程序设计技巧

中国科学院成都计算机应用研究所情报室

程 序 设 计 技 巧

张世录　程国忠　著

孟晓玲　杨明芳　编

中国科学院成都计算机应用研究所情报室

一九八八年十二月

序　　言

本书是以南充师范院微机软件班、夜大微机班的程序设计技巧讲义为兰本，结合我82年给核工业部四川某研究所讲授的有关程序技巧方面的内容编缀成一本教材。

本书中部份例子是我和我的老同事多年编写程序的经验和教训，另外一部份例子选自一些教科书。我们绝对没有评论教科书的作者们任何程序中的缺点的意图，相反地万分感谢这些作者的开创性工作。我们仅希望读者们能通过仔细分析，对比书中程序而得到教益，

毫无疑问，书中所推荐的一些好程序，随着计算机科学的日新月异的发展，随着时间的推移，将变成坏的范例。我们热忱希望同行们给予斧正。

我的老同事，成都计算机应用研究所的付研究员刘绍中同志给予本书不少中肯的指正，在这里致以衷心感谢。

编　者

年　月　日

目　录

绪　论

我们知道，计算机科学是研究计算技术的科学。计算技术包括硬件技术、软件技术和软件固化技术。

所谓软件是指存放在某种外部永久性介质中，至少能被一类计算机直接或间接识别，具有一个或多个功能的符号和数字的集合。软件分为系统软件和应用软件两类，它们都由一个或多个相对独立的程序组成。一个软件相当于一部机器，程序则类似部件。若程序中尚含子程序，则子程序相当于零件。不过软件中程序比机器中的部件独立性大。研制软件时，绝大部份时间用来编写程序，调试程序，即使软件维护阶段也总是和程序打交道，因而程序设计是软件技术中的核心。

这本书只介绍研制应用软件所涉及的程序设计技巧。

现今的程序多用高级语言编写，用汇编语言的不多，直接使用机器语言的更少。本书只以高级语言为例介绍程序设计的有关技巧，但其基本方法和原则也可为用其它语言设计程序时借鉴。

研制一个软件大约分为以下七个阶段：

1．建立物理模型；

2．建立数学模型，选择计算方法；

3．选择语言，进行程序设计；

4．录入程序，调试程序；

5．实际运行；

6．提交用户使用；

7．进行软件维护。

上述七步中，物理模型通常由用户提供，3、4、5三步难以断然分开。

软件以及它所含的程序的编制不是软任务而是硬课题，软件属于工程，因此本书所谈的一切技巧都围绕生产周期、经济效益以及用户需求在旋转。

评价一个软件通常看以下四个方面的优劣。

1．正确性

2．可读性、可移植性，通用性；

3．透明度；

4．效率。

评价一个程序好坏也用上述四条。

显然第一条是前提。正确性都无法保证的程序，可读性、可移植性、通用性以及透明度和效率更无从谈起。

可读性、可移植性、通用性与算法有关，与程序结构、数据结构甚至语句结构有关。通常良结构程序易读、易移植且通用性较强。

所谓良结构程序是指层次分明、结构清晰、功能突出、通用性强、易读、易改、易移植

的程序。

通用性，可读性，…，可移植性都具有鲜明的相对性，没有客观尺度。它主要取决于程序的使用范围，所涉及的算法的难易以及编制人员、使用人员的知识面和软件知识的多寡。

透明度是指使用该软件时，用户需要对该软件或程序的内部结构以及所涉及的数学物理知识所了解的程度。需要知道的愈少，透明度愈高。这一点与有些科学和日常生活中所谈的透明度的概念不同，甚至恰好相反。

在计算机软件已广泛地用于各阶层各领域的今天，不少使用软件的用户基本上不懂编程知识，他们对一个软件或一个程序本身的构造并不感兴趣，他们最关心的是功能、效率和使用方法。功能愈强、效率愈高、使用愈方便的软件或程序用户愈欢迎。

效率通常指运算量。对于解决同一问题，所需运算量愈少，其软件的效率愈高。效率有时也指内存开销。此时，就解决同一问题的程序讲，内存开销愈小，效率愈高。两者皆好的程序也有，但是不多。

