終端機室

● 費可仕・皮

「嘿呀！皮老弟，幾天不見，原來你窩在這兒享福啊！」

來者何人尚未瞧清，便是「啪拍」的一聲結實地落在背上，不過這一掌倒使我在冷氣房中僵硬的身子熱了一熱。

「喂！老賊，你就不能輕一點兒啊！」手摸摸背上，繼續又瞪著螢光幕。

「皮兄啊！聽大夥兒說你程式功力不賴，有空指點指點吧！」老賊順手拉開一張椅子坐下，開了一部終端機，假裝很熟練地打入學號、密碼…，還一邊說著。

心想，這賊小子，開始還叫我老弟，不一會兒便改口稱兄，又拍起馬屁來，想必是有求於我，暫且不甩他。

「皮兄，你看，這個程式我花了六小時才寫完，共有一百零九行（作得意狀），已增至七十多個版號了，但一跑就當在那裏（用手一指），說什麼除以零的錯誤……」

眞煩！方才一掌打斷了我的思緒，現在又開始喋喋不休。才一百行的程式也敢拿出來比，我坐在這兒已超過了十八小時，程式也將突破五百大關。臭蟲就快抓到了，被死賊一搞，又飛走了。順著老賊的節拍，適時地回個「哦」、答聲「嗯」，或點個頭；目光仍停留在我的螢光幕上。

「這是圖管的歪妹限我在下週三之前交出的計程作業……」老賊繼續說著故事。

咦！歪妹，哇！此妹我倒是見過一回，不但不歪，可眞是正點的可以，不知老賊這傢伙何時搭上的。

「眞是的！是哪個老師如此不憐香惜玉，竟出了個一百多行的作業。」我插了一嘴。

「她自己也笨。選課前也不先打聽每個老師教的內容。有的重視數值方法、程式技巧。有的卻一上課就只教指令、抄例題，這種最適合文學院的修了。每次背好幾個指令，人家翻手册查的，他們都裝在腦袋裏。我們理學院的應去選那些教數值方法的才好。」

「是啊！」我深表同意。「你這程式是牛頓法求根吧？我仔細瞧瞧！」看在歪妹的份上，先替她除蟲。

「程式跑到這兒，就出了錯…」他幾乎又重複一次剛剛說過的。「螢幕上印出除以零的訊息。」

「除以零發生在兩種狀況，一種是除數是零，說明白些，代表除數的變數在經過一串運算後，到了除式時已小得被電腦視作零了。另一種是被除數比除數大太多，除了之後超出電腦允許的最大數量，因此視其爲除以零。」我一口氣對他講了一番道理。

「你當我白痴啊！這些誰不懂？我早就算過，這些根本不可能是很小或是很大的數。」有點兒輕視地答道。

我偏不信邪，螢幕說的絕對忠於程式，再仔細瞧瞧。「這是你微分，即取一次導數的副程式，不錯，常用的區段用了副程式，作微分也用了數值表示式，而不自己先算出解析型式再寫入，很具一般性。」先稱讚他一番，免得遭怒目相視。「這部分我估計看看……，嗯，應在範圍之內，無傷。跳回主程式看看，主要的計算全在一個迴圈內，一個IF……THEN……GOTO……的迴圈。照錯誤的狀況來看，似乎只跑了一圈就當了，奇怪？」眞不名譽，如此簡單的程式竟抓不出蟲，不行，到時候傳出去，歪妹對我的印象豈不大壞。除以零？爲何除以零？有零嗎？「啊！有了，就是零。老賊你看，你在這變數尾巴加了個數字零，但在下次用到時，就是在此式的分母時，卻把此數字零換成了字母O，這兩個字實在太像了，鍵盤上位置又近。」老實說，這種錯我在三個月前犯了第八次之後就不再犯了，沒想到又被我瞧見了。

「怪怪！不愧是皮大俠，早知道昨兒個就來向您求助，就不會花這麼多時間了。」老賊又盡其諂媚之能，我看他眞可封上個馬屁仙的別號。

「你這種屬於拼字的錯誤其實不少見，像SUBROUTINE這個字裏面有字母O，若拼成了數字零，保證電腦告訴你找不到去副程式的路。」「而且，你的程式實在是又臭又長，我想你在找我以前一定不只有這些錯誤吧？」

