软件开发中规范性，可读性和维护性原则之浅见

章国英 （第二军医大学）

目前软件市场上，各种通用软件应运而生，但并不畅销。其原因是，所谓"通用"仅是指在某种领域中才可"通用"，而即使在这种领域中各企事单位又千差万别，任何通用软件不经某些修改是无法直接应用的。由此，通用软件在使用过程中还得不断地维护它，要维护势必就得先阅读和弄懂它，因而，仍需花许多的时间和精力。由此可见，在软件开发中应特别强调软件的规范性，可读性和维护性原则，这对微机用户和软件的管理是特别重要的，也是衡量一个软件产品质量的重要标准之一。为此，笔者就如何避免低水平的重复，以及快速地研制高质量的应用软件这个现实问题谈一些粗浅的见解。

一、软件应标准规范，以便于推广

由于微机的种类繁多，软件的标准也不一致，这在软件的交流与推广使用方面造成了一定的困难。因此，遵循软件开发中的规范性原则是提高微机工作效率的有效措施之一。

1．制定统一的编码标准 统一编码是一项相当重要的基础工作，许多研究单位都需要编码，比如，业务部门、专业职称、仪器设备等的编码，均应由有关部门负责组织统一编制，编好之后，印刷成册，供各部门共同享用，以避免各自为阵，而导致编码混乱，大量人力和物力的浪费，以及影响软件的推广与应用。

2．规范文件名和表达式形式 一个系统往往牵涉到许多文件名和表达式，在软件开发初期对文件名和表达式如有一个统一的格式约定，则对今后软件的推广和维护，对文件进行拷贝、列目录、删除等将带来不少方便。

3．统一组织软件的开发 为了充分发挥软件人员的技术力量，应尽量减少重复劳动，统一组织软件的开发工作。具体做法是：（1）加强软件开发的组织与管理工作，推广现有的各种软件成果。（2）对开发的软件产品要进行鉴定，凡通过了鉴定的软件应大力协助其推广与转让。（3）把各单位的软件设计人员组织起来，根据实力，按各个设计专题进行协同攻关，以研制出规范化的标准软件。

二、软件应清晰直观，以有助阅读

当软件开发的规模较大时，常常遇到通读程序有困难，有时读懂某个程序所花的精力往往比重编一个同样的程序要费力得多，即使有些是作者本人编写的程序，在若干年后再来读也很难立即回想起其中的设计思路，这对维护软件带来了许多麻烦。因此，软件的清晰明了，易读易懂在软件开发中显得至关重要。那么，怎样才能提高软件的可读性呢？

1．程序不要过于巧妙 在无必要时不应采用高难的编程技巧，切不可盲目追求节省存贮空间或运行速度，以防程序变得迂回曲折，晦涩难懂。若不得不采用一些技巧时，也必须在程序内部加上恰当的注释。

2．避免嵌套太深和滥用语句 过深的嵌套会使程序变得臃肿难读，一般情况下，嵌套深度以三至四层为限。嵌套太深往往也反映了模糊不清的设计思路。另外，IF…THEN语句之后不应只跟一句GOTO语句，应将一串要执行的语句都写上。并且要合理使用GOTO语句，一般仅用于循环的多重出口，意外情况处理和数据抽象化等情况，滥用则程序易形成"麻花状"，使人难以读懂。

3．程序内部要有恰当的说明 除了特别简单的程序外，所有程序都应带有内部说明，说明一般以文字形式为主，也可酌情辅之以数学式，图表或其它形式。好的说明言简意赅，帮助人们弄懂各部分代码，语句，以及子程序的作用，使用及调用方法，相互关系和设计思想，可提高整个程序的可读性。比如，子程序可以一句注释开头，说明其功能及出口行号，而在RETURN后也可接一注释，说明子程序的入口行号。至于FOR…NEXT语句应采用退格、进格或换行加辅助线等书写方式。

