NGÔN NGỮ LẬP TRÌNH FORTRAN
VÀ ỨNG DỤNG TRONG KHÍ TƯỢNG THỦY VĂN

Phạm Văn Huấn

NXB Nông nghiệp Hà Nội - 2005

Từ khóa: Ngôn ngữ, lập trình, Fortran, thuật giải, giả trình, lưu đồ, khai báo, hằng, biến, file, lệnh, tuần tự, rẽ nhánh, lặp, chương trình con, thủ tục, hàm.

ĐẠI HỌC QUỐC GIA HÀ NỘI

PHẠM VĂN HUẤN

NGÔN NGỮ LẬP TRÌNH FORTRAN
VÀ ỨNG DỤNG TRONG KHÍ TƯỢNG THỦY VĂN

NHÀ XUẤT BẢN NÔNG NGHIỆP – 2005

MỤC LỤC

Giới thiệu

Giáo trình "Ngôn ngữ lập trình Fortran và ứng dụng trong khí tượng thủy văn" là tập hợp những bài học cơ sở về lập trình mà tác giả đã dạy trong một số năm gần đây cho sinh viên các ngành khí tượng học, thủy văn và hải dương học ở Trường đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội.

Sách này nhằm giới thiệu cho sinh viên lần đầu tiên học lập trình những khái niệm cơ bản về lập trình máy tính, tóm tắt những yếu tố cơ bản và các lệnh thông dụng, đặc điểm sử dụng chúng trong ngôn ngữ lập trình Fortran. Mục tiêu cuối cùng là giúp sinh viên làm quen với các phương pháp xây dựng thuật giải các bài toán thông dụng của toán học tính toán, thống kê toán học và xử lý số liệu, rèn luyện kỹ năng lập trình để giải những bài toán xử lý và phân tích số liệu, tính toán ứng dụng ở mức độ ban đầu trong thời gian học tập và nghiên cứu ở trường đại học.

Những thí dụ và hệ thống bài tập tự luyện trong sách này có ý nghĩa minh họa, hướng sinh viên tới vận dụng các lệnh của Fortran để viết ra những chương trình ứng dụng nho nhỏ có tính cụ thể, bước đầu làm quen với những đặc thù xử lý dữ liệu quan trắc trong chuyên môn khí tượng thủy văn. Những đặc điểm khác của nội dung ứng dụng lập trình trong các chuyên ngành này như quản lý cơ sở dữ liệu, các phương pháp thống kê hiện đại, các phương pháp giải số trị những bài toán động lực khí quyển, đại dương... chưa được đề cập ở đây do khuôn khổ kiến thức chuyên môn của người học, đó là đối tượng của các môn học chuyên đề khác của chương trình học tập, nhưng từ đây đến đó thực ra cũng không xa.

Vì là tài liệu học tập về lập trình cơ sở, nội dung ngôn ngữ trong sách này cũng không bao quát hết những yếu tố trong thế giới to lớn của Fortran. Nên bắt đầu bằng những gì đơn giản nhưng được việc. Một khi người học bắt đầu biết lập trình, thấy được ứng dụng máy tính có ích trong học tập và nghiên cứu của mình sẽ nảy sinh nhu cầu tìm hiểu và khai thác Fortran trong rất nhiều tài liệu tra cứu và sách chuyên khảo khác hoặc hệ thống trợ giúp sẵn có của Fortran.

Như vậy, sách này không chỉ là tài liệu học tập cho những sinh viên các chuyên môn khí tượng thủy văn, mà có thể có ích cho sinh viên, học viên cao học nhiều chuyên ngành khác hoặc bất kì ai muốn tự học lập trình máy tính một cách nhẹ nhàng.

Trong sách này, mỗi chương được cấu tạo như một bài học. Mỗi chuyên từ, khái niệm xuất hiện lần đầu đều được in nghiêng, các câu lệnh được in chữ hoa đậm và bao trong hộp để giúp người đọc thuận tiện tra cứu khi chưa thuộc chính tả câu lệnh.

Những thí dụ minh họa được chọn lọc sao cho đơn giản, nhưng có tính điển hình, giúp người đọc liên tưởng đến lớp bài toán khác có thể cùng sử dụng cách giải này. Chương trình thí dụ luôn nhất quán áp dụng ý tưởng *chia để trị*, tức phân nhiệm vụ lớn thành các việc nhỏ hơn để thực hiện từng việc một dẫn tới kết quả cuối cùng. Với cách trình bày này, bạn đọc sẽ thấy lập trình không còn là cái gì rắc rối, khó hiểu, mà nó tự nhiên như ta vẫn giải quyết bài toán không bằng máy tính.

Những tóm tắt kinh nghiệm gỡ rối và lời khuyên về rèn luyện phong cách lập trình ở mỗi bài học có thể rất có ích cho người học. Và đây là lời khuyên đầu tiên cho người mới học lập trình: Hãy luôn tưởng tượng xem mình sẽ phải giải bài toán "bằng tay" như thế nào trước khi bắt đầu nghĩ cách viết chương trình máy tính. Hãy nhớ lấy chính tả, cú pháp của câu lệnh và việc này không khó, vì lệnh Fortran giống như một câu tiếng Anh đơn giản. Nhưng hãy rất chú ý tới chính những điều đơn giản, thí dụ khi nhìn dòng lệnh sau

PRINT * , *danh sách các mục cần in*

thì hãy cố gắng đọc kĩ hay hỏi lại xem thế nào là danh sách, thế nào là một mục in, một mục in có thể là những gì.

Tác giả

Chương 1 - Khái niệm về lập trình máy tính để giải các bài toán ứng dụng

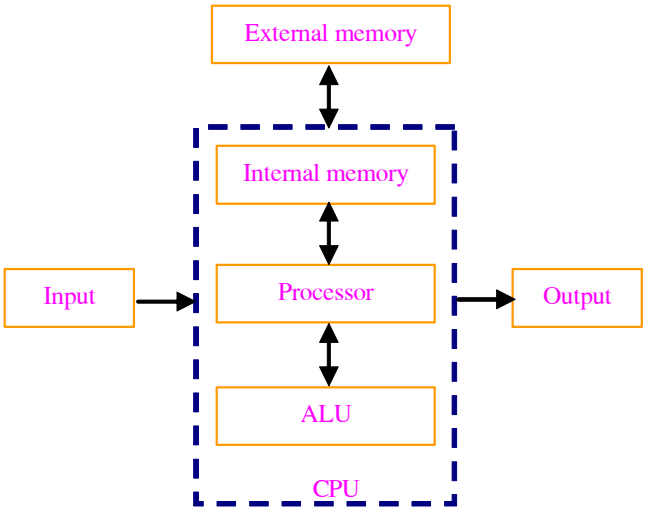
1.1. Phần cứng và phần mềm máy tính

Máy tính được thiết kế để thực hiện những thao tác tuân theo một tập những câu lệnh do người dùng viết ra, gọi là chương trình.

Các máy tính có cấu tạo chung bên trong như trên hình 1.1. Người dùng sử dụng bàn phím, chuột hoặc những thiết bị nhập dữ liệu khác để đưa thông tin vào máy tính. *Bộ xử lý* (processor) là một phần của máy tính kiểm soát tất cả các phần khác. Bộ xử lý nhận dữ liệu vào và lưu chúng ở bộ nhớ (memory). Nó nhận biết các lệnh của chương trình. Nếu ta muốn cộng hai giá trị, bộ xử lý sẽ lấy hai giá trị đó từ bộ nhớ và gửi đến *khối xử lý số học lôgic* (ALU). Khối này thực hiện phép cộng và bộ xử lý lưu kết quả vào bộ nhớ. Trong khi xử lý, bộ xử lý và khối số học lôgic sử dụng một lượng bộ nhớ nhỏ gọi là *bộ nhớ trong* (internal memory). Phần lớn dữ liệu được lưu ở *bộ nhớ ngoài* (external memory) như đĩa cứng, đĩa mềm, chúng cũng nối với bộ xử lý. Bộ xử lý, bộ nhớ trong và ALU gọi chung là *khối xử lý trung tâm* hay CPU.

Trong chương trình, ta thường lệnh cho máy tính in kết quả tính toán lên màn hình hay máy in nối với máy tính và là những *thiết bị xuất dữ liệu*.

Phần mềm chứa những chỉ dẫn hoặc lệnh mà ta muốn máy tính thực hiện. Phần mềm có thể được viết bằng nhiều ngôn ngữ và cho nhiều mục đích. Những chương trình thực hiện những thao tác chung, thường được nhiều người sử dụng gọi là những *phần mềm công cụ*. *Hệ điều hành* là tập hợp các chương trình giúp người dùng giao tiếp với máy tính. Hệ điều hành tạo một môi trường thuận tiện cho người dùng "giao tiếp" được với máy tính, thực hiện những chương trình ứng dụng như các bộ biên dịch ngôn ngữ lập trình, các phần mềm công cụ... Hệ điều hành gồm một số chương trình cho phép thao tác với file như in, sao chép, hiển thị danh sách file... Những hệ điều hành hiện đại như Windows còn giúp máy tính nhận biết và quản lý công việc của rất nhiều *thiết bị ngoại vi* nối kèm với máy tính như các thiết bị nhập, xuất dữ liệu, màn hình, máy in, máy quét ảnh, loa, các máy quan trắc chuyên dụng...



Hình 1.1. Sơ đồ khối của một máy tính

Thông thường hiện nay các chuyên gia lập chương trình viết ra rất nhiều chương trình để máy tính thực hiện, từ những chương trình đơn giản để giải các bài toán nhỏ, tính toán một vài giá trị, đến những chương trình đồ sộ xử lý thông tin phức tạp, thông minh, giải những bài toán khoa học kĩ thuật lớn, chế bản văn bản, thiết kế đồ họa, các chương trình nghe nhạc, xem phim, trò chơi, truy cập Internet. Những chương trình tương đối lớn và phức tạp thường được gọi là *những phần mềm*. Người dùng máy tính có thể sử dụng những chương trình đó. Ngày nay chúng ta có cảm giác rằng máy tính làm được tất cả mọi việc. Tuy nhiên, phải nhớ rằng tất cả những gì máy tính làm được là do nó làm việc theo một chương trình do con người tạo ra.

1.2. Thực hiện một chương trình máy tính

Thực hiện một chương trình máy tính thường còn được gọi tắt là *chạy chương trình*. Khi người dùng máy tính muốn nó làm một việc gì đó, thí dụ giải một bài toán, thì người dùng phải viết ra một chương trình để cho máy thực hiện. Người lập trình thường viết các chương trình máy tính bằng ngôn ngữ bậc cao với những câu lệnh giống như những câu tiếng Anh, dễ học và sử dụng. Ngôn ngữ Fortran cũng thuộc loại đó. Mỗi một bước ta muốn máy tính thực hiện phải được mô tả ra theo một cú pháp ngôn ngũ đặc thù (language syntax). Tuy nhiên, chương trình ta viết như vậy vẫn phải được một chương trình chuyên (bộ biên dịch - compiler) dịch thành ngôn ngữ máy thì máy tính mới hiểu và thực hiện được. Khi compiler dịch các dòng lệnh ta viết, nó tự động tìm các lỗi dịch, hay *lỗi cú pháp* (syntax error), tức các lỗi về chính tả, các dấu phân cách... Nếu chương trình viết ra có lỗi dịch, bộ biên dịch sẽ thông báo để người viết chương trình sửa. Sau khi đã sửa hết lỗi, ta chạy lại chương trình bắt đầu từ bước dịch. Một khi dịch xong, một chương trình soạn thảo liên kết (linkage editor program) sẽ thực hiện việc hoàn tất sẵn sàng cho bước thực hiện. Chính là ở bước này các lệnh ta viết được thực hiện trong máy tính. Lỗi chương trình cũng có thể xuất hiện trong bước này, gọi là *lỗi trong khi chạy chương trình* (run-time error) hay *lỗi lôgic*. Những lỗi này không liên quan tới cú pháp của lệnh, mà liên quan tới lôgic của các lệnh, chỉ lộ ra khi máy tính thực thi câu lệnh. Thí dụ, lệnh

$$X = A / B$$

là một câu lệnh đúng, bảo máy tính lấy A chia cho B và gọi kết quả là X. Tuy nhiên, giả sử nếu B bằng không, phép tính chia cho số không là phép tính sai, không có nghĩa và ta được thông báo lỗi chạy chương trình. Các lỗi lôgic không phải bao giờ cũng được thông báo. Thí dụ, nếu trong chương trình thay vì chia một số cho 0.10 ta viết thành nhân với 0.10, khi chạy chương trình sẽ chẳng có lỗi nào được thông báo, nhưng đáp số bài toán, tức kết quả mà ta mong đợi, sẽ là sai.

1.3. Quy trình giải bài toán trên máy tính

Nhìn chung công việc giải một bài toán bằng máy tính gồm năm bước sau:

1) Phát biểu bài toán một cách rõ ràng.

2) Mô tả thông tin nhập vào và xuất ra.

3) Giải bài toán bằng tay đối với tập dữ liệu đơn giản.

4) Phát triển cách giải bài toán thành dạng tổng quát.

5) Kiểm tra đáp số với nhiều tập dữ liệu khác nhau.

Bây giờ ta minh họa năm bước trên qua thí dụ bài toán tính giá trị trung bình của một tập số liệu thực nghiệm.

Bước 1: Ta phát biểu bài toán một cách rõ ràng như sau: "Tính trị số trung bình của tập các giá trị số liệu thực nghiệm".

Bước 2: Chỉ ra cụ thể số liệu vào và ra là gì, hình thức ra sao. Nếu có

tờ ghi một số giá trị của số liệu, đòi hỏi nhập vào máy qua bàn phím, khi nào hết số liệu thì gõ giá trị 0.0 để báo hết, sau đó mới tính trị số trung bình và in ra kết quả là trị số trung bình đó. Vậy thì phải mô tả ở bước 2 như sau: "Đầu vào là chuỗi các giá trị số thực khác không. Đầu ra là giá trị trung bình, sẽ là một số thực được in trên màn hình". Giả sử nếu đầu vào là một số số liệu như trên nhưng đã được ghi vào một tệp (file) trong ổ cứng, quy cách ghi cũng có những đặc điểm nhất định, thì bước mô tả vào và ra sẽ hoàn toàn khác và cách giải cũng sẽ khác. Khi đó ta phải mô tả rõ cách thức số liệu ghi trong file. Thí dụ, ta có thể mô tả dữ liệu đầu vào và đầu ra như sau: Dữ liệu đầu vào là một chuỗi số thực được ghi trong file văn bản có tên là SOLIEU.DAT với quy cách ghi như sau: dòng trên cùng ghi một số nguyên chỉ số phần tử của chuỗi, các dòng tiếp sau lần lượt ghi các số thực, mỗi số trên một dòng.

Bước 3: Dùng máy tính tay tính thử với một tập đơn giản gồm năm số liệu: thí dụ:

Thứ tự	Giá trị
1	23.43
2	37.43
3	34.91
4	28.37
5	30.62
Trung bình =	30.95

Tập số liệu này và kết quả sẽ được dùng để kiểm tra ở bước 5.

Bước 4: Trong bước này ta khái quát lại những thao tác cần làm ở bước 3. Tuần tự những thao tác này để dẫn đến giải được bài toán chính là *thuật giải* hay *thuật toán* (algorithm). Ta sẽ mô tả tuần tự từ đầu đến cuối quá trình giải. Chia quá trình thành một số khối và liệt kê những khối đó ra. Sau này chương trình máy tính sẽ tuần tự thực hiện các khối chia đó. Trong mỗi khối ta lại chi tiết hóa thêm ra đến mức có thể chuyển thành những lệnh máy tính. Vậy ở đây đã áp dụng hai phương pháp: *phân khối* và *chi tiết hoá* từng khối. Với bài toán đang xét, trường hợp dữ liệu đầu vào cần nhập từ bàn phím, ta chia thành ba khối:

- Nhập các giá trị số và lấy tổng của chúng.
- Chia tổng cho số giá trị.
- In trị số trung bình.

Cụ thể hoá từng khối sẽ dẫn tới *giả trình* của chương trình như sau:

1. Cho tổng của các giá trị bằng không.
2. Cho số số liệu vào bằng không.
3. Nhập vào từng giá trị và kiểm tra chừng nào giá trị nhập vào còn khác số 0.0 thì:
   - Cộng thêm giá trị đó vào tổng.
   - Cộng thêm 1 vào số số liệu.
4. Chia tổng cho số số liệu để được giá trị trung bình.
5. In giá trị trung bình.

Vì thuật giải đã được mô tả khá chi tiết, ta chuyển thuật giải đó thành chương trình như sau:

```
      PROGRAM TGTTB
      INTEGER DEM
      REAL X, TONG, TB
      TONG = 0.0
      DEM = 0
5     READ*,  X
      IF (X .NE. 0.0) THEN
         TONG = TONG + X
         DEM = DEM + 1
         GOTO 5
      END IF
```

```
      TB = TONG / REAL(DEM)
      PRINT 6, TB
6     FORMAT (1X, 'TRUNG BINH BANG ' , F6.2)
      STOP
      END
```

Bước 5: Trong bước này ta thử chạy chương trình đã viết với tập số liệu đã được thử bằng cách tính tay ở mục 3. Đầu ra trên màn hình máy tính phải như sau:

```
TRUNG BINH BANG 30.95
```

Ngoài ra, ta có thể chạy thử với một số tập số liệu khác nhau để tin chắc vào tính đúng đắn lôgic và hoàn hảo của chương trình đã xây dựng.

Những khái niệm thuật giải và giả trình trên đây có ý nghĩa rất quan trọng. Cách giải, phương pháp giải một bài toán chính là thuật giải. Các bài toán khoa học kĩ thuật thực hiện trên máy tính thường có thuật giải là những phương pháp của toán học hoặc của các khoa học chuyên ngành mà người lập trình đã biết. Một số nhiệm vụ, bài toán khác có thể có cách giải xuất phát từ kinh nghiệm thực tế, từ cách suy nghĩ lôgic thường ngày của chúng ta.

Thí dụ, khi giải phương trình bậc hai $ax^2 + bx + c = 0$ bằng máy tính, ta có thể tính giá trị của biệt thức Δ. Sau đó tùy giá trị của Δ có thể là: $\Delta < 0$ phương trình vô nghiệm, $\Delta = 0$ phương trình có một nghiệm kép và $\Delta > 0$ phương trình có hai nghiệm riêng biệt mà đưa ra thông báo kết quả. Trong thí dụ này, thuật toán là phương pháp quen thuộc mà chúng ta đã học trong đại số.

Một thí dụ khác: Có một danh sách sinh viên cùng với điểm của môn thi. Sắp xếp lại danh sách đó sao cho người có điểm thi cao hơn thì ở dòng trên. Ta có thể làm như sau:

Tạm thời xem người thứ nhất là người đứng đầu danh sách. Dùng ngón tay trỏ dõi theo từng người còn lại, kể từ người thứ hai cho đến hết danh sách, nếu ai có điểm thi cao hơn thì chuyển người đó lên đầu danh sách và người đang ở đầu danh sách chuyển xuống chỗ của người vừa được thay. Kết quả ta được danh sách mới với người có điểm thi cao nhất ở dòng đầu. Nhưng từ dòng thứ hai đến dòng cuối cùng của danh sách có thể thứ tự vẫn còn lộn xộn.

Bây giờ ta chỉ còn việc sắp xếp lại từ dòng thứ hai trở đi. Ta theo dõi từ người thứ ba cho đến người cuối cùng, nếu ai có điểm thi cao hơn thì được đưa lên dòng thứ hai và người đang ở dòng thứ hai sẽ bị đưa xuống dòng của người vừa thay thế. Kết quả là người ở dòng thứ hai trong danh sách mới sẽ là người có điểm thi cao thứ nhì. Nhưng từ dòng thứ ba đến cuối danh sách vẫn còn lộn xộn.

Tiếp tục, ta phải sắp xếp lại danh sách kể từ dòng thứ ba theo đúng cách như trên. Lặp lại công việc như vậy cho đến dòng trước dòng cuối cùng, ta sẽ được danh sách hoàn chỉnh sắp xếp theo thứ tự giảm dần của điểm thi.

Ta thấy, trong trường hợp này thuật giải của bài toán chính là cái cách mà chúng ta có thể vẫn thường làm trong thực tế đời sống khi phải sắp xếp lại danh sách theo thứ tự nhưng không dùng máy tính. Và cách làm "bằng tay" này cũng có thể được áp dụng làm thuật toán cho các loại bài toán sắp xếp trong máy tính.

Chúng tôi giới thiệu chi tiết hai thí dụ vừa rồi cốt là để sinh viên ý thức được rằng tư duy thuật giải, suy nghĩ về cách giải là bước quan trọng nhất khi bắt tay vào xây dựng chương trình máy tính. Có xác định được thuật giải thì mới nói đến việc lập được chương trình để giải bài toán trên máy tính.

Còn giả trình giống như những lời chỉ dẫn về phương pháp, cách giải cho ai đó tuần tự thực hiện các bước của thuật giải bài toán để dẫn tới kết

quả. Bạn hãy tưởng tượng trong đời sống nếu bạn muốn nhờ ai đó thực hiện một nhiệm vụ. Nếu nhiệm vụ đó không quá tầm thường, thì thường ít ra bạn phải giải thích tóm tắt yêu cầu, tuần tự thực hiện nhiệm vụ cho người ta - bạn đã đặt chương trình cho người ta thực hiện. Như vậy, ta thấy thực chất giả trình đã là một chương trình, chỉ có điều nó được viết ra ngắn gọn bằng vài câu, vài kí hiệu quen dùng, chưa được viết bằng một ngôn ngữ lập trình cụ thể mà thôi. Một chương trình máy tính viết bằng ngôn ngữ Fortran hay bất kỳ một ngôn ngữ nào khác chẳng qua chỉ là những lời chỉ dẫn này được viết theo qui ước kí hiệu để máy tính hiểu được mà làm thay cho ta.

1.4. Những chương trình Fortran hoàn chỉnh

Chương trình TGTTB ở mục trước có thể là một thí dụ về một chương trình hoàn chỉnh. Như vậy ta thấy một chương trình hoàn chỉnh bắt đầu bằng lệnh PROGRAM chỉ sự bắt đầu của chương trình. Dạng tổng quát của lệnh này như sau:

PROGRAM *Tên chương trình*

trong đó *tên chương trình* là một xâu ký tự gồm từ một đến sáu ký tự, bắt đầu bằng chữ cái và chỉ chứa các chữ cái la tinh và chữ số. Kết thúc chương trình là hai lệnh:

STOP

END

Lệnh PROGRAM và lệnh STOP là các lệnh tuỳ chọn, có thể không nhất thiết phải có. Nếu ta không viết ra, chúng có thể được chương trình dịch tự thêm vào. Phần thân chương trình gồm tất cả các lệnh khác nhằm thực hiện bài toán được giải và chia thành hai nhóm: nhóm các lệnh thực hiện (executable statement) và nhóm các lệnh không thực hiện (non-executable statement) hay gọi là các lệnh mô tả, lệnh khai báo. Nhóm các lệnh mô tả phải nằm ở phía trên của thân chương trình.

Như vậy các chương trình Fortran có thể có cấu trúc tổng quát như sau:

PROGRAM *Tên chương trình*
Các lệnh không thực hiện (Non-executable statements)
Các lệnh thực hiện (Executable statements)
STOP
END

Trong chương trình tính giá trị trung bình của chuỗi số thực ở thí dụ trên, ta thấy sau từ khóa PROGRAM là tên chương trình - đó là cụm chữ TGTTB. Nhóm các lệnh không thực hiện (những lệnh mô tả) gồm 2 lệnh:

```
      INTEGER DEM
      REAL X, TONG, TB
```

còn nhóm lệnh thực hiện gồm các lệnh ở tiếp sau hai lệnh trên:

```
      TONG = 0.0
      DEM = 0
5     READ*,  X
      IF (X .NE. 0.0) THEN
         TONG = TONG + X
         DEM = DEM + 1
         GOTO 5
      END IF
      TB = TONG / REAL(DEM)
      PRINT 6, TB
6     FORMAT (1X, 'TRUNG BINH BANG ' , F6.2)
```

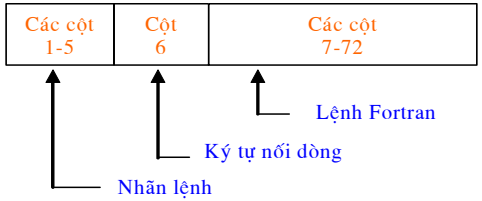
Những lệnh thực hiện thường là những lệnh gán, lệnh tính toán các

phép tính, lệnh chuyển điều khiển, đọc, ghi số liệu và một số lệnh khác. Cuối cùng chương trình có lệnh STOP và END.

Trong thực tế có thể có những chương trình lớn hơn rất nhiều, gồm hàng nghìn dòng lệnh và có cấu trúc phức tạp. Nhưng ta vẫn thấy nó có phần đầu, phần thân và phần cuối, trong phần thân chương trình cũng chỉ có hai nhóm lệnh giống như trong chương trình đơn giản trên đây.

1.5. Quy cách soạn thảo một chương trình Fortran

Các chương trình Fortran được soạn thảo nhờ một bộ soạn thảo (editor) hoặc phần mềm soạn văn bản nào đó. Các lệnh của một chương trình được viết thành các dòng nối tiếp nhau, mỗi lệnh trên một dòng mới. Trên màn hình soạn thảo chuẩn (*) người ta quy ước các cột từ 1 đến 5 (hình 1.2) dùng để ghi số hiệu lệnh hay gọi là *nhãn lệnh*, cột 6 chuyên dùng để ghi *ký tự nối dòng lệnh*, nội dung các dòng lệnh chỉ được ghi trên các cột từ 7 đến 72. Tất cả các thông tin ở quá cột 72 bị bỏ qua.



Hình 1.2. Quy cách viết lệnh Fortran trên màn hình soạn thảo

(*) Trước đây người ta phải dùng giấy chuyên dụng, gọi là *blank*, để viết chương trình Fortran. Sau đó từng lệnh chương trình từ giấy chuẩn được ghi vào một tờ bìa chuyên dụng có hình dáng đặc biệt bằng máy đục lỗ giúp tự động mã hóa từng ký tự của dòng lệnh thành một hàng lỗ với vị trí khác nhau. Ngày nay các bộ soạn thảo có thể giúp chúng ta viết các lệnh trực tiếp từ bàn phím, khi viết các ký tự và ký hiệu hiện trên màn hình để dễ theo dõi và chỉnh sửa giống như ta soạn thảo một văn bản bất kỳ, sau đó lưu trong máy tính dưới dạng file nguồn.

Nhãn lệnh là những số nguyên dương, khác không, dùng để chỉ số hiệu của dòng lệnh. Chỉ những dòng lệnh nào cần được chuyển điều khiển tới bởi những dòng lệnh khác mới nhất thiết phải có nhãn lệnh. *Dấu nối dòng lệnh* có thể là bất cứ ký tự nào ngoài ký tự trống và số không, thường người ta hay dùng dấu * hoặc dấu +, để chỉ rằng dòng hiện tại là phần nối tiếp của lệnh ở dòng trên đó. Trong các dòng lệnh có thể có những ký tự trống để dễ đọc. Trong chương trình soạn thảo có thể có những *dòng ghi chú* (comment lines); những dòng này không thuộc nội dung chương trình, không được dịch khi dịch chương trình, mà chỉ có tác dụng gợi nhớ cho người lập trình khi theo dõi kiểm tra chương trình. Tất cả các dòng ghi chú phải bắt đầu bằng một chữ cái, thường người ta dùng chữ C (chữ cái đầu tiên của từ comment), đứng ở cột thứ nhất của các cột dùng để ghi nhãn. Trong sách này sẽ luôn sử dụng chữ cái C để đánh dấu dòng ghi chú trong các chương trình.

Xây dựng một chương trình máy tính nói chung là một công việc khó và đòi hỏi tính cẩn thận, tỉ mỉ. Kinh nghiệm cho thấy rằng ngay cả đối với người lập trình thành thạo, khi viết một chương trình dù đơn giản vẫn có thể mắc lỗi, trong đó có cả những lỗi không ngờ tới. Do đó, ở một số sách dạy ngôn ngữ lập trình, người ta còn khuyên người học ngay từ đầu chú ý luyện thói quen, hay phong cách (style) soạn thảo chương trình. Một chương trình đẹp là chương trình tính đúng và nhanh cái mà ta cần tính, nhưng đơn giản, dễ hiểu và sáng sủa về cách trình bày. Trong tài liệu này dần dần cũng sẽ có những chỉ dẫn, những lời khuyên quan trọng cho người học rèn luyện phong cách soạn chương trình. Chịu khó rèn luyện những thói quen tốt cũng góp phần giúp chúng ta tiến xa.

Bài tập

1. Nếu dùng chữ *C* làm dấu nối dòng lệnh thì có được không?

2. Các nhãn trong chương trình có cần tăng dần không?

3. Nêu các bước giải bài toán sắp xếp chuỗi số nguyên nhập từ bàn phím theo thứ tự lớn dần. Viết giả trình cho bài toán đó.

Chương 2 - Những yếu tố cơ bản của Fortran

2.1. Dữ liệu và cách biểu diễn dữ liệu trong Fortran

Fortran có thể thao tác với sáu loại (kiểu) dữ liệu cơ bản thường gặp trong thực tế là: các số nguyên, số thực, số phức, số thực độ chính xác gấp đôi, các giá trị lôgic và dữ liệu văn bản. Trong chương này ta sẽ làm quen với các dữ liệu kiểu số nguyên, số thực, giá trị lôgic và văn bản (chuỗi ký tự).

Số nguyên là liệt các số thập phân với dấu +, – hoặc không có dấu. Thí dụ:

0 ; 6 ; –400 ; +1234

Các số nguyên được biểu diễn dưới dạng I. Giá trị cực đại của số nguyên gọi là khả năng biểu diễn số nguyên của máy tính.

Trong Fortran có hai dạng biểu diễn *số thực*. Dưới *dạng* F số thực gồm phần nguyên và phần thập phân, cách nhau bởi dấu chấm. Số thực có thể có dấu dấu +, – hoặc không có dấu. Nếu phần nguyên hoặc phần thập phân bằng không, có thể không cần viết ra các phần đó. Dấu chấm thập phân nhất thiết phải có mặt. Thí dụ:

–2.583 ; 14.3 ; 0.8 ; 12. ; .7 ; 14.

Giá trị cực đại và số chữ số có nghĩa cực đại trong dạng F phụ thuộc vào dạng, hay kiểu (kind) khai báo của số thực.

Dạng E biểu diễn số thực thành hai phần: phần hằng thực nằm trong khoảng từ 0,1 đến 1,0 và phần bậc. Bậc bắt đầu bằng chữ E, tiếp sau là hằng nguyên gồm không quá hai chữ số thập phân, có thể có dấu hoặc không dấu. Thí dụ số 25000 có thể viết dưới dạng E là 0.25E05. Số chữ số có nghĩa của phần hằng thực và hằng nguyên cũng tùy thuộc loại số thực khai báo.

Hằng với độ chính xác gấp đôi (dạng D) có thể viết như số với dấu chấm thập phân, chứa từ 8 đến 16 chữ số có nghĩa, hoặc như số dạng mũ với chữ D thay vì E, trong đó phần hằng thực có thể chứa tới 16 chữ số có nghĩa. Thí dụ:

2.71828182 ; 0.27182818D+1

Trị tuyệt đối cực đại của các số thực thường và độ chính xác gấp đôi bằng 10^{-79} đến 10^{75}.

Số phức biểu diễn bằng một cặp hằng thực trong dấu ngoặc đơn và cách nhau bởi dấu phảy. Thí dụ (2.1, 0.5E2) biểu diễn số phức $2{,}1 + 50\,i$ trong toán học.

Hai số trong dấu ngoặc ứng với các phần thực và phần ảo phải cùng độ chính xác biểu diễn.

Các giá trị dữ liệu văn bản dùng để biểu diễn các đoạn văn bản như tên các đại lượng, các khái niệm, thí dụ cụm chữ " Toc do", "Temperature", "BAO CAO SO 1"... Người ta còn gọi dữ liệu văn bản là dữ liệu ký tự, xâu ký tự, dữ liệu chữ.

Các chữ số 1, 2, ..., 9, 0 khi dùng với tư cách là để biểu diễn các giá trị số tương ứng thì chúng cũng là những dữ liệu kiểu văn bản.

Dữ liệu lôgic dùng để chỉ khả năng có hay không của một sự kiện, đúng hay sai của một biểu thức quan hệ. Người ta dùng hai giá trị lôgic là

.TRUE. và **.FALSE.** để chỉ hai trạng thái đối lập nhau trong những thí dụ trên và ngôn ngữ Fortran có thể xử lý với những giá trị lôgic, tức thực hiện những phép tính đối với các giá trị lôgic như trong toán học có thể thực hiện.

Sở dĩ máy tính làm được những việc như chúng ta thấy là vì nó có thể xử lý thông tin, so sánh, tính toán được với những kiểu dữ liệu này và đưa ra những kết luận, thông báo... Tất cả những thông tin chúng ta gặp trong đời sống thực tế đều có thể được biểu diễn bằng những dữ liệu kiểu này hoặc kiểu khác.

Trên đây là *những kiểu dữ liệu cơ bản* của ngôn ngữ lập trình Fortran. Sau này và ở các chương khác, chúng ta sẽ thấy còn có những kiểu dữ liệu khác được tổ chức dựa trên những kiểu dữ liệu cơ bản vừa trình bày.

Ở đây chúng ta cần lưu ý rằng những khái niệm dữ liệu trong máy tính như số nguyên, số thực... nói chung giống với những khái niệm tương ứng trong đời sống hoặc trong toán học. Nhưng đồng thời cũng có những nét khác biệt. Thí dụ, Fortran chỉ hiểu và tính toán được với những số nguyên loại thường không lớn hơn $2 \cdot 10^9$, ngôn ngữ lập trình Pascal chỉ làm việc với những số nguyên không lớn hơn 32767 và không nhỏ hơn –32768, trong khi hàng ngày chúng ta có thể viết trên giấy hoặc tính toán các phép tính với những số nguyên có giá trị tùy ý. Tình hình cũng tương tự như vậy đối với các số thực. Vậy trong máy tính có những giới hạn nhất định trong việc biểu diễn các số, không phải số nào máy tính cũng biểu diễn được và tính toán được. Tuy nhiên, với những giới hạn như hiện nay, Fortran vẫn cho phép chúng ta lập các chương trình để tính toán, xử lý với tất cả những giá trị số gặp trong đời sống và khoa học kỹ thuật.

2.2. Hằng và biến

Máy tính xử lý dữ liệu hay thực hiện những tính toán với những đại lượng. Tất cả những đại lượng đó phải được lưu giữ trong máy tính. Những đại lượng không đổi trong suốt quá trình thực hiện của chương trình gọi là *các hằng*, còn những đại lượng có thể nhận những giá trị khác nhau gọi là *các biến*. Với mỗi hằng hoặc biến, trong bộ nhớ máy tính giành ra một địa chỉ để lưu giá trị. Tên chính là ký hiệu quy ước của địa chỉ đó.

2.2.1. Tên biến và tên hằng

Tên biến trong Fortran chuẩn được biểu diễn bằng tập hợp từ 1 đến 6 các chữ cái trong bảng chữ cái la tinh (26 chữ cái) hoặc các chữ số 0, 1, ..., 9, nhưng phải bắt đầu bằng chữ cái.

Trong một chương trình các tên biến không được trùng nhau. Trong các phiên bản Fortran hiện nay, để dùng làm tên không phân biệt chữ cái hoa và chữ cái thường. Ngoài ra, còn một vài ký tự khác cũng có thể dùng để cấu tạo tên. Phiên bản Fortran 90 cho phép đặt tên với số ký tự dài hơn 6 và trong tên có thể có một số ký tự khác nữa. Tuy nhiên, sinh viên nên tập thói quen đặt tên gọn gàng theo Fortran chuẩn, bởi vì tập hợp 6 ký tự đã rất đủ để chúng ta mô tả các bài toán, kể cả những bài toán lớn và phức tạp.

Thí dụ, các tên sau đây

X ; A ; X1 ; B2T5 ; SOHANG ; SUM là hợp lệ, còn các tên sau đây là sai:

1NGAY ; HE SO ; B*T

vì trong tên thứ nhất ký tự đầu tiên là chữ số, trong tên thứ hai có ký tự dấu cách, trong tên thứ ba có ký tự (*) không phải là những ký tự dùng để đặt tên.

Quy tắc đặt tên biến trên đây cũng áp dụng đối với tên chương trình, tên hằng, tên các chương trình con và tên file. (Riêng với tên file có thể có thêm phần mở rộng gồm không quá ba chữ cái hoặc chữ số ngăn với phần tên chính bởi dấu chấm).

2.2.2. Mô tả (khai báo) kiểu biến và kiểu hằng

Kiểu của biến tương ứng với kiểu dữ liệu mà nó biểu diễn. Các biến nguyên biểu diễn các dữ liệu số nguyên, các biến thực - số thực... Trong chương trình phải chỉ rõ các biến được sử dụng biểu diễn dữ liệu kiểu nào (nguyên, thực, lôgic, phức, văn bản, số thực độ chính xác thường hay độ chính xác gấp đôi...).

Mỗi biến chỉ lưu giữ được những giá trị đúng kiểu của nó. Một biến đã mô tả kiểu là số nguyên thì không thể dùng để lưu giá trị số thực hay giá trị lôgic.

Cách *mô tả ẩn* chỉ dùng đối với các biến nguyên và thực: dùng tên biến nguyên bắt đầu bằng một trong sáu chữ cái I, J, K, L, M, N, còn tên biến thực bắt đầu bằng một trong những chữ cái ngoài sáu chữ cái trên. Nói chung, người mới học lập trình không bao giờ nên dùng cách mô tả ẩn.

Cách *mô tả hiện* dùng các lệnh mô tả hiện như INTEGER, REAL, CHARACTER, LOGICAL, DOUBLE PRECISION, COMPLEX... để chỉ kiểu dữ liệu mà các biến biểu diễn. Dưới đây là quy tắc viết những lệnh mô tả kiểu dữ liệu: tuần tự nguyên, thực, lôgic, phức, thực độ chính xác gấp đôi và ký tự văn bản:

INTEGER ***D****anh sách các biến nguyên*

REAL ***D****anh sách các biến thực*

LOGICAL ***D****anh sách các biến lôgic*

COMPLEX ***D****anh sách các biến phức*

DOUBLE PRECISION ***D****anh sách các biến độ chính xác đôi*

CHARACTER ***D****anh sách các biến ký tự*

Trong danh sách các biến sẽ liệt kê các tên biến, nếu có hơn một biến thì các biến phải cách nhau bởi dấu phảy.

Thí dụ:

```
INTEGER I, TT, DEM
REAL X1, APSUAT, MAX, TIME, DELTA
COMPLEX P1, P2, SOPH
```

chỉ rằng các biến I, TT, DEM biểu diễn các giá trị số nguyên, các biến X1, APSUAT, MAX, TIME, DELTA biểu diễn các giá trị số thực, còn ba biến P1, P2, SOPH - số phức.

Những giá trị được giữ nguyên nhất quán trong suốt chương trình (tức các hằng số) thường được gán vào các địa chỉ nhớ thông qua tên trong lệnh khai báo hằng có dạng:

PARAMETER (*ten1 = biểu thức 1, tên 2 = biểu thức 2, ...*)

Thí dụ, trong chương trình nếu ta nhiều lần dùng đến giá trị số $\pi = 3{,}141593$ thì ta có thể gán giá trị 3,141593 cho một tên hằng là PI bằng lệnh

```
PARAMETER (PI = 3.141593)
```

Lệnh sau đây

```
PARAMETER (HSMSD = 0.0026, RO = 1.0028)
```

khai báo hai hằng số: HSMSD và RO, HSMSD được gán giá trị bằng 0,0026, còn RO được gán giá trị 1,0028.

Trong chương trình tất cả những lệnh khai báo (mô tả) vừa giới thiệu trên đây thuộc loại các lệnh không thực hiện và chúng phải nằm ở đầu chương trình, trước tất cả các lệnh thực hiện.

Khái niệm về tên, kiểu dữ liệu của biến, của hằng là những khái niệm cơ bản, quan trọng trong ngôn ngữ lập trình.

Ở đầu mục này đã nói một tên thực chất là ký hiệu quy ước của một

địa chỉ trong bộ nhớ của máy tính để lưu giá trị. Lệnh khai báo biến mới chỉ đặt tên cho một địa chỉ trong bộ nhớ và quy định trong địa chỉ đó có thể lưu giữ dữ liệu kiểu gì. Còn cụ thể trong ô nhớ đó đã có chứa giá trị chưa hay chứa giá trị bằng bao nhiêu thì tùy thuộc vào các lệnh thực hiện ở trong chương trình, tại từng đoạn của chương trình. Điều này giống như ta quy ước định ra một ngăn trong tủ văn phòng để chuyên giữ các công văn, còn trong ngăn ấy có công văn hay không, hoặc có mấy công văn thì tùy thuộc lúc này hay lúc khác. Dưới đây nêu một thí dụ để minh họa ý nghĩa của việc đặt tên biến và mô tả kiểu (dữ liệu) của biến, đồng thời theo dõi giá trị của biến tại từng thời điểm của chương trình. Giả sử ta viết một chương trình để tính diện tích s của hình tam giác khi giá trị độ dài đáy b bằng 5,0 cm, chiều cao h bằng 3,2 cm, in kết quả tính lên màn hình. Chương trình sau đây sẽ thực hiện những việc đó:

```
REAL DAY, CAO                                  ! (1)
DAY = 5.0                              ! (2)
CAO = 3.2                              ! (3)
DAY = 0.5 * DAY * CAO                          ! (4)
PRINT *, 'DIEN TICH TAM GIAC BANG', DAY ! (5)
END                                            ! (6)
```

Trong chương trình này có sáu lệnh. Lệnh (1) khai báo hai biến tên là DAY và CAO dự định để lưu giá trị số thực tương ứng của đáy b và chiều cao h của tam giác. Lệnh (2) gán giá trị $b = 5{,}0$ (cm) cho biến DAY. Lệnh (3) gán giá trị $h = 3{,}5$ (cm) cho biến CAO. Lệnh (4) tính giá trị của biểu thức $0{,}5 \times b \times h$, tức diện tích s của tam giác, bằng 8 (cm^2) và gán cho biến DAY. Lệnh (5) in lên màn hình dòng chữ DIEN TICH TAM GIAC BANG và sau đó là giá trị của biến DAY. Lệnh (6) là lệnh kết thúc chương trình. Sinh viên mới học lập trình thường có thể không hiểu lệnh thứ năm, khi thấy in diện tích hình tam giác mà lại in giá trị của biến DAY. Trong đầu họ quen nghĩ khai báo DAY có nghĩa DAY là độ dài cạnh đáy tam giác. Nhưng nếu hiểu được rằng lệnh (1) khai báo REAL DAY, CAO thực ra mới chỉ dự định dùng hai tên DAY và CAO để lưu các số thực, không cần biết số thực đó bằng bao nhiêu. Ở chương trình trên, khi lệnh (2) thực hiện xong thì trong biến DAY (trong ô nhớ có tên là DAY) mới thực sự có số 5,0, tức độ dài đáy tam giác. Nhưng khi chương trình chạy xong lệnh (4) thì trong biến DAY đã là số 8,0 chứ không phải là số 5,0 nữa. Và khi thực hiện xong lệnh (5) thì trên màn hình sẽ in đúng giá trị diện tích tam giác. Nắm vững được điều này có nghĩa là đã hiểu được ý nghĩa của biến, tên biến và tuần tự làm việc của chương trình, tức các giá trị được lưu trong máy tính như thế nào trong khi chương trình chạy.