效率高的程序称为高效率程序。高效率程序通常和新算法有关。但是即使使用同一算法，有时也能编出效率大大相异的程序，这就取决于技巧了。

纯用户只关心1、3、4。软件工作者则对上述四方面都应重视。

一切软件工作者，每时每刻都不应忘记，软件是商品，因而设计程序时应为用户着想，为经济效益着想，以实现用户要求，保证诸程序正确，功能完善、使用方便、效率高为已任。

似乎这里未提及“良结构”程序。实际上经济效益四字已和它挂上了钩。原因是良结构程序结构清晰、功能突出、容易编写、不易出错，即使有错也容易修改，因而生产周期短。再加上良结构程序总是由若干便于拆卸、便于装配的模块（类似标准件）组成，很容易相互借用，更有利于降低成本、缩短周期、提高效益。更何况高效率程序通常是把良结构程序优化后才能获得的，因而本书丝毫未轻视良结构程序，相反，如何编写良结构程序是本书重点。

编写高效率程序需要一定技巧，而且不少高效率程序往往和新算法有关，因而编制和调试都较难，也难以阅读。所谓高效率，若是指运算量，则是成倍甚至成量级降低运算量所获得的效率，决不是指少几次乘除法或少调用几次函数所得到的小利。贪蝇头之利而延长编制时间，并给软件维护带来不便的作法应该舍弃。就内存开销讲，其含义也类似。

作为工程，软件或程序中的一切技巧都具有明显的实用性，只能立足和生根上述四方面。

请看下面用FORTRAN语言所编写的程序：

```
      DO 14 I= 1 , N
         DO 14 J= 1 , N
14       V ( I, J ) =(I/J) * ( J/I )
```

初看起来这段程序编得非常绝。程序编制者吃透了FORTRAN语言中整数相除其商去尾取整，巧妙地利用了当J＞I时I/J为零，当I＞J时J/I为零，即只要I≠J，（I/J）*（J/I）始终为零，只有I=J时，（I/J）*（J/I）才为1，从而生成一个n阶单位矩阵。

这个程序真的巧吗？答案是否定的。

生成一个单位矩阵这样简单的事，居然要人看完程序且经过一段时间分析思考后才能明白，其用心就让人费解。进一步分析一下上述程序的工作量，人们会发现，如此简单的事竟用了$3n^2$次乘（除）法！这些运算必要吗？请看下面仍用FORTRAN语言编写的程序：

```
      DO 20 I= 1 , N
```

```
      DO 10 J=1, N
10      V(I, J)=0.0
20    V(I, I)=1.0
```

一眼便知上面程序的功能是生成一个几阶单位矩阵。为达到生成一个单位矩阵这一目的只用了n^2+n次简单赋值（数据传递）。

若使用PASCAL语言或BASIC语言，并利用这两种语言中任何没有赋过值的变量（包括下标变量）其值自然为零，则程序更简单，计算量更小。

PASCAL程序为：

```
FOR I:=1 TO N DO V(I, I):=1.0;
```

BASIC程序为：

```
10 FOR I=1 TO N
20 V(I, I)=1.0
30 NEXT I
```

第一个程序和后三个程序比较何优何劣不言而喻。

实际上编写一些玄乎其玄的程序，故意卖弄技巧，迷惑用户，是程序设计的忌讳。它类似我国功夫中的“花拳绣腿”。应提倡使用简明实用，直截了当的语言编写程序，尽量使功能和意图一目了然。

此外还可看出语句少的程序不一定运算量就小，就好读。

一个程序应是一个“标准件”，其功能应易于识别，应具有一定通用性。为此编写程序时所用语句和符号应尽量符合习惯。勿将各段程序“铆接”甚至“焊接”在一块。

例如对石油测井曲线作数值分析处理时，常对一些点作复杂岩性分析，此时要反复解三阶线性代数方程组（最坏情况是对一个点解五次三阶代数方程组）：

$$\left.\begin{aligned} y_1v_1+y_2v_2+y_3v_3&=y\\ x_1v_1+x_2v_2+x_3v_3&=x\\ v_1+v_2+v_3&=1 \end{aligned}\right\}\qquad(1)$$

