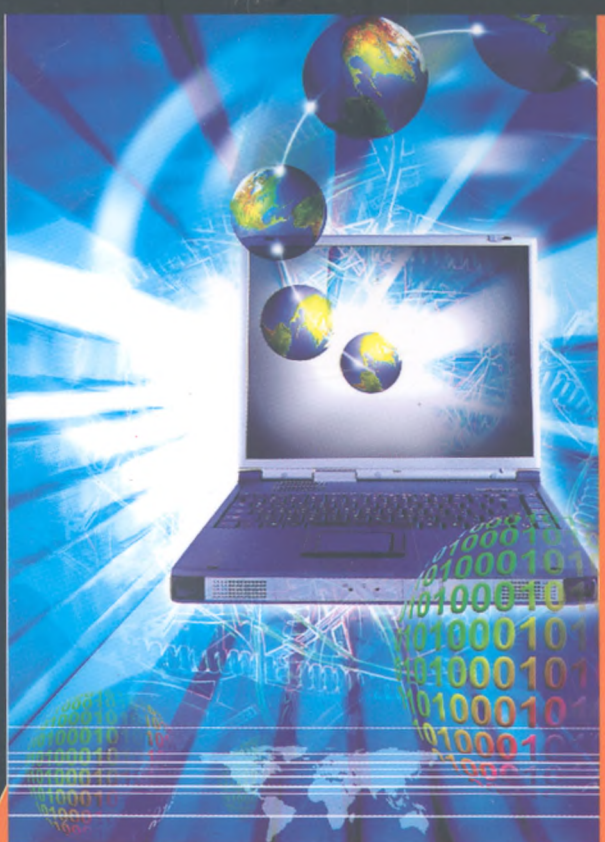
SỞ GIÁO DỤC VÀ ĐÀO TẠO HÀ NỘI

GIÁO TRÌNH

Tin học đại cương

DÙNG TRONG CÁC TRƯỜNG TRUNG HỌC CHUYÊN NGHIỆP



NHÀ XUẤT BẢN HÀ NỘ

SỞ GIÁO DỤC VÀ ĐÀO TẠO HÀ NỘI

NGUYỄN GIA PHÚC (*Chủ biên*)

GIÁO TRÌNH

TIN HỌC ĐẠI CƯƠNG

(Dùng trong các trường THCN)

NHÀ XUẤT BẢN HÀ NỘI - 2005

Lời giới thiệu

Nước ta đang bước vào thời kỳ công nghiệp hóa, hiện đại hóa nhằm đưa Việt Nam trở thành nước công nghiệp văn minh, hiện đại.

Trong sự nghiệp cách mạng to lớn đó, công tác đào tạo nhân lực luôn giữ vai trò quan trọng. Báo cáo Chính trị của Ban Chấp hành Trung ương Đảng Cộng sản Việt Nam tại Đại hội Đảng toàn quốc lần thứ IX đã chỉ rõ: "Phát triển giáo dục và đào tạo là một trong những động lực quan trọng thúc đẩy sự nghiệp *công nghiệp hóa, hiện đại hóa*, là điều kiện để phát triển nguồn lực con người - yếu tố cơ bản để phát triển xã hội, tăng trưởng kinh tế nhanh và bền vững".

Quán triệt chủ trương, Nghị quyết của Đảng và Nhà nước và nhận thức đúng đắn về tầm quan trọng của chương trình, giáo trình đối với việc nâng cao chất lượng đào tạo, theo đề nghị của Sở Giáo dục và Đào tạo Hà Nội, ngày 23/9/2003, Ủy ban nhân dân thành phố Hà Nội đã ra Quyết định số 5620/QĐ-UB cho phép Sở Giáo dục và Đào tạo thực hiện đề án biên soạn chương trình, giáo trình trong các trường Trung học chuyên nghiệp (THCN) Hà Nội. Quyết định này thể hiện sự quan tâm sâu sắc của Thành ủy, UBND thành phố trong việc nâng cao chất lượng đào tạo và phát triển nguồn nhân lực Thủ đô.

Trên cơ sở chương trình khung của Bộ Giáo dục và Đào tạo ban hành và những kinh nghiệm rút ra từ thực tế đào tạo, Sở Giáo dục và Đào tạo đã chỉ đạo các trường THCN tổ chức biên soạn chương trình, giáo trình một cách khoa học, hệ

thống và cập nhật những kiến thức thực tiễn phù hợp với đối tượng học sinh THCN Hà Nội.

Bộ giáo trình này là tài liệu giảng dạy và học tập trong các trường THCN ở Hà Nội, đồng thời là tài liệu tham khảo hữu ích cho các trường có đào tạo các ngành kỹ thuật - nghiệp vụ và đông đảo bạn đọc quan tâm đến vấn đề hướng nghiệp, dạy nghề.

Việc tổ chức biên soạn bộ chương trình, giáo trình này là một trong nhiều hoạt động thiết thực của ngành giáo dục và đào tạo Thủ đô để kỷ niệm "50 năm giải phóng Thủ đô", "50 năm thành lập ngành" và hướng tới kỷ niệm "1000 năm Thăng Long - Hà Nội".

Sở Giáo dục và Đào tạo Hà Nội chân thành cảm ơn Thành ủy, UBND, các sở, ban, ngành của Thành phố, Vụ Giáo dục chuyên nghiệp Bộ Giáo dục và Đào tạo, các nhà khoa học, các chuyên gia đầu ngành, các giảng viên, các nhà quản lý, các nhà doanh nghiệp đã tạo điều kiện giúp đỡ, đóng góp ý kiến, tham gia Hội đồng phản biện, Hội đồng thẩm định và Hội đồng nghiệm thu các chương trình, giáo trình.

Đây là lần đầu tiên Sở Giáo dục và Đào tạo Hà Nội tổ chức biên soạn chương trình, giáo trình. Dù đã hết sức cố gắng nhưng chắc chắn không tránh khỏi thiếu sót, bất cập. Chúng tôi mong nhận được những ý kiến đóng góp của bạn đọc để từng bước hoàn thiện bộ giáo trình trong các lần tái bản sau.

GIÁM ĐỐC SỞ GIÁO DỤC VÀ ĐÀO TẠO

Lời nói đầu

Giáo trình Tin học đại cương nhằm mục đích trang bị những kiến thức cơ bản nhất về tin học cho học sinh hệ trung học chuyên nghiệp cụ thể là học sinh các ngành cơ khí. Giáo trình gồm 2 phần: Đại cương về máy tính và tin học, hệ điều hành máy, phần 2 là kỹ thuật lập trình với ngôn ngữ lập trình cơ bản là Pascal. Giáo trình này còn hướng đến một mục tiêu nữa là giúp cho người đọc có thể tự học được các kỹ năng lập trình nhờ cách trình bày dễ hiểu, có nhiều ví dụ minh hoạ, sau mỗi phần đều có nhiều bài tập được sắp xếp từ dễ đến khó phù hợp với trình độ học sinh trung học chuyên nghiệp. Trên cơ sở đó học sinh có thể áp dụng lập các chương trình tính toán thông dụng của ngành chuyên môn mình học.

Giáo trình được xây dựng từ những bài giảng trong các năm qua của nhóm tác giả và tham khảo từ một số cuốn sách thông dụng, do vậy nó vừa là một cuốn giáo trình đồng thời cũng có thể là tài liệu tra cứu những ý tưởng cốt lõi nhất về kỹ thuật lập trình.

Tuy nhiên cuốn giáo trình không tránh khỏi những sai sót trong quá trình biên soạn, vì vậy rất mong được sự góp ý của các bạn đồng nghiệp cũng như của bạn đọc.

Hy vọng rằng, cuốn giáo trình sẽ là một tài liệu học tập cho học sinh các ngành cơ khí hệ trung học chuyên nghiệp nhưng đồng thời cũng là tài liệu tham khảo cho giáo viên học sinh các ngành khác.

Xin chân thành cám ơn các chuyên gia cũng như các bạn đồng nghiệp đã có những góp ý quý báu về nội dung của cuốn giáo trình và giúp đỡ chúng tôi hoàn thành chương trình này.

Nhóm tác giả

Phần I

HỆ ĐIỀU HÀNH MS-DOS

Chương 1

HỆ ĐẾM, BIỂU DIỄN THÔNG TIN TRONG MÁY TÍNH ĐIỆN TỬ

I. MỘT SỐ KHÁI NIỆM CƠ BẢN

Thông tin, lưu trữ và truyền tin

Thông tin là sự phản ánh sự vật, sự việc, hiện tượng của thế giới khách quan và các hoạt động của con người trong đời sống xã hội. Điều cơ bản là con người thông qua việc cảm nhận thông tin làm tăng hiểu biết cho mình và tiến hành những hoạt động có ích cho cộng đồng.

Thông tin được **lưu giữ** trên nhiều dạng vật liệu khác nhau như được khắc trên đá, được ghi lại trên giấy, trên bìa, trên băng từ, đĩa từ. . .

Việc lưu giữ và truyền tin chỉ có giá trị khi quá trình đó đảm bảo chính xác nội dung của nó. Để thuận tiện người ta phải biến đổi và khôi phục thông tin theo quy ước sao cho đảm bảo: chính xác, kinh tế, thời gian, không gian, mà thực chất là quá trình xử lý thông tin: mã hoá thông tin, cất giữ, truyền tin và giải mã thông tin.

Môi trường vận động thông tin là môi trường **truyền tin**, nó bao gồm các kênh liên lạc tự nhiên hoặc nhân tạo như sóng âm, tia sáng, dây dẫn, sóng âm thanh, sóng hình... Kênh liên lạc thường nối các thiết bị của máy móc với nhau hay nối với con người.

Con người có hình thức liên lạc tự nhiên và cao cấp là tiếng nói, từ đó nghĩ ra chữ viết. Ngày nay nhiều công cụ phổ biến thông tin đã xuất hiện: bút viết, máy in, điện tín, điện thoại, phát thanh, truyền hình, phim ảnh v.v.

Máy tính điện tử là công cụ hiện đại cho phép tự động hoá việc xử lý và truyền thông tin đảm bảo chính xác và nhanh gọn.

II. BIỂU DIỄN THÔNG TIN TRONG MÁY TÍNH ĐIỆN TỬ (MTĐT)

Máy tính điện tử biểu diễn thông tin trên cơ sở ghép nối các linh kiện, các mạch điện tử thực hiện hai trạng thái vật lý ký hiệu là 0 và 1.

Để mô tả trạng thái vật lý tương ứng với hai ký hiệu 0 và 1, ví dụ: Bóng điện tử có thể ở một trong hai trạng thái là sáng hoặc tắt; Một công tắc điện có thể ở một trong hai trạng thái là bật hoặc tắt; Một nguồn điện có thể có điện thế cao hay thấp v.v.

Trong máy tính là các mạch điện tử, linh kiện điện tử thể hiện hai trạng thái đó và được quy ước biểu diễn như sau:

Nếu ở trạng thái đóng:  biểu diễn ký hiệu số 1

Nếu ở trạng thái ngắt:  biểu diễn ký hiệu số 0

Thông tin biểu diễn trong máy tính có dạng dữ liệu, bao gồm những con số, chữ cái, ký hiệu được chọn lọc và tổ chức theo quy cách xác định để thuận tiện cho việc xử lý tự động. Dữ liệu được thể hiện bằng cách ghép nối theo từng nhóm các linh kiện điện tử và ở từng thời điểm, từng nhóm các linh kiện thể hiện trạng thái tín hiệu điện ký hiệu 0,1. MTĐT có thể biểu diễn được thông tin đa dạng của cuộc sống trên cơ sở mã hoá thành các ký hiệu 0 và 1 theo quy luật nào đó gọi là mã nhị phân.

Trong thực tế người ta hay ghép các linh kiện thành từng nhóm, các bộ xử lý của máy tính thường ghép theo nhóm 8, 16, 32 linh kiện để biểu diễn thông tin.

Việc dùng từ "linh kiện điện tử" trên nhằm mục đích mô tả, thực chất linh kiện điện tử là các vi mạch bán dẫn, các bộ vi xử lý thực hiện hai tín hiệu đó cực kỳ nhanh.

III. HỆ ĐẾM NHỊ PHÂN

Trong cuộc sống hàng ngày ta thường dùng số đếm thập phân (hệ cơ số 10), tức là dùng mười chữ số: 0,1,2,3,4,5,6,7,8,9 để biểu diễn một số nào đó. Trong máy tính, số và các câu lệnh đều biểu diễn bằng những dãy số nhị phân (cơ số 2) tức là để biểu diễn số nào đó theo cơ số này, chỉ dùng hai chữ số: 0 và 1 ghép lại theo trật tự.

Ví dụ: Số nhị phân: $1011_2 = 11_{10}$; $1102_2 = 13_{10}$.

IV. CHUYỂN ĐỔI HỆ ĐẾM

1. Chuyển dạng biểu diễn từ nhị phân sang thập phân:

Cho một số X dưới dạng nhị phân:

$$X= a_n a_{n-1} a_{n-2} ... a_2 a_1 a_0 \ (a_i=0 \text{ hoặc } a_i=1)$$

Muốn tìm dạng biểu diễn thập phân của X ta tính đa thức:

$$X= a_n*2^n + a_{n-1}*2^{n-1} + ... + a_1*2^1 + a_0*2^0$$

Ví dụ: Dạng thập phân của 1101_2 là:

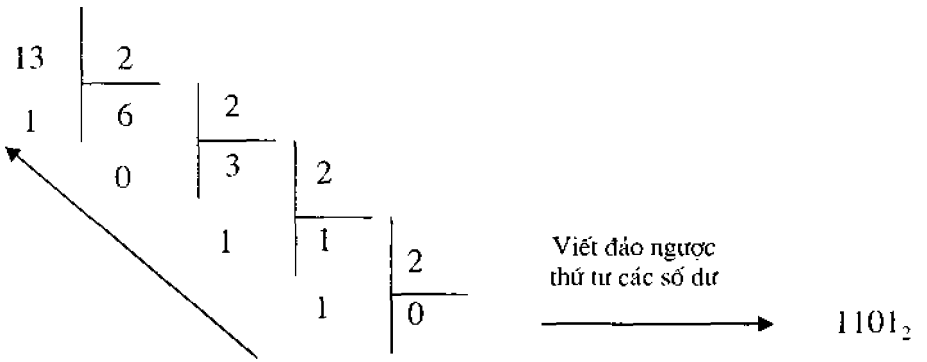
$$182^3 + 1*2^2 + 0*2^1 + 1*2^0 = 8+4+1=13$$

Vậy $1101_2 = 13$

2. Chuyển dạng biểu diễn thập phân sang nhị phân

Cho số X dưới dạng thập phân, muốn tìm dạng biểu diễn nhị phân của X ta chia X liên tiếp cho 2 và lấy số dư (chỉ gồm các số 0 và 1) theo chiều ngược lại.

Ví dụ: Cho X=13, hãy chuyển X sang biểu diễn ở dạng nhị phân.



V. BIỂU DIỄN CÁC KÝ TỰ

Hệ thống máy tính thường xuyên trao đổi số liệu, đưa ra màn hình dòng thông báo, thể hiện câu lệnh v.v. ở dạng các chữ cái và các ký hiệu thông thường, nhưng máy chỉ xử lý các tín hiệu mã hoá dưới dạng nhị phân (binary digit).

Ví dụ: Chữ cái A được mã thành số nhị phân 0100 0001

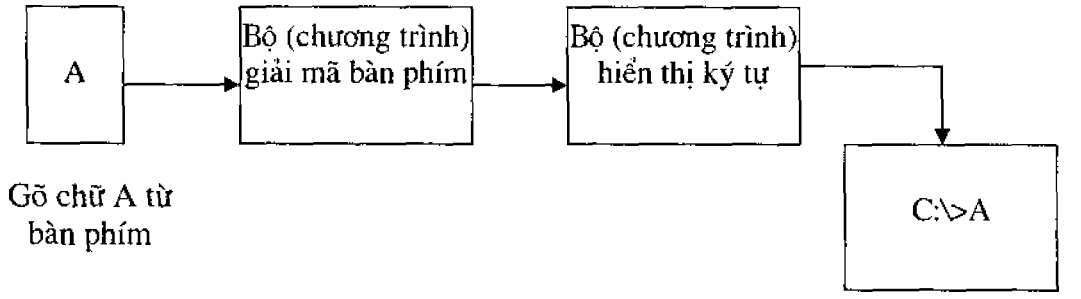
Dấu * được mã thành số nhị phân 0010 1010

Vì vậy cần chọn bảng mã nhị phân để biểu diễn các chữ cái, chữ số, dấu ngắt câu, các câu lệnh để trao đổi giữa các thiết bị trong máy và giữa các máy tính với nhau.

Bảng mã ASCII (American Standard Code for Information Interchange) được chọn làm bảng mã chuẩn quốc tế trong các máy tin học để thực hiện trao đổi thông tin. Trong đó mỗi ký tự được biểu diễn bởi 1 nhóm cố định 8 chữ số 0 hoặc 1 tức là tương đương với một số nhị phân. Tổng số tất cả các ký hiệu trong ASCII là $256=2^8$, trong đó 128 ký hiệu chuẩn và 128 ký hiệu mở rộng được dùng cho các ký hiệu đặc biệt cho ngôn ngữ của các quốc gia riêng biệt. Bảng mã chuẩn bao gồm các ký hiệu:

- Các chữ cái la tinh: 26 chữ hoa (A, B, C,..., Z) và 26 chữ thường (a, b, c,. . ., z)\
- Các chữ số tự nhiên: 0, 1, 2, 3, 4, 5, 6, 7 , 8, 9
- Các ký hiệu toán học thông dụng: +, -, *, /, =, <, >
- Các ký hiệu thường dùng: . , ; ( ) " ' [ ] ? % $ # @ ! ..
- Các ký hiệu điều khiển:'xuống dòng', 'thực hiện chương trình"': ENTER 'đánh dấu đầu bản tin', ' đánh dấu cuối bản tin' v.v.

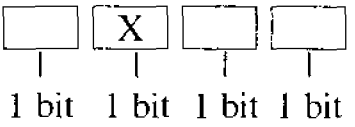
Bảng mã ASCII có 128 ký tự chuẩn (tham khảo bảng mã ASCIII ở phần phụ lục). Giả sử muốn hiển thị chữ A trên màn hình ta phải tác động vào chương trình giải mã và chương trình thể hiện các ký tự trên màn hình, quá trình đó diễn ra rất nhanh, ta có thể hình dung theo mô tả khối dưới đây:



VI. TỔ CHỨC BỘ NHỚ VÀ ĐƠN VỊ ĐO THÔNG TIN

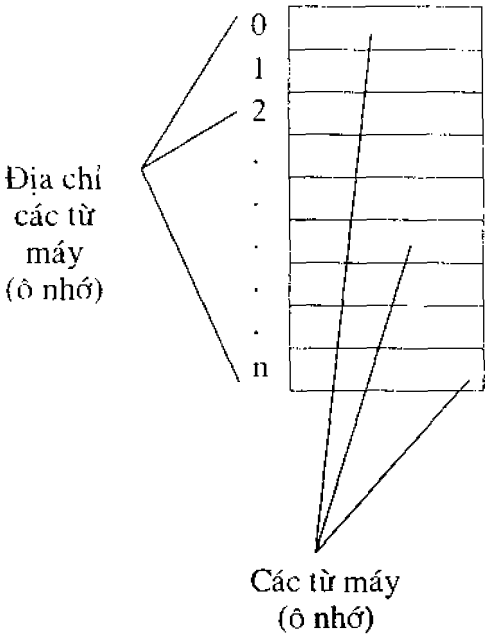
- Bit (Binary digit)

Bộ nhớ của MTĐT được tạo từ vật liệu có thể ở một trong hai trạng thái, ký hiệu tương ứng là 0 và 1. Mỗi vị trí chất liệu như thế là đơn vị nhỏ nhất gọi là bit (binary digit).



- Từ máy (Word memory)

Để tiện việc biểu diễn và cất giữ thông tin, người ta ghép 2 bit, 4 bit, 8 bit, 16 bit, 32 bit lại với nhau thành một đơn vị nhớ gọi là ô nhớ (hay còn gọi là từ máy).



- Địa chỉ

Các từ máy (ô nhớ) được đặt liên tiếp nhau và đánh số thứ tự từ: 0,1,...,n gọi là địa chỉ của các từ máy.

- Bai (Byte)

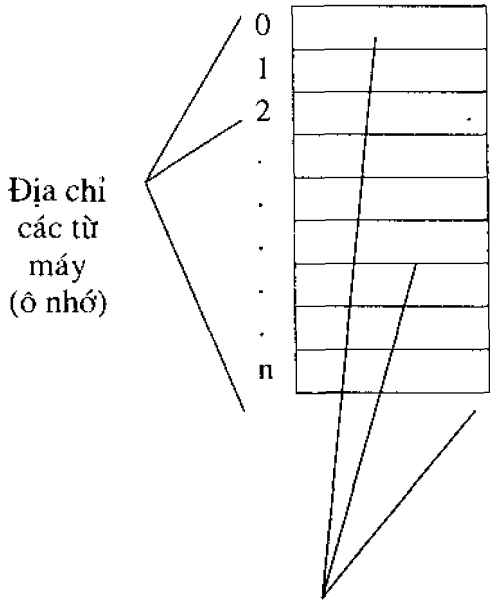
Bai là đơn vị nhớ chuẩn, bằng cách ghép liên tiếp 8 bit liền nhau tạo thành 1 byte nhớ, đọc là bai, viết tắt: B

Ngoài ra, các đơn vị dẫn xuất để đo thông tin là:

1 Kilobai (viết tắt: 1KB) = 1024B

1 Mêgabai (viết tắt: 1MB) = 1024KB

1 Gigabai (viết tắt: 1GB) = 1024MB



- Dung lượng (Memory capacity)

Khả năng chứa thông tin (trong thời gian xử lý của CPU) của bộ nhớ trong là dung lượng của bộ nhớ. Các bộ nhớ trong của MTĐT thường có dung lượng: 640KB, 1MB, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB...

Bộ nhớ trong chia thành các ô nhớ, mỗi ô nhớ đều đánh số thứ tự gọi là địa chỉ ô nhớ, nội dung thông tin chứa trong các ô nhớ.

Có thể diễn tả các ô nhớ và địa chỉ của bộ nhớ trong như hình trên.

- Cách đọc tệp thông tin và các thông báo

Thông tin được lưu trên các đĩa từ, băng từ (bộ nhớ ngoài) được gọi là dữ liệu. Dữ liệu được tổ chức thành các tệp (file) có tên tệp và độ lớn mà nó chiếm giữ không gian nhớ trên đĩa hoặc băng từ. Độ lớn của tệp được đo và thể hiện bằng bytes. Để thuận lợi và đơn giản hoá cách gọi ta thường gọi vắn tắt độ lớn các tệp, các thông báo trên màn hình bằng đơn vị KB, MB. Cứ 3 chữ số tính từ bên phải sang trái để gọi theo độ lớn của tệp:

Ví dụ: COMMAND.COM 54645B đọc là tệp COMMAND chiếm hơn 54 KB trên đĩa từ, dòng chữ 35.475,154 bytes free trên màn hình, đọc là "còn khoảng 35MB trống để chứa thông tin trên đĩa".

Dựa vào đơn vị đo thông tin ta hình dung sức chứa thông tin trên vật mang như đĩa từ chẳng hạn: Đĩa từ 1,2 MB chứa lượng thông tin dạng các ký hiệu văn bản chữ, chữ số, mỗi ký hiệu chiếm 1 byte nhớ, được tính ra: 1,2 x 1024 (KByte) x 1024 (byte) = 1258171 ký hiệu. Nếu một trang văn bản có 80 cột chữ x 80 dòng thì đĩa từ 1,2MB chứa xấp xỉ 800-1000 trang dữ liệu kiểu văn bản.

Câu hỏi ôn tập

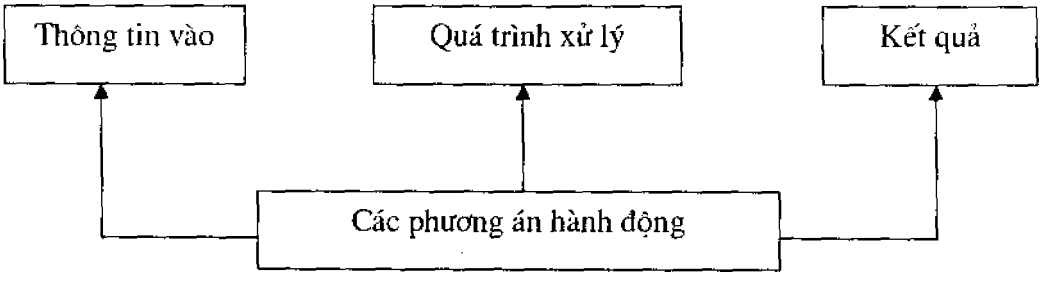
1. Trong MTĐT muốn biểu diễn thông tin cơ sở (đơn vị nhỏ nhất biểu diễn thông tin) người ta mã hoá dưới dạng nào?
2. Nêu các cách chuyển dạng biểu diễn số từ thập phân sang nhị phân và ngược lại. Nêu ví dụ cụ thể?
3. Nêu công dụng chính của bảng mã ASCII?
4. Đơn vị cơ sở biểu diễn thông tin là gì?
5. Nêu tên và độ lớn các đơn vị đo thông tin?
6. Phân biệt địa chỉ ô nhớ và nội dung ô nhớ thông tin?
7. Cách đọc và viết của các đơn vị đo thông tin?

Chương 2

KIẾN TRÚC MÁY TÍNH

I. XỬ LÝ THÔNG TIN TRONG MÁY TÍNH

Trong cuộc sống, bất cứ hoạt động nào của con người cũng đều hướng tới mục đích cụ thể. Để đạt được mục đích đó, con người cần nắm được điều kiện và tác động có thể ảnh hưởng tới hoạt động này. Từ đó phân tích và lựa chọn một chương trình hành động để đạt được hiệu quả công việc. Việc phân tích và lựa chọn phương án hành động chính là quá trình xử lý thông tin, có thể mô tả một cách tổng quát dưới đây:



Việc xử lý thông tin trong MTĐT cũng theo nguyên tắc như vậy: người sử dụng phải nhập dữ liệu vào máy (đưa thông tin vào) qua thiết bị là bàn phím hoặc các thiết bị nhập khác như máy scaner... và ra lệnh cho máy thực hiện các thao tác một cách tự động, tuần tự theo các quy tắc đã chỉ dẫn - các câu lệnh của chương trình. Mỗi thao tác là một đơn nguyên công việc, tương ứng với câu lệnh của chương trình và cuối cùng cho kết quả ở các thiết bị xuất thông tin.

Tóm lại: MTĐT hoạt động theo nguyên tắc: "Tự động thực hiện các thao tác theo một chương trình hành động đã vạch sẵn từ trước".

II. KIẾN TRÚC MÁY TÍNH

1. Bộ xử lý trung tâm (CPU: Center Proccessing Unit)

Bộ xử lý trung tâm là bộ óc của máy tính, ở đó diễn ra việc xử lý thông tin và điều khiển toàn bộ hoạt động của MTĐT.

Bộ xử lý trung tâm bao gồm 2 bộ phận: Bộ số học và logic (gọi tắt là ALU) và Bộ điều khiển.

Đặc trưng của nó thể hiện ở 2 mặt:

- Tốc độ xử lý: là các phép xử lý thực hiện được trong 1 giây. Các máy vi tính ngày nay có tốc độ xử lý lên đến từ hàng trăm triệu đến hàng tỉ phép xử lý trong một giây.

- Lượng thông tin được xử lý đồng thời tính theo đơn vị bit. Các bộ vi xử lý thường có các dạng xử lý 8 bit, 16bit, 32bit, 64bit...

Cấu tạo vật lý của bộ xử lý trung tâm rất gọn nhẹ: chỉ nhỏ bằng nửa bao diêm.

2. Bộ nhớ trong (ROM, RAM)

- Bộ nhớ ROM (Read Only Memory).

ROM là bộ nhớ tĩnh hay còn gọi là bộ nhớ chỉ đọc. Từ khi chế tạo, người ta đã nạp cố định chương trình để khởi tạo máy vào trong ROM, trong thời gian chạy máy, bộ nhớ ROM không thay đổi kể cả khi mất điện. Khi khởi tạo máy, chương trình từ ROM được đọc ra, không thể ghi thông tin vào ROM được.

- Bộ nhớ RAM (Random Access Memory)

RAM là bộ nhớ động dùng để chứa dữ liệu trong quá trình xử lý thông tin trên máy. Bộ xử lý trung tâm thường xuyên lấy dữ liệu từ bộ nhớ trong để xử lý rồi lại gửi thông tin vào đó.

Đặc trưng của bộ nhớ trong là khả năng chứa dữ liệu và tốc độ truy nhập thông tin. Bộ nhớ trong được tổ chức thành n byte nhớ và đánh số thứ tự từ 0 đến n gọi là địa chỉ của ô nhớ.

Ngoài ra, CPU còn có các ô nhớ đặc biệt để ghi nhận các câu lệnh đang thực hiện, lưu trữ các toán hạng và kết quả trung gian: chúng được gọi là các

thanh ghi, trong đó có các thanh ghi tổng để lưu giữ tạm thời các toán hạng và kết quả trung gian.

3. Bộ số học và logic (ALU)

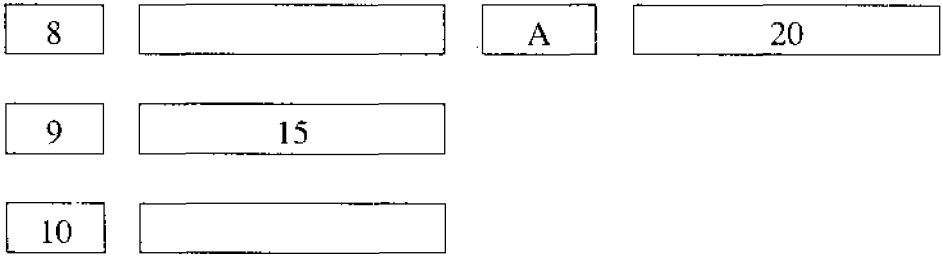
Là thành phần của CPU, nó có nhiệm vụ thực hiện các phép toán số học (cộng, trừ, nhân, chia) và các phép toán logic, dưới đây mô tả việc thực hiện câu lệnh cộng một địa chỉ:

Câu lệnh cộng có dạng:

CỘNG N

Câu lệnh này sẽ lấy số đang có ở thanh ghi tổng A cộng với số có trong ô nhớ thứ N của bộ nhớ. Kết quả thu được sẽ đặt ở thanh ghi tổng A.

Giả sử ô nhớ địa chỉ 9 trong bộ nhớ chứa số 15, thanh ghi tổng A hiện đang chứa số như ở hình dưới:

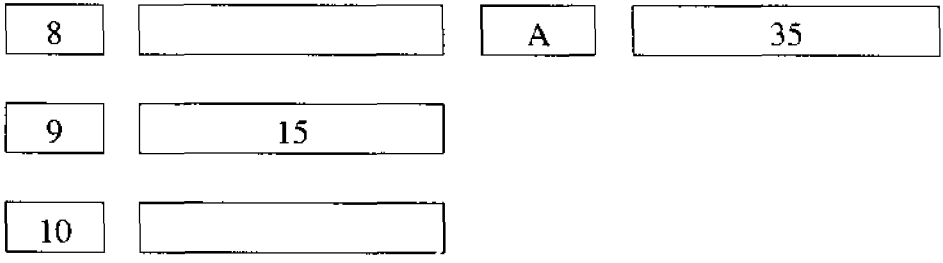


Hiện trạng của bộ nhớ và thanh ghi tổng A
(trước khi thực hiện câu lệnh)

Sau khi ALU thực hiện câu lệnh:

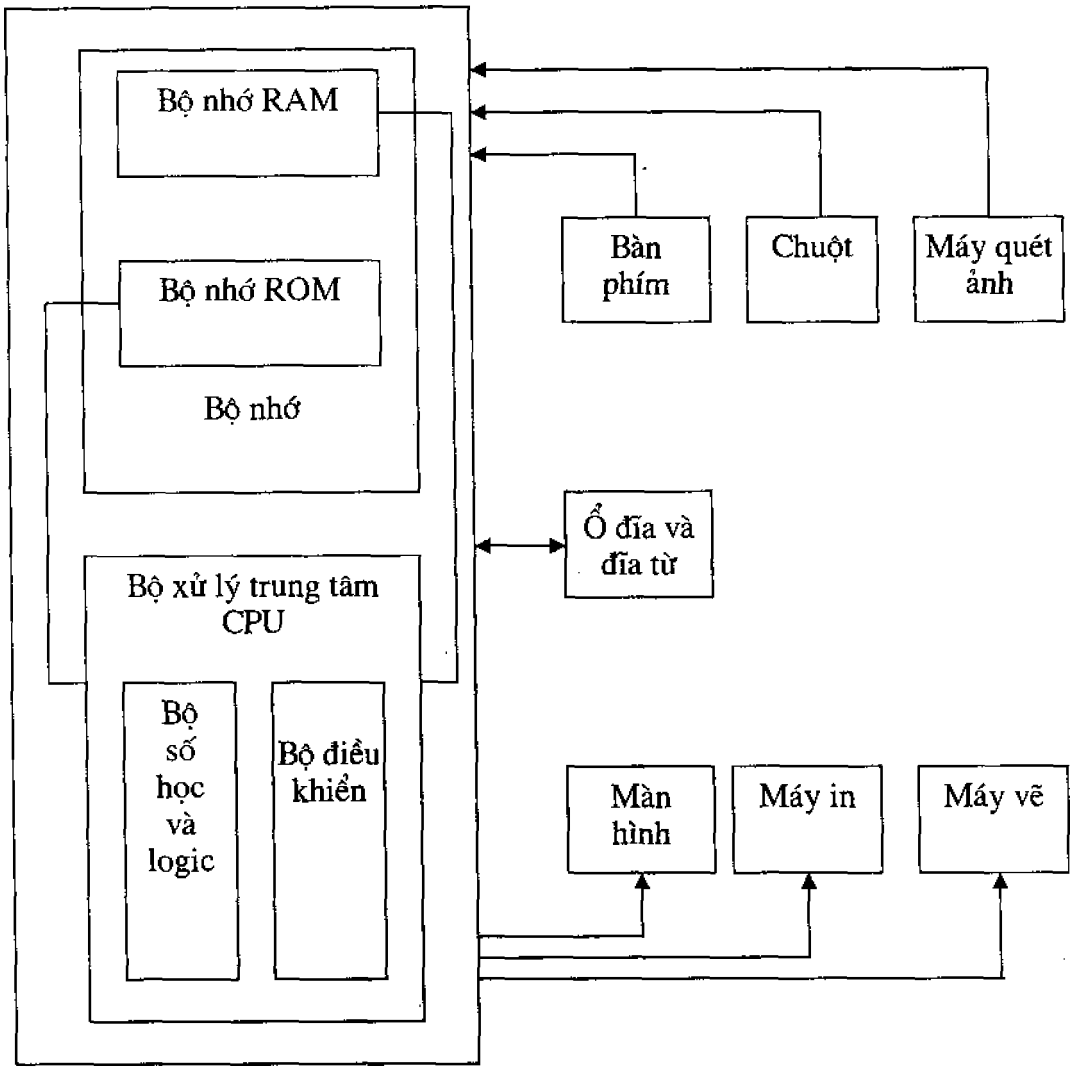
CỘNG 9

Kết quả (15 + 20 = 35) được ghi trong thanh ghi tổng A



Hiện trạng của bộ nhớ và thanh ghi tổng A
(sau khi thực hiện câu lệnh)

SƠ ĐỒ CẤU TRÚC CỦA MTĐT



III. CÁC THIẾT BỊ NGOẠI VI

Để thuận lợi trong việc giao tiếp giữa người sử dụng và máy, người ta lắp thêm cho máy một số thiết bị nhập và xuất thông tin gọi là các thiết bị ngoại vi (ổ đĩa và đĩa từ, máy in, màn hình, máy vẽ...). Các thiết bị này nối với CPU thông qua các kênh truyền tin gọi là các kênh vào ra.

Sau đây là một số thiết bị ngoại vi thường dùng hiện nay.

1. Bộ nhớ ngoài

Bộ nhớ ngoài dùng để lưu giữ thông tin lâu dài, loại thông dụng hiện nay là đĩa từ và đĩa quang (CD-ROM). Bộ nhớ ngoài thường có dung lượng lớn và có thể ghi, đọc thông tin.

1.1. Đĩa mềm (floppy disk)

Đĩa mềm là đĩa nhựa tròn, trên 2 mặt đĩa phủ màng mỏng chất từ tính (thường là ôxit sắt từ), thông tin được ghi vào đĩa theo các đường tròn đồng tâm (gọi là track) được đánh số thứ tự từ ngoài vào tâm đĩa bắt đầu từ track 0,1, 2...

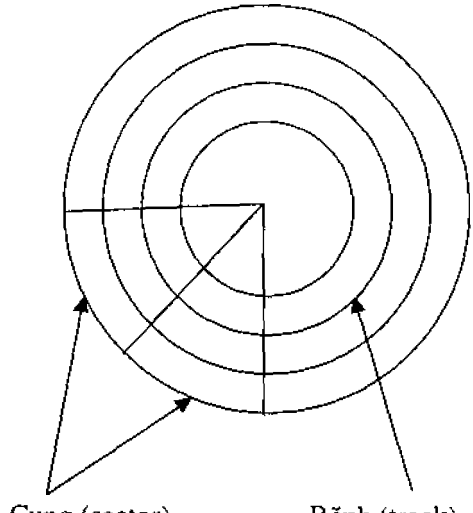
Trên các rãnh này phân đều thành 16 cung - gọi là sector, mỗi cung lại phân thành những khuông nhỏ mà đầu từ có thể ghi/đọc 1 bit. Mỗi cung thường là 512 byte.

Có hai loại đĩa mềm thường dùng:

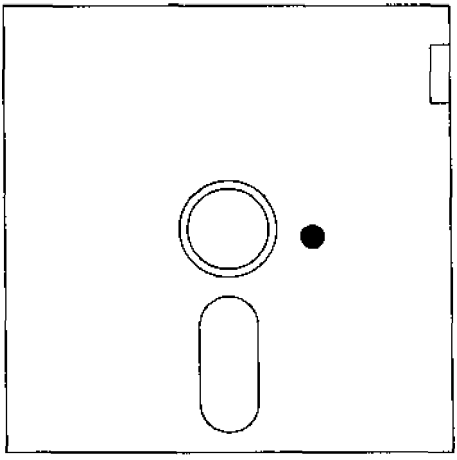
- Loại $5^{1/4}$inch, dung lượng 360KB hoặc 1,2MB
- Loại $3^{1/2}$inch, dung lượng 720KB hoặc 1,44MB

Với cấu tạo như vậy đĩa mềm có thể di chuyển thuận lợi, nhẹ nhàng.

Đối với đĩa mềm loại $5^{1/4}$inch có dung lượng 1,2MB và loại $3^{1/2}$inch có dung lượng 1,44MB khi ta nhìn thấy chữ HD (High Density) trên nhãn của đĩa đó, còn lại đều là loại đĩa có dung lượng 360KB hoặc 720KB.



Các rãnh và cung (sector)



Đĩa mềm loại $5^{1/4}$inch

Đĩa mềm loại $3^{1/2}$inch

1.2. Đĩa cứng (Hard Disk)

Đĩa cứng có cấu tạo tương tự đĩa mềm, nhưng gồm nhiều lá đĩa bằng kim loại được phủ chất từ tính gắn vào một trục quay đồng tâm, cùng tốc độ với nhau. **Đĩa gắn liền với ổ đĩa** tạo thành một khối tách biệt nối với CPU.

Đặc trưng của đĩa cứng là mật độ phủ ôxit sắt từ rất cao, **có nhiều đầu đọc/ghi truy nhập đồng thời** trên cả hai mặt đĩa. **Dung lượng của đĩa cứng rất lớn**, thường từ vài chục Mega bytes đến hàng trăm Gigabytes. **Tốc độ trao đổi thông tin giữa CPU với đĩa cứng nhanh hơn nhiều lần** so với đĩa mềm.

Đĩa cứng gắn liền với ổ đĩa trong máy nên di chuyển không thuận lợi như đĩa mềm, thường chỉ các nhà chuyên môn mới có thể tháo đĩa cứng khi sửa chữa hoặc có lý do đặc biệt.

Ổ cứng được đặt tên là C: và nếu có nhiều ổ đĩa nối với CPU sẽ phải đặt tên từ D:, E:, F:,... theo thứ tự A, B, C,... của bảng chữ cái.

