**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(Foreword)

"C ++ Development Manual" is the collective wisdom and experience of all C ++ developers. We have experienced many large-scale first-line actual combat tests and our continuous updates and improvements. We have systematically organized the book and feedback it to developers. The rapid development of the modern software industry requires more and more comprehensive qualities of developers, not only the knowledge of programming, but also the knowledge of other dimensions will also affect the final quality of software delivery. For example, defects in the table structure and index design of the database may cause architectural defects or performance risks in the software; chaos in the engineering structure leads to difficulties in subsequent maintenance; vulnerable code without authentication is vulnerable to hacking and so on. Therefore, this manual is based on the perspective of C ++ developers. It is divided into six dimensions: programming specifications, exception logs, unit tests, security specifications, engineering structure, and MySQL database. It is further subdivided into several secondary subdirectories based on the characteristics of the content. According to the strength of the binding force and the sensitivity to faults, the statute is divided into three categories: [compulsory], [recommended], and [reference]. The "explanation " in the extended information of the statute entry has appropriately expanded the content and explained what coding and implementation methods "positiveexamples" advocate. "Counterexamples" indicate the minefields that need to be guarded, as well as real error cases.

The practical and efficient bearing quality of this manual. To limit the right of traffic is actually to protect the personal safety of the public. Imagine if there was no speed limit and no traffic lights, who would dare to drive on the road. For software developers, proper specifications and standards are not to eliminate the creation and elegance of code content, but to limit excessive personalization, to work together in a way that is universal (that is, most people) recognize a unified convention, and improve collaboration efficiency . Between the words of the code is the blood in the software life. The improvement of quality is a feasible stepping pit, eliminating repeated pitting, and effectively improving quality awareness.

This version is an externally published version that can be written by all developers. We will deliver it on the GitHub platform for everyone to coordinate and write.

目 录(directory)

编程规约(Programming protocol)

命名风格(Naming style)

1.[compulsory] No naming in the code can begin with an underscore or dollar sign, nor can it end with an underscore or dollar sign.

Counterexample:\_name / name\_ / $name / name$ / \_name$ / $name\_

2. [compulsory] It is strictly forbidden to use a mixture of Pinyin and English in the naming in the code, and it is not allowed to use Chinese directly.

explanation:English spelling and grammar can make the reader understand easily and avoid ambiguity. Note that even pure Pinyin names should be avoided.

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

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

3. [compulsory] Method names, parameter names, member variables, and local variables all use the lowerCamelCase style, and must conform to the "small camel spelling" form.

explanation: The advantage of this is that you can use a series of descriptive words to express a new meaning, because there is no space in the middle, and it can be regarded as a new word (group) instead of a phrase.

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

4. [compulsory]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. [compulsory]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. [compulsory]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. [compulsory]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. [compulsory]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. [compulsory]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.[recommended]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.[ reference]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. [compulsory]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(Constant definition)

1. [compulsory]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. [compulsory]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);