好也要能确定相应的影响范围。

### 7、如果使用指针的话推荐使用智能指针

### 8、#define保护

**总述：**

所有头文件都应该使用#define来防止头文件被多重包含

**说明：**

为保证唯一性, 头文件的命名应该基于所在项目源代码树的全路径。同时为了符合宏定义的命名规范，名称要全大写。

示例：

#ifndef \_\_MAINWIDGET\_H\_\_

#define \_\_MAINWIDGET\_H\_\_

....

#endif

### 9、声明顺序

**总述：**

将相似的声明放在一起。将public部分放在最前

**说明：**

定义一般应以public:开始，后跟protected:，最后是 private:。

### 8、qss样式要写在qss文件中

**总述：**

控件的qss样式不要用一块写一块，整体统一在一个qss文件中。

**说明：**

将样式写在qss文件中一点是方便管理，而且多个控件使用相同的样式时可以减少相应的代码。并且写的qss中可以使用rcc封装，修改效果可以只修改qss文件不重新编译代码。

### 9、程序中不允许中间return

**总述：**

一个程序的return只能在程序的最后一行。

**说明：**

在程序中间return的话，如果要打印错误日志或一些操作会比较麻烦。在程序中设置状态变量在最终return时整体打印逻辑更清晰，代码也更简洁。

### 10、程序中不允许使用goto

**总述：**

程序中任何地方禁止使用goto。