「是不只，否則怎麼可能有七十多版。」他有些不好意思地說。

「得一次教訓學一次乖，自己抓蟲可培養程式技巧，像我，常泡在這兒，許多可能發生的錯誤，就算自己沒碰上，附近上機的人的咒罵聲中也可聽到不少。」我停了一會兒，又說：「牛頓法求根其實三、四十行之內應可完美的解決，你的程式中既無特別的修正，又未增加結果之有效數字，可見你程式的修練尚未到家，應該多觀摩別人的程式。」邊說邊把我的「傑作」——牛頓法的程式印在螢幕上要老賊看看妙在哪裏。

「觀摩只是說得好聽，實際上還不是抄襲。」老賊反駁道。

「抄歸抄。抄了之後還要用大腦，分析人家程式的奧妙，到底好在哪裏？是節省記憶空間還是減短了操作時間？換種方式寫有何不好？像牛頓法這類的程式

很常見，可比較各家手筆之異同。有朝一日眞要自己創作一個程式時，說不定這些抄來的技巧能有很大的幫助。」一口氣我把個人「抄」的心得全說了出來。

「大俠所言甚是，小弟由衷佩服。下回帶歪妹向您老人家道個謝。告辭！」說完，一拱手就跑了，怕我越說越得意，他今晚和歪妹的晚餐又要改成宵夜了。

× × × × ×

又是個烏雲密佈，風雨交加的日子，既然回不了家，就衝到機房玩玩吧！

「爛貨！ VAX 竟看不懂我的程式，搞了半天還搞不出來，眞是爛得可以。憨吉，你的怎麼樣？」

「等一下，別吵，就快跑出來了——媽的，又死在那兒，眞綏，比你多翹兩節課就來了，還是作不出，看來明天交不出作業了。」

尚未踏入機房的門，便聽見豆花和憨吉兩位難兄難弟在裏頭嚷嚷著。心想不妙，明兒個不正是要交篇程式作業。好在本人深謀遠慮，早在上個週末，花了卅六個小時，撐著眼皮，硬是把它拼了出來。正想拔腿轉身到隔壁的機房，免得跟這兩個懶鬼耗上，但卻來不及了。憨吉一聲「皮老大，救命啊！」叫得我只得拉張椅子坐下。

豆花和憨吉東一嘴，西一舌地向我描述他們在這兒是多麼的辛苦，似乎什麼法子都用盡了，就是得不到正確的結果。到頭來，把帳都算到 VAX 頭上了，怪它搞不清楚狀況，猛送垃圾出來……。

眞要命！老子我來這兒原本只是躲躲雨，避避暑，遇到你們兩個草包已經夠霉的了，竟然有屁不快放，還直誦你們的苦經，一唱一和，倒還蠻有節拍的。「你們的副程式能讓我看一下嗎？」好不容易遇到了休止符插了一嘴。

「邊命！」滴滴幾聲螢幕上馬上換成了他們的副程式。豆花的動作乾淨俐落，不知在我來之前他已用了多少次同樣的指命了。

「副程式不長，才十多行而已。檢查了 n 遍，就是找不出錯。」「副程式共兩個，一個負責計算，結果送回主程式後再由另一副程式負責輸出。主程式的一開頭是把資料讀進來，接著由這兩個副程式處理。」憨吉詳細地爲我解說。

沒想到此人外表看來懶散，對程式中工作分派及副程式的使用方式都有一套。可不要自己也被他錯誤的程式導入歧途。

「嘿嘿！」我冷笑兩聲。「你這程式的輸出可都是零？」

「沒錯。」兩人驚奇地回答。「你連RUN都沒 RUN 過，怎麼知道輸出全是零？太神了吧！」

「你程式這樣寫，自然輸出是零。 VAX 與你無仇，不會對你惡搞的。」本來想說「Garbage in, and Garbage out！」但怕傷了他們的自尊，第一音節未吐出，話就嚥回肚裏去了。

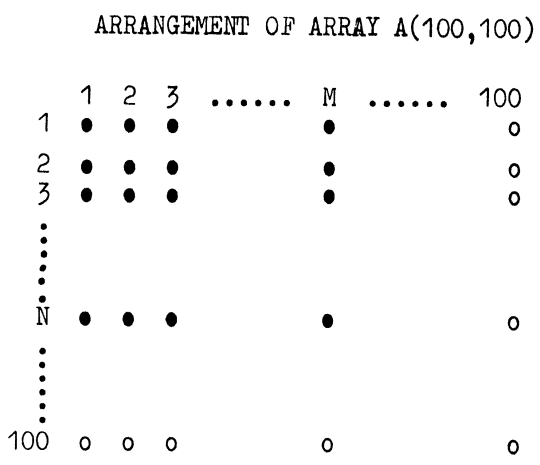
「程式中用了陣列，且陣列又是各副程式中的引數，你用的又是二維陣列，陣列大小又由使用者輸入。」「你知道多維陣列的元素排列順序嗎？」我問道。

