KS. HOÀNG HỒNG

FOXPRO

HƯỚNG DẪN SỬ DỤNG

LẬP TRÌNH CƠ BẢN VÀ NÂNG CAO

NHÀ XUẤT BẢN GIAO THÔNG VẬN TẢI

KS. HOÀNG HỒNG

FOXPRO

HƯỚNG DẪN SỬ DỤNG

LẬP TRÌNH CƠ BẢN VÀ NÂNG CAO

NHÀ XUẤT BẢN GIAO THÔNG VẬN TẢI

Chịu trách nhiệm xuất bản:

TS. NGUYỄN XUÂN THỦY

Tác giả:

KS. HOÀNG HỒNG

Biên tập:

NGUYỄN BÌNH GIANG

Trình bày:

NGUYỄN KHÁNH NGỌC

In 1000 cuốn khổ 14 cm x 20 cm.

Tại Xưởng in Nhà xuất bản Giao thông vận tải.

Giấy chấp nhận kế hoạch xuất bản số 125/XB-QLXB ngày 01/02/2002.

In xong và nộp lưu chiểu tháng 11/2002.

LỜI NÓI ĐẦU

FOXPRO là một hệ quản trị cơ sở dữ liệu tiên tiến đang được ứng dụng rất rộng rãi trong các hoạt động của đời sống kinh tế, xã hội. Ở bất cứ lĩnh vực nào - sản xuất, thương mại, dịch vụ, nghiên cứu khoa học, kỹ thuật, quản lý, văn hoá xã hội... cũng thấy sự thâm nhập của FOXPRO, và cũng có thể ứng dụng FOXPRO.

Do những ưu việt của mình, nhu cầu sử dụng FOXPRO tăng vọt mang tính đột biến. Điều đó làm nổi bật sự cần thiết phải có tài liệu về FOXPRO mang tính hướng dẫn đi từ vấn đề cơ bản đến nâng cao, đáp ứng được nhu cầu gia tăng về trình độ sử dụng hiện nay. "FOXPRO: Hướng dẫn sử dụng - Lập trình cơ bản và nâng cao" đã được soạn thảo theo định hướng đó, cuốn sách này được chia làm ba phần gồm 21 chương. Phần 1: Lý thuyết FOXPRO. Phần 2: Các ví dụ áp dụng trong chế độ thao tác trực tiếp. Phần 3: Chương trình FOXPRO. Phần 1 gồm 5 chương, giới thiệu các khái niệm cơ bản, các lệnh trong chế độ làm việc trực tiếp để tạo các tệp cơ sở dữ liệu và quản lý dữ liệu. Phần 2 gồm 2 chương, trình bày các ví dụ được áp dụng trong quản lý lương và kế toán. Phần 3 gồm 14 chương, phần này trình bày cấu trúc lập trình với các ví dụ minh họa, các kỹ thuật lập trình từ đơn giản đến phức tạp giúp các bạn nhanh chóng lập được các chương trình ứng dụng trong quản lý. Các chương trình minh họa

đều được viết ngắn gọn để giải quyết một công việc cụ thể và các bạn sẽ không phải tra cứu nhiều khi tìm hiểu nội dung chương trình và có thể áp dụng vào công việc thực tế với sự cải biên tuỳ thuộc vào công việc của bạn.

Đối tượng phục vụ của cuốn sách này là sinh viên các trường đại học, cao đẳng và các bạn muốn sử dụng FOXPRO trong công việc của mình. Tác giả hy vọng cuốn sách này sẽ giúp ích phần nào cho các bạn.

Tác giả xin chân thành cảm ơn KS Nguyễn Bình Giang và Trung tâm tư vấn xuất bản đã quan tâm giúp đỡ và tạo mọi điều kiện thuận lợi để cuốn sách được hoàn thành nhanh chóng, sớm ra mắt bạn đọc.

KS. HOÀNG HỒNG

Phần 1
LÝ THUYẾT FOXPRO

Chương 1
GIỚI THIỆU CHUNG

FOXPRO 2.6 là một hệ thống chương trình quản lý, khai thác và kết nối các tệp cơ sở dữ liệu (CSDL), được xây dựng trên cơ sở của FOXBASE nhưng ưu việt hơn nhiều. Đặc biệt trong việc tạo bảng chọn, xử lý dữ liệu kiểu memo, Foxpro 2.6 đã tỏ ra rất ưu việt. Vì vậy, Foxpro 2.6 có khả năng giúp bạn dễ dàng tạo ra các tệp cơ sở dữ liệu, kết nối, so sánh và lập các chương trình ứng dụng sáng sủa, dễ hiểu nhanh chóng đạt được các yêu cầu đề ra.

1.1. KHỞI ĐỘNG VÀ RA KHỎI FOXPRO

a. Khởi động

FOXPRO có nhiều tệp, nhưng để khởi động được, bạn phải chắc chắn có hai tệp sau:

FOXPRO.EXE và FOXPRO.OVL

Xin bạn lưu ý: Tên tệp FOXPRO.EXE là tên do Công ty Fox Software đặt ra. Tùy theo sở thích hoặc tiện dùng ở mỗi nơi mà tên tệp này có thể thay đổi.

Ví dụ: FOX.EXE, F.EXE v.v... Vì thế, khi sử dụng bạn phải tìm hiểu xem tên tệp này trên đĩa bạn đang sử dụng tên là gì. Dưới đây chúng tôi hướng dẫn theo tên ban đầu của nó.

Trước tiên, bạn phải vào thư mục chứa các tệp của FOXPRO (bạn có thể không cần thực hiện động tác trên nếu các tệp của FOXPRO thuộc thư mục gốc hoặc trong tệp

Autoexec.bat có đường dẫn đến thư mục chứa các tệp làm việc). Giả sử các tệp của FOXPRO trong thư mục FOXPRO 26, từ dấu nhắc của DOS, bạn hãy ra lệnh:

C:\>CD FOXPRO26 (Enter)

và sau đó ra lệnh tiếp:

C:\FOXPRO26>FOXPRO (Enter)

Xin bạn lưu ý: Nếu trên đĩa của bạn có cài đạt cả chương trình FOXPRO đầy đủ thì bạn có thể khởi động bằng các tệp khác như FOX, FOXPROX...

b. Ra khỏi FOXPRO

Trong khung cửa sổ lệnh (khung command) viết lệnh:

QUIT (Enter)

Hoặc ấn phím F10, ấn F và ấn Q. Con trỏ sẽ trở về dấu nhắc của DOS.

1.2. CÁC CHỮ VIẾT TẮT TRONG CUỐN SÁCH NÀY

Chữ viết tắt	Ý nghĩa
bt thức	Biểu thức
bt số	Biểu thức số
ds trường	Danh sách trường
CSDL	Cơ sở dữ liệu
bt logic	Biểu thức logic
bt ngày	Biểu thức ngày
bt ký tự	Biểu thức ký tự

Xin các bạn lưu ý:

Vì một dòng trên trang sách có hạn, nên sẽ có nhiều dòng lệnh dài hơn. Ở cuối một số dòng lệnh chúng tôi có dùng dấu

chấm phẩy để thông báo cho Foxpro biết dòng lệnh này được nối với dòng dưới. Khi thao tác trên máy, các bạn có thể viết như chúng tôi hoặc viết các lệnh này trên cùng một dòng (lúc này không có dấu chấm phẩy).

1.3. GIỚI THIỆU HỆ THỐNG BẢNG CHỌN

a. Bảng chọn ngang (Menu bar)

Sau khi khởi động FOXPRO, hàng chữ sau được gọi là bảng chọn ngang (hay còn gọi là bảng chọn chính - main menu).

System File Edit Database Record Program Window

Bảng chọn ngang biểu thị các chức năng chính của FOXPRO. Để chọn một chức năng nào đó, bạn hãy làm một trong các cách như sau (giả sử con trỏ đang ở trong khung cửa sổ lệnh).

- Ấn phím F10. Khi này trên bảng chọn ngang xuất hiện một hộp sáng. Dùng phím ← →, dịch chuyển hộp sáng đến bảng chọn cần chọn, hoặc bạn có thể ấn vào các chữ cái có màu khác (hoặc sáng hơn) trong bảng chọn cần chọn.

Ví dụ:

Chọn bảng chọn File hãy ấn vào phím F, chọn bảng chọn System ấn phím S.

- Ấn ALT + Phím có màu khác (hoặc sáng hơn) trong bảng chọn cần chọn.

Ví dụ: ALT + F, ALT + S.

Nếu bạn muốn thoát khỏi bảng chọn ngang để con trỏ trở

về khung cửa sổ lệnh, hãy ấn phím ESC.

b. Bảng chọn dọc

Trong bảng chọn ngang có nhiều chức năng và mỗi chức năng này lại có nhiều chức năng con. Để xem hoặc chọn các chức năng con của các chức năng trong bảng chọn ngang, bạn hãy vào bảng chọn ngang, chọn chức năng cần thiết và ấn phím Enter.

Bạn sẽ thấy chức năng con của chức năng này trải dọc ở phía dưới (chúng tôi tạm gọi là bảng chọn dọc).

Để chọn một chức năng trong bảng chọn dọc, bạn hãy làm tiếp một trong hai cách sau:

- Dùng phím mũi tên, dịch chuyển hộp sáng đến bảng chọn cần chọn.

- Ấn vào các chữ khác màu (hoặc sáng hơn) trong các chức năng cần chọn.

Đôi khi trong bảng chọn dọc cũng có một vài chức năng sáng lờ mờ. FOXPRO thông báo để bạn biết các chức năng này không thể chọn được.

Trong bảng chọn dọc có một số chức năng có ghi bên phải một số phím. Ví dụ: trong mục chọn System có chức năng Help. Bên phải chức năng Help có ghi phím F1. Nếu con trỏ đang ở khung cửa sổ lệnh, thay cho việc bạn phải thao tác F10, S, H, bạn có thể ấn phím F1 để đọc hướng dẫn sử dụng về FOXPRO.

c. Khung cửa sổ lệnh (Command Window)

Bình thường khung này nằm ở góc dưới phía phải. Các

lệnh để Foxpro thực hiện được đưa vào trong khung này. Foxpro lưu vào trong bộ nhớ các lệnh mà bạn đã chỉ thị.

Khi cần dùng lại lệnh cũ, hãy dùng phím mũi tên đưa con trỏ đến lệnh cần thiết và ấn Enter thay cho việc phải viết lại lệnh này.

Kích thước của khung cửa sổ lệnh cũng có thể được thay đổi cho phù hợp bằng cách: ấn F10, ấn W, chọn Size, dùng phím mũi tên để thay đổi kích thước, ấn Enter khi đã phù hợp. Cũng có thể xê dịch khung cửa sổ bằng cách ấn Ctrl-F7, dùng các phím mũi tên để di chuyển và cuối cùng ấn Enter.

Bạn có thể cất dấu khung cửa sổ để màn hình được rộng, tiện lợi quan sát các thông tin trong trường hợp cần thiết bằng cách ấn F10, ấn W, ấn H. Muốn lấy lại khung cửa sổ lệnh ấn, Ctrl+F2.

1.4. MỘT SỐ ĐIỀU CẦN LƯU Ý KHI SỬ DỤNG FOXPRO 2.6

Foxpro 2.6 bao gồm hai version: Foxpro for Dos và Foxpro for Windows. Bạn có thể sử dụng hoàn toàn những gì trình bày trong cuốn sách này cho Foxpro for Dos. Song đối với Foxpro for Windows có một vài điểm khác xin bạn vui lòng tìm hiểu trong mục: "Một số điểm cần lưu ý khi sử dụng Foxpro 2.6 for Windows" của mục này.

a. Yêu cầu cần thiết khi sử dụng Foxpro 2.6

Foxpro for Dos đòi hỏi máy IBM PC (hoặc tương thích) với 640 KB RAM. Dùng chuột rất thuận tiện nhưng không bắt buộc.

Foxpro for Windows đòi hỏi phải có ít nhất máy 80386 SX.

chuột. Microsoft Windows 3.0 (hoặc các Version cao hơn) và ít nhất 4 MB bộ nhớ RAM.

b. Một số điểm cần lưu ý khi sử dụng Foxpro 2.6 for Windows

(1). *Bảng chọn chính*

Khi bạn khởi động Foxpro 2.6 for Windows, bảng chọn chính có dạng như sau:

File Edit Database Program Run Text Window Help

Như bạn đã thấy ở bảng chọn trên, trong Foxpro 2.6 For Dos không có bảng chọn Text. Bảng chọn này dùng để chọn phông chữ, thay đổi kích cỡ của chữ và định khoảng cách giữa các hàng.

Trong mục chọn Window cũng có một số điểm khác Foxpro for Dos nhưng không quan trọng. Chức năng Help tương đương với chức năng System trong Foxpro for Dos.

(2). *Sử dụng các hộp đối thoại*

Bạn có thể sử dụng các hộp đối thoại của Foxpro 2.6 bằng chuột hoặc bàn phím. Bạn phải chọn hai cách gõ khác nhau khi cài đặt: Foxpro tương thích Dos hay tương thích Windows. Cách gõ phím tương thích với Windows cũng giống như các ứng dụng khác trên Windows. Cách gõ phím tương thích với Dos thì giống như đã dùng cho các phiên bản trước của Foxpro.

Một số điểm khác biệt của hai cách gõ này.

STT	Mục đích	Cách gõ Windows	Cách gõ DOS
1	Chọn nhiều mục	Ctrl+nháy chuột hoặc Ctrl+SpaceBar	Shift + nháy chuột hoặcShift+SpaceBar
2	Chọn nút mặc nhiên	Enter	Ctrl+Enter
3	Dùng phím nóng	Alt+phím nóng	Phím nóng
4	Chọn danh sách trái xuống	SpaceBar + Alt hay Alt + ↓	SpaceBar Enter

(3). *Khi soạn thảo*

Phím Del dùng để xoá vị trí phía trái con trỏ (nếu dùng Foxpro for Dos thì xoá vị trí con trỏ). Có điểm khác nhau này là con trỏ của Foxpro for Dos là một điểm sáng trùng lên ký tự, còn con trỏ của Windows là một đường thẳng nằm giữa hai ký tự.

1.5. SỬ DỤNG TIẾNG VIỆT TRONG FOXPRO

Hiện nay có nhiều chương trình có khả năng cài đặt phông chữ tiếng Việt trong FOXPRO, nhưng có lẽ thông dụng nhất là ViệtRes (Vietnamese Terminate and Stille Resident) - sản phẩm của Công ty Seatic, thành phố Hồ Chí Minh. Dùng ViệtRes có ưu điểm là: các thông báo trên màn hình, các kết quả in ra trên máy in hoàn toàn bằng tiếng Việt với các phông chữ khác nhau.

a. Nạp ViệtRes vào FOXPRO

Trước hết chúng ta cần chú ý là ViệtRes có hai chức năng:

soạn thảo và lồng vào các phần mềm khác (ví dụ như FOXPRO, QUATTRO...). Ở đây chúng ta chỉ quan tâm đến chức năng thứ hai.

Để khởi động được VietRes, bạn phải có các tệp của VietRes gồm: VRD.COM, VR.HPL, VRE.COM, VRLNDE.MEM... Nếu bạn không cần xem hướng dẫn sử dụng, không cần sắp xếp và tìm kiếm các chữ tiếng Việt thì chỉ cần tệp VRD.COM là đủ. Giả sử các tệp này nằm trong thư mục chứa FOXPRO. Có hai cách khởi động:

- *Cách 1*: Nạp VietRes trước khi khởi động FOXPRO.

+ Tại dấu nhắc của DOS, dùng lệnh:

C:\FOXPRO\VR/0

+ Sau đó khởi động FOXPRO như chúng tôi đã trình bày ở trên.

- *Cách 2*: Khởi động VietRes sau khi đã khởi động Foxpro.

+ Tại khung cửa sổ lệnh, viết lệnh:

!VR/0

+ Lệnh trên cũng có thể được viết trong chương trình nếu bạn muốn khi thực hiện chương trình thì VietRes được tự động nạp.

Sau khi VietRes được nạp, (chiếm khoảng 36 KB bộ nhớ trong) bạn có thể viết các chương trình tạo ra các biểu bảng bằng tiếng Việt, tạo ra các tệp CSDL có nội dung được Việt hoá hoàn toàn.

Xin các bạn lưu ý:

Không nên dùng chức năng soạn thảo trong FOXPRO vì cần đến dung lượng bộ nhớ trong khá lớn. Vì vậy rất hay gặp

lỗi khi thực hiện chương trình. Nghĩa là bạn không được khởi động VietRes với lệnh:

VR

hoặc

!VR (không có tham số /0)

b. Một số điều cần lưu ý khi sử dụng

- Để bật tắt VietRes, ấn tổ hợp phím Ctrl và phím - (phím - ở bên phải bàn phím). Ví dụ như khi dùng lệnh Browse bạn phải tắt chế độ gõ tiếng Việt rồi sau đó khởi động lại. Nếu không, bạn sẽ được chữ Bróe.

- Có nhiều cách gõ tiếng Việt trong VietRes, nhưng thông dụng nhất là cách gõ theo kiểu TELEX (là trạng thái ngầm định). Cách gõ TELEX như sau:

ô=oo ơ=ow đ=dd ă=aw

ê=ee ư=aw â=aa ư=w

Các dấu như sau:

huyền=f, sắc=s, hỏi=r, ngã=x, nặng=j

Ví dụ: khi cần chữ cộng hoà hãy gõ như sau: coongj hoaf

- Khi in ấn các kết quả, bạn phải khai báo cấu hình tương thích với máy in của bạn, bằng cách: ấn F10, chọn "in ấn", chọn "đạt tham số in"...

- Trong trường hợp khởi động VietRes trước khi khởi động FOXPRO thì bạn không thể loại bỏ VietRes ngay trong FOXPRO được mà bạn phải thoát ra khỏi FOXPRO sau đó mới loại bỏ VietRes. Song đối với trường hợp khởi động VietRes sau khi đã khởi động FOXPRO thì bạn có thể loại bỏ VietRes trước hoặc sau khi ra khỏi FOXPRO cũng được.

- Khi bạn tạo ra các bảng chọn hoặc các cửa sổ (Windows) thì các nét kẻ khung xuất hiện một số ký tự lạ.

c. Tiếng Việt thể hiện trên màn hình và in ra giấy

- Trên màn hình sẽ đảm bảo thể hiện đầy đủ các chữ thường có dấu. Song một số chữ hoa có dấu thì không thể hiện được.

- Khi in ra giấy, bạn có thể in ra các chữ hoa, các chữ thường với nhiều phông chữ khác nhau.

Ngoài việc in ấn trong chế độ soạn thảo, có thể điều khiển trong chương trình để chọn các phông chữ khi in ra máy in như bảng sau:

Mã	Ý nghĩa
CHR(27)+CHR(27)+"V"	Mở chế độ in chữ Việt
CHR(5)	Chữ hoa có dấu
CHR(1)	Chữ cao, rộng gấp đôi
CHR(23)	Chữ cao gấp đôi
CHR(4)	Chữ nghiêng
CHR(31)	Chữ rất nhỏ
CHR(6)	Chữ gạch chân
CHR(29)	Chữ rộng gấp đôi

d. Sắp xếp và tìm kiếm tiếng Việt

- Sắp xếp theo kiểu chỉ số (INDEX)

Để sắp xếp một trường theo một trật tự nào đó mà trường đó có chứa các ký tự tiếng Việt (ví dụ như Họ và tên...), bạn phải có tệp VRLNDE.MEM do Seatic cung cấp. Nếu không

bạn chỉ có thể sắp xếp theo các trường không chứa các ký tự tiếng Việt (ví dụ như: ngày sinh, lương, số lượng hàng hoá...).

Để sắp xếp theo kiểu chỉ số, bạn dùng lệnh như sau:

SYS(15,VR,<tên trường>)

trong đó:

<tên trường> là trường kiểu Character

VR là biến chứa trật tự tiếng Việt mà ViệtRes đã dùng. Biến này phải khai báo thông qua tệp VRLNDE.MEM.

Qui ước khi sắp xếp tiếng Việt như sau:

+ Không dấu
+ Huyền
+Hỏi - Ngã - Sắc
+ Nặng

- Tìm kiếm theo kiểu chỉ số

Để tìm kiếm theo kiểu chỉ số một trường nào thì trường ấy đã được sắp xếp theo kiểu chỉ số và dùng lệnh:

=SEEK (SYS(15,VR,<bthức>))

trong đó: <bthức> là giá trị cần tìm kiếm trên trường đã được sắp xếp.

Ví dụ:

```
USE HOSO
* Lệnh dưới dây sắp xếp theo kiểu chỉ số trường TEN
* (tên)
INDEX ON SYS(15,VR,TEN) TO SS
USE HOSO INDEX SS
=SEEK (SYS(15,VR,"Vân Trinh"))
?HODEM,TEN,LUON
```

Chương 2

TẠO TỆP CSDL, XEM THÔNG TIN, CẤU TRÚC TỆP CSDL

2.1. TẠO TỆP CSDL BẰNG LỆNH CREATE

Tại khung cửa sổ lệnh, hãy ra lệnh:

CREATE <tên tệp>

<tên tệp> được ghi theo qui định của Dos và không cần ghi phần mở rộng. Foxpro tự gán phần mở rộng là DBF. Nếu tệp cần tạo không thuộc thư mục hiện thời, bạn phải chỉ thị đường dẫn. Trên màn hình sẽ xuất hiện một khung để khai báo cấu trúc tệp CSDL. Phải khai báo: Field name (tên trường), Type (dạng dữ liệu trường), Width (độ rộng của trường), Dec (số thập phân nếu là trường số). Foxpro nhận các dạng dữ liệu sau:

- Character (ký tự): dữ liệu dạng này có thể chứa chữ cái, chữ số, các ký tự đặc biệt...

- Date (ngày): lưu trữ dữ liệu dạng ngày. Dạng ngầm định là: MM/DD/YY (MM: tháng, DD: ngày, YY: năm), có thể dùng lệnh SET DATE để chọn dạng ngày khác.

- Numeric (số): độ rộng dạng này ngầm định là 10. Độ rộng tối đa là 20 (kể cả vị trí cho dấu chấm thập phân).

- Float (kiểu số di động): đây là dữ liệu dạng số nhưng với dấu chấm thập phân di động (Floating Point) được viết theo

kiểu khoa học.

- Logical (logic): trường dạng này chỉ chứa một trong hai chữ cái: T (True -đúng) và F (False - sai). Độ rộng trường dạng này luôn là một. Khi viết trong các câu lệnh, in ra giấy... dạng dữ liệu này có thêm hai dấu chấm ở trước và sau. Cụ thể là: .T. và .F.

- Memo (ghi nhớ). Đây là một dạng dữ liệu rất đặc biệt để Foxpro lưu trữ các khối văn bản lớn có độ dài không cố định như tiểu sử cán bộ, lý lịch của một loại máy..

- General: dùng để chứa các bảng tính, hình ảnh hoặc âm thanh (được tạo trong các ứng dụng Windows khác).

Khi con trỏ vào vùng Type, phải chọn dạng dữ liệu cho trường. Hãy ấn phím SPACE BAR để hiện lên danh sách các dạng dữ liệu. Đưa vệt sáng đến dạng cần chọn và ấn Enter, hoặc bạn có thể thao tác theo cách sau: ấn vào chữ cái đầu của dạng dữ liệu cần chọn. Ví dụ ấn N để chọn Numeric, ấn vào C để chọn Character, .

Sau khi khai báo xong cấu trúc, hãy ấn Ctrl+W để ghi lại. Foxpro sẽ hỏi:

Input data records now (Y/N)?

<<Yes>> <No>

Nếu trả lời N, Foxpro chấm dứt lệnh tạo tệp CSDL, ngược lại, trả lời Y sẽ hiện lên màn hình để vào dữ liệu. Sau khi vào dữ liệu xong, hãy ấn Ctrl+W để ghi lại.

Có thể tạo tệp CSDL bằng cách sử dụng bảng chọn sau:

- Ấn F10, chọn File, chọn New, chọn Database.

Xin bạn lưu ý:

Đối với dữ liệu dạng memo, khi vào dữ liệu, bên cạnh tên trường có xuất hiện chữ memo và con trỏ nhấp nháy ở chữ này. Hãy ấn Ctrl+Home hoặc Ctrl+PgUp hoặc Ctrl+PgDn. Trên màn hình sẽ xuất hiện một màn hình soạn thảo để vào dữ liệu cho trường này. Sau khi vào dữ liệu xong, hãy ấn Ctrl+W.

2.2. MỞ TỆP, ĐÓNG TỆP CSDL

Một điều mong bạn luôn ghi nhớ là muốn làm việc với một tệp CSDL phải mở nó ra trước khi chỉ thị các yêu cầu cần thiết.

a. Mở tệp CSDL

Lệnh USE được dùng với nhiều chức năng và công dụng khác nhau, chúng ta sẽ tìm hiểu dần. Bây giờ chúng ta dùng với trường hợp đơn giản nhất là mở một tệp CSDL. Tại khung cửa sổ lệnh, bạn hãy ra lệnh.

USE <tên tệp>

trong đó <tên tệp> là tên tệp cần mở và không cần phải chỉ thị phần mở rộng.

Ví dụ:

USE K_HANG

Ở đây chúng ta mới làm quen với việc mở một tệp CSDL. FOXPRO cho phép mở đồng thời 25 tệp CSDL trên 25 vùng khác nhau .

b. Đóng tệp CSDL

Để đóng một tệp CSDL trong vùng làm việc hiện thời, tại khung cửa sổ lệnh, hãy ra lệnh:

USE

Chú ý: Sau lệnh USE không ghi gì cả.

Có thể dùng lệnh:

- CLOSE ALL: đóng tất cả các loại tệp đang mở
- CLOSE DATABASES: đóng tất cả các tệp CSDL

2.3. XEM THÔNG TIN

a. Phạm vi và điều kiện

- *Phạm vi*: xác định các bản ghi mà một lệnh Foxpro tác động đến. Phạm vi làm việc (trong các câu lệnh mà bạn tìm hiểu dưới đây được viết là <phạm vi>), sẽ nhận các giá trị sau:

+ NEXT <n>: làm việc với <n> bản ghi từ bản ghi hiện thời. Khi mở tệp, bản ghi hiện thời là bản ghi đầu tiên.

Khi xử lý tệp đến bản ghi nào thì bản ghi đó là bản ghi hiện thời.

+ ALL: làm việc với tất cả các bản ghi

+ RECORD <n>: làm việc với bản ghi thứ <n>

+ REST: làm việc với các bản ghi từ bản ghi hiện thời đến bản ghi cuối cùng.

- *Điều kiện*:

+ Mệnh đề: FOR <bt logic>

Các câu lệnh chỉ có hiệu lực đến các bản ghi thỏa mãn điều kiện xác định bởi <bt logic> (khi <bt logic> nhận giá trị .T. đúng)

+ Mệnh đề: WHILE <bt logic>.

Các câu lệnh chỉ có hiệu lực trên các bản ghi khi nào <bt

logic> còn nhận giá trị .T. (đúng): khi gặp bản ghi đầu tiên có <bt logic> nhận giá trị .F. (sai) câu lệnh sẽ kết thúc bất kể các bản ghi tiếp theo <bt logic> nhận giá trị đúng hay sai.

Xin bạn lưu ý:

Khi trong cùng một câu lệnh cùng xuất hiện FOR hoặc WHILE thì WHILE được ưu tiên hơn, nghĩa là nếu <bt logic> viết sau FOR là đúng, nhưng <bt logic> viết sau WHILE là sai thì vẫn kết thúc câu lệnh.

b. Lệnh ?, ??

Ví dụ:

```
USE K_HANG
? TIEN
? TIEN*10
```

Kết quả:

```
123000
1230000
```

```
??TIEN
??TIEN*10
```

Kết quả:

```
123000 1230000
```

Kết quả khi thực hiện lệnh ? luôn xuất hiện ở đầu dòng dưới (thực chất là xuống dòng) còn kết quả khi thực hiện lệnh ?? xuất hiện tại vị trí con trỏ tức là in tiếp trên cùng một dòng. Lệnh ? và ?? chỉ khác nhau ở điểm này.

Ngoài chức năng trên, lệnh ?, ?? còn thực hiện việc đưa kết quả ra màn hình hoặc máy in một phép tính số học, logic, giá trị của hàm v.v...

c. Lệnh DISPLAY

Cú pháp:

DISPLAY FILES [ON <đường dẫn>] [LIKE <tên tệp>]
[<TO PRINTER>]

Ví dụ:

DISPLAY FILES LIKE *.FXP

Lệnh trên hiện tất cả các tệp chương trình FOXPRO (*.FXP) trong thư mục hiện thời.

DISPLAY ON C:

Hiện danh sách của tất cả các tệp CSDL (*.DBF) trên ổ đĩa C.

Xin bạn lưu ý: Đặc điểm của lệnh DISPLAY là đưa kết quả ra màn hình theo từng trang 15 dòng một (nếu kết quả dài hơn một trang màn hình), bạn muốn xem tiếp phải ấn một phím bất kỳ.

d. Lệnh DIR

Cú pháp

DIR <tên tệp>

Trong <tên tệp>, bạn cũng có thể dùng ký tự ?, * để đọc một nhóm tệp.

Nếu bạn không chỉ thị <tên tệp>, thì FOXPRO sẽ đưa ra danh sách các tệp CSDL trong thư mục hiện thời.

Ví dụ:

DIR

e. Lệnh List

Được dùng tương tự như lệnh DISPLAY, nhưng có một số

điểm khác nhau như sau:

- Lệnh DISPLAY: Nếu không chỉ thị <phạm vi> thì FOXPRO hiểu là làm việc với một bản ghi hiện thời, còn lệnh LIST thì lại coi là làm việc với tất cả các bản ghi.

- Lệnh DISPLAY: Nếu số dòng kết quả lớn hơn một trang màn hình thì hiện theo từng trang, muốn xem tiếp ấn một phím bất kỳ, còn lệnh LIST chỉ dừng lại ở trang cuối.

Ví dụ:

```
LIST FOR PHUCAP>100000 FIELDS HOTEN,
SOTIEN OFF
```

2.4. LÀM VIỆC VỚI CẤU TRÚC TỆP CSDL

a. Xem cấu trúc

Bạn có thể xem cấu trúc bằng một trong hai lệnh sau:

```
LIST STRUCTURE [TO PRINTER] [TO FILE <tên tệp>]
DISPLAY STRUCTURE [TO PRINTER]
                  [TO FILE <tên tệp>]
```

Hai lệnh trên, nếu không có các tham số, đưa cấu trúc tệp CSDL đang mở ra màn hình. Nếu có chỉ thị TO PRINTER, đưa kết quả ra máy in. Nếu có chỉ thị TO FILE <tên tệp>, đưa kết quả ra tệp văn bản có tên là <tên tệp>.

b. Sửa cấu trúc

Bạn có thể thay đổi tên trường, dạng trường, kích thước, bổ sung, xoá trường hoặc thay đổi vị trí của trường.

Tại khung cửa sổ lệnh, bạn hãy ra lệnh:

```
MODIFY STRUCTURE
```

- Chèn thêm trường mới: đưa con trỏ đến đầu trường mà

bạn muốn chèn lên phía trên một trường mới. Ấn Alt+T, trên màn hình xuất hiện một khung đối thoại:

Insert Fields ^I

Delete Fields ^D

Bạn hãy ấn Enter (vì vệt sáng đang ở vị trí Insert Fields) hoặc bạn có thể làm tắt bằng cách ấn Ctrl+I.

- Xóa một trường: đưa con trỏ về đầu trường cần xoá, ấn Alt+T, chọn Delete Fields hoặc đưa con trỏ về đầu trường cần xoá, ấn Ctrl+D.

- Dịch chuyển trường: đưa con trỏ về đầu trường cần dịch chuyển, ấn phím Ctrl+PgUp hoặc Ctrl+PgDn dịch chuyển trường đến vị trí cần dịch chuyển.

- Sửa đổi nội dung một trường: đưa con trỏ đến trường cần sửa đổi và thay đổi các giá trị cần thiết.

Xin bạn lưu ý:

- Nếu kiểu dữ liệu của một trường sau khi sửa đổi phù hợp với dữ liệu của tệp cũ thì dữ liệu của tệp cũ được chuyển sang tệp mới. Nếu không phù hợp sẽ bị xóa bỏ.

- Nếu một trường mà bạn vừa thay đổi tên trường và kiểu trường hoặc độ rộng thì FOXPRO coi đây là trường hoàn toàn mới và hủy bỏ hết dữ liệu của trường này. Trong trường hợp này, bạn phải thực hiện hai lần sửa cấu trúc như sau:

+ Lần 1: sửa đổi mọi thay đổi của trường, nhưng không sửa tên trường và ghi lại.

+ Lần 2: sửa tên trường.

- Nếu tệp CSDL của bạn có trường Memo thì tệp lưu của tệp Memo .FPT sẽ đổi thành .TBK.

Nhìn chung, sau khi bạn sửa lại cấu trúc, bạn nên kiểm

tra lại dữ liệu của tệp bằng các lệnh: USE, DISPLAY hoặc BROWSE.

Ví dụ:

```
USE K_ HANG
MODIFY STRUCTURE
```

c. Lệnh sao chép cấu trúc tệp: COPY STRUCTURE

Lệnh này sao chép cấu trúc của một tệp đang mở sang một tệp mới có cấu trúc giống như tệp đang mở hoặc chỉ có một số trường trong cấu trúc tệp đang mở.

Cú pháp

```
COPY STRUCTURE <tên tệp mới>
                    [FIELDS <ds trường>]
```

trong đó: <tên tệp mới> là tên tệp cần tạo ra. Nếu bạn không chỉ thị FIELDS <ds trường> thì toàn bộ cấu trúc của tệp đang mở được sao sang tệp mới. Nếu có chỉ thị này thì chỉ các trường được liệt kê trong <ds trường> mới được sao sang tệp mới.

Ví dụ:

```
USE K_ HANG
COPY STRUCTURE TO TG:
                    FIELDS HOTEN.SOTIEN.LAI
```

d. Lệnh CREATE FROM

Cú pháp

```
CREATE [<tệp cần tạo>]
FROM [<tệp lưu cấu trúc>]
```

Tệp được tạo có cấu trúc là nội dung của các bản ghi của <tệp lưu cấu trúc>

Xin bạn lưu ý: trong chế độ trực tiếp, tức là dùng lệnh ở khung cửa sổ lệnh thì hai lệnh trên (c.d) có vẻ chẳng hay ho gì, nhưng ở chế độ gián tiếp, tức là lập chương trình bằng FOXPRO, thì hai lệnh trên vô cùng hữu hiệu. Bạn hãy tìm hiểu tiếp vấn đề này ở các chương sau.

2.5. CÁC LỆNH SỬA ĐỔI BẢN GHI

a. Các lệnh định vị con trỏ bản ghi

(1). *Lệnh GOTO*

Cú pháp

GOTO <bt số học>

hoặc

GO <bt số học>

trong đó: < bt số học> phải là số nguyên dương i nào đó. Lệnh này định vị con trỏ bản ghi đến bản ghi thứ i.

Ví dụ:

```
USE K_HANG
GO 5
? RECNO()
```

Kết quả:

5

Nếu i lớn hơn tổng số bản ghi của tệp CSDL đang mở, thì FOXPRO sẽ thông báo cho bạn biết:

Record out of range

(bản ghi ngoài khoảng)

Như vậy bạn đã viết lệnh sai.

(2). *Lệnh SKIP*

Cú pháp

SKIP <bt số học>

trong đó: <bt số học> phải là số nguyên i.

- Nếu i>0, con trỏ bị dịch chuyển về phía cuối tệp i bản ghi tính từ bản ghi hiện thời.

- Nếu i<0, con trỏ bị dịch chuyển về phía đầu tệp CSDL |i| (giá trị tuyệt đối của i) bản ghi tính từ bản ghi hiện thời.

- Nếu không có <bt số học> thì ngầm định là SKIP 1, nghĩa là tăng lên 1 bản ghi.

Ví dụ:

```
USE K_HANG
SKIP 3
? RECNO()
```

Kết quả:

```
4
SKIP - 2
? RECNO()
```

Kết quả: 2

b. Lệnh EDIT

Cú pháp

EDIT [<phạm vi>] [FIELDS <ds trường>]
[FOR <bt logic>] [WHILE <bt logic>]

Ví dụ:

```
EDIT FIELDS HOTEN, TIEN
```

Lệnh này chỉ thị cho FOXPRO hiện lên màn hình để sửa đổi chỉ các trường: HOTEN,TIEN.

c. Lệnh CHANGE

Cú pháp

CHANGE [<phạm vi>] [FIELDS <ds trường>]
[FOR <bt logic>] [WHILE <bt logic>]

Trong thực tế lệnh EDIT và lệnh CHANGE là giống nhau.

d. Lệnh BROWSE

- Lệnh BROWSE tại khung cửa sổ lệnh.

Cú pháp

BROWSE [FIELDS <ds trường>]
[FOR <bt logic 1>] [WHILE <bt logic 2>]
[[WINDOW <tên window 1>]
[IN WINDOW <tên window 2>/ IN
SCREEN]]
[TITLE <bt ký tự 1>]
[LPARTITION] [PARTITION <bt số 1>]
[LEDIT] [REDIT]
[LOCK <bt số 2>]
[KEY <btthức 1>[.<btthức 2>]]
[VALID <bt logic 3>] [ERROR <bt ký tự 3>]
[WIDTH <bt số 4>]
[FREEZE <tên trường>]
[NOAPPEND] [NODELETE]

Browse có ưu điểm hơn hẳn so với Display và List là cho hiện lên màn hình 14 bản ghi theo chiều đứng và một số trường theo chiều ngang. Mỗi bản ghi có thể được đưa ra trên một dòng để bạn có thể xem, sửa đổi một cách dễ dàng, thuận tiện.

Ví dụ: bạn có tệp CSDL quản lý lương của cán bộ, có cấu

trúc như sau:

Field	Field name	Type	Width	Dec
1	MA	Character	4	
2	HOTEN	Character	20	
3	DONVI	Character	4	
4	NGAYSINH	Date	8	
5	LUONG	Numeric	6	
6	PHUCAP	Numeric	6	
7	THUONG	Numeric	6	

với dữ liệu như sau:

MA	HOTEN	DONVI	NGAYSINH	LUONG	PHUCAP	THUONG
001	Trần Minh Hiếu	KH	17/04/62	600000	60000	200000
002	Trần Thu Hiền	KT	16/03/56	600000	70000	120000
003	Hoàng Huy Đông	HC	13/05/58	140000	64500	110000
004	Trần Minh Châu	VT	12/06/60	110000	68000	105000
005	Hoàng Khánh	KT	10/11/64	100000	55000	900000
006	Vũ Hoàng Triệu	HC	12/12/65	900000	50000	870000
007	Vũ Đường	KT	16/10/66	870000	48000	700000

Các tham số đi cùng với lệnh Browse được giải thích như sau:

(1). Khi bạn ra lệnh Browse, khung màn hình (dưới đây chúng tôi gọi là cửa sổ Browse) có thể hiện lên màn hình chính hoặc trên một khung cửa sổ tuỳ theo bạn sử dụng từ khoá IN SCREEN hoặc IN WINDOW <tên window>.

- IN SCREEN, là giá trị ngầm định, lúc này các dữ liệu sẽ hiện lên màn hình chính.

- IN WINDOW <tên window 2>, nếu bạn có chỉ thị này thì dữ liệu của tệp CSDL được đưa ra cửa sổ có tên là <tên

window 2>. Trong đó cửa sổ <tên window 2> phải được khai báo trước bằng lệnh: DEFINE WINDOW. Cửa sổ <tên window 2> được gọi là cửa sổ mẹ. Nếu cửa sổ mẹ di chuyển thì cửa sổ Browse cũng di chuyển theo và cửa sổ Browse không thể di chuyển ra ngoài cửa sổ mẹ và cũng không mở rộng kích thước lấp lên cửa sổ mẹ. Do đó, nếu bạn dùng từ khoá:

- WINDOW <tên window 1> thì cửa sổ Browse sẽ mang đặc tính của cửa sổ có tên là <tên window 1> với điều kiện là bạn đã khai báo cửa sổ <tên window 1>.

(2). Từ khoá TITLE <bt ký tự 1> dùng để thay thế tiêu đề ngầm định thành tiêu đề của bạn.

Ví dụ:

Browse TITLE 'Sử dụng lệnh Browse'

Kết quả trên màn hình:

System File Edit Database Record Program Window Browse

Sử dụng lệnh Browse.

MA	HOTEN	DONVI	NGAYSINH	LUONG	PHUCAP	THUONG
001	Trần Minh Hiếu	KH	17/01/62	600000	60000	200000
002	Trần Thu Hiền	KT	16/03/56	600000	70000	120000
003	Hoàng Huy Đông	HC	13/05/58	140000	61500	110000
004	Trần Minh Châu	VT	12/06/60	110000	68000	105000
005	Hoàng Khánh	KT	10/11/61	100000	55000	900000
006	Vũ Hoàng Triệu	HC	12/12/65	900000	50000	870000
007	Vũ Đường	KT	16/10/66	870000	48000	700000

(3). PARTITION <bt số 1>. Theo giá trị mặc nhiên thì cửa sổ Browse hiện lên chiếm cả màn hình. Song bạn có thể phân cửa sổ thành hai khu vực bằng từ khoá PARTITION <bt số

1>, trong đó <bt số 1> sẽ quy định khu vực bên trái chiếm <bt số 1> cột, còn lại là khu vực bên phải là cửa sổ Browse bình thường.

Ví dụ:

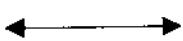
USE LUONG

browse partition 15

Cửa sổ Browse sẽ hiện lên như sau:

System File Edit Database Record Program Window Browse

MA	HOTEN	DONVI	NGAYSINH	LUONG	PHUCAP	THUONG
001	Trần Minh Hiếu	KH	17/04/62	600000	60000	200000
002	Trần Thu Hiền	KT	16/03/56	600000	70000	120000
003	Hoàng Huy Dũng	HC	13/05/58	140000	64500	110000



15 vị trí

Bạn hãy ấn Ctrl+H để di chuyển con trỏ qua lại giữa hai khu vực. Theo quy ước ngầm định thì hai khu vực này sẽ liên kết chặt chẽ với nhau, nghĩa là nếu vệt sáng ở khu vực này di chuyển thì khu vực bên kia cũng di chuyển theo. Bạn có thể xoá bỏ chế độ này bằng từ khoá NOLINK hoặc ấn Alt+B để cho menu Browse hiện lên, chọn Unlink Partitions. Nếu bạn muốn đặt lại chế độ liên kết giữa hai khu vực, ấn Alt+B, sau đó chọn Link Partitions trên bảng chọn dọc Browse.

Dữ liệu của Browse thường được trình bày theo hai dạng sau:

- *Dạng 1*: dạng Change

System File Edit Database Record Program Window Browse

MA	001
HOTEN	Trần Minh Hiếu

DONVI KH
NGAYSINH 17/04/62
LUONG 6000000
PHUCAP 60000
. THUONG 2000000

- *Dạng 1*: dạng Browse, đây là chế độ ngầm định. Các bản ghi được hiện lên theo chiều dứng. Các trường của các bản ghi hiện theo chiều ngang (bạn hãy xem mục từ khoá TITLE). Như vậy trong cửa sổ Browse, bạn có thể cho hiện dữ liệu trên toàn cửa sổ (một khu vực) hoặc phân thành hai khu vực mà mỗi khu vực sẽ được chọn một dạng nào đó tuỳ theo ý đồ của bạn.

(4). Tham số LPARTITION. Theo trạng thái ngầm định khi bạn dung lệnh partition để chia màn hình ra làm hai phần thì con trỏ sẽ ở phần màn hình bên phải. Bạn có thể thay đổi trạng thái này bằng tham số lpartition để di chuyển con trỏ sang phần màn hình bên trái ngay sau khi ra lệnh Browse.

Ví dụ:

use LUONG
browse lpartition partition 20

(5). Tham số LEDIT và REDIT

Nếu bạn không đưa các tham số trên khi phân màn hình thành hai phần thì dữ liệu được hiển thị theo dạng Browse. Nếu bạn thêm tham số LEDIT thì dữ liệu ở phần màn hình bên trái được hiển thị theo dạng Change. Nếu bạn chỉ thị

REDIT thì dữ liệu ở phần màn hình bên phải được hiển thị theo dạng Change.

Ví dụ:

```
use LUONG
browse lpartition partition 20 EDIT
```

(6). Từ khóa FIELDS <ds trường>. Khi không có chỉ thị này thì tất cả các trường được hiện lên trên cửa sổ Browse. Khi có chỉ thị này, chỉ các trường được liệt kê trong <ds trường> mới được hiện lên để xem, sửa. Các trường trong <ds trường> được viết cách nhau một dấu phẩy. Điều này rất quan trọng khi bạn sử dụng một tệp CSDL có nhiều trường trong khi đó chỉ cần xem, sửa một số trường.

Ví dụ:

Trong tệp LUONG.DBF, bạn chỉ cần sửa các trường HOTEN, LUONG, bạn dùng lệnh:

```
use LUONG
browse fields HOTEN,LUONG
```

Để việc sử dụng cơ động, đáp ứng yêu cầu người sử dụng, Foxpro 2.6 còn cho phép bạn kê khai từng trường với các tham số đi kèm theo cú pháp sau:

```
<Trường 1> [:R] [<bề rộng cột>]
                [:V = <bt thức 1> [:F] [:E=<bt ký tự 1>]]
                [:P = <bt ký tự 2>]
                [:B = <bt thức 2>,<bt thức 3>[:F]]
                [:H = <bt ký tự 3>] [:W = <bt logic 2>]
[<Trường 2>.....
```

Các tham số trên có ý nghĩa như sau:

- Tham số:R. R viết tắt Read Only, có nghĩa là dữ liệu chỉ được hiện lên màn hình để quan sát chứ không sửa được

Ví dụ:

use LUONG

browse HOTEN:R, LUONG

- Chỉ thị: <bề rộng cột> quy định bề rộng cột (tính theo số cột trên màn hình) của trường cần xem hoặc sửa. Xin bạn lưu ý: chiều rộng của trường trên tệp CSDL không thay đổi, chỉ thay đổi chiều rộng trình bày trên màn hình Browse mà thôi.

Ví dụ:

use LUONG

Browse fileds HOTEN:10,LUONG

- Tham số:V=<bthức 1>. V viết tắt của Verify, tham số này cho phép kiểm tra hợp lệ nội dung của dữ liệu ngay trên màn hình Browse. Khi con trỏ di chuyển khỏi trường mà <bthức 1> nhận giá trị .T. (đúng) thì con trỏ chuyển sang trường tiếp theo. Nếu <bthức 1> nhận giá trị .F. (sai) thì dữ liệu vào là sai, con trỏ sẽ nằm yên tại chỗ và thông báo sai sẽ hiện lên. Nếu <bthức 1> nhận giá trị 0 thì dữ liệu không hợp lệ và dữ liệu nằm yên một chỗ và không có thông báo sai hiện lên. Theo giá trị ngầm định, <bthức 1> chỉ được kiểm tra khi giá trị của trường bị thay đổi. Nếu bạn muốn kiểm tra ngay cả khi giá trị của trường không thay đổi, bạn thêm tham số:F. Tham số :F không thực hiện đối với trường memo.

- Tham số :E = <bt ký tự 1>. E viết tắt của Error. Nếu có tham số này thì khi có sai sót do tham số :V= <bthức 1> phát hiện, thì thông báo <bt ký tự 1> sẽ thay thế thông báo sai của hệ thống (thông báo Invalid Data). Thông báo <bt ký tự

1> chỉ hiện lên nếu thực hiện lệnh SET NOTIFY ON và chuông sẽ kêu lên nếu thực hiện
trước đó có lệnh SET BELL ON.

Ví dụ:

```
use LUONG
browse fields NGAYSINH:V=NGAYSINH>DATE():F;
                                    :E='Sai dữ liệu'
```

- Tham số :B=<bthức 2>, <bthức 3>. B viết tắt chữ Boundaries (miền giá trị), quy định miền giá trị nhận dữ liệu cho trường. Trong đó, <bthức 2> là giới hạn dưới còn <bthức 3> là giới hạn trên. <bthức 2>, <bthức 3> phải cùng kiểu dữ liệu với dữ liệu nhập vào. Có thể là kiểu số, kiểu ký tự hoặc kiểu ngày. Nếu dữ liệu không nằm trong giới hạn <bthức 2>, <bthức 3>, máy thông báo lỗi. Trong trường hợp ngầm định, chỉ khi nào thay đổi dữ liệu của trường thì máy mới kiểm tra, bạn có thể thêm tham số :F để chỉ thị kiểm tra trong trường hợp không thay đổi giá trị của trường.