Dưới đây là hai lời khuyên đầu tiên có lẽ quan trọng nhất đối với sinh viên mới học lập trình:

1) Sau khi tìm hiểu xong bài toán cần giải, phải cân nhắc từng đại lượng trong bài toán có kiểu dữ liệu là số nguyên, số thực, ký tự văn bản... để đặt tên và khai báo kiểu cho đúng. Kinh nghiệm cho thấy rằng sinh viên nào viết được những lệnh khai báo hệ thống các tên biến đúng, vừa đủ, sáng sủa trong phần khai báo ở đầu chương trình thì thường là sau đó viết được chương trình đúng. Còn những sinh viên không biết đặt tên cho các biến, vừa bắt tay vào soạn thảo chương trình đã loay hoay với lệnh mở file dữ liệu, tính cái này cái kia, thì thường là không hiểu gì và không bao giờ làm được bài tập.

2) Nên tuân thủ cách đặt tên của Fortran chuẩn. Ta có quyền chọn những chữ cái, chữ số nào để tạo thành tên là tùy ý, song nên đặt tên có tính gợi nhớ đến những đại lượng tương ứng trong bài tập. Thí dụ, với bài toán vừa nói tới trong mục này ta có ba đại lượng là: độ dài cạnh đáy, đường cao và diện tích tam giác. Nên khai báo tên ba biến tương ứng bằng ba từ tắt của tiếng Việt với lệnh sau:

```
REAL  DAY, CAO, DTICH
```

hoặc bằng ba từ tắt của tiếng Anh với lệnh:

```
REAL  BASE, HEIGHT, SQRE
```

hoặc bằng ba chữ cái đúng như trong đầu đề bài tập với lệnh:

```
REAL  B, H, S
```

đều là những lời khai báo đúng, dễ hiểu, trong đó lời khai báo trên cùng có lẽ là tốt nhất, lời khai báo sau cùng thì hơi quá ngắn gọn. Còn với cùng mục đích khai báo mà dùng lệnh sau đây thì mặc dù không sai, nhưng hoàn toàn không nên, rất dễ gây nhầm lẫn, mệt mỏi trong khi kiểm tra chương trình:

```
REAL  X, IC, DT
```

2.3. Biến có chỉ số (mảng)

2.3.1. Khái niệm mảng

Mảng là tập hợp có sắp xếp của các đại lượng được ký hiệu bằng một *tên* duy nhất. Các thành phần của tập hợp gọi là những *phần tử mảng*. Mỗi phần tử được xác định theo *tên* của mảng và vị trí của phần tử đó trong mảng, tức trị số của các *chỉ số*. Tên mảng được đặt tuân theo quy tắc như tên biến. Các chỉ số nằm trong dấu ngoặc đơn và nếu có hơn một chỉ số thì các chỉ số phải cách nhau bởi dấu phảy.

Thí dụ: A(1), A(2), A(3) tương ứng với cách viết thông thường cho các biến a_1, a_2, a_3 trong toán học. Vậy ở đây ta đã đặt cho tập hợp cả 3 giá trị này một tên chung là A, nhưng để chỉ giá trị thứ nhất ta thêm chỉ số 1 vào tên - A(1), để chỉ giá trị thứ hai ta thêm chỉ số 2 - A(2) và để chỉ giá trị thứ ba ta thêm chỉ số 3 - A(3).

Tương tự, các phần tử của ma trận hai chiều trong đại số

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$$

được viết trong Fortran là A(1,1), A(1,2), A(1,3), A(2,1), A(2,2), A(2,3) (chỉ số thứ nhất - số hiệu dòng, chỉ số thứ hai - số hiệu cột).

Thêm một thí dụ nữa về mảng. Một năm có 12 tháng, mỗi tháng có một tên, thí dụ trong tiếng Việt: Tháng Giêng, Tháng Hai, ..., Tháng Mười hai, trong tiếng Anh: January, February, ..., December. Ta hoàn toàn có thể gộp 12 tên tiếng Anh của các tháng trong năm vào thành một mảng có tên chung là EMONTH. Vậy mảng EMONTH sẽ là mảng có 12 giá trị (12 phần tử), mỗi phần tử là một từ chỉ tên một tháng. Khi nói đến January tức là nói tới giá trị thứ nhất của mảng EMONTH, ta viết EMONTH(1), nói đến December là nói tới giá trị thứ 12 của mảng EMONTH, ta viết EMONTH(12).

Trong Fortran IV, một phiên bản trước đây của ngôn ngữ Fortran, cho phép dùng các mảng tối đa 7 chỉ số. *Chiều* của mảng ứng với số chỉ số, còn *kích thước* của mảng ứng với số phần tử chứa trong mảng.

Chỉ số của mảng có thể được xác định bằng các hằng hoặc biến nguyên dương với trị số lớn hơn 0. Cũng có thể chỉ số xác định bằng biểu thức số học bất kỳ. Nếu dùng biểu thức kiểu thực, thì sau khi tính giá trị của biểu thức, giá trị số thực được chuyển thành số nguyên, tức cắt bỏ phần thập phân.

Trong mục 2.1 chúng ta đã nói về các kiểu dữ liệu cơ bản. Mỗi một biến kiểu dữ liệu cơ bản trong một thời điểm chạy chương trình chỉ lưu (chứa) được một giá trị. Bây giờ ta thấy mảng là một thí dụ về kiểu dữ liệu mới cấu tạo từ các kiểu cơ bản - một biến mảng trong một thời điểm có thể lưu được nhiều giá trị số nguyên, số thực, chuỗi ký tự... Nhưng cần lưu ý rằng tất cả các phần tử của mảng, tức tất cả các giá trị của mảng phải có cùng kiểu dữ liệu. Thí dụ với mảng EMONTH vừa xét, ta không thể đưa một giá trị ký tự January vào phần tử EMONTH(1) và số thực 1.27 vào EMONTH(2).

Mảng là một yếu tố rất quan trọng trong Fortran. Sau này ta sẽ thấy sử

dụng mảng trong ngôn ngữ lập trình có thể giúp viết những đoạn chương trình rất ngắn gọn, trong sáng. Đặc biệt trong các vòng lặp, chỉ bằng vài dòng lệnh có thể khiến máy tính thực hiện nhiều triệu phép tính số học.

2.3.2. Mô tả mảng

Mô tả mảng thực hiện ngay ở đầu chương trình và chứa thông tin về tên, chiều và kích thước mảng với toán tử DIMENSION:

DIMENSION $A(n_1, n_2, ..., n_\ell),\ MAT(m_1, m_2, ..., m_k)$

trong đó A, MAT - tên các mảng; $n_1, n_2, ..., n_\ell,\ m_1, m_2, ..., m_k$ – các giới hạn trên của các chỉ số – chỉ ra bằng *các hằng nguyên dương* (giới hạn dưới luôn bằng 1 và không cần chỉ định trong mô tả).

Theo mô tả này, máy tính sẽ giành trong bộ nhớ những vùng địa chỉ để lưu tất cả các phần tử của các mảng. Các phần tử của mảng nhiều chiều được lưu liên tiếp nhau sao cho chỉ số thứ nhất biến đổi nhanh nhất, chỉ số sau cùng biến đổi chậm nhất.

Có thể mô tả mảng bằng các lệnh mô tả kiểu hiện như đối với các biến thông thường, thí dụ:

```
REAL  MAX, L(7), A(20,21)
```

Trong lệnh mô tả này biến MAX được khai báo là biến số thực, có thể gọi là biến đơn, còn mảng L (biến có chỉ số) là mảng một chiều với 7 phần tử số thực, mảng A là mảng hai chiều (hai chỉ số) với giới hạn trên của chỉ số thứ nhất là 20, của chỉ số thứ hai là 21, nó gồm 420 phần tử.

Vì các giới hạn chỉ số (kích thước mảng) phải được chỉ định trước ở phần khai báo bằng các hằng nguyên dương, không thể là các biến, nên trong thực tiễn lập trình phải chú ý cân nhắc chọn các giới hạn chỉ số sao cho chúng không quá lớn làm tốn bộ nhớ, nhưng cũng phải vừa đủ để biểu diễn hết các phần tử có thể có của mảng. Thí dụ cần biểu diễn một bảng số các giá trị nhiệt độ trung bình từng tháng trong 100 năm thì ta khai báo mảng TEM(100,12) là hợp lý. Nếu dự định giải hệ phương trình đại số tuyến tính không quá 20 phương trình, ta nên khai báo các mảng REAL A(20,21), X(20) là vừa đủ để biểu diễn ma trận các hệ số $a_{i,j}$ (kể cả các hệ số tự do) và các nghiệm x_i. Với mảng EMONTH vừa nhắc trong mục này thì lệnh khai báo sau:

```
CHARACTER*9 EMONTH(12)
```

là hoàn toàn hợp lý vì một năm chỉ có 12 tháng và tên tháng dài nhất (với tiếng Anh) là September gồm 9 chữ cái.

2.4. Các hàm chuẩn

Một số phép tính như lấy căn bậc hai của một số, tính trị tuyệt đối của một số, tính hàm sin của một góc... thường xuyên gặp trong nhiều thuật toán, nên được xây dựng sẵn thành các hàm gọi là các hàm riêng có của Fortran (intrinsic functions) hay còn gọi là *các hàm chuẩn*.

Bảng 2.1 liệt kê một số hàm chuẩn của Fortran thường dùng trong sách này.

Mỗi hàm chuẩn có một tên của nó. Tên của hàm được tiếp nối với đầu vào, gọi là *đối số* của hàm, nằm trong cặp dấu ngoặc đơn. Đối số của các hàm chuẩn có thể là các hằng, biến, hay biểu thức. Nếu một hàm có nhiều đối số thì các đối số được viết cách nhau bằng dấu phảy. Khi cho các giá trị cụ thể vào các đối số thì hàm tính ra một giá trị của hàm. Vì vậy các hàm thường dùng để tính một giá trị nào đó để gán vào một biến khác, người ta gọi là gọi hàm ra để tính. Hàm không bao giờ có mặt ở bên trái dấu ' = ' của lệnh gán.

Thí dụ, những lệnh sau đây gọi các hàm để tính một số giá trị:

```
S = SIN (0.5)
TG = TAN (S)
C = COS (ANGLE * 3.141593 / 180.0)
```

Bảng 2.1. Một số hàm chuẩn của Fortran

Tên hàm và đối số	Giá trị hàm
SQRT (X)	$\sqrt{x}$ Căn bậc hai của x
ABS (X)	$\lvert x \rvert$ Trị tuyệt đối của x
SIN (X)	$\sin(x)$ x tính bằng radian
COS (X)	$\cos(x)$ x tính bằng radian
TAN (X)	$\mathrm{tg}(x)$ x tính bằng radian
EXP (X)	e^x e nâng lên luỹ thừa x
LOG (X)	$\ln(x)$ Logarit tự nhiên của x
LOG10 (X)	$\lg(x)$ Logarit cơ số 10 của x
INT (X)	Chuyển phần nguyên của số thực x thành số nguyên
REAL (I)	Giá trị thực của I (chuyển một giá trị nguyên thành giá trị thực)
MOD (I,J)	Lấy phần dư nguyên của phép chia hai số I / J

Trong lệnh thứ nhất ta gửi giá trị hằng 0,5 (radian) cho đối số của hàm SIN để nó tính ra giá trị sin của góc 0,5 và gán giá trị đó cho biến S. Trong lệnh thứ hai, ta đã gửi giá trị của biến S vào đối số của hàm TAN để tính ra tang. Còn trong lệnh thứ ba, ta đã gửi một biểu thức vào đối số của hàm COS để nó tính ra giá trị côsin của một góc có độ lớn bằng giá trị của biểu thức đó. Trong trường hợp này, máy tính trước hết phải tính (ước lượng) giá trị của biểu thức đối số, sau đó mới tính côsin theo giá trị nhận được.

Thấy rằng một hàm biểu diễn một giá trị. Giá trị này có thể được dùng trong các tính toán khác hoặc lưu ở địa chỉ nhớ khác. Một hàm chuẩn cũng có thể làm đối số của một hàm chuẩn khác:

```
XLG = LOG(ABS(X))
```

Trong Fortran có một số hàm chuẩn cho ra giá trị với kiểu cùng kiểu với đối số của mình, chúng được gọi là *các hàm tự sinh* (*generic function*). Thí dụ hàm ABS(X), nếu đối số X là số nguyên thì giá trị hàm ABS(X) cũng là số nguyên, nếu X là số thực - ABS(X) cũng là số thực. Một số hàm chỉ định kiểu của đầu vào và đầu ra. Thí dụ hàm IABS là hàm đòi hỏi đối số nguyên và cho ra giá trị tuyệt đối là số nguyên. Danh sách đầy đủ hơn về các hàm chuẩn của Fortran được dẫn trong phụ lục 1.

Khi dùng một hàm chuẩn nào đó phải đọc kỹ lời mô tả xem nó tính ra giá trị gì, điều kiện của các đối số ra sao. Thí dụ các hàm lượng giác phải dùng đối số là radian, nếu ta cho giá trị đối số là độ thì kết quả tính sẽ sai.

2.5. Lệnh gán và các toán tử số học

2.5.1. Lệnh gán

Các tính toán trong Fortran có thể chỉ định bằng lệnh gán với dạng tổng quát như sau:

Tên biến = *Biểu thức*

Bên trái dấu lệnh gán (dấu =) là tên một biến. Biểu thức bên phải có thể là một hằng, một biến, một biểu thức số học gồm các toán tử số học (bảng 2.2) thực hiện giữa các toán hạng là các hằng, biến và hàm chuẩn hay một biểu thức lôgic. Khi thực hiện lệnh gán, trước hết máy ước lượng (tính) giá trị của biểu thức bên phải, rồi gán giá trị đó cho biến bên trái, tức lưu giá trị tính được của biểu thức bên phải vào địa chỉ nhớ có tên biến bên

trái. Kiểu dữ liệu của biến và của biểu thức phải phù hợp.

Thí dụ các lệnh gán:

```
PI = 3.141593
S = PI * BKINH **2
I = I +1
```

Lệnh thứ nhất gán hằng số 3,141593 cho biến có tên là PI. Lệnh thứ hai gán giá trị của biểu thức $PI \times (BKINH)^2$ cho biến có tên là S. Lệnh thứ ba lấy giá trị hiện tại của biến I cộng thêm một đơn vị và lại gán cho chính biến I.

Ở trên đã nói, kiểu dữ liệu của biến và của biểu thức phải phù hợp. Trường hợp biến bên trái là biến thực, còn biểu thức bên phải là giá trị nguyên thì máy tính sẽ chuyển giá trị nguyên đó thành giá trị thực (số thực với phần thập phân bằng không) rồi mới gán cho biến. Khi biến bên trái là biến nguyên, biểu thức bên phải có giá trị thực, thì máy tính cắt bỏ phần thập phân của giá trị thực, đổi số thực nhận được thành số nguyên rồi mới gán nó cho biến nguyên. Các trường hợp gán sai khác chương trình dịch sẽ báo lỗi.

Không nên quan niệm lệnh gán như dấu bằng trong toán học.

2.5.2. Các phép tính số học đơn giản

Các phép tính số học hay còn gọi là các toán tử số học gồm có các phép tính cộng, trừ, nhân, chia và nâng lên luỹ thừa được ký hiệu bằng các toán tử trong Fortran như trong bảng 2.2.

Gọi là những phép tính số học bởi vì các toán hạng của các phép tính là những giá trị số, thí dụ số nguyên, số thực, số phức. Sau này chúng ta sẽ thấy máy tính có thể tính toán với những giá trị kiểu khác như giá trị lôgic, giá trị văn bản...

Bảng 2.2. Các phép tính số học

Phép tính	Dạng đại số	Trong Fortran
Cộng	$A + B$	A + B
Trừ	$A - B$	A – B
Nhân	$A \times B$	A * B
Chia	$\frac{A}{B}$	A / B
Luỹ thừa	A^3	A** 3

2.5.3. Ước lượng biểu thức số học

Khi tính giá trị của biểu thức số học, nếu biểu thức đó gồm nhiều phép tính đơn, thì máy sẽ tính toán từng phép tính đơn để nhận các kết quả trung gian, sau đó tính giá trị cuối cùng của biểu thức gọi là ước lượng. Mức ưu tiên khi ước lượng giá trị của một biểu thức số học gồm nhiều phép tính đơn nêu trong bảng 2.3.

Nếu dấu âm đứng trước tên biến đầu tiên trong biểu thức, thì nó được tính với cùng mức ưu tiên như phép trừ. Thí dụ: –A**2 bằng –(A**2), –A*B bằng –(A*B) và –A+B bằng (–A)+B.

Bảng 2.3. Mức ưu tiên các phép tính số học

Ưu tiên	Phép tính
1	Dấu ngoặc
2	Nâng lên luỹ thừa
3	Nhân và chia
4	Cộng và trừ

Khi các phép tính ở cùng mức ưu tiên thì tất cả các phép tính được thực hiện từ trái sang phải, thí dụ:

B – C + D được ước lượng bằng (B – C) + D

Riêng phép nâng lên luỹ thừa thì thực hiện từ phải sang trái:

A ** B ** C được ước lượng bằng A ** (B ** C)

Thí dụ: 2**3**2 bằng 2^9 hay 512 chứ không phải là

$$(2**3)**2 = 8^2 = 64 .$$

2.5.4. Khái niệm về cắt và các phép tính hỗn hợp

Khi một phép tính số học thực hiện với hai số thực thì đưa ra kết quả là giá trị thực. Thí dụ, khi tính chu vi hình tròn với đường kính DKINH là số thực, ta có thể dùng một trong hai lệnh sau:

```
CHUVI = PI * DKINH
CHUVI = 3.141593 * DKINH
```

Phép tính số học giữa hai số nguyên cho ra kết quả là số nguyên. Thí dụ, cho hai số nguyên I và J, trong đó I nhỏ hơn hoặc bằng J, tính số số nguyên INTERV nằm trong khoảng [I, J] có thể thực hiện bằng lệnh:

```
INTERV = J – I + 1
```

Giả sử SIDE biểu diễn giá trị thực và LENGTH biểu diễn giá trị nguyên. Bây giờ xét lệnh:

```
LENGTH = SIDE * 3.5
```

Phép tính nhân giữa hai giá trị thực sẽ cho kết quả số thực. Tuy nhiên, giá trị thực được lưu vào biến nguyên. Khi đó máy tính sẽ bỏ qua phần thập phân và chỉ lưu phần nguyên của số thực; kiểu làm tròn này gọi là *cắt,* nó khác với làm tròn thông thường cho kết quả là số nguyên gần nhất với giá trị của số thực.

Khi các phép tính số học thực hiện giữa các biến có kiểu khác nhau (hỗn hợp) thường cho kết quả rất bất ngờ. Ta xét thí dụ tính thể tích V của hình cầu bán kính thực R. Nếu dùng lệnh:

```
V = (4/3)*3.141593*R**3
```

ta sẽ thu được kết quả sai do nguyên nhân phép chia hai số nguyên 4/3 cho giá trị trung gian bằng 1, không phải 1,333333. Do đó, lệnh đúng để tính V sẽ là:

```
V = (4./3.)*3.141593*R**3
```

Vì các phép tính hỗn hợp đôi khi cho kết quả bất ngờ, ta nên cố gắng tránh dùng những biểu thức số học có phép tính hỗn hợp.

2.5.5. Khái niệm về số quá bé và số quá lớn (underflow và overflow)

Vì các giá trị lớn nhất và bé nhất có thể lưu trong một biến tuỳ thuộc vào chính hệ máy tính, một phép tính có thể đưa ra kết quả quá lớn hoặc quá bé. Xét các thí dụ sau:

```
1)      X = 0.25E20          2)      A = 0.25E–20
        Y = 0.10E30                  B = 0.10E+20
        Z = X * Y                    C = A / B
```

Kết quả số của phép nhân trong thí dụ 1 bằng 0.25E49, rõ ràng là có thể quá lớn, không lưu giữ được trong máy tính với bậc cực đại là 38, còn kết quả số của phép chia trong thí dụ 2 bằng 0.25E–49 sẽ quá bé. Trong những trường hợp này các lệnh Fortran hoàn toàn đúng, nhưng lỗi sẽ phát sinh khi chạy chương trình. Các lỗi do bậc quá lớn hoặc quá bé thường bị gây bởi những lỗi ở những đoạn trước của chương trình, thí dụ một biến chưa được gán giá trị đúng lại có mặt trong biểu thức số học.

Bài tập

1. Hãy biểu diễn thành dạng F và dạng E những số thực sau:

a) 3,14 b) 3,141593 c) 0,0026 d) $2,5 \times 10^3$

e) $-14{,}0$ f) 28,34 g) $6{,}023 \times 10^{23}$

2. Xác định những tên sai trong những tên sau đây:

a) AVERG b) PTBACHAI c) REAL

d) 2LOG đ) GPTB2 e) HS-A1

f) X1 g) THANG*1 h) MONTH2

3. Viết thành dạng Fortran những biểu thức tính sau đây:

a) Thể tích V của hình cầu theo công thức

$$V = \frac{4}{3}\pi R^3 \quad (R - \text{ bán kính}).$$

b) Hai nghiệm x_1 và x_2 của phương trình bậc hai

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (a, b, c - \text{ các hệ số của phương trình})$$

c) Giá trị hàm $y = \frac{1}{2}\sin^2 x \cos(2x - ½)$ (khi x cho bằng độ).

d) Giá trị hàm mật độ phân bố Gauss $F(x) = \frac{1}{2\pi} e^{-\frac{x^2}{2}}$

e) Thêm một đơn vị vào biến nguyên I và lưu vào biến I

f) Khoảng cách DIST giữa hai điểm A và B nếu biết các toạ độ tương ứng của hai điểm đó là (x_a, y_a), (x_b, y_b).

4. Ước lượng giá trị của các biểu thức Fortran sau đây:

```
4/3*3.141593*(3/2)**3
SQRT (I+I/2)    (nếu I = 1)
SIN ((30/180)*PI)  (nếu PI = 3.141593)
COS (60/180*3.141593)
```

5. Hãy đọc chính xác bằng ngôn ngữ Fortran những lệnh viết dưới đây:

a) I = I + K +1

b) SS = 0.5 * SIN (A*3.1416 / 180.)

c) ERR = ABS (X1 - X2)

6. Hai đoạn chương trình sau nhằm tính trị số trung bình A của ba số nguyên $i_1 = 1$, $i_2 = 2$, $i_3 = 3$ và in kết quả lên màn hình. Hãy thử xem kết quả có đúng không. Nếu thấy sai thì chỉ ra tại sao và khắc phục bằng cách nào?

a)
```
      I1 = 1
      I2 = 2
      I3 = 3
      PRINT  4 , (I1 + I2 + I3)/3
    4 FORMAT (1X, ' A = ',F4.1)
```

b)
```
      I1 = 1
      I2 = 2
      I3 = 3
      PRINT  2, 1/3*(I1+I2+I3)
    2 FORMAT (3X, F4.1)
```

7. Giả sử các cung địa lý (tám cung) được đánh số hiệu theo qui ước như sau: 1 - bắc; 2 - đông bắc; 3 - đông; 4 - đông nam; 5 - nam; 6 - tây nam; 7 - tây; 8 - tây bắc. Hướng gió quan trắc được bằng 165^o. Hãy viết biểu thức Fortran để tính số hiệu cung của hướng gió đó.

Chương 3 - Nhập và xuất dữ liệu đơn giản

3.1. Các lệnh xuất và nhập dữ liệu

Máy tính có thể nhập dữ liệu từ các nguồn, các thiết bị khác nhau. Tương tự, ta cũng có thể hướng sự xuất dữ liệu ra các thiết bị khác nhau. Trong chương này, ta xét cách nhập dữ liệu từ bàn phím và xuất dữ liệu ra màn hình hoặc máy in. Việc xuất và nhập dữ liệu có dùng các file dữ liệu sẽ được xét trong chương 6.

Lệnh xuất dữ liệu định hướng ra màn hình:

PRINT * , *Danh sách các mục in*

Lệnh nhập dữ liệu từ bàn phím:

READ * , *Danh sách các biến*

Các mục in trong lệnh in có thể là một hằng, một biến, một biểu thức. Nếu trong danh sách các mục in có từ hai mục trở lên, thì các mục phải cách nhau bởi dấu phảy. Trong danh sách các biến của lệnh nhập (đọc) dữ liệu, nếu có hơn một biến cần đọc dữ liệu, thì những biến đó phải được liệt kê cách nhau bởi dấu phảy. Các mục được in ra trên một dòng màn hình theo thứ tự được liệt kê trong danh sách. Nếu trong danh sách không có một mục in nào, thì máy tính chỉ đơn giản là xuống một dòng trên màn hình. Thí dụ, xét đoạn chương trình sau đây:

```
GOC = 30.0
PRINT* , ' Khi  X = ', GOC , ' 1/2 SinX = ', 0.5 * SIN (GOC *
* 3.141593 / 180.)
```

Ghi chú: *Trong lệnh* PRINT *vừa rồi có một dấu nối dòng. Ở đây đã dùng dấu hoa thị bên trong hình nhữ nhật nhỏ* * *để phân biệt với dấu hoa thị bình thường là ký hiệu của phép tính nhân. Từ nay về sau trong sách này ở những dòng lệnh nào có dấu nối dòng sẽ quy ước dùng ký hiệu này. Còn khi soạn chương trình trên màn hình máy tính, thì như đã nói trong mục 1.5, chúng 1 ta chỉ cần viết dấu hoa thị vào vị trí thứ 6 của dòng lệnh.*

Ta thấy trong danh sách các mục in của lệnh PRINT có 4 mục liệt kê theo thứ tự là:

1) Cụm chữ ' Khi X ='

2) Biến có tên là GOC lưu giá trị 30°

3) Cụm chữ ' 1/2 SinX ='

4) Biểu thức

0.5 * SIN (GOC * 3.141593 / 180.0)

biểu thị nửa sin của góc 30° đã đổi thành radian.

Như vậy, mục in thứ nhất và thứ ba là những hằng văn bản, mục in thứ 2 là giá trị của biến số thực GOC và mục in thứ tư là một biểu thức số thực. Trước khi in mục thứ tư, máy tính phải tính giá trị của biểu thức này (bằng 0,25), rồi sau đó mới in giá trị đó lên màn hình. Kết quả trên màn hình sẽ như sau:

```
Khi X =     30.00000     1/2  SinX =        0.2500000
```

Hãy chú ý rằng với lệnh PRINT * trên đây các mục in là những cụm

dữ liệu văn bản được in ra đúng như ta nhìn thấy trong dòng lệnh, từng ký tự một, kể cả dấu trống. Các giá trị của biến và biểu thức thực được in ra sau một khoảng trống và số những chữ số có nghĩa sau dấu chấm thập phân khác nhau. Nếu giá trị của các biến là những số khá nhỏ hoặc khá lớn, thì máy sẽ in ra những giá trị đó dưới dạng biểu diễn E hoặc D (xem mục 2.1). Kiểu in dữ liệu như trên gọi là in không được định dạng hay in không có format.

Chú ý rằng, trong READ *, sau dấu phảy là danh sách các biến, khi thực hiện lệnh này, máy tính chờ ta gõ từ bàn phím những giá trị (các ký tự văn bản, số nguyên, số thực...) *tương xứng về kiểu với danh sách biến, mỗi giá trị cách nhau một dấu phảy hay ít nhất một dấu trống, riêng những ký tự văn bản phải nằm trong cặp dấu nháy trên* (' '). Kết thúc danh sách các giá trị phải gõ lệnh phím Enter (↵). Máy tính sẽ tuần tự gán những giá trị nhận từ bàn phím vào những biến tương ứng trong danh sách biến của lệng READ. Nếu ta gõ chưa đủ số giá trị theo danh sách biến, thì máy chờ ta gõ cho đến khi đủ các giá trị mới kết thúc thực hiện lệnh READ. Nếu kiểu dữ liệu gõ vào sai so với kiểu dữ liệu của biến, thì lập tức chương trình ngừng thực hiện và báo lỗi chạy chương trình. Thí dụ lệnh

READ * , I , NAM , TEMP , GHICHU

đòi hỏi ta gõ vào từ bàn phím một số nguyên cho biến I, một số nguyên nữa cho biến NAM và một số thực cho biến TEMP, một xâu ký tự cho biến văn bản GHICHU, muốn nhập đúng yêu cầu ta có thể gõ vào bàn phím như sau:

1 1982 25.36 'SL quan trac' ↵ hay 1,1982,25.36,'SL quan trac' ↵

Lệnh in có quy cách (có định dạng):

PRINT *k* **,** ***Danh sách các mục in***

Cũng giống như lệnh in không định dạng, danh sách các mục in chỉ ra những hằng, biến hay các biểu thức cần in theo thứ tự liệt kê. Tham số *k* nguyên dương chỉ tới nhãn của lệnh FORMAT mô tả quy cách in thông tin ra màn hình như vị trí in, khoảng cách giữa các mục in, số chữ số thập phân cần in đối với giá trị số thực... Dạng tổng quát của lệnh FORMAT như sau:

k **FORMAT (*Danh sách các đặc tả*)**

trong đó *k* là *nhãn* của dòng lệnh FORMAT. Danh sách các *đặc tả* nằm trong cặp dấu ngoặc đơn báo cho máy tính biết về cách dãn dòng theo chiều thẳng đứng và bố trí các ký tự trong dòng thông tin in ra. Nếu in ra trên giấy máy in, thì *tuỳ chọn dãn dòng* báo cho máy in điều khiển kéo giấy để in sang đầu trang mới, xuống dòng mới, xuống hai dòng mới hay in ngay trên dòng hiện thời... Máy tính sẽ thiết lập mỗi dòng in bên trong bộ nhớ trước khi thực sự in dòng đó lên giấy. Vùng bộ nhớ bên trong đó gọi là *vùng đệm buffer*. Những ký tự đầu tiên trong vùng buffer gọi là ký tự điều khiển kéo giấy của máy in có những ý nghĩa như sau:

1	Sang trang mới
Dấu trống	Xuống một dòng
0	Xuống hai dòng
+	Không xuống dòng

Bây giờ ta làm quen với những đặc tả đơn giản sau đây. Trong chương 4 sẽ còn trở lại vấn đề định dạng phức tạp hơn khi làm việc với các file dữ liệu.

3.2. Các đặc tả trong lệnh FORMAT

1) ***Các đặc tả văn bản*** thường dùng để xuất dữ liệu là những ký tự, các đoạn văn bản, hay dùng in tiêu đề các báo cáo. Đặc tả văn bản cho phép đưa các ký tự trực tiếp vào buffer. Các ký tự phải nằm trong cặp dấu nháy trên hay dấu ngoặc kép. Thí dụ:

```
    PRINT 4
4   FORMAT ('1' , 'KET QUA THI NGHIEM')
```

Ta cũng có thể dùng *đặc tả* w**H** trong đó $w-$ số vị trí để xuất dữ liệu văn bản. Thí dụ

```
    PRINT 5 , Y
5   FORMAT (16H  TICH  PHAN  BANG , F9.3)
```

2) ***Đặc tả*** n**X** sẽ chèn n dấu trống vào bản ghi, thường dùng để căn giữa các tiêu đề báo cáo, thí dụ:

```
    PRINT 35
35  FORMAT ('1' , 25X , 'THI NGHIEM SO 1')
```

3) ***Đặc tả*** **A** w dùng cho các hằng và biến xâu ký tự, các thông tin văn bản, tuỳ chọn w báo cho máy tính số vị trí giành cho một biến xâu ký tự (văn bản) cần in. Mục văn bản in ra căn lề bên phải.

4) ***Đặc tả*** **I** w dùng cho số nguyên, trong đó w số vị trí dùng để in giá trị số nguyên.

5) ***Đặc tả*** **F** $w.d$ dùng biểu diễn dạng thập phân của số thực, $w-$ tổng số vị trí dành cho số thực kể cả dấu chấm thập phân, $d-$ số chữ số thập phân sau dấu chấm. Trong hai đặc tả I w và F $w.d$ các số in ra được căn lề bên phải. Nếu đặc tả thiếu vị trí để biểu diễn giá trị, thì giá trị số sẽ không được in ra, mà tại các vị trí in sẽ xuất hiện các dấu sao (*) để báo hiệu cho ta biết rằng đặc tả của lệnh FORMAT không phù hợp, cấp thiếu vị trí so với giá trị của đại lượng cần in.

6) ***Đặc tả*** **E** $w.d$ dùng ghi ra dưới dạng luỹ thừa những giá trị rất lớn hoặc rất nhỏ và khi ta chưa hình dung rõ về độ lớn của đại lượng.

Thông thường hai lệnh PRINT và FORMAT đi kèm gần nhau. Thí dụ:

```
    PRINT 5 , I, NAM, TEMP, GHICHU
5   FORMAT (1X, I3, I8, F10.2, 1X, A20)
```

Sau lệnh READ và các dữ liệu được gõ vào từ bàn phím đã nói trong mục 3.2.1, thì kết quả cặp lệnh in này trên màn hình sẽ như sau:

```
1    1982    25.36  SL quan trac
```

7) ***Các đặc tả*** **A** w**,** **I** w , **F** $w.d$ ***và*** **E** $w.d$ cũng dùng với lệnh đọc số liệu.

8) ***Đối với các giá trị lôgic*** trong Fortran dùng đặc tả **L** w, trong đó $w-$ số vị trí giành cho dữ liệu. Thí dụ theo lệnh

```
16  FORMAT (L5)
```

nếu tại một trong 5 vị trí giành cho biến có chữ **T**, thì giá trị **.TRUE.** sẽ được gán vào biến lôgic trong lệnh đọc. Khi xuất, chữ cái **T** hoặc **F** (tương ứng với **.TRUE.** hoặc **.FALSE.** sẽ in ra tại vị trí thứ 5, tức vị trí cuối cùng bên phải trong 5 vị trí.

9) Chúng ta có thể sử dụng một số đặc điểm bổ sung trong cách viết các đặc tả của lệnh FORMAT nhằm nâng cao chất lượng bản ghi, theo đúng ý định biểu diễn của mình hay làm cho lệnh FORMAT trông ngắn gọn. Có thể dùng những cách dưới đây:

Cách viết lặp lại các đặc tả**:** Thí dụ những cặp lệnh sau đây hoàn toàn

tương đương:

```
10  FORMAT (3X, I2, 3X, I2)
10  FORMAT (2 (3X, I2))
20  FORMAT (1X, F4.1, F4.1, 1X, I3, 1X, I3, 1X, I3)
20  FORMAT (1X, 2F4.1, 3 (1X, I3))
```

Dùng dấu gạch chéo (/) trong lệnh FORMAT chỉ kết thúc dòng in trước khi bắt đầu các đặc tả sau nó. Thí dụ, khi cần in dòng tiêu đề của một bảng số cùng với những tiêu đề cột, chúng ta có thể dùng:

```
      PRINT 5
5     FORMAT (1X, 'KET QUA QUAN TRAC' // 2X, 'Gio', 3X,
     * 'Toc do', 3X, 'Huong')
```

Sau khi in xong đoạn văn bản KET QUA QUAN TRAC, dấu gạch chéo thứ nhất chỉ dẫn cho máy kết thúc dòng, xuống dòng mới, dấu gạch chéo thứ hai chỉ dẫn bỏ qua ngay dòng này không in, phát sinh ra một dòng trống trước khi in các tiêu đề cột ở dòng thứ ba như ta thấy dưới đây:

```
KET QUA QUAN TRAC

Gio     Toc do     Huong
```

Dùng đặc tả bảng **T, TR, TL** để căn lề trái các tiêu đề cột một bảng số. Thí dụ các cặp lệnh cùng nhãn sau đây sẽ là tương đương với nhau:

```
600   FORMAT (F6.1, 15X, I7)
600   FORMAT (F6.1, T22, I7)
```

ở lệnh thứ hai: sau khi ghi ra số thực với 6 vị trí, nhảy ngay tới vị trí 22 để bắt đầu ghi số nguyên.

```
85  FORMAT (1X, 25X, 'Do cao', 5X, 'Huong')
```

```
85  FORMAT (T27, 'Do cao', TR5, 'Huong')
```

ở lệnh thứ hai: nhảy ngay tới vị trí thứ 27 để ghi tiêu đề "Do cao", sau đó do có đặc tả TR5 xuất phát từ vị trí hiện thời sẽ nhảy sang phải 5 vị trí để ghi tiêu đề "Huong".

Đặc tả \ có tác dụng ngăn không xuống dòng trong một lệnh in hoặc đọc. Có thể dùng đặc tả này trong trường hợp muốn viết một lời nhắc yêu cầu người dùng nhập thông tin từ bàn phím nhưng sau khi viết lời nhắc thì không xuống dòng, con nháy đứng trên cùng dòng ngay sau lời nhắc chờ người dùng nhập thông tin từ bàn phím theo yêu cầu của lệnh đọc. Thí dụ nhóm lệnh sau đây sẽ làm chức năng đó:

```
      PRINT  7
7     FORMAT (1X, 'Ten file so lieu:  ', \)
      READ (*,  '(A50)')  NAME
```

Về số lượng các đặc tả: Khi số các đặc tả nhiều hơn số mục trong danh sách các mục in, thí dụ:

```
      PRINT  1, TOCDO, KHOANG
1     FORMAT (4 F5.2)
```

máy sẽ chọn lấy số tối đa các đặc tả cần dùng, số đặc tả còn lại bị bỏ qua. Trong trường hợp này lệnh in có 2 mục in - 2 giá trị số thực, nhưng lệnh FORMAT có 4 đặc tả số thực, như vậy số đặc tả là thừa. Máy sẽ chọn lấy hai đặc tả và in bình thường như chúng ta mong muốn.

Khi số đặc tả ít hơn số mục in, thí dụ trong lệnh in sau:

```
      PRINT  20, TEM, VOL
20    FORMAT (1X, f6.2)
```

Trong trường hợp này máy căn các mục in và đặc tả cho đến hết danh sách đặc tả, sau đó có thể xảy ra hai khả năng:

1) In luôn buffer hiện tại và bắt đầu một buffer mới.

2) Quay trở lại đầu danh sách đặc tả cho đến khi gặp dấu ngoặc đơn trái và lại căn từng cặp mục in, đặc tả cho các mục in còn lại.

Trong lệnh in trên giá trị của TEM được căn theo đặc tả F6.2. Vì không có đặc tả cho VOL nên ta làm như sau:

1) In giá trị của TEM sau một vị trí trống.

2) Khi quay trở lại về phía đầu của danh sách các đặc tả (dấu ngoặc trái) và căn F6.2 cho giá trị VOL. Sau đó ta đạt tới đầu của danh sách và dấu trống để in VOL. Do đó TEM và VOL được in trên hai dòng riêng biệt.

Trong Fortran 90 cho phép các tham số độ rộng đặc tả, số lần lặp của đặc tả có thể là biến. Thí dụ FORMAT (<*m*>F<*n*>.<*k*>).

Bài tập

1. Viết đoạn chương trình đọc giá trị vào hai biến thực A và B, đổi giá trị của hai biến đó cho nhau.

2. Điều gì sẽ xảy ra khi thực hiện chương trình sau và ta nhập vào bàn phím lần lượt số 1, dấu phảy, số 10 và dấu chấm rồi gõ phím Enter.

```
      PRINT *, ' Cho cac gia tri cua hai so nguyen I1, I2 ! '
      READ * , IDAU, ICUOI
      PRINT 4, IDAU, ICUOI
4     FORMAT (1X, 'I1 = ', I5,  'I2 = ', I5)
      END
```

3. Mô tả những gì sẽ in lên màn hình khi thực hiện các lệnh dưới đây:

```
      REAL X
      X = –27.632
      PRINT 5 , X
5     FORMAT (1X, 'X = ', F7.1, ' DEGREES')
```

4. Mô tả những gì sẽ in ra máy in khi thực hiện những lệnh dưới đây:

```
      A = 3.184
      PRINT 1
1     FORMAT (1X, '0')
      PRINT 2
2     FORMAT ('+' , '-  =  ' , F5.2)
```

5. Mô tả những gì sẽ in lên màn hình khi thực hiện các lệnh dưới đây:

```
      REAL DIST , VEL
      DIST = 28732.5
      VEL = –2.6
      PRINT 10,  DIST, VEL
10    FORMAT (1X, 'DISTANCE = ' , E10.3,
     * 5X , 'VELOCITY = ' , F5.2)
```

6. Viết đoạn chương trình nhập vào từ bàn phím tên 5 môn thi của học kỳ cùng với điểm thi từng môn của mình. Tính điểm trung bình và in lên màn hình thành một bảng có hình thức như sau:

```
              BANG DIEM THI HOC KY
-------------------------------------------
TT        TEN MON HOC              DIEM
-------------------------------------------
 1        Tên môn thứ 1              8
 2        Tên môn thứ 2              7
 3        Tên môn thứ 3              8
 4        Tên môn thứ 4              7
 5        Tên môn thứ 5              9
-------------------------------------------
          Diem trung binh hoc ky     7,8
```

Chương 4 - Các cấu trúc điều khiển

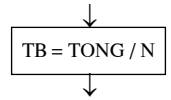
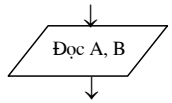
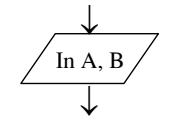
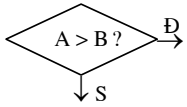
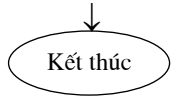
Trong các chương trước ta đã xét một vài chương trình đơn giản. Thấy rằng những chương trình này thực sự rất đơn giản, chỉ gồm một vài lệnh thực hiện tuần tự là dẫn đến kết quả bài toán cần giải. Trong chương này, sẽ giới thiệu *những lệnh của Fortran cho phép ta điều khiển được thứ tự các bước cần thực hiện*. Sự điều khiển được thực hiện thông qua những lệnh cho phép ta chọn những nhánh khác nhau trong chương trình và những lệnh cho phép ta lặp lại những phần nào đó của chương trình. Những lệnh như vậy gọi là những lệnh điều khiển.

4.1. Khái niệm về cấu trúc thuật toán

4.1.1. Các thao tác cơ bản. Giả trình và lưu đồ

Trong mục 1.3, chương 1 đã sơ lược nói về quy trình năm bước giải bài toán. Đối với những bài toán phức tạp về cách giải thì bước 4 là bước khó khăn nhất. Người lập trình phải mô tả tuần tự các công đoạn từ đầu đến cuối quá trình giải, chia quá trình này thành một số khối và liệt kê những khối đó ra để sau này chương trình máy tính sẽ tuần tự thực hiện. Trong mỗi khối người lập trình lại phải chi tiết hoá thêm đến mức có thể chuyển thành những lệnh máy tính. Cách chia khối và chi tiết hoá từng khối như vậy có thể gọi là *phương pháp chia và chinh phục*. Kết quả cuối cùng của chia khối và chi tiết hoá từng khối chính là thuật giải (algorithm).

Bảng 4.1. Các thao tác cơ bản và quy ước tương ứng trong giả trình và lưu đồ

Dạng thao tác	Chú giải giả trình	Biểu tượng lưu đồ
Tính toán	*TB* ← *TONG* / *N*	
Nhập dữ liệu	Đọc A, B	
Xuất dữ liệu	In A, B	
So sánh	Nếu A > B	
Bắt đầu thuật giải	Tên bài toán	
Kết thúc thuật giải		

Những hình thức để biểu diễn trực quan thuật giải sao cho dễ dàng chuyển thành chương trình là giả trình và lưu đồ. Một người lập trình có thể chọn hình thức này hoặc hình thức kia. Theo cách *giả trình*, mỗi cấu trúc của thuật giải được quy ước bởi một chú giải ngắn gọn gần giống với ngôn ngữ viết của chúng ta; còn trong cách biểu diễn *lưu đồ*, mỗi cấu trúc đó được mô tả bằng một biểu tượng hình học.