2. Ổ đĩa mềm (Drive)

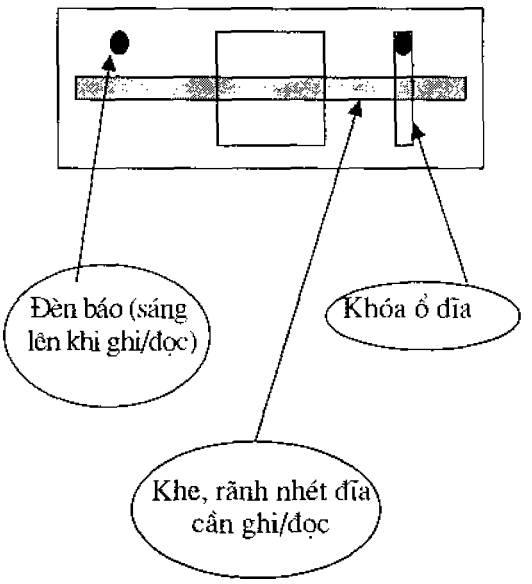
Là thiết bị dùng để đọc và ghi thông tin lên đĩa, ổ đĩa làm quay đĩa từ, trong khi đó đầu đọc của ổ đĩa chuyển động đọc theo bán kính đĩa để ghi hoặc đọc thông tin trên đĩa.

Các ổ đĩa mềm thường có 2 đầu ghi/đọc (gọi là **Head 0 và Head 1**) và tốc độ quay của đĩa thấp, nên tốc độ trao đổi thông tin với CPU chậm hơn so với đĩa cứng rất nhiều.

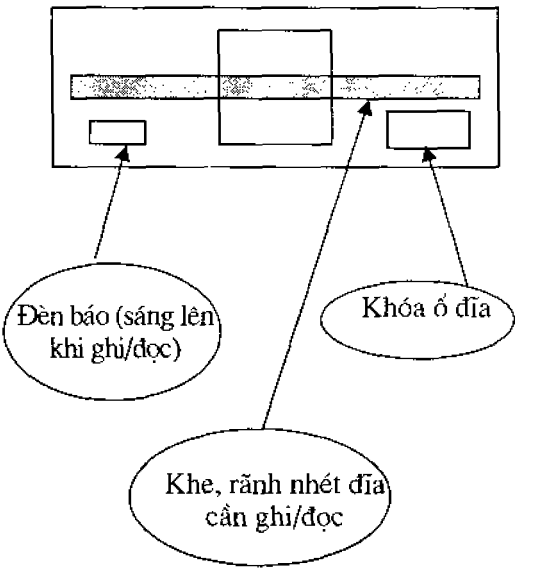
Trên các ổ đĩa mềm đều có đèn hiệu báo khi nó đang đọc hoặc ghi thông tin.

Muốn đọc hoặc ghi thông tin lên đĩa, hãy đưa đĩa vào các ổ đĩa tương ứng, đóng khoá ổ đĩa và sau đó có thể ra lệnh cho máy đọc hoặc ghi lên đĩa trên ổ đĩa hiện hành.

Khi đĩa đang đọc hoặc ghi thông tin lên đĩa, đèn hiệu báo sáng, lúc này không được di chuyển đĩa khỏi ổ đĩa.



Ổ đĩa mềm loại $5^{1/4}$ inch đã lắp đĩa để chuẩn bị ghi/đọc

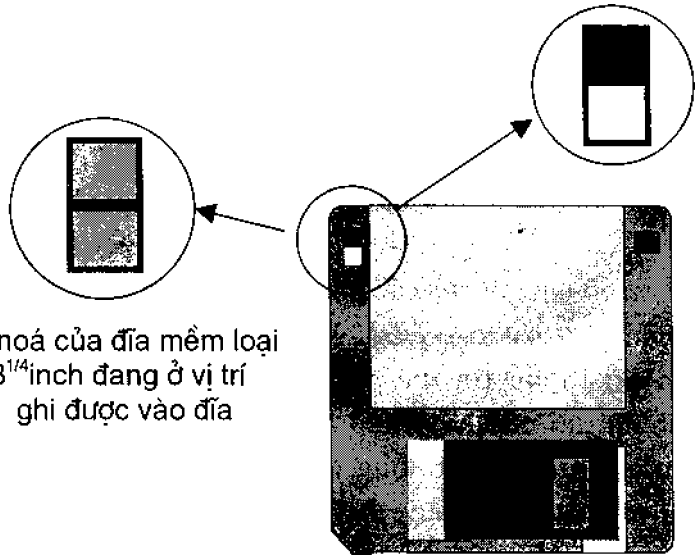


Ổ đĩa mềm loại $3^{1/2}$ inch đã lắp đĩa để chuẩn bị ghi/đọc

Chú ý khi thao tác:

- Không sờ tay vào phần đĩa để ghi đọc.
- Cầm đĩa vào phần nhãn, ngửa mặt đĩa lên trên, hướng về phần khoá đĩa, đẩy hết phần đĩa vào khe rãnh của ổ đĩa.
- Đóng khoá đĩa.

Các đĩa mềm đều có nhãn chống ghi, đối với đĩa loại $3^{1/2}$inch, như hình dưới đây là khoá đặt ở vị trí chống ghi vào đĩa (ta nhìn thấy lỗ hổng trên khoá), lúc này chỉ đọc được thông tin từ đĩa vào bộ nhớ trong để xử lý.

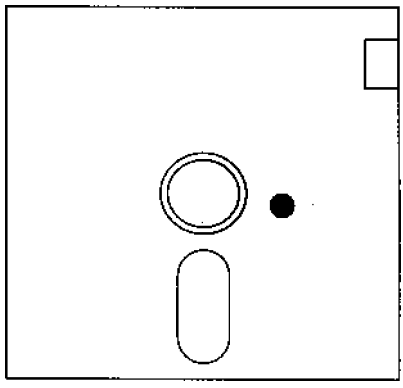


Khoá của đĩa mềm loại $3^{1/4}$inch đang ở vị trí ghi được vào đĩa

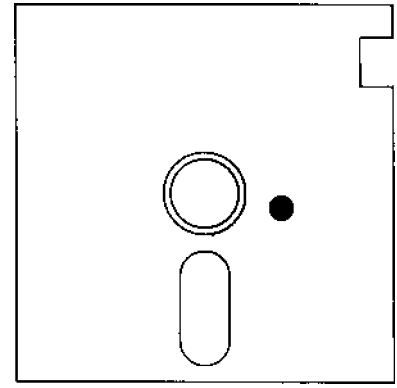
Đĩa mềm loại $3^{1/2}$inch khi đang khoá (không ghi được vào đĩa)

Ngược lại nếu ô khoá ở vị trí (không nhìn thấy lỗ hổng trên ô khoá), đẩy nút đen xuống dưới (không nhìn thấy lỗ hổng trên khoá) là được phép ghi lên đĩa.

Đối với đĩa loại $5^{1/4}$ inch, nhãn bảo vệ chống ghi sẽ bọc phần khoá lại, khi đó không thể ghi vào đĩa mềm này trên ổ đĩa được. Ngược lại nếu muốn ghi vào đĩa mềm này thì phải tháo nhãn ra, (như hình dưới).



Đĩa mềm loại $5^{1/4}$ inch khi đang khoá (không ghi được vào đĩa)



Đĩa mềm loại $5^{1/4}$ inch khi không khoá (ghi được vào đĩa)

3. Màn hình

Màn hình có cấu trúc giống như màn hình của máy thu hình, là thiết bị để hiện thông báo, hình vẽ, văn bản, ảnh, đồ thị v.v. cho người sử dụng giao tiếp với máy.

Màn hình thường có kích thước 14 inch, hiển thị trắng đen hoặc có nhiều mầu sắc.

Màn hình trắng đen: có tên gọi của các nhà chế tạo: Monochrom.

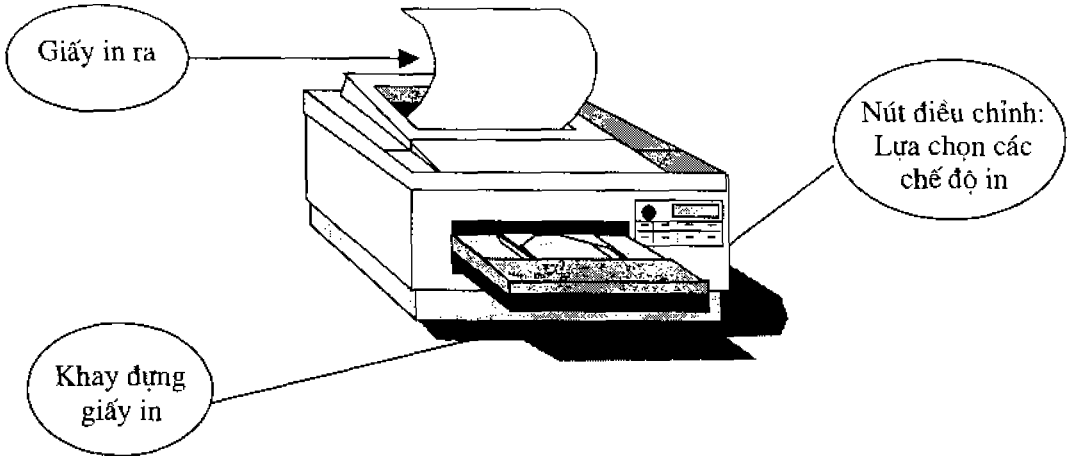
Màn hình mầu thường có các loại: EGA, VGA, SVGA.v.v.

Chất lượng hiển thị thông tin thể hiện ở độ phân giải (Resulation) cao hay thấp. Độ phân giải cao hay thấp là có số tia quét dọc và ngang nhiều hay ít. Các màn hình màu thường có độ phân giải cao, tức có số tia quét dọc, ngang nhiều hơn, chẳng hạn màn hình SVGA có độ phân giải là: 1024x768 .

Phía dưới màn hình có các nút điều chỉnh sáng, tối, tương phản sắc nét, chỉnh hình dịch lên, xuống, cho phóng to, thu nhỏ vùng sáng của màn hình.

4. Máy in (Printer)

Là thiết bị thực hiện chức năng tương tự như màn hình, điều đặc biệt là dữ liệu được in ra trên giấy. Có nhiều loại máy in: in kim và in laser. Một số máy in kim phổ biến hay dùng EPSON LQ (loại 24 kim), EPSON FX (loại 9 kim). Máy in LaserJet HP có mật độ từ 300-600 DPI (chấm/inch).

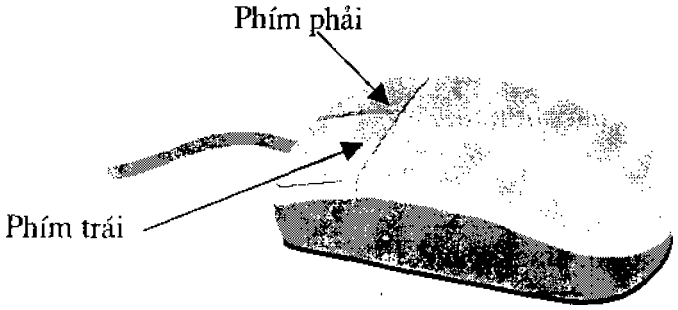


Máy in Laser khổ giấy A4

5. Chuột (Mouse)

Là thiết bị điều khiển vào, tức là để nạp các điểm của toạ độ con chạy chuột trên màn hình vào CPU, chuột thường được dùng trong giao diện đồ hoạ.

Cấu tạo hình dáng nhỏ hơn bao thuốc lá, phía trên có các phím, thường là 2 hoặc 3 phím, con chạy của chuột xác định vị trí tác động của chuột trên màn hình. Khi di chuyển chuột trên mặt bàn, con chạy chuột chuyển động tịnh tiến tương ứng.

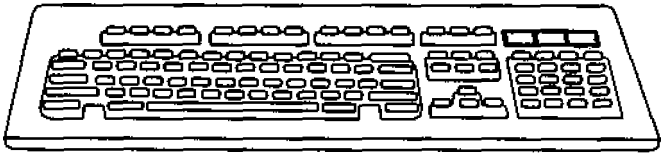


Rê chuột trên mặt bàn

6. Bàn phím (Keyboard)

Là thiết bị quan trọng trong các thiết bị đưa thông tin vào, cho phép người sử dụng nhập chương trình, ra lệnh điều khiển, nhập dữ liệu tính toán v.v.. Về cấu tạo, tương tự như bàn phím máy chữ, nhưng nó còn có thêm rất nhiều các phím điều khiển và các phím chức năng.

Bàn phím thông dụng hiện nay có 101 phím, trên đó phân ra từng nhóm các phím có chức năng giống nhau:



Vùng phím chính: gồm 58 phím các chữ cái la tinh, các chữ số, các dấu, ký hiệu thường dùng và các phím điều khiển.

Phím Caps Lock Caps Lock

Dùng để chuyển chế độ từ chữ HOA sang chữ thường và ngược lại. Nếu đèn báo Caps Lock đang sáng, các phím chữ cái ấn từ bàn phím đều hiển thị chữ hoa, nếu ta ấn phím Caps Lock đèn sẽ tắt và chuyển sang chế độ chữ thường.

Phím Shift Shift

Đối với những phím có hai ký tự, chẳng hạn: > . muốn có được ký tự phía trên là > (dấu lớn hơn) ta hãy ấn và giữ phím **Shift** đồng thời ấn phím > . rồi bỏ cả hai phím ra.

Phím Backspace Backspace hay ←

Dùng để xoá ký tự đứng trước con chạy (cursor hay còn gọi là con trỏ).

Các phím Ctrl Alt Shift

Gọi là các phím điều khiển, chúng được dùng kết hợp với các phím khác để tạo ra các lệnh điều khiển, khi chỉ ấn riêng lẻ từng phím này thì không có hiệu quả gì cả.

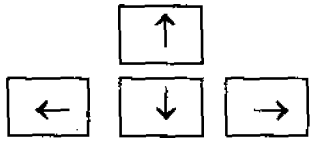
Các phím chức năng F1 F2 F11 F12

Tuỳ từng hệ thống phần mềm quy định chức năng cho các phím này, các phím này thực chất thực hiện một công việc tổng hợp các đơn nguyên thành phần.

Phím Enter Enter ↵

Khi ra lệnh cho máy ta phải ấn các phím chữ cái để biên tập câu lệnh, sau đó tại thời điểm quyết định, ta ấn phím ENTER tức là ra lệnh cho máy hãy thực hiện câu lệnh ngay lập tức.

Các phím mũi tên (arrow)
Dùng để dịch chuyển vị trí con chạy theo chiều tương ứng của các phím đó.



Phím insert (Ins) [Insert] hay [Ins]

Để chuyển chế độ ấn phím đè lên ký tự tại vị trí con chạy hay chèn vào vị trí con chạy một ký tự, đồng thời đẩy các ký tự (nếu có) kể từ vị trí con chạy (con trỏ) sang phải.

Cứ một lần ấn phím là thay đổi chế độ (và ngược lại): **CHÈN** ⇄ **ĐÈ**

Phím Delete (Del) [Delete] hay [Del]

Dùng để xoá ký tự tại vị trí con chạy (cursor hay còn gọi là con trỏ).

Các phím khác: [Esc] [Home] [End] [Page Up] [Page Down]

Tuỳ thuộc vào các hệ phần mềm quy định chức năng của chúng.

Câu hỏi ôn tập

1. Nêu nguyên tắc xử lý thông tin trong máy tính? (có thể diễn tả và giải thích bằng sơ đồ khối).
2. Nêu chức năng và quá trình thực hiện của các bộ xử lý máy tính?
3. Trình bày chức năng và công dụng các thiết bị ngoại vi: Màn hình, máy in, ổ đĩa, bàn phím, chuột?

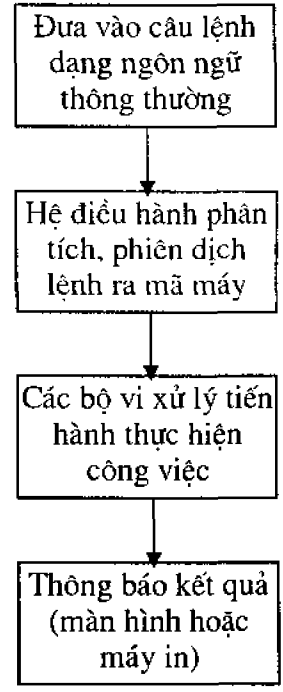
Chương 3

TỔNG QUAN VỀ HỆ ĐIỀU HÀNH VÀ FILE

I. HỆ ĐIỀU HÀNH LÀ GÌ?

Hệ điều hành là hệ thống chương trình điều khiển toàn bộ hoạt động của máy, bao gồm việc: quản lý, lưu trữ thông tin trên đĩa và các hoạt động của thiết bị ngoại vi (máy in, màn hình, bàn phím v.v.)

Hệ điều hành được lưu giữ dưới dạng các tệp trên đĩa từ, khi khởi tạo máy, hệ điều hành tự động được tải vào bộ nhớ trong và sẵn sàng chờ lệnh của người sử dụng đưa vào.



Người sử dụng (USER) ra lệnh cho máy tiến hành công việc nào đó bằng cách đưa vào câu lệnh viết ở dạng ngôn ngữ thông thường, hệ điều hành nhận lệnh, phân tích và dịch câu lệnh sang dạng mã máy, bắt máy tiến hành công việc rồi thông báo kết quả ra màn hình hoặc máy in.

Các câu lệnh được thực hiện có thể là các lệnh của hệ điều hành, các lệnh của USER tạo ra và lưu trữ thành từng tệp, đối tượng xử lý của hệ điều hành là các tệp.

Việc thực hiện trực tiếp từng lệnh trên máy gọi là thực hiện các lệnh đơn, mỗi một lệnh sẽ làm một chức năng cụ thể.

Ví dụ lệnh của HĐH để hiển thị danh sách các tệp và thư mục con trên thư mục hiện thời gồm các bước sau:

1- Biên tập dòng lệnh: gõ từ bàn phím từng ký tự tạo câu lệnh và các tham số tự chọn (nếu cần), trong bước này nếu gõ sai có thể dùng phím **Backspace** để xóa và gõ lại.

C:\> DIR

2- Nhấn phím ENTER để máy thực hiện thao tác của câu lệnh.

C:\> DIR

3- Màn hình thông báo kết quả (tuỳ theo câu lệnh).

II. TỆP TIN (FILE)

Tệp là tập hợp thông tin có liên quan logic với nhau, cùng phục vụ cho một chương trình trong MTĐT, các tệp được tổ chức với mục đích thuận lợi cho việc lưu giữ, tìm kiếm, thay đổi và xử lý theo một thể thức thống nhất mà HĐH quy định.

Các tệp được lưu trữ trên đĩa từ, trong quá trình xử lý thông tin, dữ liệu từ các tệp được tải từ đĩa từ vào bộ nhớ trong để tiện cho quá trình xử lý, sau đó lại ghi kết quả xử lý lưu trữ lên đĩa từ.

Do nội dung thông tin khác nhau, chức năng khác nhau mà HĐH quy định và phân chia các loại tệp khác nhau: tệp văn bản, tệp chương trình, tệp dữ liệu, tệp số liệu v.v.

Tệp trong máy tính có thể do USER tạo ra hoặc do sao chép, cài đặt từ các hệ phần mềm ứng dụng.

Tệp là đơn vị xử lý của HĐH. Đặc trưng chung, cơ bản của tệp là: Tên tệp (file name); độ lớn của tệp (số bytes chiếm giữ); thời gian sinh ra tệp và kiểu loại tệp (tệp thực hiện chương trình hay tệp dữ liệu, văn bản v.v.).

1. Cách đặt tên tệp (file name)

Tên tệp do người sử dụng tự đặt, sao cho dễ đọc, dễ nhớ và phản ảnh nội dung được chứa trong tệp đó. Khi đặt tên tệp phải theo quy cách như sau:

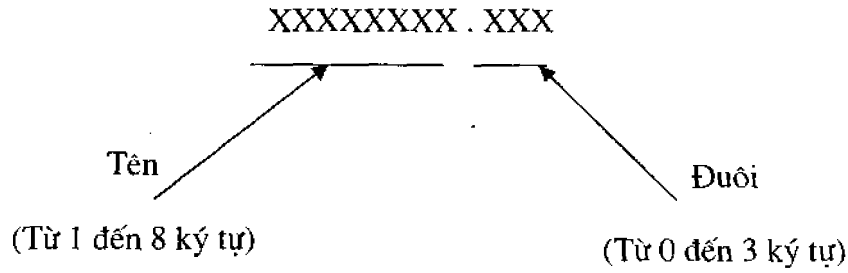
Tên tệp có hai phần: **TÊN.KIỂU**

TÊN tệp là một dãy liên tiếp (không chứa khoảng trống) các chữ cái, chữ số, các ký hiệu; có độ dài từ 1 đến 8 các ký tự, ký tự đầu tiên thường là chữ cái.

KIỂU (đuôi) tệp là dãy liên tiếp các ký tự (không kể ký tự khoảng trống: Space), độ dài từ 0 đến 3.

Kiểu còn có tên gọi là phần mở rộng hay đuôi tệp. Thực tế đuôi tệp thường do hệ thống ứng dụng tự bổ sung thêm vào tên tệp. Nhưng tên tệp có thể có hoặc không có phần đuôi tuỳ theo ta tự đặt, khi tên tệp có thêm đuôi thì phải dùng dấu (.) ngăn cách.

Tên tệp minh hoạ:



2. Ví dụ về một số loại tệp

2.1. Sử dụng hệ soạn thảo văn bản (chẳng hạn WINWORD) để viết một báo cáo, ta phải lưu giữ thông tin dưới dạng một tệp, và phải đặt tên, chẳng hạn: BAOCAO.DOC. Tệp BAOCAO.DOC là một loại tệp văn bản được soạn thảo từ hệ soạn thảo WINWORD, khi đặt tên tệp, thường ta chỉ đặt tên BAOCAO, còn phần kiểu tệp do hệ WINWORD tự động thêm vào (.DOC) và ta có tên tệp là BAOCAO.DOC.

2.2. Cũng công việc như ví dụ trên, nhưng nếu dùng hệ soạn thảo **Notepad** để viết báo cáo, và giả sử ta cũng đặt tên tệp là BAOCAO thì ta lại được tên tệp đầy đủ là: BAOCAO.TXT.

2.3. Tệp TREE.COM (cây) ta thường thấy trên đĩa C: là tệp của hệ điều hành DOS, gọi là tệp thực hiện được từ dấu nhắc của HĐH, nội dung thông tin của tệp ở dạng mã nhị phân, tức là mã máy thực hiện. Những tệp có phần đuôi .COM; .EXE; .BAT đều thực hiện được từ dấu nhắc DOS.

3. Một số tên tệp và đuôi tệp tránh đặt

CON: tên dành riêng cho thiết bị màn hình.

AUX, COM1: tên dùng cho cổng truyền nối tiếp 1.

COM2: tên cho cổng truyền nối tiếp 2.

PRN, LPTn (n=1,2,3): tên chỉ thiết bị máy in.

NUL: tên của tệp không tồn tại.

COM, EXE: là đuôi tệp của các chương trình thực hiện dạng mã máy.

Câu hỏi ôn tập

1. Thông tin quản lý của Hệ điều hành được tổ chức như thế nào?
2. Tên tệp cho phép tối đa bao nhiêu chữ cái và các ký hiệu?
3. Phần đuôi tệp có nhất thiết phải có hay không?
4. Nêu một số tên tệp và đuôi tệp tránh đặt?

Chương 4

KHỞI TẠO MÁY, HỆ ĐIỀU HÀNH MS-DOS

I. HỆ ĐIỀU HÀNH MS-DOS

Hệ điều hành MS-DOS (MicroSoft Disk Operating System) của hãng MicroSoft (Mỹ) là hệ thống chương trình tạo điều kiện thuận lợi cho người sử dụng: giao tiếp với máy, quản lý các tệp tin trên đĩa và điều khiển các thiết bị ngoại vi.

MS-DOS lưu trên đĩa thành các tệp, mỗi tệp thực hiện một chức năng, trong đó 3 tệp cốt lõi là:

MSDOS.SYS

IO.SYS

COMMAND.COM

Khi khởi tạo máy, 3 tệp này sẽ tự động được tải từ đĩa vào bộ nhớ trong để sẵn sàng thực hiện lệnh của người sử dụng và điều khiển các thiết bị của máy tính.

Đối tượng xử lý của MS-DOS là những tệp, chẳng hạn: tạo tệp, xoá tệp, sao chép tệp, đổi tên tệp v.v.

MS-DOS có nhiều Version khác nhau, trải qua thời gian các Version được nâng cấp dần, phiên bản gần đây là 6.0, 6.2, 6.22 có nhiều cải tiến thuận lợi cho người sử dụng. Hiện nay hệ điều hành Windows 9X có phần cốt lõi hoàn toàn tương thích với hệ điều hành MS-DOS. Tất cả các lệnh trình bày trong giáo trình này được áp dụng và hoàn toàn tương thích với Windows 9X.

Hệ thống các tệp lệnh của MS-DOS có hai loại: các lệnh thường trú và các lệnh ngoại trú.

Lệnh thường trú: là những lệnh thường dùng (nằm trong tệp COMMAND.COM) đã được nạp tự động vào bộ nhớ trong USER được phép ra lệnh tại bất cứ vị trí thư mục hoạt động nào.

Lệnh ngoại trú: là tên các tệp lệnh của MS-DOS, mỗi tệp là một lệnh của DOS, các tệp này thường được sao chép từ các đĩa hệ thống MS-DOS vào thư mục DOS trên ở đĩa cứng C. Như vậy, để thực hiện một lệnh ngoại trú ta phải có tệp lệnh tại thư mục hoạt động và ra lệnh từ đây.

Một số lệnh thường trú: DIR; REN; DATE; TIME; COPY; DEL; CLS; VER; CD(CHDIR); MD(MKDIR); RD(RDIR); VOL; TYPE; PATH; PROMPT; ECHO; VERIFY...

Một số tệp lệnh ngoại trú hay dùng: FORMAT.COM; TREE.COM; SYS.COM; DISKCOPY.COM; CHKDSK.EXE; SCANDISK.EXE; DEFRAG.EXE...

II. KHỞI TẠO MÁY - NẠP HỆ ĐIỀU HÀNH MS-DOS

1. Khởi tạo từ đĩa cứng C:

- Tháo các đĩa mềm khỏi ổ đĩa A:
- Bật công tắc máy, máy tự kiểm tra bộ nhớ và các thiết bị, sau đó nạp hệ điều hành DOS từ C: vào bộ nhớ trong.
- Màn hình xuất hiện dòng thông báo: tháng, ngày, năm hiện tại; nếu ngày tháng đúng thì ta chỉ bấm phím ENTER, nếu sai hãy vào lại: tháng, ngày, năm rồi bấm ENTER.
- Tiếp là dòng thông báo thời gian; nếu thời gian đúng, ta bấm phím ENTER, nếu sai, hãy vào lại thời gian cho chính xác rồi bấm phím ENTER.
- Sau cùng xuất hiện trên màn hình:

  C:\>_

  là đã khởi tạo xong.
- Điểm sáng (_) nhấp nháy gọi là con trỏ màn hình, phần ký tự đứng trước con trỏ C:\> gọi là dấu nhắc lệnh, từ đó USER đưa vào các câu lệnh và bấm phím ENTER để thực hiện câu lệnh.

2. Khởi tạo MS-DOS từ đĩa hệ thống đặt tại ổ A:

- Đưa đĩa hệ thống vào ổ A:, đóng khoá.
- Bật công tắc máy, máy tự kiểm tra bộ nhớ và các thiết bị, sau đó nạp hệ điều hành DOS từ đĩa A: vào bộ nhớ trong.

- Màn hình xuất hiện dòng thông báo: tháng, ngày, năm hiện tại; nếu ngày tháng đúng thì ta chỉ bấm phím ENTER, nếu sai hãy vào lại: tháng, ngày, năm rồi bấm ENTER.

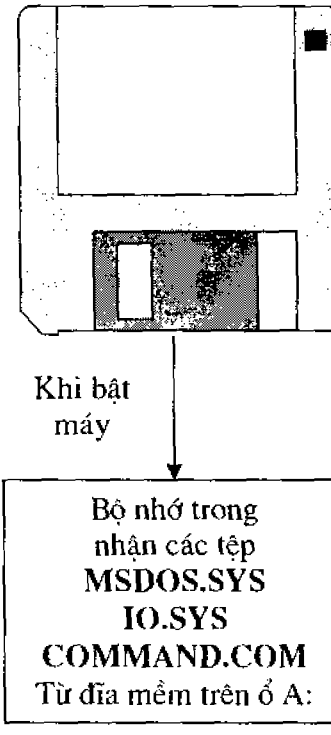
- Tiếp là dòng thông báo thời gian; nếu thời gian đúng, ta bấm phím ENTER, nếu sai, hãy vào lại thời gian cho chính xác rồi bấm phím ENTER.

- Sau cùng xuất hiện trên màn hình:

A:\>_

là đã khởi tạo xong.

- Điểm sáng (_) nhấp nháy gọi là con trỏ màn hình, phần ký tự đứng trước con trỏ **A:\>** gọi là dấu nhắc lệnh, từ đó USER đưa vào các câu lệnh và bấm phím ENTER để thực hiện câu lệnh.



Quá trình khởi tạo máy Từ đĩa mềm hệ thống trên ổ A:

Chú ý:

- Cách khởi tạo trên gọi là khởi tạo CỨNG hay còn gọi là khởi tạo LẠNH.

- Trong quá trình chạy máy, có thể xảy ra trường hợp "mất điều khiển" hay còn gọi là treo máy - tức là máy không nhận lệnh từ bàn phím, lúc đó hoặc là không thấy con trỏ màn hình, hoặc là máy phát ra tiếng còi tút tút..., hoặc chương trình thực hiện bị quẩn, không dừng v.v. Khi đó ta phải khởi tạo lại máy: gọi là khởi tạo MỀM hay còn gọi khởi tạo NÓNG bằng cách bấm đồng thời các phím: CTRL-ALT-DEL.

Khởi tạo NÓNG có thể từ C: (tháo đĩa mềm trên ổ A: ra khỏi ổ đĩa) hoặc từ đĩa hệ thống MS-DOS trên ổ A:. Khởi tạo nóng, máy chỉ kiểm tra bộ nhớ, mà không nạp cấu hình từ ROM nên nhanh hơn.

Khởi tạo NÓNG có thể thực hiện trên các máy có nút (công tắc) RESET.

Việc khởi tạo NÓNG như trên đôi khi cũng không thể đưa máy trở lại hoạt động bình thường được, lúc đó ta áp dụng cách khởi động LẠNH bằng cách tắt máy rồi bật trở lại hoặc bấm nút RESET. Sau khi tắt máy hãy chờ khoảng 30 giây rồi bật công tắc điện khởi tạo lại máy.

Câu hỏi ôn tập

1. Máy tính chưa nạp Hệ điều hành MS-DOS có thông báo như thế nào trên màn hình?
2. Dấu nhắc hệ điều hành DOS có tác dụng gì?
3. Nêu từng bước nạp DOS từ đĩa cứng C: và đĩa mềm A:
4. Các biểu hiện máy bị treo? Cách xử lý?

Chương 5

THƯ MỤC VÀ ĐƯỜNG DẪN

I. THƯ MỤC (DIRECTORY)

1. Giới thiệu

Để thuận lợi cho việc quản lý các tệp trên các ổ đĩa, hệ điều hành DOS cho phép tạo ra các thư mục, trên đó ta lưu giữ từng nhóm các tệp có chức năng liên quan với nhau. Chẳng hạn: tất cả các tệp của MS-DOS để trên thư mục DOS, vậy khi tìm tệp nào của MS-DOS ta sẽ tìm đến thư mục DOS để lấy tệp đó ra.

Thư mục được lưu giữ cùng với các tệp trên đĩa, nó có thể được tạo ra bởi USER hoặc bởi các hệ chương trình ứng dụng. Cũng giống như tệp, mọi thư mục được tạo ra đều phải đặt tên. Quy cách đặt tên thư mục giống như cách đặt tên tệp.

DOS phân ra các loại thư mục để quản lý:

+ Thư mục gốc (Root directory):

Mỗi đĩa chỉ có một thư mục gốc, do máy tạo ra khi tạo khuôn đĩa, ký hiệu là: \ (*backslash*) đứng ngay sau tên ổ đĩa.

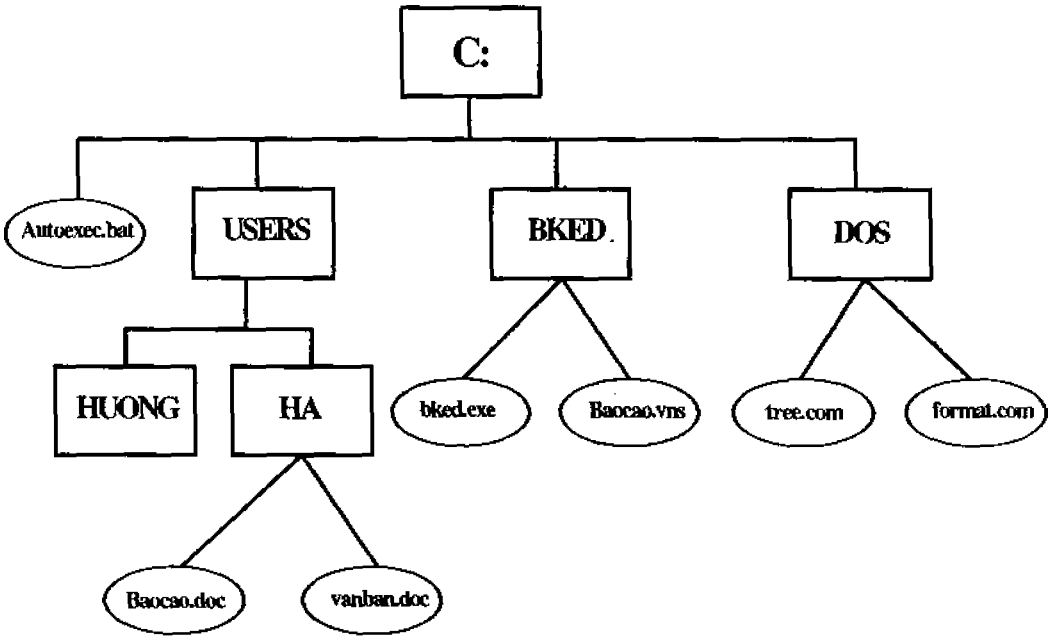
+ Thư mục con (subdirectory)

Mọi thư mục, trừ thư mục gốc, đều là thư mục con.

Ví dụ: C:\DOS>_

Trong đó C: Tên ổ đĩa
DOS: Thư mục con
\ Thư mục gốc

Các thư mục trên đĩa được tổ chức phân nhánh giống như các cành của cây, còn các tệp là lá. Ví dụ hình dưới mô tả tổ chức lưu trữ các tệp và các thư mục trên ổ đĩa cứng C:\ ở dạng cây:



Tên ở trong các hình tròn:  chỉ tên tệp

Tên ở trong các hình chữ nhật:  chỉ tên thư mục

Thư mục gốc (\) của ổ đĩa C: có các thư mục con (mức 1) là: USERS, BKED, DOS và tệp autoexec.bat. Bản thân thư mục USERS lại có hai thư mục con (mức 2): HUONG, HA.

Thư mục BKED chứa 2 tệp: BAOCAO.VNS, BKED.EXE.

Thư mục DOS chứa 2 tệp: TREE.COM, FORMAT.COM.

2. Chú ý khi đặt tên thư mục và tên tệp

- DOS không phân biệt chữ hoa hay chữ thường:
  Chẳng hạn tên tệp BAOCAO.vns = BAOCAO.VNS = baocao.vns
- Phần tên và kiểu có thể có các ký tự: chữ cái: A đến Z, chữ số: 0 đến 9, các ký hiệu: _ (gạch nối _ underscore).
- Không được chứa các ký tự đặc biệt như: khoảng trống (space), ^ (mũ - caret), # (dấu số - number sign), % (phần trăm - percent sign), $ (dollar sign), & (dấu và - ampersand), @ (dấu a vòng - at sign) và dấu gạch

ngược: \ (backslash), dấu phảy: , (comas), dấu chấm: . (period) (không kể dấu chấm tách biệt phần tên và phần kiểu).

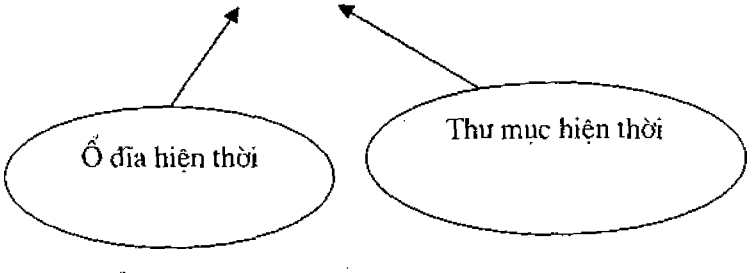
- Không được phép đặt tên tệp, tên thư mục giống nhau trên cùng thư mục.

3. Thư mục và ổ đĩa hiện thời

Ổ đĩa hiện thời, thư mục hiện thời chỉ vị trí, thời điểm USER sắp sửa thực hiện câu lệnh nào đó.

Ổ đĩa hiện thời (drive current) còn gọi là ổ đĩa chủ hay ổ đĩa hoạt động. Thư mục hiện thời (nhận biết được trên màn hình) là tên thư mục đứng ngay trước dấu lớn hơn (>) của hệ thống.

Ví dụ: Trên màn hình **C : \ HA>DEL BAOCAO.VNS**



là câu lệnh chuẩn bị thực hiện việc xoá tệp BAOCAO.VNS trong thư mục HA.

II. ĐƯỜNG DẪN (PATH)

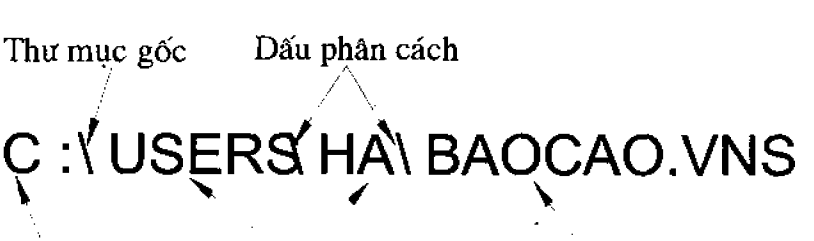
Khi tìm kiếm thư mục hoặc tệp nào đó, ta phải chỉ đường cho DOS biết hãy đi theo nhánh nào của cây thư mục để tìm ra tệp hoặc thư mục con cần chuyển tới. Việc chỉ đường đi như vậy nhờ khái niệm đường dẫn.

1. Định nghĩa

Đường dẫn là dãy liên tiếp tên các thư mục, ngăn cách nhau bởi dấu gạch chéo ngược "\" và không chứa khoảng trống. Nhưng dấu "\" đứng ngay sau tên ổ đĩa là thư mục gốc.

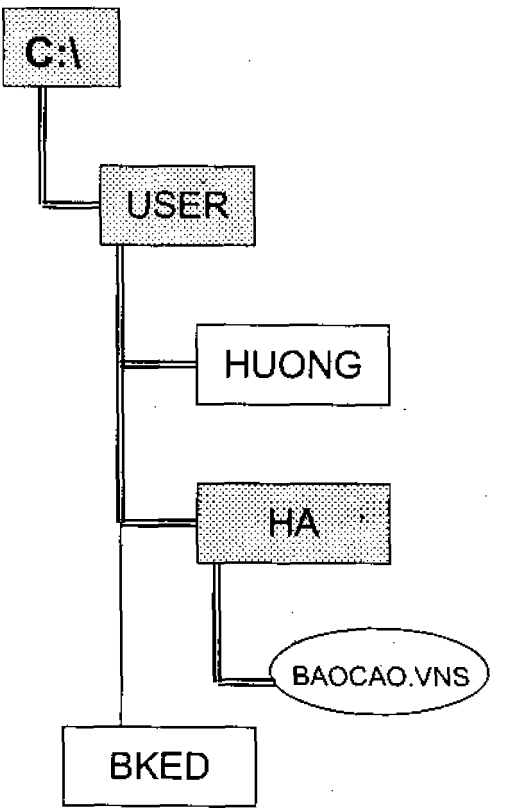
+ Đường dẫn tới một thư mục là đường dẫn mà tên cuối cùng của đường dẫn là tên thư mục cần chuyển tới (tạo ra).

Đường dẫn tới thư mục thường áp dụng cho các lệnh quản lý thư mục: CD, MD, RD.



Đường dẫn đến tệp

Đường dẫn tới tệp thường áp dụng cho các lệnh xử lý tệp: COPY, DEL, REN, DIR ...