Ví dụ:

```
use LUONG
browse fields HOTEN:R,LUONG:B=90,300:F
```

- Tham số :H<bt ký tự 3>. H viết tắt của Heading, dùng để thay đổi tiêu đề mặc nhiên của tên trường thành tiêu đề do <bt ký tự 3> ấn định.

Ví dụ:

```
use LUONG
browse fields HOTEN:H='Họ và tên'
```

- Tham số :P=<bt ký tự 2>. P viết tắt chữ Picture, dùng để chỉnh dạng khi cho hiển thị. Đề nghị bạn xem lại các mã

dùng cho Picture trong lệnh @..say..get.

Ví dụ:

```
use LUONG
browse fields HOTEN:R:15,THUONG:='999999'
```

- Tham số :W=<bt logic 1>. W viết tắt của When, dùng để quy định có hay không thâm nhập vào giá trị của trường do <bt logic 1> quy định. Nếu <bt logic 1> cho giá trị .F. (sai), bạn chỉ có thể xem chứ không sửa được, ngược lại, bạn có thể sửa được nếu <bt logic 1> cho giá trị .T. (đúng). Tham số này không áp dụng đối với trường memo.

Ví dụ:

```
use LUONG
browse fields HOTEN:15,LUONG:W=100000
```

Xin bạn lưu ý:

Foxpro dùng 8 tham số :R,:V,:F,:E,:P,:B,:H,:W dùng để thao tác với các trường khi làm việc với Browse. Bạn có thể thay dấu phân cách hai chấm (:) nằm trước mã cần chọn bằng dấu /.

(7). LOCK <bt số 2>, dùng để cố định một số vùng bên trái màn hình (gọi là Left Partition). Khi bạn dùng phím TAB để hiện dữ liệu bị khuất phía bên phải thì các vùng được cố định không bị đẩy sang trái và khuất đi. Trong trường hợp này, khung màn hình được chia thành hai khu vực: khu vực trái cố định và khu vực phải có thể di chuyển.

Ví dụ:

```
use LUONG
browse lock 3
```

LOCK thường dùng để cố định một số trường có tính chất đặc trưng cho một bản ghi. Ví dụ: MA, HOTEN, DONVI. Nếu một bản ghi có chiều dài dài hơn một trang màn hình mà bạn không dùng LOCK để xem hoặc sửa các trường bên phải (khi đó các trường có tính chất đặc trưng ở bên trái đã bị khuất vào bên trái) thì rất khó xem vì không biết dữ liệu của bản ghi nào. LOCK cố định một hay nhiều trường liên tục bên trái. Trong trường hợp bạn sử dụng LOCK để cố định các vùng không liên tục, bạn có thể dùng browse fields <ds trường> n.

Ví dụ:

use LUONG

browse fields MA, DONVI LOCK 2

(8). Tham số WIDTH <bt số 4>. Khi bạn dùng lệnh Browse, chiều rộng của trường hiện lên là chiều rộng theo cấu trúc tệp CSDL. Nếu bạn muốn ấn định chiều rộng chung cho tất cả các vùng khi dùng lệnh Browse thì bạn có thể dùng tham số WIDTH <bt số 4>. <bt số 4> sẽ quy định chiều rộng tối đa khi hiển thị một trường trên màn hình. Ví dụ <bt số 4> là 8. Đối với các trường ký tự có chiều rộng hơn 8, khi bạn gõ đến vị trí cuối phía phải thì các ký tự bên trái sẽ lùi và khuất dần. Nghĩa là bạn vẫn vào dữ liệu bình thường, không ảnh hưởng đến nội dung. Nhìn chung bạn không nên ấn định <bt số 4> nhỏ hơn chiều rộng của trường có kiểu dữ liệu kiểu số.

(9). FOR <bt logic> quy định chỉ những bản ghi nào thoả mãn <bt logic> mới được hiện lên để sửa hoặc xem.

(10). KEY <bthức 1>[, <bthức 2>], dùng để quy định phạm vi xuất hiện của các bản ghi đã được chỉ số,trong đó <bthức 1> là giới hạn dưới, <bthức 2> là giới hạn trên.

Ví dụ:

use LUONG

index on NGAYSINH to NS

browse key {13/07/49},{06/10/60}

Bạn sẽ thấy trên màn hình chỉ xuất hiện những bản ghi mà giá trị của trường NGAYSINH từ 13/07/49 đến 06/10/60.

Nếu bạn chỉ dùng lệnh:

browse key {13/07/49}

thì chỉ những bản ghi có trường NGAYSINH bằng 13/07/49 mới hiện ra trong cửa sổ Browse.

(11). FREEZE <tên trường>. Tiếng Anh Freeze có nghĩa là đông cứng. Khi bạn chỉ muốn đưa dữ liệu vào một trường trong các trường do lệnh Browse hiện ra, ví dụ như LUONG, bạn dùng lệnh:

browse freeze LUONG

(12). Tham số NOAPPEND. Khi đang ở màn hình Browse, bất cứ lúc nào mà bạn ấn phím Ctrl+N hoặc chọn Append Record trong menu browse thì bạn có thể bổ sung một bản ghi vào cuối tệp CSDL. Bản ghi này có dữ liệu hay không tùy thuộc vào lệnh SET CARRY ON/OFF. Nếu trước đó bạn ra lệnh SET CARRY ON, bản ghi mới bổ sung sẽ có dữ liệu là dữ liệu của bản ghi trước, ngược lại, nếu trước đó có lệnh SET CARRY OFF (chế độ ngầm định), bản ghi mới bổ sung sẽ rỗng.

Nếu dùng tham số NOAPPEND trong lệnh Browse, sẽ không bổ sung ban ghi mới khi bạn ấn Ctrl+N hoặc chọn Append Record trong bảng chọn Browse.

Ví dụ:

browse NOAPPEND

(13). Tham số NODELETE, dùng để ngăn việc ấn Ctrl+T hay chọn Toggle Delete trong bảng chọn Browse) khi đang sử dụng lệnh Browse để đánh dấu một bản ghi.

(14). Tham số NOEDIT/NOMODIFY, dùng trong trường hợp bạn chỉ muốn xem chứ không sửa dữ liệu.

(15). Tham số VALID <bt logic 3> [ERROR <bt ký tự 3>] Tham số này dùng để kiểm tra một bản ghi có hợp lệ hay không khi đang ở cửa sổ Browse. Tham số VALID chỉ có hiệu lực khi có thay đổi dữ liệu trên bản ghi và di chuyển con trỏ sang bản ghi khác. Tham số này không có hiệu lực đối với trường memo.

- Nếu <bt logic 3> cho giá trị .T., bạn có thể di chuyển con trỏ đến bản ghi khác.

- Nếu <bt logic 3> cho giá trị .F., khi bạn di chuyển con trỏ sang bản ghi khác thì Foxpro sẽ thông báo:

"Invalid Input".

Nếu có chỉ thị ERROR <bt ký tự 3>, <bt ký tự 3> sẽ thông báo khi vào sai dữ liệu.

- Nếu <bt logic> bằng không (0) thì con trỏ nằm im ở bản ghi và không thông báo sai dữ liệu.

Bạn cũng có thể quy định việc kiểm tra hợp lệ dữ liệu

ngay cả trước khi bạn di chuyển con trỏ hoặc ngay cả khi không thay đổi dữ liệu bằng cách thêm tham số F.

Ví dụ:

use LUONG

browse fields HOTEN,NGAYSINH,LUONG:

VALID LUONG=120000

Bạn hãy đưa con trỏ đến cột LUONG của bản ghi nào đó và sửa lại giá trị. Nếu giá trị sau khi sửa không phải là 120000, khi di chuyển con trỏ sang bản ghi khác, Foxpro sẽ thông báo:

"Invalid Input"

Xin bạn lưu ý:

+ Nếu bạn muốn kết thúc việc sửa đổi, hãy ấn Ctrl-W hoặc Ctrl-End.

+ Lệnh Browse chỉ có hiệu lực khi tệp CSDL của bạn có ít nhất một bản ghi.

- Sử dụng lệnh Browse bằng bảng chọn.

Bạn có thể sử dụng lệnh Browse bằng bảng chọn qua các bước sau:

+ Mở tệp CSDL

+ Ấn F10, chọn D (Database), chọn B (Browse)

Việc sử dụng lệnh Browse tỏ ra rất có hiệu quả trong việc lập các chương trình ứng dụng. Các bạn sẽ tiếp tục tìm hiểu các ứng dụng của lệnh này trong chương: Nhập dữ liệu bằng lệnh Browse mà chúng tôi sẽ trình bày ở Mục 12.5.

Chương 3
LÀM VIỆC VỚI CÁC BẢN GHI TỆP CSDL

3.1. LỆNH REPLACE

Trong trường hợp bạn muốn thay đổi hàng loạt giá trị các bản ghi, thay vì dùng lệnh EDIT hoặc CHANGE bạn có thể dùng lệnh REPLACE.

Cú pháp

REPLACE [<phạm vi>] <Trường 1> WITH <bthức 1>
[,< Trường 2> WITH < bthức 2>]
...
[,< Trường n> WITH < bthức n>]
[FOR <bt logic>] [WHILE <bt logic>]

Lệnh REPLACE dùng để thay thế các <bthức...> vào các trường tương ứng cho các bản ghi nằm trong <phạm vi> và thoả mãn <bt logic> sau FOR hoặc WHILE.

Các biểu thức có thể là hằng, hàm, một biểu thức, tên biến trường... Nhưng một điều bất di bất dịch là: biểu thức và trường tương ứng phải cùng kiểu dữ liệu, nếu không Foxpro sẽ thông báo:

Data type Mismatch.
(Sai dữ liệu)

và đương nhiên câu lệnh không được thực hiện.

Nếu có nhiều trường được thay đổi thì chúng được viết cách nhau một dấu phẩy (xem ví dụ dưới đây).

Ví dụ:

- Giả sử bạn đã có tệp K_HANG. DBF, bây giờ hãy giảm lãi suất của tất cả khách hàng đi 10%. Tại khung của sổ lệnh, bạn hãy ra lệnh:

```
USE K_HANG
REPLACE ALL LAI WITH LAI - LAI *10/100
```

- Bạn hãy tăng lãi suất lên 12% cho những người có mức tiền cho vay >= 1000000

```
REPLACE ALL LAI WITH LAI + LAI*12/100:
                        FOR TIEN>=1000000
```

- Giả sử, (một cách đơn giản) đến cuối tháng, bạn cần cộng số tiền lãi vào số tiền cho vay (gốc) và sau đó tăng lãi suất của tất cả khách hàng lên 7% lãi suất. Câu lệnh như sau:

```
REPLACE ALL TIEN WITH TIEN + TIEN*LAI.:
                        LAI WITH LAI+LAI*7/100
```

Bạn hãy dùng các lệnh DISPLAY ALL, LIST để xem lại kết quả sau mỗi lần ra lệnh.

Xin bạn lưu ý:

- Không được dùng <phạm vi> và <điều kiện> với các tệp có dùng tệp chỉ số.

- Khi thay thế một trường kiểu số mà giá trị của biểu thức thay thế lớn hơn chiều rộng của trường thì FOXPRO sẽ xử lý như sau:

+ Bỏ bớt phần thập phân để lấy chỗ chứa giá trị của biểu thức.

+ Nếu giá trị của biểu thức vẫn còn lớn hơn chiều rộng của trường thì FOXPRO ghi ra dạng khoa học (mũ lũy thừa).

+ Nếu bước trên cũng không thực hiện được do chiều rộng của trường nhỏ hơn 7, trong khi dạng khoa học cần tối thiểu là 7 byte để biểu diễn dạng '.9E+99), hoặc trong biểu thức là một phép chia mà mẫu số = 0. Khi này FOXPRO thông báo:

Numeric Overflow (Data Was Lost)

Nghĩa là "Tràn số" (dữ liệu đã bị mất). Nếu bạn cho hiện ra để xem (lệnh LIST hoặc BROWSE chẳng hạn), thì trường này toàn các dấu sao (*).

- Lệnh REPLACE cho phép thay thế nhiều trường trên cùng một câu lệnh, nếu chúng có cùng điều kiện thay thế như nhau (hoặc cùng không có điều kiện). Khi này bạn nên viết trên cùng một dòng lệnh. Nhưng nếu câu lệnh dài quá 254 ký tự (kể cả ký tự trống), thì bạn phải chia câu lệnh này thành nhiều dòng.

- Khi các lệnh thay thế có điều kiện thay thế khác nhau, bạn phải tách thành nhiều dòng lệnh, mỗi dòng tương ứng với một điều kiện thay thế.

- Bạn cũng có thể sử dụng menu để dùng chức năng REPLACE bằng các bước như sau:

1. Ấn F10, ấn R (Record)

2. Chọn B (REPLACE). Khung hội thoại REPLACE DIALOG xuất hiện.

Phía trên bên trái là danh sách các trường của tệp CSDL có tên ở khung Database (tệp K_HANG). Bạn hãy dùng phím mũi tên để chọn trường cần thay thế sau đó chuyển về chỗ

WITH và ấn Enter, điền biểu thức cần thay thế, chuyển về Scope để khai báo phạm vi, chuyển về FOR (hoặc While) để khai báo điều kiện. Và cuối cùng chuyển về ô <<Replace>> và ấn Enter.

Xin bạn lưu ý: Trong khung cửa sổ lệnh luôn xuất hiện lệnh Replace tương ứng. Cuối màn hình sẽ thông báo cho bạn biết số bản ghi đã được thay thế" XXX Replacements".

3.2. BỔ SUNG, CHÈN VÀ XÓA BẢN GHI

a. Bổ sung bản ghi

Có hai cách để bổ sung một bản ghi vào cuối tệp CSDL.

- Lệnh APPEND

Trong khung cửa sổ lệnh bạn hãy gõ lệnh: APPEND. Sử dụng bang chọn bằng các thao tác tuần tự như sau: ấn phím F10, ấn R (chọn Record), chọn A (chọn Append).

Sau đó trên màn hình xuất hiện một vùng sáng để vào dữ liệu như khi vào dữ liệu sau lệnh CREATE. Sau khi bổ sung xong, bạn hãy ấn Ctrl + W để ghi bản ghi này vào cuối tệp và kết thúc việc bổ sung bản ghi mới. Bạn cũng có thể dùng các phím mũi tên, phím PgUp, phím PgDn để xem lại (thậm chí có thể sửa đổi) nội dung của các bản ghi trước.

Ví dụ:

```
USE K_HANG
APPEND
```

b. Bổ sung một bản ghi trắng dùng lệnh

```
APPEND BLANK
```

Lệnh này bổ sung vào cuối tệp CSDL một bản ghi trắng, có nghĩa là bản ghi đó có mặt trong tệp, cũng chiếm một

phần bộ nhớ bằng kích thước của bản ghi khác nhưng các trường không có dữ liệu. Hay nói chính xác hơn, trường Character, Date nhận giá trị trống, trường Logical nhận giá trị .F., trường Numeric và trường FLoat nhận giá trị 0. Trường hợp này FOXPRO tạo một bản ghi trắng mà không hiện khuôn mẫu để nhập dữ liệu. Câu lệnh này rất hay được sử dụng trong lập trình bằng FOXPRO mà bạn sẽ tìm hiểu trong chương sau.

Ví dụ:

```
USE K_HANG
APPEND BLANK
```

c. Chèn một bản ghi

Trong nhiều trường hợp bạn muốn chèn vào giữa tệp CSDL một bản ghi mới để nội dung phù hợp với công việc thực tế. Bạn hãy sử dụng lệnh INSERT.

Cú pháp

. INSERT [<BLANK>] [<BEFORE>|

- Lệnh INSERT chèn một bản ghi vào ngay sau bản ghi hiện thời. Các bản ghi phía sau bị đẩy lùi xuống.

Ví dụ:

```
USE K_HANG
GO 1
INSERT
```

- Nếu có chỉ thị <BLANK> thì một bản ghi trắng được chèn vào ngay sau bản ghi hiện thời. Các bản ghi phía sau bị đẩy lùi xuống.

- Nếu có chỉ thị <BEFORE> thì một bản ghi mới được chèn vào trước bản ghi hiện thời.

Xin bạn lưu ý

- Đối với lệnh INSERT và APPEND mà trước đó bạn đã dùng lệnh SET CARRY ON thì nội dung của bản ghi trước sẽ tự động chuyển sang bản ghi mới, bạn chỉ cần sửa đổi để trở thành bản ghi cần chèn (hoặc bổ sung vào cuối)

- Nếu tệp có nhiều bản ghi thì dùng lệnh INSERT rất mất thời gian do phải dẫy lùi các bản ghi tại vị trí chèn xuống phía dưới.

d. Xoá bản ghi

- Để đảm bảo an toàn của dữ liệu, FOXPRO cho phép bạn đánh dấu các bản ghi mà bạn dự định xóa trong một tệp CSDL, để nếu cần mới quyết định bỏ dấu xóa hay xóa hẳn.

Cú pháp

DELETE [<Phạm vi>] [FOR <bt logic>]
[WHILE<bt logic>]

Lệnh này đánh dấu ở đầu các bản ghi (dấu *) trong <phạm vi> và thỏa mãn <bt logic> sau FOR hoặc WHILE. Nếu không có các tham số thì lệnh DELETE sẽ đánh dấu bản ghi hiện thời.

Ví dụ:

```
USE K_HANG
DELETE ALL FOR TIEN <1000000
USE
```

- Phục hồi bản ghi đã bị đánh dấu, dùng lệnh:

RECALL [<Phạm vi>] [FOR <bt logic>]
[<WHILE <bt logic>]

Lệnh RECALL khôi phục lại các bản ghi đã bị đánh dấu (bỏ dấu * ở đầu bản ghi) thuộc <phạm vi> và thỏa mãn <bt

logic>. Nếu không có các tham số thì khôi phục lại các bản ghi hiện thời (nếu đã bị đánh dấu).

Nếu không có FOR/WHILE thì khôi phục lại tất cả các bản ghi thuộc <phạm vi>.

Ví dụ:

```
USE K_HANG
RECALL ALL FOR TIEN >= 1000000
```

- Xóa bản ghi đã bị đánh dấu, dùng lệnh PACK.

Các bản ghi đã bị đánh dấu xóa (*) bởi DELETE sẽ bị xóa hẳn thì không còn khôi lại được nữa.

Chú ý: Lệnh này không có <phạm vi> và điều kiện.

e. Xóa tất cả mọi bản ghi trong tệp CSDL đang mở

Dùng lệnh:

```
ZAP
```

Lệnh này xoá bỏ tất cả mọi bản ghi trong tệp CSDL đang mở. Tương đương với lệnh:

```
DELETE ALL
PACK
```

3.3. ĐẶT LỌC DỮ LIỆU

Đây là một trong những khả năng mạnh của FOXPRO khi xử lý dữ liệu. Trong nhiều trường hợp, bạn chỉ muốn xử lý một số bản ghi nào đó mà không muốn làm việc với tất cả các bản ghi của tệp, bạn có thể lọc những bản ghi đó như sau:

```
SET FILTER TO <bt logic>
```

Ví dụ:

```
USE K_HANG
SET FILTER TO TIEN>=1000000
```

LIST

Khi này các lệnh của FOXPRO chỉ có hiệu lực với các bản ghi có SOTIEN (số tiền) lớn hơn hoặc bằng 1000000. Nếu không có <bt logic>, SET FILTER TO sẽ hủy bỏ lọc (nếu có) trước đó của tệp CSDL hiện thời. Trong trường hợp chỉ muốn làm việc với một số trường thì dùng lệnh:

SET FIELDS TO <ds trường>

Khi huỷ giới hạn trường đã đặt, dùng lệnh:

SET FIELDS TO

3.4. LỆNH LOCATE

Cú pháp

LOCATE [<phạm vi >] [FOR <btlogic>]
[WHILE <btlogic>]

Khi gặp lệnh này, FOXPRO sẽ lần lượt duyệt qua tệp cơ sở dữ liệu đang mở để tìm ra bản ghi đầu tiên trong <phạm vi> thỏa mãn điều kiện của <bt logic>. Sau khi đã tìm được một bản ghi thỏa mãn điều kiện, việc tìm kiếm tạm dừng lại để bạn xử lý, ví dụ như hiển thị ra màn hình bản ghi đó (DISPLAY), sửa đổi... Việc tìm kiếm chỉ được tiếp tục khi ra lệnh CONTINUE.

Nếu tìm thấy bản ghi thì:

- Hàm RECNO() cho số thứ tự của bản ghi.
- Hàm EOF() cho giá trị .F.
- Hàm FOUND() cho giá trị .T. cho biết đã tìm ra bản ghi.

Nếu không tìm thấy bản ghi nào, LOCATE thông báo 'End of Locate scope' và con trỏ bản ghi được chuyển đến vị trí kết thúc tệp EOF() (End of File) và lúc này thì:

- Hàm RECNO() cho giá trị bằng tổng số bản ghi của tệp CSDL cộng thêm 1.

- Hàm EOF() cho giá trị .T.

- Hàm FOUND() cho giá trị .F.

Ví dụ:

```
Use MSKH
Locate for DIACHI = 'NAM HA '
?RECNO()
```

Kết quả: 3

Nếu ra lệnh tiếp:

```
DISPLAY
CONTINUE
```

Bạn cũng có thể tìm theo một điều kiện bất kỳ kết hợp của nhiều yêu cầu. Ví dụ tìm người có địa chỉ là "Ha noi" và số tiền cho vay lớn hơn hoặc bằng 1000000. Lệnh như sau:

```
Locate for DIACHI='Hanoi' and TIEN=1000000
Display
```

Nếu chỉ thị <phạm vi>. Locate chỉ tìm trong <phạm vi>. ngược lại nếu không có <phạm vi>. Locate tìm trong toàn tệp.

3.5. LỆNH FIND

Lệnh này chỉ tìm kiếm thông tin trên tệp đã được sắp xếp theo kiểu chỉ số.

Cú pháp

FIND <biểu thức>

trong đó <biểu thức> có thể là biến, biểu thức và chỉ chấp nhận <biểu thức > có kết quả là một chuỗi kí tự. Vì thế chuỗi

kí tự không cần đặt trong dấu nháy đơn, nháy kép. Lệnh này so sánh <biểu thức> với giá trị của khóa sắp xếp của các bản ghi. Nếu việc so sánh thành công, (tức là tìm thấy một bản ghi có khoá sắp xếp bằng <biểu thức> (khóa sắp xếp có nghĩa là nếu bạn sắp xếp chỉ số theo DONVI thì DONVI gọi là khóa sắp xếp) thì con trỏ bản ghi sẽ dừng lại ở bản ghi đó.

Nếu việc so sánh không thành công thì con trỏ bản ghi sẽ dừng ở đâu đó tuỳ theo lệnh SET EXACT hoặc lệnh SET NEAR mà bạn đã ra lệnh trước lệnh FIND.

- Nếu trước lệnh FIND bạn ra lệnh SET EXACT OFF (đây là giá trị ngầm định) thì khi so sánh con trỏ bản ghi sẽ dừng lại ở cuối tệp CSDL nhưng hàm RECNO() sẽ cho thứ tự của bản ghi gần thành công nhất.

(*Xin lưu ý*: số thứ tự ở đây là số thứ tự bản ghi trong tệp CSDL và dương nhiên là nó vẫn gắn với bản ghi trong tệp chỉ số).

- Nếu trước lệnh FIND có lệnh SET NEAR ON thì khi so sánh không thành công, con trỏ bản ghi sẽ dừng lại ở bản ghi gần khớp nhất và hàm RECNO() cho số thứ tự của bản ghi gần khớp nhất (bạn hãy đọc điều lưu ý trên).

- Trong trường hợp lệnh FIND so sánh thành công thì chỉ có bản ghi đầu tiên có khóa sắp xếp trùng với <biểu thức> được tìm ra. Muốn tìm tiếp thì bạn phải ra lệnh SKIP (vì các bản ghi có cùng giá trị của khóa sắp xếp thì liền nhau).

- Nếu trường làm khóa sắp xếp trong tệp chỉ số có chứa dấu trống thì <biểu thức > phải được đặt trong dấu nháy.

- <biểu thức> cũng có thể là một biến ký ức.

Ví dụ:

```
USE MSKH INDEX DIACHI
FIND Ha noi
```

Chú ý: Giả sử bạn cần tìm trên tệp chỉ số được sắp xếp theo trường TEN như sau:

RECORD	HODEM	TEN
3	Tran Thu	Anh
17	Nguyen Phuong	Bich
10	Nguyen Bich	Thuy

Nếu bạn ra lệnh:

```
SET EXACT OFF
Find Thu
? HODEM, TEN, RECNO()
```

Kết quả là:

```
Nguyen Bich Thuy
```

Nhưng nếu ra lệnh:

```
SET EXACT ON
FIND Thu
? FOUND()
```

Kết quả là:

```
.F.
```

(Không tìm thấy)

Sở dĩ trong trường hợp này, không tìm thấy vì FOXPRO lấy chiều dài bản ghi trường TEN của từng bản ghi để so sánh với <biểu thức> (ở đây là Thu) và không tìm thấy bản ghi nào thỏa mãn.

3.6. LỆNH SEEK

Lệnh SEEK cũng tìm kiếm trên tệp chỉ số nhưng thuận

tiện hơn lệnh FIND.

Cú pháp

SEEK <biểu thức>

trong đó <biểu thức> có thể là:

- Một hằng (số, chuỗi kí tự). Ví dụ:

SEEK 137

SEEK 'Trâm'

- Một biến. Ví dụ:

A = 'Trâm'

SEEK A

- Một hàm
- Một biểu thức số học

SEEK 37 * 10 + 20

- Một biểu thức ngày. Ví dụ:

SEEK Date() - 10

Việc tìm kiếm cũng tương tự như lệnh FIND nhưng bạn cần chú ý hai điều khác với lệnh FIND.

- Nếu <biểu thức> là một biến thì không cần hàm & (Macro) ở phía trước.

- Nếu <biểu thức> là một chuỗi kí tự thì bắt buộc phải cho vào trong hai dấu nháy đơn, hoặc nháy kép.

Chương 4
SẮP XẾP DỮ LIỆU

4.1. CÁC YÊU CẦU SẮP XẾP DỮ LIỆU

Trước khi sắp xếp dữ liệu, bạn phải xác định sắp xếp theo trường nào, theo thứ tự tăng dần hay giảm dần, trường nào ưu tiên sắp xếp trước, có phân biệt chữ hoa và chữ thường hay không.

Ví dụ: Bạn có tệp CSDL: NHANSU.DBF có các trường DONVI, PHUCAP, bạn cần sắp xếp trường DONVI theo thứ tự tăng dần, không phân biệt chữ hoa và chữ thường và trong cùng đơn vị thì lại sắp xếp PHUCAP theo thứ tự giảm dần.

4.2. CÁC KÝ HIỆU PHÂN LOẠI

/A (viết tắt của Ascending): sắp xếp theo thứ tự tăng dần.

/D (viết tắt của Descending): sắp xếp theo thứ tự giảm dần.

/C để xác định không phân biệt chữ hoa và chữ thường đối với trường được sắp xếp có kiểu dữ liệu là Character (ký tự).

Nếu bạn không chỉ thị /A, /D thì FOXPRO hiểu là sắp xếp theo thứ tự tăng dần.

Nếu bạn có hai chỉ tiêu cho cùng một trường, ví dụ sắp xếp trường DONVI theo thứ tự tăng dần và không phân biệt chữ

hoa và chữ thường thì có thể viết gộp /AC.

4.3. LỆNH SORT

Cú pháp

SORT TO <tệp mới>

ON <trường 1> / <chỉ tiêu phân loại>

[,<trường 2> / <chỉ tiêu phân loại>]

.....

[,<trường n> / <chỉ tiêu phân loại>]

[<phạm vi>] [FOR <bt logic>]

[WHILE <bt logic>]

[FIELDS <ds trường>]

Lệnh SORT sắp xếp các bản ghi theo trường được chỉ thị sau ON và ghi ra <tệp mới>.

Nếu có chỉ thị FIELD <ds trường> thì chỉ các trường nằm trong <ds trường> mới được ghi sang <tệp mới>.

Nếu có chỉ thị <phạm vi>, chỉ các bản ghi nằm trong <phạm vi> mới được sắp xếp.

Nếu có chỉ thị FOR/WHILE <bt logic> thì chỉ những bản ghi thoả mãn <bt logic> mới được sắp xếp.

Ví dụ:

```
USE NHANSU
SORT TO SAPXEP ON DONVI/AC, PHUCAP/D
USE SAPXEP
LIST
```

4.4. LỆNH SẮP XẾP THEO CHỈ SỐ: LỆNH INDEX

a. Khái niệm tệp chỉ số

Trước hết chúng ta quan tâm đến loại tệp chỉ số cổ điển đã xuất hiện trong DBASE/FOXBASE và FOXPRO Version cũ.

Tệp chỉ số là tệp đã sắp xếp theo một chỉ tiêu nào đó của tệp CSDL, nhưng số thứ tự của các bản ghi vẫn giữ nguyên số thứ tự của tệp CSDL.

Chúng ta hãy xét một ví dụ đơn giản sau: giả sử bạn có tệp CSDL có hai trường TEN (tên) và PHUCAP (phụ cấp) như sau:

Record#	TEN	PHUCAP
1	NAM	21000
2	NHAN	17000
3	BAN	19000
4	TY	19000
5	HUONG	18500
6	TRUNG	24000

Nếu bạn sắp xếp TEN theo thứ tự tăng dần bằng lệnh Sort và dùng lệnh List để xem nội dung của tệp kết quả, bạn sẽ thấy:

Record#	TEN	PHUCAP
1	BAN	19000
2	HUONG	18500
3	NAM	21000
4	NHAN	17000

5	TRUNG	24000
6	TY	19000

Nhưng nếu bạn sắp xếp trường TEN tăng dần theo kiểu chỉ số, bạn sẽ được kết quả như sau:

Record#	TEN	PHUCAP
3	BAN	19000
5	HUONG	18500
1	NAM	21000
2	NHAN	17000
6	TRUNG	24000
4	TY	19000

Như vậy kiểu sắp xếp theo chỉ số thì sau khi sắp xếp sẽ được trường TEN tăng dần, nhưng vẫn biết được vị trí của từng bản ghi trong tệp CSDL.

Tệp chỉ số kiểu trên gọi là tệp chỉ chuẩn (Standard Index File) mang phần mở rộng là .IDX đã được dùng trong DBASE/FOXBASE và các FOXPRO Version cũ. Ngoài ra trong FOXPRO Version 2.0 còn xuất hiện hai loại tệp chỉ số mới là:

- Tệp chỉ số cơ động (Compact Index File) cũng có phần mở rộng là .IDX nhưng do kỹ thuật dồn nén đặc biệt nên dung lượng của tệp loại này chỉ còn bằng 1/6 dung lượng của tệp chỉ số chuẩn.

- Tệp chỉ số kết hợp (Compound Index File) mang phần phân loại .CDX. Trong tệp loại này có một loại đặc biệt gọi là: Structural Compound Index File được tạo ra và gắn liền với cấu trúc CSDL và nó được tự động mở ra khi tệp CSDL liên

quan đến nó được mở. Tệp chỉ số loại này cũng được cô đọng, dồn nén như loại trên.

b. Lệnh tạo tệp chỉ số trong khung cửa sổ lệnh

Trước hết, bạn phải mở tệp CSDL và sau đó dùng lệnh INDEX ON để sắp xếp theo kiểu chỉ số.

Cú pháp

INDEX ON <bthức>
TO <tệp CS.IDX>/TAG <TAG Name> [OF <tên tệp>]
FOR <bt logic> [Ascending/Descending]
[<Unique>]

trong đó: <bthức> có thể là tên trường, biểu thức mà có biến thường tham gia, <tệp CS.IDX> là tệp chỉ số dạng .IDX.

Ý nghĩa các tham số sẽ được giải thích sau đây:

1. Nếu bạn muốn tạo tệp chỉ số dạng .IDX thì viết lệnh:

INDEX ON <bthức> TO <Tệp CS.IDX>

2. Nếu bạn muốn tạo tệp chỉ số dạng kết hợp (.CDX) thì viết:

INDEX ON <bthức> TO TAG <TAG name>
[OF <tên tệp>]

Ví dụ:

Bạn có tệp CSDL tên là HSKH.DBF (hồ sơ khách hàng). Trong tệp này có các trường HOTEN (họ tên), DIACHI (địa chỉ ghi theo tên tỉnh, thành phố) và SOTIEN (số tiền mà cơ quan bạn còn nợ hoặc khách hàng vay). Bạn có thể sắp xếp theo kiểu chỉ số để tạo ra ba tệp với khóa sắp xếp là ba trường trên.

USE HSKH

INDEX ON HOTEN TO KHHOTEN
INDEX ON DIACHI TO KHDIACHI INDEX ON
SOTIEN TO KHSOTIEN

Như vậy, bạn đã tạo được ba tệp chỉ số mà các trường được sắp xếp theo thứ tự tang dần (chế độ ngầm định). Việc xem nội dung bạn sẽ tìm hiểu ở mục dưới đây. Các tệp trên có phần mở rộng ngầm định .IDX và hoàn toàn tương thích với FOXBASE và FOXPRO Version cũ. Nhưng đối với tệp chỉ số kết hợp (.CDX) bạn chỉ cần một tệp chỉ số có thể lưu trữ được thông tin của ba tệp trên. Bạn chỉ cần một tệp (ví dụ: KHCHISO) nhờ các lệnh sau:

INDEX ON HOTEN TAG TEN of KHCHISO
INDEX ON DIACHI TAG DCHI of KHCHISO INDEX ON
SOTIEN TAG TIEN of KHCHISO

Ở trên: TEN, DCHI, TIEN được coi là các Tag Name (tạm gọi là tên đính mục). Chúng hoạt động như các biến. Khi một tệp CSDL được mở thì các đính mục cũng được tự động mở theo.

Tệp chỉ số .CDX được chia thành hai loại:

- Loại khai báo sau TAG <tag name> không có phần OF <tên tệp> đi sau. Tệp này bao giờ cũng cùng tên với tệp CSDL và được mở tự động khi tệp CSDL được mở, khi tệp CSDL có biến đổi dữ liệu thì tệp chỉ số này biến đổi theo (tệp chỉ số này được gọi là .CDX Structural).

- Loại khai báo TAG <tag name> có chỉ thị OF <tên tệp> (như ví dụ trên). Tệp chỉ số loại này không được tự động mở, vì vậy bạn phải dùng các lệnh SET INDEX hoặc INDEX trong lệnh USE mà chúng tôi sẽ giới thiệu dưới đây để mở

các tệp loại này. Chúng ta tạm gọi là tệp chỉ số kết hợp độc lập. Khi một tệp chỉ số .CDX được mở (đã được tạo ra trước đó), thì khi bạn viết lệnh INDEX ON ... TAG <tag name> thì một đính mục mới được thêm vào tệp chỉ số .CDX. Nếu tệp này chưa được tạo ra thì FOXPRO tự tạo ra.

3. Tham số Ascending/Descending để chỉ thị sự sắp xếp tăng hoặc giảm (ngầm định là Ascending). Tham số Descending chỉ có ý nghĩa cho tệp .CDX còn đối với tệp loại .IDX thì tham số này không có hiệu lực .

4. Tham số Unique (duy nhất) có ý nghĩa như sau: Trong tệp CSDL của bạn có thể có nhiều bản ghi cùng một khóa sắp xếp (ví dụ cùng một tỉnh hay thành phố). Nếu bạn chỉ cần lấy ra một bản ghi đầu tiên để đưa ra tệp chỉ số, bạn hãy dùng tham số này. Nếu bạn muốn biết thông tin về những tệp chỉ số của tệp CSDL hiện thời, bạn chỉ cần ra lệnh DISPLAY STATUS. FOXPRO sẽ cho hiện lên các thông tin liên quan đến các tệp chỉ số.

Xin các bạn lưu ý:

Giống như lệnh SORT, Bạn cũng có thể dùng bảng chọn để sắp xếp một tệp theo kiểu chỉ số. Bạn ấn F10, chọn F (File), chọn New, chọn Index. Bạn hãy dùng phím <TAB> để chuyển vệt sáng đến nút <<OK>> ấn Enter. Sau đó bạn phải khai báo tiếp các tham số cần thiết.

4.5. SỬ DỤNG TỆP CHỈ SỐ

Khi bạn muốn dùng một tệp chỉ số thì bạn phải mở tệp CSDL liên quan đến nó và sau đó mở các tệp chỉ số cần thiết.

Tuy nhiên, tại một thời điểm chỉ có một chỉ số có thể làm việc được, tệp đó gọi là tệp chủ (Master Index).

a. Dùng lệnh SET INDEX

Cú pháp

SET INDEX TO [<danh sách tên tệp chỉ số>/?]
[ORDER <bt số>/<tệp chỉ số .IDX> /
[TAG] <tag name> [OF <tên tệp CDX>]]
[ASCENDING/DESCENDING]

Ý nghĩa của các tham số được giải thích như sau:

- <Danh sách tệp chỉ số>: Bạn hãy liệt kê danh sách các tệp chỉ số cần mở và chúng được cách nhau bởi một dấu phẩy (,). Tệp đứng đầu trong danh sách là tệp chỉ số chủ, nghĩa là nếu bạn dùng các lệnh như LIST, DISPLAY... thì sẽ làm việc với tệp này.

Ví dụ:

SET INDEX TO KHHOTEN, KHDIACHI, KHSOTIEN
LIST

Lệnh trên mở ba tệp chỉ số trong đó tệp KHHOTEN là tệp chỉ số chủ.

- Nếu bạn dùng lệnh

SET INDEX TO ?

Trên màn hình sẽ hiện lên khung đối thoại "Open Index File Dialog". Trong khung đối thoại này sẽ hiện tất cả các tệp chỉ số trong thư mục hiện thời để bạn chọn.

Nếu trên đĩa của bạn có nhiều tệp chỉ số và công việc của bạn cần dùng tới các tệp đó, tốt nhất bạn nên mở đồng thời như trên và khi cần làm việc đến tệp nào thì bạn chỉ định tệp

đó làm tệp chỉ số chủ. Cách làm như vậy FOXPRO thực hiện nhanh hơn rất nhiều so với việc bạn mở liên tục từng tệp một. Việc thay đổi tệp chỉ số chủ có các cách như sau:

+ *Cách 1*:

SET INDEX TO ORDER <bt số>

trong đó <bt số> là một số nguyên dương chỉ thị:

- Nếu là tệp chỉ số dạng .IDX thì lệnh trên chỉ thị tệp chỉ số có số thứ tự là <bt số> trong danh sách tệp chỉ số được mở, làm tệp chỉ số chủ.

Ví dụ:

SET INDEX TO ORDER 2

Chỉ thị tệp chỉ số đứng ở vị trí thứ hai trong danh sách tệp chỉ số được mở, làm tệp chỉ số chủ.

- Nếu là tệp chỉ số dạng .CDX, thì số thứ tự của đính mục dựa trên thứ tự được tạo ra là đính mục chủ.

+ *Cách 2*:

SET INDEX TO ORDER <tệp chỉ số .IDX> lệnh này chỉ định đích danh < tệp chỉ số .IDX> là tệp chỉ số chủ.

Ví dụ:

SET INDEX TO ORDER KHCHISO

+ *Cách 3*:

SET INDEX TO ORDER [TAG] < Tag name>

OF <tệp chỉ số .CDX> cách này chỉ thị rõ tệp chỉ số có đính mục mang tên <Tag name> của tệp chỉ số .CDX làm tệp chỉ số chủ.

Ví dụ:

SET INDEX TO ORDER TEN OF KHCHISO

Nếu bạn chỉ thị:

SET INDEX TO

thì không tệp nào là tệp chỉ số chủ và bạn có thể làm việc thoải mái với tệp CSDL.

b. Dùng lệnh USE

Như các bạn đã biết, lệnh USE dùng để mở một tệp CSDL, nhưng bạn cũng có thể dùng để mở đồng thời với một tệp chỉ số đã được tạo ra từ tệp CSDL này.

Ví dụ:

USE HSKH INDEX KHHOTEN, KHDIACHI

c. Dùng lệnh SET ORDER

Lệnh SET ORDER cũng được dùng tương tự như tham số ORDER trong lệnh INDEX TO mà chúng ta vừa xét.

Ví dụ:

SET ORDER TO 3

chỉ thị tệp thứ ba trong danh sách tệp chỉ số làm tệp chỉ số chủ.

SET ORDER TO KHHOTEN.IDX

chỉ thị tệp KHHOTEN làm tệp chỉ số chủ.

Để dễ hiểu hơn, các bạn hãy tìm hiểu ví dụ dưới đây:

Giả sử có một tệp CSDL là MSKH (mã số khách hàng) được mở với ba tệp chỉ số sau đây: KHHOTEN.IDX, KHCHISO.CDX, KHSOTIEN.IDX bằng lệnh:

USE HSKH INDEX KHHOTEN.IDX, KHCHISO.CDX,:
KHSOTIEN.IDX

trong đó tệp KHCHISO có hai đính mục là DC và MS.

Xin các bạn lưu ý: KHCHISO.CDX là tệp chỉ số kết hợp

độc lập vì nếu là Structural thì phải trùng tên với tệp CSDL (tệp HSKH). Ngoài ra, giả sử tệp MSKH.DBF còn có một tệp chỉ số Structural. CDX có các đinh mục là LAISUAT, NGAYTHANG. Tệp Structural được tự động mở chỉ mở tệp CSDL. Bạn hãy dùng lệnh DISPLAY STATUS để xem thông tin liên quan đến tệp MSKH.DBF như sau:

```
Currently Selected Database
Select area:1, Database is use: c:\FOXPRO\MSKH.DBF
Master index file c:\FOXPRO\KHHOTEN.IDX Key: TEN
Index file c:\FOXPRO\KHSOTEN.IDX Key: SOTIEN
Structural.CDX file c:\FOXPRO\MSKH.CDX
Index Tag: LAIXUAT Key: LAI
Index Tag: NGAYTHANG Key: NGAY
CDX File C:\FOXPRO\KHCHISO.CDX
Index Tag DC Key: DIACHI
Index Tag MS Key: MASO
```

Vì tệp chỉ số .IDX đánh số trước tiên nên khi bạn ra lệnh: SET ORDER TO 1 thì tệp KHHOTEN.IDX là tệp chỉ số chủ (chứ không phải là tệp KHCHISO.CDX).

Tiếp theo là các đinh mục trong tệp chỉ số Structural MSKH.CDX được đánh số thứ tự.

```
SET ORDER TO 3
Master index C:\FOXPRO\MSKH.CDX Tag: LAISUAT
SET ORDER TO 4
Master index C:\FOXPRO\MSKH.CDX
                                    Tag: NGAYTHANG
```

và cuối cùng các đinh mục của tệp chỉ số .CDX độc lập

KHCHISO.CDX mới được đánh số thứ tự.

SET ORDER TO 5
Master index C:\FOXPRO\KHCHISO.CDX Tag DC
SET ORDER TO 6
Master index C:\FOXPRO\KHCHISO.CDX Tag MS

Xin các bạn lưu ý:-

Nếu bạn chỉ thị: SET ORDER TO hoặc SET ORDER TO (không chỉ thị gì tiếp theo) thì bạn sẽ làm trực tiếp với tệp CSDL.

- Cũng có thể dùng hệ thống menu để mở các tệp chỉ số.

Trước tiên bạn ấn F10, chọn D (Database) chọn SETUP. Màn hình "Setup Dialog" xuất hiện và bạn chọn các tham số cần thiết.

4.6. CÁC LỆNH LIÊN QUAN ĐẾN TỆP CHỈ SỐ

a. Lệnh COPY INDEXES

Cú pháp

Copy INDEXES <ds tệp chỉ số .IDX>/All
[TO <tệp chỉ số .CDX>]

- Trước khi ra lệnh copy, bạn phải mở tệp CSDL kèm theo những tệp chỉ số .IDX mà bạn cần chép lên tệp chỉ số .CDX. Nếu bạn chỉ thị:

COPY INDEXES ALL

thì tất cả các tệp chỉ số được chép lên tệp chỉ số CDX, ngược lại nếu bạn chỉ thị <ds tệp chỉ số .IDX> thì chỉ những tệp được liệt kê trong <ds tệp chỉ số . IDX> mới được sao chép.

- Nếu bạn không chỉ thị TO <tệp chỉ số .CDX> thì FOXPRO coi tệp đích là tệp chỉ số kết hợp Structural. Nếu

bạn có chỉ thị TO <tệp chỉ số .CDX> thì đây là tệp chỉ số độc lập.

b. Lệnh COPY TAG

Lệnh này làm ngược lại với lệnh COPY INDEXES. tức là trích từ đính mục của một tệp chỉ số .CDX để tạo ra một tệp chỉ số loại .IDX.

Cú pháp

COPY TAG <Tag name> [OF <tệp chỉ số. CDX>]:

TO <Tệp chỉ số .IDX>

- Trước khi ra lệnh COPY TAG bạn phải đảm bảo là tệp chỉ số .CDX đã được mở bởi lệnh USE hoặc SET INDEX TO và bạn phải cung cấp <Tag name> (tên đính mục) cho cụ thể.

- Nếu tệp chỉ số .CDX là Structural thì bạn không cần phải chỉ thị OF <tệp chỉ số .CDX>. Ngược lại, nếu là tệp chỉ số kết hợp độc lập thì bạn phải chỉ thị rõ tên tệp .CDX mà từ đó bạn trích ra một tệp chỉ số .IDX.

c. Lệnh DELETE TAG

Dùng để xoá bỏ các đính mục không còn dùng nữa khỏi tệp chỉ số .CDX. Nếu bạn xoá tất cả các đính mục thì tệp chỉ số này bị xoá hoàn toàn khỏi đĩa từ.

Cú pháp

DELETE TAG <tên Tag> [OF <tên tệp .CDX>]

DELETE TAG ALL [OF <tệp .CDX>]

Bạn có thể xoá một hoặc nhiều đính mục riêng lẻ (dạng đầu) hoặc xoá toàn bộ các đính mục (dạng sau).

Xin các bạn lưu ý:

Vì lệnh liên quan đến tệp chỉ số CDX nên FOXPRO tìm đích mục được chỉ thị trong tệp CDX Structural trước (nếu có). Nếu không tìm thấy mới đi tìm ở tệp CDX độc lập đang mở.

d. Lệnh CLOSE INDEX

Lệnh CLOSE INDEX sẽ đóng tất cả các tệp chỉ số .IDX và .CDX (trừ tệp chỉ số Structural) đã được mở trong vùng làm việc hiện thời.

Chương 5
CÁC LỆNH XỬ LÝ ĐẶC BIỆT CỦA DỮ LIỆU KIỂU SỐ

5.1. LỆNH SUM

Cú pháp

SUM [<phạm vi>] [<ds trường>]
[FOR <bt logic>] [WHILE <bt logic>]
[TO <danh sách biến nhớ>]

- Lệnh SUM tính tổng tất cả các trường số (nếu không có chỉ thị <ds trường>) của tệp CSDL hiện thời và đưa vào <danh sách các biến nhớ> theo thứ tự tương ứng.

- Nếu có chỉ thị <ds trường> thì chỉ các trường số nằm trong <ds trường> mới được tính tổng.

- Nếu có chỉ thị FOR/WHILE <bt logic> thì chỉ những bản ghi thỏa mãn <bt logic> mới được tính tổng.

- Nếu có chỉ thị <phạm vi> thì chỉ những bản ghi thuộc <phạm vi> mới được xét đến. Nếu không có <phạm vi> thì SUM tất cả.

- Nếu các biến nhớ trong <danh sách biến nhớ> chưa tồn tại thì Foxpro tự tạo ra.