Dần dần ta sẽ thấy rằng, nói chung những thao tác cơ bản trong một thuật giải thường là những tính toán, nhập, xuất dữ liệu và so sánh. Nói chung một chương trình máy tính dù đơn giản hay phức tạp đến đâu cũng chỉ gồm có những thao tác cơ bản đó. Một số thao tác (hay lệnh) có thể nhóm lại với nhau tạo thành một khối hay một khối cấu trúc. Những chú giải giả trình và những biểu tượng lưu đồ chính là để thể hiện những thao tác cơ bản đó (xem bảng 4.1).

4.1.2. Các cấu trúc tổng quát trong thuật giải

Các bước trong một thuật giải có thể phân chia thành ba dạng cấu trúc tổng quát - đó là cấu trúc tuần tự, lựa chọn và lặp. *Cấu trúc tuần tự* là chuỗi các bước thực hiện một cách kế tiếp nhau. *Cấu trúc lựa chọn* (hay còn gọi là *cấu trúc rẽ nhánh*) cho phép so sánh hai giá trị, sau đó tuỳ kết quả so sánh mà định ra một chuỗi các bước khác nhau phải thực hiện. *Cấu trúc lặp* được dùng khi quá trình giải cần lặp lại một số thao tác cho đến khi thoả mãn một điều kiện. Trong thuật giải phức tạp hơn một chút có thể thấy các cấu trúc tổng quát này lồng vào nhau, trong cấu trúc lặp có những đoạn gồm những thao tác tuần tự được thực hiện, có những đoạn xuất hiện sự rẽ nhánh tuỳ theo một điều kiện so sánh nào đó.

4.1.3. Thí dụ ứng dụng thuật toán cấu trúc

Bây giờ ta tìm hiểu phương pháp xây dựng thuật giải theo kỹ thuật chia khối và chi tiết hoá từng khối, phân tích cấu trúc thuật giải thông qua một thí dụ cụ thể về bài toán phân tích các số liệu thực nghiệm.

1) Phát biểu bài toán: Xác định giá trị lớn nhất, nhỏ nhất và biên độ các giá trị của tập số liệu quan trắc.

2) Mô tả dữ liệu vào và ra: Dữ liệu vào là một chuỗi các số liệu quan trắc. Đầu ra là trị cực đại, cực tiểu và biên độ các giá trị.

3) Tính thử với tập số liệu quan trắc sau:

Chuỗi số liệu thử:

40.56
55.92
66.31
58.35
62.88
41.99
49.70
53.21

Thực hiện tìm trị cực đại như sau: Trước hết so sánh số thứ nhất của chuỗi với số thứ hai để xác định số lớn hơn, coi là cực đại tạm thời. Bây giờ xét số thứ ba và so sánh nó với cực đại tạm thời. Nếu cực đại tạm thời lớn hơn, ta xét tới số thứ tư; nhưng nếu số thứ ba lớn hơn cực đại tạm thời, ta thay thế số đó vào cực đại tạm thời. Tiếp tục quá trình này với toàn bộ chuỗi số liệu sẽ dẫn tới kết quả là cực đại tạm thời chính là trị cực đại trong cả chuỗi. Một quá trình tương tự sẽ cho phép tìm cực tiểu. Với tập số liệu đang xét, kết quả là:

Giá trị cực đại = 66.31
Giá trị cực tiểu = 40.56

Tính biên độ bằng hiệu giữa cực đại và cực tiểu = 66.31 – 40.56 = 25.73

4) Xây dựng thuật giải: Khái quát lại các bước thực hiện ở bước (3) ta có thể chia bài toán thành ba khối:

- Đọc số liệu và xác định các trị cực đại và cực tiểu
- Tính hiệu giữa cực đại và cực tiểu để nhận biên độ
- In cực đại, cực tiểu và biên độ

Với thí dụ này, ta chi tiết hoá cách giải bằng giả trình. Rõ ràng khối thứ nhất đòi hỏi phải chi tiết hoá nhiều hơn nữa, vì nó vừa bao gồm cả việc

chọn trị cực đại, cực tiểu xuất phát, vừa bao gồm cả quá trình lặp (lặp để đọc số liệu và lặp để cập nhật cực trị khi cần). Cực đại và cực tiểu xuất phát thường được gán bằng giá trị của quan trắc thứ nhất, do đó ta đọc một số liệu đầu để gán cho chúng. Sau đó ta đọc số thứ hai và đi vào vòng lặp. "Chừng nào số không phải là zero", ta cập nhật trị cực đại và cực tiểu nếu cần thiết. Bây giờ ta mô tả những bước đã đủ chi tiết này bằng giả trình như sau:

```
Giả trình:      Đọc số
                Cực đại  ← Số
                Cực tiểu ← Số
                Đọc số
                Chừng nào số không bằng zero thì
                        Nếu số > Cực đại thì
                        Cực đại  ← Số
                        Nếu số < Cực tiểu thì
                        Cực tiểu ← Số
                        Đọc số
                Biên độ  ← Cực đại – Cực tiểu
                In 'GIA TRI CUC DAI = ',  Cực đại
                In 'GIA TRI CUC TIEU = ',  Cực tiểu
                In 'BIEN DO GIA TRI  = ',  Biên độ
```

Đây là một thuật giải đơn giản. Chỉ có một khối thứ nhất cần chi tiết hoá. Thấy rằng khi thuật giải đã chi tiết hoá tới mức như vậy, thì việc chuyển thành chương trình Fortran sẽ không còn là vấn đề khó khăn. Trong các mục tiếp sau, ta sẽ nghiên cứu các lệnh Fortran chuyên trợ giúp cho việc thiết kế các cấu trúc điều khiển của bài toán này và nhiều bài toán tương tự.

4.2. Cấu trúc IF và các lệnh tương ứng

4.2.1. Biểu thức lôgic

Biểu thức lôgic được tạo bởi các *toán tử quan hệ*:

.EQ.	bằng	**.NE.**	không bằng
.LT.	nhỏ hơn	**.LE.**	nhỏ hơn hoặc bằng
.GT.	lớn hơn	**.GE.**	lớn hơn hoặc bằng

nối hai biến số ở hai bên.

Tuỳ theo quan hệ giữa hai biến số đó mà biểu thức lôgic có một trong hai giá trị lôgic:

đúng (**.TRUE.**) hoặc sai (**.FALSE.**).

Thí dụ, xét biểu thức **A .EQ. B** trong đó **A** và **B** là các biến số thực. Nếu giá trị của **A** bằng giá trị của **B** thì biểu thức lôgic sẽ có giá trị là đúng **.TRUE.**. Nếu không thì biểu thức có giá trị là sai **.FALSE.**. Tương tự, nếu **X** bằng 4,5 thì biểu thức **X .GT. 3.0** có giá trị bằng đúng **.TRUE.**.

Ta có thể nối hai biểu thức lôgic bằng một trong các *toán tử lôgic* **.OR.** và **.AND.** thành một *biểu thức lôgic kết hợp*.

Khi hai biểu thức lôgic nối với nhau bởi **.OR.** thì biểu thức lôgic kết hợp sẽ có giá trị là đúng nếu một hoặc cả hai biểu thức có giá trị là đúng. Ta có thể gọi **.OR.** là toán tử cộng lôgic.

Khi hai biểu thức nối với nhau bởi **.AND.** thì biểu thức kết hợp có giá trị đúng chỉ khi cả hai biểu thức có giá trị là đúng. Ta có thể gọi toán tử **.AND.** là toán tử nhân lôgic.

Toán tử **.NOT.** có thể đứng trước biểu thức lôgic và đổi giá trị của nó thành giá trị ngược lại. Thí dụ, nếu **A. GT. B** là đúng (giá trị bằng **.TRUE.**) thì **.NOT. A. GT. B** là sai (**.FALSE.**).

Một biểu thức lôgic có thể chứa nhiều toán tử lôgic, thí dụ như trong biểu thức sau:

.NOT. (A **.LT.** 15.4) **.OR.** KT **.EQ.** ISUM

Quyền ưu tiên từ cao nhất xuống thấp nhất là

.NOT., **.AND.** và **.OR.**

Trong biểu thức trên, biểu thức A .LT. 15.4 sẽ được ước lượng trước tiên, sau đó giá trị của nó (.TRUE. hoặc .FALSE.) được đổi ngược lại. Giá trị này sẽ được xét cùng với giá trị của KT .EQ. ISUM. Thí dụ, nếu A là 5.0, KT là 5 và ISUM là 5, thì biểu thức bên trái của toán tử .OR. có giá trị sai .FALSE., biểu thức bên phải có giá trị đúng .TRUE. và toàn bộ biểu thức sẽ có giá trị là đúng .TRUE..

Giá trị của biểu thức lôgic có thể được gán cho biến lôgic bằng lệnh gán giống như lệnh gán dùng với các biến số và biểu thức số, thí dụ:

```
LOGICAL  DONE, OK
DONE = .FALSE.
OK = DONE .AND. I .GT. 24
```

Khi so sánh hai biểu thức lôgic hay hai biến lôgic có tương đương nhau hay không, trong Fortran không dùng các toán tử quan hệ như khi so sánh hai biểu thức số, mà dùng các toán tử lôgic **.EQV.** và **.NEQV.**.

Bảng 4.2. tóm tắt quy tắc ước lượng của các toán tử lôgic cho mọi trường hợp có thể xảy ra.

Bảng 4.2. Các toán tử lôgic

A	B	.NOT. A	A.AND.B	A.OR.B	A.EQV.B	A.NEQV.B
False	False	True	False	False	True	False
False	True	True	False	True	False	True
True	False	False	False	True	False	True
True	True	False	True	True	True	False

Khi các toán tử số học, quan hệ và lôgic cùng có mặt trong một biểu thức thì các toán tử số học thực hiện trước tiên; sau đó các toán tử quan hệ dùng để phát sinh các giá trị TRUE hoặc FALSE; và các giá trị này được đánh giá bằng các toán tử lôgic theo thứ tự ưu tiên .NOT., .AND., và .OR.. Các quan hệ .EQV. và .NEQV. được thực hiện sau cùng.

4.2.2. Lệnh IF lôgic

1) Các lệnh IF lôgic có thể có một số dạng sử dụng. Dạng thứ nhất gọi là ***Logical IF*** viết như sau:

IF (*Biểu thức lôgic*) *Lệnh thực hiện*

Theo lệnh này, nếu biểu thức lôgic ở trong cặp dấu ngoặc đơn có giá trị **True** thì thực hiện lệnh nằm trên cùng dòng với biểu thức lôgic, nếu biểu thức lôgic có giá trị **False** thì không thực hiện lệnh cùng dòng mà chuyển ngay tới lệnh tiếp theo phía dưới trong chương trình. Chú ý rằng lệnh thực hiện ghi sau biểu thức lôgic có thể là một trong những lệnh tính toán (gán), xuất, nhập dữ liệu..., nhưng không thể là một lệnh **IF** khác. Biểu thức lôgic bao giờ cũng phải đặt trong cặp dấu ngoặc đơn. Thí dụ, những lệnh IF sau đây là những lệnh đúng:

```
IF (A. GT. 0.0)  SUM = SUM + A
IF (TIME .GT. 1.5)  READ *, DIST
```

2) Dạng thứ hai gọi là ***Block IF***: Nếu biểu thức lôgic có giá trị True máy thực hiện các lệnh từ lệnh 1 đến lệnh n, sau đó chuyển tới lệnh tiếp sau END IF. Nếu biểu thức lôgic có giá trị False, điều khiển chuyển ngay xuống lệnh đứng sau END IF:

```
IF (Biểu thức lôgic) THEN
        lệnh 1
        . . .
        lệnh n
END IF
```

3) Dạng thứ ba gọi là dạng ***IF – ELSE***: Khi biểu thức lôgic có giá trị True các lệnh từ 1 đến n được thực hiện, nếu biểu thức lôgic có giá trị False các lệnh từ $n+1$ đến m được thực hiện:

```
IF (Biểu thức lôgic) THEN
        lệnh 1
        ...
        lệnh n
ELSE
        lệnh n+1
        ...
        lệnh m
END IF
```

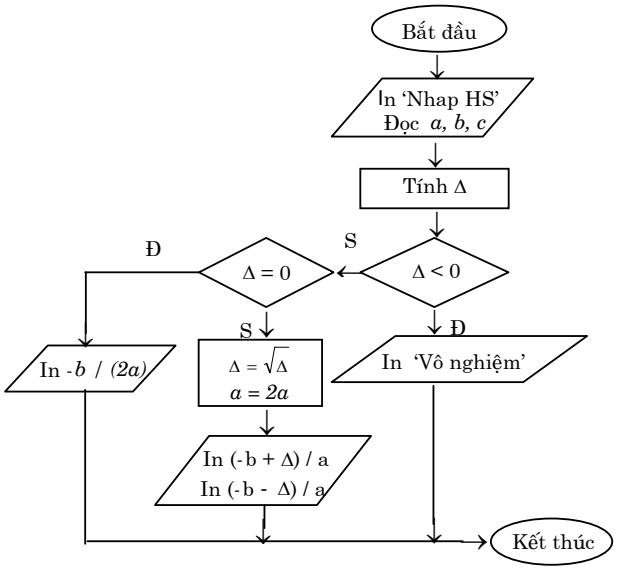
4) Dạng thứ tư gọi là ***IF – ELSE IF***: Nếu biểu thức lôgic 1 có giá trị True thì loạt các lệnh từ 1 đến m được thực hiện; nếu biểu thức lôgic 1 có trị False, biểu thức lôgic 2 có trị True thì loạt lệnh từ $m+1$ đến n thực hiện; nếu các biểu thức lôgic 1 và 2 là False và biểu thức lôgic 3 True thì các lệnh từ $n+1$ tới p thực hiện. Nếu không một biểu thức lôgic nào có giá trị True thì chỉ có các lệnh từ $p+1$ tới q được thực hiện. Trong thực tế ta có thể cấu tạo số nhánh **ELSE IF** nhiều hơn hoặc ít hơn, chứ không nhất thiết chỉ là hai nhánh như đã viết dưới đây:

```
IF (Biểu thức lôgic 1) THEN
        lệnh 1
        ...
        lệnh m
ELSE IF (Biểu thức lôgic 2) THEN
        lệnh m+1
        ...
        lệnh n
ELSE IF (Bbiểu thức lôgic 3) THEN
        lệnh n+1
        ...
        lệnh p
ELSE
        lệnh p+1
        ...
        lệnh q
END IF
```

Thí dụ 1: Sử dụng các lệnh IF lôgic để điều khiển rẽ nhánh. Lập chương trình giải hệ phương trình bậc hai

$ax^2 + bx + c = 0$ (các hệ số a, b, c nhập từ bàn phím, $a \neq 0$).

Ta có thể cụ thể hóa thuật giải của bài toán này bằng lưu đồ như trên hình 4.1. Từ đó viết mã nguồn của chương trình Fortran như dưới đây.



Hình 4.1. Lưu đồ thuật giải bài toán của thí dụ 1

```
      PRINT * , ' HE SO A BANG'
      READ * , A
      PRINT * , ' HE SO B BANG'
      READ * , B
      PRINT * , ' HE SO C BANG'
      READ * , C
      DELT = B**2 – 4.*A*C
      IF (DELT .LT. 0.) THEN
         PRINT * , ' PHUONG TRINH VO NGHIEM'
      ELSE  IF (DELT .EQ. 0.) THEN
         PRINT 5 , –B / (2.0 *A)
5     FORMAT (1X, 'NGHIEM KEP BANG' , F10.2)
      ELSE
         DELT = SQRT (DELT)
         A = 2. * A
         PRINT 7 , (–B + DELT) / A , (–B – DELT) / A
7     FORMAT (1X, 'HAI NGHIEM:  X1 = ',
     *  F10.2, 5X, 'X2 = ', F10.2)
      END IF
      END
```

4.2.3. Lệnh IF số học

Lệnh ***IF số học*** cho phép thực hiện rẽ nhánh chương trình thành ba nhánh tùy thuộc vào giá trị của biểu thức số học, dạng tổng quát của lệnh này viết như sau:

IF (*Biểu thức số học*) n_1, n_2, n_3

trong đó n_1, n_2, n_3 – nhãn của các lệnh thực hiện. Nếu biểu thức số học có giá trị âm thì điều khiển được chuyển tới lệnh có nhãn là n_1, bằng không – nhãn n_2, và dương – nhãn n_3.

Thí dụ, theo lệnh

```
IF  (I  – 10)  4, 8, 7
```

nếu $I < 10$ điều khiển chuyển đến lệnh có nhãn là 4, nếu $I = 10$ – chuyển đến nhãn 8 và nếu $I > 10$ – chuyển đến nhãn 7.

Trong lệnh

```
IF  (X – 3.5)  3, 6, 6
```

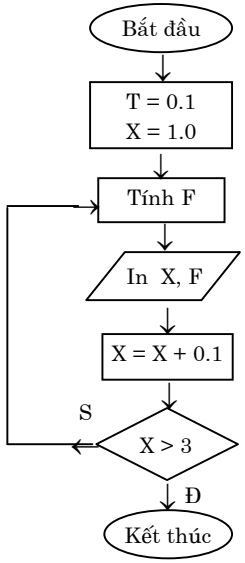
khi $X \geq 3{,}5$ điều khiển chuyển tới lệnh có nhãn là 6, khi $X < 3{,}5$ điều khiển chuyển tới lệnh có nhãn là 3.

Thí dụ 2: Dùng lệnh* IF *số học để thiết kế vòng lặp. Viết chương trình tính và in giá trị hàm

$$f(x) = e^{-x^3} \cos\ (t\,x + 1),$$

trong đó x biến thiên từ 1 đến 3 với bước 0,1 và $t = 0{,}1$.

Lưu đồ giải bài toán này tham khảo trên hình 4.2.



Hình 4.2. Lưu đồ thuật giải bài toán của thí dụ 2

```
      T = 0.1
      X = 1.0
12    F = EXP (– X ** 3) * COS (T * X + 1)
      WRITE (6 , 9)  X , F
9     FORMAT (F5.2, E12.2)
      X = X + 0.1
      IF (X – 3.0)  12 , 12 , 4
4     STOP
      END
```

4.2.4. Lệnh chuyển điều khiển vô điều kiện GO TO

Lệnh này có dạng

GO TO *n*

trong đó *n* – nhãn của lệnh mà điều kiển cần chuyển tới.

Lệnh cần chuyển tới nhất thiết phải có nhãn. Ngoài ra trong chương trình không thể có những lệnh có cùng nhãn như nhau. Lệnh GO TO có thể chuyển điều khiển tới bất kỳ lệnh thực hiện nào đứng trước hoặc đứng sau lệnh GO TO. Thí dụ:

```
      GO TO  5                7    I = I + 1
      . . .                        X (I)=Y (I)
5     X = X + 1.0                  GO TO  7
```

Thí dụ 3: Viết chương trình nhập *n* phần tử của mảng một chiều X, sắp xếp lại các phần tử mảng đó theo thứ tự tăng dần và in ra màn hình.

```
      REAL  X (20), TG
      INTEGER  N, I, J, K
      N = 10
      PRINT * , 'NHAP CAC PHAN TU MANG'
      I = 0
7     I = I + 1
      PRINT *, 'PHAN TU ', I
      READ *, X (I)
      IF (I .LT. N)  GOTO 7
C  Sắp xếp mảng X theo thứ tự tăng dần
      I = 1
2     K = I
      J = I + 1
1     IF (X (J) .LT. X (K))  K = J
      J = J + 1
      IF (J .LE. N)  GOTO 1
      TG = X(I)
      X(I) = X(K)
      X(K) = TG
      I = I + 1
      IF (I. LT. N)  GOTO 2
C  Lần lượt in các giá trị của mảng X đã sắp xếp
      I = 1
3     PRINT 8 , X(I)
8     FORMAT (F12.2)
      I = I + 1
      IF (I .LE. N) GOTO 3
      END
```

4.2.5. Lệnh GO TO tính toán

Lệnh GO TO tính toán dùng để thực hiện chuyển điều khiển tới một trong số những lệnh có nhãn được liệt kê trong lệnh GOTO tuỳ thuộc vào giá trị của một biến trong lệnh. Dạng tổng quát của lệnh như sau:

GO TO $(n_1, n_2, ..., n_m)$ **,** i

trong đó $n_1, n_2, ..., n_m$ – các nhãn của những lệnh thực hiện, i – biến nguyên không chỉ số. Theo lệnh này, điều khiển được chuyển tới một trong các lệnh $n_1, n_2, ..., n_m$ tuỳ thuộc vào giá trị của i, cụ thể khi $i = 1$ điều khiển sẽ chuyển tới lệnh có nhãn n_1, khi $i = 2$ – nhãn n_2, ..., khi $i = m$ – nhãn n_m. Nếu giá trị của i nằm ngoài khoảng $1 \le i \le m$ thì điều khiển chuyển xuống lệnh đứng sau lệnh GO TO để thực hiện.

Thí dụ, theo lệnh

```
GO TO (17 , 2 , 115 , 19) ,  KA
```

khi KA = 1 điều khiển chuyển tới lệnh có nhãn là 17, khi KA = 2 điều khiển chuyển tới lệnh có nhãn là 2, khi KA = 3 điều khiển chuyển tới lệnh có nhãn là 115 và khi KA = 4 điều khiển chuyển tới lệnh có nhãn là 19.

Thí dụ 4: ***Ứng dụng lệnh GOTO tính toán***. Viết chương trình tính giá trị của đa thức Lejandre với $x = 0{,}4$ theo công thức

$$P_\ell(x) = \begin{cases} 1 & \text{khi } \ell = 0 \\ x & \text{khi } \ell = 1 \\ \frac{1}{2}(3x^2 - 1) & \text{khi } \ell = 2 \\ \frac{1}{2}(5x^3 - 3x) & \text{khi } \ell = 3 \end{cases}$$

```
      REAL  X, P
      INTEGER L, I
```

```
      X = 0.4
      L = 0
28    I = L + 1
      GO TO (12, 17, 21, 6) ,  I
12    P = 1.0
      GO TO 24
17    P = X
      GO TO 24
21    P = 0.5 * (3.0 * X ** 2 – 1.0)
      GO TO 24
6     P = 0.5 * (5.0 * X ** 3 – 3.0 * X)
24    WRITE (* , 8)  L , P
8     FORMAT (I3 , F12.5)
      L = L + 1
      IF (L – 3)  28 , 28 , 30
30    STOP
      END
```

Thí dụ 5: ***Sắp xếp danh sách***. Viết chương trình nhập họ tên và điểm ba môn học của nhóm gồm n sinh viên. Tính điểm trung bình cộng ba môn học. In bảng có tiêu đề và các cột thứ tự, họ tên, điểm ba môn và điểm trung bình, ghi chú xếp loại theo điểm trung bình: trung bình < 6.0, khá $6 \div 8,9$, giỏi $> 9,0$. Danh sách xếp theo thứ tự từ cao xuống thấp dựa theo điểm trung bình.

```
      PARAMETER (N = 15)
      INTEGER  I , J , K , D1 (50) , D2 (50) , D3 (50), ID
      REAL  D , TB (50)
      CHARACTER * 20  TEN (50) , TENG
C  Nhập họ tên, điểm thi và tính điểm trung bình
      I = 0
7     I = I + 1
      PRINT * , ' NHAP SINH VIEN ' , I
      READ (* , '(A20)')  TEN(I)
      READ * , D1 (I) , D2 (I) , D3 (I)
      TB (I) = (D1 (I) + D2 (I) + D3 (I)) / 3.0
      IF (I .LT. N)  GO TO 7
C  Sắp xếp danh sách theo thứ tự điểm trung bình giảm dần
      I = 1
2     K = I
      J = I + 1
1     IF (TB(J) .GT. TB(K))  K = J
      J = J + 1
      IF (J .LE. N)  GO TO 1
      TENG = TEN (I)
      TEN (I) = TEN (K)
      TEN (K) = TENG
      ID = D1 (I)
      D1 (I) = D1 (K)
      D1 (K) = ID
      ID = D2 (I)
      D2 (I) = D2 (K)
      D2 (K) = ID
```

```
      ID = D3 (I)
      D3 (I) = D3 (K)
      D3 (K) = ID
      D = TB (I)
      TB (I) = TB (K)
      TB (K) = D
      I = I + 1
      IF (I .LT. N)  GO TO 2
C  In tiêu đề danh sách lên màn hình
      PRINT 100
100   FORMAT (21X , 'BANG DIEM' // , 1X , 'TT' , 7X,
     *  'HO TEN' ,9X , 'D1 D2 D3   TB XEP LOAI' /)
C  In từng sinh viên theo danh sách
 60   FORMAT (1X, I2, 1X, A20, I3, I3, I3, F5.1, 1X, 'GIOI')
 50   FORMAT (1X, I2, 1X, A20, I3, I3, I3, F5.1, 1X, 'KHA')
 40   FORMAT (1X, I2, 1X, A20, I3, I3, I3, F5.1, 1X,
     *  'TRUNG BINH')
      I = 1
  3   IF (TB (I) .LT. 9.0) THEN
        IF (TB (I) .LT. 6.0) THEN
          PRINT 40 , I , TEN (I) , D1 (I) , D2 (I) , D3 (I) , TB (I)
        ELSE
          PRINT 50 , I , TEN (I) , D1 (I) , D2 (I) , D3 (I) , TB (I)
        END IF
      ELSE
        PRINT 60 , I , TEN (I) , D1 (I) , D2 (I) , D3 (I) , TB (I)
      END IF
      I = I + 1
      IF (I. LE. N)  GO TO 3
      STOP
      END
```

Thí dụ 6: Viết ***chương trình tính tích phân xác định***:

$$I = \int_a^b x^2 \sin x$$

theo công thức hình thang với sai số $\varepsilon = 0{,}0001$; $a, b -$ cho trước.

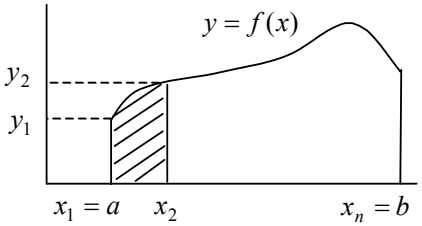
Gợi ý: Ở bước xấp xỉ đầu, xem số hình thang con $n = 1$, tích phân bằng

$$S_1 = 0{,}5\,(y_a + y_b)\,(b - a)\,.$$

Bước xấp xỉ sau tăng số hình thang con n thêm 1 và tích phân bằng (hình 4.3)

$$S_2 = \sum_i^n 0{,}5\,(y_i + y_{i+1})\,(x_{i+1} - x_i)$$

Tiếp tục tăng n đến khi $\left| S_1 - S_2 \right| < \varepsilon$.



Hình 4.3. Minh họa sơ đồ tính gần đúng tích phân xác định theo phương pháp hình thang

```
      EPSIL = 0.0001
      A = 0.0
      B = 3.141593
      S1 = 0.5 * (A ** 2 * SIN (A) + B ** 2 * SIN (B)) * (B–A)
      SOHINH = 2.0
7     DX = (B–A) / SOHINH
      HINH = 1.0
      X1 = A
      Y1 = X1 ** 2 * SIN (X1)
      S2 = 0.0
5     X2 = X1 + DX
      Y2 = X2 ** 2 * SIN (X2)
      S2 = S2 + 0.5*(Y1 + Y2) * DX
      IF (HINH .LT. SOHINH) THEN
         HINH = HINH + 1.0
         X1 = X2
         Y1 = Y2
         GOTO 5
      END IF
      IF (ABS (S2–S1) .GT. EPSIL) THEN
         SOHINH = SOHINH + 1.0
         S1 = S2
         GOTO 7
      END IF
      PRINT 3 , S2
3     FORMAT (1X , 'TICH PHAN BANG', F15.4)
      END
```

Thí dụ 7: ***Vòng lặp để tính tổng chuỗi***. Bình phương của sin của góc x tính theo công thức chuỗi như sau:

$$\sin^2 x = x^2 - \frac{2^3 x^4}{4!} + \frac{2^5 x^6}{6!} - \ldots = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}\, 2^{2n-1}\, x^{2n}}{(2n)!}.$$

Hãy viết chương trình đọc vào một góc x bằng độ, đổi ra rađian, tính và in ra bảng so sánh kết quả tính $\sin^2 x$ theo công thức này với những số số hạng chuỗi n lẻ từ 1 đến 15. Thấy rằng số hạng đầu khi $n = 1$ là x^2, mỗi số hạng tiếp sau bằng số hạng trước nhân với $\dfrac{-2x^2}{n(2n-1)}$.

Trong thí dụ này, ta ứng dụng phương pháp chia khối bài toán và chi tiết hoá từng khối như đã trình bày trong mục 4.1 để xây dựng thuật giải và diễn đạt thuật giải đó bằng lưu đồ, sau đó dẫn chương trình Fortran.

Thấy rằng bài toán có thể chia thành ba khối sau:

Khối 1: Nhập giá trị góc x.

Khối 2: In tiêu đề của bảng kết quả.

Khối 3: Tính giá trị $\sin^2 x$ theo công thức chuỗi và in ra kết quả khảo sát với số số hạng chuỗi từ 1 đến 15.

Bây giờ ta phân tích chi tiết từng khối để dẫn lưu đồ thực hiện trong mỗi khối.

Thấy rằng khối 1 có thể chi tiết hoá thành ba bước con: Vì công thức khai triển chuỗi trên đây hội tụ nhanh đối với những góc x nhỏ, do đó nếu x nằm trong khoảng:

$$90 < x \le 180 \text{ ta thay bằng góc } 180 - x,$$

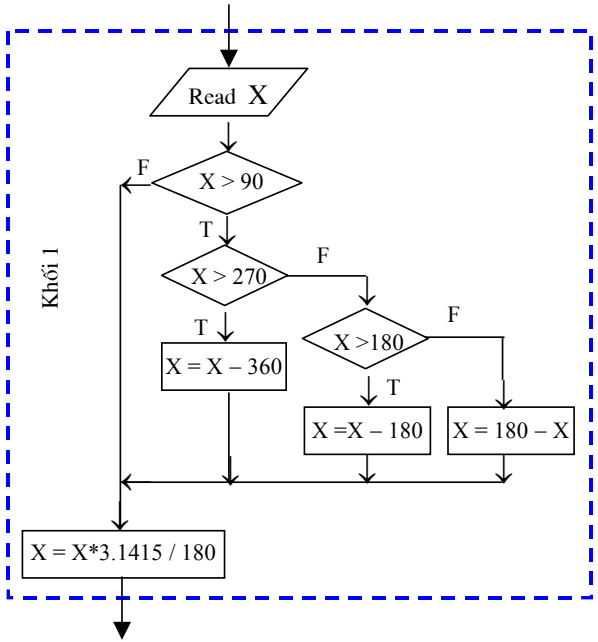
nếu x nằm trong khoảng:

$$180 < x \le 270 \text{ ta thay bằng góc } x - 180,$$

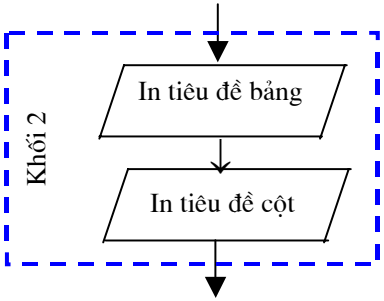
nếu x nằm trong khoảng:

$$270 < x \le 360 \text{ ta thay bằng góc } x - 360.$$

Sau đó đổi x thành radian (hình 4.4).

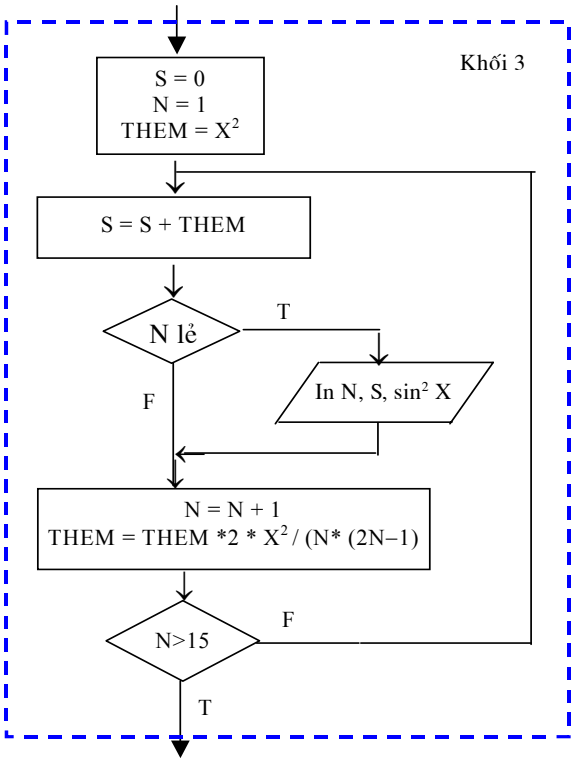


Hình 4.4. Lưu đồ khối 1 (thí dụ 7)



Hình 4.5. Lưu đồ khối 2 (thí dụ 7)

Ta thấy khối 2 chỉ gồm hai việc tuần tự là in dòng tiêu đề của bảng khảo sát, in các tiêu đề đầu bảng (hình 4.5).



Hình 4.6. Lưu đồ khối 3 (thí dụ 7)

Khối 3 là phức tạp nhất cần được chi tiết hoá một cách tối đa. Ta thấy khối này gồm các bước cụ thể sau:

- Gán 0 cho biến S (giá trị khởi tạo của $\sin^2 x$ cần tính).
- Gán 1 cho N (bắt đầu xét số hạng thứ nhất).
- Gán x^2 cho biến THEM (giá trị của số hạng thứ nhất).
- Chừng nào $N \le 15$ thực hiện tuần tự 4 bước sau:
  - ♦ Cộng số hạng (THEM) vào biến S.
  - ♦ Nếu N lẻ in giá trị N, S, $\sin^2 x$ (tính theo hàm chuẩn).
  - ♦ Tăng thêm 1 đơn vị cho N.
  - ♦ Tính lại biến THEM bằng cách nhân chính nó với $\frac{-2X^2}{N(2N-1)}$.

Giả trình này tương đương với lưu đồ khối trên hình 4.6.

Như vậy, ta đã chi tiết hoá tất cả các bước trong ba khối dưới dạng các lưu đồ. Công việc còn lại đơn giản là gắn cơ học ba lưu đồ lại ta được lưu đồ chung của toàn thuật toán. Từ đó dễ dàng chuyển sang chương trình Fortran dưới đây:

```
PRINT * , ' HAY CHO MOT GOC BANG DO'
READ *, X
IF (X .GT. 90.0) THEN
   IF (X .GT. 270.0) THEN
      X = X – 360.0
   ELSE  IF (X .GT. 180.0) THEN
      X = X – 180.0
   ELSE
      X = 180.0 – X
   END IF
END IF
```

```
      X = X * 3.141593 / 180.0
      PRINT 2
2     FORMAT (1X, 35H KHAO SAT CONG THUC BINH
     *  PHUONG SIN // , 1X , 2H N, 17H   THEO CONG THUC,
     *  17H   THEO HAM CHUAN)
      S = 0.
      N = 1
      THEM = X ** 2
5     S = S + THEM
      IF (MOD (N , 2) .EQ. 1)  PRINT 4 , N , S , SIN (X) ** 2
4     FORMAT (1X , I2 , 2F17.7)
      N = N + 1
      THEM = – THEM * 2.0 * X**2 / (N * (2 * N –1))
      IF (N .LE. 15) GO TO  5
      END
```

Thí dụ 8: Nội suy tuyến tính chuỗi số liệu quan trắc. Giả sử có những số liệu quan trắc về nhiệt độ nước biển tại các tầng sâu ở điểm có tọa độ 120°KĐ-20°VB được cho trong bảng 4.3. Lập chương trình nhập những số liệu này và nội suy giá trị nhiệt độ cho một độ sâu bất kỳ nhập từ bàn phím, thông báo lên màn hình kết quả nội suy dưới dạng như sau:

DO SAU = ... M

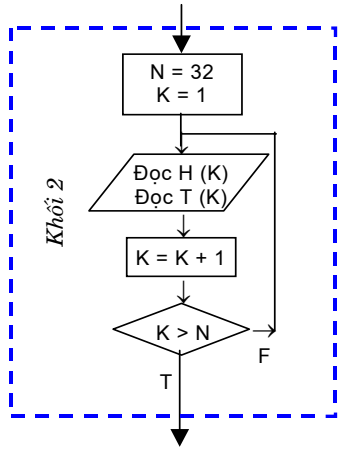
NHIET DO = ... DO C

Phân tích bài toán này, ta thấy có thể chia nó thành ba khối: 1) Nhập từ bàn phím một giá trị độ sâu tại đó cần nội suy nhiệt độ; 2) Nhập số liệu về độ sâu và nhiệt độ vào máy tính; 3) Nội suy giá trị nhiệt độ tại độ sâu cần tìm và in kết quả lên màn hình.

Khối thứ nhất rất đơn giản và quen thuộc. Để thực hiện khối thứ hai ta tổ chức một vòng lặp để tuần tự nhập độ sâu và nhiệt độ tại các điểm nút (xem lưu đồ của khối 2 trên hình 4.7).

Bảng 4.3. Phân bố nhiệt độ nước biển (°C) theo độ sâu (m)

Độ sâu	0	5	10	20	30	40	50	60
Nhiệt độ	24,31	24,26	24,20	24,18	24,13	24,05	23,98	23,89
Độ sâu	70	80	90	100	120	140	160	180
Nhiệt độ	23,87	23,57	23,14	22,74	21,31	20,03	18,49	17,58
Độ sâu	200	220	240	260	280	300	350	400
Nhiệt độ	16,66	15,61	14,73	13,97	13,47	12,93	11,40	10,18
Độ sâu	500	600	700	800	900	1000	1200	1400
Nhiệt độ	9,39	8,56	8,49	7,83	7,27	6,71	6,16	5,44



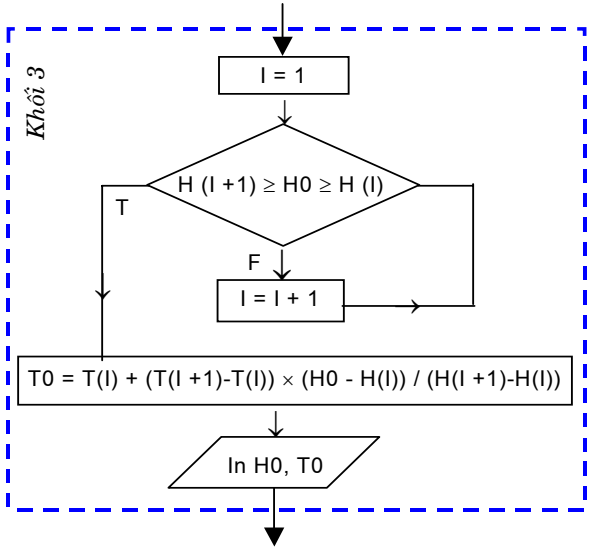
Hình 4.7. Lưu đồ khối 2 (thí dụ 8) - nhập chuỗi độ sâu và nhiệt độ

Bây giờ ta cụ thể hóa thêm khối thứ 3 và sau đó dẫn chương trình Fortran hoàn chỉnh của bài toán này.

Như đã thấy, các giá trị quan trắc nhiệt độ được cho chỉ tại 32 độ sâu gọi là 32 điểm nút. Muốn nội suy giá trị nhiệt độ tại độ sâu bất kỳ ta cần tìm xem độ sâu đó nằm giữa hai nút nào. Gọi độ sâu cần nội suy nhiệt độ là h_0. Giả sử độ sâu này nằm giữa các độ sâu nút h_i và h_{i+1}, tức thỏa mãn bất đẳng thức kép:

$$h_i \le h_0 \le h_{i+1},$$

trong đó i có thể biến thiên từ 1 đến 31. Như vậy, để tìm i, ta phải giả sử $i = 1$ và kiểm tra bất đẳng thức kép trên đây. Nếu bất đẳng thức không thỏa mãn, thì ta tăng i lên một đơn vị và tiếp tục cho tới khi bất đẳng thức thỏa mãn.



Hình 4.8. Lưu đồ khối 3 (thí dụ 8) - nội suy giá trị nhiệt độ và in kết quả

Khi tìm được i, giá trị t_0 cần nội suy có thể tính theo công thức nội suy tuyến tính như sau:

$$t_0 = t_i + \frac{(t_{i+1} - t_i)\ (h_0 - h_i)}{h_{i+1} - h_i}.$$

Tất cả những điều vừa phân tích được thể hiện trên lưu đồ khối ở hình 4.8. Dưới đây là chương trình của bài toán

```
      INTEGER N, I, K
      REAL H0, T0, H (40), T (40)
C     In lời nhắc và nhập độ sâu cần nội suy nhiệt độ
      PRINT * , ' NHAP DO SAU XAC DINH NHIET DO'
      READ *, H0
C     In lời nhắc và nhập 32 cặp giá trị độ sâu và nhiệt độ
      N = 32
      K = 1
5     PRINT *, ' NHAP DO SAU VA NHIET DO TANG ', K
      READ *, H(K), T(K)
      K = K +1
      IF (K .GT. N)  GOTO 4
      GOTO 5
C     Nội suy giá trị nhiệt độ tại độ sâu H0
4     I = N - 1
      IF (H0 .GT. H(N))  GOTO 1
      I = 1
2     IF (H0 .GE. H (I) .AND. H0 .LE. H (I+1))  GOTO 1
      I = I + 1
```

```
      GOTO 2
1     T0 = T(I) + (T(I+1)-T (I))*(H0-H(I)) / (H(I+1)-H(I))
      PRINT 3, H0
      PRINT 6, T0
3     FORMAT (1X, 'DO SAU =  ', F6.1, ' M')
6     FORMAT (1X,'NHIET DO =  ', F5.1, ' DO C')
      END
```

Qua thí dụ ở mục 4.1.3 và những thí dụ ở chương này ta thấy việc áp dụng quy trình 5 bước giải bài toán và chiến lược chia khối và chi tiết hoá từng khối để phát triển thuật giải là một công cụ lập trình rất hiệu quả. Bài toán dù lớn, có cấu trúc phức tạp cũng trở nên sáng tỏ, trực quan.

Từ thời điểm này sinh viên cần rèn luyện cho mình thói quen áp dụng phương pháp trên ngay cả với những bài tập đơn giản cũng như với những bài toán tương đối phức tạp khi thiết kế thuật giải. Còn chọn công cụ giả trình hay lưu đồ là tuỳ thích.

Bài tập

1. Hãy thể hiện bằng giả trình hoặc lưu đồ thuật toán sắp xếp các phần tử của mảng một chiều theo thứ tự giảm dần.

2. Cho các giá trị:

A = 2.2 B = −1.2 I = 1 DONE = .TRUE.

Xác định giá trị của các biểu thức lôgic sau đây:

1) A .LT. B 2) A − B .GE. 6.5
3) I .NE. 54) A + B .GE. B
5) I .LE. I −5 6) .NOT. (A .EQ. 2 * B)
7) (A .LT. 10.0) .AND. (B .GT. 5.0)
8) (ABS (I) .GT. 2) .OR. DONE
9) A .LT. B .NEQV. DONE

3. Viết chương trình tính giá trị của y theo công thức

$$y = \begin{cases} x^2 & \text{khi } x \le 0; \\ x^3 & \text{khi } x > 0, \end{cases}$$

với x cho trước.