4. 尽量限制公利变量数目，初始化每个变量 每个子程序的公用变量数目应当加以压缩，根据某些专家的研究，子程序的参数最好在三、四个为限，以减少程序设计和调试的困难，同时也便于理解。另外，各个局部变量在使用之前要初始化，即使有些语言（比如BASIC语言）允许省缺初值，但不可过分相信和依赖于具体实现的初值。一般初始化工作以集中在子程序的首部为宜，这有助于阅读和调试。

三、软件应设计合理，以利于维护

可维护性差的软件，当发生意外事件或处理情况稍有变化（诸如机构的变化，指标内容以及项目的变化，还有输入输出格式，内容的变化和软件功能的增加等）时，程序就会出错或者需要大动干戈去修改，这十分不方便。因此，这就要求软件具有灵活处理上述情况的功能，那么，怎样才能开发出既正确又便于修改和维护的好软件呢？

1. 采用模块化结构的设计方法 一个软件系统可由多个模块组成，各模块均是可分别进行处理的一个功能单位，只要模块与外界的接口关系不变，其内容的实现细节乃至修改等不影响其它模块的功能。诚然，若把一件简单的事情分到两个或更多个子程序中去做，必然在需要维护它时增加了受牵连的部分。如果把几件毫无直接联系的事情放到一个子程序里去处理，结果也会损害程序的可读性和维护性。可见，采用模块化结构的设计方法使得各种功能程序更加独立，程序的调试和修改易于进行，且能有效地防止错误在模块间的相互影响，从而提高了软件的维护性。

2. 严格检查和及时修改输入的数据 检查数据就是检测它是否在所规定的取值范围之内。如果输入的数据不符合系统的要求，软件都应显示出错信息，并提请用户重新输入。当一批数据输完后，应有重新显示和修改的功能，因为，这些输入的数据虽是经过严格的检查，但并不一定都是正确的。为了提高入库数据的正确性，应该在接收一组数据后，设置提请用户复审的提示，如用户认为"正确"，方可继续执行后续工作，否则应立即修改送错的数据。

3. 将多变的因素隐藏在个别子程序中 在程序中常有一些随时间推移可能需要修改的变量和结构，例如依赖于机器的因素，具体的数据结构和由于程序使用者方面的原因及其它因素等等，这些容易发生变化的细节应该隐藏在个别的子程序中，使它们对系统的其余部分是透明的。这样若其中某个因素发生变化，则只需对有关的子程序作局部修改，基本上不必触动其他部分，因而简化了程序的维护工作。

4. 力求挽回输入的错误命令 为了防止由于用户的误动作或一时考虑不周发出了不该发出的命令而可能引起的信息丢失或其它严重后果等，软件应设置"挽回"命令，即当发出挽回命令后，则使前一条命令作废，系统恢复到输入前一条命令之前的工作状态，这样可以增加软件的自防御能力，以及提高软件的升级能力，对维护系统是十分有利的。

总之，在软件开发中遵循规范性，可读性和维护性原则所涉及的技术是很广泛的，笔者认为，只要广大软件工作者切实注重这方面的研究和实践，结合具体条件和实践经验，总结出其中的某些规律，必将促进提高软件开发工作的水准，使软件系统更好地按要求发挥其应有的效益，从而促进我国软件产业的发展。

书讯

高级程序设计语言概论

程序设计语言是一种最重要、最有效的工具，至今已有上千种，它们的体裁各异，千姿百态。但是，它们也必然存在一些共同性的概念和规律。掌握这些规律性的东西，对软件管理人员、职业程序员和语言设计者大有裨益。电子科技大学出版社出版的《高级程序设计语言概论》由龚天富和李广星编写。该书以抽象的观点，对比的方法，基于传统程序设计语言讨论共同性的概念和规律，特别是数据类型和控制结构。该书还对函数式，逻辑式和面向对象程序设计作了专门介绍，并简单介绍了形式语义学。通过该书，读者可以提高分析、鉴赏、评价、选择、学习和设计程序语言的能力。