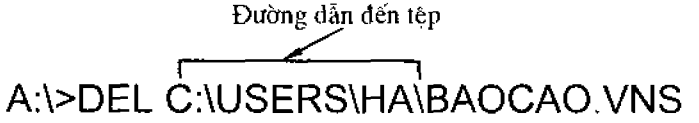
Đường dẫn (kẻ đúp) đến tệp
(cấu trúc cây thư mục)

Hình trên mô tả đường dẫn bắt đầu từ thư mục gốc của ổ đĩa C: (theo cấu trúc các tệp và thư mục) và gọi là đường dẫn đầy đủ; nghĩa là đường dẫn có tên ổ đĩa, thư mục gốc, các thư mục con.

Sử dụng đường dẫn đầy đủ thường có trong các câu lệnh mà đối tượng tác động (tệp và thư mục) không ở trên ổ đĩa và thư mục hiện hành.

2. Ví dụ

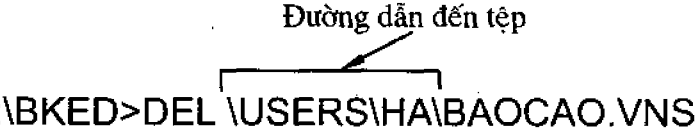
Trong câu lệnh dưới đây: ổ đĩa và thư mục hiện thời là A: và thư mục gốc. Câu lệnh xoá tệp BAOCAO.VNS trên thư mục HA của thư mục USERS tại ổ đĩa C:



Cũng như vậy, nhưng ổ đĩa hiện thời C:, thư mục gốc là hiện thời:
Câu lệnh có dạng:

Ta nhận thấy trong trường hợp này ta không cần thiết đưa ra tên ổ đĩa và tên thư mục gốc trong phần đường dẫn.

- Nếu thư mục hiện thời là thư mục BKED, muốn xoá tệp BAOCAO.VNS của thư mục HA, thì câu lệnh bây giờ lại cần phải có tên thư mục gốc.



- Vì tệp BAOCAO.VNS cùng nằm trên ổ C: (hiện thời) nên không phải viết tên ổ đĩa trên đường dẫn, nhưng phải có tên thư mục gốc "\".

Chú ý:

Độ dài tối đa của đường dẫn có thể tới 67 ký tự (kể cả ký tự chỉ tên ổ đĩa).

Câu hỏi ôn tập

1. Việc tổ chức các tệp trên cây thư mục có thuận lợi như thế nào?
2. Trên cùng một thư mục có thể có hai tệp hoặc tên thư mục giống nhau được không?
3. Thế nào là thư mục hiện thời?
4. Khái niệm đường dẫn và ý nghĩa đường dẫn khi thực hiện lệnh DOS?

Chương 6

CÁC LỆNH CỦA HỆ ĐIỀU HÀNH MS-DOS

I. LỆNH THAY ĐỔI DẤU NHẮC HỆ THỐNG

Để tạo dấu nhắc tên ổ đĩa hiện thời và thư mục hiện thời ta phải dùng lệnh:

PROMPT PG.

$P để hiển thị tên thư mục hiện thời.

$G để tạo dấu nhắc: > (dấu lớn hơn).

Thường lệnh này được đưa vào trong tệp AUTOEXEC.BAT để mỗi khi ta bật máy, tệp AUTOEXE.BAT tự động được thực hiện - tức các lệnh liệt kê trong tệp lần lượt được thực hiện, trong đó lệnh tạo dấu nhắc:

PROMPT PG.

Đối với các phiên bản từ MS-DOS 6.0 về sau này thì không cần thiết vì hệ thống tự động tạo ra dấu nhắc.

II. LỆNH HIỂN THỊ DANH SÁCH CÁC TỆP VÀ THƯ MỤC

Câu lệnh tổng quát:

DIR [Tên ổ đĩa:] [/P] [/W] [T]

Chức năng: Hiển thị danh sách các tệp, thư mục lên màn hình.

Các tuỳ chọn:

[Tên ổ đĩa:] chỉ tên ổ đĩa chứa các tệp, thư mục cần hiển thị.

[/P] cho hiển thị từng trang màn hình (Page).

[/W] hiển thị vắn tắt (không hiển thị giờ và ngày, tháng, năm cập nhật tệp) theo hàng ngang, cứ 5 tên tệp hoặc tên thư mục trên một dòng màn hình.

[T] chỉ tên thư mục hoặc tên tệp cụ thể cần hiển thị lên màn hình.

Chú ý: Các tuỳ chọn trên có thể kết hợp đưa ra trên cùng một câu lệnh.

Ví dụ 1: Lệnh hiển thị tên tệp, tên thư mục trên thư mục gốc của ổ đĩa C: lên màn hình. Trong câu lệnh, không chỉ ra tên đường dẫn, HĐH hiển thị ngay danh sách tên tệp và thư mục của thư mục hiện thời: \ (gốc.)

C:\>DIR↵

Màn hình hiển thị thông tin như sau:

```
Volume in drive C is ITCOM
Volume Serial Number is 1B5E-11FE
Directory of C:\

CONFIG SYS        45   06-27-98 12:15a
ACAD         <DIR>     05-26-98 9:12p
LUUCHU       <DIR>     05-26-98 9:15p
NU           <DIR>     05-26-98 10:09p
SETUP        <DIR>     05-26-98 10:54a
NETLOG TXT     5,741   06-06-98 8:03p
MOUSE        <DIR>     05-27-98 6:28p
AUTOEXEC BAT     152   06-27-98 12:12a
WINDOWS      <DIR>     06-06-98 7:36p
COMMAND COM   93,880   02-08-98 9:37p

       4 file(s)      99,818 bytes
      13 dir(s) 245,014,528 bytes free

C:\>
```

Các tệp và thư mục đều tương ứng có tháng, ngày, năm, giờ (a: sáng; p:chiều) chỉ thời gian thư mục và tệp được cập nhật lần cuối cùng.

Tên các tệp: tương ứng là số bytes, tệp đã sử dụng để chứa nội dung.

Tên các thư mục: tương ứng có chữ <DIR> để chỉ thư mục chứ không phải tệp.

Sau khi hiển thị xong, dấu nhắc trở về dấu nhắc hệ thống trước lúc ra lệnh.

Ví dụ 2:

Giả sử: thư mục hiện thời vẫn là gốc ở ổ đĩa C:\>.

Muốn hiển thị các tệp của thư mục DOS lên màn hình, trong câu lệnh ta phải đưa đường dẫn tới thư mục DOS.

Vì thư mục hiện thời là gốc nên đường dẫn chỉ cần DOS là đủ.

C:\>DIR DOS ↵

Sau khi thực hiện màn hình sẽ là:

```
Volume in drive C is ITCOM
 Volume Serial Number is 1B5E-11FE
 Directory of C:\DOS

.            <DIR>          08-22-98 8:15p
..           <DIR>          08-22-98 8:15p
CHKDSK EXE        28,096    02-08-98 9:36p
CHOICE COM         5,239    02-08-98 9:36p
COUNTRY SYS       30,742    02-08-98 9:50p
CSCRIPT EXE      135,168    02-09-98 9:05a
CVT EXE          122,975    02-08-98 10:12p
DEBUG EXE         20,554    02-08-98 9:37p
DELTREE EXE       19,083    02-08-98 9:49p
DISKCOPY COM      21,975    02-08-98 9:37p
            8 file(s) 383,832 bytes
            2 dir(s) 240,295,936 bytes free

C:\>_
```

Trên là danh sách các tệp có trong thư mục DOS của ổ C:, sau khi thực hiện lệnh, con trỏ trở lại dấu nhắc hệ thống ở vị trí trước lúc thực hiện câu lệnh.

Chú ý: Nếu thư mục chứa nhiều tệp và thư mục con (trên 25 tên trở lên), ta dùng tuỳ chọn /p để dừng trang màn hình, lúc đó xuất hiện dòng chữ:

```
Press any key to continue
```

Khi đó câu lệnh sẽ là:

C:\>DIR DOS/P ↵

```
Volume in drive C is ITCOM
 Volume Serial Number is 1B5E-11FE
 Directory of C:\DOS

.            <DIR>          08-22-98 8:32p
..           <DIR>          08-22-98 8:32p
ANSI SYS           9,719    02-08-98 9:50p
ATTRIB EXE        15,252    02-08-98 9:35p
BOOTDISK BAT       2,047    02-05-98 6:42p
CHKDSK EXE        28,096    02-08-98 9:36p
CHOICE COM         5,239    02-08-98 9:36p
COUNTRY SYS       30,742    02-08-98 9:50p
CSCRIPT EXE      135,168    02-09-98 9:05a
CVT EXE          122,975    02-08-98 10:12p
DOSKEY COM        15,495    02-08-98 9:38p
DRVSPACE BIN      68,871    02-08-98 9:55p
DRVSPACE SYS       2,135    02-08-98 9:55p
EDIT COM          69,902    02-08-98 9:48p
EDIT HLP          10,790    05-20-97 5:47p
Press any key to continue . . .
```

Nghĩa là hãy bấm phím nào đó để tiếp tục lệnh, đến khi thấy lại dấu nhắc hệ thống

C:\>_

- Nếu ta quan tâm đến kiểu tệp nào đó, chẳng hạn các tệp có phần đuôi là COM, hãy dùng dấu * (thay cho nhóm ký tự của tên), câu lệnh sẽ là:

C:\>DIR DOS*.COM ↵

Lúc này trên màn hình hiện lên tất cả các tệp có phần đuôi là .COM.

- Để in ra máy in những gì mà lệnh DIR hiện lên màn hình ta chỉ cần thêm >PRN vào cuối lệnh: chẳng hạn lệnh in ra máy in các tệp có phần đuôi .COM:

C:\>DIR DOS*.COM >PRN↵

- Kết quả lệnh DIR cũng có thể ghi vào tệp, khi đó ta phải đặt tên tệp chứa kết quả của câu lệnh DIR. Ví dụ ta đặt tên tệp là DIRDOS.

C:\>DIR DOS*.COM >DIRDOS↵

Sau lệnh này ta có tệp kiểu loại văn bản mã ASSCII tên DIRDOS.

III. LỆNH CHUYỂN Ổ ĐĨA HOẠT ĐỘNG

USER có thể chuyển ổ đĩa hoạt động sang ổ khác bằng cách: gõ tên ổ đĩa cần chuyển hoạt động tới rồi ấn ENTER.

Chẳng hạn ta muốn chuyển hoạt động từ đĩa C: sang đĩa A: như sau:

C:\>A: ↵

A:\>_

Hoặc khi này ta lại chuyển sang hoạt động tại ổ B:

A:\>B: ↵

B:\>_

IV. LỆNH TẠO THƯ MỤC

Để tạo thư mục con của thư mục hiện thời dùng lệnh MD (Make Directory).

MD <TÊN THƯ MỤC> ↵

Ví dụ:

- Lệnh tạo thư mục DOS từ thư mục gốc trên ổ đĩa C:

C:\>MD DOS↵

- Lệnh tạo thư mục HA - thư mục con của USERS từ thư mục gốc trên ổ đĩa C:

C:\>MD USERS\HA↵

Trong câu lệnh trên ta đã sử dụng đường dẫn, vì lúc này thư mục hiện thời là thư mục gốc.

Sau khi thực hiện, nếu không thông báo trên màn hình tức là thư mục này đã được tạo, thường dùng lệnh DIR để kiểm tra.

- Lệnh tạo thư mục con ATV của thư mục gốc trên ổ đĩa A:, thư mục hiện thời là thư mục gốc của ổ C:

C:\>MD A:\ATV↵

Trong câu lệnh trên phần đường dẫn là tên ổ đĩa và thư mục gốc A:\>

Nhận xét: Trong các ví dụ nêu trên, câu lệnh không kể thêm tên thư mục gốc (\ đứng ngay sau tên ổ đĩa), vì hiện thời đang ở vị trí thư mục gốc ra lệnh. Còn nếu thư mục hiện thời không phải là thư mục gốc thì phải chỉ ra thư mục gốc trong phần đường dẫn của câu lệnh. Ví dụ như câu lệnh sau:

- Lệnh tạo thư mục HUONG là thư mục con của thư mục USERS của ổ đĩa C:; giả sử thư mục hiện thời là thư mục BKED trên cùng ổ đĩa:

Trước lúc ra lệnh:

C:\BKED>_

C:\BKED>MD \USERS\HUONG↵

Sau khi ra lệnh:

C:\BKED>_

V. LỆNH CHUYỂN THƯ MỤC

USER có thể chuyển hoạt động tới thư mục khác trên cùng ổ đĩa từ thư mục hiện thời bằng câu lệnh CD (Change Directory):

CD <đường dẫn> ↵

Ví dụ:

- Giả sử thư mục hiện thời gốc (\) trên ổ C:, viết lệnh chuyển tới thư mục BKED để làm việc trên thư mục BKED.

Trước lúc chuyển, thư mục hiện thời là thư mục gốc:\

C:\>_

C:\> CD BKED ↵

Sau lúc chuyển, thư mục hiện thời là thư mục BKED

C:\BKED>_

Chú ý: Câu lệnh CD chỉ để chuyển hoạt động tới các thư mục trên cùng ổ đĩa, muốn chuyển hoạt động tới ổ đĩa khác phải dùng lệnh chuyển ổ đĩa.

VI. LỆNH XOÁ THƯ MỤC

Để xoá thư mục ta dùng lệnh RD (Remove Directory). Điều kiện để xoá thư mục là: Thư mục cần xoá phải rỗng (không chứa tệp, thư mục con nào khác), và nó không phải là thư mục hiện thời.

Để xoá thư mục con tên là HUONG của thư mục USERS trên ổ đĩa C:\>

C:\> RD USERS\HUONG ↵

Phần đường dẫn không cần chỉ ra tên thư mục gốc, nhưng ta phải thêm tên thư mục gốc trọng đường dẫn nếu thư mục hiện thời không phải là thư mục gốc.

- Lệnh xoá thư mục HUONG khi thư mục hiện thời lại là DOS trên C:

C:\DOS> RD \USERS\HUONG ↵

- Để xoá thư mục mà trong đó có các tệp có thuộc tính ẩn, ta phải dùng DIR/a để hiển thị cả các tệp ẩn rồi dùng lệnh DEL *.* để xoá toàn bộ các tệp ở thư mục này, làm cho thư mục trở thành rỗng, lúc đó ta mới có thể xoá thư mục này được.

Chú ý: Mọi thư mục con khi tạo ra, tự động hệ thống tạo ra thư mục:

. **<DIR>**

.. **<DIR>**

và nếu thư mục con nào chỉ còn hai thư mục trên (tất nhiên không chứa tệp nào) cũng đều coi như là thư mục đã rỗng.

VII. LỆNH HIỂN THỊ CẤU TRÚC CÂY THƯ MỤC (LỆNH NGOẠI TRÚ)

Khi có tổ chức thư mục trên ổ đĩa ta có thể xem biểu diễn cây thư mục trên màn hình cùng các tệp tương ứng. Để xem cấu trúc thư mục hiện thời dùng lệnh:

TREE [/F] ↵

Tuỳ chọn /f đưa vào dòng lệnh khi cần hiển thị cả các tệp trên các thư mục tương ứng.

Ví dụ: Hiển thị màn hình cấu trúc thư mục gốc của ổ đĩa C:

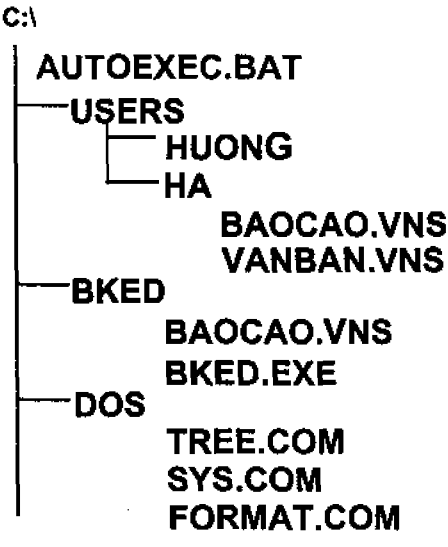
C:\> TREE ↵

Muốn xem các tệp tương ứng của từng thư mục ta thêm tuỳ chọn /f:

C:\> TREE/F↵

Hình bên hiển thị các tệp trên cùng thư mục tương ứng.

Khi có nhiều thư mục và các tệp cấu trúc sẽ trôi nhanh trên màn hình, hãy bấm phím PAUSE để dừng trôi màn hình, rồi lại ấn phím bất kỳ để tiếp tục.



Cây thư mục hiển thị trên màn hình khi dùng lệnh TREE/f

VIII. LỆNH XOÁ THƯ MỤC

Khi cần xoá thư mục cùng các tệp trong thư mục đó dùng lệnh DELTREE (lệnh ngoài).

Lưu ý: Lệnh này xoá cả các tệp ẩn, các tệp chỉ đọc, hệ thống có trong thư mục đó.

Câu lệnh xoá thư mục con từ thư mục hiện thời:

DELTREE [ổ đĩa] [đường dẫn] <tên thư mục>↵

Ví dụ: Lệnh xoá thư mục USERS trong ổ C:, giả sử thư mục hiện thời là thư mục gốc.

C:\>DELTREE USERS↵

Khi đó toàn bộ các thư mục (hình trên) USERS, HUONG, HA và các tệp trên đó đều bị xoá.

Chú ý:

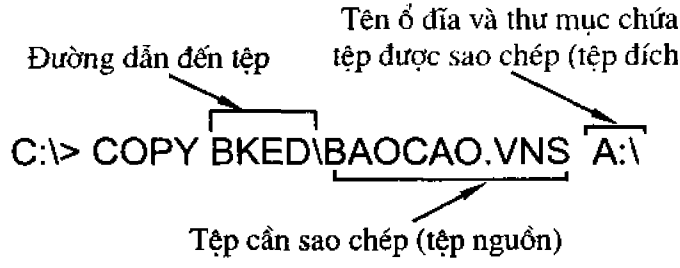
Cần thận trọng khi dùng lệnh này sau khi đã kiểm tra kỹ các thư mục và tệp cần xoá.

IX. LỆNH SAO CHÉP TỆP

1. Câu lệnh tổng quát

COPY [D1]:[path1]<file1> [D2]:[path2] [file2]

Chức năng: Sao chép tệp có tên FILE1 (đã có trên ổ đĩa D1:, theo đường dẫn PATH1) thành tệp có tên FILE2 (nội dung không thay đổi, theo đường dẫn PATH2 trên ổ D2:).



Theo quy ước phần nằm trong các dấu <, > bắt buộc phải đưa ra trong câu lệnh, phần nằm trong các dấu [,] có thể có hoặc không có trong câu lệnh thực tế, tuỳ theo mục đích sao chép của người sử dụng.

2. Ví dụ

Lệnh sau đây sao chép tệp BAOCAO.VNS trên ổ đĩa C:, trong thư mục BKED đặt vào thư mục gốc trên đĩa mềm trong ổ A:, không đổi tên tệp.

```
C:\>COPY BKED\BAOCAO.VNS A:\↵
1file(s) copied
C:\>_
```

X. LỆNH SAO CHÉP THƯ MỤC (LỆNH NGOẠI TRÚ)

Để dùng lệnh này cần có tệp XCOPY.EXE trên thư mục hiện thời. Lệnh này sẽ chép cấu trúc thư mục và các tệp có trong đó.

Câu lệnh tổng quát:

XCOPY Đường dẫn 1 Đường dẫn 2 **[/E/P/S/V]**

Đường dẫn 1: xác định đường dẫn của tên tệp hay tên thư mục nguồn cần sao chép.

Đường dẫn 2: xác định tên tệp hay thư mục đích.

Các tuỳ chọn:

/E nếu được dùng thì các thư mục con rỗng cũng được sao chép.

/S chép tất cả các thư mục con cấp thấp hơn thư mục cần sao chép.

/V kiểm tra việc sao chép các tệp lên đĩa đúng như tệp nguồn.

Có thể kết hợp các tuỳ chọn trên cùng dòng lệnh.

Ví dụ:

- Lệnh sao chép các tệp của thư mục USERS và các thư mục con của nó tới đĩa A:

C:\>XCOPY USERS A:./S↵

XI. LỆNH TẠO TỆP VĂN BẢN ĐƠN GIẢN

1. Câu lệnh

```
Copy con <tên tệp>↵
.....
.....
F6 (hoặc Ctrl-Z) ↵
```

} Phần nội dung của tệp, USER tự gõ vào.

2. Ví dụ:

Tạo tệp tên là : AUTOEXEC.BAT như sau:

C:\>COPY CON AUTOEXEC.BAT↵
@ECHO OFF↵
PROMPT PG↵
PATH=C:\DOS;C:\BKED;C:\NC;D:\TP↵
^Z↵
1file(s) copied
C:\>_

XII. LỆNH XEM NỘI DUNG TỆP VĂN BẢN

Muốn xem nội dung tệp văn bản dùng lệnh:

TYPE [ổ đĩa] [đường dẫn] <tên tệp> **[>PRN]**

Ví dụ:

- Xem nội dung tệp AUTOEXEC.BAT trên màn hình, giả sử tệp AUTOEXEC.BAT đang có trên thư mục gốc ổ đĩa C:, câu lệnh là:

C:\>TYPE AUTOEXEC.BAT ↵

- Muốn đưa tệp này ra máy in ta thêm >PRN trong dòng lệnh:

C:\>TYPE AUTOEXEC.BAT >PRN ↵

Ví dụ:

- Lệnh chép tất cả các tệp của thư mục HA sang đĩa trên ổ đĩa A:

Nếu thư mục hiện thời là gốc trên ổ C: thì câu lệnh sẽ là:

C:\>COPY USERS\HA*.* A: ↵

Nếu thư mục hiện thời là HA trên ổ C: thì câu lệnh sẽ là:

C:\USERS\HA>COPY *.* A: ↵

Nếu thư mục hiện thời là gốc A:thì câu lệnh sẽ là:

A:\>COPY C:\USERS\HA>*.* ↵

XIII. LỆNH XOÁ TỆP

1. Câu lệnh tổng quát

DEL [ổ đĩa:] [đường dẫn] <tên tệp>↵

Trong đó:

- Ổ đĩa: xác định ổ đĩa chứa tệp cần xoá.
- Đường dẫn: xác định đường dẫn đến tệp.
- Tên tệp: xác định tên tệp cần xoá.

Chú ý:

+ Các tuỳ chọn được bao trong dấu [..] có thể có hay không tuỳ theo yêu cầu, <tên tệp> bắt buộc phải có trong câu lệnh.

+ Trong dòng lệnh thực tế không có dấu [và dấu], dấu < và dấu >.

2. Ví dụ

1/ Lệnh xoá tệp BAOCAO.VNS trong thư mục con HA của thư mục USERS trên ổ đĩa C:

- Nếu thư mục hiện thời là thư mục gốc trên ổ C:

C:\>DEL USERS\HA\BAOCAO.VNS↵

- Nếu thư mục hiện thời là thư mục HA trên ổ C:

C:\ USERS\HA>DEL BAOCAO.VNS↵

- Nếu thư mục hiện thời là thư mục gốc trên ổ A:

A:\> DEL C:\ USERS\HA\ BAOCAO.VNS↵

Nhận xét trong câu lệnh này phần đường dẫn phải chỉ ra tên ổ đĩa chứa tệp cần xoá vì thư mục và ổ đĩa hiện thời không ở cùng ổ đĩa chứa tệp cần xoá.

2/ Lệnh xoá toàn bộ tệp trong thư mục con HA của thư mục USERS trên ổ đĩa C:, thư mục hiện thời HA trên ổ C:

C:\ >DEL USERS\HA*.*↵

Khi đó máy sẽ hỏi lại:

`All files in the directory will be delete, are you sure (Y/N)?`

Nghĩa là toàn bộ các tệp của thư mục HA sẽ bị xoá, có chắc không?

Trả lời bằng cách gõ phím Y↵ - lệnh đã được thực hiện

Trả lời bằng cách gõ phím N↵ - không thực hiện lệnh, trở lại dấu nhắc DOS.

3/ Lệnh xoá tất cả các tệp có đuôi .SYS trong thư mục DOS, giả sử thư mục hiện thời DOS trên ổ C:

C:\DOS>DEL *.SYS ↵

Thường dùng lệnh DIR để kiểm tra việc thực hiện lệnh.

Màn hình mô tả (dùng lệnh DIR) nội dung của thư mục DOS trước khi thực hiện lệnh xoá các tệp *.SYS như sau:

```
Volume in drive C is ITCOM
Volume Serial Number is 1B5E-11FE
Directory of C:\DOS
```

```
.          <DIR>          08-22-98  8:32p
..         <DIR>          08-22-98  8:32p
ANSI SYS            9,719  02-08-98  9:50p
ATTRIB EXE         15,252  02-08-98  9:35p
CHKDSK EXE         28,096  02-08-98  9:36p
COUNTRY SYS        30,742  02-08-98  9:50p
CSCRIPT EXE       135,168  02-09-98  9:05a
DISKCOPY COM       21,975  02-08-98  9:37p
DISPLAY SYS        17,175  02-08-98  9:50p
DRVSPACE SYS        2,135  02-08-98  9:55p
FORMAT COM         49,575  02-08-98 10:20p
KEYBOARD SYS       34,566  02-08-98  9:53p
KEYBRD2 SYS        31,942  02-08-98 10:20p
KEYBRD3 SYS        31,633  09-22-97  4:10p
KEYBRD4 SYS        13,030  02-08-98  9:53p
SYS COM            18,967  02-08-98  9:45p
       14 file(s)    439,975 bytes
       2 dir(s) 238,874,624 bytes free

C:\DOS>_
```

Màn hình mô tả (dùng lệnh DIR) nội dung của thư mục DOS sau khi thực hiện lệnh xoá các tệp *.SYS như sau:

```
Volume in drive C is ITCOM
 Volume Serial Number is 1B5E-11FE
 Directory of C:\DOS

.          <DIR>          08-22-98  8:32p
..         <DIR>          08-22-98  8:32p
ATTRIB EXE         15,252  02-08-98  9:35p
CHKDSK EXE         28,096  02-08-98  9:36p
CSCRIPT EXE       135,168  02-09-98  9:05a
DISKCOPY COM       21,975  02-08-98  9:37p
```

```
FORMAT COM          49,575   02-08-98 10:20p
SYS COM             18,967   02-08-98 9:45p
       6 file(s)  269,033 bytes
       2 dir(s)  238,874,624 bytes free

C:\DOS>_
```

XIV. LỆNH KHÔI PHỤC TỆP ĐÃ BỊ XOÁ NHẦM (LỆNH NGOẠI TRÚ)

Muốn dùng lệnh này phải có tệp lệnh UNDELETE.EXE trong thư mục DOS.

Khi xoá một tệp bằng lệnh DEL, thông tin không bị huỷ về mặt vật lý mà chỉ bị đánh dấu xoá, nếu chưa bị ghi đè lên vùng thông tin này bởi tệp thông tin khác thì vẫn có thể khôi phục lại tên tệp vừa bị đánh dấu xoá bằng lệnh:

UNDELETE [đường dẫn]<tên tệp>↵

Trong đó: đường dẫn: xác định vị trí tệp cần phục hồi; tên tệp: xác định tên tệp cần phục hồi. Ví dụ: Để khôi phục tệp BAOCAO.VNS trong thư mục BKED đã bị xoá, dùng lệnh:

- Để khôi phục tất cả các tệp đã bị xoá nhầm trong thư mục BKED, dùng lệnh:

C:\BKED>\DOS\UNDELETE ↵

Mỗi lần khôi phục một tệp, máy dừng lại bắt ta khẳng định lại có khôi phục tệp hay không (Y/N)? - gõ Y để khôi phục.

Hiện lên tên tệp cần phục hồi với chữ cái đầu là dấu ? (chẳng hạn: ?BAOCAO.VNS), ấn vào chữ cái đầu tiên của tên tệp (thay vào dấu ?) cần khôi phục và máy thông báo.

```
File successfully undelete.
```

Sau khi ra lệnh phục hồi tệp BAOCAO.VNS, màn hình như sau:

```
UNDELETE - A delete protection facility
Copyright (C) 1987-1993 Central Point Sorfware, Inc.
All rights reserved.
Directory: C:\BKED
File Speccifications: BAOCAO.VNS
   Delete Sentry control file not found.
   Deletion-tracking file not found.
   MS-DOS directory contains 1 deleted files.
   Of those, 1 files may be recovered.
Using the MS-DOS directory method.
?AOCAO  VNS  2028  12-02-1993  10:13p ...A Undelete (Y/N)?Y
Please type the first character for ?BAOCAO .VNS: B
File successfully undeleted.

C:\>_
```

XV. LỆNH ĐỔI TÊN TỆP

REN viết tắt của chữ RENAME (đổi lại tên). Ta có thể đổi lại tên của 1 hoặc 1 nhóm tên của các tệp bằng lệnh:

REN [ổ đĩa] [đường dẫn] <tên tệp 1> <tên tệp 2>

Ổ đĩa, đường dẫn: xác định đĩa, thư mục chứa tệp cần đổi tên.

Tên tệp 1: tên cũ

Tên tệp 2: tên mới

Ví dụ: Lệnh đổi tất các tên tệp trong thư mục BKED có đuôi .VNS thành các tệp có đuôi .TXT:

C:\BKED>REN *.VNS *.TXT

XVI. LỆNH TẠO KHUÔN ĐĨA MỀM (LỆNH NGOẠI TRÚ)

Việc tạo khuôn (FORMAT) đĩa mềm dùng trong hai trường hợp:

- Đĩa mới cần tạo khuôn trước khi ghi thông tin lên đó.
- Đĩa đã có thông tin (cũ), tạo khuôn sẽ xoá hết thông tin cũ và sẵn sàng cho phép ghi thông tin khác lên đĩa.

Để tạo khuôn đĩa, phải có tệp FORMAT.COM trên thư mục DOS.

Câu lệnh tổng quát:

FORMAT <ổ đĩa mềm:> [/S][/V][/U][/Q][/4]

- Ổ đĩa mềm: xác định tên ổ đĩa chứa đĩa cần tạo khuôn.

- Tuỳ chọn /S: ghi các tệp hệ thống DOS lên đĩa cần tạo khuôn, với tuỳ chọn này, sau khi tạo khuôn, đĩa khởi tạo được máy - gọi là đĩa hệ thống DOS.

- Tuỳ chọn /V: cho phép đặt tên nhãn đĩa..

- Tuỳ chọn /4: khi tạo khuôn đĩa 360KB trên ổ đĩa kiểu 1,2MB.

- Tuỳ chọn /U: để xoá sạch thông tin trên đĩa mà không thể khôi phục lại được bằng lệnh UNFORMAT.

- Tuỳ chọn /Q: format nhanh, có tuỳ chọn này hệ thống sẽ không kiểm tra lỗi trên bề mặt đĩa.

Quá trình tạo khuôn đĩa được tiến hành từng bước theo sự nhắc của máy trên màn hình, USER phải trả lời tuỳ theo yêu cầu đặt ra.

Ví dụ 1:

- Lệnh tạo khuôn đĩa trên ổ đĩa A:, thư mục hiện thời là DOS trên ổ C:, có tuỳ chọn /S (tạo đĩa hệ thống).

Quá trình FORMAT theo các bước trên màn hình như sau:

```
C:\DOS>format a:/s
Insert new diskette for drive A:
and press ENTER when ready...

Checking existing disk format.
Saving UNFORMAT information.
Verifying 1.44M
Format complete.
System transferred

Volume label (11 characters, ENTER for none)? MS-dos

    1,457,664 bytes total disk space
      393,728 bytes used by system
    2,560 bytes in bad sectors
    1,061,376 bytes available on disk
```

```
        512 bytes in each allocation unit.
      2,073 allocation units available on disk.

Volume Serial Number is 405B-13FC

QuickFormat another (Y/N)?_
```

+ **Bước 1**: Gõ dòng lệnh từ thư mục DOS trên ổ C:\DOS>format a:/s↵

+ **Bước 2**: Xuất hiện 2 dòng:

```
Insert new diskette for drive A:
and press ENTER when ready...
```

Tức là: Đưa đĩa mới vào ổ A:, đóng khoá ổ đĩa

Bấm ENTER (↵) khi đã sẵn sàng . . .

+ **Bước 3**: Máy bắt đầu kiểm tra cấu trúc, chất lượng đĩa xuất hiện dòng chữ:

```
Checking existing disk format.
Saving UNFORMAT information.
```

Máy phát hiện thấy đĩa (cũ) chứa thông tin, ghi lại để cần thiết sẽ phục hồi bằng lệnh UNFORMAT.

Tiến hành việc FORMAT đĩa và chuyển các tệp hệ thống DOS vào đĩa sau khi đã tiến hành FORMAT xong, dòng chữ thông báo theo các đoạn:

```
Verifying 1.44M
Format complete.
System transferred
```

USER phải trả lời dòng thông báo sau: cho nhãn đĩa (cho phép đến 11 ký tự), hoặc nếu không thì bấm phím ENTER.

```
Volume label 11 characters, ENTER for none)?_
```

+ **Bước 4:** Máy thông báo kết quả và hỏi có tạo khuôn cho đĩa khác hay không (Y/N)?

- Bấm phím Y↵ để tạo khuôn đĩa khác.
- Bấm phím N↵ để trở về thư mục hiện thời.

Nhận xét:

- Lệnh tạo khuôn sẽ xoá hết thông tin trên đĩa cần tạo khuôn, nếu đĩa đưa vào là đĩa chứa thông tin, cần thận trọng khi tạo khuôn đĩa, việc tạo khuôn cho đĩa mềm trên ổ B: cũng tương tự các bước như trên.

- Tên ổ đĩa phải luôn có dấu ': ' (hai chấm)

Ví dụ 2:

- Lệnh tạo khuôn đĩa 360KB trên ổ đĩa kiểu 1,2M, không tạo đĩa hệ thống:

C:\DOS>FORMAT A/4↵

Ví dụ 3:

- Lệnh tạo khuôn đĩa trên ổ đĩa B:, không tạo đĩa hệ thống:

C:\DOS>FORMAT B:↵

XVII. LỆNH SAO CHÉP ĐĨA MỀM (LỆNH NGOẠI TRÚ)

Muốn sao nguyên bản đĩa mềm dùng lệnh:

DISKCOPY [ổ đĩa nguồn:][ổ đĩa đích:][/V]

Ổ đĩa nguồn: Xác định tên ổ đĩa chứa thông tin cần sao chép.

Ổ đĩa đích: Xác định tên ổ đĩa cần sao chép thông tin vào đĩa trên đó.

/V : Kiểm tra việc sao chép có chính xác không.

Điều kiện để thực hiện lệnh này:

- Có tệp DISKCOPY.COM trên đĩa (chẳng hạn để trên thư mục DOS).
- Đĩa và ổ đĩa nguồn và đích cần phải cùng kiểu và dung lượng.

Thường việc sao chép này tiến hành trên một ổ đĩa, khi đó máy sẽ nhắc USER lần lượt đưa đĩa nguồn, đĩa đích vào từng thời điểm thích hợp.

Ví dụ: Lệnh chép đĩa trên ổ đĩa A:, kiểu đĩa và ổ đĩa: 1,2M:

C:\DOS>DISKCOPY A: A: ↵

Lưu ý: phần tên lệnh (DISKCOPY) và các tên ổ đĩa phải để 1 khoảng trống.

Quá trình diễn ra từng bước:

+ Hãy cho đĩa cần sao chép vào ổ đĩa A: và lặp khoảng từ 2 đến 4 lần như sau:

+ Hãy cho đĩa nguồn (SOURCE) vào ổ đĩa rồi bấm phím bất kỳ để tiếp.

+ Hãy cho đĩa đích (TARGET) vào ổ đĩa rồi bấm phím bất kỳ để tiếp.

Cuối cùng sau khi đã sao đĩa xong máy sẽ hỏi:

+ Có sao chép đĩa khác hay không (Y/N)?

Màn hình mô tả từng bước: hỏi và trả lời của USER như dưới đây:

```
C:\DOS>diskcopy a: a:
Insert SOURCE diskette in drive A:
```

```
Press any key to continue...
Copying 80 track, 2 side (s)
Insert TARGET diskette in drive A:
Press any key to continue...
Insert SOURCE diskette in drive A:
Press any key to continue...
Insert TARGET diskette in drive A:
Press any key to continue...
Volume Serial Numer is 13E6-3881
Copy anather diskette (Y/N)?n
C:\DOS>_
```

Dòng chữ thông báo:

```
Insert SOURCE diskette in drive A:
Press any key to continue...
```

Nghĩa là: hãy đưa đĩa **NGUỒN** vào ổ đĩa A:, bấm phím bất kỳ để tiếp tục.

```
Insert TARGET diskette in drive A:
Press any key to continue...
```

Nghĩa là: hãy đưa đĩa **ĐÍCH** vào ổ đĩa A:, bấm phím bất kỳ để tiếp tục.

```
Copy anather diskette (Y/N)?n
C:\DOS>_
```

Có sao chép đĩa khác hay không? Trả lời N (không), trở lại thư mục DOS.

XVIII. LỆNH XOÁ MÀN HÌNH

Lệnh xoá màn hình: **CLS** sẽ xoá màn hình, rồi đưa dấu nhắc của máy lên góc phía trên, bên trái màn hình.

CLS↵

Ví dụ: Màn hình trước lúc ra lệnh xoá màn hình:

```
C:\DOS>dir

 Volume in drive C is ITCOM
 Volume Serial Number is 1B5E-11FE
 Directory of C:\DOS
```

```
.            <DIR>          08-22-98  8:32p
..           <DIR>          08-22-98  8:32p
FORMAT COM        49,575    02-08-98 10:20p
ATTRIB EXE        15,252    02-08-98  9:35p
IEXTRACT EXE      17,655    02-09-98  8:20a
CHKDSK EXE        28,096    02-08-98  9:36p
KEYB COM          19,927    02-08-98  9:40p
KEYBOARD SYS      34,566    02-08-98  9:53p
KEYBRD2 SYS       31,942    02-08-98 10:20p
KEYBRD3 SYS       31,633    09-22-97  4:10p
KEYBRD4 SYS       13,030    02-08-98  9:53p
DISKCOPY COM      21,975    02-08-98  9:37p
SYS COM           18,967    02-08-98  9:45p
             12 file(s)    417,786 bytes
             2 dir(s)  240,693,248 bytes free
```

C:\DOS>CLS↵

Màn hình sau khi ra lệnh xoá màn hình chỉ còn lại dấu nhắc hệ thống:

C:\DOS>_

XIX. LỆNH XEM VÀ THIẾT LẬP THỜI GIAN CHO MÁY

Câu lệnh:

C:\>TIME.↵

Khi đó màn hình sẽ là:

```
C:\>time↵
Current time is 5:19:21.58p
Enter new time:_
```

Nếu thời gian đúng theo hiện thời ta chỉ bấm ENTER(↵).

Nếu sai ta hãy cho bấm vào thời gian theo trình tự:

Giờ (hai chữ số) : phút (hai chữ số) : giây (hai chữ số) p: chiều, a: sáng

Ví dụ: Ta thiết lập thời gian cho máy là: 02:35:10p.

Khi đó màn hình sẽ là:

```
C:\>time↵
Current time is 5:19:21.58p
Enter new time:02:35:10p

C:\>_
```

XX. LỆNH XEM VÀ THIẾT LẬP NGÀY, THÁNG, NĂM CHO MÁY

Câu lệnh:

C:\>DATE↵

Khi đó màn hình sẽ là:

```
C:\>DATE↵
Current date is Sun 08-23-1998
Enter new date (mm-dd-yy):_
```

Nếu thời gian ngày, tháng, năm đúng theo hiện thời ta chỉ bấm Enter (↵).

Nếu sai ta hãy cho bấm vào tháng, ngày, năm theo trình tự:

tháng (mm) (hai chữ số)- ngày (dd) (hai chữ số)- năm (yy) (hai chữ số)

Ví dụ: Ta thiết lập lại tháng 07, ngày 30, năm 98 cho máy là: 07-30-98

Khi đó màn hình sẽ là:

```
C:\>DATE↵
Current date is Sun 08-23-1998
Enter new date (mm-dd-yy):07-30-98↵
C:\>_
```

Câu hỏi ôn tập

1. Thực hành các lệnh cơ bản của DOS để xử lý: thư mục, tệp, nhóm tệp?
2. Dùng dấu nhắc hệ thống DOS và đường dẫn thư mục hiện thời có lợi gì? Thử bỏ đường dẫn thư mục trong dấu nhắc hệ điều hành? (lệnh PROMPT không tuỳ chọn) - nhận xét?
3. Có thể dùng lệnh CD để chuyển tới thư mục trên ổ đĩa khác được không?