Ví dụ:

Trên khung cửa sổ lệnh bạn ra lệnh:

USE MSKH
SUM SOTIEN

- Giả sử trong tệp LUONG.DBF của bạn có ba trường LCHINH (lương chính), PHUCAP (phụ cấp), THUONG (thưởng). Nếu công thức tính tiền (một cách đơn giản) theo công thức sau:

Thực lĩnh=Lương chính+ Phụ cấp + Thưởng

Bạn có thể tính tổng số tiền thực lĩnh trong toàn cơ quan như sau:

+ *Cách 1*: Tại khung cửa sổ lệnh, bạn ra lệnh:

```
SET TALK ON
SUM LCHINH + PHUCAP + THUONG
```

Kết quả trên màn hình (ví dụ)

```
123000000
```

+ *Cách 2*: Tại khung cửa sổ lệnh, bạn ra lệnh:

```
SET TALK OFF
SUM LCHINH, PHUCAP, THUONG, TO A1, A2, A3
? A1 + A2 + A3
```

Kết quả trên màn hình

```
123000000
```

5.2. LỆNH TOTAL

Cú pháp

```
TOTAL ON <khóa> TO <tệp mới>
      [<phạm vi>] [FIELD <ds trường>]
      [FOR <bt logic>] [WHILE <bt logic>]
```

Lệnh tính tổng các trường số của các bản ghi có <khóa> như nhau (Ví dụ: trong tệp CSDL có trường DONVI và bạn chọn khóa là DONVI, thì TOTAL sẽ rà xét các bản ghi có cùng đơn vị liên tục để cộng từng trường số của các bản ghi

lại với nhau). Khi xuất hiện một bản ghi có <khóa> khác với <khóa> hiện thời (ví dụ: đơn vị khác với đơn vị ở bản ghi phía trên) TOTAL sẽ ghi bản ghi có <khóa> trước vào <tệp mới> và tổng các trường số sẽ ghi vào các trường số tương ứng.

Thông thường trước khi dùng lệnh này nên sắp xếp dữ liệu theo <khóa> cần thiết.

Nếu chỉ thị <phạm vi> thì chỉ có các bản ghi nằm trong <phạm vi> mới được TOTAL xét đến.

Nếu có FOR/WHILE <biểu thức logic> chỉ có các bản ghi thỏa mãn <biểu thức logic> mới được TOTAL xét đến. Nếu chỉ thị FIELD <ds trường> thì chỉ có các trường được liệt kê trong <danh sách trường> mới được tính tổng và đưa ra tệp kết quả các trường số còn lại, số lấy giá trị của bản ghi đầu trong <khóa>.

Nếu vùng tiếp nhận không đủ rộng để lưu trữ kết quả sẽ bị tràn số.

Ví dụ:

Bạn có tệp LUONG.DBF lưu trữ các thông số về lương trong cơ quan có các trường: DONVI (đơn vị), LCHINH (lương chính), PHUCAP (phụ cấp), THUONG (thưởng), THUCLINH (tiền thực lĩnh của từng cán bộ).

Bây giờ bạn hãy tạo một tệp mới, tệp BAOCAO.DBF lưu trữ tổng số tiền lương chính, phụ cấp, thưởng, thực lĩnh của từng đơn vị. Tại khung cửa sổ lệnh bạn hãy ra lệnh:

```
USE LUONG
SORT ON DONVI/A TO SAPXEP
USE SAPXEP
TOTAL ON DONVI TO BAOCAO
```

```
USE BAOCAO
LIST
USE
```

Xin các bạn lưu ý:

Nếu không sắp xếp lại các dữ liệu theo <khóa> mà đã TOTAL thì rất có thể gặp sai sót sau:

Ví dụ: Trong tệp của bạn ở bản ghi thứ 1, 2, 3, 4, 5 đơn vị là KT (kỹ thuật), các bản ghi 6, 7, 8, 9 là HC (hành chính) bản ghi thứ 10, 11 lại là KT. TOTAL sẽ tính tổng các trường số của đơn vị KT, từ các bản ghi 1, 2, 3, 4, 5 và đưa ra tệp kết quả, sau đó lại tính tổng các trường số của đơn vị HC ở các bản ghi 6, 7, 8, 9, và cuối cùng tính tổng các trường số 10, 11 và đưa ra tệp kết quả. Hoàn toàn dễ thấy là kết quả không đúng theo ý đồ của bạn vì bạn cần tính tổng các trường số của tất cả các bản ghi có đơn vị là KT tức là phải tính các trường số của các bản ghi thứ 1, 2, 3, 4, 5, 10, 11 lại với nhau.

5.3. LỆNH COUNT

Cú pháp

COUNT [<phạm vi>] [To <biến>]
[FOR <bt logic>] [WHILE <btlogic>]

Lệnh này đếm số bản ghi nằm trong <phạm vi> thỏa mãn <bt logic> và đưa vào <biến>.

- Nếu không chỉ thị <phạm vi> lệnh sẽ hoạt động trên toàn tệp.
- Nếu không có FOR/WHILE <bt logic> lệnh sẽ đếm trong cả <phạm vi> (nếu có) hoặc trong toàn tệp (nếu không có <phạm vi>).

Ví dụ:

```
USE LUONG
COUNT TO A1
COUNT B1 FOR DONVI='HC'
COUNT C1 FOR LCHINH>=39000
```

5.4. LỆNH AVERAGE

Cú pháp

AVERAGE [<phạm vi>] [<ds trường>]
[FOR <bt logic>] [WHILE <bt logic>]
[TO<ds biến>]

Lệnh AVERAGE tính trung bình cộng của các trường số trong tệp CSDL đang mở trong vùng làm việc hiện thời.

- Nếu có chỉ thị <ds trường> chỉ các trường được liệt kê trong <ds trường> mới được tính trung bình cộng.

- Nếu có chỉ thị <phạm vi> chỉ các bản ghi nằm trong <phạm vi> mới được tính trung bình cộng, nếu không, tính trên toàn tệp.

- Nếu có chỉ thị TO <ds biến>, kết quả sẽ được đưa vào các biến tương ứng. Nếu không sẽ đưa ra màn hình nếu trước đó có lệnh SET TALK ON (ngầm định) và có tiêu đề nếu trước đó có lệnh SET HEADING ON.

Ví dụ:

Tính trung bình cộng giá tiền của một quyển sách trong kho sách của bạn. Bạn hãy ra lệnh:

```
USE SACH
AVERAGE GIATIEN TO GIATB
? GIATB
```

Phần 2

CÁC VÍ DỤ ÁP DỤNG TRONG CHẾ ĐỘ THAO TÁC TRỰC TIẾP

Chương 6

CÁC VÍ DỤ VỀ QUẢN LÝ LƯƠNG TRONG CƠ QUAN XÍ NGHIỆP

6.1. TẠO TỆP CSDL

Dùng lệnh Create để tạo tệp LUONG có cấu trúc như sau:

Fields name	Type	Width	Dec
HODEM	Character	15	
TEN	Character	7	
DONVI	Character	4	
LUONG	Numeric	6	
PHUCAP	Numeric	6	
THUONG	Numeric	6	
TAMUNG	Numeric	6	

(sau khi khai báo xong, hãy ấn Ctrl+W và Enter để ghi lại trả lời Y khi máy hỏi:

Input data records now (Y/N)?)

Hãy vào dữ liệu cho các bản ghi nhưng xin bạn lưu ý một điểm sau: trường DONVI chỉ đưa vào các đơn vị là: KH - Kế hoạch; HC - Hành chính; KT - Kỹ thuật (chúng tôi đưa ra yêu cầu này chỉ để các bạn tiện theo dõi các ví dụ tiếp theo. Tất nhiên là các bạn có thể bổ sung, sửa đổi sau khi đã tìm hiểu các ví dụ chúng tôi đưa ra dưới đây). Sau khi vào dữ liệu xong, hãy ấn Ctrl+W để ghi lại.

6.2. XEM THÔNG TIN

- Dùng lệnh

LIST

- Xem danh sách tất cả các cán bộ ở phòng Kế hoạch bằng lệnh:

Display all for DONVI='KH'

- Để khắc phục sai sót khi vào dữ liệu (chẳng hạn như đúng ra phải viết là KH thì lại đưa vào là kH, Kh...), hãy dùng lệnh sau:

Display all for upper(DONVI)='KH'

Xin các bạn lưu ý:

Hàm upper(<bthức>) sẽ biến <bthức> thành chữ hoa.

Ví dụ:

? upper('Thanh Nga')

Kết quả là: THANH NGA

Một hàm khác, hàm lower() lại ngược với hàm này, tức là biến các chữ hoa thành chữ thường.

Ví dụ:

?lower('Thanh Nga')

Kết quả là: thanh nga

Như vậy, ví dụ trên còn có thể viết lại như sau:

Display all for lower(DONVI)='kh'

- Trong trường hợp bạn muốn tìm những người ở phòng Kế hoạch đã tạm ứng tiền, dùng lệnh:

Display all for upper(DONVI)='KH' and TAMUNG>0

- Nếu muốn xem danh sách cán bộ ở phòng Kế hoạch và phòng Kỹ thuật có mức thưởng lớn hơn hoặc bằng 500.000đ thì câu lệnh như sau:

```
Display all for (upper(DONVI)='KH'
    or upper(DONVI)='KT') and THUONG>=500000
```

6.3. SỬA CẤU TRÚC

- Sửa lại cấu trúc để bổ sung thêm cột CONLAI để tính số tiền còn được nhận của mỗi cán bộ bằng lệnh:

```
Modify structure
```

Đưa vệt sáng xuống tận cuối và bổ sung thêm trường CONLAI như sau:

```
CONLAI  Numeric  7    0
```

- Thay đổi vị trí của trường THUONG lên trên trường PHUCAP bằng cách đưa vệt sáng đến cột THUONG, giữ phím Ctrl và ấn phím PgUP (nếu chuyển xuống ấn phím PgDn). (Ấn Ctrl+W và Enter để ghi lại). Để kiểm tra lại, dùng lệnh LIST.

Cũng có thể sao cấu trúc của tệp đang mở để tạo ra một tệp khác có cấu trúc y nguyên như thế bằng lệnh:

```
Copy structure to LUU
```

Tệp LUU.DBF đã được tạo ra có cấu trúc như tệp LUONG.DBF

6.4. SỬA LẠI NỘI DUNG MỘT BẢN GHI

Giả sử phát hiện thấy một bản ghi bị sai và cần sửa lại, bản ghi thứ năm chẳng hạn. Viết lệnh:

```
GO 5
EDIT
```

Cũng có thể dùng: lệnh BROWSE để chỉnh sửa (sau khi sửa xong hãy ấn phím Ctrl+W để ghi lại).

6.5. ĐÓNG TỆP VÀ KẾT THÚC CÔNG VIỆC

Dùng lệnh:

use

Cũng có thể đóng tệp bằng cách khác

close databases

hoặc close all

6.6. MỞ TỆP VÀ BỔ SUNG THÊM BẢN GHI MỚI

Sau một thời gian có thể sẽ xuất hiện thêm một số cán bộ mới. Hãy bổ sung vào tệp CSDL của bạn. Trước tiên phải mở tệp CSDL bằng lệnh:

use LUONG

và sau đó bổ sung thêm bản ghi mới bằng lệnh:

APPEND

(sau khi bổ sung xong ấn Ctrl+W để ghi và kết thúc).

6.7. CHÈN THÊM BẢN GHI MỚI

Trong nhiều trường hợp, bản ghi mới bổ sung phải vào một vị trí nào đó, phải cùng khu vực với các cán bộ ở đơn vị mà cán bộ mới sẽ làm việc chẳng hạn. Giả sử bạn muốn bản ghi mới ở sau bản ghi thứ mười, hãy ra lệnh:

GO 10

INSERT

cũng có thể ra lệnh bằng cách khác:

GO 11

INSERT BEFORE

Hai lệnh này là như nhau, nghĩa là nếu thêm tham số

BEFORE thì bản ghi sẽ được chèn vào trước bản ghi hiện thời.

6.8. SỬA LẠI NỘI DUNG NHIỀU BẢN GHI

Bây giờ hãy tính lại tiền thưởng theo chỉ tiêu sau:

Lương dưới 200.000đ :

Tiền thưởng=65% lương

Lương từ 200.000 đ đến dưới 300.000đ

Tiền thưởng=50% lương

Lương từ 300.000 đ đến dưới 400.000đ

Tiền thưởng=45% lương

Lương từ 400.000 đ trở lên

Tiền thưởng=40% lương

Các câu lệnh sẽ được thực hiện như sau:

```
replace all THUONG with 65/100*LUONG;
        for LUONG<200000
replace all THUONG with 50/100*LUONG;
        for LUONG>=200000 and LUONG<300000
replace all THUONG with 45/100*LUONG;
        for LUONG>=300000 and LUONG<400000
replace all THUONG with 40/100*LUONG;
        for LUONG>=400000
```

6.9. TÍNH SỐ TIỀN CÒN ĐƯỢC LĨNH CỦA TỪNG CÁN BỘ

Số tiền còn được lĩnh của từng cán bộ được tính theo công thức sau:

Số tiền còn lại= lương + thưởng + phụ cấp - tạm ứng

Câu lệnh sẽ là:

replace all CONLAI WITH
(LUONG+THUONG+PHUCAP-TAMUNG)

Để kiểm tra lại công việc đã làm, hãy dùng lệnh LIST.

6.10. XOÁ ĐI MỘT SỐ BẢN GHI

Giả sử có hai cán bộ chuyển sang cơ quan khác cần phải xoá tên trong danh sách, hai cán bộ số 10 và 12 chẳng hạn. Trước khi xoá một bản ghi, FOXPRO cho phép đánh dấu bản ghi và sau đó mới quyết định xoá hay không xoá (khôi phục bản ghi đã bị đánh dấu xoá). Bạn hãy đánh dấu hai bản ghi này:

```
go 10
delete
go 12
delete
```

Dùng lệnh LIST để kiểm tra lại thì sẽ thấy hai bản ghi này bị đánh dấu (dấu *) ở đầu bản ghi.

Giả sử hai cán bộ đó có một cán bộ không chuyển nữa, cán bộ số 10 chẳng hạn. Bạn hãy khôi phục lại bản ghi này bằng lệnh:

```
go 10
recall
```

Bây giờ hãy xoá các bản ghi đã bị đánh dấu:

```
pack
```

Xin các bạn lưu ý:

Cũng có thể đánh dấu và khôi phục một loạt các bản ghi theo một chỉ tiêu nào đó. Ví dụ các bản ghi có LUONG lớn hơn hoặc bằng 400000 chẳng hạn:

delete all for LUONG>=400000

6.11. SAO CHÉP BẢN GHI

Hãy lấy ra danh sách riêng của phòng Kế hoạch để đưa xuống đơn vị đó bằng cách chép các bản ghi đó ra một tệp riêng.

copy to KH for upper(DONVI)='KH'

Một tệp có tên là KH.DBF đã được tạo ra để lưu trữ lương của phòng Kế hoạch.

6.12. TÍNH TỔNG SỐ TIỀN CÒN ĐƯỢC LĨNH

SUM CONLAI to A

Biến A sẽ nhận giá trị là tổng số tiền còn được nhận trong toàn đơn vị.

SUM CONLAI to A1 for upper(DONVI)='KH'

Biến A1 nhận giá trị là tổng số tiền còn được nhận trong phòng Kế hoạch. Có thể kiểm tra từng giá trị bằng lệnh:

?A

hay

?A1

Tương tự như vậy, hãy tính tổng số tiền thưởng, tạm ứng, phụ cấp trong toàn đơn vị và trong từng phòng.

6.13. TÍNH SỐ TIỀN THƯỞNG TRUNG BÌNH CỦA CÁN BỘ

Có thể kiểm tra xem số tiền thưởng trung bình của cán bộ là bao nhiêu bằng lệnh:

AVERAGE THUONG TO B1

Tương tự như trên, biến B1 nhận giá trị là số tiền thưởng

trung bình trong đơn vị. Song xin lưu ý: nếu trong tệp của bạn có các bản ghi trắng (bản ghi không có dữ liệu) thì phép tính trên không chính xác. Cũng có thể tính số tiền thưởng trung bình của một phòng nào đó:

AVERAGE THUONG TO B1 for upper(DONVI)='KH'

6.14. SẮP XẾP

Khi vào dữ liệu rất có thể các cán bộ trong một đơn vị không cùng một khu vực. Điều này sẽ khó khăn khi quan sát. Hãy sắp xếp các cán bộ cùng một đơn vị với nhau bằng lệnh:

```
SORT on DONVI/AC to SS
use SS
list
```

Cũng có thể sắp xếp theo cách khác:

```
INDEX on DONVI to SS
use LUONG INDEX SS
list
```

6.15. XOÁ TỆP

Sau khi tạo ra tệp SS.DBF để lưu trữ các bản ghi đã được sắp xếp theo đơn vị. Giả sử tệp LUONG không cần nữa, hãy xoá đi. *Xin bạn lưu ý*: một tệp CSDL chỉ được xoá đi khi nó đã được đóng lại.

```
use
delete file LUONG.DB
```

Chương 7
CÁC VÍ DỤ VỀ TÀI CHÍNH, KẾ TOÁN

7.1. TẠO TỆP CSDL

Tạo tệp CSDL để quản lý các chứng từ kế toán có cấu trúc sau:

Field name	Type	Width	Dec
MA	Character	1	
NGAY	Date	8	
SOCT	Numeric	4	
ND	Character	30	
TKN	Character	6	
MAKHN	Numeric	3	
TKC	Character	6	
MAKHC	Numeric	3	
TIEN	Numeric	12	
NGOAITE	Numeric	9	2
TYGIA	Numeric	5	
LOAINT	Character	5	
LOAI	Character	2	

Sau khi khai báo xong cấu trúc, hãy ấn Ctrl+W để ghi lại. Các trường trên có ý nghĩa như sau:

MA: để ghi nhận số dư đầu kỳ của một tài khoản hay tiểu khoản. Trường này chỉ nhận hai chữ cái: N (nếu là số

dư đầu kỳ ghi nợ) và C (nếu số dư đầu kỳ ghi có). Nếu các bản ghi là phát sinh trong kỳ thì trường này để trống.

NGAY: ghi ngày của chứng từ.

SOCT: ghi số chứng từ.

ND: ghi nội dung của chứng từ TKN: ghi tên tài khoản ghi nợ.

MAKHN: ghi mã khách hàng của tài khoản ghi nợ đối với các tài khoản (hay tiểu khoản) có quản lý theo mã khách hàng. Nếu các tài khoản (hay tiểu khoản) không quản lý theo mã thì trường này để trống.

TKC: ghi tên tài khoản ghi có.

MAKHC: ghi mã khách hàng của tài khoản ghi có. Trường này cũng giống như trường MAKHN.

TIEN: ghi số tiền (Việt nam đồng) của chứng từ.

NGOAITE: ghi số tiền ngoại tệ. Đối với các tài khoản (hay tiểu khoản) không phải là ngoại tệ thì trường này để trống.

TYGIA: ghi tỷ giá của ngoại tệ. Đối với các tài khoản không phải là ngoại tệ thì trường này để trống.

LOAINT: ghi loại ngoại tệ. *Xin bạn lưu ý*: các loại ngoại tệ phải đưa vào chuẩn cho từng loại ngoại tệ. Đối với các tài khoản không phải là ngoại tệ thì trường này để trống. Nếu bạn chỉ quản lý một loại ngoại tệ thì trường này không cần thiết.

LOAI: trường này chỉ dùng cho các tài khoản vay nợ (của cá nhân hay ngân hàng) nếu là vay thì ghi: V, nếu là trả nợ thì ghi TN, nếu là trả lãi thì ghi TL. Các tài khoản (hay tiểu khoản) khác thì để trống.

7.2. TẠO MỘT TỆP CHO THÁNG MỚI

Giả sử tháng trước là tháng một và đã có tệp CSDL ghi các chứng từ của tháng một là THANG1.DBF. Bây giờ bạn hãy sao cấu trúc để làm cho tháng hai.

```
use THANG1
copy structure to THANG2
```

7.3. NỐI SỐ DƯ ĐẦU KỲ

Giả sử ở tháng trước bạn đã tính được số dư và lưu ra tệp SDT1.DBF. Hãy đưa số dư này vào tệp THANG2.DBF.

```
use THANG2
append from SDT1
```

Các bạn hãy vào dữ liệu cho các chứng từ đã phát sinh trong tháng.

Xin bạn lưu ý:

Nếu bạn mới bắt đầu làm việc nghĩa là chưa có tệp lưu số dư đầu kỳ thì bạn có thể vào các số dư đầu kỳ cho tệp THANG2.

Các câu lệnh dưới đây chúng tôi đều coi là tệp THANG2.DBF đã mở.

7.4. XEM NỘI DUNG CỦA TỆP

Dùng lệnh:

```
List
```

hoặc:

```
Browse
```

Ấn Ctrl+F10 để màn hình được rộng hơn. Sau khi xem

xong, hãy ấn CTRL+W để kết thúc.

7.5. LỌC CÁC CHỨNG TỪ CỦA MỘT TÀI KHOẢN

Giả sử, bạn cần quan tâm đến các chứng từ của tài khoản 501. Lọc riêng các chứng từ của tài khoản này bằng lệnh:

```
set filter to tkc='501' or tkn='501'
browse
```

Sau khi xem xong, hãy hủy bỏ chế độ lọc dữ liệu bằng lệnh:

```
set filter to
```

7.6. TÍNH RA TIỀN VIỆT CÁC CHỨNG TỪ NGOẠI TỆ

Giả sử có một số chứng từ khi vào theo số tiền là ngoại tệ. Bây giờ hãy tính thành tiền Việt nam và đưa vào cột TIEN theo công thức:

```
TIEN=NGOAITE * TYGIA
```

Câu lệnh như sau:

```
replace all TIEN with NGOAITE * TYGIA;
                                for LOAINT#' '
```

Xin các bạn lưu ý:

Lệnh trên chỉ làm việc đối với các bản ghi có LOAINT#'', có nghĩa là chỉ làm việc đối với các bản ghi là ngoại tệ, bởi vì trường LOAINT ghi loại ngoại tệ của các phát sinh. Nếu là tiền VNĐ thì trường này để trống.

Để xem thông tin, dùng lệnh:

```
Browse
```

Khi xem xong, ấn CTRL+W để kết thúc.

7.7. KIỂM TRA SỐ DƯ ĐẦU KỲ

Bây giờ kiểm tra số dư đầu kỳ đã cân bằng giữa ghi nợ và ghi có chưa (số dư đầu kỳ giữa ghi nợ và ghi có bao giờ cũng phải bằng nhau), dùng lệnh:

```
sum TIEN to co for MA='C' sum TIEN to no for MA='N'
?co
?no
```

Kết quả sẽ thông báo tổng số tiền dư đầu kỳ của ghi nợ và ghi có.

7.8. KIỂM TRA SỐ TIỀN VAY, TRẢ LÃI, TRẢ NỢ TRONG THÁNG

Dùng lệnh:

```
sum TIEN to a for LOAI='V'
?a
```

Kết quả là số tiền vay trong tháng

```
sum TIEN to a for LOAI='TL'
?a
```

Kết quả là số tiền trả lãi trong tháng.

```
sum TIEN to a for LOAI='TN'
```

Kết quả là số tiền trả nợ trong tháng

7.9. KIỂM TRA TỔNG SỐ CHỨNG TỪ PHÁT SINH TRONG THÁNG

Dùng lệnh:

```
count to a for MA=' '
?a
```

Có thể kiểm tra tổng số chứng từ phát sinh của một tài

khoản. Tài khoản 1111 chẳng hạn.

```
count to b for (TKN='1111' or TKN='1111') and MA=' '
?b
```

Xin các bạn lưu ý:

Bạn phải chỉ thị thêm and MA=' ', nếu không máy sẽ đếm cả các bản ghi là số dư đầu kỳ, vì như chúng tôi đã giới thiệu với các bạn, trường MA (mã) ghi loại số dư đầu kỳ (dư nợ hoặc dư có. Nếu là phát sinh trong kỳ thì trường này để trống mà không nhập dữ liệu.

7.10. KIỂM TRA CÁC CHỨNG TỪ CỦA MỘT KHÁCH HÀNG

Giả sử cần kiểm tra các phát sinh của khách hàng số 3 chẳng hạn.

```
set filter to MAKHN=3 or MAKHC=3
browse
```

Các bạn cũng có thể kiểm tra số tiền ghi có và ghi nợ của khách hàng này.

```
sum TIEN to no for MAKHN=3
?no
sum TIEN to co for MAKHC=3
?co
```

Bạn phải hủy bỏ chế độ lọc bằng lệnh:

```
set filter to
```

7.11. XOÁ TOÀN BỘ SỐ DƯ, MỘT SỐ BẢN GHI

a. Xoá toàn bộ số dư

Giả sử đã làm tháng hai nhưng lại phát hiện có sai sót ở

tháng một. Đương nhiên là bạn phải tính lại số dư cuối kỳ của THANG1, và như vậy bạn phải xoá đi các số dư đầu kỳ của THANG2 để đưa số dư cuối kỳ của tháng một vừa được tính lại vào.

```
delete all for ma#' '
```

Nhìn chung bạn nên dùng lệnh:

```
browse
```

để kiểm tra trước khi dùng lệnh:

```
pack
```

để xoá hẳn. Bây giờ bạn hãy nối lại số dư vào tệp THANG2 như chúng tôi đã giới thiệu ở Mục 7.2.

b. Xoá đi một số bản ghi

Giả sử, cần xoá đi một vài bản ghi, bạn có thể tiến hành bằng một trong hai cách sau:

+ *Cách 1*

Trước hết phải xác định xem bản ghi đó có số thứ tự bao nhiêu bằng lệnh:

```
display all fields NGAY,SOCT,ND,TKC,TKN,TIEN
```

Giả sử cần xoá đi bản ghi có số thứ tự là 50 chẳng hạn. Bạn hãy tiến hành như sau:

```
go 50
delete
pack
```

+ *Cách 2*. Dùng lệnh:

```
browse
```

Sau đó đưa con trỏ đến bản ghi cần xoá và ấn Ctrl+T. Nếu còn bản ghi cần xoá thì tiếp tục đưa con trỏ đến bản ghi đó và

ấn Ctrl+T. Khi thực hiện lệnh này, bên trái của bản ghi xuất hiện một vạch đen để thông báo: bản ghi này đã bị đánh dấu. Khi nhấn Ctrl+T trên các bản ghi này thì vạch đen đánh dấu sẽ không còn nữa. Bản ghi đã không bị đánh dấu xoá. Ấn Ctrl+W để ghi lại và tại khung cửa sổ lệnh, ấn lệnh:

```
pack
```

7.12. KIỂM TRA NGOẠI TỆ

Giả sử cần kiểm tra phát sinh của một loại ngoại tệ, USD chẳng hạn, bạn dùng lệnh như sau:

```
set filter to LOAINT='USD'
browse
set filter to
```

7.13. SẮP XẾP CÁC CHỨNG TỪ

Giả sử cần sắp xếp theo tài khoản ghi có để tiện theo dõi, dùng lệnh như sau:

```
sort on TKC/a to ss
use ss
browse
```

Phần 3

CHƯƠNG TRÌNH FOXPRO

Chương 8
GIỚI THIỆU CHƯƠNG TRÌNH FOXPRO

8.1. CẤU TRÚC MỘT CHƯƠNG TRÌNH FOXPRO

Mỗi một chương trình FOXPRO là một tệp chứa một hoặc nhiều dòng văn bản có quan hệ với nhau. Bạn có thể ghép hai hoặc nhiều dòng vật lý thành một dòng logic bằng cách sử dụng dấu chấm phẩy (;) ở phía cuối dòng. Tuy nhiên, việc này sẽ trở nên sai sót nếu dấu chấm phẩy ở vị trí chưa kết thúc chuỗi ký tự.

Mỗi dòng logic có thể chứa một động từ lệnh đơn lẻ. Theo sau động từ này có thể có các mệnh đề bổ sung và lời giải thích nếu cần. Bạn có thể tự do pha trộn các dòng trống với các dòng lệnh. Đặt lệnh bắt đầu từ vị trí nào đó trên dòng lệnh là tùy thuộc người sử dụng. Các lệnh NOTE và * ở đầu dòng lệnh giúp bạn biến dòng đó thành dòng giải thích và đương nhiên là muốn ghi gì cũng được. Trong trường hợp muốn ghi lời giải thích ở cuối dòng lệnh bạn phải sử dụng ký hiệu &&. Nếu lời giải thích bắt đầu ở đầu dòng lệnh với lệnh NOTE hoặc * và ở cuối dòng có dấu chấm phẩy thì dòng phía dưới cũng được coi là dòng chú thích.

Ngoài một ít ngoại lệ, mỗi lệnh đều bắt đầu bằng một động từ (như COPY, USE....). Cú pháp của mỗi động từ sẽ xác định mệnh đề nào có thể theo sau động từ lệnh trên dòng lệnh. Hai dạng thay thế của lệnh GOTO và STORE là hai

ngoại lệ.

Một số nguyên đầu dòng sẽ thay cho lệnh GOTO. Ví dụ trên dòng lệnh viết:

2

tương đương với việc viết lệnh:

GOTO 2

và dấu = ám chỉ một lệnh STORE. Ví dụ

A=3

tương đương với lệnh:

STORE 3 TO A

Ngoài những trường hợp ngoại lệ này, các lệnh của FOXPRO trên một dòng lệnh luôn có dạng sau:

<Động từ lệnh> [Mệnh đề bổ sung] [Lời chú thích]

Có nhiều động từ lệnh bao gồm hai hay ba từ khoá. Ví dụ: DEFINE POPUP, MODIFY COMMAND... Trong một số trường hợp, bạn có thể lặp lại các mệnh đề mà chẳng gây sự cố gì. Ví dụ:

APPEND BLANK BEFORE BLANK BEFORE

Có một số lệnh như ENDDO, ENDIF chẳng hạn cho phép các chú thích sau chúng mà không cần thêm &&. Mâu thuẫn này tồn tại để tương thích với dBASE III.

Thông thường bạn có thể chỉ định vị trí cho một số mệnh đề (như FOR <bt logic>, TO <tên tệp> chẳng hạn) theo một thứ tự bất kỳ. Do đó các ví dụ dưới đây là như nhau:

COPY TO LUU ALL FOR DONVI='KH'

COPY ALL TO LUU FOR DONVI='KH'

COPY ALL FOR DONVI='KH' TO LUU

Với tư cách là di sản của dBASE II, FOXPRO cũng cho

phép bạn cắt khúc các lệnh, các hàm và các từ khoá, chỉ cần bốn ký tự đầu. Có thể viết DISP thay cho DISPLAY, MODI COMM thay cho MODIFY COMMAND. Tuy nhiên bạn không được viết sai. Chẳng hạn nếu bốn ký tự đầu là đúng lệnh gốc, nhưng ký tự phía sau viết sai thì vẫn bị coi là sai.

Ví dụ: DISP, DISPL là đúng, nhưng DISPLY là sai tuy rằng đã đúng tới năm ký tự đầu (đúng ra phải viết là DISPLAY).

8.2. TẠO MỘT TỆP CHƯƠNG TRÌNH

a. Dùng lệnh MODIFY COMMAND

Tại khung cửa sổ lệnh, bạn hãy ra lệnh:

MODIFY COMMAND <tên chương trình>

<tên chương trình> được đặt theo quy định của DOS. Nếu không tạo trong thư mục hiện thời, bạn phải chỉ thị cả đường dẫn. Bạn không cần ghi phần mở rộng, FOXPRO tự gán là PRG.

Sau lệnh này sẽ xuất hiện một màn hình soạn thảo. Trong trạng thái ngầm định, màn hình soạn thảo là một phần của màn hình ở góc trên bên trái. Bạn có thể mở rộng toàn màn hình để soạn thảo bằng cách ấn Ctrl+F10. Sau khi soạn thảo xong nội dung chương trình mà bạn đang quan tâm, bạn hãy ấn Ctrl+W để ghi lại. Con trỏ trở về khung của sổ lệnh.

Ví dụ:

Modify command LUONG

b. Dùng bảng chọn

Ấn F10, chọn F, chọn Program.

Xin bạn lưu ý: Trong các ví dụ dưới đây chúng tôi viết: 'Nội dung chương trình' và sau đó là các câu lệnh của chương trình, có nghĩa là bạn phải dùng một trong hai cách trên để tạo tệp chương trình sau đó mới viết các câu lệnh. Nếu các bạn viết các câu lệnh ở khung cửa sổ lệnh thì câu lệnh sẽ trở nên vô nghĩa.

8.3. THỰC HIỆN CHƯƠNG TRÌNH

a. Dùng lệnh DO ở khung cửa sổ lệnh

DO <tên chương trình>

trong đó: <tên chương trình> là tên của chương trình mà bạn cần thực hiện. Nếu chương trình không thuộc thư mục hiện thời hoặc thư mục chứa chương trình không được khai báo bằng lệnh SET PATH, bạn phải chỉ thị cả đường dẫn.

Ví dụ:

DO LUONG

b. Dùng bảng chọn

Ấn F10, chọn Program, chọn DO.

8.4. SỬA CHƯƠNG TRÌNH

Khi thực hiện chương trình, khi dịch đến dòng lệnh bị sai về cú pháp, FOXPRO sẽ dừng lại tại dòng đó và không dịch tiếp. Bạn phải sửa, ghi lại và thực hiện chương trình đến khi không thấy thông báo sai sót thì thôi. Song bạn phải thật chú ý: không phải cứ đưa ra kết quả nghĩa là bạn đã hoàn thành công việc. Ví dụ trong chương trình, đúng ra bạn phải viết là a+b thì bạn lại viết là a-b, chương trình vẫn thực hiện "êm

ru", nhưng thực ra kết quả không thể chấp nhận được. Điều này rất hay xảy ra đối với các bạn mới lập chương trình. Có lẽ vì quá bận tâm đến xử lý lỗi do máy thông báo nên đã thở phào khi chương trình đưa ra kết quả.

Nhìn chung, bạn nên kiểm tra lại kết quả trước khi quyết định chấp nhận nó. Khi cần sửa, tại khung cửa sổ lệnh bạn hãy ra lệnh:

MODIFY COMMAND <tên chương trình>

Xin bạn lưu ý: Nếu trong chương trình, bạn cần đưa vào các lời giải thích hoặc ghi chú một điều gì đó thì bạn hãy đặt thêm dấu sao (*) vào đầu các dòng này. Khi thực hiện chương trình, FOXPRO tự động bỏ qua các dòng có dấu sao (*) ở đầu như đã nói ở trên.

8.5. TẠO TỆP CONFIG.FP

Tệp CONFIG.FP là một tệp rất đặc biệt: sau khi khởi động FOXPRO, máy đã thực hiện ngay các lệnh của tệp này. Tệp này gồm các lệnh khai báo các tham số cho FOXPRO. Ví dụ như màu màn hình, đường dẫn tới các thư mục, số biến dùng tối đa... Bạn có thể dùng bất kỳ hệ soạn thảo nào để tạo ra tệp CONFIG.FP, thậm chí có thể dùng ngay lệnh Modify command của FOXPRO để tạo tệp này. Trong trường hợp bạn dùng lệnh Modify command, bạn phải chỉ thị rõ:

Modify command CONFIG.FP

Các câu lệnh thường dùng cho tệp CONFIG.FP là:

+ Files=<bt số>

trong đó <bt số> quy định số Files được mở đồng thời khi sử dụng FOXPRO.

+ COLOR=<mã màu> quy định màu khi khởi động FOXPRO.

+ DECIMAL=<bt số> trong đó <bt số> nhận giá trị từ 0 đến 14 (ngầm định là 2) quy định số chữ số thập phân trong các phép tính

Xin bạn lưu ý: các giá trị của trường số không tuân thủ theo quy định này.

+ PATH = <đường dẫn> khai báo các thư mục để FOXPRO tìm các tệp khi không tìm thấy tệp đó trong thư mục hiện thời.

+ TALK = ON/OFF đặt chế độ có (ON) trả lời kết quả trung gian hay không (OFF). Ngầm định là ON.

+ SAFETY = ON/OFF đặt chế độ có thông báo khi có sự ghi đè tệp (ON) hay tự động ghi đè (OFF). Ngầm định là ON.

+ MVCOUNT = <bt số> <bt số> nhận giá trị từ 128 đến 3600, quy định số biến được dùng tối đa.

8.6. CHƯƠNG TRÌNH CON TRONG FOXPRO

a. Giới thiệu chung

Chương trình con là một chương trình được một chương trình khác gọi bằng lệnh:

DO <tên chương trình>

- Cú pháp chung:

<chương trình chính>
<dãy các lệnh>
DO <chương trình con 1>
[<dãy các lệnh>]
[DO <chương trình con 2>]

- Truyền tham số:

Trường hợp thường gặp là không những chương trình chính gọi thực hiện chương trình con, mà còn truyền tham số cho nó, tức là giao cho chương trình con xử lý các tham số được truyền này.

Trong trường hợp này, ở chương trình chính khi gọi chương trình con phải có thêm lệnh: WITH <danh sách tham số> và ở chương trình con phải có lệnh:

PARAMETER <danh sách tham số>.

Các tham số được cách nhau một dấu phẩy (,) và giá trị các tham số ở chương trình chính được truyền lần lượt vào các tham số khai báo ở chương trình con.

- Thủ tục: Thủ tục là một kiểu chương trình con đặc biệt. Thủ tục bắt đầu bằng lệnh:

PROCEDURE <tên thủ tục>

và kết thúc bằng lệnh:

RETURN

Nhiều <thủ tục> được ghép thành một tệp gọi là <tệp thủ tục>, có kiểu là .PRG.

Để thực hiện một thủ tục trong tệp thu tục, bạn phải khởi động tệp thủ tục bằng lệnh:

SET PROCEDURE TO <tên tệp thủ tục>

Khi thực hiện một thủ tục trong một tệp thủ tục đã được khởi động, bạn ra lệnh:

DO <tên thủ tục>

Bạn có thể gọi thực hiện một tệp thủ tục từ một chương trình nằm ngoài tệp thủ tục hoặc từ một thủ tục khác cùng nằm trong tệp thủ tục với tệp thủ tục cần thực hiện. Trong

một lần thực hiện chương trình, một thủ tục có thể được gọi thực hiện nhiều lần với các tham số khác nhau.

Nhìn chung bạn nên ghép nhiều chương trình con lại thành một tệp thủ tục vì khi mở tệp thủ tục thì các thủ tục đã được nạp vào bộ nhớ trong và khi gọi thực hiện thì tốc độ xử lý sẽ nhanh hơn so với gọi từng chương trình con riêng rẽ.

b. Các ví dụ

(1). *Ví dụ 1*: Tên chương trình VIDU1.PRG

Nội dung chương trình

```
* Mở tệp thủ tục VIDU1*
   set procedure to VIDU1
   set talk off
   set scorebroad off
   store 0 to tuoi1,tuoi2
   @10,20 say 'Tuổi người thứ nhất:' get tuoi1;
                                    picture'99'
   @11,20 say 'Tuổi người thứ hai:' get tuoi2;
                                    picture'99'
   Read
   Clear
   do case
      case tuoi1=tuoi2
         DO THUTUC1
      case tuoi1>tuoi2
         DO THUTUC2
      case tuoi1<tuoi2
         DO THUTUC3
```

```
      endcase
      set talk off
   set scorebroad
   close all
   return
* * *
   PROCEDURE THUTUC1
      @10,20 say'Tuổi người thứ nhất bằng'+;
                                 'tuổi người thứ hai'
   RETURN
   * * *
   PROCEDURE THUTUC2
      @10,20 say 'Người thứ nhất nhiều tuổi'+;
                                 'hơn người thứ hai'
   RETURN
   * * *
   PROCEDURE THUTUC3
      @10,20 say 'Người thứ nhất ít tuổi hơn'+:
                                 'người thứ hai'
   RETURN
   * * *
```

- Giải thích: Chương trình trên giúp bạn dễ thấy cách tổ chức chương trình con như thế nào. Nếu TUOI1 =TUOI2 sẽ thực hiện chương trình con THUTUC1 (PROCEDURE THUTUC1) bằng lệnh DO THUTUC1. Các trường hợp khác: TUOI1 > TUOI2, và TUOI1 < TUOI2, hoàn toàn tương tự.

(2). *Ví dụ 2*: VIDU2.PRG

Nội dung chương trình

```
set talk off
set procedure to VIDU2
clear
store 0 to SO1, SO2
@ 10,20 say 'Vào số thứ nhất: ' get SO1
@ 11,20 say 'Vào số thứ hai: ' get SO2
read
DO THUTUC1 WITH SO1,SO2
set talk off
close all
RETURN
* * *
PROCEDURE THUTUC1
Parameters x,y
   KETQUA = x*y
   DO THUTUC2 WITH KETQUA
RETURN
* * *
PROCEDURE THUTUC2
Parameters IN
clear
@ 10,20 say 'Kết quả là: '+str(IN,10,2)
RETURN
* *
```

- Giải thích:

Trong chương trình chính, sau khi vào hai giá trị SO1, SO2 thì gọi thực hiện THUTUC1 (PROCEDURE THUTUC1) và truyền hai tham số SO1, SO2 cho thủ tục này. Lúc này x

nhận giá trị của SO1, y nhận giá trị của SO2. Trong thủ tục THUTUC1 tính KETQUA=x*y thực chất là tính:

KETQUA=SO1*SO2

Câu lệnh DO THUTUC2 WITH KETQUA để gọi thủ tục THUTUC2 và truyền giá trị KETQUA cho biến IN. Hay nói cách khác, biến IN sẽ nhận giá trị của biến KETQUA. Như vậy trong một thủ tục có thể gọi thực hiện một thủ tục khác.

Chương 9
CẤU TRÚC ĐIỀU KIỆN, CẤU TRÚC LẶP VÀ CÁC CHƯƠNG TRÌNH ỨNG DỤNG

9.1. CẤU TRÚC IF...ENDIF

Cấu trúc IF...ENDIF có dạng sau:

```
IF <bt logic>
        <dãy các lệnh 1>
   ELSE
        <dãy các lệnh 2>
ENDIF
```

Giải thích:

- Nếu <bt logic> nhận giá trị .T.(đúng), thực hiện <dãy các lệnh 1> và sau đó thực hiện các lệnh sau ENDIF (nếu có).

- Nếu <bt logic> nhận giá trị .F.(sai), thực hiện <dãy các lệnh 2> (bỏ qua <dãy các lệnh 1>) và sau đó thực hiện các lệnh sau ENDIF (nếu có).

Chú ý: Trong cấu trúc này có thể không có chỉ thị:

```
ELSE
    <dãy các lệnh 2>
```

Trong trường hợp này, nếu <bt logic> nhận giá trị .T. thì thực hiện <dãy các lệnh 1>, nếu <bt logic> nhận giá trị .F. thì không thực hiện lệnh nào trong cấu trúc mà thực hiện ngay các lệnh sau ENDIF (nếu có) .

Có thể có nhiều cấu trúc IF... ENDIF lồng nhau, nhưng không được cắt nhau, nghĩa là cấu trúc ngoài phải bao gọn cấu trúc trong.

9.2. CÁC VÍ DỤ VỀ CẤU TRÚC IF... ENDIF

a. Chương trình 1

Nhập một số tính căn bậc hai của số này:

Nội dung chương trình

```
clear
set talk off
Input 'Số cần tính căn bậc hai:'to so
If so>=0
   ?'Căn bậc hai của '.so.' la:'.SQRT(so)
   Else
      ?'Số của bạn không thể tính căn bậc hai được'
Endif
```

b. Chương trình 2: CT12.PRG

Giải phương trình bậc hai: $AX^2+BX+C = 0$

Nội dung chương trình

```
Set talk off
Clear
Public delta
Stor 0 to delta
Input 'cho hệ số a: ' to a
Input 'cho hệ số b: ' to b Input 'cho hệ số c: ' to c
If a=0
      Clear
      @10,20 say 'Không phải PT bậc 2'    :
```

```
      Else
         Delta=b*b-4*a*c
      If Delta<0
         @ 10,20 say 'phương trình vô nghiệm'
      Else
      If Delta=0
         Clear
         @ 10,20 say 'PT có nghiệm kép'
         @ 11,20 say 'x1=x2='+Str(-b/(2*a),7,2)
      Endif
      If Delta>0
         Clear
         @ 10,20 say 'PT có 2 nghiệm phân biệt'
         @ 11,20 say 'x1='+;
                     Str((-b+sqrt(delta))/(2*a),7,2)
         @ 12,20 say 'x2='+;
                     Str((-b-sqrt(delta))/(2*a),7,2)
      Endif
   Endif
Endif
set talk on
Return
```

9.3. CẤU TRÚC DO CASE .... ENDCASE

a. Cú pháp

```
DO CASE
   CASE <bt logic 1>
      <dãy các lệnh 1>
```

```
    CASE <bt logic 2>
       <dãy các lệnh 2>
       .....
    CASE <bt logic n> <dãy các lệnh n>
  OTHERWISE
    <dãy các lệnh n+1>
  ENDCASE
```

Giải thích:

Máy sẽ lần lượt kiểm tra các <bt logic>. Nếu <bt logic> nhận giá trị .T. (đúng), máy thực hiện <dãy các lệnh> tương ứng và sau đó thực hiện các lệnh sau ENDCASE (nếu có). Nếu <bt logic> nhận giá trị .F. (sai), máy kiểm tra <bt logic> tiếp theo. Nếu tất cả các <bt logic> đều nhận giá trị .F., máy thực hiện dãy các lệnh sau OTHERWISE.

b. Chú ý

(1). Trong cấu trúc trên không bắt buộc phải có chỉ thị OTHERWISE. Trong trường hợp này nếu tất cả các <bt logic> đều nhận giá trị .F. (sai), máy không thực hiện lệnh nào cả.

(2). Nếu có nhiều <bt logic> nhận giá trị .T. (đúng), máy chỉ thực hiện <dãy các lệnh> tương ứng với <bt logic> đầu tiên thoả mãn, nghĩa là nhiều nhất cũng chỉ một dãy các lệnh được thực hiện.

(3). Nếu có nhiều cấu trúc DO CASE...ENDCASE bao nhau thì cấu trúc ngoài phải bao gọn cấu trúc trong, nghĩa là không được cắt nhau.

9.4. CẤU TRÚC DO WHILE ...ENDDO

a. Cú pháp

DO WHILE <bt logic>
    <dãy các lệnh 1>
  [LOOP]
    <dãy các lệnh 2>
  [EXIT]
    <dãy các lệnh 3>
ENDDO

Giải thích:

- Nếu <bt logic> nhận giá trị .T. (đúng), máy sẽ thực hiện các lệnh nằm giữa DO WHILE và ENDDO sau đó kiểm tra lại <bt logic>. Nếu <bt logic> còn nhận giá trị .T. thì sẽ tiếp tục thực hiện các lệnh đến khi nào <bt logic> nhận giá trị .F.

- Nếu gặp lệnh LOOP, máy sẽ quay trở lại đầu vòng lặp mà không thực hiện các lệnh nằm giữa LOOP và ENDDO.

- Nếu gặp lệnh EXIT, ra khỏi vòng lặp mà không thực hiện <dãy các lệnh 3>.

b. Chú ý

- Có thể không có lệnh LOOP và EXIT trong vòng lặp.

- Nếu có nhiều cấu trúc DO WHILE... ENDDO bao nhau thì cấu trúc ngoài phải bao gọn cấu trúc trong, nghĩa là không được cắt nhau.

9.5. CHƯƠNG TRÌNH FOXPRO SỬ DỤNG CẤU TRÚC LẶP DO WHILE ... ENDDO

Tính tổng của 1000 số tự nhiên đầu tiên.

Nội dung chương trình

```
Set talk off
Clear
tong=0
i=1
Do While i<=1000
   tong=tong+i
   i= i+1
Enddo
@10,20 say 'Tổng của 1000 số tự nhiên đầu tiên là:'
@11,30 say tong
Return
```

Thực hiện chương trình

```
Tổng của 1000 số TN đầu đầu tiên là:
      50500
```

9.6. CẤU TRÚC FOR...ENDFOR

a. Cú pháp

```
FOR <biến nhớ>=<btsố 1> to <btsố 2> [STEP<bt số3>]
      <dãy lệnh 1>
      [<LOOP>]
      [<dãy lệnh 2>]
      [<EXIT>]
      [<dãy lệnh 3>]
ENDFOR
```

Lệnh FOR... ENDFOR thi hành một nhóm lệnh nằm giữa FOR và ENDFOR. trong đó FOR là đầu vòng lặp và ENDFOR là cuối vòng lặp.

<Biến nhớ> nhận giá trị ban dầu là <bt số 1> và chạy đến

<bt số 2> (là giá trị kết thúc). <bt số 3> là bước nhảy sau mỗi lần thi hành các lệnh. Nếu bạn không chỉ thị <bt số 3> thì ngầm định là tăng lên một sau mỗi lần thi hành các lệnh. Trong cấu trúc này bạn có thể sử dụng các lệnh:

Lệnh LOOP để quay lại ban đầu mà không thực hiện các lệnh sau LOOP. Tuy nhiên <biến nhớ> vẫn tăng <hay giảm> coi như đã thực hiện xong một vòng lặp (coi như đã dùng tới ENDFOR).