4. Viết chương trình đọc từ bàn phím một trị số nhiệt độ Celsius, liệt kê trên màn hình ba phương án chuyển đổi: sang độ Fahrenheit, Kelvin và Rankin. Theo người dùng chỉ định phương án chuyển đổi mà in ra nhiệt độ đã cho và kết quả chuyển đổi kèm các ký hiệu nhiệt độ tương ứng. Các công thức chuyển đổi như sau:

$$T_F = T_R - 459{,}67^\circ \text{R}$$
$$T_F = \frac{9}{5} T_C + 32^\circ \text{F}$$
$$T_R = \frac{9}{5} T_K$$

5. Viết chương trình tính tích phân $I = \int_1^{15} y(x)dx$ với hàm $y(x)$ cho dưới dạng bảng các giá trị thực nghiệm như trong bảng 4.4.

Bảng 4.4

x	1,0	2,1	3,0	3,9	4,8	6,2	7,1	7,8
y	3,3	4,7	7,3	8,7	11,3	12,7	15,3	16,7
x	9,4	10,1	11,3	12,1	13,5	13,9	15,0	
y	19,3	20,7	23,3	24,7	27,3	28,7	31,3	

6. Viết chương trình cho phép đọc vào từ bàn phím một trị số của x và xác định trị số của hàm y bằng cách nội suy tuyến tính theo bảng giá trị

thực nghiệm (thí dụ bảng 4.4).

7. Hệ số nhớt phân tử ($g \cdot cm^{-1} \cdot s^{-1}$) của nước biển phụ thuộc vào nhiệt độ t (°) và độ muối S (‰) theo bảng 4.5. Viết chương trình nội suy tuyến tính bảng này cho một cặp trị số bất kỳ của t° và S.

8. Viết chương trình tính số π theo công thức khai triển chuỗi sau đây với sai số không quá 0,0001:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} ....$$

Bảng 4.5

Độ muối	0°	5°	10°	15°	20°	25°	30°
0	17,94	15,19	13,10	11,45	10,09	8,95	8,00
5	18,06	15,28	13,20	11,54	10,18	9,08	8,09
10	18,18	15,39	13,28	11,68	10,27	9,18	8,17
15	18,30	15,53	13,41	11,77	10,40	9,26	8,27
20	18,41	15,66	13,57	11,90	10,47	9,35	8,34
25	18,53	15,79	13,73	12,03	10,58	9,48	8,43
30	18,64	15,93	13,84	12,12	10,68	9,58	8,52
35	18,83	16,07	14,00	12,23	10,82	9,67	8,59

9. Viết chương trình cho phép liên tục nhập từ bàn phím hai số nguyên bất kỳ, tìm và in lên màn hình ước số chung lớn nhất của những số đó dưới dạng thông báo:

USCLN CUA CAC SO: 36 VA 24 BANG 12

và kết thúc khi nào người dùng nhập vào hai số bằng nhau hoặc một trong hai số bằng 1.

10. Lập lưu đồ thuật giải để giải gần đúng phương trình $x = f(x)$ bằng phương pháp lặp Siedel. Xấp xỉ ban đầu x_0 và sai số cho phép ε được cho trước. Nếu tìm được nghiệm với độ chính xác đã cho thì in giá trị nghiệm kèm theo số bước lặp, còn nếu sau 100 lần lặp mà chưa nhận được nghiệm thì thông báo lên màn hình dòng chữ 'KHONG TIM DUOC NGHIEM'. (Gợi ý: Theo phương pháp lặp Seidel, người ta thế giá trị x_0 tuỳ chọn vào biểu thức $f(x)$ ở vế phải của phương trình $x = f(x)$ để tính ra giá trị x_1 – gọi là xấp xỉ bậc 1, sau đó kiểm tra nếu khác nhau giữa x_1 và x_0 lớn hơn sai số cho phép ε thì giá trị x_1 lại được thế vào vế phải và tiếp tục tính x_2 (xấp xỉ bậc 2)..., quá trình này tiếp diễn cho đến khi chênh lệch giữa hai bước xấp xỉ liền nhau không lớn hơn ε thì người ta chấp nhận giá trị xấp xỉ cuối cùng làm nghiệm của phương trình $x = f(x)$.

Chương 5 - Cấu trúc lặp với lệnh DO

Trong chương 4 đã xét sự điều khiển được thực hiện thông qua những lệnh cho phép chương trình chọn những nhánh khác nhau để thực hiện. Đồng thời, ta cũng đã một số lần sử dụng kết hợp lệnh **IF lôgic** và lệnh chuyển điều khiển vô điều kiện **GOTO** để tổ chức những vòng lặp dạng:

```
n  IF (Biểu thức lôgic) THEN
     Lệnh 1
     Lệnh 2
     . . .
     Lệnh m
     GOTO n
   END IF
```

Cấu trúc này gọi là vòng lặp có điều kiện (*While Loop*): Khi và chừng nào biểu thức lôgic trong lệnh IF có giá trị **.**TRUE**.** thì nhóm lệnh từ lệnh 1 đến lệnh m lần lượt thực hiện, nhưng lệnh GOTO ở cuối luôn luôn chuyển điều khiển lên nhãn n và hình thành vòng lặp. Vòng lặp này có những đặc điểm sau:

1) Trường hợp biểu thức lôgic có giá trị **.**FALSE**.** ngay từ đầu, thì quá trình lặp sẽ không xảy ra;

2) Trong nhóm lệnh từ lệnh 1 đến lệnh m bên trong vòng lặp nhất thiết phải có một lệnh nào đó làm thay đổi giá trị của biểu thức lôgic thành **.**FALSE**.**, vậy số lần lặp phụ thuộc vào giá trị khởi đầu của biểu thức lôgic và sự biến đổi giá trị của nó bên trong chính vòng lặp.

Trong bài này ta xét một cấu trúc lặp khác mà điều kiện và số lần lặp được xác định ngay từ khi bắt đầu quá trình lặp với việc sử dụng ***vòng lặp*** **DO** (**DO** ***Loop***). Trong chương tiếp sau sẽ xét một tính năng quan trọng của vòng lặp DO, gọi là *vòng lặp ẩn*, để tổ chức nhập, xuất các biến có chỉ số rất hay gặp trong thực tiễn.

5.1. Vòng lặp DO

5.1.1. Cú pháp của lệnh DO và vòng lặp DO

Dạng tổng quát của lệnh DO như sau:

DO n $ind = ini, \lim, \text{inc}$

trong đó hằng n là nhãn của lệnh kết thúc của vòng lặp, ind – là một biến số được dùng như là *chỉ số đếm* vòng lặp, ini – *giá trị đầu* gán cho chỉ số đếm, lim – *giá trị cuối* dùng để xác định khi nào vòng lặp DO kết thúc và inc – *gia số*, giá trị được cộng vào chỉ số đếm mỗi lần vòng lặp thực hiện.

Những giá trị đầu, giá trị cuối và gia số gọi là *các tham số của vòng lặp*. Nếu trong lệnh DO không ghi gia số thì ngầm định là 1. Khi giá trị của chỉ số đếm lớn hơn giá trị cuối thì điều khiển được chuyển cho lệnh đứng sau lệnh kết thúc vòng lặp. Lệnh kết thúc vòng lặp thường dùng là lệnh CONTINUE, có dạng tổng quát là

n **CONTINUE**

trong đó nhãn n phù hợp với nhãn mà lệnh DO ở trên đã chỉ định.

Vậy dạng tổng quát của vòng lặp DO có thể viết như sau:

DO *n ind = ini , lim , inc*

Lệnh 1

. . .

Lệnh m

n **CONTINUE**

Ta lấy thí dụ giải bài toán tính tổng của 50 số nguyên dương đầu tiên

$$\sum_{i=1}^{50} i = 1 + 2 + \ldots + 49 + 50$$

để minh họa vòng lặp DO và so sánh nó với vòng lặp *While* mà ta đã xét ở bài trước:

Vòng lặp DO	Vòng lặp While
SUM = 0.0	SUM = 0.0
DO 10 NUM = 1, 50	NUM = 1
SUM = SUM + NUM	**10** IF (**NUM .LE. 50**) THEN
10 CONTINUE	SUM = SUM + NUM
	NUM = NUM + 1
	GO TO **10**
	END IF

Trong vòng lặp DO trên đây chỉ số đếm NUM được khởi xướng bằng 1. Vòng tiếp tục lặp cho đến khi giá trị của NUM lớn hơn 50. Vì tham số thứ ba bỏ qua nên NUM tự động tăng lên 1 ở cuối mỗi lần lặp. Ta thấy rằng vòng lặp DO viết ngắn gọn hơn vòng lặp *While*, nhưng cả hai tính cùng một giá trị của biến SUM. Tuy nhiên, trong vòng lặp *While* ở mỗi lần lặp biểu thức lôgic luôn phải được ước lượng lại vì mỗi lần biến NUM được thay bởi giá trị mới.. Trong khi đó ở vòng lặp DO số lần lặp đã được tính trước trong lệnh DO. Đó là sự khác nhau cơ bản của hai loại vòng lặp.

Người ta cũng có thể dùng cú pháp sau đây cho vòng lặp DO:

DO *ind = ini, lim , inc*

Lệnh 1

. . .

Lệnh m

END DO

5.1.2. Những quy tắc cấu trúc và thực hiện vòng lặp DO

1) Chỉ số đếm phải là một biến số, biến đó có thể là kiểu nguyên hoặc thực, nhưng không thể là biến có chỉ số.

2) Các tham số của vòng DO có thể là hằng, biến hay biểu thức nguyên hoặc thực. Gia số có thể là số dương, số âm, nhưng không thể bằng không.

3) Vòng DO có thể dùng bất kỳ lệnh thực hiện nào *không phải là một lệnh chuyển điều khiển, lệnh IF hay một lệnh* DO *khác* làm lệnh cuối vòng. Lệnh CONTINUE là một lệnh thực hiện chuyên dùng làm lệnh cuối vòng; mặc dù có thể dùng những lệnh khác, nhưng nói chung nên dùng lệnh CONTINUE để chỉ cuối vòng lặp một cách tường minh.

4) Sự kiểm tra kết thúc lặp thực hiện ở đầu vòng lặp. Nếu giá trị đầu của chỉ số đếm lớn hơn giá trị cuối và gia số là số dương thì sự lặp không diễn ra, các lệnh bên trong vòng lặp bị bỏ qua và điều khiển chuyển tới lệnh đứng sau lệnh cuối cùng của vòng lặp.

5) Không được thay đổi giá trị của chỉ số đếm bằng một lệnh nào khác bên trong vòng DO trong khi thực hiện vòng lặp.

6) Sau khi vòng lặp đã bắt đầu thực hiện thì những thay đổi các giá trị của các tham số không có ảnh hưởng gì tới sự lặp.

7) Nếu gia số là âm, sự lặp sẽ kết thúc khi giá trị chỉ số đếm nhỏ hơn giá trị cuối.

8) Ta có thể thoát ra khỏi vòng DO trước khi nó kết thúc lặp. Khi đó giá trị của chỉ số đếm sẽ bằng giá trị ngay trước khi thoát. (Nhưng nói chung không nên làm điều này. Nếu ta muốn thoát ra khỏi vòng lặp trước khi nó kết thúc một cách tự nhiên, thì ta cấu trúc lại vòng lặp theo kiểu vòng lặp *While* để giữ tính cấu trúc của chương trình).

9) Thực hiện xong vòng lặp, chỉ số đếm chứa một giá trị vượt quá giá trị cuối.

10) Bao giờ cũng đi vào vòng lặp thông qua lệnh DO để vòng lặp được khởi xướng một cách đúng đắn. Không bao giờ được dùng lệnh GO TO chuyển từ bên ngoài vào bên trong vòng DO.

11) Số lần lặp có thể tính bằng công thức

$$\left[\frac{\text{lim-ini}}{\text{inc}}\right]+1$$

trong đó dấu ngoặc vuông chỉ sự cắt bỏ thập phân của thương số. Nếu giá trị này âm thì sự lặp không xảy ra.

5.1.3. Thí dụ ứng dụng vòng lặp DO

Thí dụ 9**: **Lập vòng lặp bằng lệnh* **DO. Lập bảng giá trị của đa thức $3t^2+4{,}5$ trên đoạn t từ 1 đến 10 với bước $\Delta t = 1$.

```
      PRINT * , ' POLYNOMIAL MODEL'
      PRINT *
      PRINT * , 'TIME     POLYNOMIAL'
      DO 15 I = 1, 10
         POLY = 3. * REAL (I) ** 2 + 4.5
         PRINT 10 , I , POLY
10       FORMAT (1X, I2, 8X, F6.2)
15    CONTINUE
      END
```

Thí dụ 10**: **Tìm phần tử cực đại của chuỗi số $b_1, b_2, ..., b_{10}$. Ta giải bài toán này theo thuật giải biểu diễn bởi giả trình sau:

1) với i *từ* 1 *đến* 10
   nhập b_i
2) $b_{max} \leftarrow b_1$
3) với i từ 2 đến 10
   nếu $b_i > b_{max}$ thì $b_{max} \leftarrow b_i$
4) *in* b_{max}

Từ giả trình này dễ dàng chuyển thành chương trình Fortran dưới đây:

```
      REAL B(10)
      DO 2 I = 1, 10
         READ *, B (I)
2     CONTINUE
      BMAX = B (1)
      DO 3  I = 2, 10
         IF (BMAX .LT. B (I))  BMAX = B (I)
3     CONTINUE
      PRINT *, ' B MAX = ' , BMAX
      END
```

Thí dụ 11: ***Tổ chức vòng lặp với bước số thập phân***. In bảng giá trị hàm $y = \sin(x)$ tại $x = 0;\ 0{,}1;\ 0{,}2; ...; 1$. Ta đưa ra một biến nguyên I sao cho biến này sẽ nhận các giá trị 1, 2, ..., 11 tương ứng với $x = 0;\ 0{,}1;\ 0{,}2; ...; 1$. Khi đó $x = 0{,}1\,(i - 1)$.

```
      DO 17  I = 1, 11
         X = 0.1 * (I – 1)
         Y = SIN (X)
         PRINT 10 , X, Y
10       FORMAT (20X, F4.2, 10X, E10.3)
17    CONTINUE
      END
```

Hãy lưu ý rằng ở đây ta đã tránh sử dụng vòng lặp DO với các tham số thực như:

```
DO 15 X = 0.0 , 1.0 , 0.1
```

để phòng ngừa *hiện tượng cắt* trong máy tính. Giả sử rằng giá trị 0.1 được lưu như một giá trị hơi nhỏ hơn 0.1 trong hệ máy tính đang dùng, mỗi lần thêm 0.1 cho chỉ số đếm, máy có thể thêm ít hơn theo dự định. Ngoài ra, trong trường hợp này ta có thể thực hiện lặp quá mất một lần theo dự định vì giá trị giới hạn cuối cũng có thể không chính xác bằng 1.0.

5.2. Vòng DO lồng nhau

Vòng DO có thể được lồng trong một vòng DO khác, cũng giống như cấu trúc IF lồng trong cấu trúc IF khác. *Khi tổ chức các vòng* DO *lồng hãy tuân thủ những quy tắc sau đây*:

1) Vòng DO lồng bên trong không thể dùng chính chỉ số đếm cùng với vòng DO ngoài chứa nó.

2) Vòng DO lồng phải kết thúc bên trong vòng DO ngoài.

3) Các vòng DO độc lập nhau có thể dùng cùng chỉ số đếm, thậm chí khi chúng cùng nằm trong một vòng DO ngoài.

4) Khi một vòng DO lồng bên trong một vòng DO khác, thì vòng DO trong thực hiện trọn vẹn từng lần lặp ở vòng DO ngoài.

5) Mặc dù các vòng DO lồng có thể dùng cùng một dòng lệnh cuối CONTINUE, nhưng ta nên kết thúc mỗi vòng bằng một lệnh CONTINUE riêng biệt để làm sáng rõ chương trình.

Dưới đây dẫn một số thí dụ về các vòng DO đúng và các vòng DO sai:

a) Những vòng DO đúng:

```
      DO 15 I = 1, 5
         DO 10 J = 1, 8
            DO 5 K = 2, 10, 2
            . . .
 5          CONTINUE
10       CONTINUE
15    CONTINUE
```

```
      DO 15 I = 1, 5
         DO 10 K = 1, 8
         . . .
10       CONTINUE
         DO 5 K = 2, 10, 2
         . . .
 5       CONTINUE
15    CONTINUE
```

b) Những vòng DO sai:

```
      DO 15 I = 1, 5
         DO 10 J = 1, 8
            DO 5 K = 2, 10, 2
            . . .
10       CONTINUE
            . . .
 5          CONTINUE
15    CONTINUE
```

```
      DO 20 J = 1, 5
         DO 10 J = 1, 8
         . . .
10       CONTINUE
         DO 15 K = 2, 10, 2
            DO 15 K = 2, 10, 2
            . . .
15       CONTINUE
20    CONTINUE
```

Thí dụ 12: ***Tổ chức vòng DO lồng nhau***. Viết chương trình nhập 15 phần tử của mảng số thực X, sắp xếp lại các phần tử mảng theo thứ tự giảm dần và in lên màn hình các mảng cũ và mới thành hai cột.

```
      REAL X(15), Y(15)
      N = 15
      DO 3  I =1, N
         READ * , X (I)
         Y (I) = X (I)
3     CONTINUE
      DO 2  I = 1, N–1
         K = I
         DO 4  J = I + 1, N
            IF (Y (K) .LT. Y (J))  K = J
4        CONTINUE
         IF (K .NE. I) THEN
            TG = Y (I)
            Y (I) = Y (K)
            Y (K) = TG
         END IF
2     CONTINUE
      DO 7  I = 1, N
         PRINT 5 , X (I), Y (I)
7     CONTINUE
5     FORMAT (1X, 2F10.2)
      END
```

Trong thí dụ này, ta thấy có mặt ba vòng DO độc lập nhau:

DO 3 I =1, N (dòng thứ 3)

DO 2 I =1, N–1 (dòng thứ 7)

DO 7 I = 1, N (dòng thứ 18)

do đó, chúng có thể dùng cùng một chỉ số đếm là biến I. Bên trong vòng DO thứ hai, ta thấy xuất hiện một vòng DO thứ tư:

DO 4 J = I + 1, N (dòng thứ 9),

vòng DO này là vòng DO lồng, nó phải có chỉ số đếm riêng và ta dùng lệnh kết thúc nó là lệnh

4 CONTINUE

để nhấn mạnh sự phân biệt với vòng DO ngoài có lệng kết thúc là

2 CONTINUE

Thí dụ 13: Tính giai thừa. Khi số nguyên N không âm, biểu thức $N!$ gọi là giai thừa của N. Các giá trị của giai thừa được tính theo quy luật:

$$0! = 1$$
$$1! = 1$$
$$2! = 1 \times 2 = 2$$
$$3! = 1 \times 2 \times 3 = 6$$
$$\ldots$$

Giá trị của giai thừa của số nguyên N cũng còn được ước lượng bằng công thức Stirling có dạng:

$$N! = \sqrt{2\pi N}\left(\frac{N}{e}\right)^N$$

trong đó $e = 2{,}718282$. Viết chương trình in các giá trị giai thừa của các số nguyên từ 0 đến 10 theo cách tính chính xác và theo công thức ước lượng của Stirling.

```
      PRINT 4
```

```
4     FORMAT (1X, 'GIAI THUA CUA CAC SO TU 0 DEN 10'
     *  //1X, T3, 'N', T12, 'N!', T16, 'STIRLING"S FORMULA' /)
      FAC = 1.
      DO 7  I = 0, 10
      IF (I .GT. 1)  FAC = FAC * I
      PRINT 5, I, FAC, SQRT (2.*3.141593*I)*(I / 2.718282)** I
5     FORMAT (1X, I2, F10.0, F20.0)
7     CONTINUE
      END
```

Trong chương trình này, vì giai thừa được tính liên tục với các số từ 0 đến 10, nên giai thừa của một số sau được tính bằng tích của số đó nhân với giai thừa của số trước nó. Hãy chú ý cách dùng dấu gạch chéo để tạo xuống dòng khi in tiêu đề: hai dấu gạch chéo đầu chỉ định cho lệnh PRINT in xong dòng chữ GIAI THUA CUA CAC SO TU 0 DEN 10 thì xuống dòng hai lần, sau khi in dòng tiêu đề cột, dấu gạch chéo thứ ba gây xuống dòng một lần để chuẩn bị in dữ liệu theo dòng lệnh in trong vòng lặp DO. Các đặc tả T3, T12, T16 trong dòng lệnh 4 FORMAT chỉ định xuất chữ N ở vị trí 3, N! ở vị trí 12 và 13, chữ STIRLING'S FORMULA bắt đầu ở vị trí thứ 16 của dòng tiêu đề cột. Kết quả xuất ra của chương trình này sẽ có dạng dưới đây:

```
GIA TRI GIAI THUA CAC SO TU 0 DEN 10
 N         N!  STIRLING'S FORMULA

 0         1.                 0.
 1         1.                 1.
 2         2.                 2.
 3         6.                 6.
 4        24.                24.
 5       120.               118.
 6       720.               710.
 7      5040.              4980.
 8     40320.             39902.
 9    362880.            359537.
10   3628800.           3598694.
```

Bài tập

1. Tính số lần lặp trong các trường hợp dùng lệnh DO sau đây. Giả thiết rằng các chỉ số đếm là những biến nguyên:

1) DO 5 I = 1, 8
2) DO 10 COUNT = -4, 4
3) DO 10 K = 15, 3, –1
4) DO 10 TIME = –5, 15, 3
5) DO 10 TIME = 50, 250, 25
6) DO 10 IND = 72, 432, 4

2. Xác định giá trị của biến nguyên IDEM sau khi những vòng DO dưới đây thực hiện xong. Giả sử biến này được gán giá trị không trước mỗi vòng lặp.

1)
```
      DO 5 I = 1, 8
        IDEM = IDEM + 1
5     CONTINUE
```

2)
```
      DO 5 IDEX =0, 7
        IDEM = IDEM – 2
5     CONTINUE
```

3)
```
      DO 5 NUM = 8, 0, –1
        IDEM = IDEM + 2
5     CONTINUE
```

4)
```
      DO 5 M = 5, 5
      IDEM = IDEM + (–1) **M
5     CONTINUE
```

3. Một hòn đá được ném với tốc độ ban đầu v và nghiêng một góc θ so với mặt đất. Nếu bỏ qua lực cản ma sát với không khí thì khoảng cách d theo chiều ngang kể từ vị trí ban đầu và độ cao h (tính bằng mét) của nó tại thời gian t (giây) biểu thị bằng các phương trình sau:

$$d = v\,t\cos\theta,$$

$$h = v\,t\sin\theta - \frac{1}{2}g\,t^2,$$

trong đó $g-$ gia tốc trọng lực ($9{,}8\ \text{m/s}^2$). Viết chương trình đọc vận tốc ban đầu và góc và sau đó in bảng các khoảng cách và độ cao của hòn đá với thời gian cách nhau 0,25 giây cho tới khi độ cao trở thành giá trị âm, tức lúc hòn đá rơi xuống mặt đất.

4. Hãy tổ chức lại các vòng lặp trong thí dụ 13 bằng cách sử dụng kết hợp lệnh IF lôgic và lệnh chuyển điều khiển vô điều kiện GOTO. Phân tích sự khác nhau của hai cách tổ chức vòng lặp.

5. Giả sử các giá trị quan trắc hai đại lượng x và y được cho như trong bảng 4.4 (trang 79). Hãy viết chương trình tính các đặc trưng thống kê: trung bình m_x, m_y, phương sai D_x, D_y, độ lệch bình phương trung bình σ_x, σ_y, hệ số tương quan r giữa hai đại lượng và lập phương trình hồi quy dạng:

$$y = a\,x + b,$$

trong đó:

$$a = \frac{\sigma_y}{\sigma_x}r, \qquad b = m_y - a\,m_x,$$

$$m_x = \frac{\sum_{i=1}^{n} x_i}{n}, \qquad D_x = \frac{\sum_{i=1}^{n} x_i^2}{n-1} - m_x^2, \qquad \sigma_x = \sqrt{D_x}$$

$$m_y = \frac{\sum_{i=1}^{n} y_i}{n}, \qquad D_y = \frac{\sum_{i=1}^{n} y_i^2}{n-1} - m_y^2, \qquad \sigma_x = \sqrt{D_x}$$

$$r = \frac{\dfrac{\sum_{i=1}^{n} x_i y_i}{n-1} - m_x m_y}{\sigma_x \sigma_y}.$$

6. Viết chương trình tính trị gần đúng của tích phân

$$I = \int_a^b x^2 \sin x dx$$

theo công thức hình thang với sai số không lớn hơn 0,0001, xác định số hình thang cần chia để đạt sai số đó. Chương trình cho phép nhập từ bàn phím các cận tích phân và in kết quả lên màn hình thành các dòng như sau (thí dụ nếu $a = 0{,}5$ và $b = 1{,}5$):

A = 0.5

B = 1.5

SO HINH THANG = 16

TICH PHAN BANG = 0.9604

7. Viết chương trình cho phép nhập từ bàn phím một góc a tính bằng độ, đổi góc đó thành radian và tính giá trị gần đúng của $\cos a$ với độ chính xác tới 0,0001 theo công thức khai triển sau đây:

$$\cos a = 1 - \frac{a^2}{2!} + \frac{a^4}{4!} - \frac{a^6}{6!} + \ldots$$

In kết quả lên màn hình thành một dòng như sau (thí dụ):

A = 60.000 (DO) cos A = 0.5000 cos A theo hàm chuẩn = 0.5000

8. Viết chương trình cho phép nhập từ bàn phím hai số nguyên (nhỏ hơn 10) tuần tự chỉ số dòng và số cột của một ma trận. Sau đó tính các phần tử của ma trận sao cho mỗi phần tử là một số nguyên gồm hai chữ số,

chữ số đầu chỉ số thứ tự dòng và chữ số sau chỉ số thứ tự cột. In ma trận đó lên giữa màn hình dưới dạng bảng số thẳng dòng, thẳng cột, thí dụ:

11	12	13	14
21	22	23	24
31	32	33	34
41	42	43	44
51	52	53	54
61	62	63	64

Chương 6 - File dữ liệu và tổ chức file dữ liệu trong Fortran

6.1. Khái niệm về file dữ liệu và tổ chức lưu trữ dữ liệu

Ở các chương trước, trong quá trình thực hiện chương trình, khi nào cần ta đã sử dụng lệnh READ để nhập thông tin vào qua bàn phím cho chương trình xử lý. Thí dụ: khi chạy chương trình giải phương trình bậc hai trong thí dụ 1 ở chương 4, ta phải nhập ba hệ số a, b, c. Với một lượng thông tin không nhiều, thí dụ một vài giá trị số hay một vài ký tự văn bản, thì hình thức giao tiếp này giữa người và máy là bình thường. Nhưng khi làm việc với nhiều số liệu, sẽ là bất tiện nếu phải nhập dữ liệu bằng tay như vậy mỗi lần chạy chương trình. Thí dụ, trong bài toán đã xét ở thí dụ 8 (trang 74) ta phải nhập từ bàn phím hai chuỗi giá trị độ sâu và nhiệt độ gồm vài chục số liệu ở các tầng khác nhau (64 số) chỉ để nội suy một giá trị nhiệt độ. Ngoài ra, nếu trong khi chạy chương trình mà ta gõ nhầm số liệu thì phải chạy lại chương trình từ đầu và đương nhiên phải nhập lại số liệu một lần nữa. Công việc đó tỏ ra rất mệt mỏi và không tối ưu.

Vì vậy, người ta thường nhập dữ liệu vào máy một lần và lưu trong máy (đĩa cứng, đĩa mềm...) dưới dạng các tệp (file). Trong trường hợp này người sử dụng máy phải dùng một phần mềm soạn thảo nào đó để nhập dữ liệu vào máy và lưu lại dưới dạng các file. Ngoài ra, dữ liệu (thường là những giá trị số) cũng có thể do một thiết bị quan trắc có bộ phận ghi lưu

vào đĩa từ, băng từ, ổ cứng máy tính theo một quy cách nào đó sau này máy tính có thể đọc được. Dữ liệu cũng có thể là kết quả tính toán, xử lý của một chương trình máy tính, sau đó được ghi lại thành file để người sử dụng máy xem như là kết quả tính toán để phân tích, nhận xét, sau này có thể in ra giấy như là một bản báo cáo, hay để một chương trình khác đọc và tiếp tục xử lý, chế biến.

Trong chương này chúng ta sẽ nghiên cứu những lệnh của Fortran thao tác với dữ liệu, tìm hiểu những đặc điểm của chúng để hình thành kỹ thuật tổ chức lưu dữ liệu trong máy tính.

File dữ liệu là file trong máy tính chứa những thông tin có quan hệ với nhau theo một nghĩa nào đó mà một chương trình có thể đọc, hay truy cập được nếu ta muốn chương trình xử lý tự động những thông tin đó.

Chương trình máy tính truy cập đến các file theo tên của chúng. Tên file được đặt tuân theo quy tắc tên như đối với các biến. Ngoài ra tên file còn có thể có một phần mở rộng, còn gọi là đuôi file, gồm tổ hợp không quá ba chữ cái hoặc chữ số. Phần mở rộng này đứng sau phần tên chính của file và ngăn cách bằng dấu chấm.

Trong thực hành người ta thường đặt tên file bằng tập hợp một số chữ cái và chữ số có ý nghĩa gợi cho người dùng dễ nhớ đó là file chứa những dữ liệu gì. Thí dụ, nếu chúng ta có những số liệu quan trắc về nhiệt độ không khí của một số tháng muốn lưu trong một số file thì có thể nên đặt tên các file đó là NHIET.1, NHIET.2 v.v... Ở đây ngụ ý muốn dùng cụm chữ NHIET để chỉ các file đó lưu trữ số liệu về nhiệt độ, còn phần đuôi của tên file nhằm chỉ số liệu về nhiệt nhưng riêng cho tháng 1, tháng 2... Các file trong máy tính lại có thể được ghi vào những thư mục có tên khác nhau. Trong mỗi thư mục lại có thể gồm một số thư mục con cũng có tên của chúng, hình thành một cây thư mục. Một nhóm file có quan hệ tương đối với nhau theo nghĩa nào đó có thể ghi chung vào một thư mục, một số file khác thì có thể ghi trong những thư mục khác. Kinh nghiệm cho thấy rằng việc tổ chức lưu các file trong máy tính một cách có hệ thống, khoa học sẽ giảm nhẹ và nâng cao hiệu quả công việc của người sử dụng máy tính.

Xét về phương diện lưu trữ dữ liệu lâu dài thì người ta thường cố gắng ghi trong file sao cho phong phú thông tin, đáp ứng việc xử lý nhiều mục đích. Thí dụ, với file chứa những số liệu các tham số khí tượng thuỷ văn ở một trạm quan trắc nào đó, thì ngoài những giá trị số của các tham số đó, nên có thêm những thông tin về tên trạm, tọa độ trạm, thời kỳ quan trắc, có thể ghi tên các tham số quan trắc một cách tường minh...

Tuỳ theo đặc điểm và khả năng xử lý của chương trình hay phần mềm mà người ta ghi các dữ liệu trong file sao cho gọn, dễ đọc, dễ chuyển đổi từ định dạng (format) này sang định dạng khác, tức xu thế chuẩn hoá định dạng dữ liệu để nhiều chương trình, nhiều phần mềm có thể đọc được.

Trong chương này chúng ta học cách làm việc với những file dữ liệu số, làm thế nào để đọc thông tin từ file dữ liệu hiện tồn tại và làm thế nào để tạo ra file dữ liệu mới trong chương trình Fortran mà chúng ta viết.

6.2. Các lệnh nhập, xuất dữ liệu với file

Để sử dụng các file với chương trình của mình, chúng ta phải dùng những lệnh mới để thao tác với file và những mở rộng đối với một số lệnh đã nghiên cứu trong các chương trước. Những lệnh này truy cập đến tên file mà ta đã gán khi tạo lập file. Nếu ta tạo lập một file dữ liệu bằng phần mềm soạn thảo, ta gán tên cho file khi nhập dữ liệu. Nếu ta tạo ra một file bằng một chương trình, ta phải dùng một lệnh trong chương trình cấp cho file một tên nhất định.

Nếu một file chuẩn bị được dùng trong chương trình, file đó phải được mở ra trước khi có một thao tác nào đó với nó. Lệnh mở file OPEN báo cho chương trình một số thông tin cần thiết về file như tên file, file đó

mở ra để đọc hay để ghi dữ liệu. Ngoài ra lệnh OPEN gắn file được mở với một số hiệu file để khi nào chương trình truy cập file thì nó sử dụng số hiệu đó. Dạng tổng quát của lệnh OPEN mà ta sử dụng trong chương này như sau:

OPEN (UNIT = *biểu thức nguyên*, **FILE =** *tên file*, **STATUS =** *văn bản*)

trong đó *biểu thức nguyên* chỉ định một *số hiệu thiết bị* được gán cho file, *tên file* chỉ định một tên mà ta đã gán cho file khi tạo lập ra nó và *văn bản* STATUS báo cho chương trình biết file mở ra để đọc hay để ghi, file đã tồn tại hay chuẩn bị tạo ra. Nếu là file để đọc, ta chỉ định

STATUS = 'OLD'

nếu file mở ra để xuất, ta chỉ định

STATUS = 'NEW'

còn

STATUS = 'UNKNOWN'

cho phép mở file mới hoặc ghi đè lên file đã tồn tại.

Lệnh OPEN phải đứng trên những lệnh READ hoặc WRITE sử dụng file.

Để đọc file dữ liệu, ta sử dụng lệnh **READ** mở rộng dưới dạng:

READ (*Số hiệu thiết bị* , *) *Danh sách các biến*

Để ghi thông tin vào file dữ liệu ta sử dụng lệnh **WRITE**. Giống như lệnh **PRINT**, lệnh **WRITE** có thể sử dụng để xuất thông tin ra dưới dạng danh sách liệt kê và dưới dạng ghi không định dạng hoặc có định dạng:

WRITE (*Số hiệu thiết bị* , *) *Danh sách các biểu thức*

WRITE (*Số hiệu thiết bị* , *n*) *Danh sách các biểu thức*

trong đó *n* là nhãn của lệnh FORMAT (định dạng) tương ứng. Trong tất cả các dạng tổng quát trên đây *số hiệu thiết bị* phù hợp với *số hiệu thiết bị* đã gán trong lệnh OPEN. Dấu sao * đứng sau *số hiệu thiết bị* chỉ rằng ta đang sử dụng cách nhập và xuất không định dạng (không format).

Các máy tính có thể có một số thiết bị nhập hoặc xuất đi kèm. Mỗi thiết bị được gán một số hiệu. Thí dụ, nếu máy in lazer được gán số hiệu 8 thì lệnh in sau đây sẽ ghi giá trị của các biến *X* và *Y* ra máy in laser

WRITE (8 , *) X, Y

Đa số các hệ máy tính gán thiết bị nhập chuẩn (bàn phím) bằng số **5** và thiết bị xuất chuẩn (màn hình) bằng số **6**; những thiết bị này đã được dùng ngầm định với các lệnh READ * hay PRINT *. Do đó không nên dùng những số hiệu thiết bị đã gán trước này cho các file dữ liệu. Ta có thể dùng bất kỳ những số hiệu khác trong các số nguyên từ **1** đến **15** để chỉ đơn vị file.

Sau khi kết thúc đọc hoặc ghi file, các file tự động đóng lại trước khi chương trình kết thúc. Cũng có những trường hợp ta muốn chủ tâm đóng hay tách một file khỏi chương trình của mình, và điều này nên làm. Ta sẽ dùng lệnh đóng file có dạng tổng quát như sau:

CLOSE (UNIT = ***Biểu thức nguyên*****)**

Những lệnh mở, đóng file, xuất nhập thông tin với file trên đây còn có nhiều tuỳ chọn bổ sung khác nữa, sẽ được nhắc tới ở những nơi thích hợp trong các mục và các chương sau.

Dưới đây tóm tắt một số quy tắc quan trọng cần nhớ khi đọc dữ liệu từ các file:

1. Mỗi lệnh READ sẽ bắt đầu đọc với một dòng dữ liệu mới, gọi là

một bản ghi (record). Nếu còn thừa các giá trị ở dòng trước, thì những giá trị đó bị bỏ qua không đọc.

2. Nếu một dòng không chứa đủ các giá trị so với danh sách các biến cần đọc trong lệnh READ, thì các dòng dữ liệu sau đó sẽ tự động được đọc cho đến khi đủ giá trị cho các biến liệt kê trong lệnh READ.

3. Một lệnh READ không cần phải đọc tất cả các giá trị trên dòng dữ liệu hiện thời. Nhưng nó phải đọc tất cả những giá trị trên dòng ở trước giá trị mà ta muốn nó đọc. Thí dụ nếu một file có 5 giá trị ghi trên một dòng và ta cần các giá trị thứ ba và thứ tư, ta phải đọc qua các giá trị thứ nhất và thứ hai để đạt tới các giá trị thứ ba và thứ tư, nhưng ta không cần phải đọc giá trị thứ năm.

Để sử dụng đúng lệnh READ, ta cần biết các giá trị đã được ghi trong file như thế nào. Thí dụ, giả sử rằng mỗi dòng của file dữ liệu chứa hai số tuần tự biểu thị thời gian TIME và số đo nhiệt độ TEMP và ba dòng đầu tiên ghi như sau:

```
0.0     28.3        (dòng 1)
0.1     29.1        (dòng 2)
0.2     29.5        (dòng 3)
```

thì lệnh sau đây sẽ đọc được đúng một cặp giá trị thời gian và nhiệt độ từ file dữ liệu

```
READ (10, *)  TIME, TEMP
```

Nhưng sẽ là sai nếu ta dùng hai lệnh sau

```
READ (10, *)  TIME
READ (10, *)  TEMP
```

Thực hiện hai lệnh này sẽ đọc hai dòng của file dữ liệu: giá trị của biến TIME sẽ bằng 0.0 và giá trị của biến TEMP sẽ bằng 0.1. Trong trường hợp này chương trình vẫn làm việc bình thường nhưng kết quả xử lý sẽ sai. Thí dụ này minh hoạ sự quan trọng của việc kiểm tra chương trình của chúng ta đối với dữ liệu đã biết, trước khi sử dụng nó với file dữ liệu khác.

6.3. Kỹ thuật đọc các file dữ liệu

Để đọc các dữ liệu từ file dữ liệu, trước hết ta phải biết một số thông tin về file. Ngoài tên file, ta phải biết dữ liệu gì được lưu trong file và cụ thể ghi như thế nào: có bao nhiêu số ghi trên một dòng và các đơn vị đo của mỗi giá trị. Ta cũng phải biết trong file có thông tin gì đặc biệt có ích để phân định được số dòng ghi trong file, hay để xác định khi nào ta đã đọc hết dòng ghi cuối cùng. Thông tin này quan trọng, vì nếu ta thực hiện một lệnh READ sau khi tất cả các dòng ghi trong file đã được đọc hết rồi thì sẽ bị lỗi thực hiện chương trình. Ta có thể tránh lỗi đó bằng cách sử dụng thông tin về file để quyết định xem loại vòng lặp nào nên dùng khi đọc file. Thí dụ, nếu ta biết có 200 dòng ghi trong file thì đương nhiên có thể dùng vòng lặp DO thực hiện 200 lần đọc và tính toán với số liệu đọc được. Nhiều khi ta không biết trước có bao nhiêu dòng ghi trong file, nhưng ta biết dòng ghi cuối cùng chứa những giá trị đặc biệt làm cho chương trình của chúng ta có thể kiểm tra được. Thí dụ, nếu một file chứa các số liệu về thời gian và số đo nhiệt độ dưới dạng hai cột, thì cả hai cột ở dòng cuối cùng nên chứa hai số –999 để ký hiệu rằng đây là dòng cuối cùng của file. Trong trường hợp này ta có thể lập vòng lặp *While* để đọc các dòng số liệu và điều kiện kết thúc vòng lặp này là hai giá trị thời gian và nhiệt độ đều bằng –999. Có trường hợp ta không biết có bao nhiêu dòng ghi và ở cuối file cũng không có các giá trị đặc biệt để nhận biết. Khi đó ta phải nhờ đến các *tuỳ chọn* (options) của lệnh READ.

6.3.1. Số dòng ghi được chỉ định

Nếu ta biết chắc số dòng ghi, có thể dùng vòng lặp DO để xử lý file. Khi tạo lập file, ngay ở dòng ghi đầu, ta nên ghi một số thông tin chuyên

dụng về file, trong đó có số dòng ghi (số số liệu) trong file. Về sau, mỗi lần bổ sung số liệu vào file dữ liệu, ta cần sửa lại dòng ghi này. Khi xử lý file, ta đọc số này vào một biến. Sau đó dùng vòng lặp DO với biến đó làm giới hạn cuối của vòng lặp để đọc hết số liệu trong file.

Thí dụ 14: Cách đọc file có thông tin về số dòng số liệu ở dòng đầu file. Giả sử file có tên là SOLIEU.DAT chứa các giá trị trung bình ngày của nhiệt độ, độ ẩm không khí và áp suất khí quyển tại một trạm quan trắc trong nhiều ngày. Mỗi dòng của file ghi tuần tự ba đại lượng trên cho một ngày. Riêng dòng thứ nhất ghi tổng số số liệu (số ngày). Đoạn chương trình đọc số liệu từ file này và tính giá trị trung bình của ba đại lượng có thể như sau:

```
      INTEGER N, K
      REAL ND, DA, AS, NDTB, DATB, ASTB
      OPEN (UNIT = 2, FILE = 'SOLIEU.DAT', STATUS = 'OLD')
      READ (2 ,*)  N
      IF (N .LT. 1) THEN
         PRINT *, ' TRONG FILE KHONG CO SO LIEU '
      ELSE
         NDTB = 0.0
         DATB = 0.0
         ASTB = 0.0
         DO 15 K = 1, N
            READ (2, *) ND, DA, AS
            NDTB = NDTB + ND
            DATB = DATB + DA
            ASTB = ASTB + AS
15       CONTINUE
         NDTB = NDTB / REAL (N)
         DATB = DATB / REAL (N)
         ASTB = ASTB / REAL (N)
         PRINT 25 , N, NDTB, DATB, ASTB
      END IF
25    FORMAT (1X, 'SO NGAY = ' , I5 , ' ND =' , F6.2 , ' DA =' ,
     * F6.2 , ' AS =' , F7.1)
      CLOSE (2)
      END
```

Trong thí dụ này, số số liệu được đọc từ dòng thứ nhất của file và gán vào biến N. Lệnh IF kiểm tra nếu $N<1$ thì thông báo không có số liệu; nếu có số liệu thì đọc hết tất cả số liệu và tính các giá trị trung bình. Và ta thấy biến N được dùng làm tham số giới hạn cuối của lệnh DO.

6.3.2. Dòng ký hiệu kết thúc dữ liệu

Những giá trị đặc biệt dùng để đánh dấu sự kết thúc của file dữ liệu gọi là ký hiệu kết thúc (*Trailer hay Flags*). Khi tạo lập file, ta thêm một số con số đặc biệt trong dòng ghi cuối cùng. Về sau, nếu ta thêm hoặc xóa đi một số số liệu trong file, sẽ không phải sửa lại số ghi tổng số số liệu. Tuy nhiên, nếu dùng phương pháp này để đánh dấu hết file, thì phải cẩn thận sao cho chương trình của chúng ta không được đưa những giá trị đặc biệt vào xử lý như các giá trị bình thường khác. Có thể chúng ta phải ghi chú về điều này ở dòng đầu file. Ngoài ra, nếu dòng số liệu bình thường chứa bao nhiêu giá trị thì dòng ký hiệu kết thúc cũng nên có chừng đó giá trị đặc biệt

để đảm bảo cho lệnh đọc không mắc lỗi chạy chương trình. Người ta thường chọn các giá trị đặc biệt sao cho chúng khác hẳn với những giá trị bình thường, dễ nhận ra khi xem bằng mắt các số liệu trong file, thí dụ như số nguyên lớn nhất 32767, một tập hợp các số chín như 99999. Ta cũng có thể dùng cách này để đánh dấu những giá trị khuyết trong các chuỗi số liệu (*Flags - cờ hiệu báo hết file hoặc khuyết số liệu*).