「知道啊。像 A (2,3) 這個陣列的順序是A (1,1) 、A(2,1) 、A (1,2) 、A (2,2) 、A (1,3) 、A (2,3) ，對吧？難道這有關係嗎？」

「有咧！關係可大著哩！你的程式想處理的是N與M都是由使用的人輸入，限制是它們都不得大於一百，所以你們在DIMENSION 中把陣列A宣告爲A (100,100) 。不用我多說，A (N , M) 所占的部分是：（如圖中黑圈處）

也就是說A (M , N) 占的只是A (100,100) 的一個角落。」故意在此打住，吞個口水潤潤喉，冷氣使空氣太乾燥了。

「哦！好像有些懂了。非角落部分的空間A (M , N) 都沒



有用到。當主程式呼叫副程式把A（100,100）送至副程式時，副程式的A（M，N），假設A（10,10）好了，A（10,10）共100個元素，則收到的恰好是主程式中的A（1,1），A（2,1），A（3,1）……A（100,1），而其中的A（11,1），A（12,1）……A（100,1），都不是我想要的，而其餘有90個我要的元素卻並沒有傳至副程式。唉！總算找到問題了。不過，那該如何解決呢？我可不想失去這個程式操作時的彈性，硬性規定副程式中陣列元素的個數和主程式中的相等。」憨吉問了一個小問題。

「方法是有，並不十分複雜啦！」本想說實在太容易了，但怕傷了他們的自尊。「在呼叫副程式前，自己製造出一個一維的陣列，如果是A（I,J），A的大小是100×100，則A（I,J）可改爲A（100×（J－1）＋I），或是你若覺得這樣做失去了可讀性，那麼可在程式開頭先定義一個命令函數：INDEX（I，J）=N×（J－1）＋I，N是陣列的列的個數，如此一來A（INDEX（I，J））就是原來的A（I，J）了，不同的只是A已改成了一維的陣列。副程式可再如法泡製。「不過，陣列傳遞的錯誤很少在自己設計的副程式中發生。副程式中宣告虛擬陣列的大小，只要不大於相對應的實際陣列的大小即可，而且設計者自己有自己設計的習慣，主、副程式的習慣都一樣，就不應該出錯。許多例子中可看出個中奧妙。常可看見副程式中的虛擬陣列宣告大小爲1，這是一種〝假設陣列大小〞的宣告，把虛擬陣列的大小假設爲與相對應的實際陣列大小相同。剛才說的A（M，N）這類的虛擬陣列宣告算是可變的陣列度量，其大小爲M×N，當然M×N要小於或等於相對應的陣列大小，例如M·N≦100·100=10,000，M或N若有一者大於100，另一小於100，如（M，N）=（200,2）會有何種情形發生呢？自己用陣列元素排列次序的關係去對對看便知。」「若〝假設陣列大小〞型式來宣告，那麼在呼叫程式區段的陣列元素全部都要送至被呼叫的區段，通常呼叫區爲了使用時有更大的陣性，陣列大小的宣告都不小，若所有元素全送出，包括一堆不須要的也送出，似乎有些不經濟，所以用〝可變的陣列度量〞去宣告比較好，尤其在多維的情形。一維的陣列元素排列沒有多個足標帶來的困擾，順序排下即可。多維的就像剛才舉的例一樣，挺麻煩的。」「你的副程式用的是機器設備中——磁帶或碟片已存好的軟體，使用的應屬於〝可變的陣列度量〞。所以你在呼叫之前應該做出一個一維的陣列送入。一個一維的陣列可以與多維的相對應，可互爲虛擬引數，實際引數，只要熟悉陣列元素排列次序，就可以掌握它們傳遞的規則。」一口氣說了一番〝大道理〞，覺得自己很懂的樣子。