Lệnh EXIT để thoát khỏi vòng lặp cho dù <biến nhớ> chưa đạt đến <bt số 2>.

b. Ví dụ

```
Clear
Set talk off
For i=1 To 10
   ? i
Endfor
Set talk on
Return
```

9.7. CẤU TRÚC SCAN... ENDSCAN

a. Cú pháp

```
SCAN [<phạm vi>]
   [FOR <bt logic>] [WHILE <bt logic>]
      <dãy các lệnh 1>
   LOOP
      <dãy các lệnh 2>
   EXIT
      <dãy các lệnh 3>
```

ENDSCAN

Cấu trúc này sẽ đọc từng bản ghi của tệp CSDL trong giới hạn do <phạm vi> qui định. Nếu không có phạm vi thì coi như toàn bộ bản ghi được xét đến.

Các lệnh LOOP và EXIT cũng giống như trong lệnh DO WHILE... ENDDO.

Xin bạn lưu ý:

Trước khi sử dụng lệnh SCAN, bạn phải đưa con trỏ về đầu <phạm vi>.

b. Ví dụ

Giả sử có một tệp CSDL: LUONG.DBF quản lý lương của cơ quan. Bạn hãy in ra danh sách những cán bộ có lương chính (LCHINH) lớn hơn hoặc bằng 39000 và quê quán (QUEQUAN) ở Nam Định.

Nội dung chương trình

```
clear
set talk off
use LUONG INDEX ON QUEQUAN TO CSO
seek 'Nam Định'
scan for LCHINH>=39000;
                        while QUEQUAN='Nam Định'
   ? HOTEN, DONVI, LCHINH, QUEQUAN
Endscand
use
set talk on
Return
```

Nếu bạn muốn dùng cấu trúc lặp DO WHILE...ENDDO để viết chương trình trên thì phải viết như sau:

```
set talk off
clear
USE LUONG INDEX ON QUEQUAN TO CSO
seek 'Nam Định'
Do while QUEQUAN= 'Nam Định'
   if LCHINH >=39000
         ? HOTEN, DONVI, LCHINH, QUEQUAN,
   endif
   skip
Enddo
Use
Set talk on
Return
**************
```

Các bạn có thể dễ dàng thấy rằng cấu trúc SCAN...ENDSCAN có số lệnh ít hơn và chương trình dễ đọc hơn. Trong thực tế, với các bài toán như trên thì dùng cấu trúc SCAN...ENDSCAN chương trình sẽ thực hiện nhanh hơn so với cấu trúc DO WHILE...ENDDO.

Xin bạn lưu ý:

- Cấu trúc SCAN...ENDSCAN chỉ dùng cho tệp CSDL trong khi cấu trúc DO WHILE...ENDDO có thể dùng để lặp lại bất kỳ một nhóm lệnh nào trong chương trình.

- Trong cấu trúc SCAN...ENDSCAN không có lệnh SKIP, vì ngầm định là luôn tăng lên một, nghĩa là khi đọc xong một bản ghi thì luôn tăng đến bản ghi tiếp theo chứ không lùi được. Trong các trường hợp này, bạn phải dùng cấu trúc lặp DO WHILE...ENDDO.

Chương 10

HÀM TỰ TẠO TRONG CHƯƠNG TRÌNH FOXPRO

FOXPRO 2.6 đã cung cấp một hệ thống các hàm rất phong phú cho phép người sử dụng lập các chương trình ngắn gọn, sáng sủa và nhanh chóng thu được các kết quả ứng dụng. Song trong một số trường hợp có nhiều đoạn chương trình, một số các câu lệnh được lập đi lập lại nhiều lần. Foxpro cho phép kết nối chúng lại thành một hàm gọi là hàm do người dùng tự tạo (gọi tắt là hàm tự tạo).

10.1. HÀM TỰ TẠO LÀ GÌ ?

Hàm tự tạo chẳng qua là một chương trình con độc lập gồm nhiều lệnh Foxpro và mang những đặc tính tương tự như hàm cài đặt sẵn do Foxpro cung cấp. Hàm tự tạo trả về chương trình gọi nó một giá trị duy nhất thuộc một kiểu dữ liệu nào đó, trong khi đó chương trình con hoặc thủ tục (Procedure) thực hiện một công việc nào đó mà kết quả thu được giống như một chương trình độc lập.

10.2. XÂY DỰNG MỘT HÀM TỰ TẠO VÀ CÁCH SỬ DỤNG

Hàm tự tạo được bắt đầu bởi lệnh FUNCTION kèm theo tên hàm theo cú pháp sau:

FUNCTION <tên hàm>

trong đó tên hàm dài tối đa 10 ký tự bắt đầu bởi một chữ cái hoặc dấu gạch dưới và tiếp theo là bất kỳ ký tự nào trừ ký tự trống, dấu chấm phẩy (;)...

Nếu hàm tự tạo cần đến dữ liệu của chương trình gọi nó thì lệnh đi liền sau lệnh FUNCTION phải là lệnh PARAMETER như sau:

PARAMETER <các tham số>

Hàm tự tạo được kết thúc bởi:

- Gặp một lệnh Function khác
- Gặp EOF hoặc
- Gặp lệnh Return

Nếu không ghi Return thì Foxpro tự động coi như có và sẽ trả về giá trị logic .T. cho chương trình gọi nó. Nếu có lệnh Return thì ghi theo cú pháp sau:

Return [<bthức>]

trong đó <bthức> là trị duy nhất mà hàm sẽ nhận giá trị và trả cho chương trình triệu gọi hàm.

Khi đặt tên cho hàm tự tạo, bạn không được đặt tên trùng với tên của hàm gài sẵn của Foxpro vì khi thi hành thì hàm gài sẵn bao giờ cũng được Foxpro cho thi hành trước. Do đó nếu trùng tên với hàm cài sẵn thì hàm tự tạo sẽ không bao giờ được thi hành.

Nếu <bthức> sau lệnh Return không cùng với kiểu dữ liệu cần thiết mà chương trình chính sử dụng thì sẽ bị Foxpro thông báo lỗi.

Ví dụ:

```
*** Chương trình chính
set talk off
```

```
clear
@10,20 say 'Kết quả của phép toán: '+TCONG()
set talk on
return
************
*** Hàm tự tạo FUNCTION TCONG
    SO1=10
    SO2=20
return SO1+SO2
***********
```

Chương trình này sẽ bị thông báo lỗi vì kết quả của hàm là dữ liệu kiểu số (là 10 vì chính là giá trị của SO1+SO2). Vì vậy chương trình phải sửa lại bằng một trong hai cách như sau:

- *Cách 1*: Sửa ở chương trình chính

```
*** Chương trình chính
set talk off
clear
@10,20 say 'Kết quả của phép toán: '+STR(TCONG(),3)
set talk on
Return
************
*** Hàm tự tạo FUNCTION TCONG
SO1=10
SO2=20
Return SO1+SO2
***********
```

- *Cách 2*: Sửa ở hàm tự tạo

```
*** Chương trình chính
set talk off
clear
@10,20 say 'Kết quả của phép toán: '+TCONG()
set talk on
Return
************
*** Hàm tự tạo FUNCTION TCONG
SO1=10
SO2=20
Return STR(SO1+SO2,3)
***********
```

10.3. TRUYỀN THAM SỐ CHO HÀM TỰ TẠO

Như chúng tôi đã nói ở trên, hàm tự tạo có thể được truyền các tham số ở chương trình chính gọi gần giống như khi gọi chương trình con. Với ví dụ trên chúng ta có thể sửa lại như sau:

```
*** Chương trình chính
set talk off
clear
@10,20 say 'Kết quả của phép toán:';
              +STR(TCONG(10,20),3)
set talk on
Return
************
*** Hàm tự tạo FUNCTION TCONG
PARAMETER SO1,SO2
```

Return SO1+SO2
****

Khi bạn thực hiện chương trình, kết quả nhận được như sau: Kết quả của phép toán: 30

Bởi vì số 10 đã được truyền cho tham số SO1, số 20 đã được truyền cho tham số SO2 và giá trị của hàm là 30 (do lệnh return SO1+SO2).

Xin các bạn lưu ý:

- Số giá trị của hàm truyền cho các tham số có thể ít hơn số tham số nằm sau lệnh PARAMETERS, trong trường hợp này các tham số còn dôi ra trên lệnh PARAMETERS sẽ nhận giá trị logic .F.

- Nếu số giá trị cần chuyển cho các tham số lớn hơn số tham số nằm sau lệnh PARAMETERS thì Foxpro thông báo lỗi: Wrong number of parameters

10.4. LỆNH SET UDFPARMS

Lệnh SET UDFPARMS (viết tắt chữ UDF Parameter), xác định các biến được trao cho hàm tự tạo theo trị (value) hay theo quy chiếu (Reference). Theo quy ước ngầm định thì các biến được chuyển cho hàm theo trị. Điểm khác nhau giữa hai cách chuyển giá trị này như sau:

- Nếu chuyển theo trị thì giá trị của biến có thể bị hàm làm thay đổi ngay trong hàm nhưng giá trị nguyên thủy của nó trong chương trình chính gọi sẽ không bị thay đổi.

- Nếu chuyển theo quy chiếu và nếu giá trị của biến bị hàm làm thay đổi ngay trong hàm thì giá trị nguyên thủy

của biến trong chương trình chính cũng bị thay đổi theo.

Lệnh SET UDFPARMS có cú pháp như sau:

SET UDFPARMS TO VALUE/REFERENCE

Nếu SET UDFPARMS TO VALUE thì truyền theo trị, còn nếu SET UDFPARMS TO REFERENCE thì truyền theo quy chiếu.

Xin các bạn hãy tìm hiểu các ví dụ dưới đây:

```
*** Chuyển giá trị của biến theo trị
clear
set talk off
wait 'Hãy ấn một phím bất kỳ:'
set udfparm to value
store 1 to BIEN
***Giá trị của biến không thay đổi
@8,10 say 'Chuyển giá trị theo trị'
@10,10 say 'Giá trị của hàm: '+str(CONG(BIEN),3)
@12,10 say 'Giá trị của BIEN:'+str(BIEN,3)
****Chuyển giá trị theo quy chiếu
clear
set talk off
wait 'Hãy ấn một phím bất kỳ:'
set udfparm to reference
store 1 to BIEN
***Giá trị của biến thay đổi
@8,10 say 'Chuyển giá trị theo quy chiếu'
@10,10 say 'Giá trị của hàm:'+str(CONG(BIEN),3)
@12,10 say 'Giá trị của BIEN:'+str(BIEN,3)
set talk on
```

```
Return
**********
***Hàm tự tạo function CONG
parameters x
x=x+1
return x
***********
```

Khi thực hiện chương trình thì có kết quả như sau:

Chuyển giá trị theo trị

Giá trị của hàm: 2

Giá trị của BIEN: 1

Và trong trường hợp tiếp theo như sau:

Chuyển giá trị theo trị quy chiếu

Giá trị của hàm: 2

Giá trị của BIEN: 2

Trong trường hợp thứ hai khi hàm thay đổi giá trị của BIEN giá trị của nó trong chương trình chính cũng bị thay đổi theo (trong trường hợp thứ nhất không thay đổi giá trị là 1 của BIEN).

Cũng có thể chuyển các tham số mà không cần dùng lệnh SET UDFPARM TO... như sau:

+ Nếu chuyển theo trị thì cho đặt biến trong cặp dấu ngoặc.

Ví dụ CONG(BIEN) trong đó CONG là tên hàm tự tạo còn biến là tên BIEN được truyền cho hàm tự tạo.

+ Nếu chuyển theo kiểu quy chiếu thì đặt biến trước dấu @.

Ví dụ: CONG(@BIEN

Chương 11
BIẾN VÀ CÁCH SỬ DỤNG BIẾN

11.1. BIẾN NHỚ

Các biến nhớ (Memory variables) là các biến trung gian được lập ra khi thực hiện chương trình hay thực hiện các câu lệnh trong khung cửa sổ lệnh.

Ví dụ:

A = "Vân Trình"

Biến A nhận giá trị là "Vân Trình", thay cho việc bạn phải viết chuỗi "Vân Trình", bạn có thể viết A vào các biểu thức, phép so sánh. Các biến bộ nhớ được tạo ra từ các lệnh:

Store <giá trị> to <biến>

<biến> = <giá trị>

Input [<thông báo>] to <biến>

Sum ... to <biến>

Trong trường hợp bạn muốn ghi các biến nhớ lên đĩa để lần khởi động sau sẽ dùng đến (Ví dụ bạn có một biến lưu trữ số hóa đơn và bạn muốn số hóa đơn lần sau luôn bằng lần trước cộng 1), bạn hãy ghi các biến nhớ này vào đĩa theo cú pháp sau:

SAVE TO <tên tệp> [ALL LIKE <kiểu biến>/ALL
EXCEPT <kiểu biến>]

Nếu bạn chỉ thị: SAVE TO <tệp> thì tất cả các biến nhớ đang tồn tại được ghi vào tệp <tên tệp>. <tên tệp> có phần

mở rộng ngầm định là MEM.

- Nếu bạn chỉ thị:

SAVE TO <tên tệp> ALL LIKE <kiểu biến>

Chỉ có các biến nào được chỉ thị trong <kiểu biến> (bạn dùng các ký tự *, ? để chỉ thị) mới được ghi vào <tên tệp>.

- Nếu chỉ thị:

SAVE TO <tên tệp> ALL EXCEPT <kiểu biến>

Chỉ có các biến không trùng với <kiểu biến> mới được ghi vào <tên tệp>.

Ví dụ bạn đang có các biến BIEN1, BIEN2, BIEN3, AM1, AM2, CM1, CM2 tồn tại trong bộ nhớ trong.

Khi bạn ra lệnh:

SAVE TO TEPBIEN1 ALL LIKE B*.*

thì các biến có ký tự đầu là B (BIEN1, BIEN2, BIEN3) được ghi.

SAVE TO TEPBIEN2 ALL LIKE ?M*.*

thì tất cả các biến có ký tự thứ hai là M (AM1, AM2, CM1, CM2) được ghi. Trong trường hợp bạn cần dùng đến biến đã ghi trên đĩa, bạn dùng lệnh:

RESTORE FROM <tệp .MEM> [ADDITIVE]

Nếu không chỉ thị [ADDITIVE] thì khi thực hiện lệnh RESTORE sẽ hủy bỏ các biến đã tồn tại.

Ví dụ:

RESTORE FROM TEPBIEN1

11.2. BIẾN TRƯỜNG

Là các biến có tên là tên trường trong tệp CSDL để quản lý các bản ghi. Biến trường được tạo ra khi khai báo cấu trúc

tệp CSDL và vẫn tồn tại trên đĩa khi ra khỏi Foxpro hoặc mất điện và chỉ tồn tại trong bộ nhớ khi mở tệp CSDL chứa nó.

11.3. XOÁ BIẾN NHỚ

- Khi không cần đến biến nhớ, bạn hãy xóa một phần hay toàn bộ các biến nhớ để giải phóng chỗ cho các biến nhớ khác, bằng một trong các cách sau:

+ CLEAR ALL xoá tất cả các biến nhớ (đồng thời đóng tất cả các tệp CSDL ...)

+ CLEAR MEMORY xóa tất cả các biến nhớ

+ RELEASE: Lệnh này có các dạng sau:

- RELEASE <danh sách biến>: xóa các biến trong danh sách

- RELEASE ALL xóa tất cả các biến ở chế độ trực tiếp (Khi ra lệnh ở khung cửa sổ lệnh), xóa các biến riêng trong chế độ gián tiếp (trong chương trình).

+ RELEASE ALL LIKE <kiểu biến> và

+ RELEASE ALL EXCEPT <kiểu biến>. Hai từ khóa LIKE và EXCEPT được sử dụng giống như khi ghi biến vào đĩa.

11.4. SỰ KHÁC NHAU GIỮA BIẾN NHỚ VÀ BIẾN TRƯỜNG

Biến trường được lưu trữ trên đĩa khi ra khỏi Foxpro. Biến nhớ chỉ tồn tại trong khi đang sử dụng Foxpro (trừ trường hợp bạn ghi chúng vào đĩa).

Chú ý: Không được đặt tên biến nhớ trùng với biến trường

vì nếu có biến trường đang tồn tại (mở tệp CSDL chứa trường này) thì nó được ưu tiên trước và bạn không thể nào dùng được biến nhớ.

11.5. BIẾN HỆ THỐNG

Ngoài các biến nhớ của người sử dụng đưa ra, FOXPRO còn tự tạo ra và duy trì một số biến đặc biệt gọi là biến hệ thống (system memory variable). Các biến này chứa các thông tin để định dạng kết quả kết đưa ra máy in hay trên màn hình. Tên của biến hệ thống bao giờ cũng bắt đầu bằng dấu gạch dưới (_), do đó khi đặt tên biến nhớ bạn không nên đặt tên biến nhớ có ký tự gạch dưới ở đầu biến.

11.6. BIẾN MẢNG

Mảng là một dạng đặc biệt của biến nhớ, trong đó mỗi phần tử của mảng là một biến nhớ, chúng được viết cùng tên và phân biệt với nhau ở phần chỉ số. Bạn có thể dùng mảng một chiều hoặc mảng hai chiều.

a. Lệnh DIMENSION

Để có được biến mảng bạn hãy khai báo chúng trước khi sử dụng bằng lệnh:

DIMENSION <tên mảng 1> (kích thước)
[,<tên mảng 2> (kích thước)]
...

Ví dụ:

DIMENSION H(10), M(2,3)

- <tên mảng...> dài tối đa 10 kí tự, bắt đầu không phải là số và không được để trống (có dấu trống) trong tên mảng.

- <kích thước> là một số nguyên dương (nếu là mảng một chiều) hoặc hai số nguyên dương được cách nhau bởi một dấu phẩy (nếu là mảng hai chiều).

- Khi dùng lệnh DIMENSION để khai báo mảng có tên trùng với tên biến nhớ đã tồn tại thì biến này sẽ trở thành biến mảng (nếu ở Foxbase sẽ báo lỗi).

- Một mảng có tối đa 3600 phần tử và chỉ số của mảng bắt đầu từ 1. (Nếu bạn khai báo MVCOUNT = 3600 trong tệp CONFIG.FP).

- Trong một mảng, các phần tử có thể nhận các giá trị khác nhau và nếu cần các phần tử của một mảng có thể nhận các kiểu dữ liệu khác nhau.

- Một mảng hai chiều có thể được sử dụng tương tự mảng một chiều.

Ví dụ M(2,3) tương đương với M(6) và các phần tử tương ứng của nó như sau:

M(1,1) M(1,2) M(1,3) M(2,1) M(2,2) M(2,3)

tương ứng

M(1) M(2) M(3) M(4) M(5) M(6)

b. Gán giá trị cho mảng

- Gán một giá trị cho toàn mảng

Ví dụ:

```
DIMENSION M(6)
STORE 10 TO M
```

(Chú ý: chỉ ghi tên mảng và không có chỉ số)

hoặc:

```
M = 10
```

Khi này toàn mảng M (gồm 6 phần tử) đều nhận giá trị là

0. Bạn cũng có thể gán cho mảng một chuỗi ký tự, giá trị logic...

c. Gán giá trị cho từng phần tử của mảng

Ví dụ:

```
DIMENSION H(6)
STOR 3 TO H(1), H(4), H(5)
H(2) = "Gia Lượng"
H(6) = "DETECH"
```

Bạn cũng có thể dùng các lệnh ACCEPT, INPUT... để nhận giá trị từ bàn phím để gán giá trị cho từng phần tử của mảng.

Xin các bạn lưu ý: vì mảng là dạng đặc biệt của biến nhớ nên bạn cũng có thể sử dụng như một biến nhớ trong các câu lệnh.

11.7. CÁC LỆNH LÀM VIỆC VỚI BIẾN MẢNG

a. Lệnh GATHER FROM

Lệnh này dùng để chuyển dữ liệu từ một mảng vào bản ghi hiện thời của tệp CSDL đang mở.

Cú pháp

GATHER FROM <tên mảng> [FIELDS <ds trường>]

- Việc di chuyển sẽ được tiến hành từ phần tử thứ nhất của mảng vào trường thứ nhất...

- Nếu số trường trong tệp nhiều hơn số phần tử của mảng thì số trường dôi ra không được cập nhật dữ liệu. Ngược lại, nếu số trường nhỏ hơn số phần tử thì số phần tử dư ra không được chuyển các giá trị vào tệp

- Nếu bạn có chỉ thị FIELDS <ds trường> thì dữ liệu của phần tử thứ nhất của mảng được chuyển vào trường thứ nhất trong <ds trường>, phần tử thứ hai của mảng vào trường thứ hai...

- Nếu bạn chỉ thị như trên thì lệnh GATHER bỏ qua trường MEMO.

Ví dụ:

```
USE MSKH
GATHER FROM M FIELD MA, HOTEN,:
                    SOTIEN, LAI
```

b. Lệnh SCATTER

Lệnh SCATTER làm động tác ngược lại của lệnh GATHER, nghĩa là đưa các dữ liệu của bản ghi hiện thời trong tệp CSDL đang mở ra mảng.

Cú pháp:

```
SCATTER [FIELDS <ds trường>]
        TO <tên mảng>/<tên mảng> BLANK
```

- Mảng nhận dữ liệu không cần có trước, nếu chưa có Foxpro tự động tạo ra. Và kiểu dữ liệu giữa các phần tử của mảng và trường tương ứng không cần phù hợp. Foxpro sẽ lấy kiểu dữ liệu của trường để áp đặt cho kiểu dữ liệu của phần tử tương ứng của mảng. Nhưng bạn phải lưu ý: trong trường hợp <tên mảng> chưa tồn tại mà lệnh SCATTER phải tạo ra thì <tên mảng> không được trùng với tên biến nhớ.

- Nếu bạn có chỉ thị FIELDS <ds trường> thì dữ liệu của trường thứ nhất trong <ds trường> đưa vào phần tử thứ nhất của mảng trường thứ hai ... có nghĩa là bạn cũng có thể thay

đổi thứ tự dữ liệu trong các phần tử của mảng bằng cách sắp xếp lại thứ tự của trường trong <ds trường>.

- Nếu bạn sử dụng TO <tên mảng> BLANK thì một mảng được tạo ra với các phần tử có cùng kiểu dữ liệu với các trường tương ứng nhưng các phần tử này không có giá trị (rỗng).

- Nếu số trường của tệp CSDL nhiều hơn số phần tử thì số trường dôi ra sẽ không được chuyển, ngược lại nếu số phần tử nhiều hơn số trường thì số phần tử dôi ra không được nhận dữ liệu.

- Nếu bạn chỉ thị như trên thì SCATTER sẽ bỏ qua trường MEMO.

Ví dụ:

```
USE MSKH
SCATTER TO M
```

c. Lệnh APPEND FROM ARRAY

Lệnh này bổ sung vào cuối tệp đang mở một bản ghi mà dữ liệu lấy từ một mảng.

Cú pháp

APPEND FROM ARRAY <tên mảng> [FOR<bt logic>]
[FIELDS <ds trường>]

- Nếu không có chỉ thị FIELDS <ds trường> thì giá trị của phần tử thứ nhất sẽ được chuyển vào trường thứ nhất của tệp, phần tử thứ hai vào trường thứ hai... Ngược lại, nếu có chỉ thị FIELDS <ds trường> thì phần tử thứ nhất của mảng được đưa vào trường thứ nhất được liệt kê trong <ds trường>, phần tử thứ hai.

- Nếu số trường của tệp nhiều hơn số phần tử trong mảng, thì những trường dôi ra sẽ nhận những giá trị ngầm định (Zero đối với trường số, F với trường logical, ký tự trắng đối với dữ liệu ký tự...).

- Bạn có thể sử dụng tham số FOR <bt logic> để điều khiển việc ghi dữ liệu từ mảng vào tệp hay không. Chỉ khi nào giá trị của mảng thỏa mãn <bt logic> mới được ghi bổ sung vào tệp.

Ví dụ:

```
USE MSKH
APPEND FROM ARRAY H(20)
```

11.8. XEM DANH SÁCH CÁC BIẾN NHỚ

Muốn liệt kê các biến nhớ ra màn hình hoặc in ra giấy, hoặc ghi ra một tệp, có thể sử dụng một trong hai lệnh sau:

```
DISPLAY MEMORY [LIKE <dạng biến>]
        [TO PRINTER/ TO FILE <tên tệp>]
LIST MEMORY [LIKE <dạng biến>]
        [TO PRINTER/ TO FILE <tên tệp>]
```

Nếu chỉ thị TO PRINTER thì nội dung các biến sẽ được ghi ra giấy. Nếu chỉ thị TO FILE <tên tệp> thì nội dung các biến được đưa vào tệp có tên là <tên tệp>. Các thông tin về biến gồm: Tên biến, loại biến (công cộng, cục bộ hay riêng tư), nội dung của biến và cuối cùng là tổng số biến nhớ hiện có và số biến còn có thể tạo thêm cũng như số byte đã sử dụng và số byte còn trống có thể được sử dụng

11.9. ĐẶT TÊN BIẾN

Khi lập chương trình, sử dụng biến là một vấn đề rất quan trọng. Các bạn nên chú ý ngay từ khi lập các chương trình đầu tiên. Bạn phải đặt tên biến thế nào đó để các giá trị trong các chương trình con không ảnh hưởng lẫn nhau, tiết kiệm được bộ nhớ nhưng vẫn đảm bảo được nội dung chương trình.

Ví dụ 1.

Nhập tên ba học sinh, sau đó nhập điểm tổng kết của ba học sinh đó. Đưa kết quả ra màn hình tên và điểm của từng học sinh.

Nội dung chương trình

```
clear
set talk off
input 'Tên học sinh thứ nhất' to A
input 'Tên học sinh thứ hai' to B
input 'Tên học sinh thứ ba' to C
input 'Điểm của học sinh thứ nhất' to D
input 'Điểm của học sinh thứ hai' to E
input 'Điểm của học sinh thứ ba' to F
clear
?'Tên học sinh thứ nhất:',A,' Điểm:',D
?'Tên học sinh thứ hai:',B,' Điểm:',E
?'Tên học sinh thứ ba:',C,' Điểm:',F
set talk on
return
***
```

Nếu bạn dùng biến như trên rất dễ nhầm lẫn. Có thể là tên của học sinh này lại ghép với điểm của học sinh kia. Trong trường hợp này, bạn nên sử dụng như sau:

Nội dung chương trình

```
set talk off
Input 'Tên học sinh thứ nhất:' to TEN1
Input 'Tên học sinh thứ hai:' to TEN2
Input 'Tên học sinh thứ ba:' to TEN3
**********
Input 'Điểm học sinh thứ nhất:' to DIEM1
Input 'Điểm học sinh thứ hai:' to DIEM2
Input 'Điểm học sinh thứ ba:' to DIEM3
**********
clear
?'Tên học sinh thứ nhất:',TEN1,' Điểm:',DIEM1
?'Tên học sinh thứ hai:',TEN2,' Điểm:',DIEM2
?'Tên học sinh thứ ba: ',TEN3,' Điểm:',DIEM3
set talk on
return
*****
```

Rõ ràng việc sử dụng biến trong chương trình thứ hai đỡ nhầm lẫn hơn.

11.10. CÁCH SỬ DỤNG BIẾN NHỚ ĐỂ TIẾT KIỆM BỘ NHỚ

Một biến nhớ có thể:

- Chỉ cần thiết cho một đoạn chương trình nào đó, còn lại không dùng đến.
- Cần thiết cho cả một chương trình.

Vì vậy đối với các chương trình lớn, việc sử dụng biến để

tiết kiệm bộ nhớ là một điều rất cần thiết.

a. Biến chung

Các biến được tạo ra ở khung cửa sổ lệnh, các biến được một chương trình nào đó tạo ra mà trước tên của biến được khai báo bởi lệnh Public được gọi là biến chung (hay biến công cộng). Biến này tồn tại trong bất kỳ chương trình nào (nghĩa là có thể lấy ra để dùng), chúng chỉ mất đi khi ra khỏi FOXPRO, tắt máy hay có lệnh xoá chúng.

Các biến chung được khai báo bằng lệnh Public sẽ nhận dữ liệu ban đầu dạng Logic và có giá trị là .F.. Kiểu dữ liệu và giá trị của biến sẽ do các lệnh phía sau quyết định.

b. Biến cục bộ

Một biến nhớ được tạo ra trong chương trình nhưng trước nó không có lệnh Public (hoặc có lệnh Public nhưng lệnh này không cùng nằm trên một dòng với lệnh Public) được gọi là biến cục bộ. Biến cục bộ tự động mất đi khi khi kết thúc chương trình. Chỉ có chương trình cấp thấp mới dùng được biến cục bộ của chương trình cấp cao hơn.

c. Sử dụng lệnh PRIVATE để che chắn biến nhớ

Các biến nhớ cục bộ hay công cộng có thể được che chắn bằng lệnh PRIVATE, lệnh này có cú pháp như sau:

PRIVATE <ds biến nhớ>

hoặc

PRIVATE ALL [LIKE <dạng biến>/EXCEPT <dạng biến>]

Khi bạn đã sử dụng lệnh PRIVATE để khai báo một số biến bị che chắn thì việc thay đổi các giá trị của biến ở

chương trình con cấp dưới không làm thay đổi các giá trị vốn có của chương trình cấp cao hơn. Việc sử dụng các tham số LIKE <dạng biến> và EXCEPT <dạng biến> tương tự như khi ghi biến vào đĩa. Xin bạn vui lòng xem lại.

Sau đây là một ví dụ để bạn thấy rõ hơn cách sử dụng lệnh PRIVATE như thế nào. Chúng tôi đánh số lệnh của Chương trình 1 và Chương trình 2 để tiện giải thích.

- *Chương trình 1*

1. set talk off
2. store 'Giá trị ban đầu 1' to BIEN1
3. store 'Giá trị ban đầu 2' to BIEN2
4. DO CT2

* Thực hiện chương trình 2

5. ?BIEN1
6. ?BIEN2
7. Return

- *Chương trình 2*

8. ?BIEN1 9. ?BIEN2
10. PRIVATE BIEN1,BIEN2
11. store 'Đã thay đổi 1' to BIEN1
12. store 'Đã thay đổi 2' to BIEN2
13. ?BIEN1
14. ?BIEN2
15. Return

Khi thực hiện chương trình sẽ có kết quả sau:

Giá trị ban đầu 1 do lệnh 8. Giá trị ban đầu 2 do lệnh 9. Đã thay đổi 1 do lệnh 13. Đã thay đổi 2 do lệnh 14. Giá trị ban đầu 1 do lệnh 5. Giá trị ban đầu 2 do lệnh 6.

Nếu bạn không sử dụng lệnh PRIVATE ở dòng lệnh số 10 để che chắn hai biến BIEN1 và BIEN2 thì việc thay đổi giá trị của BIEN1,BIEN2 ở dòng lệnh 11,12 sẽ làm thay đổi toàn bộ giá trị của BIEN1, BIEN2 ở Chương trình 1 và kết quả sẽ có dạng như sau:

Giá trị ban đầu 1 do lệnh 8. Giá trị ban đầu 2 do lệnh 9. Đã thay đổi 1 do lệnh 13. Đã thay đổi 2 do lệnh 14. Đã thay đổi 1 do lệnh 5. Đã thay đổi 2 do lệnh 6.

Xin các bạn lưu ý:

Lệnh PRIVATE không tạo ra biến nhớ mà chỉ che chắn những biến nhớ được khai báo trong các chương trình cấp cao hơn khỏi bị chương trình hiện thời làm hỏng.

d. Sử dụng lệnh #REGION và REGIONAL

Cú pháp

#REGION <*số thứ tự*> và REGIONAL <ds các biến nhớ>

Các bạn có thể chia chương trình thành từng khu vực (region), mỗi khu vực có một số biến nhớ dành riêng cho mình được bao che khỏi ảnh hưởng của khu vực khác.

Ví dụ:

```
#REGION 1
REGIONAL A,B
store 'Vùng một' to A,B
#REGION 2
REGIONAL B,C
store 'Vùng hai' to B,C
display
**********
```

Kết qua trên màn hình:

A PRIV C "Vùng một"

B PRIV C "Vùng một" B_____2 PRIV

C "Vùng hai" B PRIV C "Vùng hai"

Xin các bạn lưu ý:

Trong khu vực hai biến B đã bị thay đổi là B_ _2 có chiều dài là 10 ký tự và như vậy giá trị ban đầu của biến B được bảo toàn. Nhìn chung việc phân chương trình thành các khu vực sẽ thuận tiện để tránh trường hợp bạn sử dụng nhầm biến đã được sử dụng với mục đích khác ở khu vực khác và việc thay đổi giá trị của nó nằm ngoài ý muốn của bạn.

e. Các ví dụ

- *Ví dụ 1:* Nhập số ngày lao động, số tiền công trung bình cong một ngày và tính số tiền được lĩnh.

Nội dung chương trình

```
set talk off
clear
public SONGAY
input 'Số ngày lao động:' to SONGAY
input 'Số tiền công TB một ngày:' to TBINH
SOTIEN=SONGAY*TBINH
?'Số tiền:',SOTIEN
set talk on
```

Thực hiện chương trình

Số ngày lao động: 20

Số tiền công TB một ngày: 20000

Kết quả trên màn hình:

Số tiền: 400000

Nếu trên khung cửa sổ lệnh, bạn ra lệnh:

Display memory

Để xem biến nhớ thì biến SONGAY còn tồn tại. Tại khung cửa sổ lệnh bạn có thể kiểm tra lại giá trị của biến SONGAY bằng lệnh:

?SONGAY

nhưng bạn không thể xem lại tiền công trung bình trong một ngày vì biến TBINH đã bị giải thể.

- *Ví dụ 2*: Tên chương trình CT1.PRG

Nội dung chương trình

```
set procedure to CT1
clear
input 'Tên của bạn:' to TEN
do CTCON1 ?TEN,TUOI
close all
Return
******
procedure CTCON1
input 'Tuổi của bạn:' to TUOI
Return
******
```

Khi thực hiện chương trình, sau khi vào các giá trị cho biến TEN và TUOI, máy sẽ thông báo: Variable not found (không tìm thấy biến). Lý do là ở chỗ, biến TUOI trong chương trình con CTCON1 là biến cục bộ nên đã bị lệnh return trong chương trình này giải phóng khỏi bộ nhớ. Như

vậy câu lệnh:

```
?TEN,TUOI
```

trong chương trình chính sẽ bị thông báo lỗi.

- *Ví dụ 3*: Tên chương trình: CT3.PRG

Nội dung chương trình

```
set talk off
set procedure CT3
clear
input 'Tên của bạn:' to TEN
DO CTCON1
close all
set talk on
return
*****
procedure CTCON1
input 'Tuổi của bạn: ' to TUOI
DO CTCON2
return
******
procedure CTCON2
?TUOI,TEN
return
*****
```

Khi thực hiện chương trình sẽ không thông báo lỗi như ví dụ 2, bởi vì: biến cục bộ chỉ bị giải thể bởi lệnh return khi trao quyền điều khiển về chương trình con cấp cao hơn. Biến cục bộ vẫn tồn tại ở chương trình con cấp dưới (biến TEN, TUOI ở chương trình con CTCON2 ở chương trình trên).

Chương 12
CÁC LỆNH NHẬP DỮ LIỆU TỪ BÀN PHÍM

12.1. LỆNH WAIT

Cú pháp

WAIT [<dòng thông báo>] TO [<biến ký tự>]
[TIMEOUT<bt số>]

Khi gặp lệnh này, trên màn hình xuất hiện <dòng thông báo> và chương trình sẽ dừng lại chờ bạn gõ một phím bất kỳ. Bạn không nhất thiết phải chỉ thị <biến ký tự>. Nếu có chỉ thị <biến ký tự>, thì ký tự bạn gõ sẽ được lưu vào <biến ký tự>. Nếu không chỉ thị <dòng thông báo> thì FOXPRO sẽ nhắc bạn:

"Press any key to continue"

nghĩa là: hãy ấn một phím bất kỳ để tiếp tục. Lệnh này rất hay được dùng trong chương trình để dừng chương trình tại một thời điểm nào đó.

Ví dụ:

WAIT 'Hãy chuẩn bị máy in. Xong ấn phím bất kỳ'

Xin các bạn lưu ý: Bạn chỉ được gõ một phím nào đó và sau đó không được gõ phím Enter. Nếu bạn gõ nhiều hơn một ký tự thì các ký tự thừa ra sẽ được lưu lại đợi đến khi chương trình có một lệnh nhập dữ liệu từ bàn phím khác

(lệnh INPUT, ACCEPT...) sẽ đưa các giá trị này vào các biến. Như vậy sẽ gây ra lỗi.

Nếu bạn có chỉ thị TIMEOUT <biểu thức số>, thì máy phải chờ bạn ấn một phím. Nếu quá thời gian này mà bạn không ấn phím nào thì <biến ký tự> sẽ nhận ký tự trống.

Chú ý: Lệnh này phải ghi ở cuối dòng, nếu không FOXPRO sẽ thông báo: "Sai cú pháp".

12.2. LỆNH ACCEPT

Cú pháp

ACCEPT [<dòng thông báo>] TO [<biến ký tự>]

<dòng thông báo> xuất hiện trên màn hình để giúp bạn hiểu được cần đưa giá trị nào vào <biến ký tự>. <biến ký tự> chỉ nhận dữ liệu kiểu ký tự, chính vì vậy bạn không cần đặt giá trị đưa vào trong dấu nháy đơn hoặc nháy kép.

Nếu <biến ký tự> chưa tồn tại thì FOXPRO sẽ tự tạo ra.

Ví dụ:

ACCEPT 'Họ và tên sinh viên:' to HOTEN

Trên màn hình sẽ xuất hiện dòng chữ:

Họ và tên sinh viên:

Bạn hãy đưa vào tên một học sinh nào đó, ví dụ:

Nguyễn Văn Trinh và ấn Enter.

? HOTEN

Kết quả:

Nguyễn Văn Trinh

Chú ý:

- Sau khi vào dữ liệu xong bạn phải ấn phím Enter (khác với lệnh WAIT).

- <dòng thông báo> luôn hiện sau dòng hiện thời. Giả sử bạn muốn <dòng thông báo> xuất hiện ở dòng 23 thì bạn phải có lệnh xử lý đến dòng 22. Ví dụ:

```
@22,0
*Xoá toàn bộ dòng 22*
ACCEPT 'Họ và tên học sinh' to HOTEN
```

Khi này dòng thông báo: 'Họ và tên sinh vien:' sẽ xuất hiện ở dòng 23.

12.3. LỆNH INPUT

Cú pháp

INPUT <dòng thông báo> TO <biến>

Lệnh này tương tự như lệnh ACCEPT nhưng có một số điểm khác là:

- <biến> có thể nhận các kiểu dữ liệu là: ký tự, số, ngày tháng, logic. Vì vậy khi đưa vào dữ liệu dạng ký tự, bạn phải đặt giữa hai dấu nháy đơn hoặc dấu nháy kép.

Ví dụ:

```
INPUT 'Họ và tên sinh viên:' to HOTEN
```

Khi đưa dữ liệu vào, bạn phải viết:

'Hoàng Gia Lượng'

12.4. LỆNH @<TỌA ĐỘ> SAY...GET...

Cú pháp

```
@<tọa độ> SAY <biểu thức> GET <biến>
        [Picture <mã> ]/[function <mã>]
        [Defaul<bthức>]
        [Range <bthức 1>][,<bthức 2>]
```

[Valid <bthức logic>/<biến số 1>]

[When <bthức logic>]

[Color <mã màu>]

READ

Lệnh này dùng để nhập dữ liệu từ bàn phím. Trong đó <biểu thức> có thể là một dòng thông báo, biến, phép tính số học, biểu thức ký tự... và được đưa ra trên màn hình tại tọa độ được chỉ thị bởi <tọa độ> trong câu lệnh. Đây là những điểm mạnh hơn so với các lệnh trên. <biến> có thể là biến trường, biến nhớ hoặc một phần tử các biến mảng. Nếu là biến trường thì tệp CSDL phải được mở trong vùng làm việc hiện thời. Nếu là biến nhớ thì <biến> phải được khai báo từ trước, nếu không bạn phải chỉ thị DEFAULT <biểu thức>.

Điểm mạnh của câu lệnh này thực chất ở các tham số đi kèm mà bạn sẽ tìm hiểu dưới đây.

a. PICTURE <mã>/FUNCTION <mã> quy định dữ liệu nhập vào phải có dạng như <mã>. Bạn hãy tìm hiểu các ví dụ sau để thấy rõ việc sử dụng <mã> tiện lợi như thế nào.

- *Ví dụ 1*

```
A=space(14)
@ 10,10 say'giá trị A=' get A picture"!!######!!"
read
```

Bạn sẽ thấy ở dòng 10, cột 10 xuất hiện dòng thông báo giá trị A= và bên cạnh có một vệt sáng có chứa con trỏ. Bạn hãy gõ ab12345cd, trong vệt sáng sẽ xuất hiện AB12345CD vì các dấu chấm than (!) sau picture đã biến chữ thường thành chữ hoa.

Nếu ở ví dụ trên bạn gõ abcdef thì khi bạn gõ phím e máy

kêu 'bíp' và không nhận chữ này vào vệt sáng vì vị trí thứ ba sau picture là dấu # qui định vị trí thứ ba trong giá trị đưa vào bắt buộc phải là số hoặc ký tự trống.

- *Ví dụ 2*

```
A=space (1A)
@ 10,10 say 'Giá trị A=' get A picture'###-###-###.##'
Read
```

Nếu bạn gõ vào số 123456789.12 thì kết quả là:

```
123-456-789.12
```

Các mã của FUNCTION

Mã	Ý nghĩa
A	Chỉ cho phép các ký tự là các chữ số
B	Dữ liệu kiểu số in ra căn lề trái
D	Dữ liệu có kiểu ngày được hiển thị theo quy định của lệnh SET DATE (chú ý: không kiểm tra dữ liệu xem có đúng ngày, tháng, năm dương lịch không)
E	Dữ liệu có kiểu ngày theo dạng của SET DATE BRITISH dù bạn có ra lệnh SET DATE là bất kỳ kiểu gì khác (không kiểm tra dữ liệu có đúng ngày, tháng, năm dương lịch không)
L	Cho in ra các các số không (0) vô nghĩa ở đầu số đưa vào (thay vì khoảng trống)
T	Bỏ bớt các ký tự trắng ở đầu vùng
Z	Nếu giá trị bằng Zero (0) thì cho in ra toàn ký tự trắng
!	Có thể đánh vào bất kỳ ký tự nào cũng được. Các chữ cái tự động biến thành chữ hoa

Các mã của PICTURE

Mã	Ý nghĩa
A	Chỉ nhận chữ cái
L	Nhận các chữ cái T, t, Y, y, F, f, N, n
X	Nhận bất kỳ ký tự nào
Y	Nhận các chữ cái Y, y, N, n. Đổi y, n thành Y, N
9	Nhận giá trị dạng số và dấu âm (-), dương (+)
#	Nhận giá trị là số, ký tự trắng dấu +, -
!	Đổi chữ thường thành chữ hoa
.	Xác định vị trí dấu chấm thập phân
,	Xác định vị trí ngăn cách hàng ngàn, triệu, tỷ
N	Chỉ cho phép đánh vào dữ liệu dạng số

Xin các bạn lưu ý: Picture "@ <mã Funtion>" và Function"<mã Function>" là như nhau. Ví dụ: Function "D" tương đương với Picture "@ D".

- *Ví dụ 3*

```
A=space(8)
set date French
@ 10,10 get A function "D"
read
```

Trên màn hình sẽ xuất hiện một vệt sáng, để bạn vào dữ liệu, theo dạng của Pháp, dd/mm/yy (ngày/tháng/năm).

- *Ví dụ 4*

```
A='Nguyen Van Trinh'
@ 10,10 say A Function "!"
```

Kết quả trên màn hình:

NGUYEN VAN TRINH

Trong câu lệnh trên bạn chỉ dùng @ <tọa độ> say FUNCTION... mà không có lệnh GET <biến>.

b. Tham số RANGE <bthức 1> [,<bthức 2>] dùng để quy định dữ liệu đưa vào phải nằm trong một giới hạn nào đó đối với kiểu ký tự, số hoặc kiểu ngày. Trong đó: <bthức 1> là giới hạn dưới và <bthức 2> là giới hạn trên. Nếu bạn chỉ có giới hạn dưới thì bạn không ghi <bthức 2>, ngược lại nếu chỉ có giới hạn trên thì bạn chỉ ghi <bthức 2> với dấu phẩy (,) ở phía trước để thông báo không có giới hạn dưới.

- *Ví dụ 1*

```
A=0
@10,10 say'giá trị A=' get A range 1,100
read
```

thì dữ liệu đưa vào phải có giá trị từ 1 đến 100. Nếu bạn đưa vào sai thì Foxpro sẽ thông báo và bạn phải ấn phím Space để vào lại.

- *Ví dụ 2*

```
A=0
@10,10 say 'Phụ cấp là: ' get A range 20000
read
```

- *Ví dụ 3*

```
A=0
@10,10 say 'Tiền thưởng' get A range ,500000
read
```

c. Tham số VALID <bt logic>/<bt số>

Chỉ thị VALID <bt logic>: nếu <bt logic> nhận giá trị .T. (đúng), thì dữ liệu được nhập vào <biến>, ngược lại phải ấn

phím Space để nhập lại.

- *Ví dụ 1*

```
KT=' '
@10,20 say 'Còn nhập nữa không C/K' get KT;
                    function "!" VALID KT$'CK'
read
```

Biến KT chỉ nhận một trong hai ký tự C,K (nếu bạn gõ c, k thì Function "!" đã chuyển thành C, K). Nếu bạn gõ các ký tự khác thì Foxpro thông báo bạn phải gõ lại.

- Chỉ thị VALID <bt số> trong đó <bt số> phải là số nguyên.

+ Nếu <bt số> bằng không (0), nghĩa là bất hợp lệ. Lúc này con trỏ nằm yên trong lệnh GET và gây ra sai sót. Nhìn chung bạn không nên dùng trường hợp này. Nếu bạn muốn sử dụng, bạn phải viết một chương trình con đặc biệt được gọi bởi VALID. Hệ thống thông báo không có hiệu lực trong trường hợp này.

+ Nếu <bt số> là dương thì trị số cho biết số vùng GET phải trả đến. Ví dụ: Bạn đang ở vùng GET thứ hai mà <bt số> là 3 nghĩa là con trỏ sẽ nhảy về vùng GET thứ năm. Nếu <bt số> vượt quá vùng GET cuối cùng thì coi như lệnh read kết thúc.

+ Nếu <bt số> là âm thì <bt số> cho biết số vùng GET phải lùi lại. Ví dụ: Bạn đang ở vùng GET thứ năm mà biểu thức số là <-2> thì con trỏ sẽ nhảy về vùng get thứ ba.

Nếu <bt số> điều khiển con trỏ vượt quá vùng GET đầu tiên thì coi như lệnh read kết thúc.

d. Tham số WHEN <bt logic>

Chỉ khi nào <bt logic> nhận giá trị .T. (đúng) thì con trỏ mới nhảy vào vùng GET để vào dữ liệu. Nếu <bt logic> nhận giá trị .F. (sai) thì con trỏ nhảy về vùng GET tiếp theo.

e. Tham số COLOR <mã màu>

Tham số này dùng để xác định chữ và nền trong vùng sáng của lệnh GET. Ví dụ: Nếu bạn muốn đặt lại màu cho lệnh:

@<tọa độ> say <biểu thức> COLOR <mã màu>

thì mã màu sẽ tương ứng với cặp màu thứ nhất trong lệnh SET COLOR TO... Nếu bạn muốn đặt màu cho lệnh:

@<tọa độ> GET <biến> COLOR <mã màu>

thì <mã màu> sẽ tương ứng với cặp màu thứ hai trong lệnh: SET COLOR TO...

- *Ví dụ 1*

```
@ 15,20 say 'Nguyen Van Trinh' :
                        picture "@!" COLOR W/B+
```

Kết quả:

NGUYEN VAN TRINH chữ trắng trên nền xanh

- *Ví dụ 2*

```
A = space (18)
@ 15,20 GET A picture "@!" COLOR ,GR+/R+
read
```

Trên màn hình sẽ xuất hiện một vệt sáng màu đỏ. Nếu bạn gõ giá trị vào sẽ có màu vàng (chữ vàng trên nền đỏ).

f. Chú ý

Bạn có thể đưa vào nhiều biến cho một lần đọc giá trị.

