

程序设计教学中学生程序设计风格的养成^{*}

陈立前, 李姗姗, 叶常春

(国防科学技术大学计算机学院, 湖南 长沙 410073)

摘 要: 计算机程序设计课程已成为普通高校本科教学中的公共基础课程之一。目前, 关于该课程的一个较普遍的问题是课程结束后学生的实际程序设计能力依然比较差, 所编写出来的程序的代码质量不高。原因之一是学生没有养成良好的程序设计风格。着重探讨程序设计风格养成的重要性, 以及如何在课程教学中引导学生重视并养成良好的程序设计风格。

关键词: 程序设计; 程序设计风格; 编程规范; 编程规则

中图分类号: G642

文献标志码: A

doi: 10. 3969/j. issn. 1007-130X. 2016. Suppl(1). 012

Programming style training in programming teaching

CHEN Li-qian, LI Shan-shan, YE Chang-chun

(College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: The course of computer programming becomes one of the fundamental courses for undergraduate students and those who do not major in computer science. When the course is completed, however, many students' practical programming abilities are still poor, and the quality of their codes is low. This paper focuses on one of the reasons, that is, the students do not possess a good programming style. We present the importance of training a good programming style and how to help students to improve their programming styles in the course of teaching.

Key words: computer programming; programming style; coding conventions; coding rules

1 引言

计算机程序设计课程不仅是计算机科学与技术专业课程体系中的基础课程, 也已逐步成为国内普通高校非计算专业的公共基础课程之一。计算机程序设计课程的主要目的是使学生掌握使用计算机程序设计语言来实现算法的方法和技能, 培养学生利用计算机进行问题求解的能力。2013 年 12 月, 美国非营利组织 Code. org 发起了“编程一小时(The Hour of Code)”活动, 该活动不仅得到了苹果、谷歌、微软、亚马逊和 Facebook 等工业界科技巨头的大力赞助, 美国总统奥巴马还为此专门录制了视频, 呼吁全民编程。由此可见, 在当今时代, 计

算机程序设计能力的重要性日益彰显。

在程序设计入门阶段, 让学生养成良好的程序设计风格至关重要。就像学生刚开始学习写字时, 首要任务是让学生养成正确的写字姿势和良好的写字习惯。同样地, 程序设计风格是程序员的一种习惯, 好习惯将让学生受益终生。因此, 计算机程序设计课程的重要任务之一是引导学生形成良好的程序设计风格。

2 程序设计风格及其重要性

程序设计风格关注程序员如何进行程序设计, 包括程序员在程序设计过程中所表现出来的特点、习惯、逻辑思路等。高级程序设计语言为程序员提

^{*} 收稿日期: 2015-08-09; 修回日期: 2015-10-16

基金项目: 国家自然科学基金(61202120)

通信地址: 410073 湖南省长沙市国防科学技术大学计算机学院计算机系

Address: College of Computer, National University of Defense Technology, Changsha 410073, Hunan, P. R. China

供了充分的自由度和灵活性,使程序员可以不拘泥于固定的形式来编写程序,只需要遵循程序设计语言的语法规则即可。但是,任何高级程序设计语言都存在引起潜在危险和错误倾向的因素^[1],而越不规范的代码隐含错误或安全缺陷的可能性越大。另外,程序员所编写的程序除了给程序员自己看,还需与团队其他成员交流。如果程序设计风格不好,会给代码的理解带来障碍,增加维护阶段的工作量。正因如此,许多优秀程序员总结出了一些经验性的但非常实用的程序设计风格,形成了一些行之有效的编程规范。这些编程规范,一方面能够减少编程出错的概率,另一方面能够在保证程序正确性的前提下使程序结构清晰、逻辑简明清晰、易读易懂。尤其,目前实际使用的软件的代码规模都比较大,大部分都需要团队开发,团队内使用一致的、良好的程序设计风格,可以使团队内成员更容易读懂和理解其他成员所撰写的代码,提高团队协作效率。

具体而言,程序设计风格涉及的方面很多,包括标识符的命名、代码的布局(如缩进、空格与空行等的使用)、注释的撰写、函数的设计等^[2]。比如,标识符命名应遵循一定的规则,如“匈牙利命名法”。另外,从软件工程的角度来看,标识符的名字应该具有一定的含义,可以从名字看出该标识符背后对应的物理含义。适当地利用空格、空行和换行能够使得程序的逻辑结构更加清晰。早在1983年,Miamia等人^[3]研究发现,缩进可以提高程序员的理解能力。当程序中使用了2~4个空格的缩进时,被测者对程序的理解分数会比对毫无缩进程序的理解分数高出20%~40%。总体而言,良好的程序设计风格能够极大地提高程序代码的质量,包括程序的可读性、可维护性、健壮性和可重用性等。