Chương 7

TỆP BATCH VÀ TỆP CẤU HÌNH

I. TỆP BATCH

Các lệnh của MS-DOS thường được cho thực hiện từng lệnh, từ dấu nhắc của Hệ điều hành MS-DOS, ta có thể gộp (bó) các lệnh đó lại thành một tệp, quy ước đặt tên có phần đuôi tệp là .BAT và chỉ cần cho thực hiện tệp BATCH này thì các lệnh liệt kê trong tệp BATCH sẽ lần lượt thực hiện.

Tệp với cách gộp như vậy gọi là tệp BATCH - tệp được xử lý tuần tự theo chuỗi các lệnh liệt kê trong nội dung của tệp BATCH.

Ví dụ:

Nội dung của tệp KHOIDAU.BAT gồm các lệnh sau:

```
@Echo off
Path=C:\NC;C:TP;C:\BKED;C:\WINDOWS;C:\DOS;C:\
CD ATV
SCAN C:
@Echo on
Echo Chuc ban mot ngay lam viec tot dep
```

Giải thích:

Tệp này được thực hiện bằng cách từ dấu nhắc hệ thống ta gõ tên tệp (không nhất thiết phải gõ phần đuôi tệp .BAT) và ENTER (↵).

C:\> KHOIDAU↵

Khi đó lần lượt các lệnh thực hiện như sau:

- Lệnh : @Echo off - không hiện lên màn hình các dòng lệnh trong tệp.
- Lệnh : Path=C:\NC;C:TP;C:\BKED;C:\WINDOWS;C:\DOS;C:\

Là lưu các đường dẫn tới các thư mục NC, TP, BKED, WINDOWS, DOS, C:\ vào bộ nhớ để bất cứ lúc nào, thư mục nào ta cũng thực hiện các tệp lệnh (có đuôi tệp: .COM, .EXE, .BAT) trong các thư mục đã được chỉ dẫn này.

- Lệnh CD ATV:

Là chuyển đến thư mục ATV (vì tệp lệnh kiểm tra virus SCAN.EXE đặt trong thư mục ATV) rồi thực hiện tệp lệnh SCAN C: (kiểm tra virus ổ đĩa C:)

- Lệnh @ECHO ON:

Là cho hiển thị các chú thích, dòng lệnh tiếp theo của tệp KHOIDAU lên màn hình.

- Lệnh Echo Chuc ban mot ngay lam viec tot dep:

Là đưa dòng chữ:' Chuc ban mot ngay lam viec tot dep' lên màn hình.

II. TỆP AUTOEXEC.BAT

Tệp này bản thân tên gọi nó mang ý nghĩa việc thực hiện nó. Khi khởi tạo máy xong, HĐH MS-DOS tìm trong thư mục gốc của ổ đĩa chứa các tệp cấu hình hệ thống và thực hiện ngay tệp AUTOEXEC.BAT. Sau đó đưa về dấu nhắc hệ thống.

Trong tệp AUTOEXEC.BAT thường liệt kê các lệnh:

- Tạo dấu nhắc PROMPT PG.
- Lệnh thông báo trước các đường dẫn thư mục: PATH=...
- Lệnh kiểm tra đĩa, kiểm tra virus, thường trú các lệnh trong RAM.
- Lệnh thông báo màn hình.

Tệp này do USER tự tạo bằng lệnh COPY CON AUTOEXEC.BAT và đưa vào một số lệnh (thường dùng) như sau:

Ví dụ:

Tệp AUTOEXEC.BAT có nội dung:

```
@Echo off
Path=C:\NC;C:TP;C:\BKED;C:\WINDOWS;C:\DOS;C:\
CD ATV
SCAN C:
@Echo on
Echo Chuc ban mot ngay lam viec tot dep
```

III. TỆP CẤU HÌNH CONFIG.SYS

Tệp cấu hình có tên gọi quy định: CONFIS.SYS, tệp này tự động được thực hiện ngay sau khi nạp ROMBIOS, sau đó mới đến tệp AUTOEXEC.BAT.

Tệp CONFIS.SYS chứa các lệnh tạo cấu hình, thiết lập các thông số cần thiết cho các thiết bị máy tính, chẳng hạn như: chọn kiểu quản lý vùng nhớ của bộ nhớ trong RAM, chọn kiểu bàn phím, thiết lập cổng truyền dữ liệu sang máy in, chuột v.v. giúp cho hệ thống và các chương trình ứng dụng có thể sử dụng các thiết bị này thuận lợi.

Nếu các lệnh thiết lập cấu hình cho các thiết bị không kể ra, máy tự động chọn các thông số thiết bị ngầm định của máy.

Tệp này thường tạo bằng lệnh tạo tệp văn bản đơn giản:

COPY CON CONFIG.SYS, tệp này phải được để tại thư mục gốc của ổ đĩa khởi động MSDOS (trên đĩa A:\ hoặc C:\)

Ví dụ:

- Tệp cấu hình CONFIS.SYS

```
FILES= 45
BUFFERS=30,0
DEVICE=C:\DOS\HIMEM.SYS
DOS=HIGH,UMB
DEVICE=C:\MOUSE\MOUSE.SYS
```

Giải thích:

- Lệnh FILES=45 - cho phép khai báo số tệp mở để xử lý đồng thời: 45 tệp.

- Lệnh BUFFERS=30,0 - tạo vùng nhớ đệm trung gian trong bộ nhớ trong để ghi, đọc tệp trên đĩa. Vùng nhớ đệm càng lớn thì tốc độ ghi/đọc đĩa càng nhanh, nhưng bộ nhớ trong (RAM) dùng để cho các phép xử lý khác bị giảm đi. Lệnh này thực tế đã khai báo vùng đệm =30 buffers, mỗi buffers=528 bytes.

- Lệnh DEVICE=C:\DOS\HIMEM.SYS - sử dụng chương trình HIMEM để quản lý vùng nhớ mở rộng.

- Lệnh DOS=HIGH,UMB - nạp HĐH DOS lên bộ nhớ cao để giành không gian bộ nhớ quy ước (640KB) dùng cho việc xử lý các chương trình ứng dụng.

- Lệnh DEVICE=C:\MOUSE\MOUSE.SYS - nạp chương trình điều khiển chuột trong thư mục C:\MOUSE.

IV. MỘT SỐ LỆNH TRONG CÁC TỆP BATCH VÀ TỆP CẤU HÌNH

1. Lệnh REM (Renmark)

Công dụng: Không cho thực hiện câu lệnh được viết sau REM trong tệp BATCH và tệp cấu hình CONFIG khi thực hiện tệp.

2. Lệnh PAUSE

Công dụng: Để hiện thông báo, tạm dừng thực hiện chờ USER bấm phím trả lời, rồi thực hiện tiếp lệnh sau.

Câu lệnh:

PAUSE <thông báo>

Tệp này thường tạo bằng lệnh tạo tệp văn bản đơn giản:

COPY CON CONFIG.SYS, tệp này phải được để tại thư mục gốc của ổ đĩa khởi động MSDOS (trên đĩa A:\ hoặc C:\)

Ví dụ:

Tệp TAOKHUON.BAT có nội dung sau:

1: REM chuong trinh tao khuon dia moi

2: REM ten chuong trinh la TAOKHUON.BAT

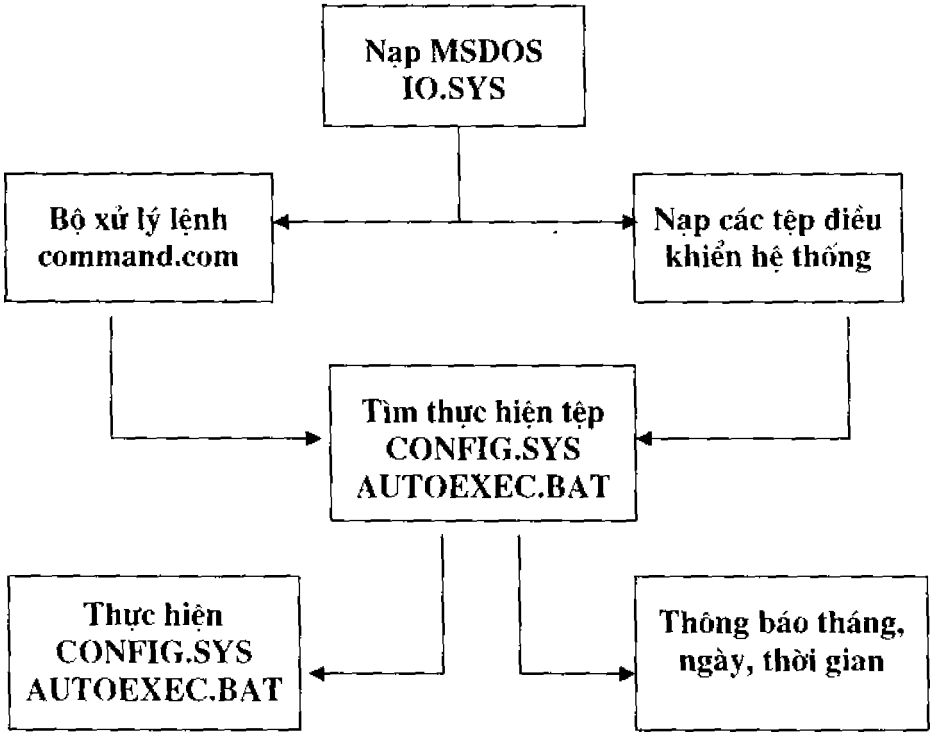
3: PAUSE dua dia moi vao o dia A:

4: C:\DOS\FORMAT A:

Khi thực hiện tệp lệnh này ta chỉ cần gõ tên tệp rồi bấm ENTER, không cần gõ phần đuôi tệp .BAT, các lệnh từ 1 đến 4 tuần tự thực hiện, các thông báo xuất hiện trên màn hình nhắc USER bấm phím trả lời.

C:\>TAOKHUON↵

Quá trình khởi tạo máy - nạp DOS, tìm thực hiện tệp CONFIG.SYS, AUTOEXEC.BAT, đưa dấu nhắc hệ thống MS-DOS:



3. Lệnh ẩn, hiện các dòng lệnh BATCH trên màn hình

Công dụng: Hiện và ẩn dòng lệnh trong tệp BATCH (khi thực hiện tệp BATCH) lên màn hình.

Câu lệnh:

ECHO ON/OFF [chú thích]

Giải thích: - Khi đặt ECHO ON thì các dòng lệnh thực hiện của tệp .BATCH sẽ hiện lên màn hình.

- Khi đặt ECHO OFF thì các lệnh của tệp BATCH không hiện trên màn hình.

4. Lệnh thiết lập chế độ ngắt

Câu lệnh:

Break = ON/OFF

Nếu có câu lệnh BREAK = ON trong tệp lệnh CONFIG.SYS thì khi chương trình nào đó đang thực hiện, muốn dừng thực hiện ta chỉ cần bấm phím Ctrl+C hoặc bấm Ctrl+Break.

5. Lệnh thiết lập biến môi trường

Công dụng: Tạo biến môi trường mà các chương trình có thể sử dụng.

Câu lệnh:

SET thư mục = ổ đĩa:\thư mục

Ví dụ:

SET TEMP = C:\TEMP

Giải thích: Tạo biến môi trường có tên là TEMP và xác lập bằng thư mục C:\TEMP. Tên ta khai báo là tên thư mục đang tồn tại. Các chương trình DOS, WINDOWS thường dùng biến này để lưu các tệp thông tin tạm thời.

6. Lệnh dùng để nạp các chương trình điều khiển thiết bị

Lệnh để nạp các chương trình điều khiển thiết bị, lệnh này thường dùng trong tệp cấu hình CONFIG.SYS.

Câu lệnh:

DEVICE = <tên chương trình điều khiển>

6.1. Ví dụ dùng lệnh đặt trong tệp CONFIG.SYS:

Nạp chương trình điều khiển chuột:

DEVICE = C:\MOUSE\MOUSE.SYS

- Nạp chương trình điều khiển tạo ổ đĩa ảo 200KB:

DEVICE = C:\DOS\RAMDRIVE.SYS 200/E

- Nạp chương trình điều khiển, quản lý vùng nhớ mở rộng:

DEVICE = C:\DOS\HIMEM.SYS

6.2. Ví dụ dùng tham số thay thế của tệp BATCH:

Các tham số thay thế của tệp BATCH có tên %0, %1, %2,... , %9.

Tạo tệp sao chép tệp, in tệp ra máy in, hiển thị nội dung tệp vừa thực hiện.

Từ dấu nhắc DOS ta tiến hành từng bước:

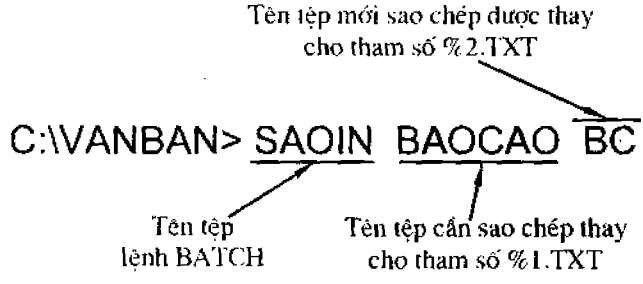
```
C:\>COPY CON SAOIN.BAT↵
   COPY %1.TXT %2.TXT↵
   TYPE %2.TXT>PRN↵
   TYPE %0.BAT↵
   Ctrl+Z↵
1 file(s) copied
```

Khi đó ta đã có 1 tệp SAOIN.BAT, muốn thực hiện tệp này ta chỉ gõ tên tệp, không cần gõ phần đuôi .BAT và từng tên tệp thay vào các tham số %1 và %2, lúc đó máy sẽ in tệp thay bởi %2 và hiện tệp SAOIN.BAT lên màn hình.

Giả sử thư mục VANBAN chứa tệp BAOCAO.TXT, khi đó ta cần sao chép và in ra máy in (máy in phải bật và nạp giấy chờ in), rồi hiển thị tệp SAOIN.BAT lên màn hình.

Tất cả các việc trên chỉ phải thực hiện SAOIN như sau:

Sau khi thực hiện lệnh này trên thư mục VANBAN có hai tệp, và tệp mới tên là: BC.TXT và đã in ra máy in nội dung tệp này.



Câu hỏi ôn tập

1. Nêu các bước từ lúc khởi tạo nạp DOS đến khi thực hiện trên màn hình có thông báo thế nào?

   Trường hợp a/

   + Không có tệp cấu hình CONFIG.SYS
   + Không có tệp AUTOEXEC.BAT

   Trường hợp b/

   + Có tệp cấu hình CONFIG.SYS
   + Có tệp AUTOEXEC.BAT

2. Tạo tệp tên CHUYEN.BAT làm các chức năng sau:

   a. Chép 1 tệp có sẵn trên A: sang đĩa C:

   b. Xoá tệp này trên đĩa A: (tệp vừa chép sang C:)

   c.Hiển thị danh sách các tên tệp, thư mục trên A:

Phần II

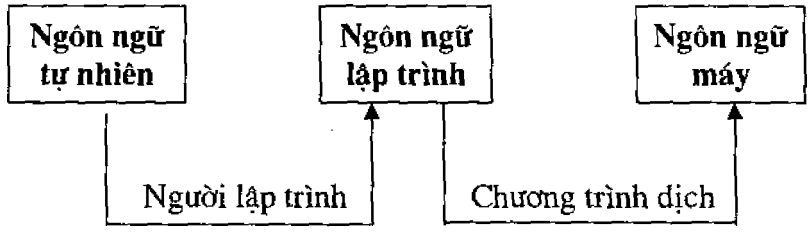
NGÔN NGỮ LẬP TRÌNH PASCAL

Chương 1

HỆ THỐNG, MÔI TRƯỜNG TURBO PASCAL

I. KHÁI NIỆM VỀ NGÔN NGỮ LẬP TRÌNH TURBO PASCAL

Ngôn ngữ lập trình là dãy các câu lệnh viết theo cú pháp nhất định nhằm ra lệnh cho máy thực hiện ý đồ của thuật toán. Có nhiều loại ngôn ngữ lập trình chẳng hạn như BASIC, FORTRAN, PASCAL, C... Mỗi ngôn ngữ lập trình có quy tắc viết khác nhau và có hệ thống chương trình dịch để chuyển đổi sang ngôn ngữ máy để máy thực hiện.



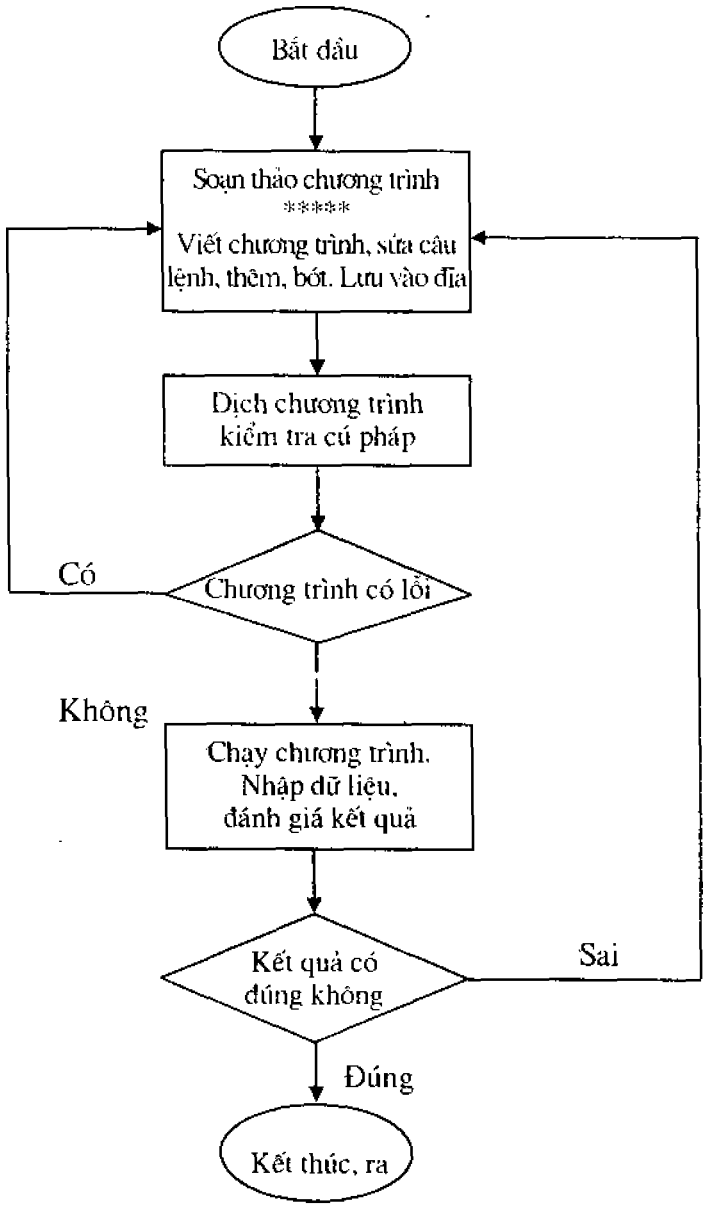
II. QUÁ TRÌNH THỰC HIỆN MỘT CHƯƠNG TRÌNH PASCAL

Các bước cơ bản cần phải tiến hành khi lập trình trên máy tính:

Bước 1 *(Soạn thảo chương trình)*: Dùng hệ soạn thảo để soạn thảo (viết) chương trình. Chương trình này gọi là chương trình nguồn, nó được lưu trên một file với phần mở rộng *.PAS (xem lại quy cách tên của file). Chương trình nguồn có thể viết mới từ đầu, có thể sửa đổi.

Bước 2 *(Dịch chương trình)*: Gọi chương trình dịch của TURBO PASCAL dịch chương trình nguồn ra mã máy, phần chương trình đã được dịch sẽ để trong bộ nhớ trong hoặc ghi ra file có phần mở rộng *.EXE *.COM. Quá trình dịch hệ thống sẽ phát hiện các lỗi sai về cú pháp như viết sai tên, thiếu các dấu... Khi phát hiện các lỗi sai ta quay về soạn thảo chương trình để sửa.

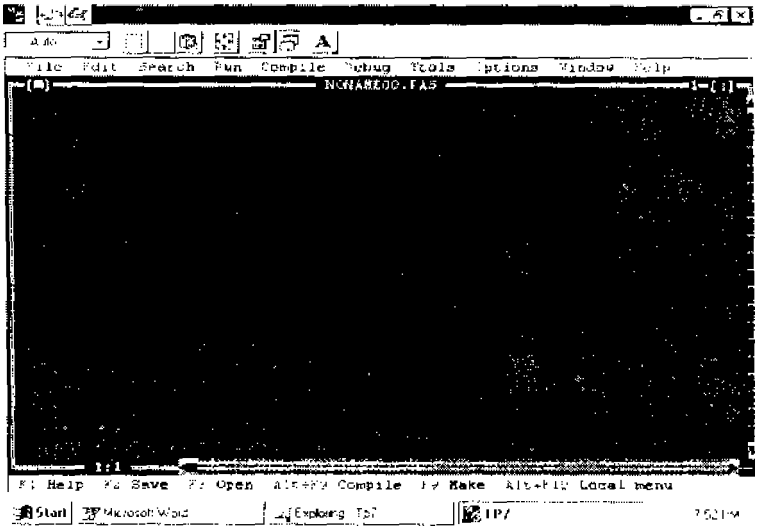
Bước 3 *(Chạy chương trình đánh giá kết quả)*: Khi chương trình chạy, thông thường ta sẽ đưa dữ liệu vào và hiển thị kết quả hay thông báo trên màn hình. Ta tự đánh giá kết quả. Nếu phát hiện kết quả không đúng với ý nghĩa thực tế ta quay lại soạn thảo để sửa lại chương trình nguồn cho phù hợp.



Lưu đồ quá trình lập trình trên TURBO PASCAL

III. CÁCH SỬ DỤNG TURBO PASCAL

1. Khởi động



Để sử dụng được Turbo Passcal chúng ta phải có tối thiểu ba tệp tin trên đĩa:

TURBO. EXE

TURBO.TP

TURBO.TPL

Lúc đó ta có thể bắt đầu làm việc với TURBO PASCAL. Để gọi chương trình TURBO PASCAL ta sử dụng lệnh gọi tệp TURBO.EXE như sau:

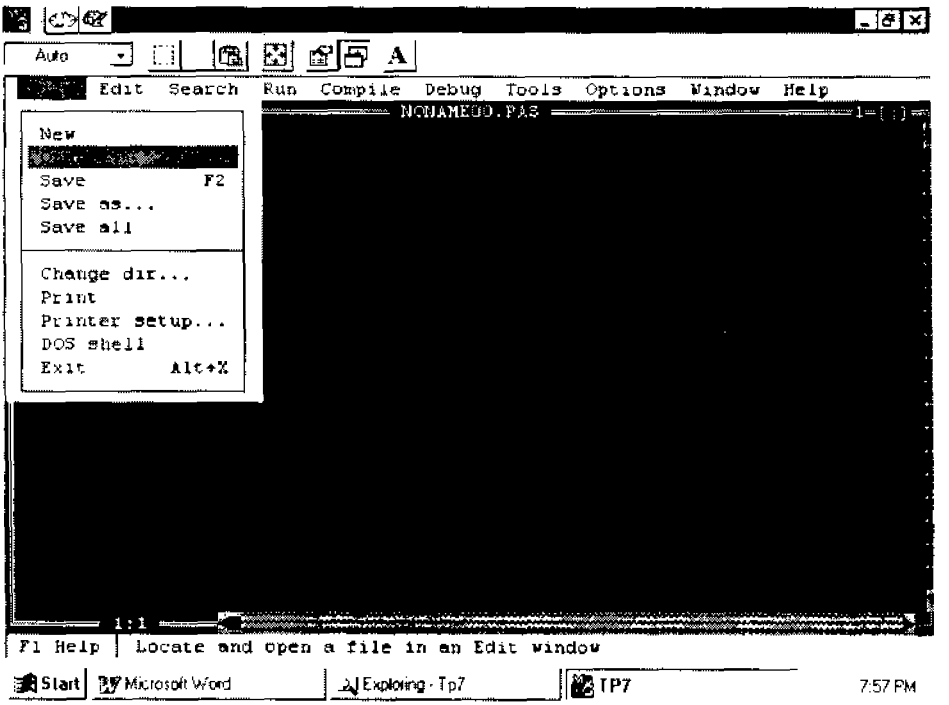
TURBO↵

Sau khi khởi động, ta thu được màn hình như trên:

Dòng trên cùng là tên các chức năng của thực đơn chính. Mỗi chức năng có thể là một lệnh hay lại là một thực đơn có nhiều chức năng tiếp theo khác.

Trên màn hình ta thấy xuất hiện các chữ cái đầu tiên của các chức năng nổi bật hơn các chữ cái khác của từ, các chữ cái nổi bật này gọi là ký tự đại diện bởi vì nếu ta gõ một trong các ký tự đại diện đó thì:

- Hoặc một lệnh được thực hiện (E chẳng hạn).
- Hoặc một thực đơn khác sẽ hiện ra (xem hình vẽ), lúc đó ta lại có thể chọn một chức năng nào đó của thực đơn hiện thời cho đến khi một lệnh được thực hiện.



Ta quy ước gọi thực đơn hiện thời là thực đơn cuối cùng xuất hiện trên màn hình (nó chứa con trỏ màn hình). Ta cũng quy ước tiếp nếu ta viết dãy chữ cái A/B điều này có nghĩa là ta chọn chức năng A của thực đơn chính sau đó chọn tiếp chức năng B của thực đơn xuất hiện sau.

Trong cửa sổ soạn thảo có dòng trạng thái:

Line 1 Col 1 Insert Indent Unident E:CUONG

Phía dưới của màn hình soạn thảo là cửa sổ thông báo.

Dòng cuối cùng có dạng:

F1-Help F5-Zoom F6-Switch F7-Trace F8-Step F9-Make F10-Menu

Dòng này miêu tả các phím chức năng.

Tiếp đây ta sẽ đề cập đến một số lệnh cơ bản:

• Lệnh **Edit**

Lệnh này khởi động trình soạn thảo văn bản, giúp chúng ta thảo văn bản của chương trình nguồn.

• Lệnh **Run**

Dùng để thực hiện chương trình có sẵn trong bộ nhớ.

• Lệnh **Compile**

Dùng để gọi thực hiện dịch chương trình. Ta sử dụng lệnh này khi muốn biên dịch chương trình trong cửa sổ soạn thảo.

Trình biên dịch của TURBO PASCAL dịch rất nhanh (27000 lệnh/phút); trong khi dịch trên màn hình xuất hiện một cửa sổ mà trong đó ta có thể quan sát thấy số lượng lệnh đã dịch. Nếu có một lỗi xảy ra, quá trình dịch sẽ dừng lại, mã hiệu của lỗi sẽ xuất hiện. Lúc đó ta gõ phím Esc cửa sổ soạn thảo sẽ hiện ra, con trỏ màn hình xuất hiện ngay sau vị trí của lỗi.

2. Trình soạn thảo văn bản

Trình soạn thảo văn bản cài trong hệ thống được thiết kế để phục vụ cho việc tạo ra các chương trình nguồn. Cách sử dụng như sau:

Giả sử chúng ta đang ở thực đơn chính, ta có hai cách gọi chương trình soạn thảo văn bản:

1. Đơn giản nhất gõ tổ hợp phím Alt-E, con trỏ màn hình sẽ xuất hiện ở cửa sổ soạn thảo, lúc này ta có thể đưa văn bản của chương trình vào máy thông qua bàn phím giống như gõ máy chữ.

2. Cách thứ hai ta gõ Alt-F, trên màn hình xuất hiện thực đơn con của menu FILE, ta gọi chức năng Load nhằm nạp một tệp vào cửa sổ soạn thảo.

Thao tác này có thể thực hiện như sau: hoặc chúng ta nạp tệp này bằng cách gõ tên (kể cả đường dẫn nếu cần) của nó vào hộp văn bản ***Load File Name***, rồi gõ phím ***Enter***, khi ấy nếu tệp này có sẵn rồi thì trang đầu tiên của nó sẽ hiện lên trong cửa sổ soạn thảo, còn nếu nó chưa tồn tại thì con trỏ màn hình sẽ xuất hiện ở vị trí đầu tiên trong cửa sổ soạn thảo, lúc đó ta sẽ bắt đầu đưa văn bản vào máy thông qua bàn phím. Trong thực đơn File ta còn thấy một số lệnh hay dùng sau (xem hình minh họa trang 74):

• **New** : được dùng khi ta muốn soạn thảo một tệp mới.

• **Save**: ghi tệp đang trong cửa sổ soạn thảo lên đĩa.

• **DOS shell**: được sử dụng khi ta muốn tạm thời quay về DOS mà không muốn thoát khỏi PASCAL.

• **Exit** : được sử dụng khi ta muốn thoát khỏi TURBO PASCAL.

Các lệnh biên tập hay được dùng có thể chia thành một số nhóm như sau:

- Các lệnh dịch chuyển con trỏ màn hình:

Để dịch chuyển con trỏ màn hình ta có thể sử dụng các tổ hợp phím hoặc phím sau:

Ctrl-E (↑) để dịch chuyển con trỏ lên dòng bên trên.

Ctrl-X (↓) để dịch chuyển con trỏ xuống dòng bên dưới.

Ctrl-S (←) để dịch chuyển con trỏ sang trái một ký tự.

Ctrl-D (→) để dịch chuyển con trỏ sang phải một ký tự.

Home để đưa con trỏ về đầu dòng.

End để đưa con trỏ về cuối dòng.

Page Up đưa con trỏ lên một trang màn hình.

Page Down đưa con trỏ xuống một trang màn hình.

- Các lệnh chèn, xoá:

Khi gọi trình soạn thảo bao giờ chúng ta cũng ở chế độ chèn (**Insert**). Chế độ này cho phép đặt một đoạn văn vào một văn bản đã có từ trước. Ở chế độ này mỗi lần chúng ta đưa vào một ký tự mới toàn bộ phần còn lại của văn bản kể từ vị trí của con trỏ sẽ dịch sang phải một vị trí để nhường chỗ cho ký tự vừa đưa vào.

Chế độ ghi đè (**Overwrite**) được sử dụng khi ta muốn thay một đoạn văn bản cũ bằng một nội dung mới. Khi đó một ký tự đưa vào sẽ thay thế ký tự hiện có ở vị trí con trỏ.

Chúng ta có thể chuyển đổi giữa hai chế độ này bằng cách gõ vào phím **INSERT** (hay **Ctrl-V**).

Sau đây là một số phím và tổ hợp phím dùng để xoá:

- **BackSpace**: để xoá một ký tự bên trái con trỏ.
- **Del**: để xoá một ký tự bên trên con trỏ.
- **Ctrl-T**: để xoá một từ bên phải con trỏ.
- **Ctrl-Y**: để xoá một dòng đang chứa con trỏ.

- Các lệnh về khối:

Ctrl-K-B: đánh dấu đầu khối.

Ctrl-K-K: đánh dấu cuối khối.

Ctrl-K-H: bỏ đánh dấu khối.

Ctrl-K-C: sao chép khối.

Ctrl-K-V: di chuyển khối.

Ctrl-K-Y: xoá khối.

Chương 2

CÁC PHẦN TỬ CƠ BẢN CỦA NGÔN NGỮ LẬP TRÌNH PASCAL

I. CÁC KÝ HIỆU CƠ BẢN

Cũng giống như các ngôn ngữ khác, TURBO PASCAL có bộ chữ cái riêng cho mình, chia làm hai nhóm như sau:

• **Các ký tự chữ và số:**

- 26 chữ cái a, b, c,..., z. (Ngôn ngữ Pascal không phân biệt chữ hoa chữ thường)
- 10 chữ số thập phân 0, 1, 2, ..., 9.

• **Các ký tự đặc biệt:**

- Ký tự trống
- Dấu các phép toán số học +,-,*, /
- Dấu các phép so sánh >,<, =,>= , <=
- Các ký tự đặc biệt khác $, #, (,), :, '
- ....

Đây là tập hợp các ký tự hợp lệ được dùng để viết các chương trình, không được dùng các ký hiệu khác ngoài các ký hiệu nói trên chẳng hạn như: ψ, π, α, δ, φ, ϖ, β, $\sum$,...

Các ký tự trên được nhóm lại theo nhiều cách khác nhau để tạo ra một số từ nhất định gọi là từ khoá. Đến lượt chúng, các từ khoá lại liên kết với nhau theo những quy tắc cần phải được tôn trọng (gọi là cú pháp) để tạo thành các câu lệnh mà chúng ta sẽ dần dần được làm quen trong quá trình giới thiệu ngôn ngữ này. Một chương trình bao gồm nhiều câu lệnh diễn đạt một thuật toán nào đó.

II. CÁC TỪ KHÓA

Các từ khoá là một bộ phận không thể tách rời của TURBO PASCAL được định nghĩa trước với ý nghĩa xác định và không thể định nghĩa lại. Các từ khoá được sử dụng để khai báo các kiểu dữ liệu (ví dụ: integer, real...), viết các toán tử (ví dụ: mod, div...), diễn đạt các câu lệnh (ví dụ: if, then ...), các từ khoá phải được dùng đúng cú pháp, không được dùng vào việc khác, người lập trình không được phép đặt tên mới trùng với các từ khoá. Sau đây là một số từ khoá:

array	do	begin
end	case	of
for	to	downto
const	type	var
div	set	else
and	or	if
repeat	while	record

III. TÊN (ĐỊNH DANH)

Tên theo định nghĩa là một dãy ký tự dùng để chỉ tên biến, tên hằng, tên kiểu dữ liệu, tên chương trình con... Tên bắt đầu bằng chữ cái, sau đó có thể là chữ cái, chữ số, dấu gạch nối, không được dùng các ký tự đặc biệt như dấu trống, dấu ":", dấu ";", dấu "?", dấu "!"... để đặt tên.

Sau đây là một số tên hợp lệ : K40, bach_khoa, CNTT, ma_tran,... còn các tên sau đây là không hợp lệ: 12A4H (vì bắt đầu bằng chữ số), kien truc (vì có chứa dấu trống).

Độ dài cực đại của tên là 63 ký tự, nếu ta đặt một tên với độ dài lớn hơn 63 ký tự thì chỉ có 63 ký tự đầu tiên có ý nghĩa.

TURBO PASCAL không phân biệt chữ hoa với chữ thường, như vậy các tên : Abc, aBC, ABC là giống nhau. Trong khi lập trình người ta thường đặt tên sao cho nó phải phản ánh được nội dung của đối tượng. Việc đặt tên theo kiểu diễn nghĩa như vậy rất có ích khi bảo trì chương trình.

IV. CÁC TÊN CHUẨN

TURBO PASCAL xác định một số tên chuẩn cho các kiểu hằng, biến, thủ tục và hàm được định nghĩa sẵn. Các tên chuẩn này có thể định nghĩa lại

nhưng điều này có thể gây ra sự nhầm lẫn vậy tốt nhất là không nên thay đổi. Sau đây là một số tên chuẩn:

sin	cos	tan
exp	false	boolean
char	odd	copy
.....		

V. CÁC DÒNG CHƯƠNG TRÌNH

Một chương trình Pascal là tập hợp của nhiều lệnh. Mỗi lệnh đều được kết thúc bởi dấu ";", đây là quy ước bắt buộc của ngôn ngữ Passcal. Mỗi lệnh có thể được viết trên các dòng khác nhau hoặc cũng có thể viết trên cùng một dòng. Nếu viết trên một dòng thì độ dài tối đa của một dòng chương trình là 127 ký tự.

VI. CÁC KIỂU DỮ LIỆU CHUẨN

Một kiểu dữ liệu là một tập hợp các giá trị mà một biến kiểu đó có thể nhận. Mỗi biến trong chương trình đều phải kết hợp với một và chỉ một kiểu dữ liệu. Một kiểu dữ liệu trong TURBO PASCAL có thể phức tạp nhưng trong mọi trường hợp nó đều được cấu thành từ các kiểu dữ liệu chuẩn như: integer, real, char, boolean. Ta lần lượt khảo sát các kiểu dữ liệu này:

1. Kiểu Integer

Còn gọi là kiểu nguyên, trong TURBO PASCAL một biến kiểu Integer có thể lấy các giá trị nằm trong khoảng [-32768, 32767]. Mỗi giá trị kiểu Integer chiếm 2 bytes bộ nhớ.

Tuy vậy việc thực hiện các phép toán đối với các dữ liệu kiểu Integer dẫn đến kết quả vượt quá phạm vi nói trên sẽ có thể không được thông báo. Vì vậy chúng ta đặc biệt lưu ý đến vấn đề này. Ví dụ biểu thức 1000*100/50 sẽ không cho kết quả là 2000 vì tích 1000*100 đã cho kết quả vượt qua 32767.

Vì kiểu Integer chỉ biểu diễn được các số nguyên khá bé nên từ TURBO PASCAL 4.0 trở đi, người ta định nghĩa thêm các kiểu nguyên khác.

shortInt	1 byte	[-128,127]
longInt	4 byte	[-2 147 483 648, 2 147 483 647]
Word	2 byte	[0,65535]
Byte	1 byte	[0,255]

2. Dữ liệu kiểu Real

Một biến kiểu Real có thể lấy các giá trị nằm trong khoảng $[2.9x10^{-39}, 1.7x10^{38}]$, một giá trị kiểu Real chiếm 6 byte bộ nhớ.

Khi thực hiện các phép toán với các giá trị Real nếu kết quả vượt ra ngoài khoảng trên chương trình sẽ dừng lại, máy tính báo lỗi tràn ô nhớ. Còn nếu kết quả quá bé trong dấu giá trị tuyệt đối sẽ được xem bằng 0.

3. Kiểu Boolean

Một giá trị kiểu Boolean chỉ có thể là một trong hai giá trị TRUE và FALSE, các giá trị này được xếp thứ tự như sau TRUE>FALSE. Một giá trị kiểu Boolean chiếm 1 byte bộ nhớ.

4. Kiểu Char

Còn gọi là kiểu ký tự bao gồm 256 ký tự trong bảng mã ASCII. Một ký tự được viết trong hai dấu nháy đơn; ví dụ: '1', 'a', 'H',... Một giá trị kiểu Char chiếm 1 byte bộ nhớ.

Một ký tự được biểu diễn trong bộ nhớ bởi giá trị của nó trong bộ mã ASCII. Ví dụ ký tự 'A' có mã ASCII là 65, sẽ được biểu diễn trong bộ nhớ bằng 1 byte có trị là 65.

Các ký tự có thể so sánh với nhau. Sự so sánh được tiến hành theo giá trị của chúng trong bộ mã biểu diễn.

Ví dụ nếu dùng ASCII thì:

'A'<'B' vì 65<66.

Trong TURBO PASCAL có hai hàm:

- Ord( ): Cho ta giá trị tương ứng của ký tự trong bộ mã, chẳng hạn ord('A')=65.
- Chr(i): Thủ tục này cho ta ký tự tương ứng với i trong bộ mã, chẳng hạn chr(65)='A'.

Chương 3

BIỂU THỨC, CÂU LỆNH VÀ CẤU TRÚC CHƯƠNG TRÌNH

Biểu thức bao gồm các toán hạng (biến, hằng, lời gọi hàm) kết hợp với nhau bằng các toán tử.

I. CÁC TOÁN TỬ

Các toán tử trong TURBO PASCAL được chia thành 5 lớp có thứ tự ưu tiên như sau:

1. Toán tử đảo dấu

Đây chính là phép trừ không có số bị trừ. Toán tử này cho phép đổi dấu của toán hạng thuộc kiểu Integer hay Real đặt ngay sau nó.

2. Toán tử NOT

Toán tử NOT phủ định giá trị của một toán hạng kiểu Boolean.

3. Các toán tử thuộc lớp nhân

Toán tử	Phép toán	Kiểu toán hạng	Kiểu kết quả
*	nhân	real	real
*	nhân	integer	integer
*	nhân	real, integer	real
/	chia	real	real
/	chia	real, integer	real
/	chia	integer	real
div	chia nguyên	integer	integer
mod	lấy phần dư	integer	integer
and	và logic	boolean	boolean

4. Các toán tử lớp cộng

Toán tử	Phép toán	Kiểu toán hạng	Kiểu kết quả
+(-)	cộng(trừ)	real	real
+(-)	cộng(trừ)	integer	integer
+(-)	cộng(trừ)	real, integer	real
or	hoặc logic	boolean	boolean

5. Các toán tử quan hệ

Các toán tử này tác dụng lên tất cả các kiểu dữ liệu chuẩn. Một điều nên nhớ rằng chỉ có thể so sánh các toán hạng cùng kiểu, riêng các toán hạng kiểu Integer và Real có thể so sánh với nhau bằng toán tử quan hệ, kết quả của phép so sánh bao giờ cũng thuộc kiểu Boolean. Các toán tử quan hệ bao gồm: =, >,<, >=, <=, <> (khác).