Thí dụ 15: ***Cách đọc file có dòng số liệu đánh dấu hết file ở cuối file và cờ hiệu báo khuyết số liệu***. Giả sử file dữ liệu với nội dung như trong thí dụ 14, nhưng được tạo ra theo cách đánh dấu kết thúc dữ liệu bằng dòng gồm ba cụm số 99999. Ngoài ra, trong các dòng số liệu bình thường có những giá trị khuyết, không quan trắc, được đánh dấu bằng con số 32767. Chương trình sau đây cho phép đọc và tính toán đúng các trị số trung bình của ba đại lượng:

```
      INTEGER N1, N2, N3
      REAL ND, DA, AS, NDTB, DATB, ASTB
      OPEN (UNIT = 2, FILE = 'SOLIEU.DAT', STATUS = 'OLD')
      N1 = 0
      N2 = 0
      N3 = 0
      NDTB = 0.0
      DATB = 0.0
      ASTB = 0.0
60    READ (2, *) ND, DA, AS
    IF (ND.NE. 99999 .OR. DA .NE. 99999 .OR. AS .NE. 99999) THEN
      IF (ND .NE. 32767.) THEN
         NDTB = NDTB + ND
         N1 = N1 + 1
      END IF
      IF (DA .NE. 32767.) THEN
         DATB = DATB + DA
         N2 = N2 + 1
      END IF
      IF (AS .NE. 32767.) THEN
         ASTB = ASTB + AS
         N3 = N3 + 1
      END IF
      GOTO 60
    END IF
    CLOSE (2)
    IF (N1 .EQ. 0 .AND. N2 .EQ. 0 .AND. N3 .EQ. 0)  THEN
      PRINT * , ' TRONG FILE KHONG CO SO LIEU '
    ELSE
      IF (N1 .GT. 0) PRINT * , ' NHIET DO TRUNG BINH = ',
     *  NDTB / REAL (N1)
      IF (N2 .GT. 0) PRINT * , ' DO AM TRUNG BINH = ',
     *  DATB / REAL (N2)
      IF (N3 .GT. 0) PRINT * , ' AP SUAT TRUNG BINH = ',
     *  ASTB / REAL (N3)
    END IF
    END
```

6.3.3. Sử dụng tuỳ chọn END

Trường hợp không biết số dòng dữ liệu trong file và không có dòng thông tin về dấu hiệu kết thúc dữ liệu trong file, ta phải sử dụng một kỹ

thuật khác. Lệnh READ trong Fortran có một tuỳ chọn giúp kiểm tra sự kết thúc của file và rẽ nhánh sang một lệnh được chỉ định nếu phát hiện hết file. Lệnh READ với tuỳ chọn này có dạng sau:

READ (*Số hiệu file* , * , END = *n*) *Danh sách các biến*

Khi nào còn dữ liệu trong file lệnh này thực hiện giống như lệnh

READ (*Số hiệu file* , *) *Danh sách các biến*

Tuy nhiên, nếu dòng dữ liệu cuối cùng đã đọc xong và ta thực hiện lệnh READ với tùy chọn END thì thay vì phạm lỗi thực hiện lệnh, điều khiển được chuyển tới lệnh có nhãn *n* trong tuỳ chọn END. Nếu lệnh READ thực hiện một lần nữa sau khi đã đạt đến cuối file, thì lỗi chạy chương trình sẽ xuất hiện.

Lệnh READ với tuỳ chọn END thực sự là một dạng đặc biệt của vòng lặp điều kiện *While*:

```
 5   READ (10, *, END = 15) TEMP
     SUM = SUM + TEMP
     N = N +1
     GOTO 5
15   PRINT * , SUM
```

Dạng đặc biệt này của vòng lặp điều kiện chỉ nên thực hiện khi nào ta không biết số dòng dữ liệu và không có dòng ký hiệu báo hết dữ liệu. Việc chọn kỹ thuật hợp lý để đọc dữ liệu từ file phụ thuộc vào thông tin trong file dữ liệu.

Thí dụ 16: Sử dụng tùy chọn END. Với file dữ liệu nội dung như trong thí dụ 14, giả sử không có dòng đầu tiên thông báo về độ dài chuỗi dữ liệu, ta thực hiện chương trình tính các trị số trung bình như sau:

```
      INTEGER N, K
      REAL ND, DA, AS, NDTB, DATB, ASTB
      OPEN (UNIT = 2, FILE = 'SOLIEU.DAT', STATUS = 'OLD')
      NDTB = 0.0
      DATB = 0.0
      ASTB = 0.0
      N = 0
C  Nếu đọc hết số liệu tùy chọn END = 15 sẽ chuyển đến lệnh 15
  5   READ (2, *, END = 15)  ND, DA, AS
      NDTB = NDTB + ND
      DATB = DATB + DA
      ASTB = ASTB + AS
      N = N + 1
      GOTO 5
 15   CLOSE (2)
      IF (N .EQ. 0) THEN
         PRINT *, ' TRONG FILE KHONG CO DU LIEU'
      ELSE
         NDTB = NDTB / REAL (N)
         DATB = DATB / REAL (N)
         ASTB = ASTB / REAL (N)
         PRINT 25 , N, NDTB, DATB, ASTB
      END IF
 25   FORMAT (1X, ' SO NGAY = ', I5 , ' ND =' , F6.2 , ' DA =' ,
     *  F6.2 , ' AS =' , F7.1)
      END
```

6.4. Tạo lập các file dữ liệu

Để tạo mới file dữ liệu, chúng ta sử dụng các lệnh OPEN và WRITE. Tuy nhiên, trước khi ta bắt đầu viết các lệnh Fortran, cần cân nhắc xem sau này đọc file dữ liệu sẽ sử dụng kỹ thuật nào trong ba kỹ thuật đã mô tả trong mục 6.3.

Khi tạo lập file với dòng ký hiệu báo hết dữ liệu phải cẩn thận lựa chọn giá trị dùng làm ký hiệu. Phải tin chắc rằng giá trị được chọn làm giá trị báo hết dữ liệu không thể nào nhầm với giá trị dữ liệu thực sự. Có thể chúng ta phải có ghi chú ở đầu file để mọi người dùng file được biết.

Nếu ta quyết định tạo file với thông tin báo tổng số dòng dữ liệu trên đầu file, thì phải chú ý cập nhật dòng đầu file mỗi khi bổ sung hoặc cắt bớt số dòng dữ liệu. Nếu số dòng dữ liệu không đúng, thì hoặc chương trình đọc sẽ đọc số dòng dữ liệu ít hơn số dòng thực có, hoặc chương trình cố đọc nhiều dòng hơn trong file thực có và dẫn đến lỗi trong khi chạy chương trình.

So sánh ba phương án tổ chức thông tin dữ liệu ở trên, ta thấy về phương diện tối ưu chương trình thì cách dùng số báo số dòng dữ liệu ở đầu file là tốt hơn cả, vì khi đọc được tổng số dòng số liệu ta có thể đọc hết dữ liệu bằng vòng lặp DO, trong khi hai phương án sau chương trình luôn luôn phải kiểm tra biểu thức lôgic trong khi đọc lặp. Ngoài ra rất có thể có những nhiệm vụ xử lý không cần đọc hết file, mà chỉ cần đọc số lượng số liệu của file ở dòng đầu.

6.5. Kỹ thuật trợ giúp tìm lỗi chương trình

Thật vô nghĩa nếu một chương trình xử lý dữ liệu mà lại đọc sai dữ liệu trong file. Mà điều này không phải là không bao giờ xảy ra. Trường hợp số dòng dữ liệu thực tế trong file có ít hơn số vòng lặp đọc dữ liệu thì chương trình sẽ báo lỗi chạy chương trình. Khi đó chúng ta buộc phải xem lại chương trình hoặc xem lại file dữ liệu và dễ dàng phát hiện lỗi ở đâu. Tuy nhiên có những trường hợp lỗi chạy chương trình không phát sinh, nhưng kết quả chương trình cho ra sai. Nếu kết quả sai vô lý, rõ ràng thì chúng ta cũng biết và tìm nguyên nhân ở chương trình hay ở file dữ liệu. Đáng sợ nhất là những trường hợp đọc "nhầm dữ liệu", đọc hơi thiếu dữ liệu. Khi đó chương trình làm việc bình thường, kết quả tỏ ra chấp nhận được, nhưng thực chất là sai hoặc không chính xác. Do đó, trong lập trình phải rất thận trọng với file dữ liệu.

Khi tìm lỗi một chương trình làm việc với các file dữ liệu, điều rất quan trọng là kiểm tra xem các lệnh nhập, xuất dữ liệu có làm việc đúng đắn, chính xác không.

Trong thực tế các file dữ liệu có thể do bản thân người lập trình xây dựng, cũng có thể người lập trình nhận được trong quá trình trao đổi dữ liệu với đồng nghiệp của mình. Trong những trường hợp đó, trước khi viết những lệnh đọc file dữ liệu phải nghiên cứu kĩ cấu trúc của file, đọc kĩ tài liệu mô tả file, phải tin chắc tuyệt đối những thông tin trong file là những thông tin gì, cách thức ghi ở trong đó ra sao thì mới đọc file đúng và chính xác. Đặc biệt lần đầu tiên làm việc với một loại file phải kiểm tra kĩ lưỡng kết quả đọc file.

Hãy nên nhớ rằng trong số những yếu tố của Fortran thì vấn đề làm việc với file có thể xem là vấn đề khó nhất và lý thú nhất.

Với các file dữ liệu nhập, ta nên thử chương trình với một file dữ liệu nhỏ, sao cho ta có thể in lên màn hình từng dòng dữ liệu khi chương trình đọc vào. Hãy kiểm tra xem chương trình có bỏ qua dòng dữ liệu, hoặc một giá trị nào không. Nếu file dữ liệu có ghi số dòng dữ liệu, thì hãy in số đó ra sau khi đọc.

Với các file dữ liệu xuất, sau khi tạo lập ra nó, hãy mở ra xem lại nội dung file. Nên xem cấu trúc file có như ta dự định không, những giá trị có đúng là nằm ở những chỗ nó cần nằm không. Ngoài ra cần phải kiểm tra

file đầu ra trong nhiều phương án chạy chương trình. Rất có thể trong một trường hợp ta thấy mọi chuyện đều ổn, nhưng đến trường hợp khác thì tình hình không phải như vậy. Chỉ có kiểm tra kĩ thì mới tránh được những lỗi tiềm ẩn khó nhận biết trong chương trình.

Bài tập

1. File dữ liệu LAB1 chứa nhữnng thông tin về thời gian và nhiệt độ trên mỗi dòng như sau:

0.0	26.5	(*dòng 1*)
0.5	28.7	(*dòng 2*)
1.0	29.1	(*dòng 3*)
1.5	29.2	(*dòng 4*)
2.0	29.4	(*dòng 5*)
2.5	29.7	(*dòng 6*)

Hãy cho biết giá trị của các biến sau khi mỗi nhóm lệnh dưới đây thực hiện. Giả sử rằng trước khi thực hiện mỗi nhóm lệnh đó, thì file dữ liệu đã được mở và chưa từng có một lệnh READ nào được thực hiện:

1) READ (1, *) TIM, TEM
2) READ (1, *) TIM1, TEM1, TIM2, TEM2
3) READ (1, *) TIM
4) READ (1, *) TIM1, TEM1
   READ (1, *) TEM
   READ (1, *) TIM2, TEM2
5) READ (1, *) TIM1
6) READ (1, *) TIM1, TIM2
   READ (1, *) TEM1
   READ (1, *) TEM1, TEM2
   READ (1, *) TIM2
   READ (1, *) TEM2

2. File dữ liệu có tên CONDAO.TEM có nội dung ghi như sau: Dòng thứ nhất - tiêu đề báo rằng đây là số liệu về biến thiên nhiệt độ không khí tại trạm Côn Đảo. Dòng thứ hai - đơn vị đo (°C). Dòng thứ ba tuần tự ghi các tham số: số năm quan trắc, tháng, năm bắt đầu và tháng, năm kết thúc quan trắc. Dòng thứ tư gồm 12 cột ghi các tháng trong năm. Các dòng tiếp sau tuần tự ghi những giá trị nhiệt độ ứng với từng tháng thành 12 cột, trong đó những tháng khuyết số liệu được ghi bằng số –9.9 (bảng phía dưới).

OSCILLATION OF TEMPERATURE OF THE AIR AT STATION CONDAO
degree C
12 1 1979 12 1990

1	2	3	4	5	6	7	8	9	10	11	12
25.2	25.7	27.3	28.7	29.0	27.7	27.2	27.5	27.2	-9.9	26.8	25.3
25.0	25.6	27.2	28.5	28.6	27.6	27.8	27.0	27.4	26.7	26.8	25.8
24.6	25.0	26.9	28.6	28.1	28.0	28.0	27.9	26.9	27.0	26.3	25.0
24.5	25.2	-9.9	-9.9	28.3	27.8	27.2	27.2	27.4	26.7	27.4	26.1
25.3	25.7	26.6	28.2	29.1	28.2	28.0	27.5	26.9	27.2	25.9	25.5
24.7	25.1	25.9	27.9	27.8	27.2	27.2	27.8	26.3	26.5	26.9	25.8
25.4	26.4	27.0	27.3	27.7	28.3	27.5	28.0	26.8	26.5	26.9	25.7
24.5	24.8	25.6	28.1	28.8	28.2	27.8	27.6	26.8	26.8	26.4	25.5
25.2	25.2	27.0	28.9	28.5	28.0	28.6	27.7	27.3	26.9	27.2	25.9
25.8	26.6	27.1	28.3	28.1	28.0	27.3	27.6	27.2	27.0	26.1	24.8
25.3	24.7	25.9	27.2	27.6	27.9	27.7	27.3	27.4	26.6	26.7	25.4
25.6	26.0	27.2	29.0	28.5	28.3	28.2	27.8	27.6	27.4	26.6	25.8

Hãy lập đoạn chương trình đọc file này và in lại lên màn hình toàn bộ dữ liệu gốc cùng biến trình năm trung bình của nhiệt độ không khí ở dòng cuối cùng.

3. Lập đoạn chương trình đọc file dữ liệu với nội dung như trong bài tập 2 và ghi lại thành file cùng tên, áp dụng kỹ thuật dùng dòng ký hiệu đánh dấu kết thúc dữ liệu trong mục 6.3.2.

4. Trong file tên là DATA1, mỗi dòng ghi thời gian tính bằng giây và nhiệt độ tính bằng độ C. Dòng cuối cùng là dòng báo hết dữ liệu chứa giá trị –999.9 cho cả thời gian và nhiệt độ. Hãy đọc file dữ liệu này và sắp xếp giá trị nhiệt độ theo thứ tự giảm dần. In chuỗi nhiệt độ đã sắp xếp thành dạng 10 giá trị một dòng. Giả sử trong file có không quá 200 dòng dữ liệu.

5. Trong file tên là DATA2, mỗi dòng ghi thời gian tính bằng giây và nhiệt độ tính bằng độ C. Không có dòng tiêu đề và không có dòng báo hết dữ liệu. Hãy đọc file dữ liệu này và in ra số giá trị nhiệt độ, giá trị nhiệt độ trung bình và số giá trị nhiệt độ lớn hơn trung bình. Giả sử trong file có không quá 200 dòng dữ liệu.

6. Viết chương trình sửa lại file CONDAO.TEM trong bài tập 2 sao cho ở mỗi dòng số liệu có chỉ năm quan trắc tương ứng ở đầu dòng, giá trị nhiệt độ trung bình năm ở cuối dòng và giá trị nhiệt độ trung bình nhiều năm của từng tháng ở dòng dưới cùng.

7. Viết chương trình tìm nghiệm gần đúng với sai số cho phép 0,0001 của phương trình $e^{-x} - \frac{1}{3}\sqrt{e^x + 3{,}7} - x = 0$ trong khoảng [0, 2] theo phương pháp lặp và in thông báo kết quả lên màn hình với 4 chữ số thập phân.

8. Viết chương trình nhập một số tự nhiên n nhỏ hơn 21, một số thực x bất kỳ nhỏ hơn 1. Xác định tổng:

$$\frac{\sin x}{\cos x} + \frac{\sin x + \sin 2x}{\cos x + \cos 2x} + \frac{\sin x + \sin 2x + \sin 3x}{\cos x + \cos 2x + \cos 3x} + ... + \frac{\sin x + \sin 2x + ... + \sin nx}{\cos x + \cos 2x + ... + \cos nx}$$

Chương 7 - Sử dụng biến có chỉ số trong Fortran

Trong chương 2, mục 2.3 đã xét cách khai báo kiểu biến có chỉ số và khái niệm mảng trong Fortran, nêu một số đặc điểm về lưu giữ đối với các biến có chỉ số hay gọi là biến mảng.

Chương này sẽ cung cấp thêm phương pháp lưu giữ và xử lý những nhóm giá trị mà không cần cung cấp tên một cách tường minh cho từng giá trị đó. Trong thực tế, ta thường xử lý một nhóm các giá trị ít nhiều liên hệ hoặc hoàn toàn không liên hệ với nhau. Trong trường hợp này, nếu sử dụng biến mảng, cả nhóm dữ liệu sẽ có một tên chung, nhưng những giá trị riêng biệt có chỉ số riêng duy nhất. Kỹ thuật này cho phép ta phân tích dữ liệu sử dụng các vòng lặp một cách thuận tiện. Trong các mục dưới đây sẽ bổ sung thêm những cấu trúc, những lệnh của Fortran cho phép thao tác thuận lợi với các biến mảng, kỹ thuật đọc dữ liệu từ file để gán vào các biến mảng v.v...

Mảng là yếu tố quan trọng và mạnh mẽ nhất của Fortran. Nếu so sánh với một số ngôn ngữ lập trình khác, thí dụ như Pascal, ta thấy trong Fortran cho phép khai báo những mảng dữ liệu rất lớn và thao tác rất mềm dẻo. Nhiều khi khả năng khai báo mảng dữ liệu lớn làm cho thuật giải của chương trình xử lý trở nên đơn giản. Ngoài ra, sử dụng mảng đúng đắn và thành thạo sẽ giúp chúng ta viết những chương trình hoặc những đoạn chương trình rất ngắn gọn.

7.1. Mảng một chiều

Trong lập trình, mảng một chiều thường dùng để biểu diễn một dòng hoặc một cột dữ liệu.

Về phương diện ngôn ngữ, một mảng là *một nhóm địa chỉ lưu giữ* trong bộ nhớ máy tính có cùng tên. Từng thành phần của mảng được gọi là phần tử mảng và được phân biệt với phần tử khác bởi tên chung kèm theo chỉ số trong cặp dấu ngoặc. Những chỉ số được biểu diễn bằng những số nguyên liên tiếp nhau, thường là bắt đầu (chỉ số đầu) bằng số nguyên 1. Những trường hợp dùng chỉ số đầu khác 1 thường liên quan tới tính thuận tiện thao tác các công thức toán học hoặc phương diện thực tiễn. Thí dụ muốn biểu diễn các hệ số a của phương trình hồi quy nhiều biến liên hệ giữa đại lượng y và các đại lượng $x_1, x_2, ..., x_m$

$$y = a_0 + a_1x_1 + a_2x_2 + ... + a_mx_m$$

ta có thể dùng mảng một chiều với tên A để chỉ tất cả các hệ số, kể cả hệ số tự do, của phương trình hồi quy này và khai báo như sau:

```
REAL  A (0 : 20)
```

Trong trường hợp này phần tử thứ nhất A(0) của mảng A biểu diễn hệ số a_0. Như vậy rất thuận tiện trong khi sử dụng các công thức của đại số.

Nếu ta có tập hợp số liệu về lượng mưa năm trong thế kỷ này tại một trạm khí tượng nào đó, ta có thể dùng mảng

```
REAL RAIN (1900 : 2000)
```

Trong trường hợp này, nếu muốn truy cập lượng mưa năm 1985, ta chỉ định phần tử mảng RAIN (1985).

Để đọc dữ liệu vào một mảng một chiều từ bàn phím hoặc từ file dữ liệu, ta sử dụng lệnh READ. Nếu muốn đọc toàn bộ mảng, ta dùng tên mảng không có các chỉ số. Ta cũng có thể chỉ định những phần tử cụ thể trong lệnh READ, thí dụ

```
READ *, B
READ *, B(1), B(2), B(3)
```

Cần chú ý rằng, trong thí dụ này, nếu mảng B theo khai báo chứa 3 phần tử thì hai lệnh READ trên tương đương nhau. Nhưng nếu mảng B chứa 8 phần tử thì có sự khác nhau quan trọng giữa hai lệnh READ trên đây, là vì: lệnh thứ nhất đọc vào toàn bộ 8 phần tử của mảng B, trong khi lệnh thứ hai chỉ đọc các giá trị của ba phần tử đầu tiên.

Các giá trị của biến mảng còn có thể đọc với ***vòng lặp* DO *ẩn***. Thí dụ, nếu muốn đọc 5 phần tử đầu tiên của mảng B ta sử dụng lệnh READ như sau

```
READ *, (B (I) , I = 1 , 5)
```

Trong lệnh này, chúng ta thấy không có mặt từ khóa DO, chỉ có chỉ số I của biến mảng B biến thiên từ 1 tới 5 với gia số bằng 1. Như vậy với một lệnh READ máy đọc được liên tục 5 phần tử của mảng B.

Thí dụ 17: Một tập hợp 50 số liệu lượng mưa năm được lưu trong file dữ liệu, mỗi số liệu một dòng. Giả sử đơn vị file là 9. Viết nhóm lệnh đọc những số liệu này vào mảng LMUA.

Cách 1: Dùng lệnh READ đọc từng số, nhưng vòng lặp thực hiện 50 lần và đọc toàn bộ mảng:

```
      REAL LMUA (50)
      DO 10  I = 1 , 50
         READ (9, *) LMUA (I)
10    CONTINUE
```

Cách 2: Dùng lệnh READ không chứa chỉ số, nó sẽ đọc toàn bộ mảng, tức đọc liền 50 phần tử:

```
REAL  LMUA (50)
READ (9, *)  LMUA
```

Cách 3: Lệnh READ chứa vòng lặp ẩn:

```
REAL  LMUA (50)
READ (9, *)  (LMUA (I), I = 1, 50)
```

7.2. Lệnh DATA

Lệnh DATA là lệnh đặc tả, thuộc loại lệnh không thực hiện. Nó dùng để khởi tạo giá trị ban đầu cho các biến đơn và các mảng. Dạng tổng quát của lệnh DATA như sau

DATA *Danh sách tên biến* / *Danh sách hằng* /

Theo lệnh này các giá trị dữ liệu trong *danh sách hằng* nằm trong hai dấu gạch chéo được gán cho các biến trong *danh sách tên biến* theo tuần tự. Kiểu của các giá trị dữ liệu cũng nên phù hợp kiểu của các biến, sao cho máy tính không phải chuyển đổi. Các lệnh DATA phải đặt trước các lệnh thực hiện, tức ở gần đầu chương trình, ngay sau những lệnh mô tả kiểu như lệnh REAL, INTEGER, LOGICAL, DIMENSION...

Thí dụ, lệnh

```
DATA A , B, C , I  / 0.0 , 32.75 , –2.5 , 10 /
```

sẽ khởi tạo giá trị 0,0 cho biến A, 32,75 cho biến B, –2,5 cho biến C và 10 cho biến I.

Chú ý rằng lệnh DATA chỉ khởi tạo giá trị ở đầu chương trình. Lệnh DATA không thể sử dụng trong vòng lặp để tái tạo giá trị các biến. Nếu cần tái tạo các biến, ta phải sử dụng các lệnh gán. Lệnh DATA cũng không thể nằm trong chương trình con.

Nếu các giá trị lặp lại trong danh sách hằng, ta có thể dùng cách viết lệnh DATA ngắn gọn. Thí dụ, nếu muốn khởi tạo giá trị 1 cho các biến I, J, K và giá trị 0,5 cho các biến X, Y, Z, thì hai lệnh sau đây tương đương nhau:

```
DATA I, J, K, X, Y, Z / 1, 1, 1, 0.5, 0.5, 0.5 /
DATA I, J, K, X, Y, Z / 3*1, 3*0.5 /
```

Lệnh DATA có thể sử dụng để khởi tạo một hoặc một số phần tử của mảng. Thí dụ, các lệnh sau khởi tạo tất các các phần tử của mảng J và TIME:

```
INTEGER J (5)
REAL TIME (4)
DATA  J, TIME / 5*0, 1.0, 2.0, 3.0, 4.0 /
```

Nhóm lệnh

```
REAL HOUR (5)
DATA HOUR (1) / 10.0 /
```

chỉ khởi tạo một giá trị của phần tử đầu tiên của mảng HOUR, các phần tử từ thứ 2 đến 5 của nó chưa biết.

Có thể sử dụng vòng DO ẩn trong lệnh DATA. Thí dụ:

```
INTEGER Y (100)
DATA (Y (I), I = 1, 50)  / 50*0 /
```

khởi tạo giá trị 0 cho 50 phần tử đầu của mảng Y, 50 phần tử còn lại chưa được khởi tạo.

7.3. Mảng hai chiều

Các lệnh mô tả mảng hai chiều giống như với mảng một chiều, khác biệt duy nhất là dùng hai tham số kích thước mảng. Mỗi phần tử mảng được truy cập bởi tên mảng với hai chỉ số nằm trong cặp dấu ngoặc.

Trong thực tế lập trình người ta thường biểu diễn các ma trận, các bảng dữ liệu gồm một số cột, mỗi cột có một số dòng giá trị thành mảng hai chiều.

Thí dụ, ma trận các hệ số đứng trước các ẩn của hệ phương trình đại số tuyến tính $a_{i,j}$ $(i = 1..10, j = 1..10)$ thường biểu diễn bằng mảng hai chiều A với lệnh mô tả như sau

```
REAL  A(10, 10)
```

Các giá trị quan trắc từng giờ về mực nước biển trong vòng một tháng có thể biểu diễn thành một bảng số liệu gồm 31 dòng, 24 cột. Các dòng tuần tự ứng với các ngày trong tháng. Các cột tuần tự ứng với 24 giờ trong một ngày. Trong Fortran, bảng số liệu này có thể biểu diễn bằng mảng hai chiều

```
REAL  SLEV (31, 24)
```

theo cách này, khi thao tác với mực nước tại một ngày, giờ cụ thể nào đó, người ta chỉ cần chỉ định phần tử SLEV (I, J), với chỉ số thứ nhất I chỉ ngày, chỉ số thứ hai J chỉ giờ trong ngày đó. Khi cần tính mực nước trung bình ngày, thí dụ của ngày thứ nhất trong tháng, người ta chỉ cần cộng tất cả các phần tử với chỉ số I = 1:

$$\text{SLEV}(1,1) + \text{SLV}(1,2) + ... + \text{SLEV}(1,24)$$

Sử dụng các mảng rất tiện lợi khi lập chương trình phân tích, tính toán với những ma trận, những tập số liệu lớn.

Thí dụ 18*: *Lập ma trận đơn vị (ma trận vuông với các phần tử trên đường chéo chính bằng 1, còn tất cả các phần tử khác bằng 0). Thí dụ ma trận kích thước $n = 3$, tức có 3 dòng và 3 cột, sẽ là

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Đoạn chương trình Fortran thực hiện việc này sẽ như sau:

```
INTEGER  IDMAT(3,3)
DO I = 1, 3
   DO J = 1, 3
      IF (I .EQ. J) THEN
         IDMAT (I, J) = 1
      ELSE
         IDMAT (I, J) = 0
      ENDIF
   END DO
END DO
```

Thí dụ 19: Đọc các giá trị mảng hai chiều từ file dữ liệu. Giả sử có các số liệu về lưu lượng nước trung bình năm của một số con sông. Những số liệu này ghi trong file SONG.LLG. Dòng trên cùng của file ghi hai số nguyên tuần tự chỉ số năm quan trắc và số con sông. Sau đó có n dòng, mỗi dòng số liệu tuần tự ứng với một năm, trong mỗi dòng có m giá trị, mỗi giá trị ứng với một con sông. Ta dùng mảng hai chiều để biểu diễn tập số liệu này, chỉ số thứ nhất của mảng chỉ thứ tự năm, chỉ số thứ hai chỉ thứ tự con sông. Đoạn chương trình sau đây cho phép đọc số liệu từ file, tính lưu lượng trung bình của tất cả các sông và in kết quả lên màn hình.

```
REAL  SLL (100, 15), TB (15)
OPEN (1, FILE = 'SONG.LLG', STATUS = 'OLD')
READ (1, *)  N, M
```

```
      DO I = 1 , N
         READ (1, *) (SLL (I , J) , J = 1 , M)
      ENDDO
      CLOSE (1)
      DO J = 1, M
         TB (J) = 0.0
         DO I = 1, N
            TB (J) = TB (J) + SLL (I, J)
         ENDDO
      ENDDO
      PRINT 4,  (TB (J) , J = 1 , M)
4     FORMAT (1X, 15 F8.0)
```

Hãy lưu ý cách đọc số liệu lượng mưa trong chương trình này. Như đã mô tả cách ghi số liệu trong file, lượng mưa được ghi thành n dòng, mỗi dòng ứng với một năm, trên mỗi dòng lại có m giá trị lượng mưa ứng với m con sông. Muốn đọc liên tục số liệu trong n năm ta đã dùng hai vòng DO lồng nhau:

```
DO I = 1 , N
   READ (1, *) (SLL (I , J) , J = 1 , M)
END DO
```

trong đó vòng DO bên trong là vòng DO ẩn với chỉ số **J** chạy từ 1 đến M. Bằng vòng lặp ẩn này ta đã đọc được m giá trị số thực ứng với m sông trên cùng một dòng.

Một cách tổng quát, đây là cách đọc thường dùng nhất để bằng một lệnh đọc có thể nhận liên tiếp tất cả các phần tử trên một hàng của ma trận.

Nếu ta dùng hai vòng lặp thông thường:

```
DO I = 1, N
   DO J = 1, M
      READ (1, *)  SLL (I, J)
   END DO
END DO
```

thì sẽ phạm sai lầm, bởi vì hai vòng DO này tương đương với $n \times m$ lệnh READ, và như ta đã biết, mỗi lần lệnh READ thực hiện xong thì đầu đọc file sẽ xuống dòng mới. Như vậy máy sẽ đọc $n \times m$ dòng trong khi trong file chỉ có n dòng số liệu.

Ta phát triển cách dùng vòng DO ẩn cho trường hợp trên cùng một dòng trong file có hai đại lượng. Thí dụ, cũng là file số liệu như đã mô tả trong thí dụ 19, nhưng trên mỗi dòng ngoài m giá trị lưu lượng còn có m giá trị độ đục ứng với m con sông. Trong trường hợp này ta khai báo thêm một biến DD (100, 15) và lệnh đọc cả lưu lượng và độ đục sẽ là:

```
DO I = 1 , N
   READ (1, *) (SLL (I , J), J = 1 , M) , (DD (I , J), J = 1, M)
END DO
```

Trường hợp ở đầu mỗi dòng có ghi năm quan trắc, ta sẽ dùng

```
DO I = 1 , N
   READ (1, *) NAM (I), (SLL (I, J) , J = 1, M) , (DD (I , J) , J = 1, M)
END DO
```

7.3. Mảng nhiều chiều

Fortran cho phép sử dụng các mảng với số chiều tối đa bằng 7. Chúng ta có thể hình dung mảng ba chiều giống như hình hộp chữ nhật tạo bởi nhiều hình lập phương con. Các phần tử của mảng ba chiều giống như những hình lập phương con, xếp thành một số lớp, mỗi lớp có một số hàng và mỗi hàng có một số hình lập phương. Từ đó ta biểu diễn vị trí của một

phần tử nào đó như là vị trí của hình lập phương con: thứ tự của nó trong một hàng bằng chỉ số I, thứ tự hàng bằng chỉ số J và thứ tự lớp - chỉ số K.

Thí dụ, mảng ba chiều có thể định nghĩa bằng lệnh:

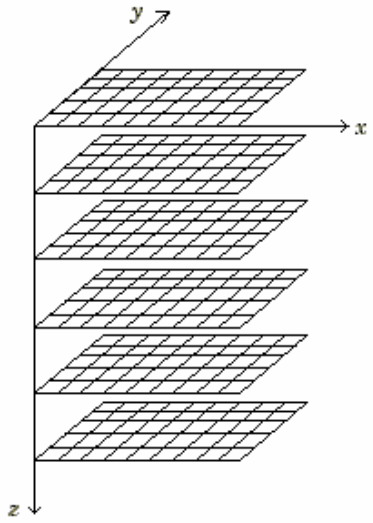
REAL T (3, 4, 4)

Nếu sử dụng tên mảng ba chiều không có chỉ số, ta xử lý mảng với chỉ số thứ nhất biến thiên nhanh nhất, chỉ số thứ hai biến thiên nhanh thứ hai và chỉ số thứ ba biến thiên chậm nhất. Thí dụ với mảng T, hai lệnh đọc sau đây là tương đương:

READ*, T

READ*, (((T(I, J, K), I =1, 3), J=1, 4), K=1, 4)

Tương tự ta hình dung mảng bốn chiều như là một chuỗi các mảng ba chiều...



Hình 7.1. Biểu diễn mảng ba chiều trong biển

Trong khí tượng thủy văn mảng ba chiều thường được dùng để biểu diễn những số liệu quan trắc trong không gian ba chiều. Thí dụ, ta có thể biểu diễn trường áp suất nước biển tại các điểm nút kinh, vĩ tuyến của một miền hình chữ nhật trên mặt biển và một số tầng sâu. Trong trường hợp này có thể quy ước chỉ số thứ nhất của mảng i biến thiên theo trục x hướng sang phía đông, chỉ số thứ hai j biến thiên theo trục y hướng lên bắc, còn chỉ số thứ ba k biến thiên theo trục z hướng thẳng đứng từ mặt xuống đáy biển để chỉ giá trị áp suất ứng với kinh độ, vĩ độ và một tầng sâu nào đó trong biển (hình 7.1). Trong khí tượng học, hai chỉ số đầu hoàn toàn tương tự, còn chỉ số thứ ba của mảng ba chiều biến thiên theo trục z hướng thẳng đứng từ mặt đất lên trên, có thể dùng để chỉ quan trắc tại một tầng cao.

Theo quy ước đó thì mảng hai chiều là một trường hợp riêng của mảng ba chiều dùng để biểu thị trường yếu tố khí tượng thủy văn nào đó trên một miền phẳng hình chữ nhật, thí dụ trường khí áp mặt đất, trường nhiệt độ nước mặt biển... Mảng bốn chiều có thể dùng để biểu diễn những trường ba chiều vừa mô tả ở trên nhưng tại nhiều thời điểm t khác nhau.

Trong thủy văn, chỉ số thứ nhất của mảng ba chiều thường dùng biểu diễn yếu tố quan trắc tại các độ sâu khác nhau của một mặt cắt, dọc theo sông ta có có thể có nhiều mặt cắt được biểu diễn bằng biến thiên của chỉ số thứ hai, yếu tố quan trắc lại có thể biến đổi theo thời gian và được chỉ định bằng chỉ số thứ ba. Nếu xét nhiều sông cùng một lúc, ta cần đến mảng bốn chiều.

Chính là trong khí tượng, hải dương học chúng ta được biết tới những mô hình dự báo thời tiết hay hoàn lưu và nhiệt muối đại dương thường sử dụng các trường ba chiều ban đầu và phát sinh ra những trường bốn chiều với kích thước khổng lồ (do độ phân giải không gian cao và bước thời gian mô phỏng, dự báo nhỏ) phải lưu trữ và quản lý trong máy tính.

Các mảng có số chiều lớn hơn bốn có thể là khó hình dung trực quan

hơn. Tuy nhiên, nếu chúng ta quy ước rõ ràng, nhất quán các chỉ số thứ nhất, thứ hai... tương ứng với biến số nào trong thực tế và nắm vững quy tắc biến thiên chỉ số của mảng thì vẫn có thể truy cập, thao tác đúng với một phần tử bất kỳ của mảng trong chương trình.

Thí dụ 20: Tính tần suất mưa. Số liệu giá trị ngày của các yếu tố khí tượng thủy văn tại trạm Hòn Dấu được lưu trong file HONDAU.MAT có quy cách ghi như sau: Dòng trên cùng ghi tên trạm. Dòng thứ 2 có hai số nguyên viết cách nhau lần lượt chỉ tổng số ngày quan trắc và số yếu tố được quan trắc. Dòng thứ ba có 6 số nguyên viết cách nhau lần lượt chỉ ngày, tháng, năm đầu và ngày, tháng, năm cuối quan trắc. Dòng thứ 4 là tiêu đề cột liệt kê tên tất cả các yếu tố được quan trắc, mỗi tên được ghi với độ rộng 8 vị trí. Các dòng tiếp theo lần lượt ghi giá trị của các yếu tố, mỗi dòng một ngày. Giả sử lượng mưa ngày ghi ở cột số 6. Viết chương trình đọc và tính xem trong suốt thời gian quan trắc có bao nhiêu lần mưa kéo dài 1 ngày, bao nhiêu lần mưa kéo dài 2 ngày liền, bao nhiêu lần mưa kéo dài 3 ngày liền...

```
      REAL  X (5000)
      INTEGER TS (5000)
      OPEN (1, FILE = 'HONDAU.MAT', STATUS = 'OLD')
      READ (1, *)
      READ (1, *) N
      READ (1, *)
      READ (1, *)
      DO I = 1, N
         READ (1, *) (X (I), J = 1, 6)
      END DO
      CLOSE (1)
      TS = 0
      I = 1
100   IF (I .GT. N) GOTO 15
      IF (X (I) .EQ. 0.0) THEN
         I = I + 1
         GOTO 100
      ELSE
         J = 1
300      IF (I .EQ. N .OR. X(I + 1) .EQ. 0.0) THEN
            TS (J) = TS (J) + 1
            I = I + 1
            GOTO 100
         ELSE
            J = J + 1
            I = I + 1
            GOTO 300
         END IF
      END IF
 15   I = N
 16   IF (TS (I) .EQ. 0) THEN
         I = I - 1
         GOTO 16
      ELSE
         DO N = 1, I
            PRINT '(2I5)', N, TS (N)
         END DO
      END IF
      END
```

Thí dụ 21: Tính ma trận tương quan của tập số liệu quan trắc các yếu tố khí tượng thủy văn. Với file số liệu trong thí dụ 20, viết chương trình đọc các thông tin cần thiết trong file và in ma trận tương quan của các yếu tố quan trắc lên màn hình.

Ta thấy, một cách tự nhiên mỗi chuỗi giá trị ngày của một yếu tố quan trắc có thể được biểu diễn thành mảng một chiều, chỉ số mảng sẽ biến thiên theo thứ tự ngày quan trắc. Tuy nhiên, ta có thể gộp tất cả các mảng một

chiều thành một mảng hai chiều với chỉ số thứ hai biến thiên theo thứ tự yếu tố quan trắc: 1, 2, ... Bằng cách dùng mảng hai chiều X (5000, 15) như trong chương trình dưới đây sẽ rất thuận tiện cho việc sử dụng các vòng lặp DO với tham số đếm của vòng DO đồng thời là chỉ số của mảng.

```
      REAL  X (5000, 15), MX (15), DX (15) , R (15, 15)
      OPEN (1, FILE = 'HONDAU.MAT', STATUS = 'OLD')
      READ (1, *)
      READ (1, *) N, M
      READ (1, *)
      READ (1, *)
      DO I = 1, N
         READ (1, *) (X (I, J), J = 1, M)
      END DO
      CLOSE (1)
C     Tính trung bình và độ lệch quân phương của M yếu tố
      DO I = 1, M
         MX (I) = X (1, I)
         DX (I) = X (1, I)*X (1, I)
         DO J = 2, N
            MX (I) = MX (I) + X (J, I)
            DX (I) = DX (I) + X (J, I) * X(J, I)
         END DO
         MX (I) = MX (I) / N
         DX (I) = SQRT (DX (I) / N - MX (I) * MX (I))
      END DO
C     Tính ma trận tương quan
      DO I = 1, M - 1
         DO J = I + 1, M
            R (I, J) = 0.0
            DO K = 1, N
               R (I, J) = R (I, J) + X (K, I) * X (K, J)
            END DO
            R (I, J) = R (I, J) / N - MX (I) * MX (J)
            R (I, J) = R (I, J) / (DX (I) * DX (J))
         END DO
      END DO
      DO I = 1, M
         R (I, I) = 1.0
      END DO
      DO I = 1, M
         PRINT 4, (R (K, I), K = 1, I - 1), (R (I, J), J = I, M)
      END DO
4     FORMAT (<M>F6.2)
      END
```

7.4. Những điều cần chú ý khi sử dụng các mảng

Trong các mục trước của chương này ta đã học sử dụng một mảng - một nhóm các địa chỉ lưu giữ các giá trị có một tên chung, nhưng phân biệt với nhau bởi một hoặc một số chỉ số. Mảng là một yếu tố mạnh mẽ nhất trong Fortran, vì nó cho phép lưu giữ một tập hợp dữ liệu lớn để dễ xử lý trong chương trình của chúng ta.

Mặc dù với tiện lợi cơ bản như trên, các mảng cũng thường có thể gây ra những lỗi mới. Một khi bạn dự định sử dụng mảng để mô tả dữ liệu, hãy tự hỏi "ta có cần sử dụng dữ liệu này nhiều lần không" và "dữ liệu này có cần phải lưu trước khi ta sử dụng nó không". Nếu câu trả lời cho các câu hỏi trên là "không", nên hạn chế dùng mảng, mà dùng các biến đơn.

Một khi mảng là cần thiết, nhưng chương trình làm việc sai, trước hết hãy kiểm tra những điều sau đây:

♠ Kích thước mảng: Mô tả mảng phải chỉ ra số phần tử tối đa dự định lưu giữ trong mảng. Mặc dù chúng ta không nhất thiết phải dùng hết tất cả các phần tử của mảng, nhưng chúng ta không được sử dụng nhiều phần tử

hơn so với số phần tử đã mô tả ở phần khai báo của chương trình. Vậy với mỗi bài toán cụ thể nếu cần sử dụng mảng, hãy hình dung trước kích thước tối đa của mỗi chiều của mảng để khai báo cho đúng, có thể hơi dư ra một ít, nhưng dư nhiều quá sẽ tốn bộ nhớ, còn khai báo thiếu thì khi chạy chương trình sẽ phát sinh lỗi lôgic.

♠ Chỉ số mảng: Hãy kiểm tra từng chỉ số, đặc biệt những chỉ số là biểu thức số học, để tin chắc rằng nó là số nguyên nằm trong giới hạn đúng đắn, không vượt ra ngoài khoảng biến thiên của chỉ số. Nếu chỉ số mảng vượt ra ngoài giới hạn cho phép thì hãy xem các biến trong biểu thức số học tính chỉ số có bị nhầm không. Có thể dùng lệnh in lên màn hình để theo dõi diễn biến của chỉ số.

♠ Vòng lặp DO: Nếu bạn dùng chỉ số mảng làm tham số đếm của vòng lặp DO, hãy tin chắc rằng bạn đã sử dụng đúng tên biến trong chương trình của bạn. Thí dụ, nếu trong một vòng lặp DO với mảng ba chiều bạn định cho chỉ số thứ ba của mảng (K) biến thiên, hãy kiểm tra xem bạn có dùng I thay vì K không. Lỗi thường gặp là chỉ số đảo ngược: Hãy tự hỏi chỗ này cần B (K, L) hay B (L, K)? Hãy có ý thức về đặt tên cho chỉ số. Tập quán chung là sử dụng biến I cho chỉ số thứ nhất, J - thứ hai và K - thứ ba; như vậy rất tiện lợi khi sử dụng các vòng lặp lồng nhau và chỉ số đếm của vòng lặp DO đồng thời là chỉ số mảng.