对于程序设计初学者而言,如果从一开始学习程序设计时就重视程序设计风格的问题,在程序设计实践中不断审视和改进自己的风格,就会逐渐形成一种良好的编程习惯,受益终生。然而,正是因为程序设计风格只是一种规范,而不需要像严格遵守程序语言的语法那样去遵守,许多程序设计初学者对养成良好程序设计风格的重要性认识还不够。这极大地影响了其将来程序设计能力的提高。

3 工业界对于程序设计风格的需求

在工业界,尤其是大型软件的开发,一般都是需要团队协作来完成,团队成员经常需要使用其他

成员所编写的代码。事实上,有调查表明,软件开发人员有很大一部分工作量是维护前任软件开发人员所编写的代码。如果开发人员没有遵循良好的、统一的程序设计风格,所编写代码的可读性会比较差,势必会增加后期的维护成本。鉴于编程规范的重要性,国内外相关组织、行业或企业都制定了相应的软件编程规范。这些编程规范限制了程序员编程的随意性,提高了代码的质量。一般认为,软件开发如果能够完全遵守这些标准,则其程序代码是易读、可靠、可移植和易于维护的。遵循良好的程序设计风格是提高代码质量最有效、最直接的手段,相应检查机制也已成为软件从设计到发布过程中不可或缺的重要环节。尤其,在航空航天、国防、交通、金融等安全攸关领域,工业界对代码质量要求很高,形成了一些严格的编程规范或标准,并且要求领域内的软件必须符合这些规范或标准。比如,DO-178B《航空运输和装备系统软件认证标准》是航空系统软件安全性的国际标准,它要求软件代码必须符合软件编码标准。

国际上,汽车工业软件可靠性联合会 MISRA (The Motor Industry Software Reliability Association)发布了针对汽车工业软件安全性的C语言编程规范 MISRA-C:1998版(包含127条规则)、MISRA-C:2004版(包含141条规则)、MISRA-C:2012版(包含143条规则)^[4]。目前,MISRA-C标准已经被汽车厂商广泛接受并作为产品设计的依据,也成为C语言安全规范的事实标准,同时也被航空、电信和医疗等领域采纳为代码质量的基础。近年来,Google公司也公布了公司内部使用的程序语言编程规范,包括《Google C++编程风格指南》、《Google Java编程风格指南》等^[5]。在国内,航天工业界制定了《GJB5369 航天型号软件C语言安全子集》^[6],已经成为国内航天领域的C语言编程标准,同时也是目前国内军工行业代码规则审查的主要参照依据。相应地,编程规则检查也被越来越多的自动化的工具所支持^[7],如LDRA Testbed、Logiscope RuleChecker、QAC、Polyspace等。

4 教学过程中程序设计风格引导的实施

正因为程序设计风格对程序质量至关重要,在计算机程序设计课程教学过程中,教师需要让学生意识到程序设计风格的重要性,并引导他们养成良好的程序设计风格。具体而言,教师可以从如下几

个方面进行引导:

(1)教师自身须做好表率。在课件、示例程序等教学材料中出现的程序都应遵循良好的程序设计风格。尤其,在课堂上现场演示程序设计过程时,更应该注意程序设计风格的引导。在具体讲解知识点时,需要把相关程序设计风格的讲解融入进去。比如,在讲解循环结构与函数定义时,可提醒学生注意循环体和函数体中语句的缩进风格。可以通过示例的方式,来比较好的和差的程序设计风格之间的差异性。

(2)建立示范代码和不规范代码案例库。让学生阅读经典的规范的代码,建立示范代码库,让学生感受什么样的程序设计风格才是好的程序设计风格;与之对应地,可以建立不规范代码案例库,教师可以把往届学生平时作业或考试中出现的不规范代码样例搜集整理起来,学生可以把自己编写的或看到的有代表性的不规范代码案例添加到不规范代码案例库中。在教学过程中,教师可以适时地、阶段性地对编程规范进行系统性的总结,发放给学生,并让学生自己整理和补充以形成适合自己的编程规则列表。

