**DTC**plusplus **C**oding **G**uidelines

C++开发手册

**© 版权声明(Copyright)**

****

**版权所有 (C) 2019 < Evilrabbit >根据GNU自由文档许可证1.2版或自由软件基金会发布的任何更新版本的条款，允许复制，分发和/或修改本手册（如果您发布时没有修改章节，没有封面文本，也没有封底文本时。许可证的副本包含在标题为“GNU自由文档许可证”的部分中。）**

**Copyright (c) 2019 < Evilrabbit >. It is permissible to copy, distribute and/or modify this manual in accordance with the terms of the GNU Free Documentation License version 1.2 or any updated version published by the Free Software Foundation (if you have not modified the chapter, no cover text, or no back cover text). A copy of the license is included in the section titled "GNU Free Documentation License.)**

  本手册(原版)版权归属原版手册编写团队。本手册遵循GUN所包含的三个协议条款，详情请参见[GPL](https://zh.wikipedia.org/wiki/GNU%E9%80%9A%E7%94%A8%E5%85%AC%E5%85%B1%E8%AE%B8%E5%8F%AF%E8%AF%81)、[LGPL](https://zh.wikipedia.org/wiki/GNU%E5%AE%BD%E9%80%9A%E7%94%A8%E5%85%AC%E5%85%B1%E8%AE%B8%E5%8F%AF%E8%AF%81)、[GFDL](https://zh.wikipedia.org/wiki/GNU%E8%87%AA%E7%94%B1%E6%96%87%E6%A1%A3%E8%AE%B8%E5%8F%AF%E8%AF%81)。保证手册、教程或其它功能、用途的文档的自由: 以确保任何人不管在商业领域还是非商业领域都可以复制和(修改或没有修改并)重新发布这个文档的自由。

**C++开发手册**

****

撰 写  开源团队

地 址  甘肃·酒泉

官 网  <https://blog.yujiay.com/>

版 本  2019 年 12 月第 1 版

字 数  2 k



前 言

《C++开发手册》是全体C++开发者的集体智慧结晶和经验总结，大家经历了多次大规模一线实战的检验及我们不断的更新和完善，系统化地整理成册，反馈给广大开发者。现代软件行业的高速发展对开发者综合素质要求越来越高，不仅是编程的知识点，其他维度的知识点也会影响到软件的最终交付质量。比如：数据库的表结构和索引设计缺陷可能带来软件上的架构缺陷或性能风险；工程结构混乱导致后续维护的困难；没有鉴权的漏洞代码易被黑客攻击等等。所以本手册以C++开发者为中心视角，划分为编程规约、异常日志、单元测试、安全规约、工程结构、MySQL 数据库六个维度，再根据内容特征，细分成若干二级子目录。根据约束力强弱及故障敏感性，规约依次分为【强制】、【推荐】、【参考】三大类。对于规约条目的延伸信息中“说明”对内容做了适当扩展和解释“正例”提倡什么样的编码和实现方式“反例”说明需要提防的雷区，以及真实的错误案例。

本手册的愿景是**以高效承载质量**。现代软件架构都需要协同完成开发，高效协作即降低协同成本，提升协同效率，所谓无规矩不成方圆，无规范则不易协作。众所周知，制订交通法规表面上是要限制行车权，实际上是保障公众的人身安全。试想，如果没有限速，没有红绿灯，谁还敢上路行驶。对软件开发者来说，适当的规范和标准绝不是消灭代码内容的创造性、优雅性，而是限制过度个性化，以一种普遍（即多数人）认可统一约定俗成的方式一起做事，提升协作效率。代码的字里行间流淌的是软件生命中的血液，质量的提升是尽可能少踩坑，杜绝踩重复的坑，切实提升质量意识。

此版本为对外公布版本，可以由全体开发者撰写，我们将交付在GitHub平台供大家协调撰写。

目 录

编程规约

命名风格

1.【强制】代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。

反例:\_name / name\_ / $name / name$ / \_name$ / $name\_

2.【强制】代码中的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。

说明:正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式 也要避免采用。

正例:digitaltechnology / DTClude / Jiuquan 等国际通用的名称，可视同英文； Note [笔记]/ Communication [通讯]