（1）式中x，y从曲线取出的测量值，x_1，y_1及x_2，y_2以及x_3和y_3都是常数，前者代表水的两种物理特征值，后者表示两种岩石相应的物理特征值。

国外一些资料和国内石油测井教科书给出了解方程组（1）的计算公式和求解过程，所提供的软件中相应部份程序也按它编写，其程序为（省掉了专业注释）：

$$D_1=(X_2-X_3)*(Y_1-Y_3)-(Y_2-Y_3)*(X_1-X_2)$$
$$B_1=(X_2-X_3)/D_1$$
$$A_1=B_1*(Y_3-Y_2)/(X_2-X_3)$$
$$C_1=-(A_1*X_2+B_1*Y_2)$$
$$D_2=(X_1-X_3)*(Y_2-Y_3)-(X_2-X_3)*(Y_1-Y_3)$$
$$B_2=(X_1-X_3)/D_2$$
$$A_2=B_2(Y_3-Y_1)/X_1-X_3)$$
$$C_2=-(A_2*X_1+B_2*Y_1)$$
$$V_1=A_1*X+B_1*Y+C_1$$
$$V_2=A_2*X+B_2*Y+C_2$$

$$V_3 = 1 - V_2 - V_1$$

若不看注释，会使阅读程序者象丈二金刚，摸不着头脑；看了注释不看（1）式也同样茫然；两者都看后，若不经较长时间仔细推敲仍难明所以。原因何在？原因在于编程者根本未考虑他在编程序，而只是机械地按公式推导者（他也只求得出一个较为对称的格式）所推出的公式进行翻译，因而程序相当特殊，仅能解（1）式（真焊得牢），再加上用了8个信手而来的中间变量，结果导致程序非常难读。

这段程序效率是否高呢？答案也不肯定。解一个三阶线性代数方程组用了十八次乘除法十八个变量，其计算量和正交变换法相近，内存开销尚无一方法有它大，因而效率偏低。这些尚属其次，关键问题在于方程组（1）虽非病态方程，但稳定性并不很好，方程（1）和方程（2）的主元素都不在对角线上，用上述丝毫不考虑误差的算法求解，自然误差不容忽视。由于进行处理的曲线有时是用数字化仪器仿的再生曲线，此时后果更难以预料。由于作复杂岩性分析需要一定专业知识，这里不作详细介绍。

若选择一个常用解方程组的算法，譬如高斯列主元消去法（略，以后介绍）问题则可得到较好解决。

由此看来，编写程序时，软件人员不要盲目生搬工程技术人员所给定的公式，应该用计算数学的观点分析和处理计算公式。否则不是效率不高就是精度不够，或者两者都差。

是否通用程序就一定比专用程序好呢？答案也是否定的，理由是，通用和专用本身就是相对的，万能程序是没有的。更主要的是，不少专用程序，由于考虑了特定环境里的特定条件，效率比通用程序高得多，有时不用专用程序就无法解决问题。用户是对效率和效果感兴趣的。正因为如此，通用程序的价值往往不高，而专用程序常常要价惊人，甚至密而不宣。

例如要从一N段环形链条中取出K段来，使其质量尽可能接近M。假设合乎要求的K段链条肯定是连续的。

对于这个问题我们如何编写程序呢？

在编写这个程序之前我们先考察1984年中学生BASIC程序设计竞赛的一道考题：小明有一只最多能装10斤的网袋。这里有白菜5斤，猪肉2斤，鱼3.5斤，酱油连瓶重1.7斤，白糖1斤，土豆5.1斤（不可分开），请设计一个程序使小明的网袋装的重量最重。

竞赛委员会给的答案是：

```
10 G1 = 0
20 FOR A = 0 TO 5 STEP 5
30 FOR B = 0 TO 2 STEP 2
40 FOR C = 0 TO 3.5 STEP 3.5
50 FOR D = 0 TO 1.7 STEP 1.7
60 FOR E = 0 TO 1
70 FOR F = 0 TO 5.1 STEP 5.1
80 G = A+B+C+D+E+F
90 IF G>10 THEN 120
100 IF G1>G THEN 120
110 G1=G : A1=A : B1=B : C1=C : D1=D : E1=E : F1=F
120 NEXT F
```

```
130 NEXT E
140 NEXT D
150 NEXT C
160 NEXT B
170 NEXT A
180 LPRINT "A=";A1,"B=";B1, "C=";C1,"D=";D1,"E=";E1,"F=";F1
190 LPRINT "G=";G1
200 END
```

A= 0　　B= 2　　C= 0　　D=1.7　　E= 1
F=5.1

G=9.799999

若用N个变量代替A、B、C、D、E、F，用M代替10，则上面程序从逻辑上讲，是可以解决前面提出的问题的。不过有两点麻烦来了，第一，当N较大时，循环重数较多，编译程序（或解释程序）承受不了；第二，上面程序的计算量O（$N\cdot 2^N$）次加法，不要说N很大，当N=30时，$N\cdot 2^N=3\cdot 10^{10}$，这虽不是一个很大的天文数字，但通常的计算机将用不少时间才能得到结果。实际上就任何程序讲，只要计算量为0（K^N），K>1，它就是一个无效程序。

