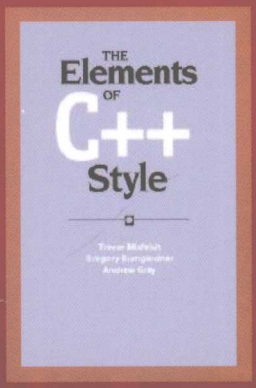
TURING 图灵程序设计丛书 C/C++系列

# The Elements of C++ Style

# C++编程风格（英汉对照）

Trevor Misfeldt
[美] Gregory Bumgardner 著
Andrew Gray

罗小平 译




- 著名软件开发公司的编程规范
- 来自世界级C++专家的经验结晶
- 打造优秀软件团队的利器

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 C/C++系列

# The Elements of C++ Style
# C++编程风格（英汉对照）

Trevor Misfeldt
[美] Gregory Bumgardner 著
Andrew Gray
罗小平 译

人民邮电出版社
北 京

**图书在版编目（CIP）数据**

C++ 编程风格：英汉对照 /（美）米斯菲尔特（Misfeldt, T.），（美）布姆加德纳（Bumgardner, G.），（美）格雷（Gray, A.）著；罗小平译 . —北京：人民邮电出版社，2008.10
（图灵程序设计丛书）
ISBN 978-7-115-17605-9

Ⅰ. C… Ⅱ. ①米…②布…③格…④罗… Ⅲ. C 语言－程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2008）第017418号

## 内 容 提 要

本书是一部久经考验、短小精悍的 C++ 编程规范。给出的 C++ 编码规范和建议主要涉及了格式、命名、文档、编程、包以及泛型等内容，能够帮助广大程序员编写出更易于理解、维护、扩展而且更有效、更专业的 C++ 代码。

本书适用于各层次 C++ 程序员。

图灵程序设计丛书

**C++编程风格（英汉对照）**

◆ 著 [美] Trevor Misfeldt, Gregory Bumgardner, Andrew Gray
译 罗小平
责任编辑 陈兴璐

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：850×1168 1/32
印张：8.25
字数：296千字 2008年10月第1版
印数：1－4 000册 2008年10月北京第1次印刷
著作权合同登记号 图字：01-2007-5286号
ISBN 978-7-115-17605-9/TP

定价：29.00元

读者服务热线：(010)88593802 印装质量热线：(010)67129223
反盗版热线：(010)67171154

# 版 权 声 明

# Preface

As commercial developers of software components, we always strive to have good, consistent style throughout our code. Since source code is usually included in our final products, our users often study our code to learn not just how the components work, but also how to write good software.

This fact ultimately led to the creation of a style guide for Java™ programming, entitled *The Elements of Java Style*.[①] The positive reception to that book, coupled with recurring questions about C++ style issues, resulted in this edition for C++.

If you've read *The Elements of Java Style* (or even if you haven't), much of the advice in this book will probably be familiar. This is deliberate, as many of the programming principles described are timeless and valid across programming languages. However, the content has been reworked and expanded here to address the unique characteristics of the C++ language.

## Audience

We wrote this book for anyone writing C++ code, but especially for programmers who are writing C++ as part of a team. For a team to be effective, everyone must be able to read and understand everyone else's code. Having consistent style conventions is a good first step!

This book is not intended to teach you C++, but rather it focuses on how C++ code can be written in order to maximize its effectiveness. We therefore assume you are already familiar with C++ and object-oriented programming. There are a number of good books about C++ basics; in particular, we recommend *The C++ Programming Language (3rd edition)*[②] and *The Design*

① Al Vermeulen, Jim Shur, EldonMetz, Scott Ambler, Greg Bumgardner, Patrick Thompson and Trevor Misfeldt. *The Elements of Java Style*. (Cambridge, UK: Cambridge University Press, 2000).

② Bjarne Stroustrup. *The C++ Programming Language, Third Edition*. (Reading, Massachusetts: Addison-Wesley, 1997).

# 前　言

作为商业软件组件的开发人员，我们总是努力让自己的代码始终遵循良好、一致的风格。因为源代码也是最终产品的组成部分，所以我们的用户常常会研读代码，以了解组件如何工作，同时学习如何编写好的软件。

这最终导致了Java编程风格指南*The Elements of Java Style*[①]一书的诞生。该书的成功，加上C++领域存在的类似需求，促成了本书的面世。

不管你是否读过《Java编码风格》，我相信本书中的多数建议你可能都很熟悉。其实这些规范基本都是不受时间限制、跨语言且久经考验的。当然，这些内容都已经根据C++语言自身的特点做了修正和扩充。

