文式程序设计

王厚峰 （华中师范大学计算中心）

摘要

本文阐述了什么是文式程序设计及这一思想出现的背景，介绍了D.E.Knuth教授设计的文式程序设计系统WEB以及进行文式程序设计的几种方法。

从七十年代初开始，程序设计的思想就进入了结构化的时代。在这一思想的影响下，程序设计方法有了很大的发展，程序既要清晰可读，又要运行正确。

然而，最终的程序并不完全令人满意。因为绝大多数程序主要用于机器运行，很少程序用于维护，而几乎没有程序用于其它人阅读；即使有用于人们阅读的程序，机器又并非能够执行。

我们在设计程序的过程中，完全有理由去考虑将可实现性、可维护性及可理解性充分地结合起来。首先，要求解的问题要有对应的程序，最好还要有正确性的验证系统；其次，在设计时要考虑到维护人员不是自己，那么如何才能使维护人员清楚地了解程序的结构和实现呢？或者，设计者本人如何在一定的时间（如三年、五年）之后很快知道自己的设计思想，以便改进呢？

著名的计算机科学家，Standford大学的教授D.E.Knuth提出了一种新的程序设计风格，他称之为“文式程序设计”(literate programming)。他认为应该改变对程序结构的传统看法，即不应该把主要精力放在要计算机做什么，更应该向人类解释我们想要计算机做什么。

为了达到这一目的，D.E.Knuth设计了一个文式程序设计系统WEB，这一系统有效地将传统意义的程序设计和程序文档化(documenting)这二个通常分离的行为结合在一起。WEB能构造二个相关程序：一个是传统意义上的程序，它主要用于计算机执行；另一个则是文式程序，它主要给人类读者阅读。

此外，作为文式程序设计的参与者，在D.E.Knuth看来，他应该具有小品文作者的风格，即，他主要关心如何对程序进行评注和解释及如何保持作品风格的优美性。他还应该具有丰富的词汇，能仔细地选择每一个变量的名字并解释其意义，努力使程序易于理解。因此，每一个概念都应该以最适合人们理解，用形式地或非形式或二者混合的方式引入。

一 文式程序设计系统WEB

文式程序设计的思想在语言和计算机程序上的具体体现是在WEB出现之后。随之，英国学者H.Thimbleby在Unix环境下开发了一个类似于WEB的文式程序设计系统，从而证明了文式程序设计的思想是可行的。

1.1 系统的构成 D.E.Knuth在设计了这一文式程序设计系统之后，把它命名为WEB。他认为，一个复杂的软件事实上可以看成是一个织品(web)，它是由一些简单部分精致编织的，只要理解了各个简单部分及它们之间的关系，我们就理解了这一复杂的软件系统。

WEB系统用Pascal语言实现，而且面向Pascal程序。即，用WEB系统提供的语言（或命令）设计的程序在转化成可执行的代码之前，先要转化成Pascal程序。该系统的构成如图1。

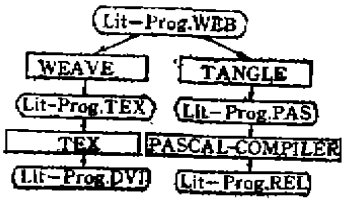

图 1

程序员按照WEB系统的要求编写程序并形成源文件Lit-Prog.WEB，"编织"程序WEAVE将这一程序转化成TEX程序并以相应的文件输出，这一输出文件再经过TEX处理器，从而得到输出文件DVI，经过打印就得到了程序文档。

Lit-Prog.WEB也可经过另一途径由TANGLE处理，将有关程序的代码集结在一起形成Pascal程序的文件Lit-Prog.PAS，该文件再经过PASCAL编译器就形成可执行的文件Lit-Prog.REL。

WEB源程序文件结构比较复杂。它所描述的程序既有正文说明，又有PASCAL代码，还有WEB系统命令。但这一文件通过WEAVE和TEX进行加工，就可将源程序转化成一个易于人们理解的文式程序；另一方面，TANGLE则将源文件转化成易于机器实现的PASCAL程序，但该程序不适应人们理解，其格式也不直观。

从图1和上述分析可知，WEB本身是二种语言的联合体，它包括文档格式语言TEX和程序设计语言PASCAL。当然，也可以用其它的语言来分别取代，如用TROFF而不用TEX，用C语言而不用PASCAL。CWEB正是分别用这二种语言取代和进行适当的改进之后而实现的。