Bài tập

1. File dữ liệu với đơn vị file 9 chứa 28 số liệu lượng mưa ngày trong bốn tuần lễ liên tiếp, ghi thành 4 dòng, mỗi tuần một dòng. Nhóm lệnh sau

```
REAL  DMUA (28)
DO I = 1, 28
    READ (9, *) DMUA (I)
END DO
```

có đọc đúng các số liệu lượng mưa ứng với từng ngày không. Nếu không đúng, chỉ ra phương án đọc đúng.

2. Viết chương trình cho phép đọc từ bàn phím ba số nguyên, kiểm tra xem ba số nguyên đó có thể chỉ ngày, tháng, năm hợp lý không. Kết quả kiểm tra ghi thành dòng thông báo thích hợp lên màn hình.

3. Viết chương trình đọc một chuỗi Y gồm 20 giá trị thực từ file EXPER, trong đó mỗi giá trị ghi trên một dòng. Lập một chuỗi Z gồm 20 giá trị thoả mãn các điều kiện:

$$Z_1 = Y_1;\ \ Z_{20} = Y_{20};\ \ Z_i = \frac{Y_{i-1} + Y_i + Y_{i+1}}{3}\ \ (i = 2..19)$$

In chuỗi xuất phát và chuỗi mới cạnh nhau thành bảng hai cột.

4. Viết chương trình dọc file RAIN chứa bảng dữ liệu lượng mưa gồm 12 dòng (mỗi dòng tương ứng một tháng) và 5 cột (mỗi cột tương ứng một năm trong các năm 1978-1982). Xác định và in bảng thông tin sau đây:

```
LUONG MUA TRUNG BINH
1978 - XXX.XX
1979 - XXX.XX
1980 - XXX.XX
1981 - XXX.XX
1982 - XXX.XX
LUONG MUA CUC DAI
THANG XX NAM XXXX
LUONG MUA CUC TIEU
THANG XX NAM XXXX
```

5. File dữ liệu tên là SCS1.TEM ghi số liệu về trường nhiệt độ nước biển trung bình tháng 1 ở vùng biển Đông có quy cách ghi như sau:

- Dòng thứ nhất gồm tuần tự các tham số: kinh tuyến biên phía tây, kinh tuyến biên phía đông, vĩ tuyến biên phía nam, vĩ tuyến biên phía bắc (các số thực) của vùng, bước lưới theo phương tây đông, bước lưới theo phương bắc nam (đo bằng phút, các số nguyên) của lưới.

- Dòng thứ hai ghi kích thước của ma trận số liệu (các số nguyên) theo dòng (phương bắc nam), theo cột (phương tây đông), theo chiều sâu từ mặt biển xuống dưới và một số nguyên –32767 chỉ giá trị khuyết của số liệu nhiệt độ.

- Phần còn lại gồm: một số nguyên chỉ tầng sâu quan trắc (mét) ghi ở một dòng; sau đó là mảng số liệu nhiệt độ ứng với tầng đó ghi thành các dòng từ bắc xuống nam, các cột từ tây sang đông (các số thực không dính nhau). Tiếp tục như vậy cho đến tầng sâu dưới cùng.

Hãy viết chương trình đọc dữ liệu, chọn ra một profil nhiệt độ cho điểm bất kỳ thuộc miền tính. Kết quả ghi lên màn hình như sau:

KINH DO XXX.XX

VI DO XX.XX

TANG (m)NHIET DO

XXXX XX.XX

XXXX XX.XX

. . . . .

6. Cho file dữ liệu SCS1.TEM đã mô tả trong bài tập 5. Hãy viết chương trình đọc dữ liệu và tính các giá trị nhiệt độ trung bình của từng tầng quan trắc và giá trị nhiệt độ trung bình toàn biển kể từ tầng mặt cho tới tầng quan trắc dưới cùng.

7. Cho file dữ liệu SCS1.TEM đã mô tả trong bài tập 5. Hãy viết chương trình đọc dữ liệu và in ra file SECT17.TEM một bảng số liệu nhiệt độ nước của mặt cắt dọc vĩ tuyến 17°N với quy cách như sau:

- Dòng trên cùng là tiêu đề:

"Phân bố nhiệt độ nước trên mặt cắt dọc vĩ tuyến 17".

- Dòng thứ hai liệt kê các kinh độ từ tây sang đông.

- Các dòng tiếp dưới ghi độ sâu tầng quan trắc ở mỗi đầu dòng tương ứng, sau đó là các giá trị nhiệt độ nước (lấy đến hai chữ số thập phân) ghi thẳng cột với những kinh độ tương ứng đã liệt kê ở dòng thứ hai. Những giá trị khuyết (–32767) ghi bằng số 99.99 hoặc năm dấu hoa thị (*****).

Chương 8 - Chương trình con loại hàm

Khi xây dựng chương trình giải một bài toán tương đối phức tạp, ta sẽ thấy chương trình thường dài và khó đọc. Nhiều khi cùng một số thao tác như nhau được thực hiện lặp lại ở một số chỗ trong một chương trình cũng làm cho chương trình của chúng ta trở thành dài hơn. Những vấn đề này có thể khắc phục bằng cách sử dụng những chương trình con (subprogram) - là một nhóm các lệnh được tách riêng ra và sau đó sẽ được gọi thực hiện khi cần trong chương trình của chúng ta. Trong Fortran có hai loại chương trình con: chương trình con loại hàm (function) và chương trình con loại thủ tục (subroutine). Trong mục 2.4 chương 2 đã giới thiệu và thỉnh thoảng trong các bài khác chúng ta đã sử dụng một vài hàm chuẩn hay hàm riêng của Fortran. Thí dụ, khi tính sin của một góc ta dùng hàm SIN, khi cần giá trị tuyệt đối của một đại lượng ta dùng hàm ABS... Những hàm này thực chất cũng là những chương trình con, nhưng chúng đã được xây dựng sẵn (hàm chuẩn) và nằm trong bộ biên dịch, chúng ta chỉ việc gọi trực tiếp trong chương trình khi cần. Trong chương này sẽ tóm tắt về những đặc điểm của các hàm chuẩn. Sau đó ta học cách tự xây dựng những chương trình con loại hàm để giải quyết những bài toán riêng của mình. Những chương trình con loại thủ tục sẽ xét trong chương 9.

8.1. Các hàm chuẩn

Một hàm tính ra một giá trị, thí dụ căn bậc hai của một số hay giá trị trung bình của một mảng. Fortran có rất nhiều hàm chuẩn (xem danh sách các hàm chuẩn trong phụ lục 1).

Những đặc điểm chính của các hàm chuẩn là:

1) Tên hàm và các giá trị đầu vào (các đối số) cùng thể hiện một giá trị.

2) Một hàm không thể được sử dụng ở vế trái của dấu = trong một lệnh gán.

3) Tên của hàm chuẩn xác định kiểu dữ liệu của đầu ra của hàm. Thí dụ, nếu tên bắt đầu bằng một trong các chữ cái từ I đến N thì giá trị hàm là số nguyên.

4) Các đối số của hàm thường cùng kiểu như hàm, trừ một số ngoại lệ (xem phụ lục 1).

5) Các đối số của một hàm phải nằm trong cặp dấu ngoặc đơn.

6) Các đối số của một hàm có thể là các hằng, biến, biểu thức hay các hàm khác.

7) Các hàm tự sinh (generic function) chấp nhận nhiều kiểu đối số và trả lại giá trị hàm cùng kiểu với đối số. (Thí dụ hàm ABS(X) nếu đối số X là số nguyên thì giá trị hàm ABS(X) cho ra giá trị tuyệt đối là số nguyên, nếu X thực thì giá trị hàm sẽ là thực.)

Thí dụ 22: Đọc từ bàn phím một số nguyên. Kiểm tra xem nó là số chẵn hay số lẻ và in ra thông báo thích hợp. Ta có thể sử dụng hàm chuẩn MOD (I, J) trong bài tập này. Hàm MOD có hai đối số nguyên I và J. Hàm này trả về số dư của phép chia I/J. Vậy chương trình giải bài tập này có thể như sau:

```
PRINT *, ' NHAP MOT SO NGUYEN '
READ *, K
IF (MOD (K, 2) .EQ. 0) THEN
```

```
      PRINT 5, K
    ELSE
      PRINT 8, K
    END IF
5   FORMAT (1X, I5, '  LA SO CHAN')
8   FORMAT (1X, I5, '  LA SO LE')
```

8.2. Các hàm chương trình con

Trong thực tế lập trình giải các bài toán khoa học kỹ thuật nhiều khi đòi hỏi những hàm chưa có trong danh sách các hàm chuẩn của Fortran. Nếu tính toán hay lặp lại thường xuyên và đòi hỏi một số bước, ta nên thực hiện như là một hàm thay vì mỗi lần cần lại phải viết ra các lệnh tính toán. Fortran cho phép chúng ta tự xây dựng những hàm của riêng mình theo hai cách: *hàm lệnh* (statement function) và *hàm chương trình con* (function subprogram). Nếu tính toán có thể viết trong một lệnh gán duy nhất, thì ta sử dụng hàm lệnh; ngược lại, nếu phải thực hiện nhiều tính toán hay thao tác mới dẫn tới một giá trị kết quả, thì ta dùng hàm chương trình con.

8.2.1. Hàm lệnh

Dạng tổng quát của hàm lệnh là

Tên hàm (Danh sách đối số) = Biểu thức

Những quy tắc phải tuân thủ khi viết và dùng hàm lệnh:

1) Hàm lệnh được định nghĩa ở đầu chương trình, cùng với các lệnh khai báo kiểu dữ liệu.

2) Định nghĩa hàm lệnh gồm tên của hàm, sau đó đến các đối số nằm trong cặp dấu ngoặc đơn ở vế bên trái của dấu bằng; biểu thức tính giá trị hàm ở vế bên phải của dấu bằng.

3) Tên hàm có thể khai báo trong lệnh khai báo kiểu; nếu không thì kiểu của hàm sẽ được xác định theo cách định kiểu ẩn.

Thí dụ 23: Diện tích của tam giác có thể tính theo hai cạnh và góc xen giữa chúng:

$$Diện\ tích = 0{,}5 \times cạnh\ 1 \times cạnh\ 2 \times \sin(\text{góc}).$$

Viết chương trình đọc độ dài ba cạnh của một tam giác và các góc đối diện mỗi cạnh. Tính và in diện tích của tam giác theo ba phương án: trong mỗi phương án sử dụng một cặp cạnh và góc tương ứng.

Trong bài tập này ta phải tính diện tích tam giác ba lần, do đó có thể dùng hàm lệnh để tính diện tích tam giác. Chương trình có thể như sau:

```
      PROGRAM  DTTG
      REAL CA, CB, CC, A, B, C, DT, DT1, DT2, DT3,
     * C1, C2, GOC
      DT (C1, C2, GOC) = 0.5 * C1 * C2 * SIN (GOC)
      PRINT *, ' Nhap ba canh tam giac theo thu tu sau:'
      PRINT *, ' Canh A   Canh B    Canh C'
      READ *, CA, CB, CC
      PRINT *, ' Nhap ba goc (radian) theo thu tu sau:'
      PRINT *, ' Doi dien: canh A   canh B  canh C'
      READ *, A, B, C
      DT1 = DT (CB, CC, A)
      DT2 = DT (CC, CA, B)
      DT3 = DT (CA, CB, C)
      PRINT *
      PRINT *, 'Cac dien tich tinh theo ba phuong an la:'
      PRINT 5, DT1, DT2, DT3
```

```
5   FORMAT (1X, 3F7.2)
    END
```

Nhận xét rằng trong chương trình này hàm tính diện tích tam giác được định nghĩa ở đầu chương trình bởi tên DT và ba đối số hình thức C1, C2, GOC và giá trị của hàm được tính chỉ bằng một lệnh gán (dòng lệnh thứ ba). Trong chương trình, ở các dòng lệnh thứ 10–12 ta gọi hàm ba lần, mỗi lần ta chuyển các biến khác nhau vào vị trí của các đối số hình thức. Kiểu dữ liệu của hàm DT được mô tả tường minh tại lệnh mô tả REAL ở đầu chương trình. Trong chương trình chính các góc được cho bằng radian. Nếu các góc nhập vào được cho bằng độ và để không cần chuyển đổi thành radian trước khi gọi hàm DT tính các diện tích, ta có thể định nghĩa lại hàm DT như sau:

```
DT(C1, C2, GOC) = 0.5*C1*C2*SIN (GOC * 3.14159 / 180.0)
```

8.2.2. Hàm chương trình con

Thực chất của hàm chương trình con là một hàm do người lập trình tự xây dựng, do đó người ta còn gọi là hàm do người dùng định nghĩa. Hàm loại này khác với hàm lệnh ở chỗ nó được tính không phải bằng một lệnh gán duy nhất mà bằng một số lệnh. Hàm chương trình con bắt đầu với lệnh không thực hiện để đặc tả hàm bằng một tên và một danh sách đối số như sau

FUNCTION ***Tên hàm (danh sách đối số)***

Sau các lệnh mô tả và tính toán, lệnh RETURN chuyển điều khiển về chương trình chính và lệnh END báo cho chương trình dịch sự kết thúc của chương trình con. Tên hàm được chọn theo quy tắc như tên hằng, tên biến của Fortran. Tên hàm có ý nghĩa mô tả ẩn kiểu giá trị của hàm nếu trong chương trình chính chưa khai báo tường minh. Trong danh sách đối số nếu có từ hai đối số trở lên thì các đối số cách nhau bởi dấu phảy. Tên các đối số cũng có ý nghĩa mô tả ẩn kiểu dữ liệu của đối số. Tuy nhiên, nên mô tả tường minh các đối số của hàm trong phần khai báo các biến của hàm. Trong phần khai báo này, ngoài các đối số còn có thể khai báo các biến khác được dùng chỉ trong nội bộ hàm chương trình con. Vậy hình dáng tổng quát của một hàm chương trình con như sau:

FUNCTION ***Tên (đối số 1, đối số 2, ...)***
Các lệnh mô tả các đối số, các biến cục bộ
Các lệnh thực hiện
RETURN
END

Các hàm chương trình con được viết tách ra khỏi chương trình chính và nằm sau lệnh END của chương trình chính. Trong chương trình chính, khi cần tới hàm con người ta thường dùng lệnh gán để gán giá trị tính được bởi hàm con vào một biến hoặc dùng trực tiếp tên hàm con trong các biểu thức. Những giá trị của *các đối số thực tế* gửi vào các *đối số hình thức* phải phù hợp về kiểu và đúng tuần tự như trong danh sách đối số. Ta xét thí dụ về xây dựng hàm con và cách dùng nó trong chương trình chính qua thí dụ 24 dưới đây.

Thí dụ 24: Các mô hình số thường tính ra các giá trị của các thành phần kinh hướng V_k và vĩ hướng V_v của tốc độ gió ở những điểm khác nhau. Từ những cặp giá trị thành phần kinh hướng và vĩ hướng cần tính ra tốc độ V và hướng gió d. Tốc độ gió tính bằng công thức

$$V = \sqrt{V_k^2 + V_v^2},$$

còn hướng gió (góc giữa vectơ gió và hướng bắc) tính theo công thức

$$d = \begin{cases} \alpha & \text{nếu} \quad V_v \geq 0,\ V_k > 0 \\ 180 - \alpha & \text{nếu} \quad V_v \geq 0,\ V_k < 0 \\ 180 + \alpha & \text{nếu} \quad V_v < 0,\ V_k < 0 \\ 360 - \alpha & \text{nếu} \quad V_v < 0,\ V_k > 0 \end{cases}$$

trong đó $\alpha = \frac{180}{\pi} \text{arctg} \left| \frac{V_v}{V_k} \right|$.

Giả sử các giá trị của các thành phần kinh hướng và vĩ hướng của tốc độ gió đã lưu trong file GIOKV.KQ1 thành hai cột, dòng đầu tiên của file ghi số dòng dữ liệu có trong file. Viết chương trình đọc file GIOKV.KQ1 và ghi kết quả tính tốc độ và hướng gió vào file mới GIO.KQ2 thành 4 cột dạng sau:

TT	V_k	V_v	m/s	HUONG
XX	XX.X	XX.X	XXX	XXX
XX	XX.X	XX.X	XXX	XXX
. . .				

Khi lập chương trình giải quyết nhiệm vụ này ta nhận thấy cần tính mô đun của tốc độ gió và hướng gió nhiều lần. Vậy có thể sử dụng các hàm, ngoài ra, để tính tốc độ gió có thể dùng loại hàm lệnh, để tính hướng gió dùng hàm chương trình con. Chương trình có thể như sau:

```
      REAL GIOK (200), GIOV (200), V, H, TOCDO, HUONG
      INTEGER I, N
C     Mô tả hàm lệnh tính mô đun tốc độ gió
      TOCDO (VK, VV) = SQRT (VK*VK+VV*VV)
      OPEN (1, FILE = 'GIO.KQ1', STATUS = 'OLD')
      READ(1,*) N
      DO I = 1, N
      READ(1,*) GIOK (I), GIOV (I)
      END DO
      CLOSE (1)
      OPEN (1, FILE = 'GIO.KQ2', STATUS = 'UNKNOWN')
      WRITE (1, 4) 'TT', 'VK', 'VV', 'M/S', 'HUONG'
4     FORMAT(1X, I3, 4F7.1)
      DO  I = 1, N
      V = TOCDO (GIOK (I), GIOV (I))
      H = HUONG (GIOV (I), GIOK (I))
      WRITE (1, 5)  I, GIOK (I), GIOV (I), V, H
5     FORMAT (1X, I3, 4F7.1)
      END DO
      END
C     Hàm chương trình con
      FUNCTION HUONG (VV, VK)
      REAL VV, VK, HG
      IF (VK .EQ. 0.0) THEN
         IF (VV .GE. 0.0) THEN
            HG = 90.0
         ELSE
            HG = 270.0
         ENDIF
      ELSE
         G = ATAN (ABS (VV / VK)) / 3.14159 * 180.0
         IF (VK .GT. 0.0) THEN
            IF (VV .GE. 0.0) THEN
               HG = G
            ELSE
```

```
            HG = 360.0 - G
         ENDIF
      ELSE
         IF (VV .GE. 0.0) THEN
            HG = 180.0 - G
         ELSE
            HG = 180.0 + G
         ENDIF
      ENDIF
   ENDIF
   HUONG = HG
   RETURN
   END
```

Trong thí dụ này, ta thấy việc tính mô đun tốc độ và hướng được thực hiện nhiều lần. Do đó đã tổ chức tính chúng trong các hàm. Vì giá trị mô đun tính đơn giản bằng một biểu thức nên đã dùng loại hàm lệnh, đó là hàm TOCDO được định nghĩa ở dòng lệnh thứ ba của chương trình chính. Việc tính hướng phải thực hiện nhờ một số phép tính và thao tác, do đó đã dùng loại hàm chương trình con HUONG. Kiểu dữ liệu của hai hàm này được khai báo tường minh ở phần khai báo trong chương trình chính.

***Thí dụ 25**: Ước lượng nghiệm của đa thức bậc 4*

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4$$

trên khoảng [−5, 5].

Để giải bài toán này, ta sử dụng hai phương pháp tìm nghiệm của phương trình là phương pháp tìm hẹp dần và phương pháp chia đôi.

Trong *phương pháp tìm hẹp dần*, miền tìm nghiệm được chia thành những khoảng đủ nhỏ sao cho thực tế có thể xem rằng trong một khoảng nào đó chỉ có một nghiệm. Xét các khoảng từ trái sang phải, ta sẽ tìm những chỗ mà đồ thị của đa thức cắt trục x: bằng cách tính các giá trị của đa thức tại các đầu mút của khoảng, nếu dấu của các giá trị của đa thức tại các đầu mút khác nhau, thì đồ thị cắt trục x và ít nhất có một nghiệm trong khoảng đó. Sau đó khoảng chứa nghiệm lại được chia tiếp thành các phụ khoảng nhỏ hơn và quá trình tìm lại bắt đầu từ đầu trái cho đến khi xác định được khoảng chứa nghiệm. Quá trình chia khoảng và tìm lặp lại cho đến khi nghiệm được xác định đủ độ chính xác.

Phương pháp chia đôi bắt đầu với một khoảng đã được biết là chứa một nghiệm. Khác với phương pháp tìm hẹp dần chia khoảng chứa nghiệm thành nhiều phụ khoảng trước khi tìm, phương pháp chia đôi chỉ chia khoảng chứa nghiệm thành hai nửa, sau đó xác định nửa nào chứa nghiệm. Sự chia đôi tiếp tục cho đến khi tìm được nghiệm với độ chính xác mong muốn.

Trong thí dụ này, ta kết hợp hai phương pháp: phương pháp tìm hẹp dần để xác định khoảng chứa nghiệm. Sau đó phương pháp chia đôi xác định nghiệm với độ chính xác cần thiết. Giả sử phương pháp chia đôi tiếp tục lặp cho đến khi nửa khoảng nhỏ hơn 0,01 và nghiệm tìm được nếu giá trị tuyệt đối của đa thức không lớn hơn 0,001.

```
      INTEGER I, N
      REAL  A (0 : 4)
      REAL TRAI, PHAI, GIUA, KHOANG, NGHIEM
      LOGICAL  XONG
      PRINT * , ' NHAP CAC HE SO A0, A1, A2, A3, A4 '
      READ*, A
      PRINT 5, A
5     FORMAT (/, ' DA THUC:'
     *  / 1X, 9X, '4', 11X, '3', 11X, '2' / 1X, 4(F7.3, ' X + '), F7.3)
      N = 0
```

```
      DO I = 1, 40
         TRAI = –5.0 + REAL (I-1) * 0.25
         PHAI = TRAI + 0.25
         IF (ABS(F(A, TRAI)) .LT. 0.001) THEN
            PRINT 15, TRAI, F(A, TRAI)
15          FORMAT (1X, 'NGHIEM = ', F7.3, 3X,
     *      'F(NGHIEM) = ', F7.3)
            N = N + 1
         ELSE IF (F(A, TRAI)*F(A, PHAI) .LT. 0.0) THEN
            XONG = .FALSE.
            KHOANG = PHAI – TRAI
20          IF (KHOANG .GT. 0.01 .AND. .NOT. XONG) THEN
               GIUA = 0.5 *(TRAI + PHAI)
               IF (ABS (F(A, GIUA)) .LT. 0.001) THEN
                  XONG = .TRUE.
               ELSE IF (F(A, GIUA)*F(A, TRAI) .LT. 0.0)
               THEN
                  PHAI = GIUA
               ELSE
                  TRAI = GIUA
               END IF
               KHOANG = PHAI – TRAI
               GOTO 20
            END IF
            IF (KHOANG .GT. 0.01) THEN
               NGHIEM = GIUA
            ELSE
               NGHIEM = 0.5 *(TRAI + PHAI)
            END IF
            PRINT 15, NGHIEM, F(A, NGHIEM)
            N = N + 1
         END IF
      END DO
      TRAI = 5.0
      IF (ABS (F (A, TRAI)) .LT. 0.001) THEN
         PRINT 15, TRAI, F (A, TRAI)
         N = N + 1
      END IF
      IF (N .EQ. 0) THEN
         PRINT *, ' KHONG NGHIEM TRONG KHOANG [-5,5]'
      END IF
      END

      REAL FUNCTION F (A, X)
      REAL A(0 : 4), X
      F=A(0) + A(1)*X + A(2)*X**2 + A(3)*X**3 + A(4)*X**4
      RETURN
      END
```

Trong chương trình này, ta đã chia miền tìm nghiệm [–5, 5] thành 40 khoảng, mỗi khoảng dài 0,25 và thực hiện việc kiểm tra từ trái sang phải xem trong những khoảng nào có thể có nghiệm bằng phương pháp tìm hẹp dần bằng vòng DO. Trong mỗi khoảng, nếu giá trị đa thức ở đầu mút trái của khoảng không khác không quá 0,001 thì nhận nghiệm bằng đầu mút trái và chuyển sang xét khoảng tiếp sau ở bên phải. Còn nếu giá trị đa thức ở hai đầu mút của khoảng đang xét khác dấu, thì ta tìm nghiệm theo phương pháp chia đôi. Quá trình lặp để liên tiếp chia đôi khoảng thực hiện bằng vòng lặp IF lôgic và lệnh GOTO vô điều kiện cho đến khi khoảng trở nên nhỏ hơn hoặc bằng 0,01 hoặc giá trị tuyệt đối của đa thức ở giữa khoảng không lớn hơn 0,001. Việc tính giá trị đa thức thực hiện nhiều lần với những giá trị x khác nhau nên ta đã tổ chức hàm **F** để chuyên làm việc này.

Thí dụ 26: Viết chương trình đọc liên tiếp từ bàn phím ba số nguyên, kiểm tra xem chúng có tuần tự chỉ ngày tháng năm hợp lý không và in ra thông báo phù hợp. Chương trình kết thúc khi ta nhập ngày tháng năm đều

là những số không.

```
      PROGRAM KTNGAY
      INTEGER ID, IM, IY
10    PRINT *, 'HAY NHAP BA SO NGUYEN'
      READ *, ID, IM, IY
      IF (ID .NE. 0 .AND. IM .NE. 0 .AND. IY .NE. 0) THEN
         IF (OKDATE (ID, IM, IY)) THEN
            PRINT*, 'CO THE LA NGAY THANG NAM HOP LY'
         ELSE
            PRINT*, 'KHONG THE LA ',
     *              'NGAY THANG NAM HOP LY'
         ENDIF
         GOTO 10
      ENDIF
      END

      INTEGER FUNCTION SNTT (M, Y)
      INTEGER M,Y
      IF (M. EQ. 2) THEN
         SNTT = 28
         IF ((MOD (Y,100) .NE. 0 .AND. MOD (Y,4) .EQ. 0) .OR.
     *       (MOD (Y,100) .EQ. 0 .AND. MOD (Y/100, 4) .EQ .0))
     *       SNTT = 29
      ELSE IF (M.EQ.4 .OR. M.EQ.6 .OR. M.EQ.9 .OR. M.EQ.11)
      THEN
         SNTT = 30
      ELSE
         SNTT = 31
      ENDIF
      RETURN
      END

      LOGICAL FUNCTION OKDATE (D, M, Y)
      INTEGER D,M,Y,NNGAY
      IF (D.LT.1.OR.D.GT.31.OR.M.LT.1.OR.M.GT.12) THEN
         OKDATE = .FALSE.
      ELSE
         NNGAY = SNTT (M, Y)
         OKDATE = D.LE.NNGAY
      ENDIF
      RETURN
      END
```

Trong chương trình này dùng hai hàm con: hàm OKDATE và hàm SNTT. Hàm OKDATE có ba đối số nguyên D, M, Y và đưa ra giá trị lôgic là .TRUE. nếu D, M, Y là những số nguyên chỉ ngày tháng hợp lý. Hàm SNTT có hai đối số nguyên và đưa ra giá trị nguyên là số ngày của tháng đang xét. Nhận thấy rằng chương trình chính gọi hàm con OKDATE, về phần mình hàm con OKDATE trong khi thực hiện lại gọi hàm con SNTT.

8.3. Chỉ dẫn gỡ rối và phong cách viết chương trình có hàm con

Kiểm tra sự làm việc đúng đắn của hàm tự xây dựng cũng giống như kiểm tra chương trình chính. Nên thử cho hàm con những giá trị đối số khác nhau xem nó có đưa ra giá trị hàm đúng đắn không. Nếu hàm con làm việc không đúng đắn, hãy kiểm tra những điểm sau:

1) Sự phù hợp về kiểu và thứ tự của đối số thực tế và đối số hình thức.

2) Khẳng định rằng trước lệnh RETURN hàm đã nhận một giá trị đúng.

3) In kiểm tra giá trị các đối số trước và sau khi gọi hàm con.

4) Có thể dùng lệnh PRINT trong hàm con để định vị chỗ lỗi trong

hàm con.

Sử dụng hàm lệnh và hàm chương trình con sẽ làm chương trình có tính cấu trúc hơn và dễ đọc. Trong chương trình con cũng nên có cấu trúc sáng rõ. Nếu hàm dài và khó đọc, hãy dùng hàm con khác trong hàm con. Một khi bạn quyết định dùng hàm con, hãy cân nhắc những điều sau đây:

1) Chọn tên hàm con sao cho tên có tính gợi nhớ.

2) Nên dùng tên các đối số hình thức trong hàm con trùng với tên của các đối số thực tế. Nếu hàm được dùng nhiều lần với những đối số thực tế khác nhau, thì hãy chọn tên đối số hình thức hoàn toàn khác để tránh sự nhầm lẫn với các biến của chương trình chính. Có thể cách sau đây là một cách nên được dùng: tên các đối số trong hàm con đặt bằng các từ tiếng Anh, còn tên các đối số thực tế - bằng các từ tương ứng của tiếng Việt.

Bài tập

1. Viết chương trình in giá trị các biểu thức sau:

$$\alpha = \frac{6{,}9 + y}{y^2 + \sqrt{1 + 2y + 3y^2}}$$

$$\beta = \frac{\sin y}{y^4 + \sqrt{1 + 2y^2 + 3y^4}}$$

$$\gamma = \frac{2{,}3z + z^4}{z^2 + \sqrt{1 + 2z + 3z^2}}$$

$$\delta = \frac{1}{\sin^2 y + \sqrt{1 + 2\sin y + 3\sin^2 y}}$$

trong chương trình hãy xây dựng hàm con tên là DENOM với đối số x chuyên để tính biểu thức

$$x^2 + \sqrt{1 + 2x + 3x^2}\ .$$

2. Hãy cải tiến chương trình tìm nghiệm của đa thức ở thí dụ 25 trang 139 sao cho nó cho phép người dùng nhập khoảng xác định nghiệm thay vì dùng khoảng [−5, 5].

3. Hãy cải tiến chương trình tìm nghiệm của đa thức ở thí dụ 25 trang 139 sao cho nó cho phép người dùng nhập kích thước của phụ khoảng trong phần tìm hẹp dần.

4. Viết hàm chương trình con nhận giá trị một số nguyên và trả về giai thừa của số nguyên đó.

5. Côsin của một góc tính theo công thức chuỗi

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

trong đó x tính bằng radian. Hãy viết hàm chương trình con COSX với đầu vào là góc tính bằng độ, tính ra côsin của góc đó với độ chính xác ≤ 0,000001, sử dụng hàm con tính giai thừa ở bài tập 4. Sau đó viết chương trình chính in bảng ba cột: cột thứ nhất (x) chứa góc từ 0 đến 360° với gia số 15°; cột thứ hai chứa côsin của góc tính theo hàm chuẩn COS của Fortran và cột thứ ba chứa côsin tính theo hàm con COSX.

6. Viết hàm chương trình con MEDIAN (X,N) với đầu vào là mảng REAL X(N) đã sắp xếp tăng hoặc giảm dần và số giá trị thực tế của mảng N, trả về giá trị của trung vị của chuỗi $x(n)$ theo định nghĩa:

- nếu n lẻ trung vị bằng $x\left(\frac{n}{2} + 1\right)$,

- nếu n chẵn trung vị bằng $\frac{x\,(n/2) + x\,(n/2 + 1)}{2}$.

7. Viết hàm chương trình con DAD (D1, M1, Y1, D2, M2, Y2) với 6 đối số hình thức kiểu số nguyên: D1, M1, Y1, D2, M2, Y2 lần lượt chỉ

ngày, tháng, năm của hai ngày bất kỳ. Hàm này sẽ có giá trị lôgic bằng TRUE nếu ngày D2, M2, Y2 là ngày muộn hơn ngày D1, M1, Y1, còn nếu không thì nó sẽ có giá trị FALSE. Sau đó hãy viết một chương trình chính cho phép ta nhập từ bàn phím một ngày bất kỳ trong quá khứ hoặc trong tương lai, xác định và in lên màn hình thứ trong tuần của ngày đó. Biết rằng ngày 1-1-2001 là ngày thứ hai.

Chương 9 - Chương trình con loại thủ tục

Trong chương 8 chúng ta đã nghiên cứu về các hàm chuẩn, các hàm lệnh và các hàm do người lập trình tự xây dựng. Trong khi một hàm chỉ giới hạn ở việc tính ra một giá trị, thì các thủ tục chương trình con (hay còn gọi là thủ tục do người lập trình tự xây dựng) có thể tính ra một số giá trị, hoặc thực hiện một số thao tác. Trong bài này ta học cách viết các thủ tục và sử dụng các thủ tục trong các bài toán ứng dụng.

9.1. Khai báo và gọi chương trình con thủ tục

Nhiều quy tắc viết và sử dụng các thủ tục chương trình con giống như các quy tắc đối với các hàm chương trình con. Dưới đây liệt kê những khác biệt giữa các thủ tục và các hàm.

1) Một thủ tục không biểu diễn một giá trị, do đó tên của nó chỉ là đại diện cho một đoạn chương trình, không chỉ định kiểu của dữ liệu đầu ra.

2) Dòng lệnh đầu tiên trong một thủ tục thông báo tên thủ tục và danh sách đối số

SUBROUTINE ***Tên thủ tục (danh sách đối số)***

3) Chương trình chính gọi một thủ tục bằng lệnh CALL có dạng tổng quát như sau:

CALL ***Tên thủ tục*** **(*danh sách đối số*)**

4) Thủ tục dùng danh sách đối số không chỉ cho đầu vào mà cả cho những giá trị gửi ra chương trình chính gọi nó. Các đối số của thủ tục được dùng trong lệnh CALL là *những đối số thực tế*, còn các đối số sử dụng trong thủ tục là *những đối số hình thức*. Các đối số trong lệnh CALL *phải phù hợp về kiểu, số lượng và thứ tự* với những đối số trong thủ tục.

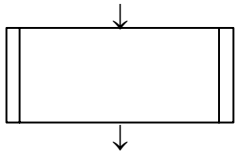
5) Một thủ tục có thể tính ra một giá trị, nhiều giá trị hoặc không giá trị nào cả. Một thủ tục có thể sử dụng một giá trị đầu vào, nhiều giá trị đầu vào hoặc không có giá trị đầu vào.

6) Nhãn lệnh, tên biến trong thủ tục được chọn độc lập với chương trình chính. Những biến dùng trong thủ tục mà không phải là các đối số của thủ tục gọi là *các biến cục bộ*, các giá trị của chúng không xử lý được trong chương trình chính.

7) Cần đặc biệt thận trọng khi sử dụng các mảng nhiều chiều trong các thủ tục. Nên chỉ định cả *kích thước khai báo* và *kích thước sử dụng thực tế* với các mảng hai hoặc nhiều chiều.

8) Giống như các hàm, lệnh RETURN ở cuối các thủ tục dùng để chuyển điều khiển trở về chương trình chính, lệnh END để báo kết thúc thủ tục.

9) Trong lưu đồ khối các thao tác được thực hiện bên trong thủ tục được ký hiệu bằng biểu tượng đồ họa sau đây:



10) Một thủ tục có thể dùng các hàm con khác hoặc gọi các thủ tục khác, nhưng nó không thể tự gọi chính nó. (Trong Fortran 90 cho phép dùng các thủ tục đệ quy có thể tự gọi chính mình.)

9.2. Những thí dụ ứng dụng chương trình con thủ tục

Những thí dụ dưới đây giúp chúng ta học cách viết các thủ tục và sử dụng nó trong chương trình chính như thế nào.

Thí dụ 27: Chương trình tính các đặc trưng thống kê: trung bình, phương sai và độ lệch chuẩn của chuỗi x gồm n số liệu quan trắc. Các công thức sau tính như sau:

$$m_x = \frac{\sum_{i=1}^{n} x_i}{n}, \qquad D_x = \frac{\sum_{i=1}^{n} x_i^2}{n-1} - m_x^2, \qquad \sigma_x = \sqrt{D_x}\ .$$

Ta thấy rằng mỗi đại lượng trên có thể tính được bằng một hàm riêng biệt. Nhưng ta cũng có thể tính luôn một lúc cả ba đại lượng bằng cách tổ chức tính chúng trong một thủ tục. Chương trình dưới đây cho phép đọc vào kích thước n và các giá trị của chuỗi x. Sau đó gọi thủ tục STAT để tính các đặc trưng thống kê. Cuối cùng là in ra kết quả.

Thấy rằng thủ tục STAT có tất cả 5 đối số hình thức, trong đó hai đối số đầu vào là mảng một chiều X và kích thước mảng N, ba đối số đầu ra là AVER, VARI và STDV. Khi gọi thủ tục này trong chương trình chính, ta gửi vào các đối số thực tế là X, N, TBINH, PSAI và DLC. Kết quả tính trung bình, phương sai và độ lệch chuẩn trong thủ tục chương trình con được lưu vào các biến TBINH, PSAI, DLC của chương trình chính. Hãy chú ý rằng: vì thủ tục chương trình con là môđun độc lập, nên tên các đối số của nó có thể trùng với tên của các biến trong chương trình chính, trong thí dụ này là đối số X và N. Ở đây ta thấy, trong chương trình con, có thể định nghĩa kích thước của mảng bằng biến N (trong lệnh REAL X(N)). Nhớ rằng điều này chỉ cho phép trong chương trình con.

```
      PROGRAM THKE
      INTEGER N, I
      REAL X(99), TBINH, PSAI, DLC
      PRINT *, ' NHAP DO DAI CHUOI (<100)'
      READ *, N
      PRINT *, ' NHAP CAC GIA TRI CUA X:'
5     FORMAT (1X, ' X(', I2, '): ')
      DO I = 1, N
         WRITE (*, 5)  I
         READ *, X (I)
      ENDDO
      CALL  STAT (X, N, TBINH, PSAI, DLC)
      WRITE(*, 8) TBINH, PSAI, DLC
8     FORMAT (1X, ' T. BINH = ', F7.2, ' PH. SAI = ',
     *F7.2, ' DL CHUAN = ', F7.2)
      END
      SUBROUTINE STAT (X, N, AVER, VARI, STDV)
      REAL X (N), AVER, VARI, STDV
      INTEGER N, I
      AVER = 0.0
      VARI = 0.0
      DO I = 1, N
         AVER = AVER + X (I)
         VARI = VARI + X (I) * X (I)
      END DO
      AVER = AVER / REAL (N)
      VARI = VARI / REAL (N–1) – AVER * AVER
      STDV = SQRT (VARI)
      RETURN
      END
```

***Thí dụ 28**: **Xử lý ngày tháng**.* Trong thực tế xử lý số liệu quan trắc khí tượng thủy văn, ta thường hay phải để ý đến ngày tháng của số liệu. Trong mục này sẽ xét một thí dụ về xử lý ngày tháng với thí dụ sau: Viết chương trình nhập vào một ngày tháng năm bất kỳ, in ra ngày tháng năm của ngày hôm sau. Việc xác định ngày tháng năm của ngày hôm sau so với ngày tháng năm hiện tại sẽ được thực hiện trong một chương trình con thủ tục, ta gọi tên là thủ tục HOMSAU. Vì ở đây kết quả sẽ cho ra ba giá trị nguyên tuần tự chỉ ngày, tháng và năm. Chương trình có thể như sau:

```
      PROGRAM TGNGAY
      INTEGER ID, IM, IY
      PRINT *, 'HAY NHAP NGAY THANG NAM BAT KY'
      READ *, ID, IM, IY
      CALL HOMSAU (ID, IM, IY)
      PRINT 20, ID, IM, IY
20    FORMAT (1X, 'NGAY HOM SAU LA ', I2, '–', I2 , '–', I4)
      END
      SUBROUTINE  HOMSAU (D, M, Y)
      INTEGER D, M, Y
      D = D + 1
      IF (D .GT. SNTT (M, Y)) THEN
         D = 1
         M = M + 1
         IF (M .GT. 12) THEN
            M = 1
```

```
        Y = Y + 1
      END IF
    END IF
    RETURN
    END
    INTEGER FUNCTION SNTT (M, Y)
    INTEGER M, Y
    IF (M .EQ. 2) THEN
      SNTT = 28
      IF ((MOD(Y,100) .NE. 0 .AND. MOD(Y,4) .EQ. 0) .OR.
     *    (MOD (Y,100) .EQ. 0 .AND. MOD (Y/100, 4) .EQ .0))
     *    SNTT = 29
    ELSE IF (M.EQ.4.OR.M.EQ.6.OR.M.EQ.9.OR.M.EQ.11) THEN
      SNTT = 30
    ELSE
      SNTT = 31
    ENDIF
    RETURN
    END
```

Các thao tác để chuyển thành hôm sau được thực hiện trong chương trình con thủ tục HOMSAU. Hãy chú ý rằng trong thủ tục con HOMSAU lại gọi thực hiện một hàm con khác là SNTT để tính số ngày của tháng đang xét. Hàm này đã được nhắc tới trong thí dụ 26, trang 142, ở đây ghi lại để sinh viên tiện theo dõi.

Dưới đây ta xét một thí dụ về xử lý số liệu khí tượng có liên quan tới ngày tháng.

Thí dụ 29: Tính các giá trị trung bình tháng của một yếu tố khí tượng thủy văn. Giả sử với file dữ liệu về các yếu tố khí tượng thủy văn đã mô tả trong thí dụ 20 (trang 124).

Nhớ lại rằng file HONDAU.MAT có quy cách ghi như sau: Dòng trên cùng ghi tên trạm. Dòng thứ 2 có hai số nguyên viết cách nhau lần lượt chỉ tổng số ngày quan trắc và số yếu tố được quan trắc. Dòng thứ ba có 6 số nguyên viết cách nhau lần lượt chỉ ngày, tháng, năm đầu và ngày, tháng, năm cuối quan trắc. Dòng thứ 4 là tiêu đề cột liệt kê tên tất cả các yếu tố được quan trắc, mỗi tên được ghi với độ rộng 8 vị trí. Các dòng tiếp theo lần lượt ghi giá trị của các yếu tố, mỗi dòng một ngày. Các giá trị ngày của nhiệt độ không khí được ghi ở cột thứ hai của file này. Viết chương trình tính giá trị trung bình tháng của nhiệt độ không khí trong tất cả các năm quan trắc.

Ở đây ta thấy, muốn tính trung bình tháng chỉ việc cộng tất cả các giá trị ngày và chia tổng cho số ngày của tháng đang xét. Nhưng vì số ngày trong mỗi tháng có thể khác nhau, nên ta cần có những thủ tục xử lý ngày tháng. Chương trình sau đây sẽ thực hiện kiểu xử lý như vậy.

```
      REAL X, TONG, TB (100, 12)
      INTEGER D1, M1, Y1, D2, M2, Y2, Y, M, I
      OPEN (1, FILE = 'HONDAU.MAT', STATUS = 'OLD')
      READ (1, *)
      READ (1, *) N
      READ (1, *) D1, M1, Y1, D2, M2, Y2
      READ (1, *)
      Y = Y1
      I = 1
2     IF (Y1 .GT. Y2) GOTO 4
      READ (1, *) X, X
      IF (D1 .EQ. 1) THEN
        TONG = 0.0
        M = SNTT (M1, Y1)
```

```
      END IF
      TONG = TONG + X
      IF (D1 .EQ. M) THEN
         TB (I, M1) = TONG / M
         IF (M1.EQ.12)  I = I + 1
      END IF
      CALL HOMSAU (D1, M1, Y1)
      GOTO 2
4     CLOSE (1)
      PRINT *, ' NHIET DO KHONG KHI TRUNG BINH THANG'
      PRINT '(A5, 12I5)', 'NAM', (J, J = 1, 12)
      K = 1
      DO I = Y, Y2
         PRINT '(A5, 12F5.1)', I, (TB (K, J), J =1, 12)
      K = K + 1
      END DO
      END
```

Ta thấy trong chương trình này đã sử dụng hàm SNTT và thủ tục HOMSAU mà chúng ta đã xây dựng trong thí dụ 28. Hàm SNTT được gọi vào mỗi ngày đầu tháng để tính số ngày của tháng đó và gán vào biến M chuẩn bị cho việc tính trung bình sau khi giá trị nhiệt độ ngày cuối cùng của tháng được cộng vào biến TONG. Còn thủ tục HOMSAU được gọi liên tục để tăng ngày hiện hành lên một ngày sau khi một số liệu được đọc.

