编程规范

1. 命名规则

类名与函数名（方法名）

采用驼峰格式命名，其中，类名的首字母大写，方法的首字母小写。
驼峰格式：单词与单词之间通过大写首字母区分。
例：
类名：class CalculateArea{

```
        //…
    }
```

类的名字为 CalculateArea，其中首字母大写，不同单词之间，通过单词首字母大写进行区分，这种风格叫做大驼峰式命名法。
函数名：void quickSort(int a[],char type){

```
        //…
    }
```

与类名命名方法相似，不同的是首字母小写，这种风格叫做小驼峰式命名法。
结构命名方法同类。

变量与常量

变量采用单词下划线分割的方式进行命名。
例：

```
int history_data[];
string student_name;
```

常量采用全大写的风格，用以和变量区分。
例：

```
const int MAX_LENGTH = 100;
#define NUMBER 100
```

命名时要根据其含义，用单词命名，如 history_data[]表明历史数据， MAX_LENGTH 表示最大长度，不要采用拼音命名，如 lishi_shujv[],ZUIDA_CHANGDU 是不建议的，这样不利于阅读。命名不要过长，一般不要超过五个单词，此外，可以适当的按照约定习惯缩写一些单词，以减少命名长度，如 NUMBER 简写为 NUM，MAX_LENGTH 简写为 MAX_LEN，message 简化为 msg 等。

全局变量与指针

在声明全局变量时，全局变量开头加 g，用以和普通作用域有限的变量作区分。
例：

```
int g_value;
int main(){
    //…
}
```

声明指针时，指针要加上前缀 ptr
例：

```
int *ptr_value;
```

2 编程风格

（1）编程时，如 for，if，while 等流程控制关键字，后面跟的代码就算只有一行，也要加上

花括号。嵌套换行时每次空出四个空格，不要使用 tab 键空格，这是由于 linux 与 windows 下空格与 tab 规范并不统一。

例：

```
for(int i=0;i<NUM;i++){
    a[i] = i;
}
```

而不要写成下面的格式

```
for(int i=0;i<NUM;i++)
    a[i] = i;
```

（2）每一行只写一条指令，不要写多条指令

例：

```
for(int i=0;i<NUM;i++){
    a[i] = i;
}
```

而不要写成下面的格式

```
for(int i=0;i<NUM;i++) a[i] = i;
```

（3）在有多个双目运算符进行判断时，要写好小括号表示清楚运算顺序。

例：

```
if((i==j)&&(i!=q)){
    //...
}
```

而不要写成

```
if(i==j&&i!=q){
    //...
}
```

（4）当某一行运算过长时，要适当的换行分割，换行不需要空格，因为没有嵌套关系：

例：

```
if((stu_id==cur_id)
  &&(stu_age<MAX_AGE)
  &&(task_num>MAX_NUM)){
    //...
}
```

而不要写成

```
if((stu_id==cur_id)&&(stu_age<MAX_AGE)&&(task_num>MAX_NUM)){
    //...
}
```

3 注释

注释写在函数的上方，按照以下格式

```
/**
 *@author：XXX
 *@name：quickSort()
 *@return: void
 *@function：快速排序
```

```
 *@para：a[]待排序数组，type 排序类型
 *其他要注意的地方
 */
void quickSort(int a[],char type){
                    //…
          }
```

除函数上方外，其他比较重要，或者逻辑复杂的地方也要进行注释以便于理解，包括函数内部，函数调用处等，要根据自己写的实际情况下注释，如果觉得某一块他人可能不好理解，那么一定要进行注释说明。

4. 记录日志

```
/************************************************************
*    文件名：  UsingSpecification.txt
*    作者：    孙霖
*    内容：    用于记录 Git 每次上传改动
*
*
************************************************************/

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
修改时间          操作人        版本          改动
2019/10/15        孙霖          1             文件建立，第一次加入日志。
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

============================================================================
序号          1
时间          2019/10/15
修改者        孙霖
文件操作      添加文件 1UsingSpecification.txt 2ModifyLog.txt
函数操作      无
其他操作
============================================================================
```

6. git 使用规范

```
/************************************************************
*    文件名：  UsingSpecification.txt
*    作者：    孙霖
*    内容：    关于 Git 的使用规范
*
*
************************************************************/

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
修改时间          操作人        版本          改动
2019/10/15        孙霖          1             文件建立，包括 Git 仓库地址、使用注意事项、
上传日志填写
```

++

==========

GitHub 仓库地址

git@github.com:2020RoboMasterUSTCVisionGroup/2020RoboMaster_USTC_Vision_Group.git

https://github.com/2020RoboMasterUSTCVisionGroup/2020RoboMaster_USTC_Vision_Group.git

说明：请还未申请 Github 账号的同学申请 Github 账号后将账号名发送给孙霖并填写入在线文档，以便获取权限以进行 pull 和 push 操作。

==========

一、使用说明

1、本 Github 仓库用于管理 2020RoboMaster_USTC_Vision_Group 即 2020 年 RoboMaster 中国科学技术大学战队视觉组开发源码的管理，请不要用于存放和上传无关文件。

2、与源代码无关的文件将会后续通知放入 NAS 网络文件存储器中。

3、请同学们自行上网查询 Git 的使用方法，同时了解利用 VSCode 对版本控制工具 Git 的支持方法，正确使用 Git 进行代码管理。

4、请同学们遵守文件攥写和编程规范，保证文件的可读性和编码一致性

5、请同学们每次上传时认真填写修改日志，以便版本管理顺利进行。

6、所上传的代码需为提前调试通过并且可以运行。

二、版本控制

1、本 Github 分为 master 和 dev 分支，其中 master 作为稳定发行版本，只用于阶段性的任务完成和 bug 修改；dev 分支用于日常大家的代码编写和开发。

2、每次提交操作时，需认真填写 commit 参数和修改日志文件 ModifyLog.txt。

3、git commit -m "大体操作，如新添、删除文件；新增、删除、更新功能"

4、修改日志文件 ModifyLog.txt 时需认真填写并使用下表格。

==========

序号	1
时间	2019/10/15
修改者	孙霖
文件操作	添加文件 1UsingSpecification.txt 2ModifyLog.txt
	删除文件 1
函数操作	**文件添加函数 Main，用于运行住函数
	**文件删除函数 FunctionDelete
其他操作	添加库 opencv，需进行***操作

==========