II. CÁC HÀM

PASCAL thiết kế sẵn một số hàm chuẩn, ta sẽ dần dần làm quen với chúng. Sau đây là một số hàm chuẩn hay dùng:

Toán học	TURBO PASCAL
x^2	sqr(x)
$\sqrt{x}$	sqrt(x)
cosx	cos(x)
sinx	sin(x)
lnx	ln(x)
e^x	exp(x)
| x |	abs(x)
.....	

III. CẤU TRÚC CỦA MỘT CHƯƠNG TRÌNH PASCAL

Bao gồm 3 phần:

1. Phần tiêu đề của chương trình
2. Phần khai báo dữ liệu
3. Phần thân chương trình.

Ví dụ 1: Để minh hoạ chúng ta xét một ví dụ đơn giản sau:

```
Program tinh_dien_tich_hinh_tron;
 const
  pi=3.14;
        var
        s, r: real;
BEGIN
r:=5;
s:=pi*r*r;
END.
```

Ta lần lượt đề cập tới từng phần của cấu trúc chương trình:

1. Phần tiêu đề

Phần này không nhất thiết phải có, nhưng nó chứa tên chương trình, thông thường tên được đặt sao cho có thể gợi nhớ đến nội dung của chương trình, do đó phần nào giúp ta phân biệt được các chương trình khác nhau. Phần này bắt đầu bằng từ khoá **Program**.

2. Phần khai báo

PASCAL yêu cầu người lập trình phải liệt kê tất cả các "đối tượng" sẽ được sử dụng trong chương trình, điều này có nghĩa là tất cả các kiểu dữ liệu, các biến, các thủ tục phải khai báo trước. Một chương trình PASCAL có thể sử dụng các dữ liệu là hằng số, hay biến số.

Hằng số là dữ liệu mà giá trị của nó do người lập trình xác định, giá trị này không thay đổi trong quá trình thực hiện chương trình.

Biến số là dữ liệu mà giá trị của nó do chính chương trình xác định, giá trị này có thể thay đổi trong quá trình thực hiện chương trình.

Phần này bao gồm nhiều mục, tuy nhiên tuỳ theo từng chương trình cụ thể phần khai báo có thể thiếu một hay nhiều mục, thậm chí không có mục nào, nhưng phần này cũng có thể dài hơn thân chương trình. Ta lần lượt xét tất cả các mục này:

2.1. Phần khai báo Unit

Đây là một nét mới tạo sức mạnh của TURBO PASCAL từ thế hệ version 4.0 trở đi, nó thể hiện ở chỗ nó cho chúng ta chia chương trình thành nhiều modul (một modul hay còn gọi là 1 Unit được biểu hiện là các dữ liệu + các chương trình con có liên quan) và biên dịch chúng một cách riêng rẽ. Làm như

vậy khi sửa chữa hay thay đổi một modul, chúng ta không cần biên dịch lại toàn bộ chương trình. Mặt khác một Unit có thể được sử dụng bởi nhiều chương trình khác nhau. Nếu một chương trình nào đó cần tới một unit cụ thể chỉ cần khai báo nó trước. Các modul này có thể do người lập trình tự xây dựng, trong ngôn ngữ PASCAL cung cấp cho chúng ta một số Unit, khi cần sử dụng Unit nào ta chỉ cần khai báo bằng từ khoá Uses tiếp theo là danh sách các Unit cần sử dụng, khai báo này được đặt tại dòng đầu tiên của chương trình ngay sau dòng tiêu đề.

Ví dụ:

Uses graph, crt, dos;

Có hai loại Unit:

- Một loại do chính người lập trình tạo ra.
- Loại unit chuẩn.

Có tất cả 7 unit chuẩn:

- **DOS**: là unit chứa các thủ tục và hàm liên quan tới các thủ tục và hàm của HĐH DOS.
- **CRT**: là unit cung cấp các phương tiện xử lý màn hình, bàn phím.
- **PRINTER**: chứa các dữ liệu và chương trình con giúp chúng ta sử dụng máy in.
- **SYSTEM**: là thư viện chuẩn.
- **GRAPH**: cung cấp các thủ tục về đồ hoạ.
- **TURBO3 và GRAPH3**: cho chúng ta các phương tiện tương thích với TURBO PASCAL 3.0.

Để sử dụng các thủ tục và dữ liệu trong các Unit ta chỉ cần khai báo. Ví dụ ta muốn vẽ đồ thị của một hàm số, ta cần tới thủ tục đồ hoạ, muốn vậy ta khai báo như sau:

uses graph;

2.2. Phần khai báo kiểu dữ liệu

Đây cũng là một điểm mạnh của TURBO PASCAL, nó cho ta khả năng tự thiết kế ra kiểu dữ liệu mới thuận tiện cho công việc lập trình của mình. Lẽ dĩ nhiên bất cứ kiểu mới nào cũng được thiết kế từ các kiểu sẵn có của TURBO PASCAL. Một kiểu mới sẽ được mô tả cụ thể bắt đầu bằng từ khoá **type**. Ta sẽ quay lại vấn đề này vào một thời điểm thích hợp nhất.

2.3. Phần khai báo biến

Biến là một đại lượng có thể thay đổi giá trị giống như khái niệm biến toán học. Biến của một chương trình thực chất là tên của một ô nhớ. Tất cả

các biến của chương trình phải được khai báo ở đầu chương trình sau từ khoá **Var**.

Cách khai báo như sau:

```
Var
 tb1: k1;
 tb2: k2;
 .....
 tbn: kn;
```

Ở đây các tbi là các tên, còn các ki là các kiểu dữ liệu có sẵn hoặc là các kiểu dữ liệu vừa được định nghĩa trong mục **type**.

Chú ý: Nếu có nhiều biến cùng kiểu ta có thể khai báo trên cùng một dòng, chẳng hạn :

```
Var
   r, s: real;
```

2.4. Phần khai báo chương trình con

Mục này được bắt đầu bằng từ khoá **function** hay **procedure**, đó chính là mục khai báo các chương trình con (ctc) mà chúng ta sẽ nói kỹ ở phần sau.

3. Thân chương trình

Phần này gồm các câu lệnh của PASCAL nằm giữa hai từ khoá "**Begin**" và "**End**".

IV. LỆNH GÁN

- Đây là một lệnh cơ bản của TURBO PASCAL, dùng để truyền giá trị của một hằng, của một biến, của một biểu thức vào một biến. Phép gán được biểu diễn bằng ký hiệu ":=".

Cách viết:

```
bien:=bth;
```

Ở đây biến là tên một biến đã được khai báo ở mục Var, còn bth là một biểu thức có cùng kiểu với biến ở vế trái. Ví dụ:

```
x:=x+1;
y:=x;
z:=3.4;
.....
```

Chương 4

CÁC THỦ TỤC VÀO RA DỮ LIỆU

I. THỦ TỤC HIỂN THỊ DỮ LIỆU RA MÀN HÌNH

Có 3 dạng viết như sau:

```
write(danh sách biến và biểu thức);
writeln(danh sách biến và biểu thức);
writeln;
```

Trong đó *danh sách biến và biểu thức* là các biến và biểu thức mà giá trị của nó cần hiển thị ra màn hình. *Danh sách biến và biểu thức* có thể chỉ có 1 hoặc cũng có thể có nhiều biến và biểu thức, nếu có trên 1 biến và biểu thức thì mỗi biến và biểu thức cách nhau bởi dấu ",".

Ví dụ:

```
Program tinh_dt;
 const
 pi=3.14;
 var
 s,r:real;
BEGIN
 r:=5;
 s:=pi*r*r;
 write(s);
END.
```

Khi thực hiện ta thu được kết quả: 7.853981634E+01.

Sự khác nhau giữa hai lệnh đầu là ở chỗ vị trí con trỏ màn hình sau khi kết thúc lệnh. Đối với lệnh thứ hai, sau khi hiển thị các giá trị, con trỏ màn hình sẽ

chuyển xuống đầu dòng tiếp theo. Còn đối với lệnh đầu, sau khi hiển thị các giá trị, con trỏ màn hình sẽ không chuyển xuống dòng tiếp theo.

Còn thủ tục writeln sẽ chỉ làm một động tác đơn giản là đưa con trỏ màn hình xuống dòng tiếp theo.

Hiển thị có quy cách

Như ta đã thấy khi hiển thị kết quả ra màn hình (trong trường hợp giá trị thực) hệ thống thể hiện dưới dạng dấu phẩy động rất khó nhận biết. Vì vậy TURBO PASCAL thiết kế thêm một cơ chế hiển thị có quy cách ra màn hình. Cách viết như sau:

write(bt:m:n);

Ở đây m và n là các số nguyên dương. Tác dụng của lệnh trên như sau: hiển thị giá trị của biểu thức bt ra màn hình trên tất cả m cột trong đó có n cột cho phần thập phân. Ví dụ nếu ta chữa lệnh write(s) thành write(s:5:2), ta thu được: 78.54.

Đối với các giá trị nguyên cách hiển thị có quy cách được viết như sau:

write(btn:m)

Trong đó btn là biểu thức có giá trị nguyên, còn m là một số nguyên, lúc đó máy sẽ dành m chỗ (cột) để hiển thị giá trị của btn, còn nếu thừa chỗ nó sẽ để trắng bên trái.

II. LỆNH IN DỮ LIỆU RA MÁY IN

Để đưa dữ liệu ra máy in, ta dùng các lệnh write và writeln với tham số đầu tiên là LST. Vì biến LST được khai báo trong **Unit printer** vì vậy cần khai báo Unit này trong mục khai báo USES ; chẳng hạn:

```
Program minh_hoa;
uses printer;
begin
        write(lst,'x=',12,'y=',3.55:4:2);
end.
```

Chương trình này sẽ in ra giấy nội dung sau:

```
x=12 y=3.55
```

III. MỘT SỐ HÀM VÀ THỦ TỤC TRÌNH BÀY MÀN HÌNH TRONG TURBO PASCAL

Các hàm và thủ tục này chứa ở trong unit CRT; vì vậy để sử dụng nó trước hết chúng ta phải có khai báo sau:

uses CRT;

Sau đây là một số hàm thủ tục hay dùng:

• **GotoXY(x,y)**: thủ tục này di chuyển con trỏ màn hình đến dòng y cột x, lưu ý rằng màn hình được chia thành 25 dòng và 80 cột. Cột đầu tiên được đánh số 1 và dòng đầu tiên cũng được đánh số 1.

• **Clrscr**: thủ tục xoá toàn bộ màn hình và đưa con trỏ về dòng 1, cột 1 của màn hình.

• **WhereX**: hàm cho giá trị kiểu byte cho biết con trỏ đang ở cột nào.

• **WhereY**: hàm cho giá trị kiểu byte cho biết con trỏ đang ở dòng nào.

• **Windows(x1,y1,x2,y2)**: thủ tục thiết lập cửa sổ hoạt động trên màn hình mà góc trái trên có toạ độ (x1,y1), góc phải dưới có toạ độ (x2,y2). Khi đã thiết lập cửa sổ hoạt động các toạ độ phải tính lại theo quy tắc:

hoành độ mới= hoành độ cũ-x1+1

tung độ mới = tung độ cũ-y1+1

Ví dụ muốn di chuyển con trỏ đến dòng 8 cột 10 trên màn hình sau khi đã thực hiện lệnh windows(5,7,75,15) ta viết gotoXY(6,2).

IV. CÁCH ĐẶT MÀU NỀN VÀ MÀU CHỮ

Để xác định màu nền ta dùng thủ tục:

textbackground(color:byte);

Để xác định mầu chữ ta dùng thủ tục:

textcolor(color:byte);

Trong cả hai trường hợp biến **color** chứa mã màu. Mã màu là các hằng được định nghĩa trong Unit CRT. Biến **color** trong thủ tục đầu có thể nhận 8 giá trị đầu của bảng dưới đây, còn biến **color** của thủ tục thứ hai có thể nhận được cả 16 giá trị:

black=0(đen)	darkgray=8(xám đậm)
blue=1(xanh da trời)	lightblue=9(xanh da trời nhạt)
green=2(xanh lá cây)	lightgreen=10(xanh lá cây nhạt)
cyan=3(xanh lơ)	lightcyan=11(xanh lơ nhạt)
red=4(đỏ)	lightred=12(đỏ nhạt)
magenta=5(tím)	lightmagenta=13(tím nhạt)
brown=6(nâu)	Yellow=14(vàng)
lightgray=7(xám nhạt)	White=15(trắng)

Ví dụ: Các lệnh sau đây sẽ hiển thị lên màn hình dòng chữ Đại học Bách khoa màu đỏ:

```
textcolor(red);
write('Đại học Bách khoa');
```

Chú ý: Thủ tục textbackground sẽ làm mất hiệu lực của thủ tục textbackground trước đó. Thủ tục này thường đi kèm với các thủ tục windows và clrscr. Để minh hoạ cách sử dụng ta xét ví dụ sau:

Giả sử ta muốn tạo một cửa sổ làm việc với màu xanh da trời và với góc bên trái có toạ độ (10,5), góc dưới có toạ độ (70,20), ta sử dụng các lệnh sau:

```
textbackground(1);
windows(10,5,70,20);
clrscr;{được cửa sổ nền xanh da trời}
```

Bây giờ nếu muốn chuyển sang cửa sổ mầu tím ta viết lại hai lệnh sau:

```
textbackground(5);
clrscr; {được cửa sổ nền xanh tím}
```

V. THỦ TỤC VÀO DỮ LIỆU TỪ BÀN PHÍM

Như ta đã thấy để vào dữ liệu có thể dùng lệnh gán, nhưng trong thực tế nếu dùng lệnh gán để vào dữ liệu sẽ rất bất tiện. Bởi vậy để vào dữ liệu ta nên dùng lệnh read và readln. Có 3 dạng viết như sau:

```
read(danh sách biến);
readln(danh sách biến);
readln;
```

Trong đó *danh sách biến* nếu có từ 2 biến trở lên thì giữa chúng phải cách nhau bởi dấu phẩy.

Hai dạng đầu giúp chúng ta gán giá trị cho các biến khi chạy chương trình; chẳng hạn nếu ta muốn gán cho x giá trị 5, y giá trị 3.45 và z giá trị 12.4, lúc đó ta viết trong chương trình lệnh sau:

```
read(x,y,z);
```

Khi chạy chương trình lúc gặp lệnh này máy sẽ dừng lại đợi ta nhập dữ liệu thông qua bàn phím, ta cần gõ:

```
5 3.45 12.4 ↵
```

Lưu ý là các cụm dữ liệu tương ứng với các biến phải cách nhau ít nhất một dấu cách và phải phù hợp về kiểu cũng như số lượng.

Giữa hai thủ tục đầu chỉ khác nhau chút ít. Đối với thủ tục đầu sau khi gõ ↵ (phím return) con trỏ màn hình không nhảy xuống dòng dưới, trong khi đó với thủ tục thứ hai khi gõ ↵ (phím return) con trỏ màn hình nhảy xuống dòng dưới.

Lệnh readln chỉ có tác dụng dừng màn hình để ta xem kết quả thực hiện chương trình.

VI. KẾT HỢP GIỮA WRITE VÀ READLN, HỘI THOẠI NGƯỜI-MÁY

Trong các ví dụ dùng read và readln vừa xét thể hiện một điểm yếu là không có chỉ dẫn trên màn hình để báo cho ta biết đang cần đưa giá trị vào cho biến nào. Vì vậy ta nên kết hợp hai thủ tục write và readln để tạo một giao diện thuận tiện khi vào dữ liệu:

Ví dụ ta nên thay lệnh read(x,y,z) bằng cụm lệnh sau:

```
write('x='); readln(x);
write('y='); readln(y);
write('z='); readln(z);
```

Khi đó lúc chạy chương trình ta thu được màn hình như sau:

```
x=5↵
y=3.45↵
z=12.4↵
```

Rõ ràng khi tiến hành như vậy ta khó mà phạm phải sai lầm.

Chương 5
CÂU LỆNH ĐIỀU KIỆN

I. CÂU LỆNH IF...THEN...ELSE

- Dạng không đầy đủ:

  if <bt> then <lệnh> ;

- Dạng đầy đủ:

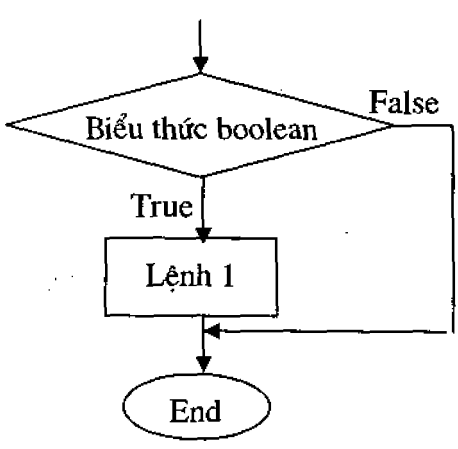
  if <bt> then <lệnh1>

  else <lệnh2> ;

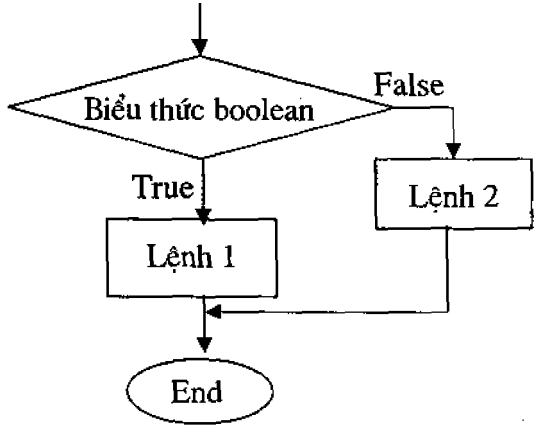
Trong đó if, then, else là các từ khoá, bt là một biểu thức boolean, còn <lệnh>, <lệnh1>, <lệnh2> là các lệnh của PASCAL.

Bộ vi xử lý sẽ hoạt động ra sao khi gặp lệnh này?

Đối với dạng không đầy đủ nếu <bt> có giá trị đúng thì thực hiện <lệnh>; trong trường hợp ngược lại thì bỏ qua không thực hiện <lệnh>. Ta có thể minh hoạ hoạt động đó bằng sơ đồ khối sau:



Dạng không đầy đủ



Dạng đầy đủ

Lưu đồ thực hiện lệnh IF ... THEN ... (nhánh ELSE nối tắt) và lệnh IF ... THEN ... ELSE...

Đối với dạng đầy đủ nếu <bt> có giá trị đúng thì thực hiện <lệnh1>, trong trường hợp ngược lại thì thực hiện <lệnh2>. Ta có thể minh hoạ hoạt động của bộ vi xử lý như sơ đồ khối trên.

Nói tóm lại câu lệnh **if** chỉ ra rằng nếu biểu thức boolean cho giá trị đúng thì câu lệnh đứng sau từ khóa **then** được thực hiện, ngược lại thì hoặc là không có câu lệnh nào hoặc là câu lệnh đứng sau **else** được thực hiện.

Chú ý: Không có dấu " ; " trước từ khoá **else**.

Những câu lệnh **if** có thể lồng với nhau nên ta có thể gặp tình huống sau:

```
if <bt> then
   if <bt2> then <lệnh1>
         else <lệnh2> ;
```

Lúc đó hệ thống sẽ giải quyết bằng cách tự động hiểu như đã viết:

```
if <bt> then
    begin
         if <bt2> then <lệnh1>
                    else <lệnh2>;
    end;
```

Điều này có nghĩa là **else** luôn luôn kẹp đôi với **if** gần nhất chưa có else.

Sau đây là một số ví dụ về cách sử dụng lệnh **if...then**:

Cho hàm số $f(x) = \begin{cases} x + \sin x & \text{khi } x \geq 0 \\ x & \text{khi } x < 0 \end{cases}$

Các câu lệnh sau đây sẽ tính giá trị của hàm sau khi nhận được một giá trị của x đọc vào máy thông qua bàn phím:

```
write('Hay doc vao mot gia tri:'); readln(x);
if x<0 then f:=x
 else f:=x+sin(x);
```

Với những kiến thức được trang bị cho tới thời điểm này bạn đọc hoàn toàn có thể viết hoàn chỉnh một chương trình cho phép tính giá trị của hàm trên.

Ví dụ: Tìm giá trị lớn nhất trong ba giá trị đọc vào máy thông qua bàn phím:

```
   Program Tim_gia_tri_lon_nhat;
    var
         a,b,c,max: real;
```

```
BEGIN
 write('Hay doc vao ba so:'); readln(a,b,c);
 if a>b then if a>c then max:=a
 else if b>c then max:=b
                 else max:=c;
 write('Gia tri lon nhat la:', max:8:2);
 readln;
END.
```

Nhân đây chúng tôi lưu ý bạn đọc về ký pháp "thụt vào" khi viết một chương trình (cách viết có cấu trúc nhô ra thụt vào), ký pháp này giúp cho chúng ta thể hiện được ý đồ của thuật toán. Đây là một tiêu chuẩn cần phải có cho một chương trình viết bằng TURBO PASCAL. Các cặp **begin**, **end** tương ứng bao giờ cũng thẳng hàng. Tất nhiên ta có thể trình bày lại chương trình trên như sau:

```
Program Tim_gia_tri_lon_nhat;
var
a,b,c,max:real;
BEGIN
write('Hay doc vao ba so:'); readln(a,b,c);
if a>b then if a>c then max:=a
else max:=c
else if b>c then max:=b
else max:=c;
write('Gia tri lon nhat la:', max:8:2);
readln;
END.
```

Hay thậm chí:

```
Program Tim_gia_tri_lon_nhat; var a,b,c,max:real; begin write('Hay doc vao ba so:'); readln(a,b,c); if a>b then if a>c then max:=a else max:=c else if b>c then max:=b else max:=c; write('Gia tri lon nhat la:', max:8:2); readln; end.
```

Ta thấy ngay hai cách trình bày sau là không sáng sủa, không đẹp mắt và làm cho chương trình trở nên khó kiểm soát, vì thế chúng ta không nên sử dụng cách viết này!

II. CÂU LỆNH LỰA CHỌN CASE

Như đã thấy nếu dùng lệnh **if** ta chỉ có thể chọn một trong hai khả năng; trong tình huống ta cần chọn một trong nhiều khả năng nếu dùng lệnh **if** sẽ có nhiều bất tiện. Trong trường hợp này tốt nhất ta nên sử dụng lệnh **case**. Lệnh này cũng có hai dạng như sau:

Dạng 1

```
case <bt> of
 hằng1: <lệnh1>;
 hằng2: <lệnh2>;
 ..............
 Hằng n: <lệnhn>
end;
```

Dạng 2

```
case <bt> of
 hằng1: <lệnh1>;
 hằng2: <lệnh2>;
 ..............
 Hằng n: <lệnhn>;
 else <lệnh n+1>
end;
```

Trong đó <bt> không chỉ là kiểu logic mà còn có thể là kiểu: integer, byte, liệt kê, miền con, nhưng không được là kiểu thực.

Chú ý: Trong trường hợp có nhiều hằng i ứng với cùng một lệnh, chúng ta có thể gom chúng lại trong một hàng.

Ví dụ:

```
Readln(thang);
case thang of
 1,2,3 : writeln('đây là tháng của quý I');
 4,5,6 : writeln('đây là tháng của quý II');
7,8,9: writeln('đây là tháng của quý III');
10,11,12 : writeln('đây là tháng của quý IV');
end;
```

Ví dụ: Chương trình tính số ngày của một tháng:

```
Var
 so_ngay, thg, nam:integer;
BEGIN
write('Hay vao mot thang:'); readln(thg);
write('Hay vao mot nam:'); readln(nam);
case thang of
        4,6,9,11: so_ngay:=30;
```

```
    2 : case (nam mod 4) of
                    0: so_ngay:=29;
            else so_ngay:=28;
            end;
            1,3,5,7,8,10,12: so_ngay:=31;
end;
            writeln('Thang', thg, 'co:', so_ngay, 'ngay');
readln;
END.
```

Chương 6

CÂU LỆNH ĐIỀU LẶP

Dùng để thực hiện lặp đi lặp lại nhiều lần cùng một số câu lệnh nào đó. nếu số lần lặp biết trước ta sử dụng lệnh FOR, nếu không ta dùng lệnh WHILE hay REPEAT.

I. CÂU LỆNH FOR

Câu lệnh FOR chỉ ra rằng lệnh thành phần phải được thực hiện lặp đi lặp lại nhiều lần chừng nào giá trị của biến điều khiển vẫn còn nằm giữa hai giá trị đầu và giá trị cuối.

Cách viết lệnh: lệnh FOR có hai dạng như sau:

- for <bdk>:= <gtđ> to <gtc> do <lệnh>
- for <bdk>:= <gtđ> downto <gtc> do <lệnh>

Trong đó for, to, do, downto là các từ khoá.

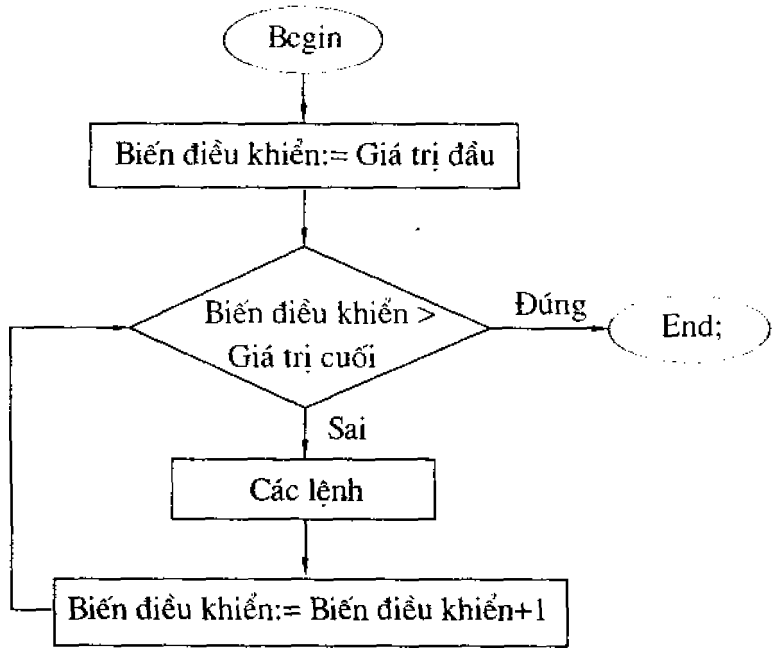
<bdk> là biến điều khiển vòng lặp có kiểu nguyên, miền con, liệt kê, nhưng không được là kiểu thực.

<gtđ>, <gtc> là biểu thức có kiểu giống như biến điều khiển.

<lệnh> là câu lệnh đơn của PASCAL, trong trường hợp có nhiều lệnh cần thực hiện ta phải đặt chúng trong cặp từ khoá begin... end.

Ta có thể miêu tả cơ chế hoạt động của lệnh FOR như sau:

Đối với dạng đầu:



Lưu đồ vòng lặp xác định FOR ... TO ... DO...

Sơ đồ minh hoạ cho hoạt động của dạng còn lại dành làm bài tập cho bạn đọc.

Để làm thí dụ ta hãy viết một chương trình cho phép tính giá trị của tổng sau:

$S = 1 + 1/2 + 1/3 + ... + 1/100.$

Ví dụ 1:

```
Program tdfor;
   var
        i , n : integer;
        s : real;
BEGIN
        write('n='); readln(n);
        s:=0;
        for i:=1 to n do s:=s + 1/i;
        writeln('gia tri tinh duoc la:', s:6:2);
    readln;
END.
```

Ta lưu ý mấy điểm sau:

1) <gtd> và <gtc> chỉ tính một lần.

2) Đối với dạng đầu nếu gtd > gtc lệnh FOR không thực hiện.
Đối với dạng thứ hai nếu gtd < gtc lệnh FOR không thực hiện.

3) Biến điều khiển không bị câu lệnh sau đó thay đổi giá trị.

4) Câu lệnh FOR có thể lồng nhau.

Ta xét thêm một ví dụ khác:

Ví dụ 2:

Hãy dùng vòng FOR để viết một chương trình in ra màn hình nội dung sau:

```
1 2 3 4 5 6 5 4 3 2 1
  2 3 4 5 6 5 4 3 2
    3 4 5 6 5 4 3
      4 5 6 5 4
        5 6 5
          6
```

Sau đây là chương trình:

```
uses crt;
  var i,j : integer;
BEGIN
        clrscr;
        for i:=1 to 6 do
            begin
                gotoXY(35+i-1, 10+i-1);
                for j:=i to 6 do write(j);
                for j:=5 downto i do write(j);
                    writeln;
            end;
        readln;
END.
```

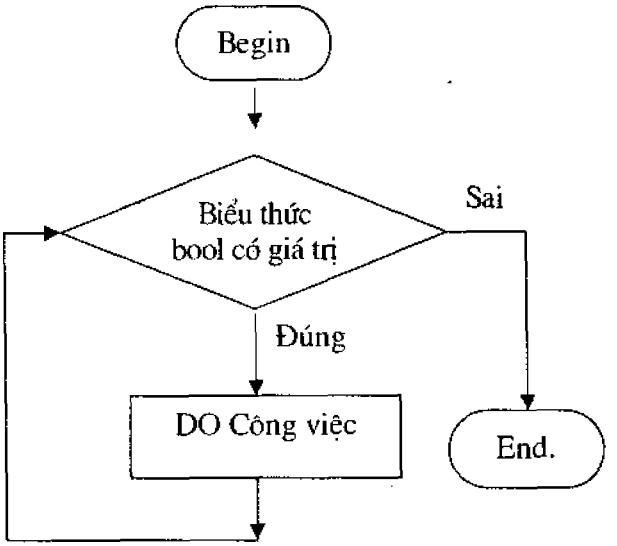
II. CÂU LỆNH WHILE

Dạng lệnh như sau:

WHILE <bt> DO <lệnh>

Trong đó WHILE, DO là các từ khoá, <bt> là biểu thức logic, biểu thức này được đánh giá trước khi lặp (vì vậy nên còn gọi là lệnh lặp với điều kiện

trước). <lệnh> sẽ thực hiện lặp lại nếu giá trị của biểu thức điều khiển vẫn còn đúng. Tất nhiên nếu giá trị của biểu thức này sai ngay từ đầu thì <lệnh> sẽ không thực hiện một lần nào cả (xem hình).



Sau đây là chương trình tính tổng đã nói đến ở mục trước, nhưng sử dụng lệnh WHILE:

Ví dụ 3:

```
Program tdwhile;
    var
        i,n: integer;
         s: real;
BEGIN
    write('n='); readln(n);
    s:=0; i:=1;
    while i<=n do
       begin
          s:=s+1/i; i:=i+1;
       end.
   writeln('gia tri tinh duoc la:', s:6:2);
    readln;
END.
```

Ví dụ 4: Tính hàm số sinx theo công thức sau:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + ... + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

Số phần tử được chọn cho tới khi đạt được độ chính xác:

$$\left| \frac{x^{2n+1}}{(2n+1)!} \right| < \varepsilon$$

Các giá trị x và ε được đọc từ bàn phím.

Để có thể tính được ta hãy tìm mối quan hệ giữa các số hạng ở vế phải: nhận xét rằng nếu gọi các số hạng này là To, T_1,...,T_n ta có:

$$T_0 = x$$

$$T_k = -\frac{x^2}{2k(2k+1)} T_{k-1} \qquad \text{với} \qquad k = 1,2,3...., n$$

Sử dụng công thức truy hồi này, chương trình được viết như sau:

```
var
   eps,x,s,T: real;
          n : integer;
BEGIN
     write('x='); readln(x);
     write('epsilon='); readln(eps);
     s:=x; T:=x; n:=1;
     while asb(T) >= esp do
          begin
              T:= T*sqr(x)/(2*n)/(2*n+1);
              s:= s+T; n:=n+1
          end;
    writeln('Gia tri ham sin tai diem', x:6:2, 'la:', s:8:6);
  readln;
END.
```

Ví dụ 5: Đọc vào máy một số tự nhiên rồi xét xem nó có là số nguyên tố hay không:

```
Program so_nguyen_to;
 var
    x, i : integer;
```

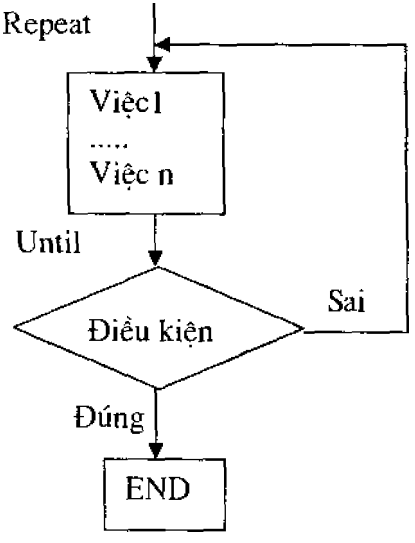
```
BEGIN
   write('Go vao mot so tu nhien:'); readln(x);
   if x<=1 then write ('Khong xet!')
      else begin
         i:=2
         while x mod i <>0 do i:=i+1;
         if i=x then writeln('x la so nguyen to')
              else writeln('x khong la so nguyen to');
         end;
   readln;
END.
```

III. CÂU LỆNH REPEAT

Lệnh này có dạng như sau:

```
REPEAT
        <lệnh1>;
        .....
        <lệnhn>;
UNTIL <bt>;
```

Trong đó REPEAT, UNTIL là các từ khoá, <lệnhn> là các lệnh của PASCAL; <bt> là biểu thức logic, câu lệnh Repeat sẽ thực hiện lặp đi lặp lại cho đến khi biểu thức logic này trở lên đúng. Bộ xử lý sẽ hoạt động như theo sơ đồ dưới khi gặp lệnh này:



Ví dụ 6: Ta tính lại tổng trên nhưng lần này ta dùng lệnh Repeat:

```
Program tdRepeat;
 var
    i, n : integer;
      s : real;
BEGIN
        write('n='); readln(n);
        s:=0; i:=1;
        repeat
                s:=s + 1/i;
                i:=i+1;
        until i>n.
     writeln('gia tri tinh duoc la:', s:6:2);
   readln;
END.
```

Ví dụ 7: Tính n!=1*2*....*n

```
var
        gt   : longint;
        i    : integer;
BEGIN
        gt:=1; i:=0;
        repeat
                i:=i+1;        {mỗi lần tăng i lên 1}
                gt:=gt*i;      {nhân dồn i vào giá trị}
        until i=n;  {nếu i còn <n thì quay ngược lên; nếu i=n thì chấm dứt}
      write(gt);
   readln;
END.
```

Chương 7

CÁC KIỂU DỮ LIỆU MỞ RỘNG

Ngoài các kiểu chuẩn PASCAL còn có một số kiểu dữ liệu mở rộng.

I. KIỂU DỮ LIỆU LIỆT KÊ

PASCAL cho phép người lập trình tự định nghĩa ra kiểu dữ liệu mới bằng cách liệt kê các thành phần của dữ liệu; danh sách các thành phần của dữ liệu được đặt trong cặp ngoặc đơn, mỗi thành phần cách nhau một dấu phẩy. Danh sách đó được miêu tả bằng một tên kiểu. Dữ liệu được định nghĩa như vậy gọi là kiểu liệt kê. Chẳng hạn:

```
Type
  mau=(do, den,trang, xanh,vang);
  t_pham=(ca, thit, trung, tom);
```

Lúc đó một biến kiểu liệt kê có thể khai báo thông qua tên kiểu đã được định nghĩa trong phần Type như sau:

```
var
    mau_sac:mau;
    th_an:t_pham;
```

Khi đó trong thân chương trình các lệnh sau đây là hợp lệ:

```
    mau_sac:=xanh;
    th_an:=thit;
```

Chú ý: Ta có thể khai báo các biến mau_sac, th_an lại như sau:

```
var
    mau_sac: (do, den, trang, xanh, vang);
    th_an : (ca, thit, trung, tom);
```

II. KIỂU DỮ LIỆU MIỀN CON (KHOẢNG CON)

Kiểu này cho phép ta tạo ra những loại dữ liệu mang sắc thái chuyên biệt (chẳng hạn phù hợp với đời thường...); cách định nghĩa như sau:

```
TYPE
    ten_kieu= can_duoi...can_tren
```

Ở đây ten_kieu là một tên của PASCAL; can_duoi, can_tren là các hằng có cùng kiểu giá trị (integer, boolean, ký tự, liệt kê) thoả mãn điều kiện:

```
    can_duoi < can_tren
```

Khi đó các giá trị của kiểu miền con sẽ được xác định trong khoảng từ can_duoi đến can_tren.

Ví dụ:

```
type
    tuoi_thanh_nien = 15..28;
    toc_do_o_to = 0..300;
var
    tuoi: tuoi_thanh_nien;
    van_toc: toc_do_o_to;
```

Khi đó phép gán tuoi:=12 hoặc tuoi:=31 là không hợp lệ;

Nhận xét kiểu miền con có hai tác dụng chính:

- Tiết kiệm bộ nhớ.
- Khi chạy chương trình máy có thể tự động kiểm tra xem biến có vượt ra ngoài giới hạn của nó không.

III. DỮ LIỆU KIỂU TẬP HỢP

Một tập hợp bao gồm một số đối tượng nào đó có liên quan với nhau. Một đối tượng trong một tập hợp như vậy gọi là phần tử của tập hợp. Trong tập hợp quan hệ thứ tự giữa các phần tử không được tính đến.

1. Định nghĩa kiểu tập hợp

Trong toán học bản chất của phần tử của một tập hợp là bất kỳ, nhưng trong PASCAL không phải như vậy, các phần tử của cùng một tập hợp phải cùng một kiểu cơ sở, TURBO PASCAL chỉ chấp nhận các tập không quá 256 phần tử, điều đó có nghĩa là trong khai báo về phần tử của tập hợp ta không thể cho phần tử đó chạy tập nền có quá 256 phần tử.

Một kiểu tập hợp được khai báo như sau:

set of <kcs>

Trong đó set, of là các từ khóa; còn kcs chỉ có thể là một trong các kiểu sau: Byte, Char hoặc là một trong các kiểu liệt kê do ta khai báo trong mục Type.

Ví dụ:

```
type
   so_chan =set of byte;
   mau     =set of (den, do, trang, vang, tim);
   hoa_qua =set of (tao, le, nho, chuoi, oi, roi);
var
   t1,t2: so_chan;
   r1,r2: hoa_qua;
   m1,m2: mau;
```

Các khai báo sau đây là không hợp lệ:

```
set of integer;
set of real;
set of string;
```

Chúng tôi nhắc lại rằng một tập hợp có nhiều nhất 256 phần tử, để chỉ rõ một tập hợp ta liệt kê các phần tử của nó trong một cặp ngoặc vuông [,]. Tập rỗng được biểu diễn bởi [].

Ví dụ ta có thể thực hiện các phép gán:

```
t1:= [1,2,3];
t2:= [];
r1:= [tao,lê];
```

2. Các phép toán trên tập hợp

Các phép toán quan hệ:

- Phép = cho giá trị là true nếu hai tập hợp bằng nhau.
- Phép <> cho giá trị là true nếu hai tập hợp khác nhau.
- Phép <=cho giá trị true nếu toán hạng bên trái được bao trong toán hạng bên phải.
- Phép >=cho giá trị true nếu toán hạng bên trái bao toán hạng bên phải.
- Phép toán in là toán tử kiểm tra sự có mặt của một phần tử trong một tập hợp; ví dụ 'a' in ['a','b','h'] cho giá trị true còn u in['a','k'] cho giá trị false.

Các phép toán hợp, giao, hiệu:

Giả sử A và B là hai tập hợp cùng kiểu thì:

- Toán tử cộng +: A+B là tập hợp mà các phần tử thuộc A hoặc thuộc B.
- Toán tử trừ -: A-B là tập hợp mà các phần tử thuộc A nhưng không thuộc B.
- Toán tử giao* :A*B là tập hợp mà các phần tử thuộc cả hai tập hợp.

Phép gán tập hợp:

Toán tử gán cũng được dùng để gán kết quả tính toán biểu thức tập hợp cho các biến tập hợp.

Chương 8.

DỮ LIỆU KIỂU MẢNG

Mảng là một cấu trúc bao gồm một số hữu hạn các phần tử có cùng kiểu, số phần tử của mảng được xác định khi khai báo; như vậy một mảng phải có một số cố định các phần tử và được sắp thứ tự có phần tử thứ nhất, phần tử thứ hai...: ví dụ mảng số nguyên, mảng số thực, mảng xâu...

Phép toán cơ bản liên quan tới mảng là phép truy nhập trực tiếp tới các phần tử của mảng để có thể tìm ra phần tử của mảng nằm ở vị trí đó, hay có thể lưu trữ dữ liệu tại vị trí đó. Phép truy nhập trực tiếp này được thể hiện thông qua tên mảng cùng với chỉ số nằm trong cặp [] (cặp ngoặc vuông; chẳng hạn x[5], a[2,3]v. v.)