如果我们利用了链条是环形的，而且合乎要求的K段链肯定是连续的这两个特定条件，编出专用程序，问题就不一样了。

```
10 INPUT "N="; N
20 LPRINT "N="; N
30 INPUT "M="; M
40 LPRINT "M="; M
50 DIM A(2*N)
60 FOR I=1 TO N
70 READ A(I):A(I+N)=A(I)
80 NEXT I
90 S=0:S1=0
100 FOR I=1 TO N
110 S=S+A(I)
120 NEXT I
130 IF S<=M THEN P=1:Q=N:GOTO 220
140 FOR I=1 TO N
150 S=0
160 FOR J=I TO I+N-1
170 S=S+A(J)
180 IF S>M THEN 210
190 IF S>S1 THEN S1=S:P=I:Q=J
```

```
200 NEXT J
210 NEXT I
220 FOR I=P TO Q
230 I0=I
240 IF I>N THEN I0=I-N
250 LPRINT I0, A(I)
260 NEXT I
270 DATA 1, 1.7, 2, 3.5, 5, 5.1
280 END
```

```
N= 6
M=10
 6          5.1
 1           1
 2          1.7
 3           2
```

这个程序利用A(I+N)=A(I)，即A(N+1)=A(1)，A(N+2)=A(2)，…，A(2N)=A(N)，巧妙地模拟出了一个环形链条。该程序只用了两重循环，不会出现循环重数超界；它的计算量为O(N²)，能为任何软件工作者和用户接受。

在本章的末尾，我们涉及到了高效率程序。

无论是软件工作者还是用户都追求高效率程序。但由于高效率程序往往和新算法有关，所用的技巧也多，因此编制和调试高效率程序都较难。故建议软件人员，尤其是阅历不深者最好不要课题一到手就向高效率程序进军，更不要设想您的软件中所有程序都是高效率程序。须知高效率程序是不多的。正如下围棋一样，不可能每一手都是手筋。一味追求高效率程序，结果不能履行合同是软件工作者的失职和耻辱。

请看下面程序：

```
10 INPUT "N="; N
20 LPRINT "N="; N
30 DIM X(N)
40 FOR I=1 TO N
50 X(I)=INT(RND*1000)
60 NEXT I
70 LPRINT "INITIAL DATA:"
80 FOR I=1 TO N
90 LPRINT X(I):" ";
100 NEXT I
110 LPRINT
120 LPRINT
130 LPRINT
```

```
140 TIME$="0:0:0"
150 FOR I=1 TO N
160 FOR J=1 TO I
170 IF X(I)<=X(J) THEN 190
180 SWAP X(I),X(J)
190 NEXT J
200 NEXT I
210 X$=TIME$
220 LPRINT"SORTED DATA:"
230 FOR I=1 TO N
240 LPRINT X(I);" ";
250 NEXT I
260 LPRINT
270 LPRINT
280 LPRINT
290 LPRINT "TIME=":X$
300 END
```

```
N=20
INITIAL DATA:
 121   651   868   729   798    73   490   454   107   950   703
 531   971   320   956   934   534   564   671   702
SORTED DATA:
 971   956   950   934   868   798   729   703   702   671   651
 564   534   531   490   454   320   121   107    73
TIME=00:00:01
N=30
INITIAL DATA:
 121   651   868   729   798    73   490   454   107   950   703
 531   971   320   956   934   534   564   671   702   740   666
 453   334   156   736   542   425    55   768
SORTED DATA:
 971   956   950   934   868   798   768   740   736   729   703
 702   671   666   651   564   542   534   531   490   454   453
 425   334   320   156   121   107    73    55
TIME=00:00:03
N=50
INITIAL DATA:
```

```
 121    651    868    729    798    73     490    454    107    950    703
 531    971    320    956    934    534    564    671    702    740    666
 453    334    156    736    542    425    55     768    513    564    741
 661    231    464    128    484    55     362    571    990    290    657
 939    379    890    797    946    323
SORTED DATA:
 990    971    956    950    946    939    934    890    868    798    797
 768    741    740    736    729    703    702    671    666    661    657
 651    571    564    564    542    534    531    513    490    484    464
 454    453    425    379    362    334    323    320    290    231    156
 128    121    107    73     55     55
TIME=00:00:08
```