「那麼，如果我用的是〝假設大小〞型的陣列宣告，是不是照我原來的程式設計就不會錯了？」豆花問道。

「你是說你不用現成的軟體，而自己去設計一個，省省吧！」這小子眞是瘋了，功力才幾分就想幹一番轟轟烈烈的事業。連師父我都不敢妄想，眞是初生之犢不畏虎。「你若要試驗是否會成功，另外再設計一個試驗的程式，由螢幕上印出主、副程式中實際及虛擬陣列元素的內容是否相符，是否是你想要的安排方式，不必在原來的程式動手脚，搞不好還把本來好好的程式破壞得滿目瘡痍。至於試驗的結果如何，自己動手、動腦，親眼看到結果，印象才深。」當我不確定結果是什麼時，我發現這招很管用

叫他自個兒去試，等他來問我爲什麼會這樣時，我再依結果去掰出一套理由來解釋。

「我這兒有一些關於副程式、函數和陣列等的應用程式，你看看研究研究吧！若想知道它們的執行結果，把它們逐字敲入電腦看看就可以了。」這招我又用了一次。

「這些例子，除了有剛才說的一些關於陣列的設定之外，陣列的足標甚至不必是從1開始，只要是整數卽可（見RELA×2）；當傳送的字串不知道長度時，也有法子解決（見passed-Length Character Arg ments及Character Data Program Example）；一些特殊的控制敍述可由SUBROUTINE PLYVOL 中的GO TO及Alternate Return Arguments中的RETURN n看到；此外，ENTRY , COMMON, EQUIVALANCE, BLOCK DATA 及庫存函數的使用在副程式設計中都相當重要。」把寶獻出後，當然要作一番介紹，以示其價值。

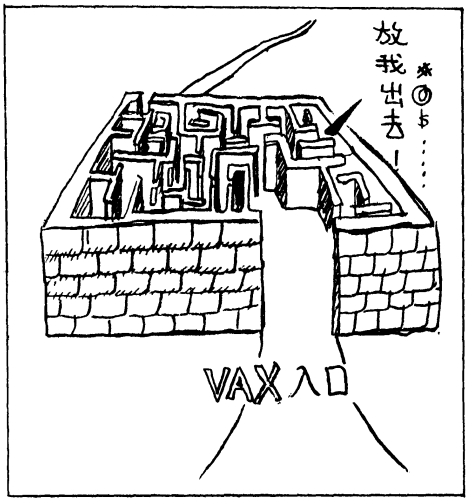
「哇！你可眞好，還特地帶這些"小抄"來見我們。這樣好了，我和豆花一人一份，分別把這些程式敲進去。你一邊講解好了。」有夠慘！沒想到太愛表現得到的後果竟是如此。等他們敲完，再等我想通這些程式的道理，眞不知何時才能脫身。

*** Array Example**

```
C       EXAMPLE
        INTEGER A(100),VAL
        READ*,N
        READ*,(A(I),I=1,N)
        READ*,VAL
        CALL SEARCH(A,N,VAL,INDEX)
        PRINT*,INDEX
        END
C
C       ASSUME A IS SORTED IN ASCENDING ORDER
C       RETURNS THE VALUE OF INDEX AT WHICH A(INDEX) .EQ. VAL
C       RETURN INDEX = 0 IF NONE
C
        SUBROUTINE SEARCH(A,N,VAL,INDEX)
        INTEGER A(1),VAL,HIBOUND,LOBOUND
        HIBOUND=N
        LOBOUND=1
        DO 40 WHILE (HIBOUND .GE. LOBOUND)
              I= (HIBOUND + LOBOUND)/2
              IF (A(I)-VAL) 10,20,30
10                  LOBOUND=I+1
              GOTO 40
20            INDEX=I
              RETURN
30            HIBOUND=I-1
40      CONTINUE
        INDEX=0
        END
```

* Array Example

```
C       EXAMPLE
        INTEGER A(100),VAL
        READ*,N
        READ*,(A(I),I=1,N)
        READ*,VAL
        CALL SEARCH(A,N,VAL,INDEX)
        PRINT*,INDEX
        END
C
C       ASSUME A IS SORTED IN ASCENDING ORDER
C       RETURNS THE VALUE OF INDEX AT WHICH A(INDEX) .EQ. VAL
C       RETURN INDEX = 0 IF NONE
C
        SUBROUTINE SEARCH(A,N,VAL,INDEX)
        INTEGER A(N),VAL,HIBOUND,LOBOUND
        HIBOUND=N
        LOBOUND=1
        DO 40 WHILE (HIBOUND .GE. LOBOUND)
                I= (HIBOUND + LOBOUND)/2
                IF (A(I)-VAL) 10,20,30
10                      LOBOUND=I+1
                GOTO 40
20              INDEX=I
                RETURN
30              HIBOUND=I-1
40      CONTINUE
        INDEX=0
        END
```