I. MẢNG MỘT CHIỀU

Một khai báo mảng phải được đặc tả bởi hai khía cạnh của mảng: kiểu các phần tử của mảng và kiểu của chỉ số. Khai báo mảng một chiều trong PASCAL có dạng sau:

```
array[kcs] of kft
```

Trong đó array, of là các từ khoá.

kcs là kiểu của chỉ số (integer, miền con, ký tự...)

kft là kiểu của các phần tử của mảng (là một kiểu dữ liệu của PASCAL nhưng không được là kiểu File).

Ví dụ: Mảng a chứa 20 số nguyên có thể khai báo như sau:

```
var
   a: array[1..20] of integer;
```

Hoặc

```
type
    day_so = array[1..20] of integer;
var
    a: day_so;
```

Khi đó trong chương trình các phép toán sau là hợp lệ:

a[1]:=4; a[2]:=2 +a[1]; a[3]:=a[1] +a[2];...

Cách gán giá trị cho biến mảng:

- Nếu có hai mảng a và b cùng kiểu như nhau và ta đã biết giá trị của b thì ta có thể thực hiện phép gán như sau:

  a:=b
- Ta thường gán giá trị cho từng phần tử của một mảng bằng cách dùng lệnh lặp.

Ví dụ: Để nhập một dãy a gồm 100 số nguyên ta có thể dùng nhóm lệnh và khai báo sau:

```
var
    a: array[1..100] of integer;
    i: integer;
    .....
    for i:= to 100 do
    begin
          write('Số hạng thứ',i,'là:'); readln(a[i]);
    end;
```

Sau đây là một số ví dụ minh hoạ cho việc sử dụng mảng để giải quyết một số bài toán:

Ví dụ 1: Viết một chương trình thực hiện các công việc sau:

- Đọc một dãy số bao gồm 50 số thực vào máy thông qua bàn phím;
- Tính tổng các số âm;

Chương trình như sau:

```
Program Tong_am;
var
   i: integer; tong:real;
   a: array[1..50] of real;
BEGIN
   for i:=1 to 50 do
      begin
         write('Số hạng thứ',i,'là:'); readln(a[i]);
      end;
   tong:=0;
```

```
    for i:=1 to 50 do
          if a[i]<0 then
              tong:=tong+a[i];
        writeln('Gia tri tinh duoc la:',tong:10:3);
    readln;
END.
```

Ví dụ 2: Cho một dãy bao gồm n số nguyên dương (n≤ 250). Viết một chương trình thực hiện các công việc sau:

1. Đọc n và dãy số đã cho vào máy;

2. Hãy cho biết dãy đã cho có bao nhiêu giá trị khác nhau và mỗi giá trị đó là giá trị của các số hạng nào của dãy.

Sau đây là chương trình:

```
Program xlday;
var
    n,i,j,k,dem,skn: integer;
            a: array[1..1000] of byte;
            kn; set of byte;
Begin
    repeat
            write('n='); readln(n);
    until (n>0) and (n <=250);
    for i:=1 to n do
        begin
            write('So hang thu',i,'la'); readln(a[i]);
        end;
    kn:=[]; skn:=0;
    for i:= to n do
        if not (a[i] in kn) then
            begin
                kn:=kn+[a[i]]; skn:=skn+1;
            end;
    writeln('Có tất cả:',skn,'giá trị khác nhau');
dem:=0;
for i:= i to n do
```

```
While kn<>[] do
    if a[i] in kn then
      begin
          dem:=dem +1;
          writeln('Giá trị khác nhau thứ',dem,'là:', a[i],
                        'và bằng các số hạng:');
          for j:=i to n do
              if a[j]=a[i] then write (j:4);
                  writeln;
                  kn:=kn-[a[i]];
      end;
  readln;
END.
```

Ví dụ 3: Cho một dãy số thực x1,... x100; hãy sắp xếp dãy đó theo thứ tự tăng dần.

Sau đây là đoạn chương trình thực hiện việc sắp xếp:

```
for i:=1 to 99 do
 for j:= i+1 to do 100 do
        if a[i]>a[j] then
        begin
            tg:=x[i]; x[i]:=x[j]; x[i]:=tg;
        end;
```

Bạn đọc hãy tự hoàn thiện chương trình.

II. MẢNG NHIỀU CHIỀU

Ngoài mảng một chiều PASCAL còn cho phép làm việc với mảng nhiều chiều vì chúng tỏ ra rất có ích; chẳng hạn mảng hai chiều đặc biệt thích hợp cho việc xử lý dữ liệu được sắp thành hàng và thành cột như khi ta làm việc với ma trận chẳng hạn.

Một mảng hai chiều được khai báo như sau:

```
array[kcs1, kcs2] of kft
```

kcs1,kcs2 : là kiểu của chỉ số(integer, miền con, ký tự...)

kft là kiểu của các phần tử của mảng(là một kiểu dữ liệu của PASCAL).

Ví dụ để đưa vào máy một ma trận 3 hàng 4 cột ta cần khai báo một mảng như sau:

```
var
   a: array[1..3,1..4] of real;
```

Ví dụ 4: Chương trình sau đây thực hiện việc cộng ma trận:

```
Program cong_ma_tran;
var
   a,b,c: array[1..4,1..5] of real;
     i,j: integer;
BEGIN
    for i:=1 to 4 do
          forj:=1 to 5 do
                read(a[i,j]);
   for i:=1 to 4 do
          for j:=1 to 5 do
                read(b[i,j]);
    for i:=1 to 4 do
          for j:=1 to 5 do
                c[i,j]:=a[i,j] + b[i,j];
    for i:=1 to 4 do
          for j:=1 to 5 do
                writeln(c[i,j]);
END.
```

Chương 9

DỮ LIỆU KIỂU XÂU KÝ TỰ

• Khai báo một kiểu xâu như sau:

```
type
    ten= string[n];
```

Trong đó ten là một tên do người lập trình đặt, string là từ khoá, còn n báo cho máy biết số ô nhớ (cụ thể là byte) tối đa cần dành sẵn cho một biến thuộc kiểu này.

Ví dụ:

```
type
        str10= string[10];
var
        s1,s2: str10;
```

Hoặc ngắn gọn hơn:

```
var
        s1,s2: string[10];
```

Lúc đó s1, s2 sẽ được cấp phát 1+10 byte bộ nhớ liên tục; ô đầu tiên để lưu trữ độ dài thực tế của xâu, các ô còn lại lưu trữ các ký tự của xâu. Chẳng hạn nếu ta có phép gán:

```
s1:='BAN TIN';
```

thì s1 có dạng sau:

7	B	A	N		T	I	N			

Một xâu có thể coi là một mảng các ký tự, ta có thể truy xuất đến từng ký tự của string giống như truy xuất đến từng phần tử của mảng bằng cách dùng đến chỉ số:

Chẳng hạn nếu s1:= 'BAN TIN' thì

write(s1[1]) sẽ in ra ký tự 'B'

write(s1[5]) sẽ in ra ký tự 'T'

Muốn tính chiều dài của xâu ta dùng hàm Length(s1)

Biểu thức đối với string

a) Phép nối string ký hiệu bởi phép cộng(+), dùng để nối nhiều string thành một. Ví dụ; 'khoa'+'40' có trị là 'khoa40'

b) Các phép toán so sánh:

s1=s2 nếu s1 giống hệt s2

s1<>s2 có trị true nếu s1 khác s2

s1<s2 có trị true nếu tại vị trí đầu tiên xảy ra sai khác, ký tự của s1 nhỏ hơn ký tự của s2; ví dụ 'ABCD'<'ABE'

c) Phép gán:

Có dạng

biến xâu:=biểu thức xâu ký tự;

ví dụ s1:='a' + 'b';

Các thủ tục xử lý xâu

a) Thủ tục length(x) cho độ dài thực tế của xâu x.

b) Thủ tục delete(x,m,n): xoá ký tự của xâu x kể từ ký tự thứ m và xoá n phần tử.

c) Thủ tục insert(y,x,n): chèn xâu y vào xâu x từ vị trí thứ n.

d) Thủ tục Val(x,i,ma): chuyển đổi xâu x thành số nguyên hoặc số thực mà chính xâu x biểu diễn nó; nếu việc chuyển đổi là thành công tham số *ma* nhận giá trị 0, trong trường hợp ngược lại *ma* nhận giá trị thứ tự của ký tự đầu tiên không chuyển đổi được.

e) Thủ tục str(r,x) chuyển đổi số r thành xâu biểu diễn x.

f) Hàm Copy(x,m,n) cho ta n ký tự của xâu kể từ ký tự thứ m.

g) Hàm concat(x,y,...,z) nối tất cả các xâu lại thành một xâu.

h) Hàm pos(y,x) cho ta vị trí đầu tiên của xâu y gặp trong xâu x, nếu không tìm thấy thủ tục pos nhận giá trị bằng 0.

Chú ý: Vì dùng một byte để chứa chiều dài nên string chỉ có thể dài tối đa 255 ký tự.

Ví dụ 5: Lập trình thực hiện các công việc sau:

1. Nhập vào một xâu ký tự.

2. Đếm và in ra màn hình số lượng mỗi loại chữ cái'A','B',...,'Z' theo mẫu sau:

A:3
B :0
.....
Z:6

Sau đây là chương trình:

```
var
    x:string;
    c: char;
    i,d:integer;
BEGIN
    write('doc mot xau:'); readln(x);
    for c:='A' to'Z' do
    Begin
        d:=0;
        for i:=1 to length(x) do
            if c=x[i] then d:=d+1;
                    writeln(c,':',D); readln;
    end;
  readln;
END.
```

Chương 10

DỮ LIỆU KIỂU BẢN GHI (RECORD)

Các kiểu dữ liệu đã xét như mảng, tập hợp chỉ mô tả được một nhóm các phần tử của cùng một kiểu. Trong các bài toán quản lý xuất hiện nhiều đối tượng cần quản lý mà chúng ta phải dùng nhiều kiểu dữ liệu để mô tả chúng. Để phục vụ mục đích này PASCAL có một kiểu dữ liệu phức là kiểu record.

Kiểu dữ liệu record là cấu trúc gồm nhiều thành phần, trong đó các thành phần không nhất thiết phải cùng một kiểu, các thành phần của một record gọi là trường.

I. KHAI BÁO KIỂU RECORD

Một kiểu record được định nghĩa sau từ khoá type theo mẫu sau:

```
type
tbg=record
        t1:k1;
        t2:k2;
        .....
        tn:kn;
        end;
```

Trong đó tbg là một tên do người lập trình đặt ra, t1,t2...,tn là các tên trường còn các k1,k2,....,kn là các kiểu dữ liệu sẵn có của PASCAL hoặc là một kiểu dữ liệu vừa mới định nghĩa ở bên trên.

Ví dụ:

```
type
dia_chi=record
            so_nha: integer;
```

```
            pho   : string[15];
            quan  : string[15];
            th_pho : string[15];
        end;
```

Ta thấy để mô tả một địa chỉ ta phải dùng hai kiểu dữ liệu khác nhau.

Chú ý:

- Nếu các trường trong record thuộc cùng một kiểu ta có thể gộp lại, chẳng hạn như trường hợp trên ta có thể khai báo lại như sau:

```
type
 dia_chi =record
            so_nha:integer;
            pho,quan,th_pho:string[15];
        end;
```

- Các record có thể lồng nhau: chẳng hạn tiếp theo khai báo dia_chi ở trên chúng ta có thể có khai báo sau:

```
 ly_lich =record
            ht:string[25];
            cv:string[15];
            lg:real;
            cho_o: dia_chi;
            n_sinh:record
                        ngay:1..31;
                        thang:1..12;
                        nam:integer;
        end;
```

II. TRUY NHẬP VÀO CÁC TRƯỜNG CỦA MỘT RECORD

Trường của một record đóng vai trò như một biến hay một phần tử của mảng, bởi vậy phép toán nào thực hiện trên các biến thì cũng áp dụng lên trường.

Để truy nhập vào một trường ta dùng cú pháp sau:

tên_bản_ghi.tên_bản_ghi...tên_bản_ghi.tên trường.

Chẳng hạn tiếp theo khai báo kiểu ly_lich trên chúng ta có thể khai báo biến của kiểu này:

```
var
   ng1,ng2: ly_lich;
```
Lúc đó trong chương trình các lệnh sau đây là hợp lệ:
```
   ng1.ht:='Nguyen Van A';
   ng1.cv:='Giam doc';
   ng1.lg:= 1200000;
   ng1.cho_o.so_nha:=45;
   ng1.cho_o.pho:='To Hien Thanh';
   v. v.
```
hoặc read(ng1.ht); read(ng1.cv);

Chú ý: Các biến cùng kiểu record có thể gán giá trị cho nhau; ví dụ:

ng1:=ng2;

Lệnh này giúp chúng ta tránh được một loạt lệnh gán như sau:

ng1.ht:=ng2.ht;
ng1.cv:=ng2.cv;
v.v.

III. LỆNH WITH...DO...

Ta thấy để truy nhập tới một trường ta không những chỉ ra tên của chương trình mà còn chỉ ra tên các bản ghi chứa trường này; điều này gây ra sự mất công và tẻ nhạt trong lập trình. Để khắc phục nhược điểm này ta sử dụng lệnh WITH với cấu trúc như sau:

```
With tên_bản_ghi do
 begin
 .....
 end;
```
Trong khoảng begin...end ta chỉ cần nêu tên các trường là đủ; ví dụ:
```
with ng1 do
   begin
    ht:='Nguyen Van A';
    cv:='Giam Doc';
    lg:=1200000;
    cho_o.so_nha:=45;
    cho_o.pho:='To Hien Thanh';
    .....
   end;
```

Hoặc để đơn giản hơn nữa ta có thể viết:

```
with ng1, cho_o do
   begin
         ht:='Nguyen Van A';
         cv:='Giam Doc';
         lg:=1200000;
         so_nha:=45;
         pho:='To Hien Thanh';
         ....
   end;
```

Bằng những cách trên rõ ràng chúng ta đã có thể truy nhập vào các trường giống như truy nhập vào các biến thông thường.

IV. MẢNG CÁC BẢN GHI

Giả sử chúng ta cần xử lý lý lịch của 100 người, lúc đó tiếp theo khai báo bên trên ta khai báo một mảng:

```
var
   danh_sach: array[1..100] of ly_lich;
```

Mỗi phần tử của mảng chính là một biến, biến này có kiểu dữ liệu là một bản ghi. Như cách khai báo như trên ta có các phép truy nhập sau:

```
danh_sach[1].ht:='Tran Te';
danh_sach[1].cv:='Ky su';
.....
Hay:
with danh_sach[1] do
      begin
            ht:='Tran Te';
            cv:='Ky su';
            .....
      end;
```

Chương 11

DỮ LIỆU KIỂU FILE (TỆP)

Trong những tiết trước ta đã thấy nhiều phương thức tạo lập biến và danh sách. Tuy vậy tất cả các phương pháp này đều chịu một tổn thất: khi tắt máy hay mất điện thì dữ liệu sẽ mất hết. Trong tiết này chúng ta sẽ học cách để vượt qua tổn thất trên nhờ cách tạo lập các file (tệp tin).

File của PASCAL là một cấu trúc dữ liệu gồm nhiều phần tử cùng kiểu được lưu trữ ở thiết bị nhớ ngoài: số phần tử là không hạn chế.

Chúng ta cũng thấy các chương trình khi chạy thường sản sinh ra những khối lượng dữ liệu lớn mà sau này chúng ta sẽ còn sử dụng lại nhiều lần, vì vậy việc lưu trữ chúng là hoàn toàn cần thiết; muốn vậy ta ghi chúng ra bộ nhớ ngoài (ổ đĩa, băng từ...) nhờ cấu trúc file (tệp). Các tệp phân biệt với nhau bởi tên của chúng. Mỗi tệp bao gồm một dãy nhiều phần tử thuộc cùng một kiểu, số lượng về nguyên tắc là không hạn chế. Điều này có nghĩa là số lượng các phần tử không cần xác định khi khai báo; thay vào đó PASCAL theo dõi các thao tác truy nhập file thông qua một cơ chế đặc biệt đó là con trỏ tệp: ở một thời điểm nhất định con trỏ tệp luôn luôn chỉ vào một phần tử xác định, mỗi khi một phần tử nào đó của tệp được đọc từ tệp hoặc ghi lên tệp, con trỏ sẽ tự động chuyển sang phần tử tiếp theo và chỉ có phần tử đang được con trỏ tệp định vị ta mới có thể truy nhập được. Do tất cả các phần tử của tệp đều có độ dài như nhau cho nên ta hoàn toàn có thể tính được vị trí của mỗi phần tử riêng biệt; cũng chính vì thế mà ta có thể chuyển con trỏ tệp tới bất kỳ phần tử nào trong file.

I. KHAI BÁO TỆP

Kiểu tệp được định nghĩa bằng FILE OF <kft> trong đó FILE, OF là các từ khóa còn <kft> là kiểu dữ liệu của các phần tử của nó.

Tên biến tệp được khai báo bằng cách sử dụng một kiểu file đã định nghĩa trước đó hay cũng có thể bằng cách khai báo kiểu trực tiếp trong dòng khai báo, ví dụ:

```
type
    T= file of integer;
    R=file of real;
    nhan_su=record
                    ten:=string[20];
                    n_sinh: integer;
                    lg:real;
            end;
    ho_so= file of nhan_su;
var
    f1,f2,f3:T;
    f4: R;
    f5: ho_so;
Hay:
type
    nhan_su=record
                    ten: string[20];
                    n_sinh:integer;
                    lg:real;
            end;
    ho_so= file of nhan_su;
var
    f1,f2,f3:file of integer;
    f4: file of real;
    f5:ho_so;
```

II. CÁC THAO TÁC TRÊN FILE

- Tạo tệp để ghi dữ liệu.

  Để mở một tệp nhằm lưu cất dữ liệu ta dùng cặp lệnh sau:

```
assign(bt, ten_tep);
rewrite(bt);
```

Trong đó:

assign, rewrite là các từ khoá.

bt là biến tệp đã được khai báo trong mục var còn ten_tep là tên tệp do người lập trình đặt ra (theo quy định của DOS), khi cần thì kèm theo cả đường dẫn.

Ví dụ:

```
assign(f1,'NGUYEN.DAT');
rewrite(f1);
```

Lệnh đầu gán tệp có tên là NGUYEN (đuôi là DAT) cho biến tệp; sau khai báo này về sau mọi thao tác trên f1 sẽ được cập nhật vào tệp trên đĩa với tên là NGUYEN.DAT.

Lệnh thứ hai là mở tệp để ghi; sau khi thực hiện lệnh này trên đĩa có tệp rỗng tên là NGUYEN.DAT, con trỏ tệp chỉ vào đầu file.

Sau khi đã mở tệp để ghi bằng hai lệnh trên ta dùng lệnh write (bt,X) để ghi dữ liệu vào tệp; ở đây X là một danh sách biến và biểu thức có kiểu là kiểu thành phần của file; mỗi giá trị trong danh sách này sẽ lần lượt ghi lên tệp trên đĩa và sau mỗi lần ghi con trỏ tệp sẽ chuyển tới thành phần tiếp theo, ví dụ:

```
write(f1,x);
for i:=1 to 12 do write(f1,i);
```

Cuối cùng sau khi kết thúc các truy nhập vào tệp ta 'đóng' tệp bằng lệnh:

```
close(bt)
```

Lệnh này như đã nói sẽ đóng tệp có tên đã gán cho bt và cập nhập tệp để phản ảnh trạng thái mới của tệp.

Sau đây là một chương trình hoàn chỉnh nhằm tạo ra một tệp với tên là 'NGUYEN.DAT' chứa 10 số tự nhiên đầu tiên.

Ví dụ 1:

```
Program td;
  var
      i: integer;
      f: file of integer;
BEGIN
      assign(f,'NGUYEN.DAT');
      rewrite(f);
      for i:=0 to 9 do
          write(f,i);
```

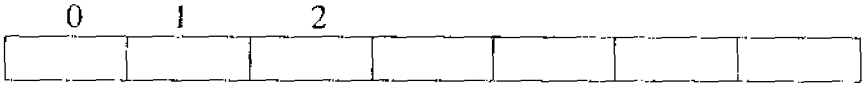
```
    close(f);
END.
```

• Mở tệp để đọc

Một chương trình muốn sử dụng dữ liệu đang lưu trữ trong một tệp tin, trước hết phải thực hiện thao tác mở để đọc như sau:

```
assign(bt,ten_tep);
reset(bt);
```

Lệnh đầu đã được cắt nghĩa; lệnh thứ hai có nghĩa như sau: tệp trên đĩa (có tên là tên_tep) vừa gán cho biến tệp bt được mở ra để sẵn sàng chờ xử lý; con trỏ tệp chỉ vào phần tử đầu tiên của tệp (phần tử đầu tiên mang số hiệu 0).



Sau khi đã mở tệp để đọc ta dùng lệnh sau để đọc các phần tử của tệp vào bộ nhớ trong:

```
read(bt,X);
```

Ở đây x là một hay nhiều kiểu biến thuộc kiểu thành phần của file, các biến này đặt cách nhau bởi một dấu phảy; mỗi biến được đọc lần lượt từ tệp trên đĩa, sau mỗi lần đọc như vậy con trỏ file sẽ tự động chuyển tới phần tử tiếp theo. Ví dụ giả sử file 'NGUYEN.DAT' đã được mở lúc đó lệnh:

read(f,x,y,z) sẽ đọc 0 vào x, 1 vào y và 2 vào z; cuối cùng sau khi đã tiến hành các thao tác ta phải đóng tệp bằng lệnh close(f).

Trong khi tiến hành đọc một tệp người ta hay dùng hàm EOF(bt) để kiểm tra xem con trỏ tệp đã chỉ vào cuối tệp hay chưa. Đây là một hàm kiểu boolean cho kết quả là TRUE nếu con trỏ tệp đã chỉ vào vị trí kết thúc của tệp, trong trường hợp ngược lại nó trả về giá trị FALSE.

Sau đây là một chương trình sử dụng hàm EOF để đọc tệp 'NGUYEN.DAT' và tính tổng của tất cả các phần tử của nó:

Ví dụ 2:

```
Program tddoc;
        var
         i,s: integer;
         f: file of integer;
BEGIN
        assign(f,'NGUYEN.DAT');
        reset(f);
```

```
        s:=0;
        while not eof(f) do
         begin
                read(f,i);
                s:=s+1;
         end;
    close(f);
END.
```

III. TRUY NHẬP TRỰC TIẾP VÀO TỆP

Như đã thấy trong một thời điểm chỉ có một phần tử của tệp có thể được truy xuất đó chính là phần tử mà con trỏ tệp chỉ vào, cho nên ta hoàn toàn có khả năng truy nhập vào bất cứ phần tử nào của tệp bằng cách đặt con trỏ tệp chỉ đúng vào phần tử đó; lệnh:

SEEK(bt,n)

giúp ta thực hiện điều này, ở đây SEEK là từ khoá, bt là biến tệp đại diện cho một tệp tin, n là một biểu thức nguyên; lệnh này đặt con trỏ tệp chỉ đúng vào phần tử mang số hiệu n (lưu ý là phần tử đầu tiên mang số hiệu 0). Ví dụ:

seek(f,5); read(f,i); đọc phần tử thứ 6 trong tệp vào biến i.

Bây giờ giả sử rằng ta đã lưu trữ 10 số thực vào một tệp có tên là 'THUC.DAT'; chương trình sau đây sẽ kiểm tra xem các phần tử của nó có đúng không, nếu sai sẽ sửa.

Ví dụ 3:

```
Program sua_tep;
  var
      f:file of real;
      tl: char;
      j: integer;
      i:real;
  BEGIN
      assign(f,'THUC.DAT'); reset(f);
      j:=0;
      while j<=9 do
          begin
              seek(f,i); read(f,i);
```

```
            writeln('phần tử số hiệu',j,'là:',i:10:2);
            write('có sửa không?(C/K)'); readln(tl);
            if tl in ['c','C'] then
                begin
                    seek(f,j); write('đọc vào giá trị mới:');
                    read(i); write(f,i);.
                end;
            j:=j+1;
        end;
    close(f);
END.
```

IV. CÁC HÀM VÀ THỦ TỤC XỬ LÝ TỆP

• Các hàm

1) filesize(bt) cho ta số lượng phần tử trong tệp, khi tệp rỗng cho ta giá trị 0.

2) filepos(bt) cho ta vị trí hiện tại của con trỏ tệp.

• Các thủ tục

1) rename

Cách viết: rename(bt,str)

Công dụng: cho phép đổi tên file đang kết hợp với biến bt bằng một tên mới chứa trong xâu str;

2) erase

Cách viết: erase(bt)

Công dụng: xoá tệp đang kết hợp với bt.

Ví dụ 4:

Ta xét bài toán: người ta cần quản lý một lớp có tối đa 60 học sinh. Với mỗi học sinh cần quản lý các thông tin sau đây: họ và tên, tuổi, địa chỉ, điểm toán, điểm văn và phân loại; phân loại được tiến hành như sau:

• Nếu Toán+Văn<=10 thì phân loại 'kém'

• Nếu (Toán+Văn<14) và (Toán+Văn>10) thì phân loại 'Trung bình'

• Nếu (Toán+Văn>=14) và (Toán+Văn<18) thì phân loại 'Khá'

• Nếu Toán+Văn>=18 thì phân loại 'Giỏi'

Sau đây là chương trình:

```
Program ql_lop;
    type
        hs=record
            ht:string[30];
            tuoi: byte;
            dc:string[35];
            t,v: real;
            pl:string[10];
         end;
    var
        f: file of hs;
        n,i,j: byte;
        x:hs;
    begin
        repeat
            write('Đọc số lượng học sinh:'); readln(n);
        until (n>0) and (n<60);
        assign(f,'LOP.DAT'); rewrite(f);
        for i:=1 to n do
            begin
             writeln('Các số liệu về học sinh thứ:',i);
             with x do
                begin
                 write('Họ và tên:');readln(ht);
                 write('Tuổi:'); readln(tuoi);
                 write('Địa chỉ:'); readln(dc);
                 write('Điểm toán:'); readln(t);
                 write('Điểm văn:'); readln(v);
                 if t+v<10 then pl:='kem';
                 if (t+v>10) and (t+v<14) then pl:='trung bình';
                 if (t+v>=14) and (t+v<18) then pl:='khá';
                 if (t+v>=18) then pl:='giỏi';
                end;
             write(f,x);
            end;
            close(f);
    END.
```

Ví dụ 5: Hãy viết một chương trình cho phép sao chép một tệp nguồn bất kỳ sang tệp đích. Tên tệp nguồn và tên tệp đích được đọc từ bàn phím; sau đây là chương trình:

```
Program sao_chep;
   uses Crt;
   var
      bt1,bt2: file of byte;
      ch: byte;
      t1,t2:string[20];
   BEGIN
      write('tep nguon:'); readln(t1);
      write('tep dich:'); readln(t2);
      assign(bt1,t1); reset(bt1);
      if ioresult <>0 then exit;
      assign(bt2,t2); rewrite(bt2);
      while not eof (bt1) do
         begin
               read(bt1,ch);
               write(bt2,ch);
         end;
   close(bt1); close(bt2);
END.
```

Chương 12 .

TEXT FILE (TỆP VĂN BẢN)

Tệp văn bản là tệp mà các phần tử của tệp là các ký tự nhưng được tổ chức thành dòng với độ dài của các dòng không nhất thiết phải bằng nhau; mỗi dòng được kết thúc bằng dấu eoln (end of line), trong PASCAL dấu này hình thành bởi hai ký tự CR (#10) và LF(#13).

File văn bản kết thúc bởi dấu end of file, trong PASCAL chính là tổ hợp ctrl+z(^z) có mã ASCII là 26. Ví dụ đoạn văn bản sau:

Tệp văn bản
Rất hay dùng

được bố trí như sau trong bộ nhớ ngoài:

Tệp văn bản	CRLF	Rất hay dùng	EOF

Chú ý:

• Tệp văn bản được tổ chức theo dòng nên việc ghi và đọc theo dòng có thể thực hiện được nhờ lệnh writeln và readln.

• Tuy tệp văn bản chứa các ký tự nhưng các lệnh writeln, readln và write, read vẫn có khả năng ghi và đọc các dữ liệu kiểu integer, real, string nhờ sự chuyển đổi thích hợp.

I. KHAI BÁO TỆP VĂN BẢN

Các biến tệp văn bản được định nghĩa theo mẫu sau:

```
var
   bt_vb:text;
```

Trong đó bt_vb là một tên, còn text là từ khoá.

II. GHI VÀO TỆP VĂN BẢN

Để ghi vào tệp văn bản ta trước hết ta dùng cặp lệnh mở tệp để ghi:

```
assign(bt,ten_tep);
rewrite(bt);
```

sau đó sử dụng một trong ba thủ tục sau:

1, write(bt,bt1,bt2,...,btn);

2, writeln(bt,bt1,bt2,...,btn);

3, writeln(bt);

Trong đó *bt* là biến file văn bản, các *bti* là các giá trị cần ghi vào tệp văn bản, chúng có thể là các biểu thức kiểu integer, kiểu real, kiểu xâu, kiểu boolean; ví dụ:

```
write(f,'xau ky tu', i+3,89.45*12);
```

Dạng thứ hai chỉ khác dạng đầu một chi tiết sau khi đưa các giá trị vào tệp nó chèn thêm ký hiệu *eoln* vào tệp.

Dạng thứ ba chỉ thực hiện việc đưa dấu hết dòng vào tệp.

III. ĐỌC DỮ LIỆU TỪ TỆP VĂN BẢN

Thao tác đầu ta phải mở tệp để đọc bằng cặp lệnh:

```
assign(bt,ten_tep); reset(bt)
```

Sau đó chỉ có thể đọc tuần tự bằng các lệnh sau:

1, read(bt,b1,b2,...,bn);

2, readln(bt,b1,b2,...,bn);

3, readln(bt);

Ở đây bt là biến tệp văn bản, các *bi* là các biến kiểu xâu, kiểu ký tự, kiểu integer, kiểu real.

Lệnh dạng 1: sau khi đọc hết các biến con trỏ tệp không chuyển xuống dòng tiếp theo.

Lệnh dạng 2: sau khi đọc hết các biến con trỏ tệp chuyển xuống dòng tiếp theo.

Lệnh dạng 3: chuyển con trỏ tệp xuống dòng tiếp theo.

Người ta xây dựng hai hàm kiểu boolean để kiểm tra thời gian đọc và ghi dữ liệu đó là hai hàm EOF và EOLN.

Chẳng hạn để việc đọc tiếp tục cho đến khi gặp dấu hết dòng ta dùng lệnh:

while not EOLN(bt) do...

và để việc đọc tiếp tục cho đến hết tệp ta sử dụng lệnh:

while not EOF(bt) do...

IV. MỞ TỆP VĂN BẢN ĐỂ GHI THÊM VÀO CUỐI TỆP (LỆNH APPEND)

Đây là lệnh chỉ dùng riêng cho tệp văn bản, cú pháp như sau:

```
assign(bt,ten_tep);
append(bt);
```

Cặp lệnh này mở tệp tin đã có theo lối ghi và định vị con trỏ tệp ở cuối tệp tin khi đó lần dùng lệnh write (hay writeln) sẽ thêm văn bản vào cuối tệp.

Ví dụ 1: Giả sử ta có tệp văn bản 'SO.TXT' chứa 50 số nguyên từ 1 đến 50; lúc đó để thêm văn bản vào cuối tệp ta dùng chương trình sau:

```
Program Them;
 var
    f: text;
    i:integer;
BEGIN
    assign(f,'SO.TXT');
    append(bt);
    writeln(f);
        for i:=51 to 100 do writeln(f,i);
    close(f);
END.
```

Để kết thúc chúng tôi giới thiệu một chương trình cho phép đếm xem một tệp văn bản có bao nhiêu dòng:

```
Program doc_tep_van_ban;
 var
    f:text;
    d:integer;
BEGIN
    assign(f,'c:config.sys');
    reset(f);
    d:=0;
    while not eof(f) do
```

```
        begin
                readln(f);
                d:=d+1;
        end;
    writeln(d);
    readln;
END.
```

V. CÁC TẬP TIN THIẾT BỊ CỦA DOS

Trong TURBO PASCAL các tệp tin chia làm hai nhóm:

1. Các tệp tin trên đĩa như đã trình bày.

2. Các tệp tin thiết bị: chính là các thiết bị vào ra như bàn phím, màn hình, máy in, các cổng; sau đây là bảng tên các thiết bị trong TURBO PASCAL:

Tên	Thiết bị	Vào	Ra
CON	Màn hình, bàn phím	x	x
LPT1	Cổng máy in số 1		x
LPT2	Cổng máy in số 2		x
LPT3	Cổng máy in số 3		x
LST	Cổng máy in số 1		x
PRN	Cổng máy in số 1		x
LST	Cổng máy in số 1		x
COM1	Cổng nối tiếp số 1	x	x
COM2	Cổng nối tiếp số 2	x	x

Chương trình sau đây cho phép ta in tất cả các số nguyên từ 1 tới 50:

Ví dụ 2:

```
Program In_so;
 var
    may_in:text;
    i:integer;
BEGIN
    assign(may_in,'PRN'); {nối máy in vào biến tệp}
    rewrite(may_in);
```

```
    for i:=1 to 50 do
            writeln(may_in,i);
            writeln(may_in,#12); {đẩy giấy khỏi máy in}
    close(may_in);
END.
```

Chương 13

CHƯƠNG TRÌNH CON (FUNCTION VÀ PROCEDURE)

Trong chương trình, chúng ta thường thấy có một hiện tượng như có những đoạn chương trình nào đó được thực hiện lặp đi lặp lại nhiều lần, tuy dữ liệu có khác nhưng bản chất các công việc lại giống nhau. Như vậy phải chăng ta có thể viết gộp những đoạn chương trình đó lại thành một chương trình con mà khi cần chỉ việc truyền dữ liệu cho nó? Tư tưởng đó cũng dẫn chúng ta tới việc chia một chương trình lớn thành nhiều phần nhỏ rồi giải quyết từng phần; sau đó sẽ ráp nối chúng lại là sẽ hoàn tất một chương trình lớn, các chương trình nhỏ này chính là các chương trình con.

Như vậy khái niệm chương trình con cho ta hình ảnh một dây chuyền sản xuất mang tính công nghiệp cao, mỗi công đoạn thực hiện một phần sản phẩm, cuối cùng chúng sẽ được ráp nối lại với nhau và sản phẩm sẽ ra đời.

Trong PASCAL có hai loại chương trình con: function và procedure; như đã nói trước đây: chương trình con phải khai báo ở mục khai báo cuối cùng của phần khai báo.

I. HÀM (FUNCTION)

1. Cấu trúc của một hàm

Cấu trúc của một hàm như sau:

function ten_ham(danh sach cac tham so): kieu;

..... > | phần khai báo

-
Begin
-
..... > | các lệnh của chương trình con khai báo
-
End;

Nhận xét: Về phương diện cấu trúc giống như chương trình chính.

Ví dụ 1:

```
function gia_thua (n:integer):real;
    var
        i:integer; k:real;
 Begin
    i:=0;
    k:=1;
while i<=n do
    begin
        i:=i+1;
        k:=k*i;
    end;
    gia_thua:=k:
End;
```

Nhận xét:

• Lệnh cuối cùng của function bao giờ cũng là lệnh gán kết quả tính toán cho tên hàm.

• function bao giờ cũng cho ta một giá trị.

Chúng ta xét thêm một ví dụ nữa:

Ví dụ 2:

```
function USCLN(x,y:integer): integer;
 var
    sd:integer;
Begin
    while y<> 0 do
    begin
        sd:=x mod y;
        x:=y;
        y:=sd;
```

```
end;
uscln:=x;
End;
```

Ví dụ 3:

```
function nguyen_to(x:integer);
 var i: integer;
    begin
          i:=2;
          while (x mod i<>0) do i:=i+1;
          if i=x then nguyen_to:=true
                else nguyen_to:=false;
    end;
```

2. Lời gọi hàm

Một khi đã khai báo một hàm chúng ta có thể gọi nó như gọi một hàm chuẩn của PASCAL.

Chẳng hạn một lời gọi hàm USCLN vừa định nghĩa ở trên có thể có dạng như sau:

```
write(uscln(6,27));
```

Chú ý :

Trong TURBO PASCAL kiểu của hàm có thể là real, integer, char, boolean, string, con trỏ, nhưng phải ở dạng tên kiểu.

Ví dụ: Các khai báo sau đây là không hợp lệ:

```
function f(x:integer): 0..65;
function g(x:integer): string[50];
```

mà cần phải khai báo như sau:

```
type
   k1=0..65;
   str50=string[50];
```

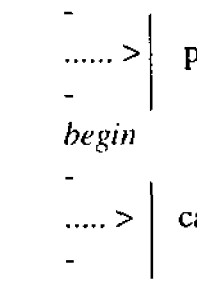
sau đó ta sẽ khai báo các chương trình con như sau:

```
function f(x:integer): k1;
function g(x:integer): str50;
```

II. THỦ TỤC (PROCEDURE)

Thủ tục có dạng:

procedure ten_thu_tuc(ds);



Trong đó *ten_thu_tuc* là một tên do người lập trình đặt theo quy định của TURBO PASCAL, còn *ds* là danh sách các nhóm tham số hình thức; các nhóm cách nhau bởi dấu chấm phảy ';', trong một nhóm các tham số có cùng kiểu cách nhau bởi dấu phảy ',', cuối cùng là dấu hai chấm ':' và kết thúc là tên kiểu dữ liệu.

Chú ý: ds có thể không tồn tại.

Ví dụ 4:

```
Procedure vào(var x,y,z:real);
 var
    tl:char;
Begin
 repeat
            write('Vao x,y,z: '); readln(x,y,z);
            write('Có sửa không: '); readln(tl);
 until tl in ['K','k']
End;
```

Chương trình con này có nhiệm vụ vào dữ liệu qua bàn phím nếu sai thì sửa lại.

Chú ý: Trong TURBO PASCAL thì kiểu của các thông số hình thức khai báo trong danh sách các nhóm tham số phải là một tên kiểu, vì vậy các khai báo sau là không hợp lệ:

```
    procedure abc(a1: array[1..10] of integer);
    procedure x1(var s: string[20]);
```

mà phải khai báo như sau:

```
type
  mang=array[1..10] of integer;
   xau=string[20];
```

sau đó:

```
procedure abc(a1:mang);
procedure x1(var s:xau);
```

III. LƯU Ý VỀ PHONG CÁCH LẬP TRÌNH

• Khi chỉ cần tính một giá trị thì nên dùng function.

• Thường thì chương trình con giải quyết một công việc nào đó theo một tham số, các giá trị nhập (tham số thực) được cung cấp qua tham trị, các giá trị xuất qua các tham biến hoặc qua kết quả của hàm; do đó:

❖ Trong chương trình con thường không có lệnh read(), readln() vì các giá trị nhập được cung cấp cho các tham số hình thức khi gọi chương trình con.

❖ Trong các chương trình con cũng thường không có lệnh write() và writeln(), vì các giá trị xuất thường được xuất ra dưới dạng giá trị của hàm hay qua tham biến.

❖ Các lệnh xuất, nhập thường để trong chương trình chính.

❖ Tuy nhiên khi chúng ta sử dụng chương trình con để làm rõ những giai đoạn trong một công việc lớn hoặc kiểm tra một việc gì đó thì trong chương trình con thường có các lệnh xuất nhập riêng cho bản thân nó.

Sau đây là một ví dụ hoàn chỉnh:

Ví dụ 5:

```
Program pt_bac2;
 var
          x,y,z:real;
 procedure cach_giai;
    var
          delta, x1,x2: real;
          procedure delta_khong_am;
                begin
           x1:= (-b+sqrt(delta))/(2*a);
           x2:= (-b-sqrt(delta))/(2*a);
    write(x1:5:2,x2:5:2);
                end;
procedure delta_am;
   begin
    write('Khong co nghiem thuc');
```

```
    end;
 begin
delta:=b*b-4*a*c;
if delta>=0 then delta delta_khong_am
 else delta_am;
    end;
BEGIN
 readln(x,y,z);
 cach_giai(x,y,z);
 readln;
END.
```

Nhận xét: Ta thấy tất cả có ba chương trình con. Nhìn vào chương trình ta thấy: chương trình chính gọi chương trình con *cach_giai*, đến lượt nó chương trình con *cach_giai* lại gọi chương trình con của nó là *delta_am* hay *delta_khong_am* tuỳ thuộc vào các giá trị của biệt thức delta.

