

代码规范

机器人代码会涉及大量矩阵运算、行业性控制逻辑等，因此机器人代码风格应结合机器人控制算法的特征进行规范，而不应一味照搬互联网行业习惯。

1. 总则

最小完备性：功能完备，代码最简，每一个字符（包括空白字符）都有价值。

对人：易读。

对机器：高效。

2. 文件命名

核心思想：具有辨识度，便于键入与结合 tab 键使用。

- ① 首字母**小写**，便于键入。
- ② 多单词使用**下划线**分割，便于 tab。
- ③ 有潜在相似文件夹命名，原始文件夹以下划线结尾，便于拓展。

3. 变量命名

核心思想：具有辨识度，结构紧凑，逻辑合理，便于阅读。

- ① 驼峰命名，**不使用下划线**，从而避免信息密度过低，整体观感混乱。
- ② 除③外一律**首字母小写**，便于键入。
- ③ 仅两种情况允许**首字母大写**：
 - (A) 矩阵，匹配数学含义一目了然。
 - (B) 具有**全局意义**的量：namespace、#define、const 等。其中#define 使用大驼峰，严禁全大写，造成阅读不便
- ④ class、struct 本体定义以**-Class、-Struct 结尾**。其中：
 - struct 用于数据结构，可包含 init、reset 等基本方法，不包含复杂逻辑。
 - class 用于具备逻辑运算功能的封装。
- ④ 固定一些常用缩写词缀：tgt-、act-、est-等前缀，-P、-V、-F 等后缀。

4. 代码结构

核心思想：结构紧凑节约屏幕空间，逻辑合理，便于阅读。

- ① if、for 等即便仅有**单行**函数体也必须加**花空号**，便于阅读与二次拓展，

避免错位。

- ② 函数的左花括号紧跟主体，而不是另起一行，以充分利用屏幕纵向空间的有效行数。
- ③ 任何表达式中，不用无意义空格。空格仅用于计算逻辑分割或对齐，例如：

```
//空格导致计算逻辑松散不清晰
bad = abc * bcde * cdefg + lmn * mnopqr * opqr + sin(xyz + abc * defg);

//空格明晰分割计算逻辑
good = abc * bcde * cdefg + lmn * mnopqr * opqr + sin(xyz + abc * defg);
```

5. 逻辑结构

核心思想：结构逻辑合理，便于人类阅读以及机器效率。

- ① 模块功能封装（面向对象）。
- ② 调用逻辑紧凑（面向过程）。
- ④ 不常用的库不要 include 到 *.h 文件污染全局。可采用 imp 封装在 .cpp 内。
- ⑤ 减少继承、飞针（指针传递）使用，包括传统指针和现代指针在内的 cast、move 语义等，避免造成阅读识别障碍与阅读逻辑混淆。
- ⑥ 避免对象或函数过多层级、多文件分散嵌套，造成阅读障碍。