- *Ví dụ 1*

```
MASO=space(4)
HODEM=space(18)
TEN=space(7)
DIACHI=space(30)
@10,10 say 'Mã số: ' get MASO picture'#### '
@11,10 say 'Họ đệm: ' get HODEM picture'@!'
@12,10 say 'Tên: ' get TEN picture'@!'
@13,10 say 'Địa chỉ: ' get DIACHI
read
```

Khi này các biến: MA, HODEM, TEN, DIACHI cùng hiện lên và con trỏ ở vùng đầu tiên. Bạn có thể di chuyển con trỏ giữa các vùng trên để sửa chữa, thay đổi giá trị miễn là không vượt quá biến DIACHI.

Trong nhiều trường hợp bạn phải đưa các biến vào một cách riêng lẻ ví dụ bạn đọc giá trị biến thứ nhất và kiểm tra dữ liệu có đúng không rồi mới vào biến tiếp theo.

- *Ví dụ 2*

Giả sử có tệp CSDL là NHANSU.DBF quản lí khách hàng gồm các trường MA (mã), HODEM (họ đệm), TEN (tên) và DIACHI (địa chỉ). Bạn hãy viết chương trình nhập dữ liệu với yêu cầu sau: Sau khi nhập mã số phải kiểm tra xem mã số này đã có chưa. Nếu có rồi phải nhập lại.

Nội dung chương trình

```
M=space(4)
HD=space8)
T=space(7)
DC=space(30)
```

```
use NHANSU
Do while .T.
   @ 10,10 say 'Mã số:' get M picture'####'
   read
   locate for MA=M
   if found()
      set color to W/R+*
      @ 2,10 say 'Đã có mã này'
      set color to W/B+
      else
      @ 2,10
   exit
   endif
Enddo
@ 10,10 say 'Họ đệm:' get HODEM
@ 10,10 say 'Tên: ' get T
@ 10,10 say 'Địa chỉ:' get DC
read
append blank
replace MA WITH M, HODEM WITH HD,:
                        TEN WITH T, DIACHI WITH DC
Return
```

Trong trường hợp này, biến MA hiện lên để bạn vào dữ liệu, sau đó các biến tiếp theo mới hiện lên. Bạn không thể đưa con trỏ lên các biến đã đọc bởi lệnh Read và như vậy bắt buộc phải chấp nhận giá trị đã đưa vào.

- *Ví dụ 3*: Các biến phải được gán giá trị ban đầu. Bạn hãy tìm hiểu ví dụ dưới đây.

Nội dung chương trình

```
public HODEM,TEN,DIACHI,LUONG
store space(8) to TEN
store space(20) to HODEM
store 0 to LUONG
clear
@ 10,10 to 15,60 double
@ 11,12 say 'Họ đệm:' get HODEM
@ 12,12 say 'Tên:' get TEN
@ 13,12 say 'Địa chỉ:' get DIACHI
@ 14,12 say 'Lương:' get LUONG picture'999999'
read
clear
Return
**********
```

Khi thực hiện chương trình bạn sẽ thấy tình trạng sau: con trỏ không thể nào nhảy đến cạnh chữ "Địa chỉ:" để bạn khai báo địa chỉ. Rất nhiều bạn thắc mắc và lúng túng khi gặp trường hợp này. Lý do là ở chỗ biến DIACHI chưa được khai báo giá trị ban đầu. Sau khi khai báo biến bằng lệnh Public, biến này mới nhận giá trị .F. và vì vậy biến không chờ giá trị mà bạn đưa từ bàn phím. Điều này hay xảy ra khi bạn dùng một mảng nhiều phần tử mà các giá trị của mảng nhận nhiều giá trị khác nhau.

12.5. NHẬP DỮ LIỆU BẰNG LỆNH BROWSE

Giả sử bạn có tệp CSDL LUONG.DBF để quản lý lương trong cơ quan có cấu trúc như sau:

Field	Field Name	Type	Width	Dec	Index
1	MA	Character	4		
2	HOTEN	Character	17		
3	DONVI	Character	5		
4	CHUCVU	Character	6		
5	LUONG	Numeric	6		
6	PCTN	Numeric	6		
7	PCTX	Numeric	6		
8	YTE	Numeric	5		
9	BHXH	Numeric	5		
10	CONLAI	Numeric	6		
Total			** 67		

Ở trên, các trường có ý nghĩa như sau:

- MA: mã cán bộ
- HOTEN: họ và tên
- DONVI: đơn vị
- CHUCVU: chức vụ
- LUONG: lương cơ bản, ở đây chúng tôi dùng là số tiền được lĩnh của lương cơ bản. Các bạn có thể dùng theo hệ số (ví dụ 2.98) và khi tính tổng số tiền được lĩnh, các bạn lấy hệ số nhân với 230.000đ.
- PCTN: phụ cấp trách nhiệm
- PCTX: phụ cấp thường xuyên, các khoản phụ cấp như tiền ăn trưa, hao mòn xe...
- YTE: đóng bảo hiểm y tế
- BHXH: đóng bảo hiểm xã hội
- CONLAI: số tiền được nhận

a. Chương trình 1

Lập chương trình nhập dữ liệu (sử dụng lệnh BROWSE), dữ liệu được đưa vào tệp LUONG.DBF.

Nội dung chương trình

```
clear
set talk off
use luong
define window nhap from 4,0 to 22,78;
   shadow color w/b+,w/b+,w/gb+
activate window nhap
browse field ma,hoten,donvi,chucvu,luong,pctn,pctx,ytc,;
                    ,bhxh,conlai in window nhap
release window nhap
set talk on
Return
**************
```

b. Chương trình 2

Lập chương trình nhập dữ liệu như trên và thêm một số yêu cầu sau:

- Khi nhập một mã các bộ, phải kiểm tra xem mã này đã có chưa, nếu có phải nhập lại.
- Các tên trường được viết theo tiếng Việt, ví dụ: trường MA được thay bằng Mã.
- Các trường sau được tự động tính:
  + YTE: được tính bằng 1% của lương chính.
  + BHXH: được tính bằng 5% của lương chính.
  + Trường CONLAI được tính theo công thức:

    CONLAI=LUONG + PCTN + PCTX - YTE - BHXH

Nội dung chương trình

```
clear
close databases
set talk off
select 0 use luong
define window nhap from 4,0 to 22,78 shadow;
                        color w/b+,w/b+,w/gb+
activate window nhap
 browse field ma:h='Mã':v=kt():e='Đã có mã này',:
    hoten:h='Họ và tên',:
    donvi:h='Đơn vị',chucvu:h='C/Vụ',:
    luong:h='LgCBản':v=Tinh(),:
    yte:h='BHYtế':w=.f.,bhxh:h='BH/XH':w=.f.,:
    conlai:h='Còn lại':w=tinh2():
    in window nhap
 release window nhap
set talk on
Return
***
** Hàm tự tạo này dùng để kiểm tra xem mã mới nhập
** vào  đã có hay chưa. Khi tìm kiếm phải tìm trên toàn
** tệp nhưng trừ bản ghi đang nhập (recno()#rr)
Function kt
rr=recno()
 mm=ma
locate for ma=mm and recno()#rr
if found()
    go rr
```

```
    return.f.
    else
    go rr
return .t.
endif
***
Function tinh
    replace yte with luong*1/100.bhxh with luong*5/100
return
 ***
Function tinh2
    replace conlai with luong+pctn+pctx-luong*6/100
Return
****
```

Xin nhắc lại một số tham số của lệnh BROWSE:

<tên trường>:h=<bthức ký tự>

Khi này tên trường được thay bằng <bthức ký tự>

<tên trường>:v=<bthức lôgic>

Sau khi nhập xong giá trị cho một trường, máy sẽ kiểm tra giá trị của <bthức lôgic>. Nếu <bthức lôgic> nhận giá trị đúng (true) thì kết quả là hợp lệ. Nếu <bthức lôgic> nhận giá trị sai (false) thì kết quả là không hợp lệ và phải nhập lại dữ liệu. <bthức lôgic> có thể là một biểu thức, một hàm tự tạo..

<tên trường>:e=<bthức ký tự>

Tham số này thường được dùng đi cùng tham số <tên trường>:v=, dùng để đưa ra lời thông báo khi một giá trị được nhập vào là không hợp lệ.

<tên trường>:w=<bthức lôgic>

Trước khi nhập dữ liệu cho trường cần nhập, máy kiểm tra <bthức lôgic>. Nếu <bthức lôgic> nhận giá trị đúng (true) thì mới được phép nhập dữ liệu cho trường này, ngược lại, bạn không thể truy nhập được vào trường này. <bthức lôgic> có thể là một biểu thức, một hàm tự tạo... Xin các bạn chú ý khi phân biệt hai tham số :v= và :w=

c. Chương trình 3

Lập chương trình nhập dữ liệu như Chương trình 2 và thêm một số yêu cầu sau:

- Tên đơn vị của một cán bộ được nhập theo mã đơn vị. Phía trên màn hình nhập dữ liệu có một thông báo tên đơn vị của cán bộ đó. Nếu một mã đơn vị không có trong danh mục đơn vị thì xuất hiện một cửa sổ đưa ra danh sách các đơn vị cùng với mã tương ứng để chọn đơn vị cần thiết. Để thực hiện công việc này cần có tệp DONVI để quản lý các đơn vị có trong cơ quan gồm hai trường sau:

Field	Field Name	Type	Width	Dec	Index
1	MA	Character	5		
2	TEN	Character	17		

- Chỉ có cán bộ nào có chức vụ là GĐ (giám đốc), PGĐ (phó giám đốc), TP (trưởng phòng), PP (phó phòng) thì con trỏ mới nhảy sang mục PCTN (phụ cấp trách nhiệm), ngược lại máy tự động bỏ qua trường này.

Nội dung chương trình

```
clear
close databases
set talk off
```

```
select 0
use donvi alias donvi
select 0
use luong alias luong
define window nhap from 4,0 to 22,78 shadow color:
                                                    schem 10
define window tb from 0,10 to 2,65 shadow color
                                                    schem 5
define window tra from 4,55 to 22,76 shadow color
                                                    schem 10
activate window nhap
browse field ma:h='Ma':v=kt():e='Đã có mã này'.:
   hoten:h='Họ và tên'.donvi:h='Đơn vị':p='@!':v=tra().:
   chucvu:h='C/Vu':p='!M GD.PGD.TP.PP.NV. '.:
   luong:h='LgCBản':v=Tinh().:
   pctn:h='P/C TN':w=chucvu='GĐ' or chucvu='PGĐ':
   or chucvu='TP' or chucvu='PP'.:
   pctx:h='P/C TX',ytc:h='BHYtế':w=.f..:
   bhxh:h='BH/XH':w=.f..:
   conlai:h='Con lai':w=tinh2() in window nhap
release window nhap
release window tra
release window tb
set talk on
Return
***
Function kt
rr=recno()
```

```
mm=ma
locate for ma=mm and recno()#rr
if found()
   return.f.
   else
      return .t.
endif
***
Function tinh
replace yte with luong*1/100,bhxh with luong*5/100
Return
***
Function tinh2
replace conlai with luong+pctn+pctx-luong*6/100
Return
****
Function tra mm=donvi
activate window tb
select donvi
locate for ma=mm
if found()
   @0,10 say 'Đơn vị: '+ten
select luong
   * Xin lưu ý: nếu đã mở một cửa sổ khác mà không có
   * lệnh mở lại cửa sổ dang nhập dữ liệu bằng lệnh
   * Browse thì chương trình sẽ kết thúc
   activate window nhap
   else
```

```
        @0,0
        activate window tra
        on key label enter keyboard chr(23)
        browse in window tra fields ma:h='Mã ĐVị'.:
              ten:h='Tên đơn vị'
        deactivate window tra
        select luong
        replace donvi with donvi.ma
        activate window nhap
        on key label enter on key
    endif
Return
************
```

Xin giải thích một vài lệnh trong chương trình:

- Tham số :p='@M GD.PGD.TP.PP.NV.'

Dùng để quy định dữ liệu nhập vào chỉ là các chuỗi nằm sau @M, khi nhập dữ liệu có thể nhập các ký tự trên hoặc ấn phím SPACE BAR để chọn.

- Tham số :p='@!' dùng để biến các ký tự nhập vào thành ký tự hoa. Ngoài ra tham số này còn cho phép định dạng dữ liệu nhập vào giống như tham số picture trong lệnh nhập dữ liệu @...say...get.

d. Chương trình 4

Đây là chương trình khá đầy đủ về ví dụ cho việc nhập dữ liệu, vì vậy mong các bạn chú ý.

Các bạn hãy lập chương trình theo các yêu cầu đã có của các chương trình trên và thêm các yêu cầu sau:

- Giả sử bạn có các tệp quản lý về lương được lưu riêng cho

các tháng được ghi theo nguyên tắc sau: ví dụ LUONG1.DBF, LUONG2.DBF..., Khi nhập, sửa dữ liệu, máy sẽ hỏi 'Nhập lương tháng nào' bạn trả lời tên tháng làm việc. Nếu tệp làm việc đã tồn tại thì mở tệp đó. Nếu tệp chưa tồn tại thì mở tệp lưu dữ liệu của tháng trước để sao chép sang.

- Giả sử trong công việc được phân ra cho nhiều người, mỗi người chỉ được quyền nhập, sửa một số trường (tạm gọi là quyền truy nhập). Trong ví dụ của chương trình này trước khi vào làm việc máy sẽ hỏi mật khẩu. Tuỳ theo mật khẩu trả lời sẽ cho phép truy nhập đến các trường tương ứng. Cụ thể là:

  + Mật khẩu = giờ + phần chẵn của phút (ví dụ 15 phút tính là 10, 05 phút tính là 0) thì được làm việc với trường LUONG
  + Mật khẩu = Ngày + giờ, được làm việc với trường PCTX (phụ cấp thường xuyên)
  + Mật khẩu = Ngày + giờ + phần chẵn của phút, được làm việc với cả hai trường LUONG và PCTX (chẳng hạn như kế toán trưởng được quyền truy nhập đến tất cả các trường)

- Khi nhập, sửa đến cuối của một bản ghi nếu là bản ghi cuối cùng thì máy phải tự động tạo ra một bản ghi trắng.

Nội dung chương trình

```
clear
close databases
set talk off
define window hoi from 5,20 to 9,60 shadow
activate window hoi
```

```
tt=0
mk=0
@0,0 say time()
@0,30 say date()
@1,3 say 'Làm việc  với tháng:' get tt picture '99'
@2,3 say 'Mật khẩu:' get mk piet '99'
read
lamviec=iif(tt<=9,'LUONG'+str(tt,1),str(tt,2))
lv=lamviec+'.dbf' release window hoi
if file(lv)
   select 0
   use &lamviec alias luong
else
   truoc=iif(tt-1<=9,'LUONG'+str(tt-1,1),str(tt-1,2))
   tep=truoc+'.dbf'
   if file(tep)
         select 0
         use &truoc
         copy to &lamviec
         use &lamviec alias luong
      else
      return
   endif
endif
mk1=.f.
mk2=.f.
t=time()
d=date()
if mk=val(subs(t,1,2))+val(subs(t,4,1))*10
```

```
    mk1=.t.
endif
if mk=day(d)+val(subs(t.1.2))
    mk2=.t.
endif
if mk=day(d)+val(subs(t.1.2))+val(subs(t.4.1))*10
    mk1=.t.
    mk2=.t.
endif
select 0
use donvi alias donvi
select luong
define window nhap from 4.0 to 22.78 shadow color :
                                                    schem 10
define window tb from 0.10 to 2.65 shadow color
                                                     schem 5
define window tra from 4.55 to 22.76 shadow color
                                                    schem 10
activate window nhap
browse field ma:h='Ma':v=kt():e='Đã có mã này'.:
    hoten:h='Họ và tên'.donvi:h='Đơn vị':p='@!':v=tra().:
    chucvu:h='C/Vu':p='cM GD.PGD.TP.PP.NV. '.:
    luong:h='LgCBản':v=Tinh():w=mk1.:
    pctn:h='P/C TN':w=chucvu='GD' or chucvu='PGD':
          or chucvu='TP' or chucvu='PP'.:
    pctx:h='P/C TX':w=mk2.yte:h='BHYtế':w=.f..:
    bhxh:h='BH/XH':w=.f..:
    conlai:h='Con lai':w=tinh2():
    in window nhap
```

```
release window nhap
release window tra
release window tb
set talk on
return
***
function kt
rr=recno()
mm=ma locate for ma=mm and recno()#rr
if found()
   return.f.
   else
   return .t.
endif
***
Function tinh
   replace yte with luong*1/100,bhxh with luong*5/100
Return
***
Function tinh2
   replace conlai with luong+pctn+pctx-luong*6/100
   if recno()=reccount()
         keyboard chr(14)
   endif
Return
****
Function tra mm=donvi
   activate window tb
   select donvi locate for ma=mm
```

```
    if found()
        @0,10 say 'Đơn vị: '+ten
        select luong
        activate window nhap
    else
        @0,0
        activate window tra
        on key label enter keyboard chr(23)
        browse in window tra fields ma:h='Mã ĐVị'.:
                ten:h='Tên đơn vị'
        deactivate window tra
        select luong
        replace donvi with donvi.ma
        activate window nhap
        on key label enter on key e
endif
Return
*****************
```

Chương 13
SỬ DỤNG BẢNG CHỌN

13.1. CÁC LOẠI BẢNG CHỌN

a. Bảng chọn tự do

Đây là một loại bảng chọn gồm nhiều mục và được sắp xếp tự do, thoải mái trên màn hình.

Ví dụ:

Nội dung công việc
1. Nhập dữ liệu
2. Sửa dữ liệu
3. Tìm kiếm
4. In, báo cáo
5. Kết thúc

b. Bảng chọn thanh ngang (Bar Menus)

Đây là loại bảng chọn được trình bày theo hàng ngang trên màn hình hay trên cửa sổ.

Ví dụ:

Nhập dữ liệu Sửa dữ liệu Tìm kiếm In, báo cáo Kết thúc

c. Bảng chọn đùn lên

Đây là một loại bảng chọn khá phổ biến được trình bày theo dạng cột đứng biểu thị một số mục chọn liên quan tới nhau. Loại bảng chọn này có thể nằm riêng biệt hoặc có thể liên kết với bảng chọn thanh ngang tạo ra một loại bảng chọn

mới.

Mỗi một bảng chọn dùn lên khi tạo ra, bạn phải đặt cho nó một cái tên để sau này khi cần thiết có thể gọi từ chương trình.

d. Bảng chọn kéo xuống (Pull-Down Menus)

Bảng chọn kéo xuống được xem là hình thức kết hợp giữa bảng chọn thanh ngang và bảng chọn dùn lên được định vị. Theo loại bảng chọn này thì mỗi mục chọn trong bảng chọn thanh ngang tương ứng với một bảng chọn kéo xuống.

Ví dụ: Trên bảng chọn thanh ngang có mục chọn "Tìm kiếm". Trong bảng chọn 'Tìm kiếm' có bảng chọn kéo xuống như sau:

Tìm theo họ tên
Tìm theo đơn vị
Tìm theo chức vụ
Tìm theo lứa tuổi
Kết thúc

Sau đây các bạn sẽ tìm hiểu cụ thể từng loại bảng chọn.

13.2. BẢNG CHỌN TỰ DO

a. Tạo bảng chọn tự do

- *Cú pháp*

@<dòng,cột> PROMPT <thông báo 1>
[MESSAGE <bt ký tự 1>]
@<dòng,cột> PROMPT <thông báo 2>
[MESSAGE <bt ký tự 2>]
.....

@<dòng,cột> PROMPT <thông báo n>

[MESSAGE <bt ký tự n>]

SET MESSAGE TO [<bt số>[LEFT/CENTER/RIGHT]]

MENU TO <biến nhớ>

- Giải thích:

Mỗi lệnh PROMPT xác định:

+ Nội dung của mục chọn hiện lên trên màn hình quy định bởi <thông báo...>.

+ Tọa độ của <thông báo...> xác định bởi <dòng,cột>. Nếu có nhiều PROMPT có cùng <dòng> thì có nghĩa là tạo một bảng chọn thanh ngang với các mục chọn trải dài trên một dòng. Nếu có nhiều PROMPT có cùng <cột> thì bảng chọn được sắp xếp theo mục chọn hàng dọc.

Lệnh SET MESSAGE TO [<bt số> [LEFT/ CENTER/ RIGHT]] xác định câu ghi chú hoặc hướng dẫn sử dụng (là các <bt ký tự...> trong các lệnh PROMPT) được đưa ra ở dòng thứ <bt số>. Nếu bạn chỉ ghi SET.MESSAGE TO thì câu ghi chú (nếu có) sẽ được đưa ra ở hàng cuối cùng trên màn hình. Ngoài ra câu ghi chú có thể ở bên trái, ở giữa hoặc bên phải nếu bạn thêm các thông số LEFT, CENTER, RIGHT.

Lệnh MENU TO <biến nhớ> có ý nghĩa như sau: khi bạn đưa một vệt sáng đến một <thông báo...> nào đó và ấn Enter thì <biến nhớ> sẽ nhận giá trị là số thứ tự của <thông báo...> được liệt kê bởi lệnh @...PROMPT... <biến nhớ> không cần phải khai báo trước.

b. Bố trí các mục chọn trên bảng chọn

Có thể dùng lệnh @...PROMPT.. để bố trí các mục chọn

theo một sơ đồ nào đấy phù hợp với yêu cầu của bạn. Có nhiều cách bố trí: theo hàng, theo cột hoặc theo một thứ tự nào đó cũng được.

+ Bố trí các mục chọn theo hàng

Mục 1	Mục 2	Mục 3	Mục 4
Mục 5	Mục 6	Mục 7	Mục 8

Muốn bố trí các mục chọn trên bảng chọn theo hàng thì trong chương trình bạn phải viết các lệnh PROMPT theo thứ tự hết dòng này mới sang dòng kia.

Ví dụ:

```
@dòng1,cột1 PROMPT 'Mục 1'
@dòng1,cột2 PROMPT 'Mục 2'
.....
@dòng2,cột1 PROMPT 'Mục 5'
.....
```

Trong trường hợp bạn bố trí các mục chọn theo hàng thì khi bạn ấn mũi tên sang phải hoặc mũi tên xuống thì vệt sáng chuyển sang mục bên phải và khi đến tận cùng bên phải của hàng đó thì mới xuống mục đầu của dòng dưới. Bạn có thể ấn phím Home để về mục đầu tiên và ấn phím End để về mục cuối cùng.

+ Bố trí các mục chọn theo cột

Mục 1	Mục 5	Mục 9
Mục 2	Mục 6	
Mục 3	Mục 7	
Mục 4	Mục 8	

Muốn bố trí các mục chọn theo cột thì khi bạn ấn mũi tên qua phải hoặc mũi tên xuống dưới, vệt sáng được di chuyển

xuống mục dưới trong cùng cột và khi đến mục tận cùng bên dưới của cột đó thì vệt sáng mới di chuyển lên mục đầu của cột kế tiếp nằm bên phải. Khi bạn ấn mũi tên qua trái hoặc đi lên thì vệt sáng di chuyển ngược lại. Bạn có thể ấn phím Home để vệt sáng về mục đầu tiên và ấn phím End để vệt sáng về mục cuối cùng.

+ Bố trí các mục chọn theo một thứ tự bất kỳ.

Nói chung, khi bạn di chuyển vệt sáng @...PROMPT <thông báo> nào được viết trước thì sẽ được đến trước bất kể tọa độ của nó như thế nào. Khi này giá trị của <biến nhớ> cũng theo thứ tự của lệnh @...PROMPT trong chương trình (chứ không theo thứ tự hàng trước, hàng sau trên màn hình).

c. Chương trình ứng dụng

Các bạn hãy tìm hiểu lệnh SET MESSAGE TO...

Nội dung chương trình

```
set talk off clear
@0,8,11,40 box chr(176)
* Lệnh trên tạo ra một khung viền bên ngoài
@1,14 say 'Menu công tác'
chon=1
@3,10 prompt '1. Chương trình nhập dữ liệu';
   message 'Chương trình nhập dữ liệu mới phát sinh'
@4,10 prompt '2. Chương trình sửa dữ liệu ';
   message 'C/trình sửa dữ liệu đã nhập nhưng bị sai'
@5,10 prompt '3. Chương trình tìm kiếm';
   message 'C/ trình tìm kiếm dữ liệu theo yêu cầu'
@6,10 prompt '4. Chương trình thống kê';
```

```
        message 'Chương trình thống kê hàng tháng'
    @7,10 prompt '5. Kết thúc công việc';
        message 'Kết thúc - good bye'
    menu to chon
    set message to 14 center
    set talk on
Return
**************
```

Khi thực hiện chương trình, trên màn hình xuất hiện như sau:

```
            Menu công tác
        1. Chương trình nhập dữ liệu
        2. Chương trình sửa dữ liệu
        3. Chương trình tìm kiếm
        4. Chương trình thống kê
        5. Kết thúc công việc
        Chương trình nhập dữ liệu mới phát sinh
```

Giải thích: nếu bạn đưa vệt sáng đến mục một thì trên dòng 14 ở chính giữa sẽ xuất hiện:

Chương trình nhập dữ liệu mới phát sinh

Nếu dùng phím mũi tên di chuyển hộp sáng thì thông báo ở dòng 14 cũng thay đổi theo do chuỗi ký tự sau message tương ứng với @...PROMPT quy định.

Để chọn một mục nào đó có hai cách:

1. Di chuyển vệt sáng (dùng các phím mũi tên, Home, End) đến mục cần chọn và ấn Enter.

2. Đánh ký tự đầu của mục cần chọn (trong ví dụ trên là 1,2,3,4,5).

13.3. BẢNG CHỌN THANH NGANG

Xin các bạn lưu ý: Các lệnh dùng để tạo bảng chọn mà chúng tôi đề cập dưới đây được chia thành hai nhóm: nhóm I và nhóm II. Các lệnh nhóm I thuộc những lệnh của dBASE/FOXBASE nhưng được thực hiện trong FOXPRO. Các lệnh nhóm II là các lệnh hoàn toàn mới của FOXPRO.

a. Tạo bảng chọn thanh ngang nhóm I

Để tạo bảng chọn thanh ngang nhóm I gồm nhiều mục chọn ngang (nằm trên đầu màn hình). Mỗi mục chọn ngang có thể có hoặc không gắn liền với một bảng chọn đùn lên. Nếu có gắn liền với một bảng chọn đùn lên thì bạn sẽ có một bảng chọn kéo xuống. Loại bảng chọn này sử dụng đến mảng hai chiều.

Cú pháp

MENU BAR <mảng 1>, <bt số 1>

MENU <bt số 2>,<mảng 2>,<bt số 3> [,<bt số 4>]

SET MESSAGE TO <bt số 5>[LEFT/CENTER/RIGHT]

READ MENU BAR TO <biến nhớ 1>,<biến nhớ 2>

[SAVE]

Muốn thực hiện loại bảng chọn nhóm I, bạn phải thực hiện theo các bước sau đây:

(1) Tạo một mảng hai chiều, kiểu ký tự (dùng lệnh DIMENSION) và gán các giá trị của mảng để thành lập bảng chọn thanh ngang.

(2) Nếu bạn muốn gắn liền mỗi mục chọn ngang một bảng chọn đùn lên thì bạn tạo cho mỗi bảng chọn này một mảng một chiều và gán các giá trị cho mảng này. Nếu không thì bỏ

qua bước này.

(3) Tiếp theo bạn cài đặt bảng chọn thanh ngang với lệnh MENU BAR và mỗi bảng chọn đùn lên với lệnh MENU

(4) Cuối cùng, khởi động toàn bộ hệ thống bảng chọn với lệnh READ MENU BAR.

Các tham số trong cú pháp trên được giải thích như sau:

- Lệnh MENU BAR <mảng 1>,<bt số 1> dùng để cài đặt bảng chọn thanh ngang qua một mảng hai chiều (<mảng 1>). Trong <mảng 1>, phần tử (i,1) là mục chọn ngang chứa nội dung tiêu đề của mục sẽ hiện ra trên bảng chọn thanh ngang ở vị trí thứ i. Bảng chọn thanh ngang có bao nhiêu mục thì chiều thứ nhất trong mảng được khai báo có bấy nhiêu phần tử. Phần tử (i,2) nếu không phải là chuỗi ký tự trống, thì dùng để chứa câu ghi chú sẽ xuất hiện ở dòng do lệnh SET MESSAGE TO <bt số 5> quy định. *Xin các bạn lưu ý*: Trước khi bạn ra lệnh MENU BAR, bạn phải khai báo mảng hai chiều và gán giá trị cho các phần tử của mảng này.

- <bt số 1> trên lệnh MENU BAR cho biết số mục chọn hiện ra trên bảng chọn thanh ngang. <bt số 1> nhỏ hơn hoặc bằng 25.

Như vậy, <mảng 1> dùng trong lệnh này là một mảng hai chiều. Chiều thứ nhất là n tương ứng với n mục chọn mà bạn muốn hiện trên bảng chọn thanh ngang. Chiều thứ hai luôn là 2. Phần tử thứ nhất chứa nội dung sẽ hiện ra trên bảng chọn, phần tử thứ hai chứa nội dung ghi chú được hiện ra trên một dòng nào đó do lệnh SET MESSAGE TO <bt số 5> qui định. Khi bạn di chuyển vệt sáng đến mục chọn nào thì dòng ghi chú tương ứng sẽ hiện ra. Dòng ghi chú có thể

không có.

- Lệnh MENU <bt số 2>,<mảng 2>, <bt số 3>, [,<bt số 4>] Lệnh này chỉ sử dụng khi bạn muốn gắn liền với mỗi một mục chọn ngang của bảng chọn thanh ngang một bảng chọn kéo xuống tương ứng. Nếu bạn không có nhu cầu này thì không phải sử dụng lệnh này.

Lệnh MENU... gài một mục chọn dọc trong một mục chọn ngang của bảng chọn thanh ngang. Số thứ tự của mục chọn thanh ngang được ấn định bởi <bt số 2>.

- <mảng 2> là một mảng một chiều chứa các chuỗi văn bản dùng làm mục chọn dọc. Có bao nhiêu mục chọn dọc thì có bấy nhiêu phần tử của mảng. Nếu mục chọn bắt đầu bằng dấu gạch ngược (\) thì có nghĩa là mục ấy không thể chọn được, vệt sáng sẽ nhảy qua mục này. Nếu mục chọn bắt đầu bằng dấu \- (dấu gạch ngược và dấu trừ) thì trên bảng chọn dọc sẽ hiện lên hàng gạch ngang. Điều này giúp bạn chia các mục chọn thành nhóm.

- Tổng số mục chọn được ấn định bởi <bt số 3>

- <bt số 4>: tham số này tùy chọn. Nếu có chỉ thị này thì cho biết số mục chọn của bảng chọn dọc được hiện ra trên màn hình. Nếu <bt số 3> lớn hơn <bt số 4> thì các mục sẽ trôi khi bạn ấn mũi tên lên hay xuống. Ví dụ: nếu bảng chọn dọc có 30 mục (<bt số 3>=30) nhưng <bt số 4> chỉ ấn định là 10 thôi thì trong bảng chọn dọc chỉ xuất hiện 10 mục. Khi bạn đưa vệt sáng đến mục thứ 10 và ấn mũi tên xuống thì mục thứ 11 xuất hiện và đương nhiên mục thứ nhất bị "biến mất". Ngược lại, giả sử trên bảng chọn dọc đang xuất hiện 10 mục chọn từ mục thứ 10 đến mục thứ 19, nếu bạn đưa vệt

sáng đến mục thứ nhất trên màn hình (tương ứng với mục thứ 10 của bảng chọn) và ấn mũi tên lên thì mục thứ chín xuất hiện và mục thứ 19 bị "biến mất".

- Lệnh READ MENU BAR TO <biến nhớ 1>,<biến nhớ 2> [SAVE]

- Lệnh READ MENU BAR khởi động toàn bộ hệ thống bảng chọn được tạo ra bởi lệnh MENU BAR và MENU... (nếu có). Tùy theo bạn chọn mục chọn nào mà gán cho <biến nhớ 1> một giá trị tương ứng.

- <biến nhớ 1> ghi nhận số thứ tự của mục chọn trên bảng chọn ngang. <biến nhớ 2> ghi nhận số thứ tự của mục chọn trên bảng chọn dọc.

Trước khi bạn ra lệnh READ MENU BAR TO, bạn khai báo giá trị cho <biến nhớ 1>, <biến nhớ 2>. Ví dụ:

store to <biến nhớ 1>,<biến nhớ 2>

- Tham số SAVE mang ý nghĩa như sau: nếu bạn không dùng tham số này thì khi bạn chọn một mục nào đó trên bảng chọn dọc thì bảng chọn bị xóa đi và màn hình nguyên thủy có trước bảng chọn được khởi động sẽ hiện ra. Nếu có tham số SAVE thì bảng chọn vẫn tiếp tục hiện trên màn hình sau khi đã chọn xong một mục chọn nào đó.

b. Chương trình ứng dụng tạo bảng chọn thanh ngang nhóm I

Các bạn hãy lập lại Chương trình 1 của bảng chọn thanh ngang tự do.

Nội dung chương trình

```
set talk off
clear
```

```
store 1 to NGANG,DOC
dimension D(5,2)
D(1,1)='1. Chương trình nhập dữ liệu'
D(1,2)=' Chương trình nhập dữ liệu mới phát sinh'
D(2,1)='2. Chương trình sửa dữ liệu'
D(2,2)='Chương trình sửa dữ liệu đã nhập nhưng sai'
D(3,1)='3. Chương trình tìm kiếm'
D(3,2)='Chương trình tìm kiếm theo yêu cầu'
D(4,1)='4. Chương trình thống kê'
D(4,2)='Chương trình thống kê hàng tháng'
D(5,1)='5. Kết thúc công việc'
D(5,2)='Kết thúc -good bye'
set message to 10 center
menu bar D,5
read menu bar to NGANG,DOC
set talk on
Return
**********************
```

Giải thích: Trong chương trình này các mục chọn và các lời ghi chú được đưa vào các phần tử của mảng D. Các mục chọn được đưa ra theo hàng ngang trên màn hình. Khi bạn di chuyển vệt sáng đến mục nào thì lời giải thích sẽ hiện ra giữa dòng 10.

Tùy theo vệt sáng đang ở mục nào mà bạn ấn phím Enter, biến NGANG sẽ nhận giá trị là số thứ tự của mục chọn này trên bảng chọn thanh ngang. Trong trường hợp này, vì không có mục chọn dọc nên biến DOC không có ý nghĩa. Sở dĩ có biến này để đảm bảo tính tổng quát của cú pháp.

Xin các bạn lưu ý:

- Với bảng chọn thanh ngang, bạn không thể đánh ký tự đầu của mục chọn để vào mục chọn đó mà bắt buộc phải dùng các phím mũi tên.

- Nếu trên bảng chọn thanh ngang có quá nhiều tiêu đề không đủ để biểu thị nằm ngang trên màn hình (nhưng không quá 25 mục) thì ở tận cùng bên phải có dấu --> nhắc bạn ấn mũi tên sang phải để cho hiện thêm các mục khác.

c. Bảng chọn thanh ngang nhóm II

Muốn tạo một bảng chọn thanh ngang nhóm II, bạn phải tiến hành các bước sau đây:

(1). Sử dụng lệnh DEFINE MENU để tạo ra bảng chọn thanh ngang và gán cho nó một cái tên (menu name).

(2). Sử dụng một loạt lệnh DEFINE PAD để khai báo những mục chọn trên bảng chọn thanh ngang. Mỗi mục chọn gán cho nó một cái tên (pad name).

(3). Với mỗi mục chọn trên bảng chọn thanh ngang, bạn có thể tiến hành các công việc sau đây:

3.1 Nếu mục chọn tương ứng với một bảng chọn dọc thì sử dụng lệnh: ON PAD...ACTIVATE POPUP để cho hiện lên bảng chọn dọc đó.

3.2 Nếu mục chọn tương ứng với một bảng chọn thanh ngang cấp dưới thì sử dụng lệnh ON PAD ...ACTIVATE MENU để cho hiện lên bảng chọn thanh ngang cấp dưới.

3.3 Nếu mục chọn tương ứng với một chương trình con thì sử dụng lệnh ON SELECTION PAD để thực hiện chương trình con này.

(4). Tiếp theo, bạn sử dụng lệnh DEFINE POPUP để tạo ra bảng chọn dọc nếu bạn sử dụng mục 3.1 hoặc sử dụng lệnh DEFINE MENU nếu bạn sử dụng mục 3.2.

(5). Nếu bạn muốn khi khởi động bảng chọn thanh ngang thì một chương trình con nào đó được thực hiện, bạn hãy dùng lệnh ON SELECTION MENU. Chương trình này sẽ được thực hiện khi bạn chọn bất cứ mục chọn nào của mục chọn thanh ngang.

(6). Cuối cùng, bạn hãy sử dụng lệnh ACTIVATE MENU để khởi động toàn bộ hệ thống bảng chọn.

Bây giờ chúng ta sẽ tìm hiểu kỹ hơn từng lệnh đã nêu ở trên.

(1). *Lệnh DEFINE MENU*

Lệnh DIFINE MENU dùng để tạo ra bảng chọn thanh ngang có cú pháp như sau:

DEFINE MENU <tên bảng chọn>
[BAR [AT LINE <bt số 1>]]
[IN [WINDOW]<tên cửa sổ>/IN SCREEN]
[KEY <tên phím>] [MARK <bt ký tự 1>]
[MESSAGE <bt ký tự 2>] [NOMARGIN]
[COLOR <cặp màu>/COLOR <SCHEME(bt số)>]

Các tham số có ý nghĩa như sau:

- BAR [AT LINE <bt số 1>] cho phép bạn tạo một bảng chọn thanh ngang được hoạt động tương tự như hệ thống menu của FOXPRO (Foxpro system menu) với những đặc trưng như sau:

+ Khi một mục chọn được thực hiện thì bảng chọn thanh ngang không còn hiện dịch (không xuất hiện

trên màn hình).

+ Một hàng bảng chọn thanh ngang sẽ nằm trên đầu màn hình (hoặc cửa sổ). Nếu kích thước số mục chọn vượt quá màn hình hoặc cửa sổ thì số mục chọn vượt quá sẽ bị khuất trong màn hình. Bạn hãy dùng phím < . > để tìm chúng.

- AT LINE <bt số 1> quy định vị trí của bảng chọn thanh ngang xuất hiện ở dòng <bt số 1>.

- KEY <tên phím> quy định để khởi động bảng chọn thanh ngang thì ấn phím là <tên phím>.

- MARK<bt ký tự>: các mục chọn được đánh dấu bởi <bt ký tự>.

- IN [WINDOW] <tên cửa sổ>/IN SCREEN

Cho phép chỉ định bảng chọn thanh ngang sẽ hiện lên:

+ Màn hình chứ không phải cửa sổ đang hoạt động nếu có chỉ thị IN SCREEN.

+ Một cửa sổ nào đó nếu có chỉ thị:

IN <tên cửa sổ>

hoặc

IN WINDOW <tên cửa sổ>

- MESSAGE <bt ký tự 2> dùng để đưa <bt ký tự 2>. là một lời giải thích hoặc một thông báo. ra màn hình.

- COLOR <cặp màu>/COLOR <SCHEME <bt số> qui định màu của bảng chọn

Ví dụ: Bạn cần tạo một bảng chọn thanh ngang có tên là VATTU được hiện lên ở dòng số 1 của màn hình. Bảng chọn được hoạt động tương tự như Foxpro system menu. Để khởi động bảng chọn, ấn phím Alt+Z. Các mục chọn được đánh

dấu bởi dấu sao (*). Lệnh DEFINE được viết như sau:

```
DEFINE MENU VATTU BAR AT LINE 1 ;
                                KEY ALT+Z MARK '*'
```

(2). *Lệnh DEFINE PAD*

Lệnh DEFINE PAD được sử dụng để khai báo các mục chọn thanh ngang của bảng chọn thanh ngang có cú pháp như sau:

```
DEFINE PAD <tên mục chọn> OF
                <tên bảng chọn thanh ngang>
    PROMPT <bt ký tự 1> [AT <dòng,cột>]
    [KEY <tên phím>] [MARK <bt số 2>]
    [MESSAGE<bt ký tự 3>] [SKIP [FOR<bt logic>]]
    [COLOR <cặp màu>/COLOR SCHEME <bt số 1>]
```

- Bảng chọn thanh ngang có bao nhiêu mục thì bạn phải dùng bấy nhiêu lệnh DEFINE PAD. Mỗi mục có một tên (<tên mục chọn>) khác nhau.

- Các mục chọn sẽ được khởi động theo thứ tự được khai báo bởi lệnh DEFINE PAD.

- Đoạn văn bản xuất hiện trên màn hình của mục chọn do tham số PROMPT <bt ký tự 1> quy định. Bạn có thể đặt phím nóng nếu cần. Bạn cũng có thể đặt ở dòng và cột tùy theo ý của bạn khi sử dụng tham số AT <dòng,cột>. Nếu tất cả các <dòng> đều bằng nhau thì đây là bảng chọn thanh ngang. Nếu tất cả các <cột> đều bằng nhau thì các mục chọn được sắp xếp theo cột đứng.

Nếu bạn không chỉ thị AT <dòng,cột> thì mục chọn được khai báo đầu tiên sẽ hiện lên màn hình (hoặc cửa sổ) ở dòng 0 phía tận cùng tay trái. mục thứ hai hiện bên phải của mục

thứ nhất...

Ví dụ:

```
Define menu VATTU
Define pad Mot of VATTU prompt 'Nhập dữ liệu'
Define pad Hai of VATTU prompt 'Sửa - Hủy dữ liệu'
Define pad Ba of VATTU prompt 'Tìm kiếm '
Define pad Bon of VATTU prompt 'Thống kê - Báo cáo'
Define pad Mot of VATTU prompt 'Kết thúc'
```

(3). *Lệnh ACTIVATE*

Sau khi bạn đã khai báo bảng chọn thanh ngang bởi lệnh DEFINE MENU và khai báo các mục chọn bởi lệnh DEFINE PAD, bạn có thể sử dụng lệnh ACTIVATE MENU để khởi động bảng chọn thanh ngang đã được khai báo cho hiện lên màn hình (hoặc cửa sổ). Cú pháp như sau:

ACTIVATE MENU <tên bảng chọn thanh ngang>
[NOWAIT [PAD <tên mục chọn ngang>]]

Bạn có thể dùng phím mũi tên sang phải, sang trái để di chuyển từ mục chọn này sang mục chọn khác. Bạn cũng có thể di chuyển nhanh về mục chọn bằng cách ấn phím ký tự đầu tiên của tiêu đề mục chọn.

Ngoài ra lệnh SET CONFIRM còn ảnh hưởng đến việc xác nhận mục chọn. Lệnh này có cú pháp như sau:

SET CONFIRM ON/OFF

- Nếu SET CONFIRM OFF (đây là trị mặc nhiên) thì bạn có thể chọn một mục bằng một trong hai cách. Cách 1: di chuyển về mục cần chọn và ấn phím Enter hoặc phím Space Bar. Cách 2: ấn ký tự đầu của mục cần chọn.

- Nếu SET CONFIRM ON thì khi di chuyển về mục cần

chọn bằng cách ấn ký tự đầu tiên của tiêu đề mục chọn, mục chọn không thực hiện ngay mà bạn phải xác nhận thêm bằng cách ấn phím Enter hoặc Space Bar.

+ Tham số NOWAIT

Khi một bảng chọn được hiện lên màn hình (hoặc cửa sổ) và được khởi động, việc thực hiện chương trình sẽ ngừng lại cho tới khi một mục chọn ngang được chọn hoặc ấn phím <ESC>.

+ Tham số PAD <tên mục chọn ngang>, khi bảng chọn thanh ngang được khởi động thì mục chọn ngang được lựa chọn là mục đầu tiên. Nếu bạn muốn lựa chọn mục khác thì bạn thêm tham số PAD <tên mục chọn ngang>. Như vậy, khi khởi động mục chọn thanh ngang thì vệt sáng tự động chuyển về mục chọn mang tên <mục chọn thanh ngang>.

(4). *Lệnh ON SELECTION MENU*

Lệnh này mang cú pháp như sau:

ON SELECTION MENU

<tên bảng chọn thanh ngang>/ALL [<lệnh>]

Lệnh này cho phép bạn gán một chương trình cho bảng chọn thanh ngang. Khi bạn chọn bất cứ mục chọn ngang nào của bảng chọn thì chương trình do <lệnh> ấn định sẽ được thi hành. Chương trình có thể là một câu lệnh, một chương trình con...

Chú ý: Lệnh ON SELECTION MENU phải được đặt nằm giữa lệnh DEFINE MENU và ACTIVATE MENU.

Lệnh ON SELECTION MENU có những dạng sau:

- ON SELECTION MENU <tên mục chọn> <lệnh>: chương trình được chỉ thị bằng <lệnh> sẽ được thi hành đối

với tất cả các mục chọn được chọn.

- ON SELECTION MENU <tên mục chọn> sẽ giải phóng chương trình chỉ thị bởi <lệnh> được gán với tên mục chọn.

- ON SELECTION MENU ALL: giải phóng tất cả các chương trình gán cho mọi bảng chọn thanh ngang.

(5). *Lệnh ON SELECTION PAD*

Khi chọn một mục chọn ngang trên bảng chọn thanh ngang, có thể sử dụng lệnh SELECTION PAD để thi hành một lệnh khác hoặc một chương trình con nào đó. Lệnh ON SELECTION PAD mang cú pháp như sau:

ON SELECTION PAD <tên mục chọn>
OF <tên bảng chọn> [<lệnh>]

- <lệnh> là một lệnh hoặc một chương trình con phải thi hành khi mục chọn mang tên <tên mục chọn> được chọn.

- Nếu bạn chỉ thị:

ON SELECTION PAD <tên mục chọn>
OF <tên bảng chọn>

mà không có phần <lệnh> có nghĩa là bạn cần giải phóng chương trình con cho mục chọn này.

- Lệnh ON SELECTION PAD phải đặt nằm giữa hai lệnh DEFINE MENU và ACTIVATE MENU.

(6). *Lệnh ON PAD*

Khi chọn mục chọn trên bảng chọn thanh ngang sẽ có nhiều khả năng xảy ra như sau:

6.1. Một chương trình được thi hành

6.2. Một bảng chọn thanh ngang cấp dưới được hiện lên

6.3. Một bảng chọn dọc được khởi động và hiện lên

Đối với trường hợp 6.1 chúng ta đã tìm hiểu ở lệnh ON SELECTION PAD hoặc ON SELECTION MENU. Hai trường hợp còn lại do lệnh ON PAD tác động, cú pháp như sau:

ON PAD <tên mục chọn ngang> OF
<tên bảng chọn ngang 1>
[ACTIVATE POPUP <tên bảng chọn dọc>/
ACTIVATE MENU <tên bảng chọn ngang 2>]

- Khi cần khởi động loại và tên của bảng chọn nào thì bạn chọn lệnh tương ứng.

- Nếu chỉ thị ON PAD <tên mục chọn ngang> OF <tên bảng chọn ngang 1> mà không có các tham số đi cùng thì bảng chọn tương ứng với <tên mục chọn ngang> sẽ được giải phóng khỏi bảng chọn thanh ngang <tên bảng chọn ngang1>.

13.4. BẢNG CHỌN ĐÙN LÊN

a. Tạo bảng chọn đùn lên nhóm I

Bảng chọn đùn lên nhóm I gồm một hình chữ nhật mà cạnh trên có thể có một tiêu đề (Title), trong hình chữ nhật là các hàng chứa các mục chọn dọc của bảng chọn. Bạn có thể chọn một mục bằng cách sử dụng các phím mũi tên di chuyển hộp sáng đến mục cần chọn và ấn Enter.

Ví dụ:

C... món ăn
Phở bò
Phở gà
Thịt gà tần
Cá rán

Cháo lươn

Tái dê

Để di chuyển nhanh đến mục trên cùng, bạn an phím PgUp. Để di chuyển đến mục cuối cùng, bạn ấn phím PgDn.

Bảng chọn loại này cho phép định tọa độ cho góc trên bên trái của hình chữ nhật chứa bảng chọn. Nhưng bạn phải chú ý không được thấp quá, tọa độ xác định phải đủ chỗ cho bảng chọn xuất hiện trên màn hình.

Cú pháp của loại bảng chọn này như sau:

@<dòng,cột> MENU <mảng>,<bt số 1>,<bt số 2>

[TITLE <bt ký tự>] [SHADOW]

READ MENU TO <biến nhớ> [SAVE]

Các tham số có ý nghĩa như sau:

(1). *Lệnh @...MENU*

+ @<dòng,cột> xác định tọa độ đỉnh trên bên trái của hình chữ nhật chứa bảng chọn.