* Array Example

```
        SUBROUTINE RELAX2(EPS)

        PARAMETER (M=40, N=60)
        DIMENSION X(0:M,0:N)
        COMMON X

        LOGICAL DONE

1       DONE = .TRUE.

        DO 10 J=1,N-1
        DO 10 I=1,M-1
           XNEW = (X(I-1,J)+X(I+1,J)+X(I,J-1)+X(I,J+1))/4
           IF (ABS(XNEW-X(I,J)) .GT. EPS) DONE = .FALSE.
10         X(I,J) = XNEW

        IF (.NOT. DONE) GO TO 1

        RETURN
        END
```

* Adjustable Arrays

The function in the following example computes the sum of the elements of a two-dimensional array. Note the use of the dummy arguments M and N to control the iteration.

```
      FUNCTION SUM(A,M,N)
      DIMENSION A(M,N)
      SUM = 0.0
      DO 10 J=1,N
      DO 10 I=1,M
10    SUM = SUM + A(I,J)
      RETURN
      END
```

The following statements are sample calls on SUM:

```
DIMENSION A1(10,35), A2(3,56)
SUM1 = SUM(A1,10,35)
SUM2 = SUM(A2,3,56)
SUM3 = SUM(A1,10,10)
```

The upper- and lower-dimension bound values are determined once each time a subprogram is entered. These values do not change during the execution of that subprogram even if the values of variables contained in the array declaration are changed. For example:

```
DIMENSION ARRAY(9,5)
L = 9
M = 5
CALL SUB(ARRAY,L,M)
END

SUBROUTINE SUB(X,I,J)
DIMENSION X(-I/2:I/2,J)
X(I/2,J) = 999
J = 1
I = 2
END
```

In this example, the adjustable array X is declared as X(-4:4,5) on entry to subroutine SUB. The assignments to I and J do not affect that declaration.

* Passed-Length Character Arguments

The following example of a function subprogram uses a passed-length character argument. The function finds the position of the character with the highest ASCII code value; it uses the length of the passed-length character argument to control the iteration. Note that the processor-defined function LEN is used to determine the length of the argument.

```
      INTEGER FUNCTION ICMAX(CVAR)
      CHARACTER*(*) CVAR
      ICMAX = 1
      DO 10 I=2,LEN(CVAR)
10    IF (CVAR(I:I) .GT. CVAR(ICMAX:ICMAX)) ICMAX=I
      RETURN
      END
```

* Subroutine Subprograms

The following example contains a subroutine that computes the volume of a regular polyhedron, given the number of faces and the length of one edge. It uses the computed GO TO statement to determine whether the polyhedron is a tetrahedron, cube, octahedron, dodecahedron, or icosahedron. The GO TO statement also transfers control to the proper procedure for calculating the volume. If the number of faces is not 4, 6, 8, 12, or 20, the subroutine sends an error message to the user's terminal.

Main Program

```
COMMON NFACES, EDGE, VOLUME
ACCEPT *, NFACES, EDGE
CALL PLYVOL
```

* Adjustable Arrays

The function in the following example computes the sum of the elements of a two-dimensional array. Note the use of the dummy arguments M and N to control the iteration.

```
      FUNCTION SUM(A,M,N)
      DIMENSION A(M,N)
      SUM = 0.0
      DO 10 J=1,N
      DO 10 I=1,M
10    SUM = SUM + A(I,J)
      RETURN
      END
```

The following statements are sample calls on SUM:

```
DIMENSION A1(10,35), A2(3,56)
SUM1 = SUM(A1,10,35)
SUM2 = SUM(A2,3,56)
SUM3 = SUM(A1,10,10)
```

The upper- and lower-dimension bound values are determined once each time a subprogram is entered. These values do not change during the execution of that subprogram even if the values of variables contained in the array declaration are changed. For example:

```
DIMENSION ARRAY(9,5)
L = 9
M = 5
CALL SUB(ARRAY,L,M)
END

SUBROUTINE SUB(X,I,J)
DIMENSION X(-I/2:I/2,J)
X(I/2,J) = 999
J = 1
I = 2
END
```

In this example, the adjustable array X is declared as X(-4:4,5) on entry to subroutine SUB. The assignments to I and J do not affect that declaration.

