代码规范 – c++

说明

本文的目的是规范代码书写，使代码具有较好的可读性。
文档中条目除带有[建议]，[提倡]，[不提倡]说明的之外，其他均为**强制要求**。

- 一般的，建议 c++程序员都去阅读并参考一下[google 编程风格指南](http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml)，这是一个更加详细的规范，但略有复杂，本规范对其进行了简化。

本规范为 2012 年”亚视编码规范简易版”的继承和替代版，2014 年 5 月 5 日开始启用。

语言

- 头文件保护
  所有头文件都应该使用#define 防止头文件被多重包含，命名格式为：
  <PROJECT>_<PATH>_<FILE>_H_

  ```
  #ifndef FRPL_CORE_TIME_H_
  #define FRPL_CORE_TIME_H_
  ...
  #endif // FRPL_CORE_TIME_H_
  ```

- 宏定义
  尽量不使用宏定义，用内联函数，const，枚举代替
  ```
  const int cNUM_COUNT = 6;
  ```

- 函数参数
  定义函数时，参数顺序为：输入参数在前，输出参数在后。
  新添加的输入参数，也要置于输出参数之前

- 命名空间
  命名空间使用小写字母，长度应尽量短，可使用缩写
  代码库的命名空间 frpl，其他项目代码的命名空间为"项目名"
  不提倡使用 using（避免污染命名空间，提高编译执行速度）,例如

  ```
  using namespace std;   // bad
  string aa;             // bad
  ```
  而应使用
  ```
  std::string aa;        // good
  ```

格式

- 使用"{"时新起一行。在整个函数只有一行时，不考虑。

- 空格 vs 制表符
  使用空格代替制表符，每次缩进 4 个空格。
  为了保留使用 tab 的编码习惯，可以在 IDE 上设置用空格替换制表符（每次使用 tab 时，IDE 会自动转为 4 个空格）
  - VS2010 设置方式：Tools ->Options ->Text Editor , C/C++ -> Tabs,选择 Insert spaces，设置 Tab size:4
  - Sublime2 设置方式：首选项->设置默认->修改为"tab_size":4, "translate_tabs_to_spaces": true

- 比较操作符，复制操作符“=”、“+=”，算术操作符“+”，“%”，逻辑操作符“&&”、“&”，位域操作符“<<”等双目操作符的前后要加空格
  a += b;
  k = x + y;

- [建议]单目操作符("++"、"--")、函数参数缺省值"="前后有空格。其他单目操作符("!"、"~"、"&")前后无空格

- 逗号,分号只在后面加空格
  void func(int a, int b);

- 以下情况可以省略空格
  1.如一行语句过长，高级运算符前后的空格可以省略
  *y = a*b + c*d;*

  2.数组下标内的运算符前后空格可以省略
  *s = a[i*3+2];*

命名

- 通用命名规则
  必须使用有意义的单词或缩写进行命名，可以使用一些通用的单词缩写（如 msg 等），可以但不提倡以英文单词的前 3~4 个字母作为其缩写；
  命名规则：以大小驼峰式命名法为主,小写字母加下划线为辅
  - 小驼峰法：第一个单字以小写字母开始，第二个单字的首字母大写。例如：firstName、lastName。
  - 大驼峰法：每一个单字的首字母都采用大写字母，例如：FirstName、LastName、CamelCase。

- 文件：大驼峰法，无下划线
  SystemManager.h

- 目录：在能够表意清楚的情况下尽量使用小写字母和缩写。少用下划线。
  src/core/

- 类型(类、结构体、枚举)：大驼峰法，无下划线
  class VideoAcqAVI ...
  struct DetectMsg ...

- 变量：小写字母加下划线、全小写、小驼峰法均可，要在同一文件中保持一致
  std::string table_name;
  std::string tablename;
  std::string tableName;

  常量建议在名称前加 c，全局变量建议在名称前加 g
  const int cUseLimit = 9;
  int gCashBalance;

- 函数
  全局函数、类内普通函数：大驼峰法，无下划线
  bool DetectVehicle()

  类内 get/set/inline 函数：小写字母+_+变量名
  void set_table_name()
  void set_tableName()

```
class VideoAcqAVI()
{
public:
    bool Init();
    void set_tableName();
private:
    std::string tableName;
};
```

- 枚举值、宏命名：全大写+下划线(不建议使用宏)
  #define MY_EXCITING_ENUM_VALUE

- 前缀（类型名、全局函数、链接库）
  暴露在命名空间全局作用域中的类型名、全局函数均以 Fr 开头(即类内函数不属于此范畴)
  动态、静态链接库均以 Fr 开头
  [建议]用于内部使用的类型、函数、库可以使用 Fri 开头

注释

- doxygen 注释风格

- 文件头注释

```
/**
 *Copyright (c),2013, Freative
 *
 *@brief   简述文件完成的主要功能
 *@author  作者列表
 *@version 版本
 *@date    完成日期
 *
 */
```

- 类，函数，变量注释

```
/**
 * @class   类名
 * @brief   简述
 * @author  作者列表
 * @note    细节描述
 */
```

- 函数注释

```
/**
 * @brief xx 函数
 * @param[in] a  参数说明
 * @param[in] b  参数说明
 * @param[in] c  参数说明
 * @param[out] buf  输出结果
 * @return 0：函数执行成功。
 * @return 1：函数执行失败，原因xxx
 */
```

- 类成员函数、成员变量注释

```
/** 成员变量描述 */
int   m_var;
int   m_color;       /**< 颜色变量 */
```

- 枚举类型注释

```
/**   @brief   枚举类型说明   */
enum   AnotherEnum
 {
     V1,   /**< 值描述   */
     V2    /**< 值描述 */
};
```

- 代码注释

对于实现代码中巧妙的、晦涩的、有趣的、重要的地方加以注释

- TODO,HACK,FIXME 注释
  这类注释用于在代码中留下标记，以便日后查找更改。
  为方便搜索，采用如下格式。TODO/HACK/FIXME(姓名或联系方式):注释内容

注释标识	功能
TODO	有一些额外的工作还没有做（但没有 bug）
HACK	采用了丑陋的方式解决了问题
FIXME	有问题尚未解决

  尽量不用 FIXME，而是应 fix 掉这个问题。如果因时间不足解决方法丑陋，请用 HACK 标记
  // TODO(someone@gmail.com):xxx
  // HACK(tom):xxx
  // FIXME(jerry):xxx

windows 特性

- 使用 VS 进行编译时，将警告级别设置为 3 级或更高，并将所有 warning 当作 error 处理

- 很多代码会包含 stdafx.h，为方便代码共享，不建议显式包含此头文件，而改为启用编译器选项 FI 以自动包含
  - 设置方法：工程中移除（但不删除）stdafx.h 及相关文件，cpp 中去掉包含此头文件的代码。右键工程-> Properties -> C/C++ -> Command Line 添加/FI stdafx.h

代码库与项目开发

- 代码库（FRPL）会补充一些额外的规范用于限定该项目的源代码。

- 在其他项目中，也可以补充额外的规范，且为了保证项目组的代码一致性，可以有与本规范不同的约定。这些补充和变化都应在项目文档中注明。

- 对于历史遗留项目，不改变原代码风格