(3)注重程序设计风格方面的评分和评价。在对学提交提交的代码进行评判时,除了应考虑程序本身功能性的正确,还应考虑程序设计风格方面的评分和评价。对程序设计风格进行评分,能有效激励学生更主动、深入地学习和贯彻良好的程序设计风格。在上机实践环节,学生碰到程序出错来咨询教师时,容易发现:对于初学者而言,很多程序出错的情况可以归结为没有遵守良好的程序设计风格。其实,只要学生把代码进行规范化后,程序的结构和思路立刻清晰了,学生自己就会发现程序出错的原因并知道如何改正。在这种情况下,教师更应该注意程序设计风格方面的引导,而不是直接帮学生纠正错误。

(4)鼓励学生之间互评。在上机实践环节或平时作业批改中,可以让学生之间结对互相点评对方的程序,指出不良的程序设计风格或难以理解的代码片段,这样容易让学生亲身体会到程序设计风格对于可读性的重要性。进一步,可以有选择地让学生把自己的代码通过投影仪展现给其他同学,让其他同学来点评,提出如何改进程序设计风格方面的意见和建议。通过学生之间的互相点评,一方面可以提高学生学习的积极性和主动性,又可以让学互相学习、互相提醒。

(5)利用编程规则检查工具或代码规范化工具。

从目前国内程序设计教学班的设置情况来看,除了小班教学学生人数较少外,大部分教学班的学生人数常常有八九十人,甚至达到数百人。教师如果通过手工方式来对学生的代码进行批注,将耗费大量时间。为了减轻教师的工作量,可以利用编程规则检查工具或代码规范化工具来检查学生的程序设计风格。比如,GNU Indent 开源软件^[8]通过插入和删除空格对 C 语言代码进行格式化,使代码可读性更好,并能把 C 语言代码从一种风格转化为另一种风格。常用的编程规则检查工具还有 CppLint(针对 Google C++ 编程风格指南)、Splint 等,代码规范化工具有 Uncrustify(含图形化界面 UniversalIndentGUI)、PrettyPrinter 等。可把这类自动化的工具推荐给学生,可以让学生感受到什么是规范的代码格式以及如何应用自动化的工具对程序代码进行规范化,增强他们关于程序设计风格的体验^[9]。

前面提到了,程序设计风格的引导必须融入到课堂教学的知识点讲授中。下面以几个具体的示例来说明如何在讲授知识点时引导学生形成相关的良好程序设计风格。由于控制结构是程序设计初学者学习的重点,也是容易在逻辑上出错的地方。下面,主要以 C 语言中的控制结构为例来进行说明。

(1)使用选择结构时防止逻辑遗漏的程序设计风格引导。

在讲授选择结构时,按照知识点顺序,一般都会先讲授单分支 if 选择结构,然后再讲授双分支 if-else 选择结构。但是,这容易给程序设计初学者一个误导,让学生先考虑使用单分支 if 选择结构,当发现还需考虑条件不满足的情形时再来考虑 else 分支怎么写。在这种导向下,初学者很容易由于疏忽而造成逻辑上的遗漏,漏写了 else 分支。但是,如果教师把讲授顺序交换下,先讲双分支 if-else 选择结构,让学生碰到需要做条件判断时,首先考虑使用双分支 if-else 选择结构。只有在确认双分支在某种条件判断下其中一个分支不需要做任何动作时,再考虑使用单分支 if 选择结构。另外,从软件维护者的角度来看,他们希望看到的程序的结构是完整的。完整的结构表明程序员在编程时已经考虑过了各种情况,没有遗漏。实际上,为了防止在使用选择结构时由于疏忽而造成遗漏,MISRA C、国军标 GJB5369-2005 等编程规范中都有对应的强制类型的编程规则,如:

①规则 1:在“if...else if”语句中必须使用 else

分支。

②规则 2: 在 switch 语句中必须有 default 语句。

(2) 使用选择结构及循环结构时防止对所辖语句范围理解错误的程序设计风格引导。

在讲解循环结构时,为了提高代码可读性并防止人为失误,需要强调:在 while、do-while、for 结构的循环体中都应该使用复合语句(即使用 {})。换言之,即便循环体只有一条简单语句,也建议使用大括号,形成复合语句。同样地,对于 if 结构,then 分支的语句也建议是一条复合语句;对于 else 结构,如果后面不是紧跟一条 if 语句,也应该写成复合语句。比如,“if (C) S1;else S2;”建议写成“if (C) {S1;} else {S2;}”。同样地,MISRA C、国军标 GJB5369-2005 等编程规范中都有对应的强制类型的编程规则:

①规则 1: while、do-while 和 for 语句的循环体必须是一条复合语句。

②规则 2: if 结构后面必须紧跟一条复合语句。同样地,else 分支后面如果不是另一条 if 语句则需要是一条复合语句。

在教学上机实践环节中,笔者发现许多学生嫌麻烦不愿意加大括号 {} 形成复合语句,而更偏好于使用缩进。学生常常误以为相邻几条缩进一致的简单语句可看作是一条复合语句,这与程序本身的语义存在差异,所以往往导致程序运行结果与预期不一致。以下是这种情形的示意:

```
for (i=0; i<SIZE; ++i)
{
    buffer[i]=0; /* 即使循环体只有一条简单语句,
                也需要加上大括号 */
}

while (new_data_available )
    process_data (); /* 循环体没有加大括号,不符合
                    规范 */

service_watchdog(); /* 这条语句是后续维护加的,
                    虽然从缩进对齐看起来像是循环体的一部分,
                    但根据实际语义,该语句只有在循环结束后才能执行 */
```

(3) 使用条件判断时防止机器表示理解不到位的程序设计风格引导。

在使用选择结构和循环结构编写科学计算程序时,经常需要对浮点类型变量的值进行条件判断。在程序设计初学者的眼里,浮点数对应到数学上的实数,常误以为浮点运算是精确的。但是,由于浮点数是用有限位的二进制来表示的,而二进制

浮点表示不能精确地表示所有的实数。因此,数学上看似相等的两个实数表示在计算机内部的表示可能不同,此时用“相等”或“不相等”这种精确比较方法可能得出与期望相反的结果。因此,如果在选择结构和循环结构的条件判断中对浮点数进行“相等”或“不相等”比较,很容易出错。但是这种差异,只有从机器表示的角度才能体现出来,而从高级语言层面很难分辨差别。所以,MISRA C、国军标 GJB5369-2005 等编程规范中都有相应的强制类型的编程规则:

①规则 1: 避免对浮点数作“相等”或“不相等”判断。

②规则 2: for 语句的控制表达式中不能包含任何浮点类型的对象。

5 程序设计风格相关实验

本文使用我校开发的 C 程序编程规则检查工具^[10](其界面如图 2 所示),对我校非计算机专业某班级 66 名学生在计算机程序设计课程中作为平时作业提交的 18 道程序设计题的代码开展了分析实验。其中,单个学生提交的代码行数之和最高的约 700 行左右。

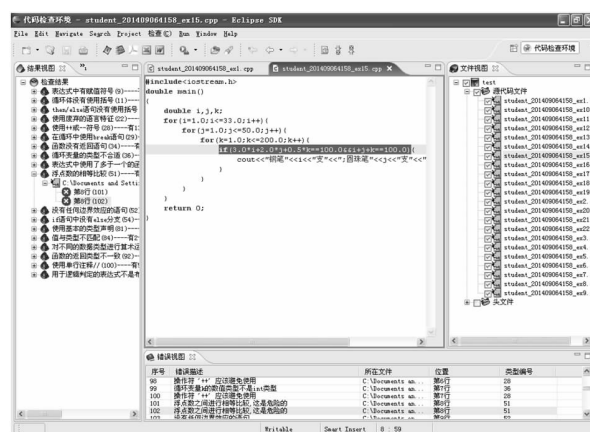


Figure 1 GUI interface of the tool for checking C coding rules

图 1 C 程序编程规则检查工具界面

表 1 给出了实验中平时作业提交的代码中违反情况较多的编程规则的统计情况。

在此基础上,我们研究了学生的期末综合成绩(综合考虑了期末机试成绩、期中机试成绩及平时作业成绩)与平时作业编程规则违反情况之间的关系,结果如图 2 所示。从图 2 所示的整体趋势来看,相对而言,期末综合成绩高的学生在平时作业中编程一般也越规范。

Table 1 Statistical results of the violations of coding rules (partial results)

表 1 编程规则违反情况概略统计报表 (部分结果)

编程规则	文件中违反该规则的条目数
使用++或--符号	2 648
对不同的数据类型进行算术运算	1 146
then/else 语句没有使用括号	984
if 语句中没有 else 分支	719
用于逻辑判断的表达式不是布尔类型的	569
值与类型不匹配	490
循环体没有使用括号	463
数组没有指定大小	120
表达式中使用了多于一个的函数	108
函数的返回类型不一致	78
空语句	44
空的 else 子句	29