+ <mảng> là mảng một chiều chứa các mục chọn dọc của bảng chọn (ví dụ như: Nhập dữ liệu, Sửa dữ liệu...).

+ <bt số 1> xác định tổng số mục chọn dọc trong bảng chọn. Số mục chọn không được quá 128. Ví dụ: <mảng> có 20 phần tử nhưng <bt số 1> là 10 nghĩa là 10 phần tử đầu tiên được dùng làm mục chọn dọc của bảng chọn.

+ <bt số 2> chỉ ra số mục chọn xuất hiện đồng thời trên màn hình.

Nếu <bt số 1> lớn hơn <bt số 2> thì chỉ <bt số 2> mục chọn được hiện trên màn hình. Trong trường hợp này bạn có thể dùng phím mũi tên xuống để hiện tiếp các mục bên dưới và đương nhiên các mục phía trên bị khuất dần (vì vậy chúng

tôi tạm gọi là bảng chọn dùn lên)

+ TITLE <bt ký tự> dùng để đưa ra tiêu đề ở khung bảng chọn. Tham số này có thể không có.

+ SHADOW: nếu có tham số này thì một bóng đen hiện lên sau bảng chọn. Theo ngầm định thì không có bóng đen.

(2). *Lệnh READ MENU TO*

Lệnh READ MENU TO <biến nhớ> [SAVE] dùng để khởi động bảng chọn dùn lên xác định bởi lệnh @...MENU. Bạn có thể gán cho <biến nhớ> một giá trị trước nào đó. Sau khi khởi động, vệt sáng sẽ nằm ở mục chọn có số thứ tự là giá trị đã gán cho <biến nhớ>. Khi bạn chọn mục nào đó trên bảng chọn thì thứ tự của mục chọn sẽ được ghi vào <biến nhớ>. Nếu bạn ấn phím ESC khi đang ở bảng chọn thì <biến nhớ> sẽ nhận giá trị Zero (0).

Xin các bạn lưu ý:

Bảng chọn loại này cho phép số mục chọn tối đa là 128 và chiều dài mỗi mục không quá 76 ký tự (kể cả ký tự trống).

- Dữ liệu cho một mục nào đó bắt đầu bằng dấu gạch chéo ngược (\), đó là mục có hiện lên màn hình nhưng không đưa vệt sáng vào mục đó được.

- Dữ liệu cho một mục bắt đầu bằng dấu gạch chéo ngược và dấu gạch nối (\-), đó không phải là một mục chọn mà chỉ là đường gạch ngang (----------) phân chia trong bảng chọn.

- Nếu bạn chọn tọa độ theo một trong các trường hợp sau, thì FoxPro sẽ thông báo lỗi: Position is off the screen (vị trí nằm ngoài màn hình).

+ Hàng gạch dưới của bảng chọn đến biên dưới không còn đủ hai dòng trống.

+ Cột gạch dứng bên phải của bảng chọn đến biên phải của màn hình không còn đủ hai vị trí trống.

- Nếu không dùng lệnh CLEAR để xóa màn hình trước khi cho bảng chọn hiện ra, đồng thời cho tham số READ MENU có tham số SAVE thì bạn có thể lưu lại hình ảnh của nhiều bảng chọn bên cạnh nhau hoặc chồng lên nhau.

b. Tạo bảng chọn đùn lên nhóm II

Các bước cần tiến hành khi tạo lập bảng chọn đùn lên nhóm II như sau:

- Dùng lệnh DEFINE POPUP để khai báo bảng chọn đùn lên.

- Sử dụng một loạt lệnh DEFINE BAR để khai báo từng mục chọn dọc của mục chọn này. Có bao nhiêu mục chọn dọc thì có bấy nhiêu lệnh DEFINE BAR.

- Nếu một mục chọn dọc được chọn sẽ thực hiện một chương trình con (hoặc một lệnh hay một nhóm lệnh) thì bạn phải sử dụng lệnh ON SELECTION BAR liên quan đến mục chọn này. Lệnh này phải nằm giữa lệnh DEFINE POPUP và ACTIVATE.

- Sử dụng lệnh ACTIVATE POPUP để khởi động bảng chọn đùn lên.

- Trong trường hợp bảng chọn đùn lên tương ứng với một mục chọn ngang của bảng chọn thanh ngang, bạn phải sử dụng lệnh ON PAD...ACTIVATE POPUP.

Các bạn hãy lần lượt xem các lệnh này.

(1). *Lệnh DEFINE POPUP*

Cú pháp

DEFINE POPUP <tên bảng chọn>

[FROM <dòng1.cột1>] [TO <dòng2.cột2>]
[FOOTER <bt ký tự 1>] [TITLE <bt ký tự 2>]
[KEY <tên phím>] [MARGIN]
[COLOR <cặp mầu>/COLOR SCHEME <bt số>]

Lệnh này dùng để tạo một bảng chọn đùn lên mà các tham số có ý nghĩa như sau:

- Tham số FROM <dòng 1, cột 1> [TO <dòng 2, cột 2>] có ba trường hợp xảy ra:

+ Nếu ghi FROM <dòng 1, cột 1> TO <dòng 2, cột 2> thì Foxpro sẽ tạo một khung cho bảng chọn là một hình chữ nhật có đỉnh trên bên trái là <dòng 1, cột 1>, đỉnh dưới bên phải là <dòng 2, cột 2>.

+ Nếu ghi FROM <dòng 1, cột 1> (không có chỉ thị TO <dòng 2, cột 2>) thì FOXPRO sẽ tự động chỉnh kích thước của khung cho phù hợp, nghĩa là bề ngang của khung lấy chiều dài dài nhất của đoạn văn bản trong các mục chọn.

+ Nếu không ghi FROM...mà chỉ ghi TO <dòng 2, cột 2> thì Foxpro đặt góc trên bên trái có tọa độ là 0,0 và tọa độ bên phải là <dòng 2, cột 2>.

Chiều đứng của bảng chọn sẽ bị giới hạn bởi kích thước của màn hình. Nếu khung của bảng chọn không đủ chứa tất cả các mục chọn thì buộc lòng phải cho cuốn trôi lên, nghĩa là bạn phải dùng các phím mũi tên lên hoặc xuống để đọc phần bị khuất.

- Tham số TITLE <bt ký tự 2>: nếu có chỉ thị này thì <bt ký tự 2> sẽ xuất hiện trên đường viền, phía trên, chính giữa. Nếu <bt ký tự 2> dài hơn khung bảng chọn thì sẽ bị xén bớt.

- Tham số FOOTER <bt ký tự 1>: tham số này dùng để

đưa <bt ký tự 1> xuống dưới cùng của bảng chọn và ở chính giữa.

Các mục chọn dọc của bảng chọn có thể được tạo ra theo cách sau: tạo ra với một loạt lệnh DEFINE BAR. Có bao nhiêu mục chọn dọc thì có bấy nhiêu lệnh DEFINE BAR. Muốn xoá bỏ một mục chọn đã tạo ra trước đó, bạn sử dụng lệnh RELEASE BAR.

(2). *Lệnh DEFINE BAR*

Lệnh DEFINE BAR dùng để khai báo một mục chọn dọc của bảng chọn đùn lên. Có bao nhiêu mục chọn thì có bấy nhiêu lệnh DEFINE BAR. Những mục chọn cùng một bảng chọn thì phải ghi rõ tên bảng chọn. Lệnh DEFINE BAR có cú pháp như sau:

```
DEFINE BAR <bt số 1>
    OF <tên bảng chọn> PROMPT <bt ký tự 1>
       [BEFORE <bt số 2>/AFTER <bt số 2>]
        [MESSAGE <bt ký tự 4>]
       [COLOR <cặp màu>/COLOR SCHEME <bt số>]
```

(3). *Lệnh ACTIVATE POPUP*

Lệnh này nằm ở cuối nhóm lệnh khai báo một bảng chọn đùn lên dùng để khởi động bảng chọn. Lệnh này có cú pháp như sau:

```
ACTIVATE POPUP <tên mục chọn>
    [AT <dòng,cột>] [BAR <bt số>]
```

Tham số AT <dòng,cột> cho phép bạn hiển thị bảng chọn đùn lên ở tọa độ <dòng, cột> nào đó. Tham số này, nếu ghi rõ ra, sẽ ưu tiên hơn so với tham số FROM <dòng 1, cột 1>... TO <dòng2, cột2> trên lệnh DEFINE POPUP.

- Tham số BAR <bt số>: thông thường khi khởi động bảng chọn, con trỏ sẽ tự động di chuyển về mục chọn dọc thứ nhất. Nhưng nếu bạn chỉ thị BAR <bt số> thì con trỏ sẽ được định vị ở mục chọn do <bt số> quy định. Ví dụ: <bt số> là 3 thì con trỏ sẽ nhảy về mục chọn dọc số 3.

(4). *Lệnh ON SELECTION POPUP*

Cú pháp

ON SELECTION POPUP

<tên bảng chọn>/ALL [<lệnh>]

Lệnh này dùng để gán một đoạn chương trình cho bảng chọn. Khi bạn chọn bất kỳ một mục chọn dọc nào của bảng chọn thì đoạn chương trình do <lệnh> ấn định sẽ được thi hành. Đoạn chương trình này có thể là một câu lệnh hay một chương trình con. Lệnh ON SELECTION POPUP phải được đặt nằm giữa lệnh DEFINE POPUP và ACTIVATE POPUP.

Lệnh ON SELETION POPUP có những dạng sau:

- ON SELECTION POPUP <tên bảng chọn> <lệnh>: đoạn chương trình do <lệnh> ấn định chỉ được thi hành khi <tên bảng chọn> được khởi động và hiện lên..

- ON SELECTION POPUP ALL <lệnh>: đoạn chương trình sẽ được thi hành đối với tất cả mọi bảng chọn hiện thời.

- ON SELECTION POPUP <tên bảng chọn>: sẽ giải phóng đoạn chương trình (lệnh) gán cho bảng chọn mang tên <tên bảng chọn>.

- ON SELECTION POPUP ALL: sẽ giải phóng tất cả các đoạn chương trình gán cho mọi bảng chọn.

Nếu bạn muốn gán riêng cho từng mục chọn dọc trên bảng chọn đùn lên, bạn có thể sử dụng lệnh ON SELECTION BAR

được đề cập dưới đây.

(5). *Lệnh ON SELECTION BAR*

Khi chọn một mục chọn dọc trên bảng chọn đùn lên, bạn có thể sử dụng lệnh ON SELECTION BAR để ra lệnh thi hành một đoạn chương trình nào đó. Cú pháp như sau:

ON SELECTION BAR <bt số> OF <tên bảng chọn>
[<lệnh>]

Các tham số có ý nghĩa như sau:

- <lệnh> là một đoạn chương trình được thực hiện khi mục chọn mang số <bt số> được chọn. Xin bạn lưu ý: phải khai báo tên của bảng chọn.

- Nếu bạn chỉ ghi

ON SELECTION BAR <bt số> OF <tên bảng chọn>

mà không có phần lệnh nghĩa là bạn yêu cầu giải phóng chương trình con gán cho mục chọn này.

Lệnh ON SELECTION BAR phải được đặt giữa lệnh DEFINE POPUP và ACTIVATE POPUP.

(6). *Lệnh ON BAR*

Khi bạn chọn một mục chọn dọc trên bảng chọn, sẽ có nhiều tình huống xảy ra:

(a) Một bảng chọn đùn lên được khởi động và hiện lên.

(b) Một bảng chọn thanh ngang khởi động và hiện lên.

(c) Một đoạn chương trình được thi hành.

Trường hợp (c) do lệnh ON SELECTION POPUP tác động mà các bạn đã tìm hiểu ở trên. Hai trường hợp (a), (b) do lệnh ON BAR tác động. Lệnh này mang cú pháp như sau:

ON BAR <bt số> OF <bảng chọn đùn lên 1>
[ACTIVATE POPUP <bảng chọn đùn lên 2>]

ACTIVATE MENU <bảng chọn thanh ngang>]

Bạn muốn khởi động loại bảng chọn nào thì chọn loại bảng chọn và tên bảng chọn cần thiết. Bạn không được quên chỉ thị <bảng chọn đùn lên 1>.

Một mục chọn dọc nào đó có một bảng chọn được gán sẽ mang dấu hiệu là một mũi tên nằm ở phía bên phải của tên mục. Mũi tên này thông báo cho bạn biết mục chọn này có một bảng chọn để bạn chọn tiếp. Bạn nhớ chỉ thị tham số MARGIN để thêm chỗ trắng ghi dấu mũi tên này.

Nếu bạn chỉ ghi

ON BAR <bt số> OF <bảng chọn đùn lên 1>

mà thôi thì bảng chọn được gắn với mục chọn <bt số> của <bảng chọn đùn lên 1> sẽ được giải phóng.

c. Các chương trình ứng dụng bảng chọn đùn lên nhóm II

(1). *Chương trình 1*

Các bạn hãy chép nội dung chương trình sau vào trong máy để thấy rõ ý nghĩa của chương trình.

```
set talk off
clear
****
define popup NHAN_SU from 5,5 title 'Nội dung'
   define bar 1 of NHAN_SU prompt '\<Nhập-sửa dữ liệu'
   define bar 2 of NHAN_SU prompt '\<Tìm kiếm'
   define bar 3 of NHAN_SU prompt 'Thống\<Kê-báo cáo'
   define bar 4 of NHAN_SU prompt 'Kết t\<Húc'
   ********
   define popup N_SUA from 8,17 title 'Nhập -sửa'
   define bar 1 of N_SUA prompt '\<Nhập hồ sơ mới '
```

```
define bar 2 of N_SUA prompt '\<Sửa hồ sơ'
define bar 3 of N_SUA prompt '\<Hủy hồ sơ'
define bar 4 of N_SUA prompt '\<Trở về B/chọn chính'
on selection bar 4 of N_SUA
Return
*******
define popup T_KIEM from 8.17 title 'Tìm kiếm' define
bar 1 of T_KIEM prompt 'Tìm\<Một h/sơ mới'  define
bar 2 of T_KIEM prompt 'Tìm theo \<Đơn vị' define
bar 3 of T_KIEM prompt 'Tìm theo\<Chức vụ'  define
bar 4 of T_KIEM prompt '\<Trở về B/chọn chính'
on selection bar 4 of T_KIEM
Return
********
define popup T_KE from 8.17 title 'Thống kê-báo cáo'
define bar 1 of T_KE prompt 'T-kê t\<Trình độ'
define bar 2 of T_KE prompt 'T-kê \<Nghề nghiệp'
define bar 3 of T_KE prompt 'T-kê \<Lứa tuổi'
define bar 4 of T_KE prompt 'T-kê \<Mức lương'
define bar 5 of T_KE prompt 'Trở \<Về B/chọn chính'
on selection bar 5 of THONG_KE Return
*********
on selection bar 1 of NHAN_SU activate popup N_SUA
on selection bar 2 of NHAN_SU activate popup
                                                T_KIEM
on selection bar 3 of NHAN_SU activate popup T_KE
on selection bar 4 of NHAN_SU Return
*****
```

```
activate popup NHAN_SU
set talk on
return
***********
```

(2). *Chương trình 2*

Giả sử trong chương trình tìm kiếm của bạn gồm có: Tìm một hồ sơ, tìm theo đơn vị, tìm theo chức vụ. Bạn có một tệp CSDL: tệp DONVI.DBF quản lý các đơn vị trực thuộc Công ty. Hãy lập chương trình theo yêu cầu sau: Nếu bạn chọn mục chọn 'Tìm theo đơn vị' thì sẽ hiện lên danh sách các đơn vị trong Công ty để bạn chọn đơn vị cần tìm.

Nội dung chương trình

```
*******************
set talk off
clear
use DONVI
define popup TIM_KIEM from 8,17 title 'Tìm kiếm'
define bar 1 of TIM_KIEM prompt 'Tìm \<Một hồ sơ'
define bar 2 of TIM_KIEM prompt 'Tìm theo \<Đơn vị'
define bar 3 of TIM_KIEM prompt 'Tìm theo \<Chức vụ'
define bar 4 of TIM_KIEM prompt '\<Kết thúc'
*** ****
on selection bar 1 of TIM_KIEM DO TIM_1_HS
on selection bar 2 of TIM_KIEM do donvi
on selection bar 3 of TIM_KIEM DO TIM_CVU
on selection bar 4 of TIM_KIEM RETURN
*****
activate popup TIM_KIEM
```

```
set talk off
close all
return
*****
procedure TIM_1_HS
clear
@22,20 say 'Thực hiện chương trình tìm một hồ sơ'
return
*****
procedure TIM_CVU
clear
@22,20 say 'Thực hiện chương trình tìm theo chức vụ'
return
*****
procedure DONVI
bay=0
define popup DON_VI from 5,50 title 'D/các đơn vị':
        prompt fiel donvi scroll shadow
        on selection popup DON_VI do TIMDONVI
activate popup DON_VI
return
 **************
procedure TIMDONVI
    a=donvi
    clear
    @20,20 say 'Thực hiện chương trình tìm theo đơn vị'
    @21,20 say 'Các bản ghi có đơn vị là:'+a
    deactivate popup DON_VI
```

```
wait
clear
return
**********
```

Xin các bạn lưu ý:

Các bạn có thể ghép chương trình trên vào Chương trình 1.

13.5. BẢNG CHỌN KÉO XUỐNG

Có thể coi bảng chọn kéo xuống (Pull - Down Menu) như là một dạng phối hợp của bảng chọn thanh ngang và bảng chọn dùn lên. Bảng chọn kéo xuống gồm một bảng chọn thanh ngang ở đầu màn hình với dòng "ghi chú" ở một dòng nào đó trên màn hình do lệnh SET MESSAGE TO <bt số> xác định và các bảng chọn dùn lên tương ứng với mỗi mục chọn trên bảng chọn thang ngang. Mỗi khi bạn dùng phím mũi tện sang phải hoặc sang trái để di chuyển vệt sáng đến một mục nào đó trên bảng chọn thanh ngang thì bảng chọn dùn lên ở vị trí tương ứng lại hiện ra ở phía dưới.

a. Tạo bảng chọn kéo xuống

Bảng chọn kéo xuống được sử dụng như sau:

MENU BAR <mảng 1>.<bt số 1>

MENU <bt số 2>.<mảng 2>.<bt số 3>[.<bt số 4>]

SET MESSAGE TO <bt số>

READ MENU BAR TO <biến nhớ 1>.<biến nhớ 2> [SAVE]

Các bước để tạo bảng chọn kéo xuống như sau:

(1). Tạo một mảng hai chiều <mảng 1> với lệnh

DIMENSION để chứa các tiêu đề và các dòng chú thích cho bảng chọn nằm ngang. Bảng chọn có bao nhiêu mục chọn thì mảng có bấy nhiêu phần tử cho một cột. <mảng 2> là mảng một chiều, kiểu ký tự để chứa các mục cho mỗi bảng chọn dùn lên tương ứng.

(2). Khai báo và gán giá trị cho từng mảng để làm "bảng chọn kéo xuống" tương ứng với mỗi tiêu đề. Khi gán giá trị cho các phần tử của mảng, bạn để ý đến mấy điểm sau:

- Mục chọn bắt đầu bằng dấu gạch chéo ngược (\) là mục có hiện ra trên bảng chọn nhưng bạn không thể đưa vệt sáng đến để chọn được.

- Mục chọn là dấu gạch chéo ngược và dấu gạch nối (\-), đó không phải là một mục mà chỉ là đường phân chia các mục thành từng nhóm.

- Chiều dài tối đa của mỗi mục là 76.

(3). Tiếp theo, cài đặt bảng chọn thanh ngang bằng lệnh MENU BAR <tên mảng 1>.<bt số 1>.

(4). Cài đặt các bảng chọn dùn lên tương ứng với các lệnh MENU <bt số 2>.<mảng 2>.<bt số 3> [.<bt số 4>].

(5). Xác định dòng để đưa ra các câu thông báo cho bảng chọn thanh ngang bằng lệnh SET MESSAGE TO <bt số>.

(6). Dùng lệnh STORE để khai báo hai biến nhớ kiểu số: <biến nhớ 1>, <biến nhớ 2> để ghi nhận số cột và số hàng mà bạn lựa chọn.

(7). Khởi động toàn bộ bảng chọn đã được khai báo bởi lệnh READ MENU BAR

b. Chương trình ứng dụng

Nội dung chương trình

```
**********
set talk off
set color to w/b+
clear all
set escape off
set color to w/gb+,w/r+ d
dimension NGANG(4,2),NHAP(3),TIMKIEM(4),;
          TKE(4),CHAMDUT(2)
*****
NGANG(1,1)='Nhập - sửa'
NGANG(1,2)='Nhập - sửa hồ sơ cán bộ'
NGANG(2,1)='Tìm kiếm'
NGANG(2,2)='Tìm kiếm hồ sơ cán bộ, danh sách cán bộ'
NGANG(3,1)='Thống kê'
NGANG(3,2)='Thống kê theo các chỉ tiêu'
NGANG(4,1)='Kết thúc'
NGANG(4,2)='Kết thúc, về DOS hoặc K/thúc chương trình'
MENU BAR NGANG,4
menu 1,NHAP,3,3
menu 2,TIMKIEM,4,4
menu 3,TKE,4,4
menu 4,CHAMDUT,2,2
*****
NHAP(1)='Nhập hồ sơ cán bộ'
NHAP(2)='Sửa hồ sơ cán bộ'
NHAP(3)='Sao hủy hồ sơ cán bộ'
*****
```

```
TIMKIEM(1)='Tìm một cán bộ'
TIMKIEM(2)='Tìm D/sách một đơn vị'
TIMKIEM(3)='Tìm D/sách cán bộ theo một ngành'
TIMKIEM(4)='D/sách cán bộ tốt nghiệp đại học'
 *****
TKE(1)='Thống kê theo lứa tuổi'
TKE(2)='Thống kê theo trình độ'
TKE(3)='Thống kê theo ngành'
TKE(4)='Thống kê theo mức lương'
******
CHAMDUT(1)='Kết thúc chương trình'
CHAMDUT(2)='Trở về DOS'
******
MENU BAR NGANG.4
menu 1,NHAP,3,3
menu 2,TIMKIEM,4,4
menu 3,TKE,4,4
menu 4,CHAMDUT,2,2
store 1 to CHINH,DOC
set message to 22
do while .t.
   read menu bar to CHINH,doc save
   if CHINH=0 or CHINH=4
      if DOC=1
         clear
         exit
      endif
   if DOC=2
```

```
        clear
        quit
      endif
   endif
   enddo
   set talk on
   set color to w/b+
   Return
   ********************
```

Giải thích:

- Khi bạn dùng phím mũi tên sang phải. sang trái để di chuyển mục chọn trên bảng chọn nằm ngang nào đó thì một mục chọn dọc tương ứng với mục chọn ngang đó sẽ trải dọc ở phía dưới. Bạn hãy dùng phím mũi tên lên hoặc xuống để chọn vệt sáng đến mục cần chọn và ấn Enter. Một chương trình (hoặc một đoạn chương trình) tương ứng với mục chọn được thực hiện.

- Bạn có thể ấn phím PgUP. PgDn để di chuyển vệt sáng về mục chọn đầu tiên hoặc cuối cùng của bảng chọn kéo xuống.

Chương 14
IN ẤN VÀ TRẢ LỜI KẾT QUẢ

14.1. ĐƯA KẾT QUẢ RA MÀN HÌNH

Trong trạng thái ngầm định, các lệnh @<tọa độ> say.... lệnh ?...đưa kết quả ra màn hình. Trong thực tế đây chỉ là một dạng để kết xuất thông tin. Bạn có thể đưa kết quả ra máy in, đưa ra tệp văn bản. Để đặt lại chế độ đưa kết quả ra màn hình, bạn dùng lệnh SET DEVICE TO SCREEN.

14.2. ĐƯA KẾT QUẢ RA TỆP VĂN BẢN

Trường hợp thường gặp là đưa kết quả nhận được ra tệp văn bản để xem hoặc sửa đổi trước khi quyết định in chính thức hoặc loại bỏ. Việc đưa kết quả ra tệp văn bản có nhiều điểm lợi như sau:

- Nếu tọa độ của kết quả đưa ra lớn hơn tọa độ cho phépcủa màn hình thì không thể đưa kết quả ra màn hình mà bạn phải đưa vào tệp văn bản và sau đó có thể dùng bất kỳ một hệ soạn thảo văn bản nào để xem hoặc sửa.

- Nếu kết quả cần đưa ra máy in nhưng khi đang lập chương trình mà bạn đã đưa ngay ra máy in thì có thể mất nhiều thời gian vì chương trình chưa hoàn thiện.

- Trong một số trường hợp bạn muốn xem hoặc có thể phải hiệu chỉnh lại một chút trước khi quyết định in chính thức.

- Bạn có thể chuyển kết quả thu được kết nối với một kết

quả khác hoặc sang một hệ ứng dụng khác...

Để đưa kết quả ra tệp văn bản, bạn dùng lệnh sau:

SET DEVICE TO FILE <tên tệp>

Các bạn hãy tìm hiểu chương trình sau:

Giả sử bạn có tệp CSDL là K_HANG để quản lý danh sách khách hàng của Công ty. Hãy lập chương trình tìm các khách hàng ở HA NOI (Hà Nội) cùng với các tham số kèm theo. Kết quả được đưa ra tệp văn bản. Biết rằng trường để quản lý khách hàng ở tỉnh nào có tên là TINH.

Nội dung chương trình

```
set talk off
clear
set device to FILE ketqua.txt
use K_HANG
@1,40 say 'BAO CAO THONG KE'
@2,20 say 'Danh sách các khách hàng ở Hà Nội'
@3,1 say replicate('-90)
h=4
st=1
do while not eof()
   if upper(TINH)='HA NOI'
      @h,1 say str(st,3)
      @h,5 say TENKH
      @h,30 say DIACHI
      @h,61 say D_THOAI
      @h,72 say FAX
      @h,83 say EMAIL
   st=st+1
```

```
    h=h+1
  endif
  skip
  enddo
  @h.1 say replicate('-',90)
  set device to screen
  return
  ***************
```

Trong trường hợp muốn xem nội dung, có thể dùng bất kỳ hệ soạn thảo văn bản nào. Thậm chí có thể dùng ngay lệnh Modify command của Foxpro. Khi này bạn phải chỉ thị rõ: modify command ketqua.txt. Chương trình trên không thể đưa kết quả ra màn hình được vì tọa độ được chỉ định đưa thông tin ra nằm ngoài màn hình (lệnh @h,83 say ...)

Xin các bạn lưu ý: Nếu ở trên bạn quên không dùng lệnh set device to screen mà đã dùng lệnh Modify Command để xem nội dung của tệp thì chẳng thấy gì. Vì khi này tệp đang được mở. Các câu lệnh liên quan đến hỏi đáp trên màn hình ở các chương trình ứng dụng sau (lệnh <tọa độ> say ...) không được hiện ra màn hình đến khi gặp lệnh Set device to screen.

14.3. ĐƯA KẾT QUẢ RA MÁY IN

Để đưa các lệnh @<tọa độ> say ... ra máy in, trước các lệnh này, bạn hãy ra lệnh: SET DEVICE TO PRINTER. Để kết thúc việc đưa kết quả ra máy in và vào trạng thái khác, chẳng hạn đưa kết quả ra màn hình, bạn ra lệnh: set device to screen.

Đối với các lệnh đưa kết quả ra máy in bởi lệnh ?, bạn hãy

dùng lệnh SET PRINTER ON trước các lệnh này. Nếu cần kết thúc chế độ này, bạn hãy ra lệnh SET PRINTER OFF.

a. Mật độ in hàng ngang

Bạn có thể điều khiển trong chương trình hoặc dùng nút điều khiển trên máy in để chọn mật độ in hàng ngang. Mật độ này từ 5,6,7,8,9,10,12,13,17,20,4 kí tự trên một inch tùy theo từng loại máy.

b. Chiều cao của một chữ chuẩn

Chiều cao một chữ chuẩn bằng 1/10 inch = 2,54 mm. Nhưng với máy in Epson, chữ có chiều cao 3,2 mm, ngoại trừ mật độ chữ (PITCH)là 15 và chữ LQ in theo dạng mũ hay dạng hệ số thì chỉ cao 2,3 mm. Tuy nhiên bạn cũng có thể in cao gấp đôi chiều cao bình thường.

c. Khoảng cách giữa các hàng

Để xác định số hàng in trên một inch, có thể chọn.

1/6 inch = 4.23 mm cho 6 hàng trên một inch

1/8 inch = 3.17 mm cho 8 hàng trên một inch

7/72 inch = 2.4 mm cho 10.28 hàng trên một inch

Chương 15
XỬ LÝ DỮ LIỆU KIỂU NGÀY

15.1. CÁC KIỂU HIỂN THỊ DỮ LIỆU DẠNG NGÀY

Dữ liệu kiểu ngày - tháng - năm (gọi tắt là kiểu ngày) mang mã D khi tạo tệp CSDL. Độ rộng của trường kiểu ngày là 8 vị trí trong đó hai vị trí là ngày, hai vị trí là tháng, hai vị trí là năm. Hai vị trí còn lại là dấu ngăn cách (ví dụ như dấu /) giữa ngày, tháng và năm.

Kiểu ngày được ghi theo dạng sau:

(các ký hiệu dưới dây: M: tháng, D: ngày, Y: năm)

- Theo AMERICAN (ngầm định theo dạng này):

  MM/DD/YY

Ví dụ: 04/17/02

- Theo British, French và DMY: DD/MM/YY

Ví dụ: 17/04/02

- Theo Italian

  DD-MM-YY

Ví dụ: 17-04-02

- Theo ANsi: YY.MM.DD

Ví dụ: 02.04.17

- Theo Japan: YY/MM/DD

Ví dụ: 02/04/17

Khi cần chọn ngày tháng theo một dạng nào đó, ví dụ theo dạng của French (Pháp), bạn ra lệnh như sau:

Set date French

Việc gán các biến có dạng ngày, hãy đưa vào cặp dấu { }.

Ví dụ:

```
Store {06/03/02} to NGAY
```

Foxpro sẽ tạo một biến nhớ có dạng ngày để chứa ngày 06/03/02.

Xin các bạn lưu ý: Foxpro vẫn chấp nhận kiểu gán biến ngày theo dạng của FoxBase. Với ví dụ trên, bạn có thể dùng lệnh:

```
store Ctod('06/03/02') to NGAY
```

15.2. CÁC LỆNH SET LIÊN QUAN ĐẾN DỮ LIỆU KIỂU NGÀY

Ngoài lệnh SET DATE mà các bạn vừa tìm hiểu ở trên, còn có một số lệnh SET khác liên quan đến dữ liệu kiểu ngày.

a. Lệnh SET CENTURY ON/OFF

Lệnh SET CENTURY quy định biến dạng ngày hoặc hàm dạng ngày sẽ được biểu thị theo dạng nào. Nếu SET CENTURY ON thì năm được biểu thị theo bốn số. Nếu SET CENTURY OFF (ngầm định theo dạng này) thì năm được biểu thị bằng hai số.

Ví dụ:

```
SET CENTURY ON
Set Date French
?Date()
```

Kết quả là: 17/04/02

b. Lệnh SET MARK TO

Lệnh SET MARK TO dùng để ấn định ký tự phân cách

ngày, tháng, năm trên dữ liệu kiểu ngày. Lệnh này được sử dụng theo cú pháp sau:

SET MARK TO <bt ký tự>

Khi này <bt ký tự> được dùng làm dấu phân cách.

Ví dụ:

```
SET MARK TO '*' set date French
?date()
```

Kết quả là: 17*04*02

Bạn cũng có thể ra lệnh:

```
store '*' to DAU
SET MARK TO DAU
```

Nếu bạn chỉ dùng lệnh SET MARK TO thì sẽ trở lại giá trị mặc nhiên của dạng ngày.

15.3. CÁC HÀM BIẾN ĐỔI NGÀY RA KIỂU KÝ TỰ VÀ NGƯỢC LẠI

a. Hàm CTOD()

Hàm CTOD(), viết tắt của chữ Character TO Date, dùng để biến đổi dữ liệu là ký tự có dạng ngày thành dữ liệu kiểu ngày. Hàm CTOD() được dùng theo cú pháp sau:

CTOD(<bt ký tự>)

Ví dụ:

```
set date French
store CTOD('17/04/02') to NGAY
```

Khi này biến NGAY sẽ nhận giá trị có kiểu ngày.

b. Hàm DTOC()

Cú pháp

DTOC(<bt ngày>[,1])

Hàm DTOC() viết tắt chữ Date TO Character, biến đổi biểu thức ngày thành một chuỗi ký tự tương ứng. Các lệnh SET DATE và SET CENTURY sẽ điều khiển dạng biểu thị của dữ liệu.

<bt ngày> có thể là một hàm, một biến hoặc một biểu thức.

Nếu trong hàm DTOC(), bạn thêm thông số 1 thì hàm DTOC sẽ kết xuất thành chuỗi YYMMDD hoặc YYYYMMDD rất tiện dùng làm chỉ tiêu khi sắp xếp.

Hàm này rất tiện dùng khi bạn cần kết một chuỗi ký tự với một dữ liệu kiểu ngày.

Ví dụ:

```
clear
set date French
@10,10 say 'Hôm nay là ngày: '+DTOC(date())
```

c. Hàm DTOS()

Hàm DTOS, viết tắt của chữ Date TO String, dùng để biến đổi một biểu thức ngày thành một chuỗi 8 byte dạng YYYYMMDD. Hàm này tương tự như hàm DTOC() có thêm chỉ số 1, cụ thể là: DTOC(<bt ngày>,1).

15.4. CÁC HÀM CHO BIẾT NGÀY, THÁNG, NĂM, THỨ CỦA DỮ LIỆU KIỂU NGÀY

a. Hàm DOW(), CDOW()

Cú pháp

DOW(<bt ngày>)

CDOW(<bt ngày>)

Hàm CDOW() viết tắt của chữ Character Day Of Week. Hàm cho kết quả là ngày trong tuần bằng tiếng Anh (Monday, Tuesday...) của <bt ngày>.

Ví dụ:

```
set date French
store {17/04/02} to X
?CDOW(X)
```

Kết quả là: Sunday

Nếu bạn dùng hàm DOW() thì kết quả sẽ đưa ra một số của ngày trong tuần, trong đó Chủ nhật ứng với 1, thứ Hai ứng với 2....

Hai hàm trên rất tiện dùng để quản lý các công việc có liên quan đến ngày trong tuần, ví dụ như các chuyến bay vào ngày thứ Ba..

b. Hàm DAY()

Cú pháp

DAY(<bt ngày>)

Hàm cho kết quả là ngày của <bt ngày>.

Ví dụ: Giả sử bạn có tệp CSDL là DANHSACH.DBF trong đó có trường NGAYSINH dạng Date. Hãy tìm những người sinh vào ngày mồng một.

```
Clear
use DANHSACH
display all fields HOVATEN,NGAYSINH;
                                for DAY(NGAYSINH)=1
```

c. Hàm CMONTH(), MONTH()

Cú pháp

CMONTH(<bt ngày>)

MONTH(<bt ngày>)

Hàm CMONTH() cho kết quả là tháng của <bt ngày> dưới dạng tiếng Anh (January, Februay...). Nếu bạn dùng hàm MONTH() thì kết quả là dạng số.

Ví dụ: Giả sử có tệp CONGVAN.DBF quản lý các công văn của Công ty. Trong đó có trường NGAY dạng date để quản lý ngày nhận và gửi công văn. Hãy tìm các công văn từ ngày 01 đến ngày 15 của tháng 1.

```
clear
use CONGVAN
display all for DAY(NGAY)<=15:
                                    and MONTH(NGAY)=1
```

d. Hàm YEAR()

Cú pháp

YEAR(<bt ngày>)

Hàm cho kết quả là năm dưới dạng bốn chữ số của <bt ngày>.

Ví dụ:

```
A=date()
?A
2002
```

15.5. CÁC HÀM CHUYỂN ĐỔI CÁCH HIỂN THỊ DỮ LIỆU KIỂU NGÀY

a. Hàm DMY()

Cú pháp

DMY(<bt ngày>)

Hàm DMY(<bt ngày>) chuyển <bt ngày> thành dạng Châu Âu "DD Month YYYY" hoặc "DD Month YY".

b. Hàm MDY(<bt ngày>)

Hàm MDY(<bt ngày>) chuyển <bt ngày> thành dạng Mỹ: Month DD, YYYY hoặc Month DD, YY với D là ngày, Month là tháng viết bằng tiếng Anh và Y là năm.

Nếu SET CENTURY OFF thì năm chỉ có hai số. Nếu SET CENTURY ON thì năm ở dạng bốn số.

Các bạn hãy tìm hiểu các ví dụ dưới đây:

Ngày hệ thống là: 17/02/2002

```
set century off
?MDY(date())
```

Kết quả là:

```
February 17, 02
```

Nếu bạn ra lệnh:

```
set century on
?MDY(date())
```

thì kết quả là:

```
February 17, 2002
```

Nếu bạn ra lệnh:

```
set century off
?DMY(date())
```

thì kết quả là:

```
17 February 02
```

Nếu bạn ra lệnh:

```
set century on
```

?DMY(date())

thì kết quả là:

17 February 2002

15.6. CÁC HÀM SO SÁNH VÀ TÍNH TOÁN

a. Hàm BETWEEN()

Cú pháp

BETWEEN(<bthức 1>,<bthức 2>,<bthức 3>)

Hàm BETWEEN() xác định xem giá trị của <bthức 1> có nằm giữa <bthức 2> và <bthức 3> hay không. Nếu có thì hàm cho giá trị là .T.. nếu không thì hàm cho giá trị .F... Xin bạn lưu ý: <bthức 2> phải là giá trị dưới và <bthức 3> phải là giá trị trên.

Ví dụ: Giả sử bạn có một tệp CSDL tên là TIENNO.DBF (tiền nợ) để quản lý số tiền của các khách hàng đang nợ tiền Công ty của bạn. trong đó có trường NGAYTRA dạng date ghi ngày phải trả nợ. Hãy lập chương trình in ra danh sách các khách hàng đã quá hạn trả từ 1 đến 30 ngày.

Nội dung chương trình

```
set talk off
use TIENNO
scan for BETWEEN(NGAYTRA,date()-30,date())
?TENKH
?DIACHI ?SOTIEN
?NGAYTRA
endscan
```

b. Hàm EMPTY()

Cú pháp

EMPTY(<bthức>)

Hàm này được dùng cho nhiều kiểu dữ liệu, vì vậy chúng tôi cũng xin được giới thiệu đầy đủ các chức năng của hàm để bạn tham khảo.

Hàm EMPTY() kiểm tra xem <bthức> có rỗng không. Nếu rỗng thì cho giá trị .T.(True), nếu sai thì cho giá trị .F.(False).

<bthức> được coi là rỗng khi:

- Kiểu ký tự: là chuỗi rỗng, ký tự trắng
- Kiểu số có giá trị là 0 (Zéro)

Kiểu ngày: ngày rỗng (' / / ')

(tương đương với CTOD(''))

- Kiểu Logic: giá trị sai (F)
- Kiểu Memo: trường memo chưa có dữ liệu

Các bạn hay tìm hiểu ngày rỗng mà chúng tôi sẽ trình bày dưới đây:

+ Việc gán một giá trị dạng ngày không phù hợp

Ví dụ:

store ctod('30/15/03') to NGAY

+ Các biến trường dạng date chưa có dữ liệu.

Xin các bạn lưu ý: việc tính toán một phép tính số học với một ngày rỗng cũng cho một kết quả là ngày rỗng.

c. Hàm GOMONTH()

Cú pháp

GOMONTH(<bt ngày>,<bt số>)

Hàm cho giá trị là ngày, tháng, năm tính từ <bt ngày>

sau số tháng là <bt số>.

Nếu <bt số> là dương thì kết quả nhận được là ngày sau <bt ngày>. Nếu <bt số> là âm thì kết quả nhận được là ngày trước <bt ngày>.

Ví dụ:

```
set date French
store {17/04/62} to NGAY
?GOMONTH(NGAY,5)
```

Kết quả:

```
17/09/62
```

Nếu bạn ra lệnh:

```
?GOMONTH(NGAY,-4)
```

thì kết quả nhận được sẽ là:

```
17/12/61
```

15.7. CÁC CHƯƠNG TRÌNH ỨNG DỤNG

a. Chương trình 1

Giả sử (một cách đơn giản), bạn có một tệp CSDL là HSCB.DBF để quản lý hồ sơ cán bộ trong đơn vị gồm các trường HODEM,TEN, NGAYSINH, DIACHI, trong đó trường NGAYSINH dạng Date. Hãy viết chương trình nhập dữ liệu.

Nội dung chương trình

```
clear
set talk off
set date French
@9,18 to 16,60 double
use HSCB
append blank
```

```
do while .t.
   @ 10,20 say 'Họ đệm:' get HODEM
   @ 11,20 say 'Tên:' get TEN
   @ 12,20 say 'Ngày sinh:' get NGAYSINH
   @ 13,20 say 'Địa chỉ:' get DIACHI
   read
   kt=' '
   @ 15,20 say 'Có nhập tiếp không: (C/K)' get kt;
                              function"@!" valid kt$'CK'
   read
   if kt='C'
      append blank
      else
      clear
      exit
   endif
enddo
close databases
Return
*******************
```

b. Chương trình 2

Giả sử bạn có tệp VAYTIEN.DBF quản lý số tiền vay của Công ty gồm các trường: TENKH (tên khách hàng), MA (mã khách hàng), NGAY (ngày vay), SOTIEN (số tiền), LAI (lãi). Hãy lập chương trình theo yêu cầu sau:

- Khi thực hiện chương trình, nếu đã vay khách hàng nào quá ba tháng thì tính lãi và gộp vào số tiền cho vay (gốc).

Ngày vay được tính lại là ngày cho vay lần trước sau ba tháng.

- Các dữ liệu cũ phải được lưu lại.

Biết rằng lãi được tính theo ngày và không có trường hợp quá sáu tháng chưa tính lãi. Trường NGAY có dạng Date.

Nội dung chương trình

```
***********
   set date French
   set talk off
   set safety off
   close all
   clear
   dimension d(5)
   luu=space(8)
   @ 10,20 say 'Tên tệp lưu: ' get luu
   read
   use VAYTIEN
   copy to &luu
   go 1
   a=date()
   do while .not.eof()
      scatte to D
      If (year(A)-year(NGAY)=1 .and.:
                        month(A)+12-month(NGAY)>=3) .or.:
                                 (year(A)=year(ngay) .and.:
                              month(A)-month(NGAY)>3) .or.:
                                 (month(A)-month(NGAY)=3.:
                                    and.day(a)>=day(NGAY))
```

```
            tg=D(3)
            D(3)=gomonth(D(3),3)
            D(4)=D(4)+(D(3)-tg)*D(5)*D(4)
            gather from D
        endif
    skip
    enddo
    clear
    display all
    close all
    Return
**************
```

c. Chương trình 3

Giả sử bạn có tệp HANG.DBF quản lý hàng bán được gồm có các trường: TENKH (tên khách hàng), MAKH (mã khách hàng), NGAY (ngày), DONGIA (đơn giá), DVT (đơn vị tính) SOLUONG (số lượng). Hãy lập chương trình để kiểm tra số tiền bán được giữa hai ngày bất kỳ (kể cả ngày đầu và ngày cuối). Biết rằng trường NGAY có dạng Date.

Nội dung chương trình

```
set talk off
set date French
store space(8) to NGAYDAU,NGAYCUOI
clear
@ 10,20 say 'Từ ngày: ' get NGAYDAU function"@D"
@ 11,20 say 'Đến ngày: ' get NGAYCUOI function"@D"
read
```

```
n1=CTOD(NGAYDAU)
n2=CTOD(NGAYCUOI)
TONG=0
use HANG
do while .not. eof()
   if between(NGAY,n1,n2)
      TONG=TONG+SOLUONG*DONGIA
   endif
   skip
enddo
clear
@ 10,20 say "Tổng số tiền"
@ 11,20 say "Từ ngày:"+NGAYDAU+;
                                   " Đến ngày:"+NGAYCUO
@ 12,20 say "Là: "+str(TONG,10)
Return
****
```

Chương 16
PHÂN NHÁNH CHƯƠNG TRÌNH

16.1. LỆNH ON KEY

a. Cú pháp

ON KEY [<lệnh>]

Trong trường hợp bạn muốn sử dụng phím nào đó bất kỳ để thực hiện một lệnh, bạn hãy sử dụng lệnh ON KEY.

Tại một thời điểm chỉ có một lệnh nằm trong ON KEY, lệnh là hiện thời. Nếu có nhiều lệnh ON KEY thì chỉ có lệnh cuối cùng là hiện thời.

Nếu trong chương trình, vừa có lệnh ON KEY và lệnh ON ESCAPE có hiệu lực thì sẽ có các trường hợp sau đây:

- Nếu có lệnh set escape on mà bạn ấn phím ESC thì <lệnh> nằm trong câu lệnh on escape được thực hiện.

- Nếu có lệnh set escape off mà bạn ấn phím ESC thì <lệnh> nằm trong câu lệnh ON KEY được thi hành.

b . Các chương trình ứng dụng

Giả sử cần xem lướt qua hồ sơ của các cán bộ. Khi phát hiện thấy một vấn đề nào đó cần xem chi tiết, bạn hãy ấn một phím bất kỳ để chương trình dừng lại. Giả sử tệp CSDL của bạn có tên là HOSO.DBF và có các trường như trong chương trình sau.

Nội dung chương trình

```
set talk off
clear
ON KEY DO CHO
use HOSO
Do while not eof()
   @10,20 say 'Họ và tên: '+HOVATEN
   @11,20 say 'Ngày sinh: '+DTOC(NGAYSINH)
   @12,20 say 'Quê quán: '+QUEQUAN
   @13,20 say 'Địa chỉ: '+DIACHI
   @14,20 say 'Chức vụ: '+CHUCVU
   @15,20 say 'Lương: '+str(LUONG,6)
   skip
Enddo
Return
**************
Procedure CHO
   b=inkey(100)
   wait
Return
```

16.2. LỆNH ON KEY LABEL

Cú pháp

ON KEY [LABEL <tên phím>] [<lệnh>]

Lệnh ON KEY LABEL là một lệnh rất hay được sử dụng trong chương trình Foxpro để khai báo các phím được sử dụng khi muốn thực hiện lệnh. Khác với ON KEY=, trong một thời điểm, bạn có thể có nhiều lệnh ON KEY LABEL cùng có hiệu lực.

Tên phím có thể là một chữ cái, một chữ số hoặc là một tên đặc biệt gán cho phím. Dưới đây là bảng gán các tên cho các phím.

Phím (trên bàn phím)	Tên phím
←	LEFTARROW
→	RIGHTARROW
^	UPARROW
↓	DNARROW
Home	HOME
End	END
PgUp	PGUP
PgDn	PGDN
Del	DEL
Backspace	BACKSPACE
SpaceBar	SPACEBAR
Ins	INS
Tab	TAB
Shift+Tab	BACKTAB
Enter	ENTER
F1 đến F10	F1,F2,F3...
{	LBRACE
Ctrl+F1 đến Ctrl+F10	Ctrl+F1, Ctrl+F2...
shift+F1 đến shift+F9	shift+F1, shift+F2...
Alt+F1 đến Alt+F10	Alt+F1, Alt+F2...
Alt+0 đến alt+9	Alt+0, Alt+1...
Alt+A đến Alt+Z	Alt+A, Alt+B...
Ctrl+	Ctrl+LEFTARROW

Phím (trên bàn phím)	Tên phím
Ctrl+	Ctrl+RIGHTARROW
Ctrl+Home	Ctrl+HOME
Ctrl+End	Ctrl+END
Ctrl+PgUp	Ctrl+PGUP
Ctrl+PgDn	Ctrl+PGDN
Ctrl+A đến Ctrl+Z	Ctrl+A, Ctrl+B,..
Escape	ESC
}	RBRACE

Bạn cũng có thể sử dụng lệnh ON KEY LABEL trong khung cửa sổ lệnh theo chế độ trực tiếp.

16.3. CÁC CHƯƠNG TRÌNH ỨNG DỤNG

a. Chương trình 1

Hãy lập chương trình để khi ấn một phím mũi tên sẽ thông báo lên màn hình bằng tiếng Việt tên phím đó là gì.