## 读者对象

本书适合所有编写C++代码的人，尤其是那些团队中的C++程序员。一个团队要想有成效，每个人都必须能够阅读并理解其他人的代码。拥有一致的风格约定将是个良好的开始！

本书无意于教你C++，而是专注于应该如何编写最有效的C++代码。因此，我们假设你已经熟悉C++和面向对象编程。关于C++基础知识有很多优秀的图书，特别是*The C++ Programming Language*（第3版）[②]和

---

① Al Vermeulen, Jim Shur, EldonMetz, Scott Ambler, Greg Bumgardner, Patrick Thompson and Trevor Misfeldt. *The Elements of Java Style*. (Cambridge, UK: Cambridge University Press, 2000).（英汉对照版《Java编程风格》，人民邮电出版社，2008）

② Bjarne Stroustrup. *The C++ Programming Language, Third Edition*. (Reading, Massachusetts: Addison-Wesley, 1997).

*and Evolution of C++*,[①] both by Bjarne Stroustrup, the designer of the C++ language.

① Bjarne Stroustrup. *The Design and Evolution of C++*. (Reading,Massachusetts: Addison-Wesley, 1994).

*The Design and Evolution of C++*[①]，这两本书的作者都是C++语言的设计者Bjarne Stroustrup。

① Bjarne Stroustrup. *The Design and Evolution of C++*. (Reading,Massachusetts: Addison-Wesley, 1994).

# 1

# Introduction

style: 1b. the shadow-producing pin of a sundial.
2c. -the custom or plan followed in spelling, capitalization, punctuation, and typographic arrangement and display.

—*Webster's New Collegiate Dictionary*

The syntax of a programming language tells you what code it is possible to write—what machines will understand. Style tells you what you ought to write—what humans reading the code will understand. Code written with a consistent, simple style is maintainable, robust, and contains fewer bugs. Code written with no regard to style contains more bugs, and may simply be thrown away and rewritten rather than maintained.

Attending to style is particularly important when developing as a team. Consistent style facilitates communication, because it enables team members to read and understand each other's work more easily. In our experience, the value of consistent programming style grows exponentially with the number of people working with the code.

Our favorite style guides are classics: Strunk and White's *The Elements of Style*[①] and Kernighan and Plauger's *The Elements of Programming Style*.[②] These small books work because they are simple: a list of rules, each containing a brief explanation and examples of correct, and sometimes incorrect, use. We followed the same pattern in this book. This simple treatment—a series of rules—enabled us to keep this book short and easy to understand.

① William Strunk, Jr., and E. B. White. *The Elements of Style, Fourth Edition.* (Allyn & Bacon, 2000).

② Brian Kernighan, and P. J. Plauger. *The Elements of Programming Style.* (New York: McGraw-Hill, 1988).

# Contents

# 目　录

# 第1章

# 引　言

风格（style）：1b. 日晷上阴影产生的指针。
2c. 拼写、大小写、标点符号、排版和显示应该遵守的习惯或方法。

——《韦氏新版大学词典》

编程语言的语法告诉你可以写什么样的代码——条件是机器能够阅读并理解。而风格告诉你应该如何写代码——条件是人类可以阅读并理解。按照一致、简洁的风格编写的代码，可维护性好、健壮，包含的Bug更少。而不遵守风格的代码会包含更多Bug，难以维护，往往最终被人们抛弃或重写。

所以，在一个开发团队中，风格非常重要。编码时始终遵守风格，将降低沟通成本，因为团队成员可以轻松阅读并理解彼此的代码。以我们的经验来看，编码风格的重要性会随团队规模呈指数式增长。

备受大家推崇的经典风格指南有Strunk和White的*The Elements of Style*[①]、Kernighan和Plauger的*The Elements of Programming Style*[②]。这两本小册子的最大特点就是简单：全部规则以名单形式给出，每个规则均包含简要的文字说明和例子（多数是正确例子，有时候也给出反例）。本书采用了同样的组织方法，简单将保证篇幅短小和内容易于理解。

① William Strunk, Jr., and E. B. White. *The Elements of Style, Fourth Edition.* (Allyn & Bacon, 2000).

② Brian Kernighan, and P. J. Plauger. *The Elements of Programming Style.* (New York: McGraw-Hill, 1988).

Some of the advice that you read here may seem obvious to you, particularly if you've been writing code for a long time. Others may disagree with some of our specific suggestions about formatting or indentation. What we've tried to do here is distill many decades of experience into an easily accessible set of heuristics that encourage consistent coding practice (and hopefully help you avoid some C++ traps along the way). The idea is to provide a clear standard to follow so programmers can spend their time on solving the problems of their customers instead of worrying about things like naming conventions and formatting.

