Cpp-Programming-Style-Guideline

Guoning-Chen

2021 年 07 月 19 日

Contents

CHAPTER 1

版权信息

英文名：C++ Programming Style Guidelines

中文名：C++ 编程风格指南

原作者：*Geotechnical Software Services*

原作版本：4.9

原作时间：2011 年 1 月

英文原文：http://geosoft.no/development/cppstyle.html

译者：Guoning-Chen

译者邮箱：cgn1874@163.com

翻译版本：v1.0

Github 仓库：https://github.com/Guoning-Chen/Cpp-Programming-Style-Guideline

在线阅读：https://cpp-programming-style-guideline.readthedocs.io/zh_CN/latest/docs/translator_introduction.html

CHAPTER 2

译者前言

2.1 内容简介

众所周知，对于一名合格的程序员而言，保持良好的代码风格非常重要。好的代码风格能够让别人（也不只是别人，还可能是写完代码 N 天之后的自己）快速地掌握一段代码的逻辑结构，从而高效地进行维护，这对于团队协作开发一个大型项目是非常有利的。另外，良好的代码风格甚至还能够从源头上避免自己无意间造成的一些 bug。

关于代码风格的书或在线文档有很多，已经出版的书籍有《重构-改善既有代码的设计》、《代码整洁之道》等等；很多大公司也有完善的编程规范，例如：Google 开源项目风格指南 (中文版)。虽然这些资料的内容都很详细、很权威，但缺点就是——**太长**（Google 编程规范中文版就长达 5 万字），不太适合 **C++ 初学者**。

我接触到这份指南是在刚开始学习 C++ 的时候，由 MOOC 公开课——北京邮电大学《C++ 程序设计（面向对象进阶）》的老师推荐的。**之所以说这份指南适合初学者，主要有以下原因：**

1. **字数少**：翻译之后不到 8000 字。
2. **格式友好，方便阅读**：这里面其实是一条一条的建议，只有 90 条左右；每条建议还给出了示例和**之所以这样做的原因**（这一点对于记忆很有用！）；另外格式很清晰，方便多次回看。
3. **内容很基础，初学者也能用得到**：学习编程规范的最好方法就是**边学边用**，但是像 Google 编程规范中的很多内容，初学者的日常编程其实是用不到的，而这份规范中的大部分内容都很基础，也很常用；又因为它很短，用到的时候可以很快翻到相应的地方，不至于太影响编程体验。
4. **有利于培养代码规范的意识**：正如作者前两条建议所说——**“只要能够提升可读性，允许采用不同于该指南的做法。”“如果你个人很抵触该指南的某项建议，可以不采纳。”作者一直在强调，重要的是这种下意识为阅读代码的人着想的意识**，而不是某种固定、死板的做法，这一点让我受益很多。

总之，**这篇指南适合以下读者：**

1. **从一开始就想养成良好编程习惯的 C++ 初学者**

   对你来说，这份短小精悍的指南将是一个非常好的选择。

2. **想要从现在开始纠正之前的编程习惯的 C++ 使用者**

   相信会有很多人像我一样，因为畏惧 Google 编程规范之类文档的过于庞杂而不知道从何处开始，这份指南同样适合你们。万事开头难，你完全可以把阅读这份指南当做一个相对轻松的开头，等到培养一定的习惯之后再考虑学习那些更加复杂、也更加完善的编程指南。

2.2 注意事项

1. 下划线标出的单词或句子：代表翻译得不太合适但未找到更好的译法，已在文中用下划线标出；
2. **加粗的下划线的英文单词：这种单词在原文中出现多次，我推测可能是术语，在译者附录**中给出了我自己的理解；
3. 由于 Markdown 表格内的代码行首的空格无法显示，只能用 · 代替，例如下面 `return` 前的两个 · 代表有 2 个空格：