Nguyên tắc hoạt động như sau: *gọi chương trình con ở lệnh thứ n, sau khi thực hiện xong chương trình con lại quay về lệnh thứ n.*

Ở ví dụ trên, khi gặp lệnh readln(x,y,z) ta phải đưa vào máy qua bàn phím ba giá trị x,y,z; sau đó chương trình chính gọi chương trình con *cach_giai* và truyền cho a,b,c ba giá trị tương ứng x,y,z vừa nhập vào; nhờ đó mà chương trình con *cach_giai* tính được delta; sau đó tuỳ theo giá trị của delta mà gọi *delta_am* hay *delta_khong_am*; cuối cùng khi đã thực hiện một trong hai procedure vừa nói thì hệ thống trở lại lệnh sau lệnh gọi *cach_giai*, gặp lệnh readln, lệnh này dừng màn hình cho ta xem kết quả, để kết thúc ta gõ phím enter.

IV. CÁCH TRUYỀN THAM SỐ

Để đề cập tới vấn đề này ta xét một bài toán như sau: Cho 3 số thực, hãy viết một chương trình cho phép tìm số lớn nhất của ba số trên rồi in ra kết quả.

Ví dụ 6:

```
Program so_lon;
 var
    n1,n2,n3,l,m:real;
 procedure so_sanh(a,b:real ;var max:real);
```

```
    begin
    if a>b then max:=a
  else max:=b;
    end;
BEGIN
 readln(n1,n2,n3); l:=0;m:=0;
 so_sanh(n1,n2,l);
 so_sanh(n3,l,m);
 writeln('Số lớn nhất là: ',m);
END.
```

Chỉ với một ví dụ đơn giản như trên ta đã thấy việc sử dụng chương trình con có tham số là cách cho phép một thủ tục được gọi nhiều lần trong một chương trình với các kết quả khác nhau tuỳ theo giá trị thật được truyền cho các tham số khi gọi; những tham số này được gọi là các tham số hình thức. Danh sách các tham số hình thức được khai báo trong cặp ngoặc đơn ngay sau tên của thủ tục theo quy tắc sau:

1. Các tham số cùng kiểu được khai báo trong cùng nhóm, cách nhau bởi một dấu phảy.

2. Các nhóm tham số hình thức cách nhau bởi dấu chấm phảy.

Khi chương trình con được gọi các giá trị thật mới được truyền vào thay thế cho tham số hình thức vì vậy hai danh sách tham số (hình thức và thật)phải phù hợp với nhau về cả ba phương diện: số lượng, kiểu và thứ tự.

Có hai cách truyền tham số cho chương trình con:

1. Truyền bằng trị thông qua tham trị.

2. Truyền bằng biến thông qua tham biến.

Sau đây chúng ta lần lượt khảo sát hai cơ chế này:

• Truyền bằng biến:

Những tham số hình thức thuộc loại này được khai báo trong chương trình con sau từ khoá var, các tham số thật của chúng phải là các biến; các tham số hình thức loại này gọi là tham biến; các giá trị thật truyền cho chúng có thể bị thay đổi trong chương trình con và khi ra khỏi chương trình con sự thay đổi này vẫn còn hiệu lực (có nghĩa là khi ra khỏi chương trình con tham biến vẫn giữ nguyên giá trị cuối cùng mà nó nhận được).

• Truyền bằng trị:

Những tham số hình thức thuộc loại này cũng khai báo ngay ở cặp vòng đơn sau tên chương trình con nhưng không có từ khóa var đứng trước; các giá trị thật mà nó nhận được từ chương trình mẹ có thể thay đổi trong chương trình con, nhưng trong mọi trường hợp điều này không làm thay đổi giá trị của tham số thật trong chương trình mẹ.

Quay trở lại chương trình trên ta thấy: ở lần gọi đầu: so_sanh(n1,n2,l) l được truyền bằng biến, điều này có nghĩa là khi ra khỏi chương trình con l sẽ nhận giá trị mới nhận được trong chương trình con; chính vì vậy mà nhờ lần gọi thứ hai: so_sanh(n3,l,m) ta mới thu được kết quả mong muốn m=max{n1,n2,n3}.

Vậy cái gì xảy ra khi ta khai báo thủ tục như sau:

procedure so_sanh(a,b,max: real)?

Ta thấy ngay sau lần gọi đầu, so_sanh(n1,n2,l) l vẫn giữ giá trị ban đầu của nó (l=0), do đó kết quả sẽ nằm ngoài sự mong đợi của chúng ta.

Để hiểu rõ hơn về vấn đề này ta xét thêm một ví dụ kinh điển sau:

Ví dụ 7:

```
Program cach_truyen;
 var
    x,y: integer;
 procedure vi_du(x1:integer, var y:integer);
            begin
                    x1:=x1+1;
                    x2:=x2+2;
                    writeln(x1,x2);
            end;
    BEGIN
            x:=0; y:=5;
            vi_du(x,y);
            write(x,y);
    END.
```

Khi thực hiện chương trình ta thu được kết quả như sau:

1 6 (do chương trình con in ra)

0 6 (do chương trình chính in ra)

Sở dĩ như vậy là vì x1 là tham trị còn x2 là tham biến. Tóm lại, khi truyền một giá trị cho tham trị thì giá trị được truyền không thay đổi, còn khi truyền một giá trị cho tham biến thì giá trị đó sẽ bị thay đổi.

V. PHÂN BIỆT THAM TRỊ VÀ THAM BIẾN

• Tham trị:

* Không có từ khoá var đứng trước khai báo.

* Được cấp một ô nhớ riêng khi chương trình con được gọi và bị xoá bỏ khi chương trình con kết thúc.

* Tham số thật tương ứng là một biểu thức.

* Tham trị thực ra là một biến cục bộ nhận giá trị khởi đầu là trị của tham số thật tương ứng.

* Những thay đổi của tham trị không gây ảnh hưởng tới giá trị của tham số thật tương ứng.

• Tham biến:

* Đi sau var trong khai báo.

* Tham số thật tương ứng phải là biến.

* Thực chất của việc truyền tham số theo biến là một sự truyền địa chỉ.

* Những thay đổi trên tham biến thực chất là được thực hiện trên tham số thật tương ứng.

Để kết thúc ta lưu ý thêm mấy điểm sau:

- Sau khi chương trình con được thực hiện xong mọi biến cục bộ (kể cả tham trị) đều bị xoá bỏ trong bộ nhớ.

- Khi cần truyền tham số một cấu trúc dữ liệu lớn (mảng chẳng hạn) cho một chương trình con chúng ta nên sử dụng kỹ thuật theo biến, vì nếu truyền theo trị ta sẽ tốn thêm bộ nhớ để tạo bản sao và thời gian cần thiết để tạo bản sao đó.

VI. VẤN ĐỀ TẦM VỰC

Trong TURBO PASCAL khi nói đến một khối ta hiểu đó là một chương trình hay một chương trình con.

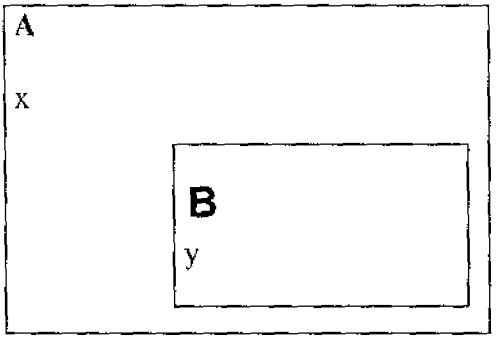
Trong một khối các đối tượng (nhãn, biến, kiểu, hằng, chương trình con) cần được khai báo trước khi sử dụng. Khai báo một đối tượng là xác nhận sự tồn tại của nó trong một khối.

Vậy một vấn đề cần quan tâm là: một đối tượng có thể nhận biết được ở những nơi nào? Ví dụ ta có khối A chứa khối B và x là biến của A còn y là của B, lúc ấy:

1.Trong B có quyền truy xuất x không?

2.Trong A có thể truy xuất y không?

Để giải quyết vấn đề này ta đưa vào định nghĩa sau: tầm vực của một đối tượng là vùng nó được biết tới, có thể được sử dụng , ta có quy tắc sau:



Tầm vực của một đối tượng trải ra từ chỗ nó được khai báo cho đến hết khối mà khai báo đó thuộc về, kể cả mọi khối trong khối đó. Ngoại trừ các trường hợp sau:

* Trường hợp có khai báo lại trong một khối con, tức là nếu khối A chứa khối B và trong cả hai đều có khai báo x thì trong khối B chỉ có thể truy xuất đến đối tượng x mang tên của chính nó (x của A và của B không phải là một đối tượng, chúng chỉ trùng tên); ta nói đây là một hiện tượng bị "che".

* Trường hợp nhãn và goto: nếu trong một khối có lệnh goto X thì nhãn X phải được khai báo ngay ở trong khối đó.

Sau đây ta xét một vài trường hợp cụ thể để làm rõ vấn đề này:

1. Vấn đề tầm vực của một chương trình con

Như ta đã thấy một chương trình có thể có nhiều chương trình con và trong mỗi chương trình con đến lượt nó có thể khai báo nhiều chương trình con khác; vậy đối với một chương trình con thì nơi nào có thể gọi nó thực hiện? Đó chính là vấn đề tầm vực của chương trình con.

Để dễ dàng trong việc cập nhật vấn đề này, ta xét ví dụ sau:

```
Program A0;
var
procedure B1;
```

```
var x.y: interger;
procedure B2;
        var x: char;
                begin
                :
                end;
procedure B3;
        var...
                begin
                :
                end;
        Begin {B1}
                :
        End; {B1}
Procedure C1;
    var...
        Begin {C1}
        :
        End; {C1}
BEGIN
:
END.
```

Chúng ta hãy trả lời mấy câu hỏi sau:

• Thủ tục B2 có thể được gọi ở những đâu?

Vì B2 khai báo trong B1 nên B2 chỉ có thể được truy xuất ở bên trong B1; nghĩa là:

- Có thể gọi B2 từ một vị trí trong thân B1, trong thân B3 và cả trong chính thân B2. (gọi đệ quy);

Nói tóm lại tầm vực của B2 là toàn bộ B1 (nói nôm na là tầm vực của một chương trình con là toàn bộ cha của nó).

• Thủ tục B1 có thể truy xuất ở những đâu?

Tầm vực của B1 là A0 cho nên:

- A0 có thể gọi B1.
- C1 có thể gọi B1.
- B1 có thể gọi B1; nghĩa là:

+ Trong thân B1 có thể gọi B1.

+ Trong thân các chương trình con của B1 (B2,B3) có thể gọi B1.

• Tương tự C1 có thể được truy xuất từ trong:

- A0
- C1
- B1...

2. Vấn đề tầm vực của biến; biến toàn cục, biến cục bộ, biến không cục bộ

Trở lại với ví dụ trên chúng ta có:

Những biến khai báo trong B1 là những biến cục bộ trong thân B1; chúng được biết đến trong toàn bộ B1; do vậy chúng có thể được truy xuất trong B2 và B3.

1) Trong thân của A0 và thân của C1 những biến cục bộ của B1 không được biết đến.

2) Những biến khai báo trong A0 gọi là biến toàn cục; những biến này có thể truy xuất ở khắp mọi nơi.

Kết luận: Tầm vực của biến là phạm vi khối khai báo mà nó thuộc về.

3) Tuy nhiên nếu trong B2 có khai báo một biến trùng tên với một biến của B1 thì trong B2 chỉ biết đến biến mà chính nó khai báo. (ở ví dụ bên trong B2 x luôn luôn được hiểu là biến cha).

Sau đây là một số ví dụ giúp chúng ta nhận thức rõ hơn về các loại biến:

Ví dụ 8:

```
Program VD1;
var x:integer;
 procedure p1;
    var x:integer;
    begin
            x:=1;
    end;
BEGIN
    x:=0;
    p1;
    writeln(x);
END.
```

Khi chạy chương trình ta sẽ thu được giá trị 0 trên màn hình, vì x của chương trình chính và của p1 là khác nhau; sau khi p1 chạy xong, các biến của nó bị xoá khỏi bộ nhớ, vì vậy không ảnh hưởng gì đến cha của nó.

Ví dụ 8:

```
Program VD2;
   var x:integer;
    procedure p2;
          begin
                x:=1;
           end;
BEGIN
    x:=0;
    p2;
    writeln(x);
END.
```

Khi chạy VD2 giá trị 1 sẽ được in ra màn hình.

Lời giải thích xin nhường cho bạn đọc.

Lưu ý: Trong thân chương trình con ta nên hạn chế việc truy xuất đến các biến không cục bộ. Sự làm thay đổi các biến cục bộ trong thân chương trình con gọi là hiệu ứng lề; hiệu ứng lề làm chương trình kém trong sáng; khiến ta khó quản lý các biến và làm cho chương trình chạy sai, nhất là khi ta dùng biến không cục bộ làm điều khiển vòng lặp.

Ví dụ 10:

```
Program VD3;
   var x:integer;
   procedure p3 (x:integer);
   begin
          x:=1;
   end;
BEGIN
          x:=0;
          p3(x);
          writeln(x);
END.
```

Khi chạy chương trình VD3 giá trị 0 sẽ được in ra. Bởi vì khi gọi p3(x) thì tham trị x của p3 được cung cấp giá trị khởi đầu là trị của tham số thật x (tức là 0); sau đó phép gán x:=1 được thực hiện và p3 hoàn thành nhiệm vụ. Tham trị x thực chất là một biến cục bộ nên bị xoá bỏ khi thoát khỏi chương trình con, vì vậy trị của biến toàn cục không hề bị thay đổi và vẫn là 0.

Ví dụ 11

```
Program VD4;
   var x: integer;
   procedure p4 (var x:integer);
         begin
               x:=1;
         end;
BEGIN
         x:=0;
         p4(x);
         writeln(x);
END.
```

Khi chạy chương trình giá trị 1 được in ra. Tại sao? Xin nhường lời giải thích cho bạn đọc.

Ví dụ 12: Viết thủ tục cho phép sắp xếp một dãy số thực bao gồm tối đa 100 phần tử theo thứ tự tăng dần:

```
Procedure sx_tang (var x:day100;nh:integer);
   var
         i,j:integer; tg:real;
    begin
         for i:=1 to nh-1 do
               for j:=i+1 to nh do
                     if a[i] > a[j] then
                           begin
                                 tg:=a[i]; a[i]:=a[j]; a[j]:=tg;
                           end;
   end;
```

Ví dụ 13: Viết thủ tục cho phép tính tích hai ma trận:

```
procedure nhan(a,b:mt; var c:mt; m,n,k:integer);
```

```
var
        i,j,t: integer;
begin
        for i:=1 to m do
                for j:=1 to k do
                        begin
                                c[i,j]:=0
                                for t:=1 to n do
                                        c[i,j]:=c[i,j]+ a[i,t]*b[t,j];
                        end;
end;
```

PHẦN BÀI TẬP VÀ MỘT SỐ BÀI MẪU

• Thuật toán

Bài 1: Trình bày thuật toán tìm mượn một quyển sách ở thư viện KHOA HỌC, biết tên tác giả, nhớ không chính xác tên sách.

Thuật toán 1:

Bước 1: Đến thư viện Khoa học, tìm đến tủ phân loại sách theo tên tác giả. Theo cách xếp thứ tự của từ điển tìm hộp phích có chữ cái trùng với chữ đầu tên tác giả.

Bước 2: Rút hộp phích này, theo thứ tự từ điển tìm tên tác giả.

Bước 3: Nếu không tìm thấy tên tác giả thì sang bước 10.

Bước 4: Theo thứ tự từ điển, tìm tên các sách của tác giả này.

Bước 5: Nếu không tìm thấy sách thì sang bước 10.

Bước 6: Ghi lại tên sách, tên tác giả, mã sách vào phiếu mượn.

Bước 7: Đưa cho nhân viên thư viện, chờ trả lời.

Bước 8: Nếu trả lời không còn thì sang bước 10.

Bước 9: Cầm sách về sử dụng.

Bước 10: Kết thúc.

Bài 2: Tìm tất cả tổ hợp chập M của N phần tử.

Thuật toán 2:

Bước 1: Đọc N và M vào.

Bước 2: Gán $C_i = i$ với i = 1, 2, ..., M.

Bước 3: Đưa ra dãy $C_1, C_2, ..., C_M$.

Bước 4: Với i=M, M-1,..., 1 tìm i sao cho $C_i \neq N - M+i$. Nếu tìm được i thì sang bước 5, không thì sang bước 7.

Bước 5: Gán $C_i = C_i+1$, $C_j = C_{j-1} +1$ với $j = i+1, i+2,..., M$.

Bước 6: Quay về bước 3.

Bước 7: Kết thúc.

Bài 3: Nêu thuật toán dưới dạng sơ đồ khối, tính căn bậc hai một số dương A với sai số nhỏ hơn ε theo công thức lặp sau đây:

$$X_{n+1} = (X_n + A/X_n) \text{ với } X_0 = 1$$

Bài 4: Cho dãy số X_1 , X_2 , ... , X_n. Trình bày thuật toán sắp xếp lại dãy này theo thứ tự tăng dần.

Thuật toán 3:

Bước 1: Nhập dãy $X_1, X_2, ..., X_n$.

Bước 2: Gán i =1.

Bước 3: Gán j = i+1.

Bước 4: Kiểm tra $X_j < X_i$. Nếu đúng thì sang bước 5, nếu không thì sang bước 6.

Bước 5: Đổi chỗ X_i và X_j .

Bước 6: Gán j =j+1.

Bước 7: Lặp lại từ bước 4 cho đến khi j >N.

Bước 8: Gán i = i+1.

Bước 9: Lặp lại từ bước 3 cho đến khi i = N.

Bước 10: Kết thúc.

Bài 5: Trình bày thuật toán dưới dạng sơ đồ khối, tìm trung bình cộng của N+1 giá trị X_i nằm cách đều nhau trên đoạn [A,B] , (X_0 =A , X_N =B).

• Chỉ thị gán giá trị, thủ tục vào ra dữ liệu

Bài 6: Tính và đưa ra màn hình giá trị của biểu thức A sau đây:

$$A= (x^3 + \sin(b) - e^{0.19238}) / (5 + e^b + c^{(0.20345 + x)}).$$

Trong đó x = -1,5172

$$b = 2^x + x - 31,769$$

$$c = \text{Log}_7 (x^4 + 5^x) + \text{Ln} (x^2 + 5^b).$$

Hướng dẫn: - Trong PASCAL để tính luỹ thừa $f = a^b$ (a>0) ta viết:

$$f = \exp (b * \ln(a)).$$

- Hàm mẫu ln(x) cho ta lnx, vì vậy khi gặp logarit cơ số khác e ta phải biến đổi theo công thức: $\text{Log}_a x = \ln (x)/\ln(a)$.

Program c3b6;

```
Const
   x = -1.5172;
var
   a,b,c: real;
```

```
begin
    b:= exp(x * ln(2)) + asb(x) / x * exp(5 * ln(x)) - 31.769;
    c:= (ln(sqr(sqr(x)) + exp(x* ln(5))) / ln(7) + ln(x*x + exp( b* ln(5)));
    a:= (x * sqr(x) + sin (b) - exp(0.19238)) / (5 + exp(b) +
    abs(c) / c * exp((0.20345 + x) * ln(c));
    writeln(' a=',a);
    writeln('b=',b);
    writeln('c=',c);
   readln
end.
```

Bài 7: Lập chương trình đọc toạ độ Đề-các 3 điểm A, B, C từ bàn phím rồi tính các góc A, B, C, độ dài các cạnh tam giác ABC và đưa kết quả ra màn hình.

Bài 8: Lập chương trình đọc chiều dài các cạnh a, b, c của tam giác ABC rồi tính diện tích, chiều dài các đường cao và đưa kết quả ra màn hình.

- **Chỉ thị rẽ nhánh**

Bài 9: Lập chương trình giải bất phương trình bậc nhất ax + b > 0.

Program c4b1;

```
var a, b : real;
BEGIN
    write (' vao a, b ='); readln(a,b);
if a=0 then
    if b> 0 then
    writeln(' Bat phuong trinh dung voi moi x')
    else writeln(' Bat phuong trinh vo nghiem');
if a>0 then
    writeln(' nghiem bat phuong trinh la x>:' , -b/a:10:2);
if a< 0 then
    writeln (' nghiem bat phuong trinh la x<:',-b/a:10:2);
readln;
 END.
```

Bài 10: Lập chương trình nhập vào hai số a, b rồi tính $y = 15x^2 + x + 72$, trong đó:

$$X = \begin{cases} \dfrac{a+b}{3} + b & \text{nếu a < b} \\ 15.172 & \text{nếu a = b} \\ \dfrac{a-b}{a^2+b^2} & \text{nếu a > b} \end{cases}$$

Bài 11: Lập chương trình nhập các hệ số a, b, c, d, e, f vào máy từ bàn phím rồi giải và biện luận hệ phương trình sau:

ax + by = c
dx + ey = f

```
Program c4b4;
Uses crt;
   { Crt là unit chứa thủ tục xoá màn hình Clrscr}
Var
   a, b, c, d, e, f : real;
   dt, dx, dy : real;
Begin
   Clrscr;
   Write (' a, b, c ='); readln ( a, b, c);
   Write (' d, e, f ='); readln ( d, e, f );
   dt := a*e - d*b;
   dx:= c*e - f*b;
   dy:= a*f - d*c;
   if dt<>0 then
         write (' x= ', dx/dt :10:4, ' y = ', dy/dt:10:4)
   else
         if dx<>0 then
               write('** Hệ phương trình vô nghiệm**')
         else
               write('** Hệ phương trình vô định**');
   Readln;
End.
```

Bài 12: Các nước thành viên của một tổ chức quốc tế được chia thành 5 loại đánh số từ 1 đến 5. Hàng năm mỗi nước phải đóng hội phí theo quy định: nước loại 1, 2, 3, 4 đóng tương ứng 1%, 0.7%, 0.5%, 0.1% tổng thu nhập quốc

dân, nước loại 5 đóng 1 triệu đôla. Lập trình nhập tên nước , loại tổng thu nhập quốc dân (nếu cần), tính và đưa ra màn hình tên nước và số tiền phải đóng góp.

```
Program c4b5;
Uses crt;
Var
   tn, dg: real;
   l: byte;
BEGIN
Clrscr;
Write(' Quốc gia loại : '); readln(l);
   If l<>5 then
         Begin
         Write (' Tổng thu nhập quốc dân hàng năm: ');
               Readln(tn);
         End;
   Case l of
         1: dg:=tn*0.01;
         2: dg:=tn*0.007;
         3: dg:=tn*0.005;
         4: dg:=tn*0.001;
         5: dg:=1E6;
         end;
write (' Tổng số tiền phải đóng góp là:' , dg:10:2,'USD');
readln;
End.
```

- **Các chỉ thị chu trình**

Bài 13: Xác định số lần lặp của chu trình trong chương trình sau:

```
Program Btc5b1;
Var
   a, b : boolean;
   x : real;
BEGIN
   a:=true; b:=a; x: =0;
   While a or b do
```

```
        Begin
                x:=2*x +1;
                if x>0 then
                        begin
                                a:=false;
                        if x<200 then b:=false;
                        end;
                writeln('x=', x);
        end;
END.
```

Bài 14: Xác định số lần lặp của chu trình trong chương trình sau:

```
Program c5b2;
Var
   x, y, z : real;
   n: integer;
BEGIN
   x:=2; y:=x; z:=y; n:=0;
   repeat
          n:=n+1;
          x:=x+z; y:=-y;
          z:=exp(n*ln(z));
   until (n>3) or (x*y*z < 1000);
  writeln(x:10:2,' ',y:10:2);
END.
```

Bài 15: Lập chương trình xếp các dấu * thành tam giác cân n dòng với n đọc từ bàn phím.

```
Program c5b3;
Uses crt;
Var n, i, j : integer;
BEGIN
Clrscr;
Write ('Cho biết số dòng '); readln(n);
Clrscr;
For i:=1 to n do
```

```
    Begin
        Gotoxy (40-i, i);
        For j:=1 to 2*i-1 do write('*');
        Writeln;
    End;
Gotoxy (10,24);
Writeln (' ấn một phím bất kỳ để tiếp tục!');
Repeat until keypressed;
END.
```

Bài 16: Lập chương trình nhập một dãy số nguyên vào từ bàn phím cho đến khi gặp số 0 rồi tính tổng các số dương và trung bình cộng của các số âm.

```
Program c5b4;
Uses crt;
Var i, a, sd, sa, d : integer;
BEGIN
Clrscr;
I:=1; sd:=0; sa:=0; d:=0;
Writeln(' Vào các số nguyên');
Repeat
Write(' số thứ ',i,'='); readln(a);
if a>0 then sd:=sd + a
else if a<0 then
    begin
        d:=d +1;
        sa:=sa +a;
    end;
i:=i+1;
until a=0;
writeln (' Tổng các số dương =',sd);
if d=0 then writeln(' Không có số âm!') else
writeln(' Trung bình cộng các số âm =', sa/d:10:2);
readln;
END.
```

Bài 17: Tìm số nguyên lớn nhất thoả mãn điều kiện:

a. $3n^5 - 317n < 5$

b. $-4n + 151\sqrt{n+3} \geq 0$

c. $7n^3 + 31n^2 - 10^{-6} < 0$

d. $e^n - 1992\lg n \leq 0$

e. $30n^2 - n^n - 1 \geq 0$

```
Program c5b5;
Uses crt;
Var n: longint;
   Ch: char;
BEGIN
Clrscr;
Gotoxy(10,23);
Writeln(' Chú ý : Trường hợp b chương trình chạy',
 'hơi lâu! Xin vui lòng chờ');
gotoxy(1,1);
repeat
n:=1;
   repeat
   write(#13#10' Bạn muốn làm phần nào?','(A/B/C/D/E/K-Kết thúc):');
   ch:=upcase (readkey);
   until ch in ['A','B','C','D','E','K'];
   Case ch of
   'A': while 3*sqr(sqr(n)*n - 317*n < 5 do n:=n+1;
   'B' : while -4*n + 151*sqrt(n) + 3 >=0 do n:=n+1;
   'C': while 7*sqr(n)*n + 81*sqr(n) - 1e6 <0 do n:=n+1;
   'D': while exp(n) - 1992*ln(n)/ln(10) <= 5 do n:=n+1;
   'E' : while -exp(n*ln(n)) + 30*n*n - 1 > 0 do n: =n+1
   end;
if ch<>'K' then writeln('n=',n);
until ch='K';
readln;
END.
```

Bài 18: Cho biết giá trị nào sẽ hiện trên màn hình khi thực hiện chương trình sau:

```
Program c5b6;
Var
   i: integer;
BEGIN
   i:=7;
while i>1 do
begin
            if odd(i) then i:=i*3+1
            else i:=i div 2;
            writeln(i);
   end;
readln;
END.
```

Bài 19: Lập chương trình đọc x, n từ bàn phím rồi tính :

$$S = 1 + \frac{x}{2} + \frac{x^2}{3} + \cdots + \frac{x^n}{n+1}.$$

Bài 20: Lập chương trình đọc x, n từ bàn phím rồi tính.

$$S = 1 + x + \frac{x^2}{n!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

Bài 21: Lập chương trình đọc x, n từ bàn phím rồi tính:

$$S = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \cdots + \frac{(-1)^n x^n}{n!}$$

Bài 22: Lập chương trình phân tích số nguyên n thành thừa số nguyên tố.

Bài 23: Lập chương trình tìm các số có 3 chữ số sao cho số đó bằng tổng lập phương các chữ số của nó.

Bài 24: Lập chương trình liệt kê và đếm các số nguyên tố từ 2 đến n.

```
Program c5b12;
Uses crt;
Var        ok: boolean;
   S, p, n, m, i : integer;
```

```
BEGIN
Clrscr;
Write (' Cho vào n = '); readln(n);
Writeln ('Các số nguyên tố từ 2 đến ',n,' là :');
S:=0;
{ duyệt các số từ 2 đến }
for i:=2 to n do
 begin {duyệt các ước của i từ 2 đến √i }
ok:= true ; p:=2; m:=round(sqrt(i));
    while ok and (p <= m) do
    begin
     if i mod p = 0 then ok:=false;
     p:=p + 1;
    end;
    if ok then { i là số nguyên tố}
    begin
            s:=s + 1;
            write (s:5,') ',i:5);
            if s mod 6 = 0 then writeln;
    end;
 end;
readln;
END.
```

• Kiểu mảng

Bài 25: Lập trình đưa vào từ bàn phím 12 số nhỏ hơn 80 đặc trưng cho năng suất 12 tháng rồi dùng kí tự '*' để vẽ biểu đồ ngang lên màn hình.

```
Program c6b1;
uses crt;
var   nangsuat: array [1..12] of integer;
            i,j: integer;
BEGIN
clrscr;
write(' Vao nang suat cua thang: ');
for i:=1 to 12 do
```

```
    begin
    write('thang thu ',i,'=');        readln(nangsuat[i]);
    end;
clrscr;
writeln('BIEU DO NANG SUAT 12 THANG');
for i:=1 to 12 do
    begin
    for j:=1 to nangsuat[i] do write('*');
    writeln;
    end;
writeln(' An ENTER!'); readln;
END.
```

Bài 26: Lập chương trình nhập ma trận A có m hàng và n cột. Đưa ma trận A ra màn hình để kiểm tra. Các dữ liệu đọc vào từ bàn phím.

```
Program c6b2;
uses crt;
type      matran=array[1..30,1..30] of integer;
var       a: matran;
          i, j, m, n: integer;
BEGIN
clrscr;
Write(' so dong cua ma tran A, m='); readln(m);
write(' So cot cua ma tran A, n ='); readln(n);
writeln(' Vao ma tran A:');
for i:=1 to m do
   for j:=1 to n do
          begin
                write('a[',i, ',' ,j, ']=');
                readln(a[i,j]);
          end;
clrscr;
writeln(' ma tran A nhu sau :');
for i:=1 to m do
```

```
    begin
            for j:=1 to n do write(' ',a[i,j],' ');
            writeln;
    end;
write(' An ENTER de ket thuc! '); readln;
END.
```

Bài 27: Lập chương trình nhập ma trận vuông A có n hàng, n cột vào máy rồi đưa ma trận tam giác dưới và ma trận tam giác của A ra màn hình. Các dữ liệu đưa vào từ bàn phím.

```
Program c6b3;
uses crt;
type    matran=array [1..30,1..30] of integer;
var     a: matran;
        i, j, n: integer;
BEGIN
clrscr;
write(' So dong va cot cua ma tran A , n =');
readln(n);
writeln(' vao ma tran A: ');
for i :=1 to n do
    for j:=1 to n do
            begin
                    write('a[' ,i, ',' ,j, ']=');
                    readln(a[i,j]);
            end;
clrscr;
writeln(' ma tran tam giac tren ');
for i:=1 to n do
    begin
    for j:=1 to n do
            begin
                    gotoxy(4*(j+1),3+i);
            write(' ',a[i,j],' ');
```

```
            end;
        writeln;
        end;
    write(' An ENTER de tiep tuc !'); readln;    clrscr;
    writeln('Ma tran tam giac duoi');  writeln;
    for i:=2 to n do
        begin
        for j:=1 to i-1 do write(a[i,j],' ');
        writeln;
        end;
    writeln;
    write(' An ENTER de ket thuc !');        readln;
    END.
```

Bài 28: Lập trình nhập đọc từ bàn phím dãy n số thực rồi đếm xem có bao nhiêu số dương, tính tổng của chúng. Đưa kết quả ra màn hình .

```
Program c6b5;
Uses crt;
Var         n, i, dem : integer;
            d : real;
            a : array [1.. 50] of real;
BEGIN
clrscr;
write ('n = '); readln (n);
d:=0; dem :=0;
for i:=1 to n do
    begin
            write (' a(',i,')='); readln(a[i]);
            if a[i]>0 then
                    begin
                     dem:=dem+1;
                     d:=d + a[i];
                    end;
    end;
 writeln (' Trong ',n,'số có ',dem,'so dương');
```

```
writeln (' Tổng các số dương trong',n,' số là :', d:12:3);
writeln (' ấn ENTER !'); readln;
END.
```

Bài 29: Lập trình đọc từ bàn phím dãy n số nguyên rồi đếm xem có bao số là lẻ. Đưa ra màn hình số lượng và các số lẻ.

Bài 30: Lập chương trình thực hiện các việc sau:

- Đọc từ bàn phím một dãy n số nguyên.
- Sắp các số lẻ lên đầu dãy, các số chẵn xuống cuối dãy.
- Đưa ra màn hình dãy số đã sắp, số lượng các số lẻ và tổng của chúng.

Bài 31: Lập trình đưa dãy số a_1, a_2,..., a_n vào máy từ bàn phím. Đưa ra màn hình số bé nhất và thứ tự của nó trong dãy số.

```
Program c6b15;
Uses crt;
Var
   A:array[1..100] of real;
   I,n,tt: integer;
   Min: real;
BEGIN
Clrscr;
Min:=1.e20; tt:=0;
Write('so luong so,n='); readln(n);
For i:=1 to n do
   Begin
         Write(' a(',i,')='); readln(a[i]);
         If a[i]<min then min:=a[i];
   End;
Writeln (' So be nhat =' ,min); writeln;
Writeln (' do la so thu: ');
For i :=1 to n do
If a[i]=min then writeln(i);
Readln;
END.
```

Bài 32: Lập trình đọc vào từ bàn phím dãy n số. Xếp một trong các số bé nhất vào vị trí thứ nhất. Đưa cả hai dãy ra màn hình.

```
Program c6b8;
Uses crt;
Var     a: array [1..100] of real;
        i, n, tt: integer;
        Min: real;
BEGIN
Clrscr;
Min:=1.e20; tt:=0;
Write (' so luong so , n='); readln(n);
For i:=1 to n do
   Begin
    Write('a(',i,')='); readln(a[i]);
   If a[i]<min] then
        Begin
            Min:=a[i];
            Tt:=1;
        End;
   End;
Writeln(' so be nhat=' ,min); writeln;
Writteln('day ban dau:');
Fr i:=1 to n do writeln(a[i]:10:3);
Tg:=a[1]; a[1]:=a[tt]; a[tt]:=tg;
Writeln('day xep lai la');
For i:=1 to n do writeln(a[i]:10:3);
Readln;
END.
```

Bài 33: Lập trình nhập từ bàn phím một dãy n số. Xếp các số lớn nhất lên đầu dãy, tiếp theo là các số bé nhất rồi đến các số còn lại. Đưa cả hai dãy ra màn hình.

Bài 34: Lập trình đọc từ bàn phím dãy n số. Đổi chỗ số lớn nhất và số bé nhất cho nhau. Đưa cả hai dãy ra màn hình.

Bài 35: Lập chương trình xếp thứ tự một dãy số theo thuật toán ở bài tập 4. Đưa dãy đã xếp ra màn hình.

```
Program c6b11;
Uses crt;
Var     a: array[1.. 100] of integer;
        I, j, n: byte;
        Tg: integer;
BEGIN
Clrscr;
Write('n='); readln(n);
Writeln(' Vao day so:');
For i:=1 to n do
   Begin
        Write('a[',i,']='); readln (a[i]);
   End;
For i:=1 to n -1 do
   For j:=1 to n do
        If a[i]>a[j] then
             Begin
                  Tg:=a[i]; a[i]:=a[j]; a[j]:=tg;
             End;
Writeln(' Day da sap la:');
For i:=1 to n do write(a[i]:5);
Readln;
END.
```

Bài 36: Lập một chương trình làm các công việc sau:

a) Đọc dãy n số nguyên từ bàn phím.

b) Xếp lại dãy theo thứ tự tăng.

c) Đọc thêm một số từ bàn phím.

d) Chèn số này vào đúng vị trí của nó trong dãy đã xếp (không được xáo trộn rồi xếp lại).

e) Đưa dãy ban đầu, dãy đã sắp xếp và dãy đã chèn ra màn hình.

```
Program c6b24;B
Uses crt;
Var         a: array [1..100] of integer;
            n, i, j, sm, tg: integer;
BEGIN
Clrscr;
Write('so luong so:'); readln(n);
For i:=1 to n do
   Begin
         Write('vao so thu ',i,'='); readln(a[i]);
   End;
Writeln;
Writeln(' Day ban dau la:');
For i:=1 to n do { dua day ban dau ra man hinh}
Write (' ',a[i]);
Writeln;
For i:=1 to n-1 do { xep theo thu tu tang}
   For j:=1 to n do
   Begin
         If a[j]< a[i] then
               Begin
               Tg:=a[i]; a[i]:=a[j]; a[j]:=tg;
               End;
   End; {xep xong thu tu}
Writeln; writeln(' Day da xep thu tu:');
For i:=1 to n do { dua day da xep ra man hinh}
   Write(' ',a[i]);
Writeln;
{chen so moi}
writeln; writeln(' cho vao so moi:'); readln(sm);
i:=1 { tim vi tri chen}
while (sm>a[i]) and (i<=n) do i:=i+1;
```

```
writeln('chen vao vi tri thu ',i);
readln;
if i>n then a[i]:=sm else
   begin
         for j:=n downto i do a[j+1]:=a[j];
         a[i]:=sm;
   end;
writeln; writeln('day da chen:');
{ dua day da chen ra man hinh}
for i:=1 to n+1 do Write(' ',a[i]); writeln;
writeln;writeln('an ENTER'); readln;
END.
```

• Kiểu xâu

Bài 37: Lập trình đọc vào một câu (ít hơn 30 ký tự) từ bàn phím rồi đếm xem câu đó có bao nhiêu từ.

```
Program c7b1;
uses crt;
Var      x, xx, cau: string[30];
         l, i, dem: byte;
BEGIN
clrscr;
write (' Cho vào một câu không quá 30 ký tự :'); rewadln(cau);
l:=length(cau); dem:=0;
for i:=1 to l-1 do
   begin
         x:=copy(cau,i,1); xx:=copy(cau,i+1,1);
         if (x =' ') and (xx<> ' ') then dem :=dem+1;
   end;
if cau[1]<> ' ' then dem := dem+1;
writeln;
writeln(' số từ trong cau: ', dem);
```

```
writeln; writeln;
writeln ('ấn ENTER để kết thúc');
readln;
END.
```

Bài 38: Lập trình đọc một câu vào từ bàn phím rồi đưa ra màn hình dưới dạng một cột. Ví dụ đọc vào (TRUNG HỌC CÔNG NGHIỆP), đưa ra màn hình:

TRUNG
HỌC
CÔNG
NGHIỆP

```
Program c7b2;
uses crt;
Var     x, xx, cau: string[30];
        l, i: byte;
BEGIN
clrscr;
write ('Cho vào một câu không quá 30 ký tự :'); rewadln(cau);
l:=length(cau); dem:=0;
for i:=1 to l do
   begin
        x:=copy(cau,i,1); xx:=copy(cau,i+1,1);
        if (x =' ') and (xx<> ' ') then writeln
        else write(x);
   end;
writeln; writeln;
writeln ('ấn ENTER để kết thúc');
readln;
END.
```

Bài 39: Lập chương trình:

a) Đọc từ bàn phím một xâu;

b) Đọc một ký tự từ bàn phím rồi đếm số lượng ký tự này trong xâu, thông báo kết quả trên màn hình.

```
Program c7b3;
uses crt;
var         c: char;
            x, cau: string[30];
            l, i: byte;
            d: integer;
BEGIN
clrscr;
writeln(' Bạn hãy cho vào một câu:') ; readln(cau);
writeln; write(' Cần đếm ký tự nào? ') ; readln(c);
l:=length(cau);
d:=0;
for i:=1 to l do
   begin
            x:=copy(cau,i,1);
            if x=c then d:=d+1;
   end;
writeln;
writeln('Số ký tự ', c , ' trong câu là : 'd);  writeln;
readln;
END.
```

Bài 40: Lập chương trình đọc từ bàn phím một câu, tìm xem câu đó có bao nhiêu từ, sau đó đọc vào từ bàn phím một ký tự, tìm xem câu đó có bao nhiêu từ bắt đầu bằng ký tự đã đọc, đưa kết quả ra màn hình.

```
Program c7b4;
uses crt;
var         x, xx, cau, sao, kt: string[30];
            l, i, s1: byte;
BEGIN
clrscr;
write (' Cho vao mot cau :');      readln(cau);
l:=length(cau); s1:=0;
```

```
for i:=1 to l-1 do
    begin
    x:=copy(cau,i,1); xx:=copy(cau,i+1,1);
    if (x=' ') and (xx<>' ') then s1:=s1+1:
    end;
if cau[1]=' ' then s1:=s1-1;
writeln;
writeln('So tu trong cau la: ' s1+1);
writeln('An ENTER de ket thuc!');
writeln;
readln;
write(' ban can dem bat dau tu ky tu gi?');
readln(kt);
sao:= copy(cau,1,1); sao:=upcase(sao[1]);
if sao=upcase(kt[1]) then s1:=1 else s1:=0;
for i:=1 to l do
    begin
    x:=copy(cau,i,1); xx:=copy(cau,i+1,1);
xx:=upcase(xx[1]);
if (x=' ') and (xx=upcase(kt[1])) then s1:=s1+1;
end;
writeln(' So tu bat dau bang ky tu ' , kt , 'la :' ,s1);
writeln;
writeln(' An ENTER de ket thuc!');
readln;
END.
```

Bài 41: Lập chương thực hiện các công việc sau:

- Đọc vào từ bàn phím một xâu kí tự.
- Sắp xếp lại xâu đó sao cho ký tự thứ 1 đổi chỗ cho ký tự lẻ cuối cùng của xâu; ký tự thứ 3 đổi chỗ cho ký tự sát cuối cùng của xâu;...; ký tự ở vị trí chẵn không đổi.
- Đưa xâu đã sắp xếp ra màn hình.