Nội dung chương trình:

```
*****************
set talk off
clear
on key label RIGTARROW ?'Phím mũi tên sang phải'
on key label LEFTARROW ?'Phím mũi tên sang trái'
on key label UPARROW ?'Phím mũi tên lên trên'
on key label DNARROW ?'Phím mũi tên xuống dưới'
set escape on
store .T. to KIEMTRA
@0,0 say 'Hãy ấn một phím mũi tên'
```

```
    do while KIEMTRA
    enddo
    set talk on
    Clear
Return
**************
```

b. Chương trình 2

Giả sử bạn có tệp LUONG.DBF để quản lý lương của cán bộ, nhân viên trong Công ty. Hãy lập chương trình theo yêu cầu sau:

- Khi thực hiện chương trình màn hình được chia thành hai phần: phần bên trái hiện Họ tên, Đơn vị của cán bộ (mỗi người một dòng).

Khi di chuyển con trỏ (vệt sáng) đến dòng nào thì bên phải hiện các thông tin về người đó (mỗi dòng hiện một trường).

- Nếu ấn phím Enter sẽ hiện màn hình sửa dữ liệu của người đó.

- Nếu ấn phím F3 thì sẽ hiện màn hình nhập dữ liệu cho cán bộ mới.

- Nếu bạn ấn phím F8 thì sẽ đánh dấu bản ghi hiện thời. Nếu bạn ấn phím ESC thì kết thúc chương trình đồng thời khôi phục các bản ghi đã bị đánh dấu xóa (bỏ dấu xóa).

- Nếu bạn ấn phím CTRL+W sẽ kết thúc chương trình đồng thời xoá các bản ghi đã đánh dấu.

Xin bạn lưu ý: Tệp LUONG.DBF có cấu trúc như sau:

Fields name	Type	Width	Dec
HOTEN	Character	20	
DONVI	Character	10	
CHUCVU	Character	10	
LUONG	Numeric	6	
THUONG	Numeric	6	
PHUCAP	Numeric	6	
TAMUNG	Numeric	6	
GHICHU	Character	10	

Nội dung chương trình

```
*****************
 set talk off
set safety off
clear
set escape on
set device to screen
use luong
on key label Enter do nhap_sua with 'NHAP'
on key label F3 do nhap  sua with 'SUA'
on key label F8 do xoa
on key label ESC do thoat
on key label CTRL+W pack
define window NHAP from 5,5 to 15,75 double:
                        title 'Nhập=Enter. sửa=F3. Xóa=F8,'+:
                                'Kết Thúc=ESC.ghi=Ctrl+W'
activate window NHAP
browse in window nhap lpartition partition 32 redit:
                                                    noedit
on key
```

```
set safety on
set talk on
close all
release window NHAP
Return
**************
Procedure nhap_sua
parameter bien
on key
deactivate window nhap
define window nhap1 from 5.20 to 18.60 panel
activate window NHAP1
clear
if bien='NHAP'
   append blank
endif
scatter to d
Do while .t.
   @1.5 say 'Họ và tên:' get d(1)
   @2.5 say 'Đơn vị: ' get d(2)
   @3.5 say 'Chức vụ: ' get d(3)
   @4.5 say 'Lương: ' get d(4) picture'999999'
   @5.5 say 'Thưởng: ' get d(5) picture'999999'
   @6.5 say 'Phụ cấp: ' get d(6)
   @7.5 say 'Tạm ứng: ' get d(7)
   read  kt=' '
   @9.7 say 'Đã đúng chưa (D/S)' get kt function"!";
                                          valid kt$'DS'
   read
```

```
   if kt='D'
      gather from d
      clear
      exit
   endif
Enddo
on key label Enter do nhap_sua with 'NHAP'
on key label F3 do nhap_sua with 'SUA'
on key label F8 do xoa
on key label ESC do thoat
on key label CTRL+W pack
release window nhap1
activate window nhap
retry
return
************
Procedure xoa
   deactivate window nhap
   clear
   on key
   define window nhap2 from 8,20 to 12,60 panel
   activate window nhap2
   ck=' '
   @ 1,5 say 'Chắc chắn không (C/K)' get ck function"!";
                                          valid ck$'CK'
   read
   if ck='C'
      dele
   endif
```

```
on key label Enter do nhap_sua with 'NHAP'
on key label F3 do nhap_sua with 'SUA'
on key label F8 do xoa
on key label ESC do thoat
on key label CTRL+W pack
release window nhap2
activate window nhap
retry
return
*******************
Procedure thoat
   recal all
   clos all
   clear
Return
*********************
```

Xin bạn lưu ý:

Trong nhiều trường hợp, bạn phải khởi động lại máy bằng cách ấn phím F10, chọn File, chọn Quit. Phải xử lý như vậy là do lý do sau: Giả sử trong chương trình của bạn có lệnh On key label Enter DO NHAP. Nghĩa là khi ấn phím Enter thì thực hiện chương trình NHAP. Nhưng khi bạn chưa kịp vô hiệu hóa lệnh này thì chương trình đã bị lỗi và bạn phải dừng chương trình lại. Khi bạn ấn phím Enter ở bất kỳ thời điểm nào thì Foxpro đều cho gọi thực hiện chương trình NHAP. Như vậy bạn cũng không thực hiện được lệnh QUIT ở khung cửa sổ lệnh. Để tránh tình trạng này, bạn nên cài bẫy ở đầu chương trình một phím nào đó để khi ấn phím này, lệnh ON KEY (vô hiệu hóa các ON KEY LABEL) có hiệu lực

Chương 17
XỬ LÝ LỖI TRONG CHƯƠNG TRÌNH FOXPRO

17.1. CÁCH XỬ LÝ LỖI THÔNG THƯỜNG

Khi một chương trình đang thực hiện bị lỗi (do sai cú pháp...), chương trình sẽ bị dừng lại và trên màn hình xuất hiện:

Cancel Suspend Ignore

Bạn hãy chọn một trong ba giải pháp trên, chúng có ý nghĩa như sau:

a. Cancel

Hủy bỏ luôn chương trình đang thi hành dở dang, nghĩa là mọi việc mà Foxpro đang thi hành bị ngừng hoàn toàn. Trong trường hợp này, các biến nhớ bị loại bỏ (trừ các biến chung). Các tệp CSDL đang dùng trong chương trình vẫn ở tình trạng mở.

Sau khi sửa các lỗi sai trong chương trình và cho chạy lại thì chương trình phải chạy lại từ đầu dù chương trình đã chạy mất rất nhiều thời gian trước khi gặp lỗi. Có thể là một số tệp CSDL đã được cập nhật, do đó bạn phải duy trì lại trạng thái ban đầu của chương trình. Điều này rất phiền

phức khi chương trình của bạn có liên quan đến xử lý các bản ghi của nhiều tệp CSDL.

Ví dụ: Cần đưa vào một tệp trung gian (mà cấu trúc được tạo trước khi lập chương trình) danh sách các học sinh tiên tiến trong trường trước khi cho in ra máy in. Khi tính các lớp 10 và lớp 11 vẫn bình thường, nghĩa là các học sinh tiên tiến của lớp 10 và lớp 11 vẫn được đưa vào trong tệp trung gian. Nhưng nếu khi tính cho các lớp 12 có sự cố thì khi chạy lại chương trình, bạn phải xóa đi tất cả các học sinh đã có trong tệp. Nếu không, các học sinh đã được đưa vào từ trước lại được cập nhật một lần nữa và như vậy sẽ gây ra sai sót.

Một ví dụ khác. Giả sử bạn lập chương trình để tăng lương cho cán bộ lên một số phần trăm nào đấy. Các đơn vị đầu vẫn thực hiện một cách bình thường nhưng đến gần cuối có trục trặc. Nếu bạn sửa lỗi trong chương trình và chạy lại thì các cán bộ ở đơn vị đầu lại được tăng lương một lần nữa...

Nhìn chung, đối với các chương trình có liên quan đến xử lý các bản ghi của tệp CSDL, bạn phải thật cẩn thận khi ấn C để chọn Cancel.

b. Suspend. Tạm treo. Có thể lỗi sai của chương trình có thể tạm thời khắc phục được. Bạn có thể tạm treo chương trình lại để qua khung cửa sổ lệnh ra một lệnh trực tiếp để thay thế cho câu lệnh bị sai rồi ra lệnh Resume để tiếp tục thực hiện chương trình. Khi bạn ra lệnh resume, Foxpro sẽ tiếp tục thi hành lệnh tiếp theo sau lệnh bị sai.

Suspend tạm thời chuyển chế độ làm việc của Foxpro từ chế độ chương trình sang chế độ trực tiếp qua khung cửa sổ lệnh. Hiện trạng của chương trình, của các tệp, các biến nhớ

đều được giữ nguyên như ngay trước khi chương trình gặp lỗi. Ngoài việc có thể ra lệnh theo kiểu trực tiếp để thay thế câu lệnh bị sai và sau đó tiếp tục thực hiện chương trình. Suspend còn cho phép xem xét, kiểm tra nội dung của biến nhớ để tìm ra lý do lỗi và chọn hướng xử lý. Nhìn chung, bạn nên tận dụng khả năng này của Foxpro vì qua đó có thể tìm ra lỗi một cách dễ dàng và chương trình không phải thực hiện từ đầu như lệnh Cancel.

Xin các bạn lưu ý: Việc sửa sai khi chương trình đang Suspend chỉ có tính chất tạm thời. Nếu muốn sửa trên tệp chương trình thì bạn phải gọi chương trình bằng một hệ soạn thảo nào đó và sửa chỗ sai thì mới có tính lâu bền.

Tất nhiên việc chọn Suspend không phải lúc nào cũng ổn thỏa. Lý do của nó là ở chỗ: Giả sử trong chương trình bạn có vòng lặp tới hàng triệu lần và trong vòng lặp có lỗi. Chắc chắn không thể Suspend hàng triệu lần để xem chương trình sẽ đi đến đâu. Trong trường hợp này bạn nên chọn Cancel.

c. Ignore

Có thể gõ phím I để chọn Ignore tức là bỏ qua, mặc kệ lỗi sai cứ cho chương trình thực hiện câu lệnh tiếp theo. Bạn có thể chọn giải pháp này khi bạn biết rằng lỗi sai đó không ảnh hưởng đến các lệnh tiếp theo. Ví dụ như lệnh SET, lệnh đưa một dòng thông báo nào đó ra màn hình...

Các cách trên là các xử lý lỗi thông thường của Foxpro. Nếu lập các chương trình lớn, bạn nên sử dụng kỹ thuật đặt bẫy để phát hiện sai với lệnh ON ERROR mà chúng tôi sẽ trình bày dưới đây.

17.2. KỸ THUẬT ĐẶT BẪY PHÁT HIỆN SAI VỚI LỆNH ERROR

a. Cú pháp

ON ERROR [<lệnh>]

Lệnh này thường được đặt ở đầu chương trình. Khi gặp lệnh này, Foxpro không xử lý lỗi thông thường như chúng tôi đã trình bày ở trên. Trong trường hợp này Foxpro sẽ xử lý như <lệnh> được đặt sau ON ERROR, với lệnh có thể là một lệnh, lời gọi một chương trình, thủ tục...

Ví dụ: Khi có lỗi, bạn muốn FoxPro đưa ra câu thông báo: 'Chương trình bị lỗi' và đưa ra tiếng 'Bip', bạn hãy viết câu lệnh sau ở đầu chương trình:

ON ERROR ?'Chương trình bị lỗi'+chr(7)

Nếu bạn ra lệnh ON ERROR không kèm theo một lệnh gì phía sau thì sẽ vô hiệu hóa lệnh ON ERROR <lệnh> đã có. Lệnh ON ERROR có thể được viết trong chương trình hoặc ở khung cửa sổ lệnh, song lệnh này chỉ có ý nghĩa thực sự khi bạn kết hợp với các hàm mà chúng tôi sẽ trình bày dưới đây để phát hiện lỗi số bao nhiêu, tên tiếng Anh, tên chương trình đang có sự cố.

b. Các hàm liên quan đến lệnh ON ERROR

- Hàm ERROR(), kết quả là một số cho biết lỗi số bao nhiêu trong danh sách các lỗi của Foxpro.

Ví dụ: Trong chương trình của bạn có lệnh như sau:

ON ERROR ?ERROR()

A=17+'Nguyễn Văn Trinh'

Khi thực hiện chương trình sẽ cho thông báo:

107

Để biết được nội dung cụ thể của lỗi trong chương trình, bạn nên kết hợp với hàm MESSAGE().

- Hàm MEESAGE(), cho kết quả là một câu bằng tiếng Anh thông báo nội dung của lỗi mà bạn đang gặp.

Ví dụ:

```
ON ERROR ?ERROR().MESSAGE()
M=100*'Nguyễn Văn Trinh'
```

Khi thực hiện chương trình sẽ cho thông báo:

107. Operator/operand type mismatch

- Hàm LINE(), cho biết dòng lệnh mà Foxpro đang xử lý gặp lỗi.

- Hàm PROGRAM(), cho biết tên chương trình đang có lỗi.

Bảng các lỗi thường gặp.

Lỗi số	Nội dung sai (hàm MESSAGE())	Ý nghĩa
1	File does not exist	Tệp không có
4	End of file encountered	Đã kết thúc tệp rồi mà còn SKIP nữa
5	Record is out of range	Bạn đã ra lệnh đến một bản ghi có số thứ tự lớn hơn tổng số bản ghi của tệp CSDL
7	File already exists	Tạo một tệp trùng với tệp đã có
10	Syntax error	Sai cú pháp

Lỗi số	Nội dung sai (hàm MESSAGE())	Ý nghĩa
12	Variable not found	Không tìm thấy biến (biến không có)
16	Unrecognized command verb	Động từ chính của lệnh bị sai
107	Operator/operand type mismatch	Không phù hợp kiểu dữ liệu

Xin các bạn lưu ý: Đối với hàm MESSAGE(), nếu bạn thêm đối số 1, cụ thể là MESSAGE(1), thì kết quả của hàm là câu lệnh đang có lỗi.

17.3. CÁC CHƯƠNG TRÌNH ỨNG DỤNG

a. Chương trình 1

Hãy viết chương trình để khi gặp một lỗi thường gặp (theo bảng trên), Foxpro sẽ đưa một câu thông báo bằng tiếng Việt.

Nội dung chương trình

```
Save screen to MANHINH
clear
set status off
set talk off
CAU_SAI=message(1)
do case
   case error()=1
      *File does not exists
      ?'Tệp bạn dùng trong lệnh:'
      ?CAU_SAI
```

```
    ?'Không có'
case error()=5
    *Record out of range
    ?'Câu lệnh: '
    ?CAU_SAI
    ?'Đã bị sai vì bạn đã ra lệnh xử lý một ban'
    ?'ghi có số thứ tự lớn hơn tổng số bản ghi
    ?'có trong tệp CSDL'
case error()=7
    * File already exists
    ?'Tệp chỉ định trong câu lệnh: '
    ?CAU_SAI
    ?'trùng với một tệp đã có'
case error()=10
    * Syntax error
    ?'Trong câu lệnh: '
    ?CAU_SAI
    ?'bạn đã viết sai cú pháp'
case error()=16
    * Unrecognized command verb
    ?'Bạn đã viết sai động từ lệnh trong câu lệnh:'
    ?CAU_SAI
case error()=107
    * Operator/operand type mismatch
    ?'Sai kiểu dữ liệu trong câu lệnh:'
    ?CAU_SAI
    Otherwise
    ?'Xin bạn vui lòng tự dịch câu sau ra tiếng'
```

```
        ?' Việt: ' ?MESSAGE()
        ?'để hiểu lỗi của bạn trong câu lệnh: '
        ?CAU_SAI
    endcase
    CAU_SAI=CAU_SAI+space(80-len(CAU_SAI))
@ row()+2,0 say '************************'
@ row()+1,0 say 'Bạn hãy sửa tạm câu lệnh dưới dây'
@ row()+1,0 say 'để thực hiện tiếp chương trình'
@ row()+1,0 say '************************'
@ row()+1,0 get CAU_SAI
read
&CAU_SAI
wait 'Xin hãy ấn một phím bất kỳ để tiếp tục'
restore screen from MANHINH
Return
*************
```

b. Chương trình 2

Giả sử bạn đã có chương trình XULY.PRG để đưa ra câu thông báo lỗi bằng tiếng Việt mà chúng tôi vừa đưa ra ở trên (Chương trình 1). Hãy viết một chương trình có sử dụng đến nội dung của Chương trình 1.

Nội dung chương trình

```
on error do XULY
set talk off
@ 10,20 say 'Chương trình thử nghiệm'
@ 11,20 say 'Ngày: '+date()
@ 13,20 say 'Chào tạm biệt
```

return

****

Khi thực hiện chương trình máy sẽ thông báo:

Sai kiểu dữ liệu trong câu lệnh:

@ 18,20 say 'Ngày: '+date()

****

Bạn hãy sửa tạm câu lệnh dưới đây

để thực hiện tiếp chương trình

@ 18,20 say 'Ngày: '+date()

Bạn hãy sửa lại là:

@ 19,20 say 'Ngày: '+dtoc(date()

Máy sẽ thông báo:

Xin hãy ấn một phím bất kỳ để tiếp tục.

Bạn hãy ấn một phím bất kỳ. Máy tiếp tục thông báo:

Trong câu lệnh:

@ 13,20 say 'Chào tạm biệt

bạn đã viết sai cú pháp

****

Bạn hãy sửa tạm câu lệnh dưới đây

để thực hiện tiếp chương trình

@ 13,20 say 'Chào tạm biệt

Bạn hãy sửa lại là:

@ 13,20 say 'Chào tạm biệt'

(trong câu lệnh bị sai bạn đã thiếu dấu nháy (') ở phía cuối)

Máy tiếp tục thông báo:

Xin hãy ấn một phím bất kỳ để tiếp tục.

Bạn hãy ấn một phím bất kỳ để tiếp tục chương trình. Việc sử dụng lệnh ON ERROR <lệnh> rất có ích khi lập

chương trình, nhất là khi bạn đang lập chương trình theo kiểu 'phác thảo' ban đầu nghĩa là còn nhiều trục trặc. Vì vậy xin bạn vui lòng đọc lại tóm tắt lệnh này. Lệnh ON ERROR có ba cách dùng:

1. ON ERROR <lệnh>
2. ON ERROR <DO chương trình con/ thủ tục con>
3. ON ERROR

Cách dùng thứ nhất là đơn giản, chỉ có tính chất sơ bộ nội dung lỗi, hoặc một thông báo nhắc nhở gì đó khi gặp lỗi.

Cách dùng thứ hai rất có ích khi lập chương trình. Nếu bạn có điều kiện, xin vui lòng đọc Phụ lục: 'Các thông báo lỗi của Foxpro 2.6' để lập một chương trình con (hoặc thủ tục) để xử lý lỗi đầy đủ hơn, có dạng như Chương trình 1 mà chúng tôi vừa trình bày ở trên. Bạn hãy tận dụng các hàm ERROR(), MESSAGE(), LINE(), PROGRAM() để lập chương trình.

Cách dùng thứ ba để vô hiệu hóa lệnh ON ERROR <lệnh> đã có.

Chương 18
SỬ DỤNG SỰ THAY THẾ MACRO (&)

18.1. CÁCH SỬ DỤNG

Dấu & được gọi là sự thay thế Macro (Macro substution). Lệnh & chỉ được dùng với biến dạng ký tự.

a. Cú pháp

Lệnh Macro được dùng với cú pháp sau:

& <bt ký tự>

Trong đó <bt ký tự> là một biến nhớ đã tồn tại. Để hiểu ý nghĩa của lệnh &, các bạn hãy xem các ví dụ dưới đây.

b. Các ví dụ:

- *Ví dụ 1*

```
a=10
b='a'
```

Nếu ra lệnh:

```
?b
```

Kết quả sẽ là:

```
a
```

Nhưng nếu bạn ra lệnh:

```
?&b
```

thì kết quả sẽ là:

```
10
```

Như vậy lệnh & (Macro) sẽ cho giá trị của biến mà tên biến này do biến nằm sau lệnh & lưu trữ.

- *Ví dụ 2*

M='Cộng hòa'

N='M'

Nếu bạn ra lệnh:

?N + 'Xã hội'

thì kết quả là:

M Xã hội

Nhưng nếu bạn ra lệnh:

&N+ 'Xã hội'

thì kết quả là:

Cộng hòa Xã hội

Điều này có nghĩa là lệnh & có thể tham gia vào các phép toán, các biểu thức so sánh...như là việc sử dụng một biến thông thường.

- *Ví dụ 3*

MAU='Set color to W/B+*'

?MAU

Kết quả trên màn hình:

Set color to W/b+*

Nhưng nếu ra lệnh:

&MAU

thì sẽ đặt chế độ chữ trắng trên nền xanh và nhấp nháy.

18.2. CÁC CHƯƠNG TRÌNH ỨNG DỤNG

a. Chương trình 1

Giả sử trong máy của bạn có lưu trữ điểm của học sinh

theo từng lớp. Hãy lập chương trình để xem điểm trung bình của một lớp bất kỳ.

Nội dung chương trình

```
clear
close all
TEP=space(8)
@ 10,20 say 'Bạn cần xem lớp nào: ' get TEP
read
use &TEP
display all fiel HODEM,TEN,TBHK
**********
```

b. Chương trình 2

Giả sử bạn là kế toán trong Công ty. Trong máy của bạn có các chứng từ được ghi theo từng tháng ở các tệp. Hãy lập chương trình để xem chứng từ của một tài khoản bất kỳ của một tháng bất kỳ.

Xin các bạn lưu ý: Tên các trường là TKC (tài khoản có) TKN (tài khoản nợ)

Nội dung chương trình

```
clear
set talk off
close all
TEP=Space(8)
TK=space(5)
@ 10,10 say 'Tên tệp làm việc:' get TEP
@ 11,10 say 'Tên tài khoản:' get TK
read
```

```
use &TEP
set filter to TKC=&TK or TKN=&TK
display all fiel SOCT.NGAY.ND.TKC.TKN.TIEN
set talk off
set filter to
Return
**********
```

c. Chương trình 3

Lập chương trình nhập dữ liệu về điểm của học sinh trong một trường phổ thông trung học gồm 12 môn. Mỗi môn gồm bốn trường là Điểm hệ số 1, Điểm hệ số 2, Điểm thi và trung bình học kỳ, trong đó ba trường đầu phải nhập dữ liệu (trường thứ tư máy sẽ tính). Với yêu cầu là điểm được nhập theo môn, nghĩa là các điểm hệ 1, hệ 2, điểm thi được nhập đồng thời.

Xin bạn lưu ý: Nội dung chương trình dưới đây chỉ để bạn tìm hiểu xem lệnh & tiện lợi như thế nào. Để chương trình sử dụng được đầy đủ, bạn nên "trang điểm" lại một chút cho phù hợp với công việc của bạn.

Để thực hiện chương trình này, trước hết bạn nên tạo một tệp CSDL là MON.DBF gồm các trường sau:

Fields name	Type	Width	Dec
TENMON	Character	11	
HE1	Character	11	
HE2	Character	11	
THI	Numeric	4	1
TBHK	Numeric	4	1

Tệp này có công dụng: trường TENMON ghi tên các môn cần tính điểm, các trường HE1, HE2, THI, TBHK có nội dung ghi tên các trường về điểm hệ 1 ,hệ 2, thi, trung bình học kỳ của môn đó trong tệp ghi điểm của học sinh. Việc đặt tên trường cho tệp ghi điểm thống nhất theo bảng sau:

TENMON	HE1	HE2	THI	TBHK
VAN	VANHE1	VANHE2	VANTHI	VANTBHK
SU	SUHE1	SUHE2	SUTHI	SUTBHK
DIA	DIAHE1	DIAHE2	DIATHI	DIATBHK
TOAN	TOANHE1	TOANHE2	TOANTHI	TOANTBHK
LY	LYHE1	LYHE2	LYTHI	LYTBHK
HOA	HOAHE1	HOAHE2	HOATHI	HOATBHK
CDAN	CDANHE1	CDANHE2	CDANTHI	CDANTBHK
NN	NNHE1	NNHE2	NNTHI	NNTBHK
SVAT	SVATHE1	SVATHE2	SVATTHI	SVATTBHK
KTCN	KTCNHE1	KTCNHE2	KTCNTHI	KTCNTBHK
KTNN	TNNHE1	KTNNHE2	KTNNTHI	KTNNTBHK
TDUC	TDUCHE1	TDUCHE2	TDUCTHI	TDUCTBHK

Có nghĩa là môn văn trong tệp ghi điểm có trường ghi điểm hệ số 1 là VANHE1, trường ghi điểm hệ số 2 là VANHE2...

Xin các bạn lưu ý:

- Tất cả các trường hệ số 1 là dạng ký tự độ rộng 20 vị trí
- Các trường hệ số 2 là dạng ký tự độ rộng 12 vị trí
- Các trường điểm thi và trường trung bình học kỳ dạng số (Numeric) độ rộng là bốn vị trí trong đó có một vị trí thập phân.
- Ngoài ra cần khai báo thêm trường HODEM (họ đệm) và

'EN (tên) dạng ký tự và nên khai báo thêm trường trung bình tất cả các môn trong học kỳ dạng Numeric và trường Hạnh kiểm dạng ký tự.

+ Các điểm gõ vào cách nhau một ký tự.

Giả sử tệp CSDL về điểm của học sinh đã có họ tên của học sinh.

Nội dung chương trình

```
**********************
set color to w/b+
set talk off
set safety off
clear all
set status off
set scoreboard off
public s.tb.j
mon=space(6)
lop=space(6)
@ 12.20 say 'Cần nhập điểm cho lớp nào? ' get lop
read
do while .t.
   clear
   @ 3.25 to 20.55 double
   @ 4.27 say 'Nhập Điểm môn nào ?'
   @ 5.35 say repl('-',10)
   @ 6.30 prompt 'Văn'
   @ 7.30 prompt 'Sử'
   @ 8.30 prompt 'Địa'
   @ 9.30 prompt "Toán'
```

```
@10,30 prompt 'Lý'
@11,30 prompt 'Hóa'
@12,30 prompt 'Công dân'
@13,30 prompt 'Ngoại ngữ'
@14,30 prompt 'Sinh vật'
@15,30 prompt 'KTCN'
@16,30 prompt 'KTNN'
@17,30 prompt 'Thể dục'
@18,30 prompt '**Kết thúc'
menu to chon
Do case
case chon=1
    m='VAN'
case chon=2
    m='SU'
case chon=3
    m='DIA'
case chon=4
    m='TOAN'
case chon=5
    m='LY'
case chon=6
    m='HOA'
case chon=7
    m='CDAN'
case chon=8
    m='NN'
case chon=9
```

```
    m='SVAT'
case chon=10
    m='KTCN'
case chon=11
    m='KTNN'
case chon=12
    m='TDUC'
case chon=13
    clear
    exit
Endcase
select 2
use mon
locate all for ten=m
m1=he1
m2=he2
m3=thi
m4=tblik
select 1
use &lop
clear
dau=0
cuoi=0
count to so
@1.21 to 3.50 @2.22 say 'Lớp có: '+space(10)+' Học sinh'
set color to w/B+*
@2.30 say str(so,2)
set color to w/b+
```

```
Do while .t.
   @8.18 to 13.60 double
   @10.20 say 'Nhập/sửa từ học sinh thứ: ' get dau;
                                                picture'99'
   @11.20 say ' đến học sinh thứ: ' get cuoi picture'99'
   read
   if cuoi<so or dau>cuoi
      @19.23 to 21.56  set color to w/b+*
      @20.30 say 'Sai dữ liệu'
      set color to w/b+
   else
      @20,20
      exit
   endif
   Enddo
   clear
define window nhap from 5.20 to 17.60 panel
activate window nhap
@1.1 say 'Môn: '
@1.28 say 'H/sinh: '
@2.1 say replicate('-',37)
do while dau<=cuoi
   set color to w/b+*
   @1.5 say m
   @1.36 say str(dau,2)
   set color to w/b+
   go dau
   @3.7 say 'Ho va ten: '+HODEM+TEN
```

```
    @4,3 say replicate('*',35)
    @5,3 say 'Hệ số 1: ' get &m1
    @6,3 say 'Hệ số 2: ' get &m2
    @7,3 say 'Điểm thi: ' get &m3
    read
    do tinh with &m1
    tong1=s
    sl1=j
    do tinh with &m2
    tong2=s
    sl2=j
    tbkt=round((tong1+tong2*2)/(sl1+sl2*2),1))
    tb=round((tbkt*2+&m3)/3,1))
    @8,3 say 'T/bình Học kỳ: '
    set color to w/b+*
    @8,20 say str(tb,4,1)
    set color to w/b+ kt=' '
    @9,3 say 'Đã dúng chưa:(D/S)' get kt function"@!";
                                                valid kt$'DS'
    read
    if kt='D'
       replace &m4 with tb
       dau=dau+1
       @8,20 say space(4)
    endif
Enddo
release window nhap
Enddo
```

```
release window nhap
set talk on
set scoreboard on
Return
*************
Procedure TINH
   param c
   dimension
   aa(18)
   aa=0
   j=0
   c1=ltrim(c)
   n=len(c1) j=0
   s=0
   if n>0
      i=1
      j=1
      do while c1#' '
         aa(i)=val(subs(c1,1,at(' ',c1)-1))
         i=i+1
         c1=subs(c1,at(' ',c1)+1)
         j=j+1
      enddo
      j=j-1
      s=0
      i=1
      do while i<=j
         s=s+aa(i)
```

```
            i=i+1
            enddo
         endif
      return
```

Xin giải thích qua một chút. Trong chương trình trên có một thủ tục (Procedure) là TINH. Thủ tục này dùng để chuyển các điểm đưa vào dưới dạng ký tự thành dạng số và tính tổng số điểm của từng hệ số (biến s) và số điểm của hệ số đó (biến J). Chương trình chính ghi nhận các giá trị đó sau mỗi lần gọi và tính điểm trung bình kiểm tra theo công thức:

$$\text{TBKT} = \frac{\text{T/số điểm hệ số 1} + (\text{T/số điểm hệ số 2}) \times 2}{\text{Số lần điểm hệ số 1} + (\text{số lần điểm hệ số 2}) \times 2}$$

(câu lệnh: tbkt=round((tong1+tong2*2)/(sl1+sl2*2).1))

Điểm trung bình học kỳ được tính theo công thức:

$$\text{TBHK} = \frac{(\text{Trung bình kiểm tra}) \times 2 + \text{điểm thi}}{3}$$

(câu lệnh: tb=round((tbkt*2+&m3)/3.1))

Nếu bạn có điều kiện bạn nên sửa lại thủ tục TINH để có thể kiểm tra nếu một điểm đưa vào lớn hơn 10 thì chương trình báo lỗi (ở trên chính là các phần tử của mảng aa).

Chương 19
XỬ LÝ DỮ LIỆU KIỂU MEMO

19.1. TẠO CỬA SỔ CHO TRƯỜNG MEMO

Cú pháp

SET WINDOW OF MEMO <tên window> trong đó <tên window> đã được khai báo bởi lệnh DEFINE WINDOW.

Lệnh này thông báo cho FoxPro biết cửa sổ để sửa, xem dữ liệu trường memo là cửa sổ nào (trong các lệnh sửa, xem dữ liệu: EDIT, BROWSE, CHANGE...)

Xin các bạn lưu ý: Trong các lệnh trên, nếu muốn vào khung cửa sổ, bạn phải đi con trỏ vào trường memo rồi ấn Ctrl+Home, Ctrl+PgUp, Ctrl+PgDn. Tuy nhiên, khung cửa sổ sẽ tự động hiện lên mà không cần ấn Ctrl+Home...nếu trong lệnh @ GET bạn có ghi điều khoản OPEN.

19.2. NHẬP DỮ LIỆU KIỂU MEMO

a. Sử dụng lệnh Create

Giả sử bạn cần tạo một tệp CSDL gồm có các trường sau:

Fields name	Type	Width	Dec
HOVATEN	Character	21	
QTCT	Memo	10	

Trong đó có một trường (trường QTCT) có dữ liệu kiểu memo để quản lý quá trình công tác của cán bộ. Bạn hãy khai báo cấu trúc theo yêu cầu trên và vào dữ liệu cho tệp CSDL này. Khi bạn viết xong tên của một cán bộ thì con trỏ nhảy xuống ở chữ "memo". Ví dụ như sau:

HOVATEN Nguyễn Văn Trinh
QTCT memo

Ấn Ctrl+PgUp. Ctrl+PgDn hoặc Ctrl+Home để hiện lên một khung cửa sổ. Bạn hãy vào dữ liệu cho trường QTCT của bản ghi này như khi soạn thảo một văn bản. Sau khi vào dữ liệu xong, ấn Ctrl+W để ghi và kết thúc việc vào dữ liệu cho trường này.

Xin các bạn lưu ý

- Khi một trường memo đã có dữ liệu, nếu bạn dùng bất cứ một lệnh xem nội dung của một bản ghi như display, list, browse...nếu trường memo đã có dữ liệu sẽ xuất hiện chữ "Memo" (chữ M hoa). Nếu trường memo chưa có dữ liệu thì sẽ xuất hiện chữ memo (chữ m thường).

- Số trường memo tối đa có thể tạo cho một tệp CSDL là 255.

b. Lệnh APPEND MEMO

Dữ liệu dạng văn bản có thể nạp vào trường memo qua câu lệnh:

APPEND MEMO <trường memo> FROM <tệp> [OVERWRITE]

Lệnh này chuyển dữ liệu ở tệp có tên là <tên tệp> vào <trường memo>, trong đó <tên tệp> phải chỉ thị đầy đủ cả ổ

đĩa, đường dẫn (nếu không ở trong thư mục hiện thời hoặc ở trong thư mục đã khai báo đường dẫn) và phần mở rộng. Dữ liệu sẽ được ghi nối đuôi lên <trường memo> nếu không có chỉ thị OVERWRITE. Ngược lại, nếu có chỉ thị OVERWRITE thì dữ liệu sẽ ghi đè lên <trường memo> (dữ liệu cũ bị mất).

c. Các lệnh BROWSE, EDIT, CHANGE

Trong các lệnh trên, bạn đưa con trỏ đến trường memo sau đó ấn: Ctrl+PgUp, Ctrl+PgDn, Ctrl+Home để vào dữ liệu, sau đó ấn Ctrl+End để kết thúc.

19.3. HIỂN THỊ DỮ LIỆU KIỂU MEMO

a. Lệnh @...GET

Cú pháp

@<tọa độ> GET <biến trường memo>
[[OPEN] WINDOW <tên cửa sổ>]

Nếu có chỉ thị WINDOW <tên cửa sổ> thì FoxPro cho phép bạn đưa dữ liệu của <biến trường memo> ra khung cửa sổ do <tên cửa sổ> quy định.

Nếu bản ghi có nhiều trường memo, muốn hiển thị trường nào thì bạn dùng lệnh @..GET trường ấy.

Nếu có chỉ thị OPEN, Foxpro tự động mở khung cửa sổ và hiện lên màn hình. Nếu muốn hiển thị nhiều trường memo, con trỏ sẽ dừng ở trường memo do @...GET được khai đầu tiên. Dùng phím TAB để di chuyển về trường Memo khác thuộc bản ghi. Bạn có thể hiệu chỉnh các dữ liệu này.

Muốn kết thúc, có thể:

- Ấn Ctrl+W hoặc Ctrl+End để ghi lại những gì vừa thay

đổi.

- Ấn Ctrl+Q hoặc Esc để thoát ra và không ghi lại những thay đổi. Trong trường hợp này FoxPro sẽ hỏi lại bạn:

Discard change Yes/No ?

Hãy trả lời Y nếu không muốn ghi lại những thay đổi.

b. Dùng các lệnh khác

- Có thể dùng các lệnh EDIT, CHANGE, BROWSE giống như để nhập dữ liệu.

- Dùng các lệnh DISPLAY, LIST.

Các lệnh này có thể ghi dữ liệu kiểu memo ra một tệp khác trên đĩa hoặc in ra máy in. Nhưng bạn phải ghi rõ tên trường memo, nếu không, hai lệnh này chỉ liệt kê tên các trường của tệp CSDL mà thôi. Khi này trường memo chỉ xuất hiện chữ memo.

19.4. CÁC LỆNH VÀ CÁC HÀM LIÊN QUAN ĐẾN TRƯỜNG MEMO

a. Lệnh SET MEMOWIDTH

Cú pháp

SET MEMOWIDTH TO <bt số>

Lệnh này ấn định chiều dài kết xuất trường memo được hiển thị lên màn hình hoặc khung cửa sổ (hoặc in ra máy in, ghi ra đĩa).

Giá trị ngầm định là 50 byte, có thể thay đổi <bt số> từ 8 đến 256 byte.

Ví dụ:

SET MEMOWIDTH TO 70

Đặt chế độ đưa dữ liệu ra màn hình (hoặc máy in) 70 ký tự/ 1 dòng.

b. Hàm AT()/ATC()/RAT()

Cú pháp

AT(<bt ký tự>,<trường memo>[,<bt số>])

ATC(<bt ký tự>,<trường memo>[,<bt số>])

RAT(<bt ký tự>,<trường memo>[,<bt số>])

Các hàm trên dùng để tìm xem <bt ký tự> có trong <trường memo> hay không. Nếu có, cho kết quả là vị trí xuất hiện <bt ký tự>. Nếu không, cho kết quả giá trị bằng 0.

Nếu có chỉ thị <bt số>, các hàm sẽ dò tìm lần xuất hiện <bt số> của <bt ký tự> trong <trường memo> nằm ở vị trí nào.

Hàm AT() và ATC() tìm từ trái sang phải còn hàm RAT() thì tìm từ phải sang trái.

Ví dụ:

?AT('huy chương',TIEUSU)

c. Hàm ATLINE()/ATCLINE()/RATLINE()

Cú pháp

ATLINE(<bt ký tự>,<trường memo>)

ATCLINE(<bt ký tự>,<trường mẹmo>)

RATLINE(<bt ký tự>,<trường memo>)

Ba hàm trên tìm xem chuỗi ký tự <bt ký tự> có xuất hiện trong <trường memo> hay không. Hai hàm ATLINE() và ATCLINE() bắt đầu tìm từ đầu chuỗi trở đi (trái qua phải) còn hàm RATLINE() tìm từ cuối chuỗi trở đi (phải qua trái).

Nếu việc tìm thành công thì giá trị nhận được là số thứ tự

của dòng mà <bt ký tự> xuất hiện trong <trường memo>.

Nếu không tìm thấy, giá trị nhận được là 0. Số thứ tự của dòng mà các hàm trên đưa ra sẽ tùy thuộc vào lệnh SET MEMOWIDTH, nên nếu bề dài do SET MEMO WIDTH ấn định khác nhau thì số thứ tự của dòng do các lệnh trên đưa ra cũng khác nhau.

Ví dụ: Trong tệp CSDLNHANSU.DBF dùng để quản lý nhân sự trong cơ quan. Muốn kiểm tra xem cán bộ nào đã nhận huy chương, tiến hành các bước như sau:

```
use NHANSU
store to 'Huy chương' to chuoi
locate for atline(chuoi,TIEUSU)#0
display fiel HODEM,TEN
```

Kết quả:

```
Nguyễn Văn Trinhcontinue
```

d. Hàm MEMLINES()

Cú pháp

```
MEMLINES(<trường memo>)
```

Hàm MEMLINES(), viết tắt chữ memo lines cho biết tổng số dòng của <trường memo>. Tổng số dòng này tùy thuộc vào lệnh SET MEMOWIDTH.

Ví dụ:

```
use NHANSU
set memowidth to 50
?MEMLINES(TIEUSU)
```

Kết quả:

```
30
```

e. Hàm MLINE()

Cú pháp

MLINE(<trường memo>,<bt số>[,bt số 2>])

Hàm MLINE() cho biết nội dung cụ thể của một dòng nào đó của <trường memo>.

Ví dụ:

```
close all
use NHANSU
store 'Huy chương' to chuoi
locate for atline(chuoi.TIESU)#0
x=atline(chuoi.TIEUSU)
?HODEM.TEN
?'Dòng thứ:'+str(x.3)+ 'là: '+mline(TIEUSU.x)
```

Kết quả:

```
Nguyễn Văn Trinh
Dòng thứ 3 là: Đã nhận 5 Huy chương vàng
```

19.5. CÁC CHƯƠNG TRÌNH ỨNG DỤNG

a. Chương trình 1

Giả sử bạn có một tệp CSDL là HOSO.DBF, trong đó có trường memo QTCT quản lý quá trình công tác của từng cán bộ. Lập chương trình để in ra danh sách các cán bộ đã học tập ở Liên xô (cũ).

Nội dung chương trình

```
set talk off
use HOSO
?' Danh sách '
```

```
?' Đã học tập tại Liên xô (cũ)'
store 'Liên xô' to CHUOI
Do while not eof()
   if atline(CHUOI,QTCT)
      ?HOVATEN
   endif
   skip
Enddo
Return
*********
```

b. Chương trình 2

Giả sử bạn có tệp CSDL là K_HANG.DBF trong đó có trường memo là DIACHI để quản lý địa chỉ của khách hàng. Trong tệp này còn có trường HOVATEN và trường NGAYTRA dạng Date để quản lý ngày phải trả tiền của khách. Hãy lập chương trình in ra danh sách các khách hàng (gồm họ và tên, địa chỉ, số ngày quá hạn) đến ngày phải trả hoặc đã quá ngày phải trả tiền với yêu cầu địa chỉ chỉ in 20 ký tự trên một dòng.

Nội dung chương trình

```
set talk off
clear
set device to screen
set date French
set memowidth to 20 NGAY=date()
USE K_HANG
set filter to NGAY>=NGAYTRA
```

```
@1.10 say ' Danh sách các khách hàng phải trả nợ'
@2.20 say 'Tính đến ngày: ' +dtoc(NGAY)
@3.20 say replicate('-',20) h=4 st=1
Do while not eof()
   if h>20
      wait 'Hãy ấn một phím bất kỳ để xem tiếp'
      @4.0 clear to 24.79
      h=4
   endif
@h,5 say str(st,2)+'.'
@h.10 say HOVATEN
@h+1.10 say 'Ngày phải trả: '+DTOC(NGAYTRA)
@h+2.10 say 'Quá hạn:'+str(NGAY-NGAYTRA,3)+;
                                                    'ngày'
i=memline(DIACHI)
@h+3.10 say 'Địa chỉ: '
j=1
Do while j<=i
   @h+3,20 say mline(DIACHI,j)
   j=j+1
enddo
h=h+3+i
st=st+1
skip
Enddo
set filter to
set talk on
Return
```

****

Kết quả khi thực hiện chương trình có dạng:

Danh sách các khách hàng phải trả nợ

Tính đến ngày:06/03/02

1. Giang Thu Khánh Ngọc

Ngày phải trả: 03/03/o2

Quá hạn: 3 ngày

Địa chỉ:

Nhà 1000c, tổ 50

Phường Phương Liên

Đống đa - Hà Nội

2. Giang Hồng Vân Anh

Ngày phải trả: 06/06/o2

Quá hạn: 0 ngày

Địa chỉ:

Phòng 64 nhà N3 Khu tập thể K83

Từ Liêm - Hà Nội

****

Chương 20
LÀM VIỆC VỚI NHIỀU TỆP CSDL

20.1. ĐỊNH VÙNG LÀM VIỆC CHO TỆP CSDL

Foxpro cho phép mở đồng thời tối đa 25 tệp CSDL với điều kiện phải chỉ định vùng làm việc khác nhau cho mỗi tệp CSDL được mở. Nếu không, khi mở tệp thứ hai bằng lệnh USE thì tệp thứ nhất bị đóng lại vì chúng cùng trên một vùng.

Ví dụ: Cách mở hai tệp như sau là không thực hiện được.

```
USE MSKH
USE K_HANG
```

Khi mở tệp K_HANG.DBF thì tệp MSKH.DBF đã bị đóng lại mất rồi.

Trong trường hợp này phải dùng lệnh SELECT <vùng làm việc> để chỉ định vùng trước khi mở tệp CSDL. Mỗi vùng làm việc như vậy phải gán cho nó một cái tên để khỏi nhầm lẫn và dễ nhận diện. Việc gán tên được quy định như sau:

- Mỗi chữ cái từ A đến J đối với 10 vùng làm việc đầu tiên.
- Một số từ 1 đến 25 (tương ứng với 25 vùng).
- Một tên nào đó được gọi là bí danh (Alias).

Nếu mở đồng thời hai tệp như trên, có thể thực hiện như sau:

```
SELECT 1                && có thể dùng SELECT A
USE MSKH
```

```
SELECT 2
USE K_HANG          && có thể dùng SELECT B
```

Cách dùng như trên để thông báo cho FOXPRO mở hai tệp CSDL ở hai vùng khác nhau và chúng luôn ở trạng thái sẵn sàng làm việc.

Ngoài ra Foxpro còn cho phép mở một tệp CSDL trên nhiều vùng khác nhau qua từ khóa AGAIN.

Ví dụ:

```
SELECT 1
USE MSKH
SELECT 4
USE K_HANG
SELECT 16
USE MSKH AGAIN
```

Hai câu lệnh cuối có thể thay bằng:

```
USE MSKH AGAIN IN 16
```

Khi muốn làm việc với tệp MSKH.DBF, chỉ cần chỉ thị:

```
SELECT 1 && hoặc SELECT A
```

Khi làm việc với tệp K_HANG.DBF, bạn hãy chỉ thị:

```
SELECT 2 && hoặc SELECT B
```

mà không cần mở lại chúng. Điều này làm tăng tốc độ thực hiện công việc, vì không phải mở lại tệp và rất thuận tiện khi công việc đòi hỏi phải xử lí nhiều tệp có liên quan đến nhau.

Việc định vùng làm việc không nhất thiết phải theo một thứ tự nào. Khi mở đồng thời nhiều tệp CSDL trên các vùng khác nhau, các lệnh dịch chuyển, định vị con trỏ bản ghi ở vùng này hoàn toàn không ảnh hưởng đến con trỏ bản ghi ở vùng khác, trừ khi chúng được liên kết với nhau bởi lệnh SET RELATION.

Xin bạn lưu ý: Sau khi khởi động Foxpro hoặc sau khi dùng lệnh đóng tệp CSDL như Close all, Close Databases, vùng làm việc hiện thời luôn là 1 (vùng A).

20.2. GÁN BÍ DANH CHO CSDL

a. Bí danh là gì ?

Như chúng tôi đã trình bày ở trên, Foxpro cho phép mở đồng thời 25 tệp CSDL. Các tệp thường viết tắt nên có thể bị nhầm lẫn. Để khắc phục tình trạng này, Foxpro cho phép bạn gán bí danh trong các câu lệnh khi mở tệp.

Cú pháp:

USE <tệp CSDL> ALIAS <bí danh>

Bí danh là một dãy tối đa 10 kí tự, không chứa dấu trống. Không đặt trong hai dấu nháy đơn, nháy kép. Không dùng bí danh cho hai hay nhiều tệp được mở đồng thời.

Ví dụ:

```
SELECT 10
USE QLNS ALIAS QLNHANSU
```

và khi cần đến vùng này làm việc bạn có thể chỉ thị:

```
SELECT 10
```

hoặc:

```
SELECT QLNHANSU
```

Rõ ràng là cách thứ hai dễ nhớ, đỡ nhầm lẫn hơn.

Ví dụ trên là bí danh do người sử dụng gán, ngoài ra FoxPro có thể tự gán bí danh cho tệp CSDL.

b. Bí danh do Foxpro tạo ra

Trong nhiều trường hợp, Foxpro phải tự động gán bí danh cho tệp CSDL. Ví dụ: Bạn muốn mở một tệp CSDL trên hai

vùng khác nhau mà quên không gán bí danh cho nó. Foxpro bắt buộc phải can thiệp và gán cho chúng hai bí danh khác nhau. Nếu không, nếu cứ để trạng thái mặc nhiên thì chúng có cùng bí danh và như vậy gây lỗi. Foxpro tự động gán bí danh từ A đến J cho 10 vùng làm việc đầu tiên từ W11 đến W25 đối với 15 vùng còn lại, nếu trong 15 vùng này có sử dụng từ khóa AGAIN và không có từ khóa ALIAS mà tệp CSDL trước đó đã mở cũng không sử dụng từ khóa ALIAS.

Xin các bạn lưu ý:

- Foxpro cho phép dùng câu lệnh SELECT, trong trường hợp này thì vùng làm việc nào thấp nhất sẽ được chọn để gán cho tệp CSDL. Và tốt hơn hết, hãy gán cho tệp CSDL định mở này một bí danh và sử dụng bí danh cho công việc tiếp theo. Điều này giúp bạn đỡ mất thời gian suy nghĩ và chọn lựa.