1.2 文式程序设计 文式程序设计的主旨在于所设计的程序不仅要面向计算机，而且要面向人类读者这种双重性。它不限定特殊的格式，也不限制特殊的设计方法。此外，所设计的文式程序并不是一定包含有一个完整的，传统意义上的独立程序。就象WEB本身的含义那样，它是由一个个简单的部分通过一定的逻辑关系构成。通常，一个程序包括若干段，段中可以包括具有某一逻辑意义的程序代码，也可包括变量定义、宏定义和某一部分的规范描述，还可以包括有关这一段内容的非形式的文字注释。

为了说明WEB文式程序设计的特点，下面引入了D.E.Knuth给出的文式程序的一部分。该程序要解决的问题是，对于输入的一对整数M和N，其中M<N，要有序地输出M个随机数，其中的每个随机数都是小于N的正整数而且没有二个随机数是相等的。

D.E.Knuth给出的文式程序由十四段构成，这里只引入前三段，仅说明其风格而不是为了给出一个完整解。

1. Introduction 〈文字注释〉

由于文字注释的内容较长，这里省略不写。本段主要是有关随机数这一问题的文字描述。

2. 〈文字注释〉

```
DEFINE read-terminal(#)≡read(t+y, #)
  {input a value from the terminal}
DEFINE print(#)≡Write(t+y, #)
  {Ouput to the telminal}
DEFINE print-ln(#)≡Write-ln(t+y, #)
  {Output to the terminal and end the line}
```

本段主要是给出三个宏定义。同上段一样这一段也省略了〈文字注释〉的具体描述。

3. Here's an outline of the entire pascal program:

```
PROGRAM sample;
  VAR 〈Global variables 4〉;
〈The random number generation procedure 5〉;
  BEGIN 〈The main program 6〉;
  END
```

这是源程序中第三段的完整描述，其中，第二段的DEFINE和第三段的PROGRAM，VAR，BEGIN和END在原文中均是小写的黑体印刷。

后面的各段其格式与上述一致。每一段

开始，都用“文字注释”对本段的内容进行描述；其后是关于程序代码的有关说明，也可以没有程序代码，如第一段。

我们从第三段可以看出，该段既表示了本段的内容，又表明了与其它段的关系。VAR部分表明，全局变量在第四段说明；而之后的描述表明，随机数产生过程在第五段定义；主程序则在第六段描述。

在WEB系统中进行文式程序设计的另一个特点是全局变量的说明完全是随意的，并不强求将所有全局变量完整地在某一段中说明。从上例的第三段可知，全局变量在第四段说明，而实际上在该程序中，第七、九及十三段也是关于全局变量的说明。这就使得我们在设计过程中，完全可以根据需要来进行。

1.3 源程序的格式及主要控制命令 上例是文式程序，而WEB源程序则是由文式程序加上适当的控制符组成。其结构与文式程序一致，也由段构成。

WEB源程序中的段包括三个部分，其格式如下：

《控制段头符》

《文字注释》

《定义》

《控制段头符》用来表示一个段的开始，它有二种形式：“@*”和“@␣”。前一种形式表示跟在该符号后面的一个字符串以第一个句点为结束点作为该段的标题，而且在进行WEAVE和TEX处理之后，该标题要出现在文档的目录中；后一种形式仅仅用于表示一段的开始。

《文字注释》就是用形式或非形式的方式对本段的内容进行注解。在第一段、第三段所省略的《文字注释》及第三段开始的“Here's an outline of entire pascal program”都是对本段内容的解释。这不同于一般高级语言中的注释语句，因为WEB把《文字注释》本身当作是一种语言成份。但WEB本身也提供了一种类似于高级语言中注释语句的形式。它可以对《文字注释》中的内容和概念作有加亮别，这是用一对双线中括号“[[”及“]]”来表示的。

《定义》则又包括如下几种不同形式：

·宏定义。在文式程序中，宏定义由“DEF INE”声明（见例子的第二段）。但在WEB源程序中，则可利用“@d”符“DEFINE”取代来表示宏定义。

·代码定义。代码定义表示传统意义上高级语言的一个程序。在例子的第三段除《文字描述》部分之外的就是这一情况。在形成WEB源程序时，往往要在代码之前加上控制符“@p”。

·描述定义。这里所说的描述定义是指有关高级语言程序的描述。如例子的第三段中的尖括号部分。对这种形式，在WEB中用控制符“@”加以说明。例：@〈Global variable 4@〉@；

·内容定义。在例子第三段的尖括号部分只是一般性的描述，那么它们具体的描述是什么呢？这还得查看第四、五和六段，这种对应的具体描述我们暂称“内容定义”。在第四段中的内容定义（省略不感兴趣的部分）是：

```
〈Globale variable 4〉≡
        M:integer;
        N:integer;
```