   ```
   int getMaxIterations(){··return 25;}
   ```

4. 如果你发现翻译内容有误或对下划线部分有新的见解，欢迎在issue中指出或通过邮箱联系我，同样欢迎任何其他的意见。

CHAPTER 3

1 简介

该指南包含了 C++ 开发社区中对于 C++ 编程的常用建议。

这些建议的来源包括：一些资源中的现有标准、个人经验、本地需求以及参考资料 [1]~[4]。

我们之所以提出新的指南而不是简单遵循上述内容是有一些原因的。主要是因为上述内容太宽泛，需要建立更具体的规范（尤其是命名规范）。而且该指南采用了独特的排版，在检查项目代码的过程中使用起来要比其它指南方便得多。另外令人有些困惑的是，编程建议经常将代码风格问题和编程技术问题弄混。本文档主要关注代码风格而不包含任何与 C++ 技术相关的建议。想了解更多有关 c++ 编程风格的内容，请参考C++ 编程实践指南。

尽管集成开发环境（IDE）通过权限可视化、代码着色、自动排版等功能可以提高代码可读性，但编程人员从来都不应该依赖这些特性。源代码应该总是被放到比编写代码的 IDE 更重要的位置，尽可能提升其独立于 IDE 的可读性。

3.1 1.1 建议的排版

所有建议按照主题进行了分组，同时标注了序号，便于回顾。

每条建议都采用如下排版：

第 3 部分的内容很重要。编程标准和指南容易引发 ** “宗教战争” **，因此说明建议的使用背景很重要。

> 【译注】宗教战争：原文为 religious wars，个人理解为分不清对或错、无意义的争论，因为每个人都有自己的编程习惯。

3.2 1.2 建议的重要性

该指南中的 **must**、**should** 和 **can** 都有特殊的含义。**must（必须）**代表该建议必须遵循；**should（应该）**代表强烈推荐；**can（可以）**代表一般建议。

CHAPTER 4

2 整体建议

CHAPTER 5

3 命名规范

5.1 3.1 通用命名规范

【译注】setter：目测是指给类中的成员变量赋值的一类成员函数

【译注】通用变量：原文为 Generic variables，我理解为出现在多个函数的参数列表中的同名形参。

5.2 3.2 特殊的命名规范

CHAPTER 6

4 文件

6.1 4.1 源文件

6.2 4.2 包含文件和 include 语句

CHAPTER 7

5 声明

7.1 5.1 类型

7.2 5.2 变量

7.3 5.3 循环

7.4 5.4 条件表达式

7.5 5.5 其他

CHAPTER 8

6 布局和注释

8.1 6.1 布局

【译注】fall through：原意为落空、失败，中文注释时可以写成// 继续执行。

8.2 6.2 空格

8.3 6.3 注释

CHAPTER 9

7 参考资料

[1] Code Complete, Steve McConnell - Microsoft Press

[2] Programming in C++, Rules and Recommendations, M Henricson, e. Nyquist, Ellemtel (Swedish telecom)

[3] Wildfire C++ Programming Style, Keith Gabryelski, Wildfire Communications Inc.

[4] [C++ Coding Standard, Todd Hoff]

[5] [Doxygen documentation system]

CHAPTER 10

译者附录

部分单词没有找到合适的译法，在上面的译文中保留了英文形式，根据我搜集的资料尝试解释如下：

- **side effect**（第 33,42,47 条）：直译是“副作用”，但听起来并不直白。根据维基百科,side effect 是指某个操作、函数或表达式在返回目标值之外，还修改了作用域之外的内容，例如：
  - 修改了非局部变量、静态局部变量、通过引用传递的可变参数
  - 执行 I/O 操作
  - 调用其他会产生 side effect 的函数
- **concept**（第 20,47 条）：根据维基百科，我的理解是：concept 是指**某个类可以执行的操作**。
- **construct**（第 49,57 条）：第 49 条谈到 struct 结构体和第 57 条谈到 do-while 循环时，都提到了 construct 这个单词，第 49 条是说 class 完全可以代替 struct，第 57 条是说完全可以用 while 循环和 for 循环代替 do-while，这两条建议都在强调：减少使用 construct 的数量有助于提高代码可读性。根据维基百科对于 **language construct** 的解释，我认为可以把本文档中的 construct 理解为**代码的布局或结构**，当用 C++ 来实现某种功能时，代码结构越单一、代码结构形式的种类越少，当然就越容易让别人看懂。