这是算法最简便，也是最慢的一种排序法：比较交换排序法。它的计算量为 $O(N^2)$

下面是一个在比较交换排序法的基础上，“优化”后的排序法，暂称为优化比较排序法。

```
10 INPUT"N=";N
20 LPRINT"N=";N
30 DIM X(N)
40 FOR I=1 TO N
50 X(I)=INT(RND*1000)
60 NEXT I
70 LPRINT"INITIAL DATA:"
80 FOR I=1 TO N
90 LPRINT X(I);" "
100 NEXT I
110 LPRINT
120 LPRINT
130 LPRINT
140 TIME$="0:0:0"
150 K=N-1
160 J=1
170 L=0
180 FOR I=J TO K
190 IF X(I)>=X(I+1) THEN 230
200 IF L=0 THEN J1=I-1
210 SWAP X(I),X(I+1)
220 L=I
230 NEXT I
240 IF L=0 THEN 280
250 K=L
```

```
260 IF J1<=0 THEN 160
270 J=J1 : GOTO 170
280 X$=TIME$
290 LPRINT "SORTED DATA :"
300 FOR I=1 TO N
310 LPRINT X(I); " ";
320 NEXT I
330 LPRINT
340 LPRINT
350 LPRINT
360 LPRINT "TIME="X$
370 END
```

```
N=20
INITIAL DATA :
 121   651   868   729   798   73   490   454   107   950   703
 531   971   320   956   934   534   564   671   702
SORTED DATA :
 971   956   950   934   868   798   729   703   702   671   651
 564   534   531   490   454   320   121   107   73
TIME=00 : 00 : 02
N=30
INITIAL DATA :
 121   651   868   729   798   73   490   454   107   950   703
 531   971   320   956   934   534   564   671   702   740   666
 453   334   156   736   542   425   55   768
SORTED DATA :
 974   956   950   934   868   798   768   740   736   729   703
 702   671   666   651   564   542   534   531   490   454   458
 425   334   320   156   121   107   73   55
TIME=00 : 00 : 04
N=50
INITAL DATA :
 121   651   868   729   798   73   490   454   107   950   703
 531   971   320   956   934   534   564   671   702   740   666
 453   334   156   736   542   425   55   768   513   564   741
 661   231   464   128   484   55   362   571   990   290   657
 939   379   890   797   946   323
SORTED DATA :
```

```
990     971     956     950     946     939     934     890     868     798     797
768     741     740     736     729     703     702     671     666     661     657
651     571     564     564     542     534     513     531     490     484     464
454     453     425     379     362     334     323     320     290     231     156
128     121     107     73      55      55
TIME=00:00:11
```

这个算法本质上仍是比较排序法，只不过在比较过程中增强了一些判别，以便减少数据交换次数。作者用心虽然良苦，但效果如何呢？从运行的结果可看出不仅劳而无功，反而弄巧成拙，原因何在？作者忽略了他在比较过程中增加的判断和一些相应措施，照样要耗费机时的。

比较排序法虽慢，但程序简单，可读性甚好，若N较小，譬如在50以内时，不失为一较好的排序法。而优化排序法就不一样，效率比比较法更低，由于外循环用条件语句加转向语句构成，又增加了不少判断比较，因而可读性就降低了。