在WEB源程序中，对〈Globale variable 4〉≡就要变成：

@〈Globale variable 4@〉=

关于段的构成，并不一定要上述三个部分都同时出现。事实上，除《段头控制符》不得省略外，其它部分均可略去。

此外，在WEB中还有几个用于单词索引的控制符“@^”、“@.”和“@1”。所列举的这些控制符也只是WEB的主要控制符。

1.4 二种输出处理

·Pascal程序的输出。通过TANGLE对WEB源文件处理，删除《段头控制符》、《文字注释》以及其它控制符；将宏定义转换成Pascal的内部函数或过程；顺序地将《内容定义》代入到《描述定义》部分；对应于文式程序，在Pascal程序的相应地方增加段开始和段结束注释如{3:}和{:3}，以便调试修改。经过这一转换，就得到了相应的pascal程序。

·程序文档输出。通过WEAVE进行处理而得到的以TEX为后缀的文件实际上是一个更难读的文件，但这一文件能使处理器TEX将WEB源程序形成一个结构化的程序即文式程序。实际上，输出文档包括三个部分：

1)一个目录表。它使得读者能很快对文式程序有一个总体了解，也便于查询。目录项是由"@*"标识的段对应的标题及该段的序号。

2)文式程序。文式程序实际上是程序设计者所设计的直接结果。在设计好后，再由其它工作人员或设计者本人增加适当的控制符才形成WEB源程序。经过WEAVE和TEX处理，可使得格式更加优美。输出时，编号是自动生成的。此外，"@*"控制符能控制自动换页，即它所标识的段为某一页的起始段。

3)一个参照索引表。索引表中包括变量出现的段号，但如果变量标识符长度为1，则只索引其变量被说明的那一段号，而不索引引用这一变量的段号。除变量外，对感兴趣的名字也可索引。索引表的最后是对<内容定义>所在的段进行索引。

二 文式程序设计的不同方法

提出文式程序思想的主要目的是要提高程序的可读性和可理解性。那么，对于程序设计者而言，在进行文式程序设计的时候，就应该很自然地把握问题，使得它以更符合人们思维的方式得以解决。

2.1 结构程序设计方法 结构程序设计方法是在著名的计算机科学家Dijkstra和Wirth等人的倡导下形成的。该方法导要构造结构清晰的程序，最具代表的算是"自顶向下"的思想。

我们知道，自顶向下的程序设计思想就是先对整个问题进行全局描述，逐步地将这一大的问题分解成一系列的小问题，然后各个击破，从而获得整个问题的求解。这也是通常的逐步求精。我们所举的例子就采用这一思想。另一方面，自底向上的思想则是先定义最基本的过程和数据结构，逐步地建立较大的、功能更加完善的子程序，直到整个问题求解。自顶向下的思想适应于对整个程序评注，它能使读者强烈意识到下一步将要做什么。这对于理解者来说是有利的。然而，对设计者而言，这就要求他保持大量的、清醒的解题方案，而且下一步应该做什么，这对于设计者往往是难决。一旦进入设计角色，就希望他一直设计下去，直到每个小问题都获解为止。但自底向上的方法往往可使设计者较顺利进行。一旦有了基本的过程和结构，就可构造更高级的子程序，但这却使读者很难在开始就把握程序的总体组织。

自顶向下和自底向上各有利弊，WEB系统并不强求设计者采用什么方式。D. E. Knuth在设计文式程序时，主张采取"意识流"的思想。即该采用自顶向下思想的地方就用自顶向下的方式设计，该用自底向上方式的时候就不要勉强地用自顶向下设计。但从他所设计的几个例子来看，都采用了自顶向下的思想。

在Knuth看来，程序设计是一门艺术，它完全是一种个人的活动。M. Jackson在设计文式程序时，采用了他自己所倡导的方法Jackson方法。该方法是面向数据结构的。他编写了一个有关雇客和帐单表的问题的文式程序。但刚开始阅读这一程序时，并非很好理解。我们用D. Wall的话说"…在理论上，这一方法应该使所设计的程序很容易理解，也很容易修改，因为它将程序文本与导出其文本的原理紧密地联结起来了。但实际上，我不得不抱憾地说，这使得程序更加模糊。甚至在阅读他所写的程序之前，我曾仔细阅读过他的书，但仍然难以清晰地了解他的程序的结构，直到…"。