反例:DaZheCommodity [打折] / getPingfenByName() [评分] / int 某变量 = 3;

3.【强制】方法名、参数名、成员变量、局部变量都统一使用lowerCamelCase风格，必须遵守“小骆驼拼写法”形式。

说明:这样做的好处是可以用一连串的描述性单词来表述一个新的含义，因为中间没有空格，可以被视为一个新词（组），而不是一个短语。

正例:localValue / getHttpMessages / inputUserId 等国际通用的名称，可视同英文； Note / Communication

4.【强制】类名使用UpperCamelCase风格，必须遵守“帕斯卡拼写法”（大骆驼拼写法）形式，但以下情形例外:ServerHTTP / IDCard [包含有缩写词的]。

正例:XMLWebServices / TCPUdpDeal / MixedCase / DeletePage

反例:XmlWebServices / TcpUdpDeal / mixedCase / mixedcase / mixed case / mixed\_case / deletePage

5.【强制】常量命名单词c后面跟UpperCamelCase风格的单词、组。

正例: cPlayer / cCompanyName / const int cNumber = 7;

反例:Player / CompanyName / Number

6.【强制】同一常量多命名单词c后面跟UpperCamelCase风格的单词、组，再添加下划线从阿拉伯数字0开始递增，不要嫌名字长。

正例: cPlayer\_0 / cPlayer\_1

反例:cPlayer-0 / cPlayer-1

7.【强制】抽象类命名时使用Abstract开头（直接定义为抽象类也不可省略Abstract开头）；异常类命名使用Exception结尾；测试类命名以他要测试的类的名称开始，以Test结尾。

正例:

1. **class** AbstarctSchool
2. {
3. **public** **void** run(String[]args)
4. {
5. …
6. }
7. };
8. abstract  **class** StudentsTest
9. {
10. **public** **void** run(String[]args)
11. {
12. …
13. }
14. };

8.【强制】中括号是数组类型的一部分，数组定义如下: String[] args ;

反例:使用String args[]的方式来定义。

9.【强制】杜绝完全不规范的缩写，避免望文不知义。

说明:正此类随意缩写严重降低了代码的可阅读性。

反例:AbstractClass“缩写”命名成AbsClass；condition“缩写”命名成condi。

10.【推荐】为了达到代码自解释的目的，任何自定义编程元素在遵循命名规范的同时，使用尽量完整的单词组来表达其意。

正例: 从远程仓库拉取代码的类命名为PullCodeFromRemoteRepository;

反例:变量int a;的随意命名方式。

11.【参考】枚举类型名建议带上Enum后缀，枚举成员名称需要全部大写，单词间用下划线隔开。

说明:枚举其实就是特殊的常量类，且构造方法通常情况下被默认强制是有私有。

正例: 枚举名为ProcessStatusEnum的成员名称：SUCCESS或UNKOWN\_REASON;

12.【强制】各层命名规范:

A) Service/DAO层方法命名规范

1)获取单个对象的方法用get做前缀。

2)获取多个对象的方法用list做前缀。

3)获取统计值的方法用count做前缀。

4)插入的方法用insert做前缀。

5)删除的方法用delete做前缀。

6)修改的方法用update做前缀。

B) 领域模型命名规范

1)数据对象:xxxDT中xxx即为数据表名。

2)数据传输对象: xxxDTO中xxx为业务领域相关的名称。

常量定义

1.【强制】long/longlong或者Long赋初始值时，使用大写的L，不能是小写的l，部分IDE字体容易跟数字1混淆，造成误解。

说明:Long a = 2l;写的是数字的21，还是Long型的2?不容易一眼看出。

2.【强制】不允许任何未经定义的常量直接出现在代码中。

反例:String key = “Id#MYnotecloud” + tradeId;

Cache.put(key,Value);