Qua thí dụ 20 và thí dụ này, sinh viên cũng cần chú ý ghi nhớ cách đọc số liệu kiểu bỏ qua một số cột trong file chứa bảng số liệu có nhiều cột.

Dưới đây là hai thí dụ liên quan tới các thủ tục thao tác chuỗi thường có thể rất có ích trong thực tiễn xử lý thống kê chuỗi số liệu khí tượng thủy văn.

Thí dụ 30: Chèn một giá trị vào danh sách. Trong thí dụ này, ta viết một thủ tục cho phép chèn một giá trị mới vào một danh sách đã sắp xếp. Các đối số của thủ tục gồm mảng một chiều X, biến COUNT chỉ số giá trị dữ liệu thực tế trong mảng, biến LIMIT chứa kích thước mô tả của mảng và biến NEW chứa giá trị cần chèn vào mảng. Trường hợp phần tử mới được chèn vào mảng đã đầy, tức giá trị COUNT bằng giá trị LIMIT thì giá trị cuối cùng trong mảng sẽ bị cắt bỏ.

```
      SUBROUTINE INSERT (LIMIT, NEW, COUNT, X)
      INTEGER LIMIT, NEW, COUNT, X (LIMIT), J, K
      LOGICAL DONE
      DONE = .FALSE.
      J = 1
5     IF (J .LE. COUNT .AND. .NOT. DONE) THEN
         IF (X (J) .LT. NEW) THEN
            J = J + 1
         ELSE
            DONE = .TRUE.
         END IF
         GOTO 5
      END IF
      IF (J .GT. COUNT) THEN
         IF (COUNT .LT. LIMIT) THEN
            COUNT = COUNT + 1
            X (COUNT) = NEW
         END IF
      ELSE
         IF (COUNT .LT. LIMIT) COUNT = COUNT + 1
         DO K = COUNT, J+1, –1
            X (K) = X (K – 1)
```

```
        END DO
        X (J) = NEW
    END IF
    RETURN
    END
```

Thí dụ 31: Xóa một giá trị khỏi danh sách. Trong trường hợp này, ta tìm trong danh sách giá trị bằng giá trị định xóa và loại giá trị đó khỏi danh sách. Khác với thủ tục chèn, trong thủ tục xóa không cần đối số LIMIT vì khi xóa một giá trị khỏi danh sách thì số phần tử thực của mảng chỉ có thể nhỏ đi, không sợ chỉ số mảng vượt quá kích thước mô tả của mảng.

```
    SUBROUTINE DELETE (OLD, COUNT, X)
    INTEGER OLD, COUNT, X (COUNT), J, K
    LOGICAL DONE
    DONE = .FALSE.
    J = 1
5   IF (J .LT. COUNT .AND. .NOT. DONE) THEN
        IF (X (J) .LT. OLD) THEN
            J = J + 1
        ELSE
            DONE = .TRUE.
        END IF
        GOTO 5
    END IF
    IF (J .GT. COUNT .OR. X (J) .GT. OLD) THEN
        PRINT *, 'GIA TRI XOA KHONG CO TRONG DANH
        SACH'
    ELSE
        COUNT = COUNT - 1
        DO K = J, COUNT
            X (K) = X (K + 1)
        END DO
    END IF
    RETURN
    END
```

9.3. Những chỉ dẫn gỡ rối khi sử dụng các thủ tục

1) Kiểm tra xem các biến, các biểu thức trong danh sách đối số ở lệnh CALL có phù hợp về kiểu và thứ tự như trong lệnh khai báo thủ tục không.

2) Nên khai báo tường minh tất cả các biến trong thủ tục để tránh định kiểu ngầm định sai.

3) Sử dụng lệnh PRINT trong thủ tục để định vị lỗi.

4) Kiểm tra thử từng thủ tục trước khi gộp chúng vào chương trình chính cùng với những chương trình con khác.

5) Kiểm tra từng thủ tục với một số tập dữ liệu để phát hiện những điều kiện đặc biệt gây lỗi.

6) Phong cách lập trình:

- Khi quyết định sử dụng chương trình con thủ tục hãy chọn tên thủ tục sao cho nó có tính gợi nhớ.

- Hãy sử dụng tên các biến trong danh sách đối số của thủ tục cùng tên với các biến là đối số thực tế trong chương trình chính. Trong trường hợp thủ tục được gọi nhiều lần với những đối số thực tế khác nhau, hãy chọn tên trong danh sách đối số khác biệt để tránh nhầm với các biến trong chương trình chính.

- Trong danh sách đối số, nên liệt kê riêng các đối số đầu vào trước, sau đó mới đến các đối số đầu ra.

Bài tập

1. Giả sử có mảng một chiều X với 100 giá trị thực. Hãy viết một thủ tục tạo ra mảng Y theo cách mỗi phần tử của mảng Y bằng phần tử tương ứng của mảng X trừ đi phần tử nhỏ nhất.

2. Viết một thủ tục nhận một mảng giá trị thực X với 50 hàng và 2 cột và trả lại chính mảng đó nhưng dữ liệu được sắp xếp lại theo chiều tăng dần của cột thứ 2.

3. Viết một thủ tục nhận một mảng giá trị thực X với n dòng m cột và trả về một mảng Y cùng số dòng, số cột nhưng dữ liệu được biến đổi sao cho các phần tử tương ứng của cột thứ nhất và cột thứ J được đổi chỗ cho nhau.

4. Giả sử cho trước hai ma trận A (n dòng, m cột) và ma trận B (m dòng, l cột). Tích AB sẽ là ma trận C (n dòng, l cột) với các phần tử được tính theo công thức

$$c_{i,j} = \sum_{k=1}^{m} a_{i,k}\, b_{k,j} \quad (i = 1, ..., n; j = 1, ..., l) .$$

Viết thủ tục TICHMT (A, B, N, M, L, C) với các đối số đầu vào là ma trận A, ma trận B, các tham số N, M, L và đối số đầu ra là ma trận C.

5. Hệ phương trình đại số tuyến tính n ẩn

$$\left.\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & ... & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & ... & + & a_{2n}x_n & = & b_2 \\ ... & & ... & & ... & & ... & & ... \\ a_{n1}x_1 & + & a_{n2}x_2 & + & ... & + & a_{nn}x_n & = & b_n \end{array}\right\}$$

được viết dưới dạng ma trận như sau

$$\boldsymbol{A\,x = b}$$

trong đó

$$A = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix}; \quad b = \begin{pmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{pmatrix}; \quad x = \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{pmatrix}.$$

Hãy viết thủ tục GAUSS (A, B, N, X) nhận vào các mảng A, B, số ẩn N của hệ và tính ra mảng X theo phương pháp loại biến của Gauss. Xem giải thích về phương pháp Gauss trong phụ lục 2.

Chương 10 - Kiểu dữ liệu văn bản

Ngoài những dữ liệu số như các số nguyên, số thực, máy tính còn có thể lưu giữ và xử lý những dữ liệu văn bản như những chữ cái, những đoạn văn bản, những chữ số và một số ký hiệu khác. Trong Fortran gọi chung những dữ liệu này là dữ liệu ký tự. Trong chương này chúng ta xét thêm những đặc điểm khai báo những dữ liệu ký tự, một số thao tác với những dữ liệu ký tự và ứng dụng của chúng trong xử lý thông tin.

10.1. Tập các ký tự của Fortran

Tập ký tự của Fortran gồm 26 chữ cái tiếng Anh, mười chữ số từ 0 đến 9, dấu trống và 12 ký hiệu sau đây:

+ – * / = ( ) , . ' $:

Ngoài ra còn một số ký tự khác tùy thuộc vào những hệ máy tính khác nhau.

Các hằng ký tự bao giờ cũng nằm trong cặp dấu nháy trên. Trong hằng ký tự dấu nháy trên ' được biểu thị bằng hai dấu nháy trên ‘’ (không phải dấu ngoặc kép). Thí dụ chữ LET'S của tiếng Anh sẽ được viết là 'LET''S'.

Thông thường người ta xử lý trong máy tính những từ, những dòng chữ gồm một số ký tự ghép lại với nhau. Trong trường hợp đó người ta gọi là *xâu ký tự*. *Độ dài của xâu ký tự* là số ký tự được ghép lại trong xâu ký tự đó. Một ký tự cũng có thể coi là một xâu ký tự với độ dài bằng 1. Do đó, ta gọi chung dữ liệu xâu ký tự là dữ liệu ký tự hay dữ liệu văn bản. Dưới đây là thí dụ về các hằng ký tự và độ dài tương ứng của chúng:

'CHU NHAT'	8 ký tự
'SENSOR 23'	9 ký tự
'08:40–13:25'	11 ký tự
'LE QUY DON'	10 ký tự
' '	2 ký tự
''''	2 ký tự

10.2. Các dạng khai báo biến ký tự

♦ Biến ký tự được khai báo bằng lệnh mô tả dạng tổng quát như sau:

CHARACTER * n Danh sách biến

trong đó *n* chỉ số ký tự (độ dài) trong mỗi xâu ký tự. Thí dụ lệnh

CHARACTER * 8 TEN, NGAY

chỉ rằng TEN và NGAY là những biến chứa 8 ký tự mỗi biến.

♦ Lệnh **CHARACTER** biến thể sau đây cho phép ta khai báo những biến ký tự với độ dài khác nhau trên cùng một dòng lệnh

CHARACTER TITLE * 10, NUOC * 2

♦ Một mảng chứa một số phần tử, mỗi phần tử có giá trị là một xâu ký tự được khai báo bằng một trong hai cách tương đương như sau:

CHARACTER * 4 NAME (50)

CHARACTER NAME (50) * 4

♦ Các xâu ký tự cũng có thể được dùng trong các chương trình con.

Xâu ký tự phải được khai báo bằng lệnh CHARACTER trong cả chương trình chính và chương trình con. Cũng như các mảng dữ liệu số nguyên, số thực, trong chương trình con có thể khai báo biến ký tự mà không cần chỉ định rõ độ dài xâu và kích thước mảng. Thí dụ:

```
CHARACTER * (*)  BCC
CHARACTER * (*) NAME (N)
```

10.3. Nhập, xuất dữ liệu ký tự

Khi xâu ký tự được dùng trong lệnh xuất toàn bộ xâu được in ra. Những dấu trống được tự động chèn vào xâu để tách riêng xâu ký tự với những mục in khác cùng dòng. Trong lệnh nhập, giá trị của biến ký tự phải được bao trong cặp dấu nháy trên. Nếu số ký tự trong cặp dấu nháy nhiều hơn so với độ dài đã mô tả của biến ký tự, thì những ký tự thừa ở bên phải sẽ bị bỏ qua (bị cắt bỏ); nếu số ký tự ít hơn – những vị trí thừa ở bên phải được tự động điền bằng các dấu trống. Để in xâu ký tự trong lệnh xuất có định dạng, có thể dùng đặc tả A (*chú ý*, có thể không cần chỉ rõ số vị trí dành cho mục in). Thí dụ: với đoạn chương trình:

```
      CHARACTER * 20 THU
      PRINT *, ' HAY NHAP MOT NGAY TRONG TUAN'
      READ *, THU
      PRINT 5, THU
5     FORMAT (1X, 'NGAY VUA NHAP LA ', A)
      END
```

thì tương tác trên màn hình sẽ như sau:

```
HAY NHAP MOT NGAY TRONG TUAN
'CHU NHAT' ↵
NGAY VUA NHAP LA CHU NHAT
```

Thấy rằng số ký tự gõ vào biến THU chỉ bằng 8, không dài tới 20 như đã khai báo. Nhưng khi in ra màn hình ta không thấy rõ những dấu trống được tự động điền vào phía bên phải. Nếu lệnh FORMAT của lệnh in có dạng

```
FORMAT (1X, A, ' LA NGAY VUA NHAP')
```

thì trên màn hình sẽ thấy rõ những dấu trống như sau:

```
HAY NHAP MOT NGAY TRONG TUAN
'CHU NHAT'
CHU NHAT          LA NGAY VUA NHAP
```

10.4. Những thao tác với dữ liệu ký tự

10.4.1. Gán các giá trị ký tự

Những giá trị ký tự có thể được gán cho các biến ký tự bằng lệnh gán và một hằng ký tự. Nếu hằng có độ dài nhỏ hơn số ký tự đã khai báo của biến, thì các dấu trống sẽ tự động được điền vào bên phải; nếu hằng có độ dài lớn hơn - các ký tự thừa sẽ bị bỏ qua. Thí dụ:

```
CHARACTER * 4  MONHOC (3)
```

```
MONHOC (1) = 'TOAN'
MONHOC (2) = 'LY'
MONHOC (3) = 'HOA HOC'
```

Trong những lệnh trên đây ta khai báo mảng MONHOC gồm 3 phần tử, mỗi phần tử là một xâu dài 4 ký tự. Vậy trong MONHOC (1) sẽ lưu 'TOAN', trong MONHOC (2) sẽ lưu 'LY**bb**', trong MONHOC (3) sẽ lưu 'HOA**b**' (chữ **b** chỉ dấu trống). Qua thí dụ này ta thấy tầm quan trọng của việc sử dụng các xâu có cùng độ dài mô tả của biến; nếu không các lệnh sẽ xử lý sai.

Một biến ký tự cũng có thể được gán giá trị của biến ký tự khác bằng lệnh gán, thí dụ

```
CHARACTER * 4 LOAI1, LOAI2
LOAI1 = 'GIOI'
LOAI2 = LOAI1
```

Sau lệnh gán này, cả hai biến LOAI1 và LOAI2 đều lưu xâu ký tự 'GIOI'.

Lệnh DATA cũng có thể dùng để khởi xướng giá trị của các biến ký tự. Thí dụ sau gán 12 tên tháng tiếng Anh vào mảng THANG:

```
CHARACTER * 3 THANG (12)
DATA THANG / 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
*  'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec' /
```

10.4.2. So sánh các giá trị ký tự

Biểu thức lôgic trong lệnh IF lôgic cũng có thể là một phép so sánh các biến, hằng ký tự. Thí dụ, nếu các biến THANG, CH, TG là những biến ký tự, các lệnh sau đây là những lệnh đúng:

```
IF (THANG .EQ. 'FEB') NGAY = 28
IF (CH (I) .GT. CH (I+1)) THEN
   TG = CH (I)
   CH (I) = CH (I+1)
   CH (I+1) = TG
END IF
```

Khi đánh giá một biểu thức lôgic với các xâu ký tự, trước hết chương trình xét độ dài của hai xâu. Nếu một xâu ngắn hơn xâu khác, thì xâu ngắn hơn được bổ sung thêm các dấu trống ở bên phải sao cho hai xâu trở thành có cùng độ dài. Việc so sánh hai xâu ký tự cùng độ dài thực hiện từ trái sang phải theo từng ký tự một. Hai xâu bằng nhau nếu chúng có cùng những ký tự trong cùng một thứ tự. Các ký tự được so sánh với nhau theo *chuỗi thứ tự so sánh* (collating sequence). Chuỗi này liệt kê các ký tự từ thấp đến cao. Thí dụ, một phần của chuỗi thứ tự so sánh đối với các ký tự ASCII liệt kê các ký tự dưới đây:

Chuỗi thứ tự so sánh của các ký tự:

```
b"#$%&()*+,-./
0123456789
:;=?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Theo chuỗi này, những so sánh sau là đúng:

```
'A1' < 'A2'
'JOHN' < 'JOHNSTON'
'175' < '176'
'THREE' < 'TWO'
'$' < 'DOLLAR'
```

Nếu các xâu ký tự *chỉ chứa các chữ cái*, thì thứ tự từ thấp đến cao là thứ tự alphabê, được gọi là *thứ tự từ vựng* (lexicographic ordering).

10.4.3. Trích ra xâu con

Xâu con là một phần được trích ra từ xâu xuất phát và giữ nguyên thứ tự ban đầu. Trong Fortran xâu con được viết bằng *tên của xâu xuất phát, kèm theo hai biểu thức nguyên nằm trong cặp dấu ngoặc đơn, cách nhau bởi dấu hai chấm. Biểu thức thứ nhất chỉ vị trí đầu tiên ở xâu xuất phát mà từ đó xâu con được trích ra. Biểu thức thứ hai chỉ vị trí cuối cùng*. Thí dụ, nếu xâu 'FORTRAN' được lưu trong biến LANG, ta có thể có những xâu con như sau

Biến	*Xâu con*
LANG (1 : 1)	'F'
LANG (1 : 7)	'FORTRAN'
LANG (2 : 3)	'OR'
LANG (7 : 7)	'N'

1) Ta có thể không viết biểu thức thứ nhất trong cặp dấu ngoặc đơn nếu giá trị của nó bằng 1 và có thể không viết biểu thức thứ hai nếu giá trị của nó bằng độ dài của xâu xuất phát. Ta cũng có thể không viết cả hai biểu thức. Nhưng trong cả ba trường hợp vẫn phải có dấu hai chấm (:) ở trong cặp dấu ngoặc. Thí dụ:

LANG (:4)	là 'FORT'
LANG (5:)	là 'RAN'
LANG (:)	là 'FORTRAN'

2) Khi phép trích ra xâu con sử dụng cùng một tên biến, các biểu thức trong cặp dấu ngoặc đơn không được phủ lên nhau. Thí dụ, nếu biến LANG chứa xâu 'FORMATS', thì lệnh

LANG (7: 7) = LANG (6: 6)

sẽ biến giá trị của LANG thành 'FORMATT'. Nhưng lệnh sau đây sẽ sai không thể thực hiện được

LANG (3: 5) = LANG (2: 4)

3) Những trường hợp như: các vị trí đầu hoặc cuối không phải là số nguyên, là số âm, vị trí đầu lớn hơn vị trí cuối, vị trí đầu hoặc vị trí cuối có giá trị lớn hơn độ dài mô tả của xâu con, việc trích ra xâu con sẽ không thể thực hiện đúng đắn.

Thí dụ 32: Đếm số ký tự trong một văn bản. Giả sử một bức điện dài 50 ký tự. Hãy đếm số từ trong bức điện đó. Ta biết rằng trong một văn bản soạn đúng thì các từ cách nhau bằng một dấu trống, do đó ta chỉ cần đếm số dấu trống trong văn bản và số từ sẽ bằng số dấu trống cộng thêm một. Với trường hợp này chương trình sau sẽ đếm được đúng số từ:

```
      CHARACTER * 50 MESSGE
      INTEGER COUNT, I
      COUNT = 0
      DO 10 I = 1, 50
      IF (MESSGE (I: I) .EQ. ' ') COUNT = COUNT + 1
10    CONTINUE
      PRINT 5, COUNT + 1
5     FORMAT (1X, 'BUC DIEN GOM ', I2, ' TU')
      END
```

10.4.4. Kết hợp các xâu ký tự

Kết hợp hay *cộng* là thao tác ghép hai hoặc một số xâu ký tự vào thành một xâu ký tự. Thao tác này thực hiện bởi hai dấu gạch chéo //. Thí dụ muốn có từ WORKED ta có thể dùng phép kết hợp

'WORK' // 'ED'

Nhóm lệnh sau đây cho phép viết ra ngày tháng theo quy cách tiếng Việt, tức thêm các gạch chéo ngăn cách giữa các ký hiệu ngày, tháng và năm:

```
CHARACTER DAY*2,MONTH*2,YEAR*4,DATE*10
READ *, DAY, MONTH, YEAR
DATE = DAY//'/'//MONTH//'/'//YEAR
PRINT *, DATE
END
```

Theo nhóm lệnh này, nếu khi thực hiện lệnh READ ta gõ từ bàn phím '05' '10' '1999' ↵ thì trên màn hình sẽ in ra:

05/10/1999.

10.4.5. Những hàm chuẩn xử lý xâu ký tự

- **Hàm INDEX**

Hàm này có hai đối số kiểu xâu ký tự, đưa ra một số nguyên chỉ vị trí của xâu thứ hai trong xâu thứ nhất. Thí dụ nếu ta có biến STR chứa mệnh đề 'TO BE OR NOT TO BE' và dùng lệnh

K = INDEX (STR, 'BE')

thì biến K sẽ có giá trị 4 vì xâu 'BE' xuất hiện lần đầu tiên trong xâu STR ở vị trí thứ 4.

- **Hàm LEN**

Hàm LEN có một đối số kiểu xâu ký tự, nó đưa ra một số nguyên chỉ độ dài của xâu đó. Hàm này rất có ích trong những chương trình con chấp nhận các xâu ký tự độ dài bất kỳ nhưng cần biết độ dài thực tế ở trong chương trình con.

Thí dụ 33: Cấu tạo tên viết tắt của người. Viết chương trình đọc từ bàn phím họ tên đầy đủ (gồm họ, chữ đệm và tên) của một người và in lên màn hình dạng viết tắt. (Thí dụ, nếu nhập vào họ tên đầy đủ như sau:

TRAN CONG MINH,

thì dạng in ra sẽ là

T. C. MINH.

Chương trình NAMEED dưới đây cho phép ta gõ từ bàn phím một xâu ký tự gồm cả họ, chữ đệm và tên trên cùng một dòng nhưng cách nhau bởi một dấu trống. Thủ tục con EXTR cho phép tách riêng phần họ, chữ đệm và tên dựa vào vị trí các dấu trống trong họ tên đầy đủ. Sau đó thủ tục EDIT ghép các chữ cái đầu tiên của phần họ, chữ đệm kèm theo các dấu chấm và dấu trống với tên để cấu tạo thành tên viết tắt.

```
      PROGRAM NAMEED
      CHARACTER HO *10, DEM *10, TEN *20, HOTEN *25
      PRINT *, 'Nhap ho, chu dem, ten cach nhau 1 dau trong'
      READ 5, HOTEN
5     FORMAT (A)
      CALL EXTR (HOTEN, HO, DEM, TEN)
      CALL EDIT (HO, DEM, TEN, HOTEN)
      PRINT *, HOTEN
      END

      SUBROUTINE EXTR (XHOTEN, XHO, XDEM, XTEN)
      CHARACTER * (*) XHO, XTEN, XDEM, XHOTEN
      INTEGER B1, B2
      B1 = INDEX (XHOTEN, ' ')
      B2 = B1 + INDEX (XHOTEN (B1 + 1:) , ' ')
```

```
XHO = XHOTEN (:B1-1)
XDEM = XHOTEN (B1+1: B2-1)
XTEN = XHOTEN (B2+1:)
RETURN
END

SUBROUTINE EDIT (XHO, XDEM, XTEN, XHOTEN)
INTEGER L
CHARACTER *(*) XHO, XTEN, XDEM, XHOTEN
XHOTEN = XHO(1: 1) // '.'
L = INDEX (XHOTEN, ' ') + 1
XHOTEN (L: L + 2) = XDEM (1: 1) // '. '
XHOTEN (L + 3:) = XTEN
RETURN
END
```

• **Các hàm CHAR** và **ICHAR**

Các hàm này thao tác với các ký tự trong *chuỗi thứ tự so sánh* dùng trong máy tính. Nếu một máy tính có 256 ký tự trong chuỗi thứ tự so sánh của nó, thì các ký tự này được đánh số từ 0 đến 255. ***Hàm* CHAR** *nhận một đối số nguyên và đưa ra một ký tự trong chuỗi thứ tự so sánh ở vị trí ứng với số nguyên đó*. ***Hàm* ICHAR** *là hàm ngược của hàm* **CHAR**. Nó nhận đối số là biến một ký tự và trả về một số nguyên ứng với vị trí của ký tự đó ở trong chuỗi thứ tự so sánh.

Vì các máy tính khác nhau có các chuỗi thứ tự so sánh khác nhau, nên các hàm này có thể dùng để xác định vị trí của những ký tự trong chuỗi thứ tự so sánh.

Thí dụ, nếu bạn muốn in ra màn hình tất cả các ký tự trong chuỗi thứ tự so sánh được dùng trong máy tính của mình từ vị trí 0 đến 255 có thể dùng chương trình sau:

```
PROGRAM CSCHAR
DO I = 0, 255
    PRINT *, I, ' ', CHAR (I)
END DO
END
```

Chương trình sau đây cho phép in ra màn hình vị trí của các chữ cái in hoa tiếng Anh, những chữ cái thường và những chữ số từ 0 đến 9 trong chuỗi thứ tự so sánh trong máy tính bạn đang dùng:

```
PROGRAM COLSEQ
CHARACTER *70 SET
SET (1: 26) = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
SET (27: 52) = 'abcdefghijklmnopqrstuvwxyz'
SET (53: 62) = '0123456789'
DO I = 1, 62
   PRINT *, SET (I : I), ICHAR (SET (I : I))
END DO
END
```

Với các máy tính thông dụng ngày nay, nếu chạy chương trình này, ta sẽ thấy tập các chữ số từ '**0**' đến '**9**' tuần tự có vị trí từ **48** đến **57**, tập các chữ cái hoa tiếng Anh từ '**A**' đến '**Z**' có vị trí từ **65** đến **90** và tập các chữ cái thường tiếng Anh từ '**a**' đến '**z**' có vị trí từ **97** đến **122** trong chuỗi thứ tự so sánh. Các vị trí còn lại trong chuỗi thứ tự so sánh sẽ ứng với những ký tự khác, trong đó có những ký tự chuyên dùng để biểu diễn các chữ cái

Hy Lạp, các ký tự dùng để kẻ biểu bảng... Ta có thể khai thác những chi tiết này để viết những thủ tục rất có ích như in biểu bảng khá đẹp khi xuất dữ liệu lên màn hình, tự động tạo các tên file trong chương trình... khi cần thiết.

- **Các hàm LGE, LGT, LLE, LLT**

Những hàm này cho phép ta so sánh những xâu văn bản dựa trên *chuỗi thứ tự so sánh ASCII*. Những hàm này sẽ có ích nếu một chương trình có so sánh các xâu hay sắp xếp ký tự và được dùng trong các máy tính khác nhau. Những hàm này trả về một giá trị lôgic - TRUE hoặc FALSE tuỳ thuộc kết quả so sánh hai đối số kiểu xâu ký tự. Thí dụ, nếu ta có hai biến ký tự XAU1, XAU2 thì **LGE (XAU1, XAU2)** sẽ cho giá trị TRUE nếu XAU1 lớn hơn hoặc bằng XAU2 về phương diện từ vựng. Các hàm **LGT, LLE** và **LLT** thực hiện các phép so sánh "*lớn hơn về từ vựng*", "*nhỏ hơn hoặc bằng về từ vựng*" và "*nhỏ hơn về từ vựng*". Nhớ rằng *các hàm này dựa trên chuỗi thứ tự so sánh ASCII chứ không phải chuỗi thứ tự so sánh của máy tính*.

Trong Fortran 90 còn có **các hàm ADJUSTL, ADJUSTR** dùng để dồn một xâu ký tự về trái hoặc về phải bằng cách cắt bỏ những dấu trống ở phía trái hoặc ở phía phải của xâu đó. **Hàm TRIM** cắt bỏ những dấu trống ở đuôi một xâu văn bản và giảm độ dài xâu cho tương xứng [*].

Thí dụ 34: Sắp xếp danh sách theo thứ tự alphabê. Viết chương trình đọc từ bàn phím tên và số điện thoại của 20 người. In lên màn hình danh sách sắp xếp thứ tự alphabê theo tên người. Trong thí dụ này ta sử dụng các hàm so sánh đối với *bảng thứ tự so sánh ASCII*.

[*] Trong thực tế hàm này và cả hàm **LEN** nữa không làm việc đúng như người ta mô tả nó trong tài liệu, độ dài xâu văn bản nhận được vẫn chỉ là độ dài mô tả chứ không phải độ dài thực tế.

```
      PROGRAM NMSORT
      CHARACTER *8 TEN(20), TEL (20), TEMP
      DO I = 1, 20
         PRINT *, 'NHAP TEN NGUOI THU ', I
         READ 5, TEN(I)
         PRINT *, 'SO DIEN THOAI'
         READ 5, TEL (I)
      ENDDO
    5 FORMAT (A)
      DO I = 1, 19
         K = I
         DO J = I+1, 20
            IF (LGT (TEN (K), TEN (J)))  K = J
         END DO
         TEMP = TEN (K)
         TEN (K) = TEN (I)
         TEN (I) = TEMP
         TEMP = TEL (K)
         TEL (K) = TEL (I)
         TEL (I) = TEMP
         PRINT *, TEN (I), TEL (I)
      END DO
      PRINT *, TEN (20), TEL (20)
      END
```

Thí dụ 35: Mã hoá bức điện. Mã hóa bức điện là làm cho dòng văn bản bình thường của bức điện có một dạng khác thường chỉ có người mã hóa mới hiểu được nội dung của nó. Người ta có thể mã hoá một bức điện theo cách sau: Lấy một xâu gồm 62 chữ cái và chữ số làm *khoá*. Từng chữ

cái bình thường trong bức điện được mã hoá bằng một chữ cái trong khoá sao cho chữ A bình thường được thay bằng chữ cái đầu tiên trong khoá, chữ B được thay bằng chữ cái thứ hai... Dưới đây là một chương trình nhận từ bàn phím một bức điện và in ra màn hình dạng mã hoá của bức điện đó. Trong chương trình này ta dùng một khoá là chuỗi các chữ cái và chữ số sắp xếp theo thứ tự khác thường như sau:

YXAZKLMBJOCFDVSWTREGHNIPUQ

yxazklmbjocfdvswtreghnipuq9087564312

Việc mã hoá các chữ cái trong bức điện được thực hiện trong thủ tục con ENCODE.

```
      PROGRAM MSGCOD
      CHARACTER DIEN*255, MADIEN*255,
      CHARACTER KHOA*62, ALPH*62
      ALPH = ‘ABCDEFGHIJKLMNOPQRSTUVWXYZ’ //
     *  ‘abcdefghijklmnopqrstuvwxyz0123456789'
      KHOA = 'YXAZKLMBJOCFDVSWTREGHNIPUQ’ //
     *  ‘yxazklmbjocfdvswtreghnipuq9087564312'
      PRINT*, 'ENTER A MESSEAGE’,
     *  ‘(MAXIMUM 255 LETTERS)'
      READ (5, '(A255)') DIEN
      CALL ENCODE (KHOA, ALPH, DIEN, MADIEN)
      PRINT 5, MADIEN
    5 FORMAT (1X, /, 1X, 'THIS IS ENCODED AS' /, 1X, A /)
      END
      SUBROUTINE ENCODE (KEY, ALP, MESSGE, SECRET)
      CHARACTER MESSGE * (*), SECRET * (*)
      CHARACTER ALP * (*), KEY * (*), LETTER
      DO I = 1, LEN (MESSGE)
         LETTER = MESSGE (I : I)
         J = INDEX (ALP, LETTER)
         IF (J .EQ. 0) THEN
            SECRET (I : I) = LETTER
         ELSE
            SECRET (I : I) = KEY (J : J)
         END IF
      END DO
      RETURN
      END
```

Bài tập

1. Các biến K và J sẽ có giá trị bằng bao nhiêu sau khi thực hiện nhóm lệnh sau đây:

```
CHARACTER *18 STRG
INTEGER K, J
STR = 'TO BE OR NOT TO BE'
K = INDEX (STRG, 'BE')
J = INDEX (STR (K + 1:), 'BE') + K
```

2. Giả sử các bức điện được mã hoá bằng một khoá như trong thí dụ 31, tức dùng chuỗi các chữ cái và chữ số:

YXAZKLMBJOCFDVSWTREGHNIPUQ

yxazklmbjocfdvswtreghnipuq9087564312

Người ta giải mã như sau: Từng chữ cái trong mã điện sẽ được thay thế bởi một chữ cái trong bảng chữ cái alphabê theo quy tắc: nếu chữ cái trong mã điện trùng với chữ cái thứ nhất trong khoá thì chữ cái đó thay

bằng chữ A trong bảng chữ cái alphabê, nếu trùng với chữ cái thứ hai thì thay nó bằng chữ B... Thí dụ, giả sử mã điện là dòng chữ

DKKG YG YJRWSRG EYGHRZYU

thì theo quy tắc trên, ta có bức điện được giải mã như sau:

MEET AT AIRPORT SATURDAY

Viết chương trình cho phép đọc từ bàn phím một bức điện dưới dạng mã hoá và in lên màn hình dạng đã giải mã của nó.

3. Giả sử danh mục số điện thoại của những người quen của bạn lưu trong file TELNUM dưới dạng những dòng gồm tên người đầy đủ và số điện thoại của mỗi người với format A30, A8. File không có dòng đầu báo thông tin về số dòng dữ liệu và cũng không có dòng ký hiệu cuối file báo hết dữ liệu. Hãy viết chương trình đọc vào từ bàn phím một tên người nào đó, sau đó kiểm tra xem người đó có trong danh mục điện thoại của bạn không. Nếu không có thì đưa ra thông báo 'KHONG CO TRONG DANH MUC', nếu có thì in ra tên người cùng với số điện thoại tìm được sao cho số điện thoại được đặt trong cặp dấu ngoặc ngay sau tên.

4. File dữ liệu ADDR chứa khoảng 50 tên người và địa chỉ. Dòng thứ nhất của mỗi người chứa họ tên đầy đủ (30 ký tự) gồm họ, chữ đệm và tên. Dòng thứ hai chứa địa chỉ số nhà và đường phố (35 ký tự), tên thành phố (15 ký tự) và số điện thoại (15 ký tự). Mỗi xâu ký tự được ghi trong cặp dấu nháy trên. Hãy viết chương trình đọc dữ liệu và in ra thông tin về từng người theo mẫu nhãn sau đây (thí dụ):

HUY, N. Q.
91 NGUYEN THIEN THUAT
NHA TRANG, (058)832536

Mỗi nhãn cách nhau bốn dòng. Chú ý sau tên thành phố là dấu phảy, không nên để một dấu cách nào trước dấu phảy đó.

5. Giả sử bạn đã biết rằng ngày đầu năm của một năm là ngày thứ mấy trong tuần lễ. Hãy viết chương trình in tờ lịch tháng Giêng của năm đó dưới dạng dễ nhìn.

6. Giả sử bạn đã biết ngày đầu năm của một năm nào đó là thứ mấy trong tuần lễ. Hãy viết chương trình in tờ lịch của một tháng, năm bất kỳ trong tương lai dưới dạng dễ nhìn. Tháng và năm nhập từ bàn phím.

7. Viết chương trình in bảng các toán tử lôgic (bảng 4.2, chương 4, trang 56).

8. Viết thủ tục TDBANG (N, TENCOT) trong đó N là đối số nguyên, TENCOT là mảng một chiều gồm N phần tử văn bản chuyên dùng để in ra một tiêu đề cột của bảng. Thí dụ nếu chương trình gọi thủ tục này và chuyển đối số thực tế bằng 12 và một mảng 12 tên viết tắt tháng tiếng Anh ‘JAN’, ’FEB’, ‘MAR’, ‘APR’, ‘MAY’, ‘JUN’, ’JUL’, ‘AUG’, ‘SEP’, ‘OCT’, ‘NOV’, ‘DEC’ thì chương trình sẽ in ra tít đầu bảng có dạng như dưới đây:

JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC

9. Số liệu giá trị ngày của các yếu tố khí tượng thủy văn tại trạm quan trắc được lưu trong file ASCII có quy cách ghi như sau: Dòng trên cùng ghi tên trạm. Dòng thứ 2 có hai số nguyên viết cách nhau lần lượt chỉ tổng số ngày quan trắc và số yếu tố được quan trắc. Dòng thứ ba có 6 số nguyên viết cách nhau lần lượt chỉ ngày, tháng, năm đầu và ngày, tháng, năm cuối quan trắc. Dòng thứ 4 là tiêu đề cột liệt kê tên tất cả các yếu tố được quan trắc, mỗi tên được ghi với độ rộng 8 vị trí và căn bên phải. Các dòng tiếp

theo lần lượt ghi giá trị của các yếu tố, mỗi dòng một ngày. Tính giá trị trung bình tháng của tất cả các yếu tố trong tất cả các năm quan trắc. Kết quả ghi vào những file mới, mỗi yếu tố một file, sao cho tên file trùng với tên của yếu tố quan trắc.

Chương 11 - Những đặc điểm bổ sung về file

11.1. Các file nội tại (Internal Files)

Khi một đơn vị file trong các lệnh nhập hoặc xuất là tên của một biến ký tự, thì lệnh sẽ chuyển dữ liệu từ một vùng lưu giữ nội tại trong bộ nhớ sang một vùng khác. Những vùng lưu giữ này được gọi là các *file nội tại* (internal file). Thí dụ, ta có thể đọc dữ liệu từ một xâu ký tự thay vì đọc từ một dòng dữ liệu trong file thông thường với những lệnh sau đây:

```
CHARACTER * 13 DATA1
INTEGER I, J
REAL X
DATA1 = '137 65 42.17'
READ (DATA1, *) I, J, X
```

Những lệnh trên đây có nghĩa rằng chúng ta khai báo một biến có kiểu văn bản DATA1 với độ dài 13 ký tự. Sau đó gán cho biến này dòng văn bản:

‘137 65 42.17 ‘

Đó là việc bình thường, chúng ta đã biết từ trước đến nay. Nhưng hãy chú ý đến lệnh cuối cùng. Đó là lệnh:

```
READ (DATA1, *) I, J, X
```

Trông lệnh này giống như một lệnh đọc dữ liệu bình thường, chỉ có khác là thay vì đơn vị file trong cặp dấu ngoặc đơn ta đã đưa tên biến DATA1 vào đó. Kết quả là sau lệnh đọc này các đoạn văn bản biểu diễn những chữ số 137, 65 và 42.17 đã được đọc ra như là những số nguyên, số thực và gán vào các biến nguyên I, J và biến thực X trong danh sách các biến cần đọc của lệnh READ một cách đúng đắn. Sau những lệnh này giá trị các biến số sẽ như sau:

I sẽ bằng 137, J sẽ bằng 65 và X bằng 42.17.

Đây là một đặc điểm rất quan trọng của Fortran. Ta sẽ thấy ích lợi của đặc điểm này của file nội tại qua thí dụ sau:

```
      INTEGER PTR
      REAL AMOUNT
      CHARACTER * 15 TEMP
      . . .
      . . .
      READ (12, 5) TEMP
5     FORMAT (A10)
      IF (INDEX (TEMP, '$') .NE. 0) THEN
      PTR = INDEX (TEMP, '$')
      TEMP (PTR: PTR) = ' '
      END IF
      READ (TEMP, *) AMOUNT
```

Với đoạn chương trình này dữ liệu từ trong đơn vị file (12) thông thường (*file ngoại*) được đọc vào biến ký tự TEMP. Trong trường hợp dữ liệu có kèm theo dấu $ ở bên trái (cách viết dấu đô la đằng trước và dính liền số tiền của những người Mỹ thường làm), thì lệnh đọc

```
READ (12,5) TEMP
```

vẫn không mắc lỗi về kiểu dữ liệu. Sau đó ta xử lý, thay ký tự $ bằng ký tự dấu trống và đọc lấy giá trị số thực AMOUNT bằng lệnh đọc file nội tại

```
READ (TEMP, *) AMOUNT
```

Nhận thấy rằng lệnh đọc dữ liệu từ file nội tại hoàn toàn tương tự lệnh đọc các file thông thường. Thay vì số hiệu thiết bị hay số hiệu file, ta ghi tên biến (ở đây là biến **TEMP**) vào vị trí của thiết bị hay số hiệu file.

Bây giờ ta xét một thí dụ về sử dụng file nội để chuyển đổi dữ liệu số thành dữ liệu văn bản. Giả sử ta muốn tạo ra 12 tên file lần lượt là 'GIO.1', 'GIO.2', ..., 'GIO.12'. Đoạn chương trình sau đây có thể làm được việc đó:

```
INTEGER J
CHARACTER *6 TENF(12), TAM
DO J = 1, 12
   IF (J .LT. 10) THEN
      WRITE (TAM, '(I1)') J
   ELSE
      WRITE (TAM, '(I2)') J
   END IF
   TENF (J) = 'GIO' // '.' // TAM
END DO
```

11.2. Các file truy nhập tuần tự (Sequential Files)

Các file được sử dụng trong tất cả các thí dụ từ trước tới nay gọi là *file truy nhập tuần tự* vì một khi file đã được tạo ra, ta không thể cập nhật một bản ghi đơn lẻ nào trong nó. Muốn thay đổi một bản ghi, ta phải đọc các thông tin trong bản ghi, sửa đổi nó và sau đó ghi vào một file khác. Bây giờ ta sẽ xét **lệnh OPEN** *phức tạp có thêm những chỉ định khác so với*

những thí dụ trước đây:

```
OPEN (UNIT = Biểu thức nguyên,
*       FILE = Biểu thức ký tự,
*       ACCESS = Biểu thức ký tự,
*       STATUS = Biểu thức ký tự,
*       FORM = Biểu thức ký tự,
*       IOSTAT = Biến nguyên,
*       RECL = Biểu thức nguyên,
*       BLANK = Biểu thức ký tự,
*       ERR = Nhãn lệnh chuyển điều khiển)
```

Biểu thức nguyên trong chỉ định **UNIT**, thường là một hằng, được sử dụng trong các lệnh READ hoặc WRITE để chỉ đơn vị file được dùng.

Biểu thức ký tự trong chỉ định **FILE** là tên của file cần mở. Hai chỉ định vừa rồi chúng ta đã quen dùng trong các chương trước.

Biểu thức ký tự trong chỉ định **ACCESS** phải có giá trị bằng 'DIRECT" hoặc 'SEQUENTIAL' dùng để chỉ file thuộc loại truy cập trực tiếp hay truy cập tuần tự. Nếu vắng mặt chỉ định này thì ngầm định là 'SEQUENTIAL' như trước đây chúng ta đã dùng.

Biểu thức ký tự của chỉ định **STATUS** có thể có một trong những giá trị là 'NEW' (để chỉ file mới sẽ tạo ra bằng lệnh WRITE), hoặc 'OLD' (file đang tồn tại), hoặc 'UNKNOWN' (chưa rõ), hoặc 'SCRATCH' (file xuất, sẽ bị xóa khi chương trình kết thúc).

Biểu thức ký tự trong chỉ định **FORM** hoặc có giá trị là 'FORMATTED' hoặc là 'UNFORMATTED' hay 'BINARY'. Các file FORMATTED có thể dùng với cả lệnh READ và WRITE có định dạng hoặc dùng với các lệnh nhập, xuất đơn giản. Trong file UNFORMATTED dữ liệu được truy cập như là các xâu nhị phân, không phải là các số hay các ký tự. Nếu chỉ định FORM vắng mặt thì ngầm định sẽ là 'FORMATTED' đối với các file tuần tự và 'UNFORMATTED' đối với các file trực tiếp.

IOSTAT có thể dùng để khôi phục lỗi khi mở file. Nếu không có lỗi khi mở file, biến nguyên sẽ có giá trị 0. Nếu có lỗi, thí dụ không tìm thấy file với tên đã chỉ định, thì một giá trị khác 0 sẽ được lưu trong biến. Người ta thường kiểm tra giá trị của biến này để quyết định hành động tiếp theo. Thí dụ