- Việc sử dụng bí danh là không bắt buộc, song trong một số trường hợp việc gán bí danh cho tệp tỏ ra rất hữu ích. Ví dụ trong chương trình bạn phải mở một tệp CSDL mà tên tệp khi thực hiện chương trình mới được nhận từ bàn phím. Mặc dù tên tệp chưa biết nhưng bạn vẫn có thể gán cho nó một bí danh và làm việc với tệp CSDL thông qua bí danh này. Ví dụ:

```
Input "Tên tệp làm việc" to TEP
select 3
USE TEP ALIAS LAMVIEC
```

20.3. CÁC LỆNH CẬP NHẬT TỪ MỘT TỆP CƠ SỞ DỮ LIỆU KHÁC

a. Lệnh APPEND FROM

Có thể nối dữ liệu từ một tệp CSDL khác vào tệp CSDL đang mở bằng lệnh:

USE <tệp nguồn>

APPEND FROM <tệp bổ sung>

[FIELDS <danh sách trường> [FOR <bt logic>]

Lệnh này sẽ nối dữ liệu từ <tệp bổ sung> vào <tệp nguồn>.

- Nếu có chỉ thị FIELDS <ds trường> thì chỉ những trường nằm trong <ds trường> mới được nối bổ sung. Nếu không có chỉ thị này thì tất cả các trường cùng có trong hai tệp (cùng tên và cùng kiểu) sẽ được chép từ <tệp bổ sung> sang <tệp nguồn>.

- Nếu có FOR <bt logic>, chỉ những bản ghi ở <tệp bổ sung> thỏa mãn <bt logic> mới được chép sang <tệp nguồn>. (*Xin các bạn lưu ý*: trong câu lệnh này không có từ khóa WHILE).

- Những bản ghi nào bị đánh dấu xóa (dấu *) sẽ không được chép sang <tệp nguồn> nếu lệnh SET DELETE ON đang có hiệu lực. Ngược lại, những bản ghi này vẫn được chép như thường.

- Nếu độ rộng của trường trong <tệp bổ sung> lớn hơn độ rộng trường tương ứng trong <tệp nguồn>, thì dữ liệu nhận được ở <tệp nguồn> sẽ bị cắt bớt đi nếu là dữ liệu kiểu kí tự (character) hoặc chứa một loạt dấu * nếu là dữ liệu kiểu số.

Ví dụ:

Bạn có tệp DIEM.DBF để quản lý điểm của học sinh trong trường. Sau một thời gian xuất hiện thêm một số học sinh mới từ trường khác chuyển đến và bạn đưa vào tệp DIEMBS.

DBF có cùng cấu trúc với tệp DIEM.DBF. Bạn có thể kết nối hai tệp này lại với nhau bởi lệnh sau:

USE DIEM

APPEND FROM DIEMBS

Chú ý: <tệp bổ sung> phải ở trạng thái đóng khi thực hiện lệnh APPEND FROM.

b. Lệnh UPDATE

Trong trường hợp cập nhật chỉ sửa đổi dữ liệu các trường của bản ghi mà không thêm bản ghi, bạn có thể lưu tất cả các dữ liệu cần cập nhật vào một tệp và dùng lệnh UPDATE.

Cú pháp

UPDATE ON <khóa> FROM <tệp 2>

REPLACE <trường 1> WITH <biểu thức 1>

[, trường 2> WITH <biểu thức 2>]

...

[<RANDOM>]

Trước khi dùng lệnh UPDATE, tệp chính (tệp cần sửa đổi dữ liệu) phải được mở trong vùng làm việc hiện thời và <tệp 2> (tệp mang dữ liệu sửa đổi) phải được mở trong vùng làm việc khác. <khóa> sắp xếp phải là một trường có cả trên hai tệp đều phải sắp xếp theo <khóa> này. Nếu <tệp 2> không được sắp xếp theo <khóa> thì bạn phải chỉ thị thêm từ khóa RANDOM. Nếu khóa gồm nhiều trường kết nối lại (ví dụ: trong tệp TONKHO.DBF, khóa sắp xếp bao gồm mã kho và mã vật tư gộp lại) thì bạn phải:

+ Thay đổi cấu trúc tệp (lệnh modify structure) bổ sung trong cả hai tệp một trường tạm thời có chiều dài bằng tổng các trường kia cộng lại.

+ Dùng lệnh REPLACE thay thế trường tạm thời này các giá trị của các trường kia cộng lại.

+ Sau đó mới dùng lệnh sắp xếp.

Xin các bạn lưu ý:

- <trường...> chỉ là tên trường của tệp chính (tệp cần sửa đổi dữ liệu) và không được dùng là tên trường của <tệp 2>.

- <biểu thức...> có thể là hằng, biến hoặc một biểu thức phức tạp và phải cùng kiểu dữ liệu với <trường...> được cập nhật. <biểu thức> thường chứa một tên trường hoặc một biểu thức có chứa tên trường của <tệp 2>.

- Lệnh UPDATE hoàn toàn không có phạm vi và điều kiện.

Chúng ta tìm hiểu các ví dụ sau đây:

(1). *Lệnh UPDATE không có từ khóa RANDOM*

Giả sử bạn có một tệp CSDL có tên là Luong.dbf và có các trường Luong, Ma, Phucap, Thuclinh. Bạn lại có một tệp khác: tệp Thuong.dbf có các trường Ma, Sotien quản lý tiền thưởng tương ứng với các bản ghi của tệp Luong.dbf (thông qua Ma). Cần phải tính số tiền thực lĩnh của từng cán bộ và đưa vào cột Thuclinh theo công thức:

Thực lĩnh = Lương chính + Phụ cấp + Số tiền

Trong đó: Lương chính và Phụ cấp là giá trị trong tệp Luong.dbf, còn Số tiền là giá trị trong tệp Thuong.dbf.

Các bước tiến hành như sau:

```
Select 2
Use Thuong
Index on Ma to ThuongCS
Use Thuong Index ThuongCS alias Capnhat
```

```
select 1
Use Luong
Index on Ma to LuongCS
Use Luong Index LuongCS
Update on Ma from Capnhat:
    Replace Thuclinh with Luong+Phucap+:
    Capnhat->Sotien
```

(2). *Lệnh UPDATE có từ khóa RANDOM*

Chúng ta trở lại ví dụ trên. Trong trường hợp này <tệp 2> (tệp Thuong.dbf) không cần phải sắp xếp theo <khóa> (theo Ma). Các bạn hãy viết lại như sau:

```
Select 2
Use Thuong alias Capnhat
Select 1
Use Luong Index on Ma to LuongCS
Update on Ma from Capnhat:
   Replace Thuclinh with Luong + Phucap +:
                        Capnhat -> Sotien Random
```

20.4. LỆNH SET RELATION

Lệnh này cho phép kết nối thông tin giữa hai tệp, nghĩa là tệp này có thể bổ sung thông tin cho tệp kia.

Ví dụ:Trong tệp hóa đơn không cần đánh vào tên khách hàng vì như vậy sẽ tốn chỗ và mất thời gian, mà chỉ cần đưa vào mã số khách hàng. Khi nào cần in tên khách hàng, chỉ cần đem mã số khách hàng tra trong tệp hồ sơ khách hàng để tìm ra tên khách hàng tương ứng.

a. Cú pháp

SET RELATION TO

[<bthức 1> INTO <bt số 1>/<bt ký tự 1>]

[,<bthức 2> INTO <bt số 2>/<bt ký tự 2>]

.....

Khi đặt mối quan hệ giữa hai tệp CSDL thì tệp chính phải được mở trong vùng làm việc hiện thời và các tệp phụ (tệp có liên quan) được mở trong vùng khác.

Trên một lệnh SET RELATION có thể đặt mối quan hệ với nhiều tệp phụ khác bằng cách liệt kê các mối quan hệ.

Một điều bắt buộc phải thực hiện trước khi dùng SET RELATION là các tệp có liên quan phải được sắp xếp theo chỉ số theo các <bthức..> sau TO.

b. Các ví dụ

Các bạn hãy tìm hiểu các ví dụ dưới đây để hiểu rõ hơn lệnh SET RELATION.

(1). *Ví dụ 1*

Giả sử chúng ta có hai tệp CSDL: tệp Thang2.dbf và tệp Thang3.dbf quản lý giá cả thị trường của hai mặt hàng gạo và thịt trong tháng hai và tháng ba. Bây giờ hãy so sánh giá cả của từng mặt hàng của từng ngày trong tháng ba với cùng ngày đó trong tháng hai. Chúng ta tiến hành công việc như sau:

```
Select 2
Use Thang2 alias HAI
Select 3
Use Thang3 alias BA
set relation to recno() into HAI
List off next 5 recno(2).Ngay.HAI->Ngay.;
```

Gao,HAI->Gao,Thit,HAI->Thit

Kết quả trên màn hình có dạng như sau:

Recno(2)	NGAY	HAI-NGAY	GAO	HAI->GAO	THIT	HAI->THIT
1	01/03/02	01/02/02	2100	2050	9700	9600
2	02/03/02	02/02/02	2070	2040	9600	9500
3	03/03/02	03/02/02	2050	2080	9700	9700
4	04/03/02	04/02/02	2110	2100	9900	9800
5	05/03/02	05/02/02	2200	2150	9700	9800

Trong ví dụ trên khi con trỏ bản ghi dọc đến bản ghi nào (bản ghi thứ bao nhiêu) của tệp Thang3.dbf thì trong tệp Thang2.dbf con trỏ bản ghi cũng dịch chuyển theo do lệnh ...to recno()... (hàm recno() cho giá trị là số thứ tự của bản ghi hiện thời, nếu recno(2) cho giá trị là số thứ tự của bản ghi hiện thời trong vùng làm việc số 2.

Nếu không dùng lệnh SET RELATION mà chỉ dùng lệnh như sau:

```
select 2
use Thang2 alias HAI
select 3
use Thang3 alias BA
list off next 5 recno(2),Ngay,HAI->Ngay,Gao.:
                                HAI->Gao,Thit,HAI->Thit
```

thì kết quả trên màn hình sẽ là:

Recno(2)	NGAY	HAI-NGAY	GAO	HAI->GAO	THIT	HAI->THIT
1	01/03/02	01/02/02	2100	2050	9700	9600
1	02/03/02	01/02/02	2070	2050	9600	9600
1	03/03/02	01/02/02	2050	2050	9700	9600
1	04/03/02	01/02/02	2100	2050	9900	9600
1	05/03/02	01/02/02	2200	2050	9700	9600

Có thể thấy ngay rằng khi xử lý một bản ghi ở tệp THANG3.DBF thì con trỏ bản ghi ở tệp THANG2.DBF vẫn nằm nguyên ở bản ghi số 1 vì không có lệnh nào làm thay đổi vị trí con trỏ trong bản ghi của tệp THANG2.DBF.

(2). *Ví dụ 2.* Giả sử bạn có ba tệp sau:

- Tệp HSKH.DBF để quản lý hồ sơ khách hàng gồm các trường: MSKH (mã số khách hàng). TENKH (tên khách hàng). DIACHI (địa chỉ). Tệp này có nội dung như sau:

Record #	MSKH	TENKH	DIACHI
1	0001	C/ty DETECH	65 Nguyễn Du
2	0002	C/ty KALBEFAMA	107 Vị Xuyên
3	0003	C/ty Cơ khí Nam hà	109 Cống Hậu
4	0004	C/ty VIGEMTECH	193 Cầu Giấy

- Tệp SANPHAM.DBF để quản lý các mặt hàng trong Công ty gồm các trường: MASP (mã sản phẩm). TENSP (tên sản phẩm). DONVT (đơn vị tính). DONGIA (giá tiền trên một đơn vị sản phẩm). Tệp này có nội dung như sau:

Record#	MASP	TENSP	DONVT	DONGIA
1	0001	Xe đạp	Chiếc	500000
2	0002	Bàn	Chiếc	100000
3	0003	Bia	Hộp	3900
4	0004	Đường	Kg	4100

+ Tệp thứ ba là tệp HOADON.DBF để quản lý các hóa đơn bán hàng gồm các trường: MAKH (mã khách hàng). SOHD (số hóa đơn). MASP (mã sản phẩm). NGAY (ngày bán hàng). SOLUONG (số lượng hàng hóa). Tệp này có nội dung như sau:

Record#	SOHD	MAKH	MASP	NGAY	SOLUONG
1	0017	0001	0003	17/04/02	17
2	0018	0003	0001	18/04/02	13
3	0019	0002	0002	18/04/02	10

Hãy lập chương trình đưa ra báo cáo về hàng hóa gồm đầy đủ các tham số cần thiết: mã khách hàng, tên khách hàng, số hóa đơn, ngày, mã sản phẩm, tên sản phẩm, số lượng, đơn vị tính, đơn giá và thành tiền. Như vậy là phải kết nối ba tệp trên lại với nhau.

Nội dung chương trình

```
set talk off
close all
select 1
use HSKH
index on MAKH to HSKH
use HSKH index HSKH alias MOT
select 2
use SANPHAM
index on MASP to SANPHAM
use SANPHAM index SANPHAM alias HAI
select 3
use HOADON
set relation to MAKH into HSKH.:
                                    MASP into SANPHAM
@1,20 say 'BAO CAO THONG KE BAN HANG'
@2,20 say 'Tháng 04 năm 2002'
@3,30 say 'Công ty DETECH Co.'
@4,0 say replicate('*',80)
```

```
@5.0 say 'STT'
@5.5 say 'Mã KH'
@5.11 say 'Tên k/hàng'
@5.24 say 'Số HĐ'
@5.30 say 'Ngày' @5.39 say 'MãSP'
@5.44 say 'TênS/ph'
@5.53 say 'S/lg'
@5.58 say 'ĐVTính'
@5.65 say 'Đ/giá'
@5.71 say 'Tiền'
@6.0 say replicate('*',80)
st=1
h=7
tong=0
do while not eof() @h,0 say str(st,2)
   @h,5 say MAKH
   @h,11 say MOT->TENKH
   @h,24 say SOHD
   @h,30 say NGAY
   @h,39 say MASP
   @h,44 say HAI->TENSP
   @h,53 say SOLUONG
   @h,58 say HAI->DONVT
   @h,65 say HAI->DONGIA
   tt=SOLUONG * HAI->DONGIA
   tong=tong+tt
   @h,71 say str(tt,8)
   h=h+1
```

```
    st=st+1
  skip
  enddo
  @h,0 say replicate('*',80)
  h=h+1
  @h,10 say 'Tổng số tiền: '
  @h,70 say str(tong,9)
  Return
```

Khi thực hiện chương trình sẽ thu được kết quả sau:

BAO CAO THONG KE BAN HANG

Tháng 04 năm 2002

Công ty DETECH Co.

TT	MãKH	TênKH	SốHĐ	Ngày	MãSP	TênS/ph	S/lg	ĐVTính	Đ/giá	Tiền
1	001	C/tyDETECH	017	17/04/94	003	Bia	17	Hộp	3900	66300
2	003	C/ty B	018	18/04/94	001	Xe đạp	13	Chiếc	500000	6500000
3	992	C/tyC	019	18/04/94	002	Bàn	10	Chiếc	100000	1000000

**** Tổng số tiền: 7566300

Dùng SET RELATION rất tiện lợi vì không phải vào các tham số có tính cố định như tên khách hàng, địa chỉ khách hàng, tên sản phẩm... mà chỉ cần đưa mã của khách hàng, mã của sản phẩm. Khi thực hiện chương trình sẽ tăng tốc độ rõ rệt bởi vì mối quan hệ giữa các tệp đã được thiết lập và đưa vào bộ nhớ.

Chương 21
TÌM KIẾM VÀ XỬ LÝ THÔNG TIN

Trong các chương trước, các bạn đã tìm hiểu một số kỹ thuật lập trình Foxpro. Trong đó đã có một số chương trình tìm kiếm thông tin và kết xuất dữ liệu ra màn hình, ra tệp hoặc máy in. Song trong thực tế xuất hiện một số nhu cầu đòi hỏi người lập chương trình phải đưa ra các khả năng tìm kiếm cơ động hơn. Khi thực hiện chương trình, người thực hiện chương trình mới điền các yêu cầu và máy phải tìm kiếm theo các yêu cầu đó. Chúng ta cứ coi như phải thực sự hỏi đáp giữa người và máy. Ví dụ như khi thực hiện chương trình mới yêu cầu tìm những người có tuổi từ 25 đến 30 ở phòng Kế hoạch. Nhưng ngay sau đó lại yêu cầu đưa ra danh sách những người ở phòng Kỹ thuật hoặc Hành chính có quê quán ở Hà Nội. Tất nhiên, các bạn có thể đưa ra những lệnh tìm kiếm ở ngay khung cửa sổ lệnh. Nhưng mọi chuyện chẳng giản đơn như thế nếu là hồ sơ được lưu trữ trên nhiều tệp, có mã hóa và lại có nhiều trường memo chẳng hạn. Chỉ có lập chương trình mới kết nối chúng lại được với nhau. Ngoài việc giải quyết vấn đề trên, trong chương này còn đưa ra xử lý các kết quả thu được. Cụ thể là người thực hiện chương trình có thể in ấn hoặc đưa ra màn hình chỉ một số trường cần thiết cho công việc. Ví dụ lúc này thì muốn in ra bảng báo cáo gồm họ và tên, quê quán, ngày sinh. Nhưng lúc

khác thì muốn in ra họ và tên, địa chỉ, mức lương hiện tại của cán bộ, mà việc chọn dữ liệu đưa ra chỉ khi thực hiện chương trình mới quyết định.

Bây giờ chúng ta tìm hiểu dần từng vấn đề trên. Để tiện việc theo dõi, chúng tôi chia ra việc tìm kiếm theo từng kiểu dữ liệu và cuối cùng mới tìm kiếm trên các kiểu dữ liệu bất kỳ.

21.1. TÌM KIẾM THÔNG TIN ĐỐI VỚI CÁC TRƯỜNG KÝ TỰ

a. Tìm kiếm với các yêu cầu xảy ra đồng thời

Giả sử các bạn có một tệp (là VIDU.DBF) quản lý cán bộ gồm có các trường:

Field	Field Name	Type	Width	Dec	Index
1	HOTEN	Character	20		
2	QUEQUAN	Character	10		
3	DONVI	Character	10		
4	CHUCVU	Character	10		
5	TDVH	Character	5		

Hãy lập chương trình với các yêu cầu sau:

- Khi thực hiện chương trình, trên màn hình xuất hiện:

  Họ và tên
  Quê quán
  Đơn vị
  Chức vụ
  Trình độ văn hóa
  ** Kết thúc

- Khi đưa vệt sáng đến một cột nào đó và ấn Enter (ví dụ

như cột quê quán chẳng hạn) thì ở phía dưới xuất hiện:

Chọn yêu cầu:

Quê quán:

và bên cạnh chữ quê quán để bạn đưa vào quê quán cần tìm kiếm (ví dụ như 'Hà Nội').

- Sau đó bảng trên lại xuất hiện để bạn chọn chỉ tiêu khác. Ví dụ như "Trình độ văn hóa' yêu cầu 'Đại học'.

- Nếu đưa con vệt sáng đến cột '** Kết thúc' và ấn Enter thì kết thúc việc vào các yêu cầu. Ví dụ như ở trên phải đưa ra danh sách các cán bộ có quê quán ở Hà Nội có trình độ văn hóa là Đại học.

Sau đây là nội dung chương trình:

```
********
clear
set talk off
close all
use VIDU
n=fcount()
dimension m(n),tb(n+1)
tb(1)='Họ và tên'
tb(2)='Quê quán '
tb(3)='Đơn vị'
tb(4)='Chức vụ'
tb(5)="Trình độ văn hóa'
tb(6)='Kết thúc'
i=1
Do while i<=n
   m(i)=fiel(i)
```

```
    i=i+1
Enddo
dk=' '
Do while .t.
    clear
    @ 10,20 menu tb,n save
    read menu to chon
    if chon=n
        clear
        exit
    endif
    yc=space(20)
    clear
    @ 12,20 say 'Chọn yêu cầu: '
    @ 13,20 say tb(chon) get yc
    read
    tg=m(chon)
    if dk#' '
        ** Chú ý ở cuối lệnh là dấu nháy đơn nằm giữa hai
        ** dấu nháy kép
        dk=dk+'.and. '+'upper('+tg+')'+;
                                    '='+""+rtrim(alltrim(yc))+""
        else
              dk='upper('+tg+')'+'='+""+upper(rtrim(yc))+""
        ** Câu lệnh trên có ý nghĩa như sau:
        ** Giả sử bạn chọn cột 'Quê quán' và yêu cầu tìm
        ** những người có quê quán ở 'Ha Noi' thì kết quả
        ** của lệnh trên là upper(QUEQUAN)='HA NOI'
```

```
    endif
  ** Câu lệnh dưới dây để các trường đã chọn yêu cầu rồi
  ** thì không được chọn lại.
  tb(chon)='\'+tb(chon)
  Enddo
  set filter to &dk
  list
  set talk on
  Return
*******
```

b. Chương trình 1

Tìm kiếm các bản ghi thỏa mãn một trong các yêu cầu đặt ra. Ví dụ như bạn cần tìm những người hoặc là ở phòng Kế hoạch, hoặc là có quê quán ở Hà Nội.

Các bạn chỉ cần sửa lại chương trình trên ở câu lệnh dưới lời ghi chú: ** Chú ý... như sau

```
dk=dk+'.or.'+'upper('+tg+')'+;
                              '='+"'"+rtrim(alltrim(vc))+"'"
```

c. Chương trình 2

Tìm kiếm các bản ghi với các điều kiện xảy ra đồng thời nhưng một điều kiện có thể được chọn nhiều khả năng. Ví dụ tìm những người có đơn vị là Hành chính hoặc Kỹ thuật và Quê quán là Hà nội, hoặc Nam Định hoặc Thái Bình.

Nội dung chương trình

```
***************
  clear
  set talk off
```

```
close all
use VIDU
n=fcount()
dimension m(n),tb(n+1),ss(10)
** Chúng tôi dùng biến n chỉ để cho chương trình có
** tính tổng quát
i=1
Do while i<=n-1
   m(i)=field(i)
   i=i+1
Enddo
tb(1)='Họ và tên'
tb(2)='Quê quán '
tb(3)='Đơn vị'
tb(4)='Chức vụ'
tb(5)='Trình độ văn hóa'
tb(n+1)='Kết thúc'
dk=' '
Do while .t.
   clear
   @ 10,2 menu tb,n+1 save
   read menu to chon
   if chon=n+1
      clear
      exit
   endif
   yc=space(50)
   clear
```

```
@12,20 say 'Chọn yêu cầu: '
@13,10 say tb(chon) get yc
   ** Khi đưa vào các yêu cầu, các bạn chú ý:
   ** Nếu cần tìm người có Quê quán là Ha Noi hoặc
   ** Nam Dinh, thì viết: Ha Noi.or.Nam Dinh
read
tg=m(chon)
i=1
y1=alltrim(yc)
so=at('.or.',y1)
if so>0
   do while len(y1)#0
   ** Đoạn chương trình dưới đây đưa vào các phần
   ** tử của mảng các yêu cầu. Ví dụ tìm theo đơn vị
   ** bạn gõ: Hanh chinh.or.Ky thuat, thì phần tử
   ** thứ nhất (ss(1)) là Hanh chinh, phần tử thứ
   ** hai của mảng là: Ky thuat
   if at('.or.',y1)#0
      ss(i)=substr(y1,1,at('.or.',y1)-1)
      y1=substr(y1,at('.or.',y1)+4)
      i=i+1
      else
      ss(i)=y1
      y1=''
    endif
Enddo
else
   ss(1)=alltrim(yc)
```

```
        i=1
endif
*******
j=1
yeucau=' '
do while j<=i
   if yeucau=' '
   ** Chú ý các dấu nháy đơn nằm trong hai dấu
   ** nháy kép để được: ví dụ. DONVI='KY THUAT'
   yeucau='upper('+tg+')'+'='+"'"+upper(ss(j))+"'"
   else
      yeucau=yeucau+'.or.'+'upper('+tg+')'+'='+"'"+;
                                          upper(ss(j))+"
   endif
    j=j+1
   enddo
** Câu lệnh dưới để nhóm các khả năng cho một yêu
** cầu lại với nhau để kết hợp với các yêu cầu của
** trường khác
   yeucau='('+yeucau+')'
   if dk#' '
         dk=dk+'.and.'+yeucau
      else
         dk=yeucau
   endif
   tb(chon)='\'+tb(chon)
enddo
set filter to &dk
```

```
    list
    set talk on
return
*******
```

d. Chương trình 3

Đề nghị các bạn tự lập chương trình để tìm các bản ghi chỉ cần xảy ra một trong các trường hợp. Ví dụ tìm những người có Quê quán là Hà Nội hoặc Nam Hà, đơn vị là Kế hoạch hoặc Kỹ thuật. (Các bạn chỉ cần thay từ .and. ở cuối chương trình trên bằng từ .or.)

e. Chương trình 4

Lập chương trình tìm kiếm các bản ghi theo các trường ký tự với yêu cầu bất kỳ. Ví dụ: Tìm các bản ghi có Quê quán ở Hà Nội hoặc Hải Phòng và chức vụ là Trưởng phòng.

Nội dung chương trình

```
    clear
    set talk off
    close all
    use VIDU
    n=fcount()
    dimension m(n),tb(n+1),ss(10)
    i=1
    Do while i<=n-1
       m(i)=fiel(i)
       i=i+1
    Enddo
    tb(1)='Họ và tên'
```

```
tb(2)='Quê quán '
tb(3)='Đơn vị'
tb(4)='Chức vụ'
tb(5)='Trình độ văn hóa'
tb(n+1)='Kết thúc'
dk=' '
hd=' '
bay=0
Do while .t.
   clear
   @10,2 menu tb,n+1 save
   read menu to chon
   if chon=n+1
      clear
      exit
   endif
** Lệnh if bay#0 để điều khiển việc chọn biểu thức kết
** hợp chỉ tiến hành đối với yêu cầu thứ hai trở đi
if bay#0
   @0,18 to 5,40 double
   @1,20 say 'Chọn kiểu kết hợp:'
   @2,20 say replicate('*',20)
   @3,20 prompt 'VA ' @4,20 prompt 'HOAC '
   menu to cc
   if cc=1
      hd='.and.'
   else
      hd='.or.'
```

```
    endif
endif
bay=1
yc=space(50)
clear
@12,20 say 'Chọn yêu cầu: '
@13,10 say tb(chon) get yc
read
tg=m(chon)
i=1
y1=alltrim(yc)
so=at('.or.',y1)
if so>0
    Do while len(y1)#0
        if at('.or.',y1)#0
            ss(i)=substr(y1,1,at('.or.',y1)-1)
            y1=substr(y1,at('.or.',y1)+4)
            i=i+1
            else
            ss(i)=y1
            y1=''
        endif
    Enddo
else
    ss(1)=alltrim(yc)
    i=1
endif
j=1
```

```
    yeucau=' '
    Do while j<=i
       if yeucau=' '
          yeucau='upper('+tg+')'+'='+"'"+upper(ss(j))+"'"
          else
          yeucau=yeucau+'.or.'+'upper('+tg+')'+:
                                          '='+"'"+upper(ss(j))+"'"
       endif
       j=j+1
    Enddo
    yeucau='('+yeucau+')'
    ** Chú ý: biến hd đã nhận giá trị là .or. hoặc.and. từ
    ** trên
    if dk#' '
       dk=dk+hd+yeucau
       else
          dk=yeucau
    endif
    tb(chon)='\'+tb(chon)
  Enddo
  set filter to &dk
  list
  set talk on
  Return
*******
```

Đề nghị các bạn dựa vào các chương trình trên để lập chương trình tìm kiếm theo các yêu cầu phức tạp hơn. ví dụ: (các dấu mở đóng ngoặc dưới đây để nhóm các yêu cầu) (Qué

quán ở Ha Noi hoặc Hai Phong) và (chức vụ là trưởng hoặc phó phòng) hoặc (trình độ đại học hoặc trên đại học)).

Chú ý việc nhóm các điều kiện là bất kỳ. Để lập được chương trình này, các bạn chỉ cần bổ sung vào Chương trình 5 một đoạn chương trình để hỏi có cần mở ngoặc không trước khi đưa vào yêu cầu cho một trường. Và sau khi vào yêu cầu xong cho trường đó thì lại hỏi đã đóng ngoặc chưa.

21.2. TÌM KIẾM THÔNG TIN ĐỐI VỚI CÁC TRƯỜNG SỐ

Việc tìm kiếm thông tin đối với trường số tương tự như đối với trường ký tự. Song các bạn cần lưu ý là đối với trường số có các khả năng xảy ra như sau:

Bằng một giá trị.

Nhỏ hơn (hoặc bằng) một giá trị.

Lớn hơn (hoặc bằng) một giá trị.

Lớn hơn (hoặc bằng) một giá trị và nhỏ hơn (hoặc bằng) một giá trị khác.

Bây giờ chúng ta hãy xét một ví dụ đơn giản sau.

Giả sử bạn có tệp LUONG.DBF gồm các trường sau:

Field	Field Name	Type	Width	Dec	Index
1	HOTEN	Character	18		
2	DONVI	Character	3		
3	LUONG	Numeric	6		
4	THUONG	Numeric	6		
5	PHUCAP	Numeric	6		
6	TAMUNG	Numeric	6		

Hãy lập chương trình tìm các bản ghi với yêu cầu đưa vào cho một hoặc hai trường là bất kỳ.

```
** Nội dung chương trình
    clear
    set talk off
    close all
    use luong
    n=fcount() dimension m(n),tb(n+1),ss(10)
    i=1
    do while i<=n
       m(i)=field(i)
       i=i+1
    enddo
    tb(1)='\ Họ và tên'
    tb(2)='\Đơn vị '
    tb(3)='Lương'
    tb(4)='Thưởng'
    tb(5)='Phụ cấp'
    tb(6)='Tạm ứng'
    tb(n+1)='Kết thúc'
    dk=' '
    hd=' '
    bay=0
    do while .t.
       clear
       @10,2 menu tb,n+1 save
       read menu to chon
       if chon=n+1
          clear
          exit
       endif
```

```
** Lệnh if bay#0 để điều khiển việc chọn biểu thức kết
** hợp chỉ tiến hành đối với yêu cầu thứ hai trở đi
if bay#0
   @0,18 to 5,40 double
   @1,20 say 'Chọn kiểu kết hợp:'
   @2,20 say replicate('*',20)
   @3,20 prompt 'VA ' @4,20 prompt 'HOAC '
   menu to cc
   if cc=1
      hd='.and.'
   else
      hd='.or.'
   endif
endif
bay=1
yc=space(50)
clear
@12,20 say 'Chọn yêu cầu: '
@13,10 say tb(chon) get yc
read
** Chú ý: Việc đưa vào các yêu cầu thống nhất như sa
** Nếu bạn muốn tìm những bản ghi có lương lớn hơn
** hoặc bằng 200.000 và nhỏ hơn 400.000 thì viết như
** sau: >=200000.and.<400000
** Nếu cần tìm những bản ghi có thưởng lớn hơn
**  200.000 thì ghi >200000. Còn nếu phụ cấp bằng
**  30000 thì viết  =30000.....
tg=m(chon)
i=1
```

```
y1=alltrim(yc)
so=at('.and.',y1)
if so>0
   ss(1)=subs(y1,1,at('.and.',y1)-1)
   ss(2)=subs(y1,at('.and.',y1)+5)
   ss(2)=alltrim(ss(2))
   i=2
   else
   ss(1)=alltrim(yc)
   i=1
endif
j=1
yeucau=' '
do while j<=i
   if yeucau=' '
      yeucau=tg+ss(j)
      else
      yeucau=yeucau+'.and.'+tg+ss(j)
   endif
   j=j+1
enddo
yeucau='('+yeucau+')'
if dk#' '
   dk=dk+hd+yeucau
   else
   dk=yeucau
endif
** Lệnh dưới không cho phép những trường đã được
** chọn yêu cầu rồi lại được chọn một lần nữa.
```

```
    tb(chon)='\'+tb(chon)
enddo
set filter to &dk
list
set talk on
Rerurn
*******
```

Xin bạn lưu ý: Đối với các trường có dạng dữ liệu kiểu ngày hoàn toàn tương tự như đối với dạng số. Dữ liệu kiểu logic thì đơn giản hơn vì chỉ gồm hai giá trị: .T. (true) và .F. (false). Mong các bạn vui lòng tự viết chương trình cho các kiểu dữ liệu trên và kết nối tất cả lại thành một chương trình thống nhất. FOXPRO cung cấp cho bạn một hàm là TYPE() để kiểm tra trường mà bạn đang chọn yêu cầu có kiểu dữ liệu nào để bạn chọn cách lựa chọn tương ứng.

21. 3. XỬ LÝ THÔNG TIN

Trong mục này các bạn sẽ tìm hiểu cách kết xuất thông tin ra màn hình, máy in, ra tệp các thông tin mà khi thực hiện chương trình mới lựa chọn. Tệp CSDL được lựa chọn đưa ra làm việc là bất kỳ.

Các bạn hãy lập chương trình theo yêu cầu sau:

- Khi thực hiện chương trình máy mới hỏi tên tệp làm việc.

- Sau khi đưa vào tên tệp làm việc, trên màn hình xuất hiện các trường có trong tệp này. Bạn hãy chọn trường cần in ra bằng cách đưa vệt sáng đến trường này và ấn Enter.

Máy sẽ hỏi (ví dụ)

Tên trường: HOTEN

Tên cột khi in ra là:

Bạn hãy đưa tên cột để in ra cho trường HOTEN. Ví dụ bạn gõ vào là:

Họ và tên

- Cứ tiếp tục như vậy bạn chọn các trường khác.

- Sau khi chọn xong hãy đưa vệt sáng đến cột 'Kết thúc' và ấn Enter.

- Tiếp tục, bạn trả lời số dòng tiêu đề cho báo cáo in ra. Nếu bạn muốn có những dòng trống giữa các dòng tiêu đề thì bạn cũng phải tính cả dòng này. Sau đó bạn vào nội dung cho từng tiêu đề.

- Bạn hãy điền ký tự dùng để kẻ báo cáo (ví dụ như ký tự * hoặc dấu - chẳng hạn).

- Và cuối cùng bạn chọn việc đưa kết quả ra màn hình hoặc máy in hay ra tệp văn bản.

Sau đây là nội dung chương trình:

```
**************
set talk off
set safety off
clear
********
public socot,sotieude,kytu,rong,stt,kieuin
kytu=' '
store 0 to socot,sotieude,rong,stt
tep=space(8)
set color to
clear
@10,20 say 'Tệp làm việc: ' get tep
```

```
read
clear
select 1
use &tep
so=fcount()
dimension m(so+1),truong(so),ten(so)
store space(20) to ten
store space(11) to truong i=1
Do while i<=so
   m(i)=fields(i)
   i=i+1
Enddo
m(so+1)='Kết thúc'
tt=iif(so+1>15,15,so+1)
i=1
********** **
** Đoạn chương trình dưới đây để chọn trường nào cần
** và khi in thì tên cột của trường đó là gì
Do while .t.
   clear
   @ 10,30 menu m,tt save
   read menu to chon
   if chon=so+1
      clear
      exit
      else
      truong(i)=m(chon)
      @ 11,20 say 'Tên trường: '+truong(i)
```

```
    @12,20 say 'Tên cột khi in ra là: ' get ten(i)
    read
    i=i+1
    m(chon)='\'+m(chon)
    endif
Enddo
socot=i-1
*********
i=1
do while i<=socot
    ten(i)=alltrim(ten(i))
    i=i+1
enddo
clear
****
** Đoạn chương trình dưới đây để khai báo số tiêu đề
** nội dung cụ thể của từng tiêu đề
@10,20 say 'Số dòng tiêu đề:' get sotieude picture'99'
read
clear
dimension std(sotieude)
store space(60) to std
i=1
do while i<=sotieude
    @i,1 say 'Tiêu đề: '+str(i,2) get std(i)
    read
    std(i)=alltrim(std(i))
    i=i+1
    enddo
```

```
clear
**************
** Khai báo ký tự dùng để kẻ bảng
@ 10,20 say 'Ký tự dùng để kẻ: ' get kytu
read
i=1
rong=0
**********
** Đếm chiều rộng của bảo cáo cần in ra
** Nếu nội dung dài hơn tiêu đề của cột thì tính
** là chiều dài của nội dung. Ngược lại lấy chiều
** dài của tiêu đề do while i<=socot.
if len(ten(i))>fsize(truong(i))
   rong=rong+len(ten(i))+2
   else
   rong=rong+fsize(truong(i))+2
endif
i=i+1
enddo
rong=rong+5
****
clear
@ 7,18 to 12,56 double
@ 8,20 prompt 'Đưa kết quả ra màn hình'
@ 9,20 prompt 'Đưa kết quả ra tệp văn bản'
@ 10,20 prompt 'Đưa kết quả ra máy in'
@ 11,20 prompt 'Kết thúc'
menu to chon
do case
```

```
    case chon=1
        do MANHINH
    case chon=2
        do TEP
    case chon=3
        do MAYIN
    case chon=4
        clear
endcase
***************
Procedure MAYIN
** Chọn các chế độ in và loại giấy để phù hợp với
** kết quả thu được
if rong<=80
    kieuin='A4 - In doc- 10 Ky tu/Inch'
    sodong=chr(27)+"C"+chr(50)
    ma_in=chr(18)+chr(20)+chr(27)+"0"+chr(27)+"P"
    kt=80 endif  if rong>80.and.rong<=95
    kieuin='A4 - In doc- 12 Ky tu/Inch'
    sodong=chr(27)+"C"+chr(50)
    ma_in=chr(18)+chr(20)+chr(27)+"0"+chr(27)+"M"
    kt=95
endif
if rong>95.and.rong<=110
    kieuin='A4 - In ngang- 10 Ký tự/Inch'
    sodong=chr(27)+"C"+chr(30)
    ma_in=chr(18)+chr(20)+chr(27)+"0"+chr(27)+"P"
    kt=110
endif
```

```
if rong>110.and.rong<=130
   kieuin='A4 - In ngang- 12 Ký tự/Inch'
   sodong=chr(27)+"C"+chr(30)
   ma_in=chr(18)+chr(20)+chr(27)+"0"+chr(27)+"P"
   kt=130
endif
clear
@10,20 say 'Hãy chọn: '+kieuin
wait 'Hãy chuẩn bị máy in-Xong ấn phím bất kỳ'
set device to printer
@0,0 say sodong
@0,0 say ma_in
do chitiet
eject
set device to screen
return
*************
Procedure TEP
clear
tentep=space(12)
@10,20 say 'Tên tệp cần chứa kết quả:  get tentep
read
set device to file &tentep
kt=rong
do CHITIET
set device to screen
hoi=' '
clear
@12,20 say 'Có xem kết quả không C/K' get hoi;
```

```
                                        function"!" valid hoi$'CK'
    read
    If hoi='C'
       modify command &tentep
    endif
    clear
    return
    **************
    Procedure manhinh
    clear
    if rong>=80
       @10.20 say 'Không nên đưa kết quả ra màn hình'
       @11.20 say 'vì dữ liệu quá dài'
       hoi=' '
       @12.20 say 'Có đưa kết quả ra tệp không C/K':
                get hoi function"!" valid hoi$'CK'
       If hoi='C'
          clear
          do TEP
       endif
       else
       kt=80
       do CHITIET
    endif
Return
************
Procedure CHITIET
** Thủ tục này in chi tiết kết quả thu được.
** Chế độ kết xuất ra máy in, màn hình hay tệp do
```

```
** các lệnh chọn phía trên quyết định.
** Các câu lệnh dưới để các dòng tiêu đề ra giữa màn hình
** hay trang in.
i=1
do while i<=sotieude
   @i,int((kt-len(std(i)))/2) say std(i)
      i=i+1
enddo
h=i+1
** Câu lệnh dưới tính lề trái để kết quả đưa ra ở giữa
** màn hình hay trang in.
le=int((kt-rong)/2)
@h,0+le say replicate(kytu,rong)
h=h+1
@h,0+le say 'STT'
j=5
i=1
** Các câu lệnh dưới đưa tên các cột.
** Biến j để tính xem sau mỗi cột thì tăng thêm bao
** nhiêu cột.
do while i<=socot
   @h,j+le say ten(i)
   if len(ten(i))>fsize(truong(i))
      j=j+len(ten(i))+2
      else
      j=j+fsize(truong(i))+2
   endif
   i=i+1
   enddo
```

```
h=h+1
@h,0+le say replicate(kytu,rong)
select 1
go 1
st=1
h=h+1
** Các câu lệnh dưới để đưa dữ liệu cụ thể
do while .not.eof()
   @h,0+le say str(st,3)
   i=1
   j=5
   do while i<=socot
      tg=truong(i)
      @h,j+le say &tg
      if len(ten(i))>fsize(truong(i))
         j=j+len(ten(i))+2
         else
         p=0
         j=j+fsize(truong(i))+2
      endif
      i=i+1
   enddo
   h=h+1
   st=st+1
   skip
   enddo
   @h,0+le say replicate(kytu,rong)
   return
   ***********
```

Phụ lục

BẢNG THÔNG BÁO LỖI

(SẮP XẾP THEO TÊN TIẾNG ANH)

STT	Mã (lỗi số)	Tên lỗi (tiếng Anh)	Ý nghĩa
1	24	ALIAS name already in use	Bí danh đã dùng rồi. Định mở một tệp CSDL với một chữ cái từ A đến J hoặc bí danh đã dùng rồi.
2	13	ALIAS not found	Không tìm thấy bí danh. Định chọn (SELECT) một tệp CSDL ngoài vùng từ A đến J hoặc bí danh chưa có.
3	1153	Attempt to move file to different device	Không được phép đổi tên tệp và chuyển sang ổ đĩa khác cùng một lúc.
4	1907	Bad drive specifier	Chỉ định sai ổ đĩa. Dùng tên ổ đĩa sai trong lệnh DIR.
5	1908	Bad width or decimal place argument	Khai sai chiều dài hoặc số thập phân. Hoặc ghi sai độ dài hoặc sai tham biến trong hàm STR().
6	38	Beginning of file encountered	Đã tới đầu tệp. Con trỏ bị đặt trước bản ghi đầu tiên.

7	62	Beyond String	Hết chuỗi. Đã truy nhập đến ký tự sau byte cuối cùng của biến nhớ dạng ký tự hay của trường trong tệp CSDL.
8	1101	Cannot open file	Không mở được tệp.
9	42	CONTINUE without LOCATE	CONTINUE không có LOCATE. Sử dụng một lệnh CONTINUE mà trước đó không có lệnh LOCATE.
10	9	Data type mismatch	Kiểu dữ liệu không phù hợp. Lỗi này thường xảy ra khi thay đổi giá trị một trường của tệp CSDL (REPLACE ...) bằng một <bthức> không phù hợp với dữ liệu của trường.
11	4	End file encountened	Hết tệp. Con trỏ chỉ sau bản ghi cuối cùng (kết thúc tệp) trong tệp.
12	1214	Endtext without text	Thiếu TEXT. Đã sử dụng ENDTEXT mà trước đó không sử dụng TEXT.
13	7	File already exists	Đã có tệp này. Lỗi thường gặp khi đổi tên tệp thành tên mới đã có trong thư mục hoặc khi dùng lệnh SORT...
14	3	File is in use	Tệp đang được dùng.

15	1127	FOR/WHILE need Logicallogic expressions.	FOR/WHILE Cần biểu thức Logic. Lỗi xảy ra khi biểu thức sau FOR hoặc WHILE không phải là biểu thức logic.
16	1211	IF/ELSE/ ENDIF mismatch	IF/ELSE/ENDIF không phù hợp. Thiếu ELSE hay ENDIF ứng với IF hoặc có ENDIF. ELSE mà không có IF.
17	1208	Illegal macro substution	Thay thế macro không hợp lệ. Sử dụng biến nhớ không phải là biến chuỗi cho macro.
18	1305	Illegal value	Giá trị không hợp lệ.
19	43	Insufficient memory	Thiếu bộ nhớ. Không đu bộ nhớ để nạp chương trình.
20	31	Invalid subscript reference	Lỗi chỉ số dưới của mảng. Sử chỉ số mảng định nghĩa bằng lệnh DIMENSION nằm ngoài chỉ số của chỉ số mảng.
21	58	LOG domain error	Lỗi về miền xác định của hàm LOG. Đối số của hàm LOG phải lớn hơn 0.
22	41	MEMO file missing	Thiếu tệp MEMO. Lỗi xảy ra khi dùng một tệp CSDL mà tệp trường MEMO liên

			quan đến nó đã bị xóa đi.
23	55	Memory variable is invalid	Tệp biến nhớ không có giá trị. Lỗi xảy ra khi dùng lệnh RESTORE FROM với một tệp không phải là tệp biến nhớ.
24	1213	Mismatched case structure	Cấu trúc case không phù hợp. Các lệnh Case, Endcase không có lệnh DO CASE tương ứng.
25	1340	Missing (	Thiếu (. Thiếu dấu mở ngoặc ở tên hàm.
26	1300	Missing)	Thiếu). Thiếu dấu đóng ngoặc ở tên một hàm...
27	1306	Missing ,	Thiếu dấu ,
28	1231	Missing operand	Thiếu toán hạng. Dùng một toán tử có hai toán hạng nhưng thiếu toán hạng thứ hai. Ví dụ: ?(12+19)/
29	1145	Must be a character, numeric key fields	Khoá phải là kiểu ký tự, ngày tháng hoặc số.
30	1232	No PARAMETER found	Thiếu lệnh PARAMETER. Khi statement chạy một chương trình bằng lệnh DO với một danh sách tham số

			nhưng chương trình được gọi không bắt đầu bởi lệnh PARAMETER với các tham số tương ứng.
31	27	Not numeric expresion	Không phải là biểu thức số. Lỗi thường xảy ra khi thực hiện các lệnh SUM, AVERAGE.. trên các trường không phải dạng số.
32	101	Not suspended	Không có lệnh dừng. Dùng lệnh RESUME nhưng không có lệnh SUSPEND.
33	107	Opertor/ Operand type Mismathch	Toán tử và toán hạng không phù hợp về kiểu. Lỗi thường xảy ra khi thực hiện các phép tính các giá trị khác kiểu dữ liệu hoặc thực hiện một phép tính số học đối với hai giá trị logic.
34	56	Out of disk	Quá dung lượng đĩa.
35	1217	Picture ERROR in GET statement	Lỗi về picture trong lệnh Thường là điều kiện picture trong lệnh... GET chứa một picture không có giá trị về định dạng.
36	30	Position is off the screen	Vị trí vượt quá màn hình. Số dòng hay số cột trong câu lệnh vượt quá kích thước của

			màn hình (hoặc cửa sổ).
37	125	Printer not ready	Không liên lạc được với máy in (máy in chưa sẵn sàng làm việc).
38	1202	Program too large	Chương trình quá lớn.
39	5	Record is out of range	Bản ghi quá giới hạn. Lỗi xảy khi có lệnh làm việc với một bản ghi mà số thứ tự vật lý của nó lớn hơn tổng số bản ghi có trong tệp.
40	61	SQRT domain error	Lỗi về miền xác định căn bậc hai. Hàm SQRT() chỉ chấp nhận các đối số không âm.
41	1234	Subscripts out of bounds	Các chỉ số của biến vượt quá giới hạn. Lỗi xảy ra khi chỉ định một phần tử mảng có các chỉ số vượt quá các giới hạn xác định trong câu lệnh DIMENSION.
42	10	Syntax error	Sai cú pháp. Lệnh viết không đúng cú pháp. Lỗi cú pháp có thể là viết sai lỗi chính tả hoặc một câu lệnh không hợp ngữ cảnh...

MỤC LỤC

Chương 2

TẠO TỆP CSDL, XEM THÔNG TIN, CẤU TRÚC CSDL 18

Chương 3

LÀM VIỆC VỚI CÁC BẢN GHI TỆP CSDL 42

Chương 4

SẮP XẾP DỮ LIỆU 54

Chương 9

CẤU TRÚC ĐIỀU KIỆN, CẤU TRÚC LẶP VÀ CÁC CHƯƠNG TRÌNH ỨNG DỤNG 104

Chương 12

CÁC LỆNH NHẬP DỮ LIỆU TỪ BÀN PHÍM 137

Chương 19

XỬ LÝ DỮ LIỆU KIỂU MEMO 246

Chương 20

LÀM VIỆC VỚI NHIỀU TỆP CSDL 256

Giá: 25.000đ