2.2 联合验证并基于抽象数据类型的方法 在程序设计方法学研究中，与结构程序设计方法并行发展的另一方法是以Floyd和Hoare等人为代表的"程序正确性证明"方法。但这一方法一开始就受到了不少人的反对。因为写好程序再进行验证本身是一种

"马后炮"的事。许多人认为证明应该与编程联合起来。Gries在他的"程序设计科学"中就持这种观点。

抽象数据类型则是近十年来发展的一种极为重要的程序设计方法。通过运用它们，程序员能在更高，更抽象的级别上发挥抽象思维的才智，降低程序的复杂性。该方法是将"运算抽象"和"控制抽象"统一于一体的抽象方法，它使得现代数学理论更好地运用于程序设计。

Gries在进行文式程序设计时，所倡导的就是既要联合程序验证，又要基于抽象数据类型这一思想。

一个数据类型的规范列出了该类型所支持的所有运算以及运算产生的效果。读者只需了解抽象特征而不必知道实现细节，这是实现者的事。这使得读者在程序级上更易理解。

在文式程序设计中，将程序设计和验证同时进行可提高程序的正确性，因为程序设计者采用了二种不同的形式机制，即逻辑和代码。同时给程序提供形式验证可有利于程序的文档化。但不会增加程序的可读性，甚至有可能降低。

为了说明这一思想的可行性，Gries也用文式程序设计的风格设计了一个解决例子中所要解的Kunth随机数问题的文式程序。

文式程序在很大程度上提高了程序的可读性和可理解性。结构程序设计的思想能使文式程序结构更加清晰。随着可读性的增强，我们也可把重点转向程序的正确性和数学抽象性方面，使得程序既易理解，又能进一步保证正确性。当然，程序设计的方法仍在发展之中，把它们有效地运用到文式程序设计中来是可能的。

文式程序设计是D.E.Knuth于1984年最早提出来的，他认为"……一个有经验的系统程序员，想得到最好的软件产品，同时需要二种语言，一个是排版语言，用于文档的排版，另一个是程序设计语言，用于程序构造。…。通过结合这二种语言，[illegible]们能够形成一种新的程序设计风格，使我们理解复杂软件结构的能力得到最大限度地发挥。…"。Knuth首先运用这一思想到他所设计的WEB系统，他用文式程序设计的思想设计了WEAVE，WEB，TANGLE.WEB.和TEX.WEB，然后设计了一个TANGLE.PAS，先将TANGLE.PAS编译成可执行的文件，再通过对TANGLE.WEB处理来验证其自身的正确性。又通过正确的TANGLE将WEAVE和TEX转化成可执行的程序。因而，他没有花多长的时间就实现了WEB首先的版本。J.Bentley只花了二个晚上的时间就读完了TEX的文档，他赞美道："…就象一个建筑学生将花一个下午的时间赞美F.L.Wright的一个建筑一样，同样的原因，对Knuth的工作，将大任务分解成子程序，优美的算法和数据结构以及其编码风格使得它成为功能很强的，可移植而又可维护的系统，我十分赞叹"

随着文档化技术的日益成熟，文式程序设计将会越来越受到重视，那么，从时间上讲，是否有必要去编制文式程序呢？在通过比较之后发现，这完全必要。从计算机来看，它要处理大量语句，但Kunth教授发现，同TANGLE处理一个WEB文件所花的时间并不大于编译器所花的时间，即使相等，也只相当于运行二次PASCAL编译器。同样，WEAVE也不花费多少时间。但唯一例外的是进行TEX处理时要多花一些时间，一页需要一秒钟，六十页的文档就需要一分钟。但这对于以后的阅读十分容易，实际上还是节约了时间。另一方面，从WEB来看，写和调试一个WEB程序并不多于写和调试一个Pascal程序所要的时间。

参考文献

1. D.E.Knuth Literate Programming : The computer Journal Vol.27,No.2 1984
2. H. Thimbleby Experiences of 'Literate Programming' using CWEB: The Computer Journal Vol. 27, No.2,1986
3. J.Bentley Programming Pearls :ACM, Vol. 29 No.5. No.6, 1986
4. J.Bentley Programming Pearls :ACM Vol. 30 No.4, 1987
5. Van Wyk Literate Programming :ACM Vol. 30 No.7, No.12, 1987
6. Van Wyk Literate Programming :ACM Vol. 31 No.8, 1988
7. 何克清编著 《计算机软件工程学》 武汉大学出版社 1983
8. 陈火旺等编著 《程序设计方法学基础》 湖南科学技术出版社 1987