```
CHARACTER TEN *12, TEMP *70
PRINT *, 'GO TEN FILE'
READ (*, '(A12)') TEN
OPEN (UNIT=15, FILE=TEN, STATUS='OLD', IOSTAT=IERR)
IF (IERR .EQ. 0) THEN
   . . .
   . . .
   . . .
ELSE
   PRINT*, 'LOI MO FILE ',IERR
END IF
```

Đặc tả IOSTAT cũng có thể dùng với các lệnh READ và WRITE.

Chỉ định **RECL** cần cho các file truy cập trực tiếp, không dùng với các file truy cập tuần tự. Biểu thức nguyên nó chỉ định độ dài của một bản ghi.

Biểu thức ký tự của chỉ định **BLANK** có thể là 'NULL' hoặc 'ZERO'. Nếu đặc tả là 'NULL' các dấu trống trong các trường số bị bỏ qua, nếu là 'ZERO' các dấu trống được xem là các số 0. Ngầm định là 'NULL'.

Chỉ định **ERR** là tuỳ chọn và có giá trị để xử lý lỗi. Nếu lỗi xảy ra trong khi thực hiện lệnh OPEN hay một lệnh nào đó có chứa chỉ định này thì chương trình sẽ chuyển điều khiển tới lệnh có nhãn ghi trong chỉ định ERR thay vì tạo ra lỗi thực hiện chương trình. Chỉ định ERR cũng dùng với các lệnh READ và WRITE.

• **Lệnh CLOSE** là một lệnh thực hiện, nó ngắt một file ngoại khỏi chương trình. Dạng tổng quát như sau:

CLOSE (UNIT = *Biểu thức nguyên,*
* **STATUS** = *Biểu thức ký tự,*
* **IOSTAT** = *Biến nguyên,*
* **ERR** = *Nhãn lệnh chuyển điều khiển***)**

Lệnh CLOSE và các chỉ định là tuỳ chọn. Chỉ định STATUS trong lệnh CLOSE có giá trị 'KEEP' có nghĩa file được giữ lại, 'DELETE' có nghĩa file không cần nữa và nên xóa đi.

• **Lệnh REWIND**

REWIND (UNIT = *Biểu thức nguyên,*
* **IOSTAT** = *Biến nguyên,*
* **ERR** = *Nhãn lệnh điều khiển***)**

dùng để chuyển về vị trí bản ghi thứ nhất trong file tuần tự.

• **Lệnh BACKSPACE**

BACKSPACE (UNIT = *Biểu thức nguyên,*
* **IOSTAT** = *Biến nguyên,*
* **ERR** = *Nhãn lệnh điều khiển***)**

chuyển vị trí đọc ngược lại về phía trước một bản ghi trong file tuần tự.

• **Lệnh ENDFILE**

ENDFILE (UNIT = *Biểu thức nguyên,*
* **IOSTAT** = *Biến nguyên,*
* **ERR** = *Nhãn lệnh điều khiển***)**

ghi vào file một bản ghi chỉ sự kết thúc file khi file đã được tạo ra.

11.3. Các file truy cập trực tiếp (Direct-Access Files)

Các bản ghi trong các file truy cập trực tiếp được truy cập không theo cách tuần tự, mà theo thứ tự được chỉ định trong chương trình. *Khi một file trực tiếp được mở, chỉ định* ACCESS *trong lệnh* OPEN *phải đặt là* 'DIRECT' *và độ dài của bản ghi phải được cho với chỉ định* RECL. *Các lệnh* READ *và* WRITE *phải chứa chỉ định* REC *để cung cấp số hiệu của bản ghi cần truy cập*.

Dạng tổng quát của các lệnh READ hoặc WRITE với file truy cập trực tiếp như sau:

READ (*Số hiệu file, nhãn lệnh* FORMAT,
* **REC** = *Biểu thứ nguyên***)** *Danh sách biến*

WRITE (*Số hiệu file, nhãn lệnh* FORMAT,
* **REC** = *Biểu thức nguyên***)** *Danh sách biến*

Biểu thức nguyên trong chỉ định **REC** dùng để chỉ số hiệu bản ghi cần xử lý. Các chỉ định ERR và IOSTAT có thể được sử dụng với các lệnh READ hoặc WRITE trực tiếp. Tuỳ chọn END có thể chỉ dùng với lệnh READ. Khi tổ chức file truy cập trực tiếp, người ta thường sử dụng số thứ

tự hoặc số hiệu phân biệt - một phần của bản ghi làm số hiệu bản ghi. Thí dụ các số hiệu phân biệt của sinh viên trong một trường đại học thường bắt đầu bằng 00001 rồi đến 00002... Do đó thông tin về sinh viên số 00210 có thể được lưu trong bản ghi 210. Đôi khi có thể thực hiện một số tính toán với một trường của bản ghi để nhận được số hiệu của nó.

File truy cập trực tiếp thường được tạo ra bằng cách ghi thông tin vào một cách tuần tự, với bản ghi bắt đầu bằng 1 và tăng lên 1 mỗi lần có một bản ghi mới được viết vào. File này có thể xử lý theo thứ tự tuần tự bằng cách thay đổi số hiệu bản ghi từ 1 đến tổng số tất cả các bản ghi. Tuy nhiên, ưu điểm của file trực tiếp sẽ thể hiện rõ khi chúng ta muốn cập nhật thông tin trong một số bản ghi của file. Thay vì đọc từng bản ghi một cách tuần tự, tìm bản ghi mà ta muốn cập nhật, ta chỉ cần chỉ định số hiệu bản ghi và bản ghi đó tự động được xử lý. Khi cập nhật thông tin xong, ta có thể ghi thông tin mới vào bản ghi. Nếu trong lệnh READ ta chỉ định một số hiệu bản ghi mà bản ghi đó không tồn tại thì sẽ xảy ra lỗi. Để khôi phục lỗi, chỉ định ERR cần phải có mặt trong lệnh READ.

11.4. Lệnh truy vấn INQUIRE

Lệnh INQUIRE có hai dạng:

INQUIRE (FILE = *biểu thức ký tự, danh sách chỉ định truy vấn*)

INQUIRE (UNIT = *biểu thức nguyên, danh sách chỉ định truy vấn*)

Lệnh này là một lệnh thực hiện, nó truy vấn thông tin về file hay số hiệu file. Bảng 9.1 liệt kê những chỉ định truy vấn. Thí dụ:

```
INQUIRE (FILE = 'TSDATA', SEQUENTIAL = TRALOI)
INQUIRE (UNIT = 12, SEQUENTIAL = TRALOI)
```

Bảng 11.1. Các chỉ định truy vấn của lệnh **INQUIRE**

Chỉ định truy vấn	Kiểu biến	Giá trị truy vấn file FILE	Giá trị truy vấn đơn vị file UNIT
ACCESS =	CHARACTER	'SEQUENTIAL' 'DIRECT'	'SEQUENTIAL' 'DIRECT'
BLANK =	CHARACTER	'NULL' 'ZERO'	'NULL' 'ZERO'
DIRECT =	CHARACTER	'YES' 'NO'	–
ERR =	INTEGER	*Số hiệu lệnh xử lý lỗi*	*Số hiệu lệnh xử lý lỗi*
EXIST =	LOGICAL	.TRUE. .FALSE.	.TRUE. .FALSE.
FORM =	CHARACTER	'FORMATTED' 'UNFORMATTED'	'FORMATTED' 'UNFORMATTED'
FORMATTED =	CHARACTER	'YES' 'NO' 'UNKNOWN'	–
IOSTAT =	INTEGER	*Mã lỗi*	*Mã lỗi*
NAME =	CHARACTER	–	*Tên file nếu file đó không phải là file loại scratch*
NAMED + =	LOGICAL	–	.TRUE. .FALSE.
NEXTREC =	INTEGER	Số hiệu bản ghi tiếp theo trong file truy cập trực tiếp	Số hiệu bản ghi tiếp theo trong file truy cập trực tiếp
NUMBER + =	INTEGER	Đơn vị file	–
OPEND =	LOGICAL	.TRUE. .FALSE.	.TRUE. .FALSE.
RECL =	INTEGER	Độ dài bản ghi	Độ dài bản ghi
SEQUENTIAL =	CHARACTER	'YES' 'NO' 'UNKNOWN'	–
UNFORMATTED =	CHARACTER	'YES' 'NO' 'UNKNOWN'	–

Thí dụ 36: Sự tương tác giữa người dùng và chương trình. Giả sử khi chương trình yêu cầu người dùng gõ một tên của file dữ liệu để mở ra làm việc trong chương trình. Trường hợp file đó không tồn tại, chương trình sẽ kết thúc bởi lỗi thực hiện. Nếu ta dùng lệnh INQUIRE, chương trình có thể xác định file đó có tồn tại không và nếu không tồn tại, chương trình nhắc người dùng gõ một tên file khác. Các lệnh sau đây thực hiện sự tương tác này:

```
      CHARACTER *70 TENFIL, TIT
      LOGICAL XONG, OK, CO
      XONG = .FALSE.
      OK = .FALSE.
      PRINT *, 'NHAP TEN FILE'
      READ *, TENFIL
5     IF (.NOT. XONG) THEN
         INQUIRE (FILE = TENFIL, EXIST = CO)
         IF (.NOT. CO) THEN
            PRINT *, 'FILE KHONG TON TAI'
            PRINT *, 'NHAP TEN KHAC HOAC GO THOI'
            READ *, TENFIL
            IF (TENFIL .EQ. 'THOI') XONG = .TRUE.
         ELSE
            XONG = .TRUE.
            OK = .TRUE.
         ENDIF
         GOTO 5
      ENDIF
      IF (OK) THEN
         OPEN (UNIT = 10, FILE = TENFIL, STATUS = 'OLD')
         . . .
         . . .
         . . .
      END IF
      END
```

Bài tập

1. Viết chương trình đếm và in số bản ghi trong các file DATA1 và DATA2. Giả sử các file đó là file tuần tự và mỗi bản ghi chứa hai giá trị thực với format sau:

FORMAT (F6.2, 1X, F6.2)

Nếu lỗi xảy ra khi mở file, hãy in thông báo lỗi thay vì in số bản ghi.

2. File TEM60.JAN lưu trường ba chiều nhiệt độ nước biển Đông tháng Giêng độ phân giải 1^o kinh vĩ có quy cách ghi như sau: Dòng đầu tiên gồm 5 số nguyên cách nhau lần lượt chỉ kinh độ mép trái, kinh độ mép phải, vĩ độ mép trên, vĩ độ mép dưới của miền không gian và số tầng sâu. Dòng thứ hai ghi độ sâu (số nguyên) của tầng trên cùng. Sau đó là bảng giá trị nhiệt độ (số thực cách nhau) với số cột bằng số điểm nút theo kinh tuyến, số dòng bằng số điểm nút theo vĩ tuyến. Các tầng tiếp theo ghi hoàn toàn tương tự. Giá trị nhiệt độ khuyết hoặc rơi vào vùng đất liền được ghi bằng số 99.99. Viết chương trình tính giá trị nhiệt độ nước trung bình toàn biển Đông.

3. Với file số liệu của bài tập 2, viết chương trình đọc thông tin trong file và tạo cho mỗi điểm nút thuộc miền tính một file đặt tên theo quy tắc sau: bắt đầu bằng chữ K, sau đó đến các chữ số chỉ kinh độ điểm, sau đó chữ V và các chữ số chỉ vĩ độ điểm, đuôi file là '.BLN'. Trong các file có

quy cách ghi như sau, dòng trên cùng có một số nguyên chỉ số tầng quan trắc thực tế của điểm, một dấu cách và chữ số 1. Sau đó liệt kê liên tiếp giá trị nhiệt độ và tầng sâu ứng với nhiệt độ đó với dấu ngược lại.

4. Giả sử có file dữ liệu lưu giá trị quan trắc của một số yếu tố khí tượng thủy văn tại trạm hải văn, có quy cách ghi như sau:

- Dòng thứ nhất có hai số nguyên 1 và 2 cách nhau một dấu trống.

- Dòng thứ hai ghi tên trạm (không quá 100 ký tự).

- Dòng thứ ba ghi hai số nguyên chỉ số dòng dữ liệu (không quá 5000) và số yếu tố quan trắc (không quá 12) cách nhau ít nhất một dấu trống.

- Dòng thứ tư lần lượt ghi tên các yếu tố được quan trắc, mỗi tên với định dạng A8.

- Dòng thứ 5 lần lượt ghi đơn vị đo của từng yếu tố quan trắc, cũng với định dạng A8.

- Mỗi dòng trong các dòng tiếp sau lần lượt ghi giá trị quan trắc của các yếu tố, mỗi giá trị ghi với định dạng F8.2.

Viết chương trình cho phép nhập tên file từ bàn phím, đọc dữ liệu và lập phương trình hồi quy giữa biến thứ nhất (biến phụ thuộc) và biến thứ hai (biến độc lập). In kết quả ra màn hình theo quy cách sau: giả sử tên biến thứ nhất là Tw, biến thứ hai là Ta, phương trình phải viết có dạng:

Tw = 0.915 Ta + 1.237

(Ghi chú: xem công thức trong phụ lục 3).

5. Cải tiến chương trình trong bài tập 4 để cho phép người dùng tuỳ ý chỉ định biến phụ thuộc và biến độc lập từ bàn phím.

6. Với file dữ liệu đã mô tả trong bài tập 4, lập chương trình tính phương trình hồi quy nhiều biến giữa yếu tố quan trắc thứ nhất (biến phụ thuộc) và các yếu tố quan trắc 2, 3, 6, 8, 9. In kết quả lên màn hình dưới dạng phương trình hồi quy với tên các yếu tố đã ghi trong file.

Gợi ý: Xem phương pháp thiết lập phương trình hồi quy tuyến tính nhiều biến trong phụ lục 4.

7. File HESOA.MAT lưu các giá trị của các hệ số của hệ phương trình đại số tuyến tính theo quy cách như sau: Dòng thứ nhất có một số nguyên chỉ số phương trình. Các dòng tiếp sau ghi các giá trị các hệ số, kể cả hệ số tự do ứng với từng phương trình, mỗi phương trình trên một dòng, mỗi hệ số ghi với định dạng F8.4, thí dụ:

```
4   1.1161  0.1254  0.1397  0.1490  1.5471
    0.1582  1.1675  0.1768  0.1871  1.6471
    0.1968  0.2071  1.2168  0.2271  1.7471
    0.2368  0.2471  0.2568  1.2671  1.8471
```

Viết chương trình đọc file và giải hệ phương trình bằng phương pháp loại biến Gauss. Kết quả in ra màn hình gồm: viết lại hệ phương trình, sau đó cách ra một dòng rồi ghi các nghiệm ở dòng cuối cùng, thí dụ, ứng với ma trận các hệ số như trên phải có kết quả trên màn hình như sau:

```
1.1161X1 + 0.1254X2 + 0.1397X3 + 0.1490X4 = 1.5471
0.1582X1 + 1.1675X2 + 0.1768X3 + 0.1871X4 = 1.6471
0.1968X1 + 0.2071X2 + 1.2168X3 + 0.2271X4 = 1.7471
0.2368X1 + 0.2471X2 + 0.2568X3 + 1.2671X4 = 1.8471


1.04059   0.98697   0.93505   0.88130
```

Gợi ý: Xem phương pháp giải hệ phương trình đại số tuyến tính theo sơ đồ loại biến Gauss trong phụ lục 2.

8. File HESOAB.MAT lưu các giá trị của các hệ số của hệ phương trình đại số tuyến tính theo quy cách như đã mô tả trong bài tập 7. Giả sử

ma trận các hệ số $A=[a_{ij}]$ là ma trận đối xứng, tức $a_{ij}=a_{ji}\quad (i,j=1,2,...,n)$.

Hãy viết chương trình đọc file các hệ số và giải hệ phương trình. In kết quả theo quy cách của bài tập 7.

Gợi ý: Trường hợp ma trận các hệ số A là ma trận đối xứng, nên dùng phương pháp căn bậc hai để giải hệ phương trình đại số tuyến tính (phụ lục 2).

Tài liệu tham khảo

1. Etter D. M. Structured Fortran 77 for engineers and scientists. Fourth edition. The Benjamin/Cummings Publishing Co., Inc. California, 1993, 616 p.
2. Koffman Elliot B., Friedman Frank L. Fortran with engineering applications. Fifth Edition. Addison-Wesley Publishing Co. Massachusetts-..., 1993, 664 p.
3. N. V. Kopchenova and I.A. Maron. Computational Mathematics. Worked examples and problems with elements of theory. Mir Publishers, Moscow, 1975.
4. Васильевич О. Б. Современный Фортран. “Диалог-Мифи”. Москва, 1998, 397 с.
5. Васильевич О. Б. Фортран для профессионалов: Математическая библиотека IMSL. “Диалог -Мифи”, Москва, 2000, 448 с.
6. Тюрин Ю. Н., Макаров А. А. Статистический анализ данных на компьютере. “ИНФРА” - Москва, 1998, 528 с.

Phụ lục 1: Bảng các hàm chuẩn của FORTRAN

Trong bảng các hàm chuẩn dưới đây, tên của các đối số sẽ chỉ kiểu dữ liệu theo quy ước sau:

Đối số		*Kiểu dữ liệu*
X	→	thực
CHX	→	xâu ký tự
DX	→	độ chính xác đôi
CX	→	phức
LX	→	lôgic
IX	→	nguyên
GX	→	tự sinh (in đậm, nghiêng)

Tên hàm	Kiểu hàm	Định nghĩa
SQRT(X)	Thực	$\sqrt{X}$
DSQRT (DX)	Độ chính xác đôi	$\sqrt{DX}$
CSQRT (CX)	Phức	$\sqrt{CX}$
ABS (X)	Thực	$\lvert X\rvert$
IABS (IX)	Nguyên	$\lvert IX\rvert$
DABS (DX)	Độ chính xác đôi	$\lvert DX\rvert$
CABS (CX)	Phức	$\lvert CX\rvert$

Tên hàm	Kiểu hàm	Định nghĩa
EXP (X)	Thực	e^{X}
DEXP (DX)	Độ chính xác đôi	e^{DX}
CEXP (CX)	Phức	e^{CX}
LOG (GX)	Kiểu theo GX	$\log_e GX$
ALOG (X)	Thực	$\log_e X$
DOG (GX)	Độ chính xác đôi	$\log_e DX$
CLOG (CX)	Phức	$\log_e CX$
LOG10 (GX)	Kiểu theo GX	$\log_{10} GX$
ALOG10 (X)	Thực	$\log_{10} X$
DLOG10 (DX)	Độ chính xác đôi	$\log_{10} DX$
REA L(GX)	Thực	Chuyển GX thành giá trị thực
FLOAT (IX)	Thực	Chuyển IX thành giá trị thực
SNGL (DX)	Thực	Chuyển DX thành độ chính xác đơn
ANINT(X)	Thực	Làm tròn tới số thực gần nhất
DNINT(DX)	Độ chính xác đôi	Làm tròn tới số thực gần nhất
NINT(X)	Nguyên	Làm tròn tới số nguyên gần nhất
IDNINT (DX)	Nguyên	Làm tròn tới số nguyên gần nhất
AINT (X)	Thực	Cắt phần thập phân của *X*
DINT (DX)	Độ chính xác đôi	Cắt phần thập phân của *DX*
INT (GX)	Nguyên	Cắt *GX* thành số nguyên
IFIX (X)	Nguyên	Cắt *X* thành số nguyên
IDINT (DX)	Nguyên	Cắt *DX* thành số nguyên
SIGN (X, Y)	Thực	Gán dấu của *Y* cho $\lvert X\rvert$
ISIGN (IX, IY)	Nguyên	Gán dấu của *IY* cho $\lvert IX\rvert$
DSIGN (DX, DY)	Độ chính xác đôi	Gán dấu của *DY* cho $\lvert DX\rvert$

Tên hàm	Kiểu hàm	Định nghĩa
MOD (IX,IY)	Nguyên	Lấy số dư của phép chia *IX* / *IY*
AMOD (X,Y)	Thực	Lấy số dư của phép chia *X* / *Y*
DMOD (DX,DY)	Độ chính xác đôi	Lấy số dư của phép chia *DX* / *DY*
DIM (X,Y)	Thực	*X* – (cực tiểu của *X* và *Y*)
IDIM (IX,IY)	Nguyên	*IX* – (cực tiểu của *IX* và *IY*)
DDIM (DX,DY)	Độ chính xác đôi	*DX* – (cực tiểu của *DX* và *DY*)
MAX (GX,GY,...)	Kiểu theo GX, GY, ...	Cực đại của (*GX, GY,* ...)
MAX0 (IX,IY,...)	Nguyên	Cực đại của (*IX, IY,* ...)
AMAX1 (X,Y,...)	Thực	Cực đại của (*X, Y,* ...)
DMAX1 (DX,DY,...)	Độ chính xác đôi	Cực đại của (*DX, DY,* ...)
AMAX0 (IX,IY,...)	Thực	Thực, cực đại của (*IX, IY,* ...)
MAX1 (X,Y,...)	Nguyên	Cực đại của (*X, Y,* ...)
MIN (GX,GY,...)	Kiểu theo GX,GY,...	Cực tiểu của (*GX, GY,* ...)
MIN0 (IX,IY,...)	Nguyên	Cực tiểu của (*IX, IY,* ...)
AMIN1 (X,Y,...)	Thực	Cực tiểu của (*X, Y,* ...)
DMIN1 (DX,DY,...)	Độ chính xác đôi	*Cực tiểu của (DX, DY,* ...)
AMIN0 (IX,IY,...)	Thực	Cực tiểu của (*IX, IY,* ...)
MIN1 (X,Y,...)	Nguyên	Cực tiểu của (*X, Y,* ...)
SIN (X)	Thực	$\sin X$ (*X* - rađian)
DSIN (DX)	Độ chính xác đôi	$\sin DX$ (*DX* - rađian)
CSIN (CX)	Phức	$\sin CX$
COS (X)	Thực	$\cos X$ (*X* - rađian)
DCOS (DX)	Độ chính xác đôi	$\cos DX$ (*DX* - rađian)
CCOS (CX)	Phức	$\cos CX$
TAN (X)	Thực	$\mathrm{tg}X$ (*X* - rađian)
DTAN (DX)	Độ chính xác đôi	$\mathrm{tg}DX$ (*DX* - rađian)

Tên hàm	Kiểu hàm	Định nghĩa
ASIN (X)	Thực	$\arcsin X$
DASIN (X)	Độ chính xác đôi	$\arcsin DX$
ACOS (X)	Thực	$\arccos X$
DACOS (DX)	Độ chính xác đôi	$\arccos DX$
ATAN (X)	Thực	$\mathrm{arctg}X$
DATAN (DX)	Độ chính xác đôi	$\mathrm{arctg}DX$
ATAN2 (X,Y)	Thực	$\mathrm{arctg}\,(X/Y)$
DATAN2 (DX,DY)	Độ chính xác đôi	$\mathrm{arctg}\,(DX/DY)$
SINH (X)	Thực	$\mathrm{sh}X$
DSINH (DX)	Độ chính xác đôi	$\mathrm{sh}DX$
COSH (X)	Thực	$\mathrm{ch}X$
DCOSH (DX)	Độ chính xác đôi	$\mathrm{ch}DX$
TANH (X)	Thực	$\mathrm{th}\,X$
DTANH (DX)	Độ chính xác đôi	$\mathrm{th}DX$
DPROD (X, Y)	Độ chính xác đôi	Tích của *X* và *Y*
DBLE (X)	Độ chính xác đôi	Chuyển X thành độ chính xác đôi
CMPLX (X)	Phức	$X+0i$
CMPLX (X, Y)	Phức	$X+Yi$
AIMAG (CX)	Thực	Phần ảo của *CX*
REAL (CX)	Thực	Phần thực của *CX*
CONJG (CX)	Phức	Liên hợp của *CX*, $a-bi$
LEN (CHX)	Nguyên	Độ dài của xâu ký tự *CHX*
INDEX (CHX, CHY)	Nguyên	Vị trí của xâu CHY trong xâu *CHX*

Tên hàm	Kiểu hàm	Định nghĩa
CHAR (IX)	Ký tự	Ký tự ứng với vị trí thứ *IX* trong chuỗi so sánh
ICHAR (CHX)	Nguyên	Vị trí của ký tự *CHX* trong chuỗi so sánh
LGE (CHX, CHY)	Lôgic	Giá trị của biểu thức (*CHX* lớn hơn hoặc bằng *CHY* về từ vựng)
LGT (CHX, CHY)	Lôgic	Giá trị của biểu thức (*CHX* lớn hơn *CHY* về từ vựng)
LLE (CHX, CHY)	Lôgic	Giá trị của biểu thức (*CHX* nhỏ hơn hoặc bằng *CHY* về từ vựng)
LLT (CHX, CHY)	Lôgic	Giá trị của biểu thức (*CHX* nhỏ hơn *CHY* về từ vựng)

Phụ lục 2: Phương pháp Gauss giải hệ phương trình đại số tuyến tính

$$\left.\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \ldots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \ldots & + & a_{2n}x_n & = & b_2 \\ \ldots & & \ldots & & \ldots & & \ldots & & \ldots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \ldots & + & a_{nn}x_n & = & b_n \end{array}\right\}$$

hay $$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b} \qquad (*)$$

$$A = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}; \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{pmatrix}; \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix}.$$

Nếu ma trận A không suy biến, tức

$$\det A = \begin{vmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{vmatrix} \neq 0$$

thì hệ (*) có nghiệm duy nhất. Có thể tính nghiệm theo công thức Cramer

$$x_i = \frac{\det A_i}{\det A},$$

trong đó A_i – ma trận A với cột i bị thay thế bằng cột các số hạng tự do b.

1. Phương pháp loại biến Gauss giải hệ phương trình đại số tuyến tính:

Thí dụ cho hệ

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= a_{15} \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= a_{25} \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 &= a_{35} \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 &= a_{45} \end{aligned}\right\} \quad (1)$$

Giả sử phần tử chính $a_{11} \neq 0$. Chia phương trình thứ nhất cho a_{11}, ta có

$$x_1 + b_{12}x_2 + b_{13}x_3 + b_{14}x_4 = b_{15}, \quad (2)$$

với $$b_{1j} = \frac{a_{1j}}{a_{11}} \quad (j = 2, 3, 4, 5).$$

Dùng phương trình (2) để loại ẩn x_1 khỏi các phương trình số 2, 3, 4 của hệ (1): Muốn vậy, nhân phương trình (2) tuần tự với a_{21}, a_{31}, a_{41} và tuần tự lấy các phương trình số 2, 3, 4 trừ đi các tích tương ứng vừa nhận được, ta có ba phương trình:

$$\left.\begin{aligned} a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + a_{24}^{(1)}x_4 &= a_{25}^{(1)} \\ a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + a_{34}^{(1)}x_4 &= a_{35}^{(1)} \\ a_{42}^{(1)}x_2 + a_{43}^{(1)}x_3 + a_{44}^{(1)}x_4 &= a_{45}^{(1)} \end{aligned}\right\} \quad (3)$$

trong đó

$$a_{ij}^{(1)} = a_{ij} - a_{i1}b_{1j} \quad (i = 2, 3, 4; \quad j = 2, 3, 4, 5) \quad (4)$$

Bây giờ chia phương trình thứ nhất của hệ (3) cho phần tử chính $a_{22}^{(1)}$ ta có:

$$x_2 + b_{23}^{(1)}x_3 + b_{24}^{(1)}x_4 = b_{25}^{(1)}, \quad (5)$$

trong đó

$$b_{2j}^{(1)} = \frac{a_{2j}^{(1)}}{a_{22}^{(1)}} \quad (j = 3,\ 4,\ 5).$$

Bằng cách tương tự như khi loại x_1, bây giờ ta loại x_2 khỏi các phương trình thứ ba và thứ tư, ta có:

$$\left.\begin{aligned} a_{33}^{(2)}x_3 + a_{34}^{(2)}x_4 &= a_{35}^{(2)} \\ a_{43}^{(2)}x_3 + a_{44}^{(2)}x_4 &= a_{45}^{(2)} \end{aligned}\right\}. \quad (6)$$

trong đó

$$a_{ij}^{(2)} = a_{ij}^{(1)} - a_{i2}^{(1)}b_{2j}^{(1)} \quad (i = 3, 4; \quad j = 3, 4, 5). \quad (7)$$

Chia phương trình thứ nhất của hệ (6) cho phần tử chính $a_{33}^{(2)}$, ta có:

$$x_3 + b_{34}^{(2)}x_4 = b_{35}^{(2)}, \quad (8)$$

trong đó

$$b_{3j}^{(2)} = \frac{a_{3j}^{(2)}}{a_{33}^{(2)}} \qquad (j = 4, 5).$$

Sau đó nhờ (8) ta loại x_3 khỏi phương trình thứ hai của hệ (6), nhận được:

$$a_{44}^{(3)} x_4 = a_{45}^{(3)}$$

trong đó

$$a_{4j}^{(3)} = a_{4j}^{(2)} - a_{43}^{(2)} b_{3j}^{(2)} \qquad (j = 4, 5) \tag{9}$$

Như vậy ta đã đưa hệ (1) về hệ tương đương có ma trận các hệ số là ma trận tam giác

$$\left.\begin{aligned} x_1 + b_{12}x_2 + b_{13}x_3 + b_{14}x_4 &= b_{15} \\ x_2 + b_{23}^{(1)}x_3 + b_{24}^{(1)}x_4 &= b_{25}^{(1)} \\ x_3 + b_{34}^{(2)}x_4 &= b_{35}^{(2)} \\ a_{44}^{(3)}x_4 &= a_{45}^{(3)} \end{aligned}\right\} \tag{10}$$

Từ (10) xác định các ẩn

$$\left.\begin{aligned} x_4 &= a_{45}^{(3)} / a_{44}^{(3)} \\ x_3 &= b_{35}^{(2)} - x_4 b_{34}^{(2)} \\ x_2 &= b_{25}^{(1)} - x_4 b_{24}^{(1)} - x_3 b_{23}^{(1)} \\ x_1 &= b_{15} - x_4 b_{14} - x_3 b_{13} - x_2 b_{12} \end{aligned}\right\} \tag{11}$$

Vậy thủ tục giải hệ phương trình đại số tuyến tính bậc nhất quy về hai quá trình:

a) Quá trình thuận: đưa hệ (1) về dạng tam giác (10);

b) Quá trình nghịch: tìm ẩn theo các công thức (11).

Nếu phần tử chính của hệ bằng không thì chỉ cần thay đổi chỗ của các phương trình trong hệ tương ứng để làm cho phần tử chính khác không.

Số phép tính số học N cần thực hiện trong phương pháp Gauss bằng

$$N = \frac{2n(n+1)(n+2)}{3} + n(n-1).$$

Vậy số phép tính số học xấp xỉ tỷ lệ với luỹ thừa bậc ba của số ẩn.

2. Phương pháp căn bậc giải hệ phương trình đại số tuyến tính trong trường hợp ma trận A là ma trận đối xứng

Phương pháp này thuận lợi trong trường hợp hệ phương trình

$$A\,\boldsymbol{x} = \boldsymbol{b} \tag{12}$$

có ma trận A là ma trận đối xứng, điều thường gặp trong các bài toán kỹ thuật.

Theo phương pháp này ma trận A được biểu diễn thành tích của hai ma trận tam giác chuyển vị

$$A = T'\,T \tag{13}$$

trong đó

$$T = \begin{pmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ 0 & t_{22} & \dots & t_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & t_{nn} \end{pmatrix}, \qquad T' = \begin{pmatrix} t_{11} & & \dots & 0 \\ t_{12} & t_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ t_{1n} & t_{2n} & \dots & t_{nn} \end{pmatrix}$$

Nhân hai ma trận T' và T và cho tích bằng ma trận A, ta suy ra cá công thức tính các phần tử t_{ij}:

$$t_{11} = \sqrt{a_{11}}, \quad t_{1j} = \frac{a_{1j}}{t_{11}} \quad (j > 1)$$

$$t_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} t_{ki}^2} \quad (1 < i \le n) \qquad (14)$$

$$t_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} t_{ki} t_{kj}}{t_{ii}} \quad (i < j)$$

$$t_{ij} = 0 \quad khi \quad i > j$$

Như vậy ta đã thay hệ (12) bằng hai hệ tương đương

$$T'\boldsymbol{y} = \boldsymbol{b}, \qquad T\boldsymbol{x} = \boldsymbol{y} \qquad 15)$$

hay

$$\left.\begin{aligned} &t_{11}y_1 = b_1 \\ &t_{12}y_1 + t_{22}y_2 = b_2 \\ &\cdots\cdots\cdots\cdots\cdots \\ &t_{1n}y_1 + t_{2n}y_2 + \ldots + t_{nn}y_n = b_n \end{aligned}\right\} \qquad (16)$$

$$\left.\begin{aligned} t_{11}x_1 + t_{12}x_2 + \ldots + t_{1n}x_n &= y_1 \\ t_{22}x_2 + \ldots + t_{2n}x_n &= y_2 \\ \cdots\cdots\cdots\cdots\cdots & \\ t_{nn}x_n &= y_n \end{aligned}\right\} \qquad (17)$$

Từ đó suy ra các công thức tính:

$$y_1 = \frac{b_1}{t_{11}}, \quad y_i = \frac{b_i - \sum_{k=1}^{i-1} t_{ki} y_k}{t_{ii}} \quad (i > 1) \qquad (18)$$

$$x_n = \frac{y_n}{t_{nn}}, \quad x_i = \frac{y_i - \sum_{k=i+1}^{n} t_{ik} x_k}{t_{ii}} \quad (i < n) \qquad (19)$$

Vậy quá trình thuận gồm tính các phần tử của ma trận T theo các công thức (14). Quá trình nghịch là tính các ma trận cột y và x theo các công thức (18), (19).

Phụ lục 3: Phương pháp bình phương nhỏ nhất trong phân tích hồi quy

1. Mô hình tuyến tính

Mô hình hồi quy tuyến tính có dạng:

$$y = f(x) = ax + b\,.$$

Theo phương pháp bình phương nhỏ nhất, các hệ số hồi quy a và b trong phương trình trên được tìm sao cho tổng bình phương sai số bằng

$$E = \sum_{k=1}^{n} (y_k - ax_k - b)^2$$

cực tiểu. Lần lượt lấy đạo hàm biểu thức này theo a, b và cho bằng không, ta được hệ phương trình sau đây để xác định a và b:

$$a\sum_{k=1}^{n} x_k^2 + b\sum_{k=1}^{n} x_k = \sum_{k=1}^{n} x_k y_k \,,\; a\sum_{k=1}^{n} x_k + b\,n = \sum_{k=1}^{n} y_k \,.$$

Vậy các hệ số hồi quy được tính theo các công thức sau:

$$a = \frac{\sum_{k=1}^{n} x_k \sum_{k=1}^{n} y_k - n\sum_{k=1}^{n} x_k y_k}{(\sum_{k=1}^{n} x_k)^2 - n\sum_{k=1}^{n} x_k^2} \qquad (20)$$

$$b = \frac{\sum_{k=1}^{n} x_k \sum_{k=1}^{n} x_k y_k - \sum_{k=1}^{n} x_k^2 \sum_{k=1}^{n} y_k}{(\sum_{k=1}^{n} x_k)^2 - n\sum_{k=1}^{n} x_k^2}\,, \qquad (21)$$

hay hệ số b còn có thể tính theo công thức:

$$b = \frac{\sum_{k=1}^{n} y_k - a\sum_{k=1}^{n} x_k}{n}\,. \qquad (22)$$

2. Mô hình đa thức

Phương pháp bình phương nhỏ nhất cũng có thể áp dụng để tính các hệ số hồi quy đa thức dạng

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^m\,.$$

thí dụ đối với mô hình bậc hai

$$f(x) = a_0 + a_1 x + a_2 x^2\,.$$

Lấy đạo hàm tổng sai số theo các hệ số và cho bằng không ta có hệ sau đây để xác định các hệ số hồi quy bậc hai:

$$\begin{cases} a_2 \sum_{k=1}^{n} x_k^2 + a_1 \sum_{k=1}^{n} x_k + a_0 n = \sum_{k=1}^{n} y_k \\ a_2 \sum_{k=1}^{n} x_k^3 + a_1 \sum_{k=1}^{n} x_k^2 + a_0 \sum_{k=1}^{n} x_k = \sum_{k=1}^{n} x_k y_k \\ a_2 \sum_{k=1}^{n} x_k^4 + a_1 \sum_{k=1}^{n} x_k^3 + a_0 \sum_{k=1}^{n} x_k^2 = \sum_{k=1}^{n} x_k^2 y_k \end{cases} \quad (23)$$

Về nguyên tắc ta có thể sử dụng phương pháp này để tìm phương trình đa thức bậc bất kỳ. Tuy nhiên trong thực tế phương pháp trở thành không ổn định khi bậc đa thức lớn hơn vì các sai số làm tròn số trong máy tính.

3. Mô hình phi tuyến

Phương pháp bình phương nhỏ nhất có thể áp dụng cho hàm bất kỳ, nhưng hệ các phương trình để tìm các hệ số có thể phi tuyến, và do đó không thể giải được bằng cách sử dụng các phương trình tuyến tính. Tuy nhiên, trong một số trường hợp, một hàm phi tuyến có thể chuyển thành một hàm tuyến tính. Thí dụ về một hàm có thể tuyến tính hoá là

$$f(x) = bx^a \quad (24)$$

Nếu lấy loga hai vế của phương trình này, ta có

$$\ln f(x) = a \ln x + \ln b\,. \quad (25)$$

Nếu ký hiệu

$$g(x) = \ln f(x) \quad (26)$$

$$\tilde{b} = \ln b \quad (27)$$

$$\tilde{x} = \ln x \qquad (28)\ \tilde{y} = \ln y \quad (29)$$

ta có

$$g(x) = a\tilde{x} + \tilde{b} \quad (30)$$

Với phương trình (30) các hệ số hồi quy a và $\tilde{b}$ tính theo các công thức

$$a = \frac{\sum_{k=1}^{n} \tilde{x}_k \sum_{k=1}^{n} \tilde{y}_k - n \sum_{k=1}^{n} \tilde{x}_k \tilde{y}_k}{(\sum_{k=1}^{n} \tilde{x}_k)^2 - n \sum_{k=1}^{n} \tilde{x}_k^2} \quad (31)$$

$$\tilde{b} = \frac{\sum_{k=1}^{n} \tilde{x}_k \sum_{k=1}^{n} \tilde{x}_k \tilde{y}_k - \sum_{k=1}^{n} \tilde{x}_k^2 \sum_{k=1}^{n} \tilde{y}_k}{(\sum_{k=1}^{n} \tilde{x}_k)^2 - n \sum_{k=1}^{n} \tilde{x}_k^2} \quad (32)$$

Vậy công việc tính toán gồm: chuyển đổi các giá trị số liệu x_k và y_k theo các công thức (28), (29), tính các tổng, kết quả thế vào các phương trình (31), (32) để tìm a và $\tilde{b}$. Giải phương trình (27) đối với b và đặt vào phương trình (24).

Phụ lục 4: Sơ đồ ứng dụng phương pháp hồi quy nhiều biến

Giả sử có n quan trắc đối với biến phụ thuộc y và các biến độc lập $x_1, x_2, ..., x_m$. Phương trình hồi quy được thiết lập như sau:

$$y = a_0 + a_1x_1 + a_2x_2 + ... + a_mx_m .$$

Các hệ số hồi quy a_i $(i = 1,...,m)$ được chọn sao cho thoả mãn

$$\delta = \sum_{i=1}^{n} (y - a_0 - a_1x_1 - a_2x_2 - ... - a_mx_m)^2 = \min$$

Lần lượt lấy đạo hàm biểu thức trên theo $a_0, a_1, a_2, ..., a_m$ và cho các đạo hàm bằng không, ta có hệ $m+1$ phương trình để xác định các hệ số a

$$\begin{array}{ccccccccccc}
na_0 & + & [x_1]a_1 & + & [x_2]a_2 & + & ... & + & [x_m]a_m & = & [y] \\
[x_1]a_0 & + & [x_1x_1]a_1 & + & [x_2x_1]a_2 & + & ... & + & [x_mx_1]a_m & = & [yx_1] \\
[x_2]a_0 & + & [x_1x_2]a_1 & + & [x_2x_2]a_2 & + & ... & + & [x_mx_2]a_m & = & [yx_2] \\
... & & ... & & ... & & ... & & ... & & ... \\
[x_m]a_0 & + & [x_1x_m]a_1 & + & [x_2x_m]a_2 & + & ... & + & [x_mx_m]a_m & = & [yx_m]
\end{array}$$

(33)

Hệ phương trình này gọi là hệ phương trình chính tắc để xác định các hệ số hồi quy. Dưới dạng ma trận ta viết hệ này như sau:

$$\begin{pmatrix}
n & [x_1] & [x_2] & ... & [x_m] \\
[x_1] & [x_1x_1] & [x_2x_1] & ... & [x_mx_1] \\
[x_2] & [x_1x_2] & [x_2x_2] & ... & [x_mx_2] \\
... & ... & ... & ... & ... \\
[x_m] & [x_1x_m] & [x_2x_m] & ... & [x_mx_m]
\end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ ... \\ a_m \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ ... \\ b_m \end{pmatrix}$$

(34)

với dấu [ ] ký hiệu phép lấy tổng $\sum_{1}^{n}$.

Để tìm các hệ số hồi quy $a_0, a_1, a_2, ..., a_m$ ta phải giải hệ phương trình chính tắc theo phương pháp loại biến Gauss hoặc phương pháp căn bậc hai đã mô tả trong phụ lục 2 vì ma trận hệ số của các phương trình chính tắc là ma trận đối xứng. Dưới đây dẫn hai thủ tục hỗ trợ cho việc lập hệ phương trình đại số tuyến tính chuẩn tắc (34) – SUBROUTINE LHPTCT và giải hệ phương trình đó bằng phương pháp loại biến Gauss – SUBROUTINE GAUSS.

```
SUBROUTINE LHPTCT (Y, X, A, N, M)
INTEGER N, M, I, J, K
REAL Y (10000), X (10000, 50), A (0 : 50, 0 : 51)
A (0, 0) = N
DO J = 1, M
    A (0, J) = 0.0
    DO K = 1, N
        A (0, J) = A (0, J) + X (K, J)
    END DO
END DO
A (0, M + 1) = 0.0
DO K = 1, N
    A (0, M + 1) = A (0, M + 1) + Y (K)
END DO
```

```
DO I = 1, M
    A (I, M + 1) = 0.0
    DO K = 1, N
        A (I, M + 1) = A (I, M + 1) + Y (K) * X(K, I)
    END DO
END DO
DO I = 1, M
    DO J = I, M
        A (I, J) = 0.0
        DO K = 1, N
            A (I, J) = A (I, J) + X (K, I) * X (K, J)
        END DO
    ENDDO
ENDDO
DO I = 1, M
    DO J = 0, I - 1
        A (I, J) = A (J, I)
    END DO
END DO
RETURN
END
```

SUBROUTINE GAUSS (M, A, X)

```
INTEGER M
REAL A (0 : 50, 0 : 51), X (0 : 50)
DO I = 0, M - 1
    K = I
    AMAX = ABS (A (K, K))
    DO J = I + 1, M
        R = ABS (A (J, I))
        IF (AMAX .LT. R) THEN
            AMAX = R
            K = J
        END IF
    END DO
    IF (K .NE. I) THEN
        DO J = I, M + 1
            AMAX = A (I, J)
            A (I, J) =A (K, J)
            A (K, J) = AMAX
        END DO
    END IF
    DO J = I + 1, M + 1
        A (I, J) = A (I, J) / A (I, I)
    END DO
    DO J = I + 1, M
        DO K = I + 1, M + 1
            A (J, K) = A (J, K) - A (J, I) * A (I, K)
        END DO
    END DO
END DO
X (M) = A (M, M + 1) / A (M, M)
DO I = M - 1, 0, -1
    X (I) = A (I, M + 1)
    DO J = I + 1, M
        X (I) = X (I) - A (I, J) * X (J)
    END DO
END DO
RETURN
END
```