## Disclaimer

We have dramatically simplified the code samples used in this book to highlight the concepts related to a particular rule. In many cases, these code fragments do not conform to conventions described elsewhere in this book—they lack real documentation and fail to meet certain minimum declarative requirements. Do not treat these fragments as definitive examples of real code!

## Acknowledgments

Books like these are necessarily a team effort. Major contributions came from the original authors of *The Elements of Java Style*: Al Vermeulen, Scott Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur, and Patrick Thompson. Both that book and this one have some roots in "C++ Design, Implementation, and Style Guide," written by Tom Keffer, and the "RogueWave Java Style Guide," and the "Ambysoft Inc. Coding Standards for Java," documents to which Jeremy Smith, Tom Keffer, Wayne Gramlich, Pete Handsman, and Cris Perdue all contributed. We'd also like to thank our former colleagues at Rogue Wave Software, from whom we've learned a lot over the years, and who have thus contributed to this work in both tangible and intangible ways.

Thanks also to the reviewers who provided valuable feedback on drafts of this book, including Ken Baldwin, Brand Hunt, Jim Shur, and Steve Sneller.

This book would certainly never have happened without the help and encouragement of the folks at Cambridge University Press, particularly Lara Zoble, who kept us on track throughout the writing and publication process.

本书建议的一些规则在不少读者（尤其编程时间很长的）看来可能是显而易见、理所当然的。另一些读者可能对我们在程序格式和排版等方面的一些细节问题上的建议有不同看法。没关系，我们的目的是萃取前辈们几十年编程经验的精华，归纳总结为易学习、有启发、较成体系的实践规则。通过这样一个清晰的规则，帮助程序员将主要精力转移到解决客户实际问题上（同时知道在编程时如何避开一些C++陷阱），而不用总是担心命名习惯、代码格式等琐碎事情。

## 声明

在本书中，为了突出与特定规则相关的概念，我们将代码示例尽可能做了简化。在多数情况下，这些代码片段可能并未遵循书中其他地方描述的规则（比如缺少文档、声明不完全等），因此请不要在工作中直接使用这些示例中的代码！

## 致谢

一本图书往往是一个团队共同努力的结晶。本书的主要贡献来自于*The Elements of Java Style*的作者Al Vermeulen、Scott Ambler、Greg Bumgardner、Eldon Metz、Trevor Misfeldt、Jim Shur和Patrick Thompson。另外，*The Elements of Java Style*和本书又根植于Tom Keffer编写的图书*C++ Design, Implementation, and Style Guide*和Jeremy Smith、Tom Keffer、Wayne Gramlich、Pete Handsman及Cris Perdue合编的文档*RogueWave Java Style Guide*、*Ambysoft Inc. Coding Standards for Java*之上。我们还要感谢过去在Rogue Wave软件公司的同事们，从他们那里我们学到了很多东西，因此他们对本书也有无形但切实的贡献。

还要感谢为本书草稿提出了颇具价值意见和建议的审稿人，他们是Ken Baldwin、Brand Hunt、Jim Shur和Steve Sneller。

最后要说的是，如果没有剑桥大学出版社编辑们的帮助和鼓励，本书可能永远不会面世；尤其是Lara Zoble，他跟踪了本书写作和出版的全过程。

# 2

# General Principles

While it is important to write software that performs well, many other issues should concern the professional developer. All *good* software performs well. But *great* software, written with style, is predictable, robust, maintainable, supportable, and extensible.

## *1. Adhere to the Style of the Original*

When modifying existing software, your changes should follow the style of the original code.[①] Do not introduce a new coding style in a modification, and do not attempt to rewrite the old software just to make it match the new style. The use of different styles within a single source file produces code that is more difficult to read and comprehend. Rewriting old code simply to change its style may result in the introduction of costly yet avoidable defects.

## *2. Adhere to the Principle of Least Astonishment*

The *Principle of Least Astonishment suggests* you should avoid doing things that would surprise a user of your software. This implies that the means of interaction and the behavior exhibited by your software must be predictable and consistent,[②] and, if not, the documentation must clearly identify and justify any unusual patterns of use or behavior.

To minimize the chances that a user would encounter something surprising in your software, you should emphasize the following characteristics in the design, implementation, packaging, and documentation of your C++ software:

① Jim Karabatsos. "When does this document apply?" In "Visual Basic Programming Standards." (GUI Computing Ltd., 22 March 1996.)

② George Brackett. "Class 6: Designing for Communication: Layout, Structure, Navigation for Nets and Webs." In "Course T525: Designing Educational Experiences for Networks and Webs." (Harvard Graduate School of Education, 26 August 1999.)