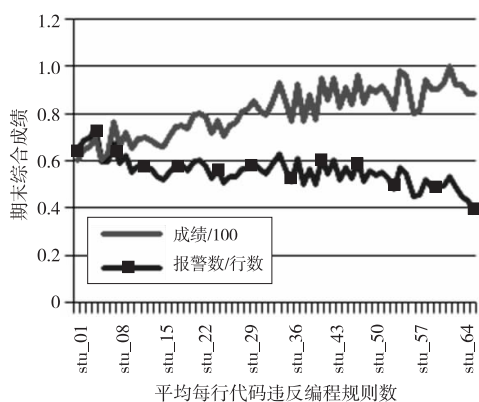


Figure 2 Experimental results showing the relationship between student scores in the final exam and the average number of coding rule violations per line of code in students' homework

图 2 学生平时作业中平均每行代码违反编程规则数与期末综合成绩之间的关系图

6 结束语

良好的程序设计风格能够有效提高编程效率,降低出错率,并增强程序代码的可读性、可维护性、健壮性和可重用性等。引导学生形成良好的程序设计风格,是计算机程序设计课程教学中非常重要的任务。不管是从学生程序设计综合素质的要求,还是从工业界的实际需求来看,强化程序设计风格的养成,对于提高学生的程序设计的实际应用能力和未来职业发展都有着重要意义。

参考文献:

- [1] Howard M, LeBlanc D. Writing secure code (Second edition) [M]. Beijing: China Machine Press, 2005. (in Chinese)

- [2] He Ling-min, Xu Xiang, Lu Hui-juan, et al. Training of C++ programming style[J]. Computer Education, 2011(9): 64-67. (in Chinese)
- [3] Miara R J, Musselman J A, Navarro J A, et al. Program indentation and comprehensibility[J]. Communications of the ACM, 1983, 26(11): 861-867.
- [4] The Motor Industry Software Reliability Association. MISRA-C 2012: Guidelines for the use of the C language in critical systems[EB/OL]. [2015-07-10]. <http://www.misra.org.uk/publications/tabid/57/default.aspx#label-c3>.
- [5] Google C++ style guide[EB/OL]. [2015-07-03]. <http://google-styleguide.googlecode.com/svn/trunk/cppguide.html>.
- [6] Commission of Science, Technology and Industry for National Defense of China; Safe subset of C language for space armament software; GJB5369-2005[S]. Beijing: Commission of Science, Technology and Industry for National Defense of China, 2005. (in Chinese)
- [7] Li Shu-hao, Qi Zhi-chang. Programming rule checking: An underlying practice for software quality assurance[J]. Computer Science, 2003, 30(11): 1. (in Chinese)
- [8] GNU Indent - beautify C code[EB/OL]. [2015-07-03]. <http://www.gnu.org/software/indent/manual/>.
- [9] Zhang Liang-de, Ge Xiang-wei, Liu Dong-sheng. Application of indent software in programming style teaching[J]. Computer Education, 2010(5): 82-84. (in Chinese)
- [10] Li Feng, Wen Yan-jun, Qi Zhi-chang, et al. Research on checking program rules algorithm which based on MYGCC[J]. Computer Engineering & Science, 2012, 34(2): 67-72. (in Chinese)

附中文参考文献:

- [1] Howard M, LeBlanc D. 编写安全的代码(第二版)[M]. 北京:机械工业出版社, 2005.
- [2] 何灵敏, 许翔, 陆慧娟, 等. C++教学中编程习惯的养成[J]. 计算机教育, 2011(9): 67-67.
- [6] 国防科学技术工业委员会. 航天型号软件 C 语言安全子集: 国军标 GJB5369-2005[S]. 北京: 国防科学技术工业委员会, 2005.
- [7] 李书浩, 齐治昌. 程序设计规则检查: 一种保障软件质量的基本方法[J]. 计算机科学, 2003, 30(11): 1.
- [9] 张良德, 葛湘巍, 刘东升. Indent 软件在编程风格教学中的应用[J]. 计算机教育, 2010(5): 82-84.
- [10] 李锋, 文艳军, 齐治昌, 等. 基于 MYGCC 的编程规则检查算法研究[J]. 计算机工程与科学, 2012, 34(2): 67-72.

作者简介:



陈立前(1982-), 男, 湖南茶陵人, 博士, 助理研究员, CCF 会员(E200028451M), 研究方向为高可信软件。
E-mail: lqchen@nudt.edu.cn

CHEN Li-qian, born in 1982, PhD, assistant researcher, CCF member (E200028451M), his research interest includes dependable software.