* Passed-Length Character Arguments

The following example of a function subprogram uses a passed-length character argument. The function finds the position of the character with the highest ASCII code value; it uses the length of the passed-length character argument to control the iteration. Note that the processor-defined function LEN is used to determine the length of the argument.

```
      INTEGER FUNCTION ICMAX(CVAR)
      CHARACTER*(*) CVAR
      ICMAX = 1
      DO 10 I=2,LEN(CVAR)
10    IF (CVAR(I:I) .GT. CVAR(ICMAX:ICMAX)) ICMAX=I
      RETURN
      END
```

* Subroutine Subprograms

The following example contains a subroutine that computes the volume of a regular polyhedron, given the number of faces and the length of one edge. It uses the computed GO TO statement to determine whether the polyhedron is a tetrahedron, cube, octahedron, dodecahedron, or icosahedron. The GO TO statement also transfers control to the proper procedure for calculating the volume. If the number of faces is not 4, 6, 8, 12, or 20, the subroutine sends an error message to the user's terminal.

Main Program

```
COMMON NFACES, EDGE, VOLUME
ACCEPT *, NFACES, EDGE
CALL PLYVOL
```

```
TYPE *, 'VOLUME=', VOLUME
STOP
END
```

Subroutine

```
      SUBROUTINE PLYVOL
      COMMON NFACES, EDGE, VOLUME
      CUBED = EDGE**3
      GO TO (6,6,6,1,6,2,6,3,6,6,6,4,6,6,6,6,6,6,6,5), NFACES
      GO TO 6
1     VOLUME = CUBED * 0.11785
      RETURN
2     VOLUME = CUBED
      RETURN
3     VOLUME = CUBED * 0.47140
      RETURN
4     VOLUME = CUBED * 7.66312
      RETURN
5     VOLUME = CUBED * 2.18170
      RETURN
6     TYPE 100, NFACES
100   FORMAT (' NO REGULAR POLYHEDRON HAS ',I3,'FACES.'/)
      VOLUME = 0.0
      RETURN
      END
```

* Alternative Return

The following example illustrates the use of alternate return specifiers to determine where control is to be transferred on completion of the subroutine. The SUBROUTINE statement argument list contains two dummy alternate return arguments corresponding to the actual arguments *10 and *20 in the CALL statement argument list. The decision about which RETURN statement to execute depends on the value of Z, as computed in the subroutine. Thus, if Z is less than zero, the normal return is taken; if Z is equal to zero, the return is to statement label 10 in the main program; if Z is greater than zero, the return is to statement label 20 in the main program.

Main Program

```
CALL CHECK(A,B,*10,*20,C)
TYPE *, 'VALUE LESS THAN ZERO'
GO TO 30
10 TYPE *, 'VALUE EQUALS ZERO'
GO TO 30
20 TYPE *, 'VALUE MORE THAN ZERO'
30 CONTINUE
   .
   .
   .
```

Subroutine

```
SUBROUTINE CHECK(X,Y,*,*,Q)
   .
   .
   .
50 IF (Z) 60,70,80
60 RETURN
70 RETURN 1
80 RETURN 2
END
```

* ENTRY in Subroutine Subprograms

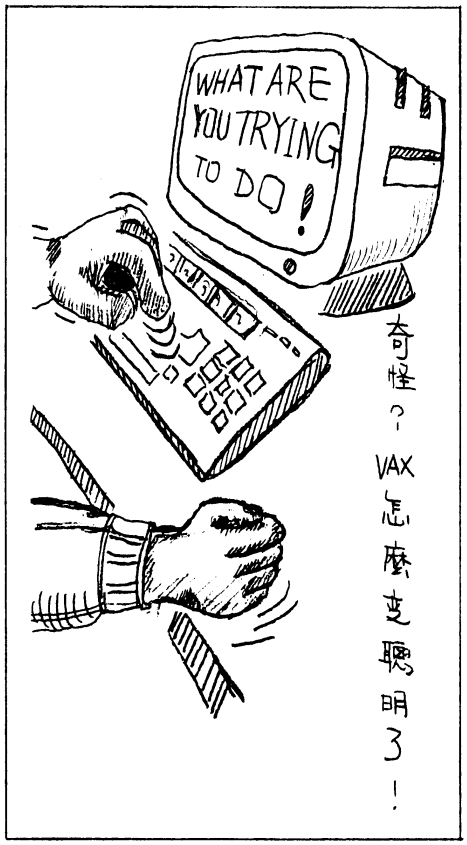
a function subprogram that computes the hyperbolic functions sinh, cosh, and tanh.

```
      REAL FUNCTION TANH(X)

C     Statement function to compute twice sinh

      TSINH(Y) = EXP(Y) - EXP(-Y)
```

```
C     Statement function to compute twice cosh

      TCOSH(Y) = EXP(Y) + EXP(-Y)

C     Compute tanh

      TANH = TSINH(X)/TCOSH(X)
      RETURN

C     Compute sinh

      ENTRY SINH(X)
      SINH = TSINH(X)/2.0
      RETURN

C     Compute cosh

      ENTRY COSH(X)
      COSH = TCOSH(X)/2.0
      RETURN
      END
```