提高程序的效率方法甚多，但不是小改小造，通常对那些成形算法，“优化”已不起作用，此时应选择新算法，再看下面程序：

```
10 REN "THE SORTING METHOD OF STACK"
20 INPUT "N="; N
30 LPRINT "N="; N
40 DIM A(N)
50 LPRINT
60 LPRINT "INITIAL DATA:"
70 LPRINT
80 FOR I=1 TO N
90 A(I)=INT(RND*1000)
100 LPRINT A(I); " "
110 NEXT I
120 LPRINT:LPRINT:LPRINT
130 M=N
140 TIME$="0:0:0"
150 FOR I=INT(M/2) TO 1 STEP-1
160 K=I
170 GOSUB 330
180 NEXT I
190 FOR I=M TO 2 STEP-1
200 SWAP A(I), A(1)
210 K=1
220 M=I-1
230 GOSUB 330
240 NEXT I
```

```
250 X$=TIME$
260 LPRINT"SORTED DATA:"
270 LPRINT
280 FOR I=1 TO N
290 LPRINT A(I);" ";
300 NEXT I
310 LPRINT:LPRINT:LPRINT:LPRINT:"TIME=";X$
320 END
330 X=A(K)
340 L=2*K
350 ON SGN(L-M)+2 GOTO 360, 380, 440
360 IF A(L)<A(L+1) THEN 380
370 L=L+1
380 IF A(K)<=A(L) THEN 440
390 A(K)=A(L)
400 A(L)=X
410 K=L
420 L=2*K
430 GOTO 350
440 RETURN
```

```
N=20
INITIAL DATA
 121   651   868   729   798    73   490   454   107   950   703
 531   971   320   956   934   534   564   671   702
SORTED DATA:
 971   956   950   934   868   798   729   703   702   671   651
 564   534   531   490   454   320   121   107    73
TIME=00:00:01
N=30
INITIAL DATA:
 121   651   868   729   798    73   490   454   107   950   703
 531   971   320   956   934   534   564   671   702   740   666
 453   334   156   736   542   425    55   768
SORTED DATA:
 971   956   950   934   868   798   768   740   736   729   703
 702   671   666   651   564   542   534   531   490   454   453
 425   334   320   156   121   107    73    55
TIME=00:00:03
```

```
N=50
INITIAL DATA
 121   651   868   729   798    73   490   454   107   950   703
 531   971   320   956   934   534   564   671   702   740   666
 453   334   156   736   542   425    55   768   513   564   741
 661   231   464   128   484    55   362   571   990   290   657
 939   379   890   797   946   323
SORTED DATA:
 990   971   956   950   946   939   934   890   868   798   797
 768   741   740   736   729   703   702   671   666   661   657
 651   571   564   564   542   534   531   513   490   484   464
 454   453   425   379   362   334   323   320   290   231   156
 128   121   107    73    55    55
TIME=00:00:05
N=60
INITIAL DATA:
 121   651   868   729   798    73   490   454   107   950   703
 531   971   320   956   934   534   564   671   702   740   666
 453   334   156   736   542   425    55   768   513   564   741
 661   231   464   128   484    55   362   571   990   290   657
 939   379   890   797   946   323   412   424   731   219   220
 763   682   715   933   262
SORTED DATA:
 990   971   956   950   946   939   934   933   890   868   798
 797   768   763   741   740   736   731   729   715   703   702
 682   671   666   661   657   651   571   564   564   542   534
 531   513   490   484   464   454   453   425   424   412   379
 362   334   323   320   290   262   231   220   219   156   128
 121   107    73    55    55
TIME=00:00:07
```

上面是用堆垒法编写的排序程序。它的运算量为0（N·logN），从运算法结果看，它的效率高多了。虽然程序难看些，对不懂堆垒法的人讲，程序不易看懂，但是对于了解堆垒法的讲，照样可明白各句的含义。

作为本章的结束语，我们得指出，书中所介绍的技巧都具有一定实用性，但具体如何取捨，应因人（设计者）而异，因事而异，还要因对象而异，不可作为教条，生搬硬套。

第一篇　良结构程序

良结构程序是指层次分明，结构清晰，功能突出、易阅读、易维护修改，易移植的程序。良结构程序的总的特点是易阅读。

由于良结构程序有上述特点，因而容易编制、调试，不易出错，出了错也容易查觉、改正，故生产周期短。

良结构程序通常不是高效率程序。但由于高效率程序往往和新算法有关并涉及一定技巧，编制、调试都较为困难，生产周期较长。软件人员在接受一个课题后，为了如期履行合同，常只编写良结构程序。在良结构程序调通的基础上进行优化，使其部份程序成为高效率程序。原良结构程序通常仍保留，作为软件查证、维护的参数。

一个优秀软件通常由若干高效率程序和良结构程序组成。正如下围棋不可能手手都是手筋，下象棋无法着着是妙棋一样，即使优秀软件，良结构程序的比例也占得很大。

良结构程序和高效率程序并没有断然分界线，这是因为新算法不可能永远是新算法，一旦它被相当多的人掌握，且又有更新更好的算法出现，那么用它编出的程序，即使编得很好很合理（此时大多数软件人员能看懂了）也只能是良结构程序了。

正因为如此，如何编制良结构程序是每个软件工作者都应重视的大事和基本功。

编制良结构程序须从以下几方面着手。