*** Multiple Function Name Usage**

```
C     Compare ways of computing sine

      PROGRAM SINES
      REAL*z8 X, PI
      PARAMETER (PI=3.14159265358979323BD0)
      COMMON V(3)

C     Define SIN as a statement function

      SIN(X) = COS(PI/2-X)
      DO 10 X = -PI, PI, 2*PI/100
      CALL COMPUT(X)

C     Reference the statement function SIN

10    WRITE (6,100) X, V, SIN(X)
100   FORMAT (5F10.7)
      END

      SUBROUTINE COMPUT(Y)
      REAL*8 Y

C     Use intrinsic function SIN as actual argument

      INTRINSIC SIN
      COMMON V(3)

C     Generic reference to double-precision sine

      V(1) = SIN(Y)

C     INTRINSIC FUNCTION SINE AS ACTUAL ARGUMENT

      CALL SUB(REAL(Y),SIN)
      END

      SUBROUTINE SUB(A,S)

C     Declare SIN as name of user function

      EXTERNAL SIN

C     Declare SIN as type REAL*8

      REAL*8 SIN
      COMMON V(3)

C     Evaluate intrinsic function SIN
```



```
      V(2) = S(A)

C     Evaluate user-defined SIN function

      V(3) = SIN(A)
      END

C     Define the user SIN function

      REAL*8 FUNCTION SIN(X)
      INTEGER FACTOR
      SIN = X - X**3/FACTOR(3) + X**5/FACTOR(5)
     1      - X**7/FACTOR(7)
      END

      INTEGER FUNCTION FACTOR(N)
      FACTOR = 1
      DO 10 I=N,1,-1
   10 FACTOR = FACTOR * I
      END
```

* Character Data Program Example

```
      CHARACTER C, ALPHABET*26

      DATA ALPHABET/'ABCDEFGHIJKLMNOPQRSTUVWXYZ'/

      WRITE (6,90)
   90 FORMAT (' CHARACTER EXAMPLE PROGRAM OUTPUT'/)

      DO I=1,26
          WRITE (6,*) ALPHABET
          ALPHABET = ALPHABET(2:)//ALPHABET(1:1)
      END DO

      CALL REVERSE(ALPHABET)
      WRITE (6,*) ALPHABET

      CALL REVERSE(ALPHABET(1:13))
      WRITE (6,*) ALPHABET

      CALL FIND_SUBSTRINGS('UVW', ALPHABET)
      CALL FIND_SUBSTRINGS('A', 'DAJHDHAJDAHDJA4E CEUEBCUEIAWSAWQLQ')

      WRITE (6,*) 'END OF CHARACTER EXAMPLE PROGRAM'
      END

      SUBROUTINE REVERSE(S)
      CHARACTER T, S*(*)

      J = LEN(S)
      DO I=1,J/2
          T = S(I:I)
          S(I:I) = S(J:J)
          S(J:J) = T
          J = J - 1
      END DO
      END

      SUBROUTINE FIND_SUBSTRINGS(SUB,S)
      CHARACTER*(*) SUB, S
      CHARACTER*132 MARKS

      I = 1
      MARKS = ' '

   10 J = INDEX(S(I:),SUB)
      IF (J .NE. 0) THEN
          I = I + (J-1)
          MARKS(I:I) = '^'
          I = I+1
          IF (I .LE. LEN(S)) GO TO 10
      END IF

      WRITE (6,91) S, MARKS
   91 FORMAT (2(/1X,A))
      END
```