Bài 42: Lập chương trình kẻ một bảng MENU. Dữ liệu đưa vào từ bàn phím gồm:

1) Tên tiêu đề menu: tiêu đề,
2) Số lượng các menu: N,
3) Nội dung từng menu: nd[i], i=1,2,3,...,N.

Yêu cầu lập chương trình thể hiện trên màn hình các nội dung trên nằm trong một khung hình chữ nhật. Chú ý tính kích thước khung hình chữ nhật phụ thuộc vào N và độ dài các xâu tiêu đề và nội dung.

```
Program c7b6;
Uses crt;
var     tieude: string;
        nd: array[1..10] of string;
        max, d, c, h, n, i : integer;
BEGIN
clrscr;
write(' Cho tieu de MENU :'); readln (tieude);
max:=length(tieude);
write(' Co may dong :'); readln(d);
for i:=1 to d do
   begin
   write(' Noi dung dong ' ,i, ': '); readln(nd[i]);
   if max < length(nd[i]) then max:=length(nd[i]);
   end;
clrscr;
max:= max+4;
gotoxy(10,3);
writeln(tieude);   writeln;
gotoxy(9,4);
for i:=1 to length(tieude)+1 do writeln ('=');
c:=trunc((80-max)/2); h:=5;
gotoxy(c,h);
for i:=1 to max+5 do write('*');
for i:=1 to d do
   begin
        gotoxy(c,h+i);
```

```
            write('*',i,'. ', nd[i]);
            gotoxy(c+max+4, h+i);       writeln('*');
    end;
gotoxy(c,h+d+1);
for i:=1 to max+5 do write('*');
writeln;
writeln('An ENTER de ket thuc ! ');
readln;
END.
```

Bài 43: Cho một văn bản gồm không quá 60 dòng, mỗi dòng không quá 80 ký tự. Lập chương trình tính số lần xuất hiện của từng chữ cái trong văn bản và tần suất xuất hiện của chúng. Đưa kết quả ra màn hình. (Tần suất xuất hiện chữ cái x bằng số lần xuất hiện x chia cho tổng số các chữ cái trong văn bản không phân biệt chữ hoa hay chữ thường).

Bài 44: Cho một dãy các họ tên. Lập chương trình tìm xem trong dãy đó có bao nhiêu họ khác nhau và số lần xuất hiện của chúng. Đưa ra màn hình các họ và số lần xuất hiện của chúng.

Bài 45: Lập chương trình đọc một văn bản từ bàn phím rồi thay xâu hà nội (nếu có) bằng xâu HÀ NỘI.

Bài 46: Lập chương trình đọc vào một văn bản rồi chèn vào văn bản đó ở vị trí thứ i một xâu mới. Văn bản xâu mới và i đọc từ bàn phím. Đưa văn bản lúc đầu và lúc cuối ra màn hình.

```
Program c7b10;
Uses crt;
Var     cau, ch: string [30];
        i: byte;
BEGIN
clrscr;
write(' ban hay cho vao mot cau :'); readln(cau);
writeln;
write( 'Can chen nhung ky tu nao ? '); readln(ch);
ch:= ' ' + ch + ' ' ;
writeln;
write ( ' Chen vao vi tri thu may ? ');    readln(i);
clrscr;
```

```
writeln(' Cau luc dau : ' ,cau);
insert(ch, cau, i );
writeln;
writeln (' Cau da chen : ', cau 0;
writeln;
writeln (' An ENTER de ket thuc! ');
readln;
END.
```

• Kiểu bản ghi

Bài 47: Lập chương trình nhập một danh sách n sinh viên gồm họ tên, năm sinh, điểm thi vào trường (là điểm toán + điểm lý + điểm hoá). Đưa danh sách này ra màn hình.

Program c8b1;

```
Uses crt;
type        tulieu=record
            ten: string;
            namsinh: integer;
            diem: real;
end;
Var         sinhvien: array[1..100] of tulieu;
            n, i: integer;
BEGIN
 clrscr;
write(' So luong sinh vien = '); readln(n);
writeln(' Nhap du lieu:');
for i:=1 to n do
with sinhvien[i] do
    begin
            write (' Ho ten sinh vien thu ',i, ' : ');readln(ten);
            write(' Nam sinh: ');  readln(namsinh);
            write(' Diem thi vao truong: ');       readln(diem);
    end;
clrscr;
gotoxy(10,2);     write(' DANH SACH SINH VIEN');
gotoxy(10,3);     write(' ------------------------------');
```

```
gotoxy(5,6);              write(' tt');     gotoxy(10,6);
write(' Ho va ten');
gotoxy(40,6);     write(' Nam sinh ');
gotoxy(50,8);     write(' Diem thi vao truong ');
for i:=1 to n do
with sinhvien[i] do
   begin
         gotoxy(5,8+i); write(' ',i);
         gotoxy(10,8+i); write(' ',ten);
         gotoxy(40,8+i); write(' ',namsinh);
         gotoxy(50,8+i); write(' ',diem:3:1);
   end;
readln;
END.
```

Bài 48: Lập trình đọc vào máy một danh sách n sinh viên và điểm thi môn tin học từ bàn phím rồi đưa ra màn hình danh sách các sinh viên không đạt (dưới 5 điểm) kèm theo điểm. Biểu mẫu đưa ra có dạng sau:

```
-------------------------------------------------------------------------------
| TT |              HO VA TEN                |          DIEM THI             |
-------------------------------------------------------------------------------
|    |                                       |                               |
```

```
Program c8b2;
Uses crt;
type      tulieu=record
          ten: string;
          diem: byte;
end;
var       sinhvien: array [1..100] of tulieu;
          n, i, tt: integer;
BEGIN
clrscr;
write(' So luong sinh vien =');   readln(n);
writeln(' Nhap du lieu');
for i:=1 to n do
with sinhvien[i] do
```

```
    begin
            write(' Ho ten sinh vien thu ', i ,' : ');
            readln(ten);
            write(' Diem thi TIN HOC : '); readln(diem);
end;
 clrscr;    gotoxy(10,2);
write(' DANH SACH SINH VIEN KHONG DAT MON TIN HOC');
gotoxy(5,4);
for i:=1 to 70 do write('-');
gotoxy(5,5); write(' | TT');
gotoxy(10,5); write(' | HO VA TEN ');
gotoxy(40,5); write(' | DIEM THI TIN HOC');
gotoxy(74,5); write(' | ');
gotoxy(5,6);
for i:=1 to 70 do write ('-');
tt:=0;
for i:=1 to n do
with sinhvien[i] do
    begin
    if diem<5 then
            begin
                    tt:=tt+1;
                    gotoxy(5,7+i); write(' | ',tt);
                    gotoxy(10,7+i); write(' | ';ten);
                    gotoxy(40,7+i); write(' | ',diem);
                    gotoxy(74,7+i); write(' | ');
            end;
    end;
readln;
END.
```

Bài 49: Lập chương trình nhập từ bàn phím một danh sách sinh viên gồm: họ và tên, năm sinh. Đưa ra màn hình họ và tên những sinh viên lớn tuổi nhất.

```
Program c8b3;
Uses crt;
type        ho_so = record
```

```
            ho_ten: string;
            nam_sinh: integer;
    end;
    Var     sinhvien: array[1..100] of ho_so;
            i, n, min: integer;
    BEGIN
     clrscr;
    write ('Cho so sinh vien: '); readln(n);
    for i:=1 to n do
    with sinhvien[i] do
    begin
       write ('Cho ho va ten SV ', i ,' : ');
       readln(ho_ten);
       write ('Nam sinh SV ', i ,' : ' ); readln(nam_sinh);
    end;
    i:=1; min:=sinhvien[i].nam_sinh;
    for i:=2 to n do
       if sinhvien[i]< min then min:=sinhvien[i].nam_sinh;
    writeln(' Cac sinh vien nhieu tuoi nhat la :');
    for i:=1 to n do
       if sinhvien[i].nam_sinh=min then
            writeln(sinhvien[i].ho_ten);
            writeln;
    writeln('An ENTER de ket thuc ! ');
    readln;
    END.
```

Bài 50: Lập chương trình nhập từ bàn phím một danh sách sinh viên gồm: họ và tên, năm sinh, giới tính, quê quán. Đưa ra màn hình danh sách này và danh sách các sinh viên là nữ và sinh sau năm 1975.

```
    Program c8b4;
    Use crt;
    type    tulieu = record
            ten: string;
            namsinh: word;
            gioitinh: string;
```

```
            quequan: string;
end;
Var         sinhvien: array[1..50] of tulieu;
            n, i ,tt : integer;
BEGIN
 clrscr;
write(' So luong sinh vien = ');   readln (n);
writeln (' Nhap du lieu ');
for i:=1 to n do
with sinhvien[i] do
    begin
    write(' Ho ten sinh vien thu ', i ,' : ' ); readln(ten);
    write(' gioi tinh (nam/nu) : ' ); readln(gioitinh);
    write(' que quan: (tinh) '); readln(quequan);
    end;
clrscr;
gotoxy(10,2); write (' DANH SACH SINH VIEN ');
gotoxy(10,4); write('--------------------------------');
gotoxy(5,8); write(' TT');
gotoxy(10,8); write(' HO VA TEN ');
gotoxy(40,8); write(' NAM SINH');
gotoxy(55,8); write(' GIOI TINH ');
gotoxy(70,8); write(' QUE QUAN ');
for i:=1 to n do
with sinhvien[i] do
    begin
            gotoxy(5,8+i); write(' ', i );
            gotoxy(10,8+i); write(' ', ten);
            gotoxy(40,8+i); write(' ' , namsinh);
            gotoxy(55,8+i); write(' ' , gioitinh);
            gotoxy(70,8+i); write(' ' , quequan);
    end;
readln;
clrscr;
gotoxy(10,2);
```

```
write(' DANH SACH SINH VIEN NU SINH SAU 1975 ');
gotoxy(10,4);
write (' ----------');
gotoxy(5,8); write (' TT ');
gotoxy(10,8); write (' HO VA TEN ');
gotoxy(40,8); write(' NAM SINH ');
tt:=0;
for i:=1 to n do
with sinhvien[i] do
   begin
   if (gioitinh='nu') and (namsinh>1976) then
         begin
         tt:=tt+1;
         gotoxy(5,8+tt); write (' ',tt);
         gotoxy(10,8+tt); write (' ',ten);
         gotoxy(40,8+tt); write (' ',namsinh);
         end;
   end;
readln;
END.
```

Bài 51: Lập chương trình thực hiện các việc sau:

a) Nhập từ bàn phím một danh sách bản ghi gồm các trường:

TENTHUOC: string[30]

NAMHETHAN: integer;

trong đó TENTHUOC chứa tên thuốc, NAMHETHAN chứa năm hết hạn của thuốc.

b) Xoá khỏi danh sách những loại thuốc có năm hết hạn trước 1992.

c) Đưa ra màn hình tên các loại thuốc đến năm 1992 là hết hạn.

Bài 52: Một điểm trong mặt phẳng toạ độ nguyên được cất vào kiểu dữ liệu PoinType định nghĩa như sau:

```
type   PoinType = record
         x, y: integer;
       end;
```

Lập chương trình thực hiện các việc sau:

a) Nhập một dãy các điểm vào từ bàn phím.

b) Kiểm tra các điểm đã cho có thẳng hàng hay không? Nếu thẳng hàng hãy đưa ra màn hình phương trình của đường thẳng dưới dạng Ax + By + C = 0.

Bài 53: Lập chương trình để thực hiện các việc sau:

a) Nhập thông tin từ bàn phím về tình hình thời tiết trong ngày của khu vực.

Mỗi bản tin là một bản ghi gồm các trường :

- Ngày, tháng, năm.
- Địa điểm đo: xâu ký tự độ dài không quá 35.
- Lượng mưa: giá trị thực.
- Nhiệt độ: giá trị nguyên.

Số lượng bản ghi không biết trước, dấu hiệu kết thúc nhập là bản ghi có trường ngày tháng năm để trống.

b) Hãy tìm xem ngày nào và ở địa điểm nào có nhiệt độ cao nhất ? Đưa kết quả ra màn hình.

c) Đưa ra màn hình lượng mưa trung bình trong ngày của khu vực.

```
Program c8b7;
Uses crt;
Type      bg = record
          ng:0..31;
          th:1..12;
          nam:1900..2010;
          dia_diem: string[35];
          l_mua: real;
          nh_do: integer;
end;
Var       lmtrb: real;
          nt, nx, i, n, ix, iy: integer;
          bt: bg;
          bc: array [1..1000] of bg;
BEGIN
   clrscr;          n:=0;
repeat
   with bt do
   begin
          write('Ngay '); ix:= wherex+3;
```

```
                    iy:=wherey; read(ng);
        if ng<>0 then
                begin
                gotoxy(ix,iy); ix:=ix+10;
                write(' Thang '); read(th);
                gotoxy(ix,iy) write(' Nam');
                readln(nam);
                write(' Dia diem: '); readln(dia_diem);
                write(' Luong mua '); readln (l_mua);
                write(' Nhiet do '); readln(nh_do);
                writeln(' **********');
                end;
end;
if bt.ng <> 0 then
   begin
        inc(n); bc [n]:=bt;
   end;
until bt.ng=0;
nt:= -273; nx:=0; lmtrb:=0;
for i:=1 to n do
   begin
   if nt < bc[i] .nh_do then
        begin
            nt:=bc[i].nh_do; nx:=i;
        end;
   lmtrb:= lmtrb + bc[i].l_mua
   end;
writeln(#13#10' Nhiet do cao nhat quan sat duoc la :'nt:3);
writeln( ' Do duoc tai:' , bc[nx].dia_diem);
writeln(' Trong ngay:', bc[nx].ng, 'thang', bc[nx].th, 'nam' , bc[nx].nam);
writeln(#13#10'***** Luong mua trung binh trong ngay cua khu vuc la:' ,
lmtrb/n:10:2);
readln;
END.
```

• **Kiểu tệp**

Bài 54: Viết chương trình đọc một dãy số từ bàn phím rồi ghi chúng vào đĩa mềm, sau đó đọc từ đĩa ra dãy số đó và tính tổng của chúng.

Bài 55: Lập một chương trình thực hiện các công việc sau:

- Đọc từ bàn phím một danh sách sinh viên gồm họ tên, giới tính, năm sinh.
- Ghi dữ liệu ra đĩa mềm với tên file là SVLOP_X
- Tìm các sinh viên là nữ và sinh trước 1972 rồi đưa kết quả ra màn hình.

Bài 56: Lập chương trình thực hiện các việc sau:

- Đọc từ bàn phím một dãy n số nguyên.
- Ghi dãy số đó vào đĩa mềm.
- Sắp các số lẻ lên đầu dãy, các số chẵn xuống cuối dãy mà không được sử dụng thêm mảng mới.
- Đưa ra màn hình dãy số đã sắp, số lượng các số lẻ và tổng của chúng.

Bài 57: Viết chương trình thực hiện các việc sau:

a) Đọc từ bàn phím một danh sách gồm họ tên, môn thi thứ nhất, điểm môn thi thứ nhất, môn thi thứ hai, điểm môn thi thứ hai.

b) Ghi vào đĩa mềm với tên file QLHT.

c) Đọc dữ liệu từ file QLHT, tìm những sinh viên phải thi lại (có ít nhất một môn không đạt). Đưa họ tên, các môn thi và điểm tương ứng ra màn hình.

```
Program c9b4;
Uses crt;
type    phdiem = record
        ht: string[22];
        mh1: string[10];
        d1: real;
        mh2: string[10];
        d2: real;
end;
Var     Pd: phdiem;
        F1: file of phdiem;
        n, i: integer;
BEGIN
clrscr;
```

```
write (' So phieu diem n = '); readln(n);
assign(f1, ' QLHT ');
rewrite(f1);
for i := 1 to n do
with pd do
   begin
         writeln (' Vao phieu diem thu ', i );
         write(' Ho ten: '); readln(ht);
         write(' Mon thu 1: '); readln(mh1);
         write(' Diem mon thu 1: '); readln(d1);
         write(' Mon thu 2: '); readln(mh2);
         write(' Diem mon thu 2: '); readln(d2);
         write(f1,pd);
   end;
close(f1);
clrscr;
writeln( ' DANH SACH SINH VIEN THI LAI');      writeln;
Assign(f1, 'QLHT');
reset(f1);
while not eof (f1) do
   begin
    read(f1,pd);
    with pd do
    if (d1 < 5) or (d2 < 5 ) then
         writeln (ht :25, mh1:10, ': ' , d1:3:1 , mh2: 10, ':',d2:3:1);
   end;
Close(f1);
writeln;
writeln(' An ENTER de ket thuc ! ');
readln;
END.
```

Bài 58: Viết chương trình thực hiện các việc sau:

a) Tạo hai file f1, f2 là những file text để ghi dữ liệu từ bàn phím.

b) Nối file f2 vào cuối file f1 và tính độ dài file f1.

c) Đưa nội dung file f1 ra màn hình.

Bài 59: Có n mặt hàng (n <50), mỗi mặt hàng gồm tên (string không quá 10 ký tự), số lượng (integer) và đơn giá (real) . (Đơn giá của một mặt hàng là giá một đơn vị sản phẩm của mặt hàng đó).

1) Lập chương trình nhập các thông tin từ bàn phím và cất vào một file trên đĩa kiểu record với tên là DULIEU.

2) Lập một chương trình thực hiện các công việc sau:

a) Đọc file DULIEU và viết lên màn hình tất cả tên mặt hàng chứa trong file đang xét.

b) Vào từ bàn phím tên mặt hàng và số lượng cần xuất hay nhập. Chương trình sẽ cập nhật số lượng mới của mặt hàng đang xét vào file DULIEU. Nếu vào không đúng tên mặt hàng, chương trình sẽ thông báo lỗi và đòi hỏi vào lại.

c) Tìm trong file DULIEU và ghi lên một file Text trên đĩa với tên BAOCAO.TXT tất cả những mặt hàng (gồm tên, số lượng, tổng giá trị) thoả mãn tổng giá trị lớn hơn hay bằng một giá trị cho trước từ bàn phím. (Tổng giá trị của một mặt hàng bằng đơn giá nhân với số lượng của mặt hàng đó).

Bài 60: Lập chương trình kiểm tra xem một file có tên đưa từ bàn phím có trong đĩa ở ổ chủ không? Nếu có thì cho biết độ dài của nó. Tổ chức chương trình hội thoại để thực hiện nhiều lần cho đến khi ấn phím ESC.

```
Program c9b7;
Uses crt;
Var     f: file of byte;
        s: string[79];
        ch: char;
BEGIN
clrscr;
gotoxy(12,24);
write( 'An phim bat ki de tiep tuc. ' , ' An ESC de ra khoi chuong trinh. ');
gotoxy(1,1);
repeat
   write (' Cho biet ten file can tim :'); readln(s);
   assign(f,s);
{$I-}
reset(f);
{$I+}
```

```
    if IOResult=0 then
        writeln(' File ' , s , ' co kich thuoc la :', filesize(f),' bytes.')
    else
        writeln(' Khong tim thay File ' , s , ' ! ');
    writeln;        ch:= readkey;
    if wherey>=22 then
        begin
            window(1,180,23);
            gotoxy(1,1);
            clrscr;
        end;
until ch = #27;
END.
```

Bài 61: Giả sử đã có các file F1, F2 và F3 là những file text. Hãy lập chương trình tạo file FR từ F1 và F2 sao cho mỗi bản ghi của FR được tạo bằng cách ghép các bản ghi tương ứng của F1, F2. Nếu các file F1, F2 có số lượng bản ghi khác nhau thì ghi phần dư vào cuối file F3.

Bài 62: Lập chương trình thực hiện các việc sau:

a) Tạo hai file FA, FB để ghi dữ liệu, trong đó mỗi bản ghi của FA chứa ba dữ liệu thực, mỗi bản ghi của FB chứa một dữ liệu nguyên và một dữ liệu xâu ký tự. Các dữ liệu sẽ được đưa vào từ bàn phím.

b) Tạo file FS từ FA và FB sao cho mỗi bản ghi của FS được tạo bằng cách ghép các bản ghi tương ứng của FA, FB. Nếu các file FA, FB có số lượng bản ghi khác nhau thì bỏ phần dư của một trong hai file đó.

c) Đưa các bản ghi FS ra màn hình.

```
Program c9b9;
Uses crt;
type    ra = record
            x1, x2, x3: real ;
end;
    rb = record
            n1: integer;
            s1: string;
    end;
    rc = record
```

```
            x1, x2, x3: real;
            n1: integer;
            s1: string;
    end;
Var         i: integer;
            ba: ra;
            bb: rb;
            bc: rc;
            f1: file of ra;
            f2: file of rb;
            f3: file of rc;
            ans: char;
BEGIN clrscr;
assign (f1,' FA ');
rewrite(f1);
writeln(' Nhap du lieu cho file FA : ');
repeat
    write(#13#10 ' Cho 3 so thuc :');
    readln(ba.x1,ba.x2,ba.x3);
    write(f1,ba);
 repeat
    write (' Co nhap tiep du lieu cho file FA nua khong ?(C/K)');
    ans :=upcase (readkey);
 until ans in ['C', 'K'];
until ans= 'K';
close (f1);
assign (f2,' FB ');
rewrite (f2);
writeln (#13#10' Nhap du lieu cho file FB: ');
repeat
    write (#13#10 ' Cho so nguyen va mot xau:');
    write ( ' n = '); readln(bb.n1);
    write ( ' xau : '); readln (bb.s1);
    write (f2,bb);
 repeat
```

```
    write (' Co nhap tiep du lieu cho file FB nua khong ?(C/K)');
    ans :=upcase (readkey);
  until ans in ['C' , 'K'];
 until ans= 'K';
 close (f2);
 assign (f3,'FC ');
 rewrite(f3); i:=0;
 assign(f1,'FA');   reset(f1);
 assign(f2,'FB');   reset(f2);
 while (not eof(f1)) and not (eof(f2)) do
    begin
    read(f1,ba);   read(f2,bb);
    bc.x1:=ba.x1; bc.x2:=ba.x2; bc.x3:=ba.x3;
    bc.n1:=bb.n1; bc.s1:=bb.s1;
    write(f3,bc);  inc(i);
    with bc do
          begin
    writeln(#13#10 '***** Ban ghi thu ' , i ,' :');
    writeln('x1 , x2 , x3 = ', x1:12:3,x2:12:3, x3:12:3);
    writeln(' n = ',n1);
    writeln(' Xau : ' ,s1);
    delay(1000);
          end;
    end;
 close(f1); close(f2); close(f3);
 END.
```

Bài 63: Hãy lập chương trình làm các việc sau:

a) Tạo một file, mỗi bản ghi gồm:

- Tên sản phẩm: xâu < 21 ký tự.
- Mã sản phẩm: gồm 2 mã, mỗi mã là một số nguyên.
- Đơn giá: số thực.

Tên file đọc từ bàn phím và kết thúc vào dữ liệu khi gặp tệp rỗng.

b) Cập nhật đơn giá sản phẩm của thông tin ghi trên file dựa theo mã thứ 2 và đơn giá sản phẩm tương ứng. Nếu gặp mã mới thì thông báo và bỏ qua.

• Thủ tục, hàm, chương trình Pascal có cấu trúc

Bài 64: Lập chương trình tính min/max của một hàm số trên đoạn [a, b] bằng cách dùng thủ tục và hàm. Các giá trị a và b đưa vào từ bàn phím, tìm min hay max tuỳ chọn.

```
Program c10b1;
Uses crt;
Var       a, b, ketqua: real ;
          chon: integer;
Procedure menu;
begin
   writeln(' ***** TIM MIN/MAX HAM SO *****');
   writeln(' 1- Tim min f(x) ');
   writeln(' 2- Tim max f(x) ');
   writeln(' 3- Ket thuc ');
   writeln('-----------------------');
   writeln(' An so de chon ! '); readln(chon);
end; {Menu}
Function f(x:real): real;
begin
   f:=x*x - 5*x + 6 ;
end;
Procedure minf (aa, bb: real; var mi: real);
Var       x, dx: real;
   i : integer;
Begin
mi := 1e10; dx:=(bb-aa)/200;
for i:=1 to 200 do
   begin
         x:=aa+(i-1)*dx;
         if f(x) <mi then mi:=f(x);
   end;
end; {minf}
Procedure maxf (aa, bb: real; var ma: real);
var       x, dx: real;
          i: integer;
```

```
Begin
    max:=-1e10; dx:=(bb-aa)/200;
    for i:=1 to 200 do
            begin
                    x:=aa+(i-1)*dx;
                    if f(x)>ma then ma:=f(x);
            end;
End; {maxf}
Procedure thongbao;
Begin
    gotoxy(50,15);
    writeln('* An ENTER de tiep tuc !');
    repeat until keypressed;
end; { thong bao }
BEGIN
clrscr;
menu;
while chon<> 3 do
    begin
    writeln;
    write(' a, b ='); readln(a, b);
    if chon=1 then minf(a,b,ketqua);
    if chon=2 then maxf(a,b,ketqua);
    writeln(' ket qua =',ketqua:16:2);
    thongbao;
clrscr; menu;
    end;
END.
```

Bài 65: Lập chương trình nhập toạ độ n điểm trong không gian (x1, y1, z1) rồi tìm toạ độ các đỉnh của một hình hộp có các cạnh song song với các trục toạ độ và chứa tất cả các điểm trên.

```
Program c10b2;
Uses crt;
Type vt = array[1..100] of real;
Var         n:type
```

```
            x, y, z: vt;
            xl,yl,zl,xn,yn,zn: real;
Procedure Nhap;
Var i:byte;
Begin
repeat
    write('#13#10' Tong so diem :');
    readln(n);
until (n>0) and (n<100);
writeln('Nhap toa do cac diem :');
for i :=1 to n do
    begin
            write('X[', i ,'] , y[', i ,'] : ');
            readln(x[i], y[i],z[i]);
    end;
End; {end of NHAP}
Procedure minmax(var w:vt; var v1,v2:real);
Var i:byte;
Begin
v1:= w[1];
v2:=v1;
for i:=1 to n do
    if w[i] <v1 then v1:=w[i]
     else if w[i]>v2 then v2:=w[i];
end; {end of MINMAX}
BEGIN
clrscr;
nhap;
minmax(x,xn,xl);
minmax(y,yn,yl);
minmax(z,zn,zl);
writeln(' Toa do hai dinh xac dinh hop can tim la:');
writeln(' Dinh duoi ben trai : x = ',xn :6:2,' y=',yn:6:2,'z=',zn:6:2);
writeln('Dinh tren ben phai : x= ',xl:6:2, 'y= ' ,yl:6:2,' z=',zl:6:2);
repeat until keypressed
END.
```

Bài 66: Lập thủ tục đổi các toạ độ tương ứng của hai vectơ a, b cho nhau.

```
Procedure swap(var a, b :vt; n: integer);
Var         i: integer;
            r: real;
Begin
   for i:=1 to n do
            begin
            r:=b[i];
            b[i]:=a[i];
            a[i]:=r;
end;
End.
```

Bài 67: Cho hai ma trận A, B có cùng kích thước. Lập một thủ tục đổi chỗ các phần tử tương ứng của A và B cho nhau. Đưa ra màn hình A, B ban đầu và sau khi đã đổi chỗ.

Bài 68: Lập thủ tục tính tích hai ma trận, thủ tục chuyển vị ma trận. Dùng các thủ tục này lập chương trình đọc vào ma trận A kích thước m, n rồi tính $A.A^{*}$ trong đó A^{*} là chuyển vị của A.

Bài 69: Lập thủ tục kiểm tra xem xâu A có chứa xâu B như một xâu con hay không và nếu có thì xem chứa bao nhiêu lần.

```
Program c10b6;
Uses crt;
Var         a, b, c, d: string;
            k, l, i, j: byte;
BEGIN
clrscr;
write (' Xau A : '); readln(a);
write (' Xau B : '); readln(b);
c:=a; k:=0; l:=length(b); d: =' ' ; j:=pos(b,c);
d := chr(219); { '■'}
while j>0 do
    begin
            inc(k);
            delete(c,j,1);
            insert(d,c,j);
```

```
            j:=pos(b,c);
    end;
if k = 0 then writeln(' Xau ' , a , ' Khong chua xau ', b , '. ')
else
    writeln(' Xau ' ' , b , ' ' ' co mat trong xau ' ' ' , a , ' ' ' ', k , ' lan . ');
readln;
END.
```

Bài 70:

a) Lập hàm xác định chỉ số kể từ đó xây B tham gia vào xâu A như một xâu con lần thứ k.

b) Viết chương trình dùng hàm trên để tìm chỉ số của xâu A mà từ chỉ số đó xâu B tham gia vào xâu A như một xâu con lần thứ k. Các xâu A, B và số k nhập từ bàn phím.

```
Function indxk(a , b : string; k :byte) :byte;
Var         i, j, l: byte;
            c, d: string;
            ok: boolean;
Begin
c:=a; l:=length(b); d:=' ';
d:= chr (219);
i:= pos (b, c); j:=1;
ok:=(i>0) and (j=k);
while (i>0) and (j<k) do
begin
            inc(j);
            delete(c,i,1);
            insert(d,c,i);
            i:=pos(b,c);
            ok:=(i>0) and (j=k);
    end;
if ok then indxk:=i else indxk:=0;
End; { het ham INDXK }
Program c10b13;
Uses crt;
Var a,b: string;
    k: byte;
```

```
Function indxk(a , b : string; k :byte) :byte;
Var     i, j, l: byte;
        c, d: string;
        ok: boolean;
Begin
c:=a; l:=length(b); d:=' ';
d := chr (219);
i:= pos (b, c); j:=1;
ok:=(i>0) and (j=k);
while (i>0) and (j<k) do
begin
        inc(j);
        delete(c,i,1);
        insert(d,c,i);
        i:=pos(b,c);
        ok:=(i>0) and (j=k);
   end;
if ok then indxk:=i else indxk:=0;
End; { het ham INDXK }
BEGIN
clrscr;
write(' Xau A: '); readln(a);
write(' Xau B: '); readln(b);
write(#13#10'K = ' ); readln(k);
writeln(indxk(a,b,k);
readln;
END.
```

Bài 71:

a) Lập thủ tục biến đổi các ký tự của xâu A theo quy tắc sau: Nếu ký tự thứ i ($1 \le i \le$ length(A) A_i) có mặt trong xâu B thì nó được thay thế bằng ký tự tương ứng ở xâu C (các xâu B, C có độ dài bằng nhau và bằng m). Các ký tự A_i không có mặt trong B được giữ nguyên.

b) Lập chương trình nhập các xâu A, B, C từ bàn phím rồi sử dụng thủ tục trên biến đổi xâu A theo quy tắc đã nêu. Đưa kết quả ra màn hình.

```
Program translate(var a, b, c: string; m: byte);
Var i, j: byte;
Begin
for i:=1 to length(a) do
   begin
         j:= pos(a[i], b);
         if j>0 then a[i] :=c[j];
   end;
End;
Program c10b8;
Uses crt;
Var      s1, s2, s3: string;
   ans: char;
   n: byte;
Program translate(var a, b , c: string ; m: byte);
Var i, j: byte;
Begin
for i:=1 to length(a) do
   begin
         j:= pos(a[i], b);
         if j>0 then a[i] :=c[j];
   end;
End;
BEGIN
clrscr;
repeat
   write(#13#10' Xau nguon :'); readln(s1);
   repeat
         write(' Xau khoa :'); readln(s2);
         write( 'Xau thay :'); readln(s3);
         n:=length(s2);
   until n = length(s3);
   translate(s1,s2,s3,n);
   writeln(' Ket qua thay the :',s1);
   repeat
```

```
            write(#13#10' Co lam tiep nua khong ?(C/K)');
            ans:=upcase(readkey);
        until ans in [' C ' , ' K '];
    until ans='K'
    END.
```

Bài 72: Lập thủ tục viết một câu vào vị trí (x, y) và thủ tục xoá câu từ vị trí (x, y) trở đi, trong đó ($1 \le x \le 80$, $1 \le y \le 25$). Dùng các thủ tục đã viết lập chương trình để cho dòng chữ ** DAI HOC BACH KHOA HA NOI ** chạy trên màn hình từ trái sang phải, rồi chạy trên đường chéo màn hình.

```
Program DEMO;
Uses crt;
Const
    a= ' DAI HOC BACH KHOA HA NOI';
    b= ' KHOA TIN HOC ' ;
    c= "PHONG MAY TINH';
Procedure writexy (x ,y: integer; st: string);
Begin
    gotoxy(x,y); write(st);
End;
Procedure delxy (x,y: integer);
Var         i: integer;
Begin
    gotoxy(x, y);
    for i:=x to 80 do write(' ');
end; {delxy}
Procedure qc1(st: string);
Var k, x, y: integer;
Begin
    for k:=25 to 75 do
            begin
                    x:=80-k; y:=10;
                    writexy(x,y,st); delay(100);
                    delxy(x, y);
                    if keypressed then exit;
            end;
```

```
End; {qc1}
Procedure qc2(st: string);
Var k, x, y: integer;
Begin
   for k:=1 to 64 do
         begin
               x:=65-k; y:=(25- (k div 3));
               writexy(x,y,st); delay(100);
               delxy(x, y);
               if keypressed then exit;
         end;
End; {qc2}
   { chuong trinh chinh }
BEGIN
textbackground(1);
clrscr;
writexy(2,2,'An ENTER de ketthuc !');
textcolor(3);
repeat
   qc1(a); qc2(a); qc1(b); qc2(b); qc1(c); qc2(c);
until keypressed;
END.
```

Bài 73: Lập thủ tục Di: Đổi dấu tất cả các phần tử trên dòng i của ma trận A; thủ tục Cj: Đổi dấu tất cả các phần tử trên cột j của ma trận A; lập chương trình dùng các thủ tục trên để biến đổi ma trận A thành ma trận có tổng các phần tử cùng dòng hoặc cùng cột không âm.

```
Program c10b10;
Uses crt;
Var      q: array[1..20,1..40] of real;
         i, j, m, n: integer;
         ok: boolean;
Function d(i:integer):real;
Var      j: integer;
         s: real;
Begin
```

```
    s:=0;
    for j:=1 to n do s:=s+q[i,j];
    d:=s;
End; {End of function D(i).}
Function c(j:integer):real;
Var         i: integer;
            s: real;
Begin
    s:=0;
    for i:=1 to m do s:=s+q[i,j];
    c:=s;
End; {End of function C(j).}
Procedure sc(j:integer);
{Doi dau cot j}
Var         i: integer;
Begin
    writeln(' Doi dau dong',j);
    delay(500);
    for i:=1 to m do q[i,j]:=-q[i,j];
End; {End of Procedure sc(i).}
Procedure sd(i:integer);
{Doi dau hang i}
Var         j: integer;
Begin
    writeln(' Doi dau dong',i);
    delay(500);
    for j:=1 to n do q[i,j]:=-q[i,j];
End; {End of Procedure sd(i).}
BEGIN
clrscr;
write(#13#10 ' M ,N ='); readln(m,n);
writeln( ' Nhap mang Q:');
for i:=1 to m do
    for j:=1 to n do
```

```
    begin
        write('Q[' , i , ',' , j , ']=');
        readln(q[i,j]);
    end;
ok:=false;
while not ok do
    begin
    ok:=true;
    for i:=1 to m do if (d(i) <0 ) then
        begin sd(i); ok :=false; end;
    for j;=1 to n do if (c(j) < 0 ) then
        begin sc(j); ok:=false; end;
    end;
writeln(' Ket qua : ');
for i:=1 to m do
    for j:=1 to n do
        writeln(' Q[', i, ',', j, ']=', q[i,j]:6:2);
readln;
END.
```

Bài 74: Lập chương trình nhập một văn bản T, đếm các ký tự khác nhau và số lần xuất hiện các ký tự đó trong T, tính tỷ số giữa số lần xuất hiện từng ký tự và độ dài văn bản T, hiển thị trên màn hình hoặc in ra máy in tuỳ chọn.

```
Program c10b11;
Uses crt, printer;
Var     t: array[1..10000] of char;
        s1: array[char] of integer;
        i, ans: char;
        n: integer;
BEGIN
 repeat
    clrscr;
    for i:=chr(0) to chr(255) do s1[i]:=0;
gotoxy(1,1);
write(' Van ban vao:'); gotoxy(21,24);
write(' Ket thuc van ban vao: An Ctrl +Z. ');
```

```
    window(0,0,80,22);
    gotoxy(15,1); n:=0;
    repeat
       ans:=readkey; write(ans);
       inc(n);
       t[n]:=ans;
       s1[ans]:=s1[ans]+1;
    until ord(ans) =26;
    dec(n);
    repeat
       writeln;
       write(' Dua ket qua ra dau? (S: Man hinh, P: May in).');
       ans:=upcase(readkey);
    until ans in [' S ' , ' P '];
    writeln;
    Case ans of
       'S' : if n=0 then writeln(' Van ban trong !')
     else
       for i:=chr(0) to chr(255) do
        if (s1[i]<>0) and 9i<>chr(26)) then
              writeln(i:3,' Tan so: ',s1[i]:6,' Tan suat:' ,s1[i]/n:10:8); 'P': if n=0
then writeln(lst, 'Van ban trong!')
     else
       for i:=chr(0) to chr(255) do
        if (s1[i]<>0) and 9i<>chr(26)) then
              writeln(i:3,' Tan so: ',s1[i]:6,' Tan suat:' ,s1[i]/n:10:8);
    end;
    repeat
       write(#13#10' Co lam tiep nua khong? (C/K)');
       ans:=upcase(readkey);
    until ans in [' C ', ' K ']
    until ans='K';
    END.
```

PHỤ LỤC
BẢNG MÃ CHUẨN ASCII

Số TT	Ký tự	Số TT	Ký tự	Số TT	Ký tự	Số TT	Ký tự
0	NUL	32	Space	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	700	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	405	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w

24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

TÀI LIỆU THAM KHẢO

1. N. Wirth - ***Cấu trúc dữ liệu + Giải thuật = Chương trình*** - NXB Đại học và Giáo dục chuyên nghiệp, 1991.
2. K. Jensen and N. Wirth - ***PASCAL, User Manual and Report*** - Springer - Verlag, 1974.
3. N. Wirth - ***The Programming Language PASCAL*** - Acta infomatica,1, No 1, 1974.
4. P.Le Beux et Tavernien - ***PASCAL par la pratique*** - Cybex, 1999.
5. Boussard et Mahl - ***Progremmion avancée*** - Erolles, 1998.
6. Tô Văn Nam - ***Tin học đại cương*** - NXB Khoa học và Kỹ thuật.
7. Phạm Văn Ất - ***TURBO PASCAL 5 - 6*** - NXB Giáo Dục, 1993.
8. Đoàn Nguyên Hải - ***Lập trình căn bản*** - NXB Đại học Quốc gia Tp. Hồ Chí Minh, 1999.
9. Quách Tuấn Ngọc - ***Ngôn ngữ lập trình Pascal*** - Đại học Bách khoa Hà Nội.
10. Nguyễn Xuân My - ***Bài tập lập trình PASCAL*** - NXB Thống kê, 1997.
11. Dương Viết Thắng - ***Bài tập Pascal*** - Đại học Bách khoa Hà Nội.

MỤC LỤC

NHÀ XUẤT BẢN HÀ NỘI
4 - TỐNG DUY TÂN, QUẬN HOÀN KIẾM, HÀ NỘI
ĐT: (04) 8252916, 8257063 - FAX: (04) 8257063

GIÁO TRÌNH

TIN HỌC ĐẠI CƯƠNG

NHÀ XUẤT BẢN HÀ NỘI - 2005

Chịu trách nhiệm xuất bản:
NGUYỄN KHẮC OÁNH
Biên tập:
TRƯƠNG ĐỨC HÙNG
Bìa:
VĂN SÁNG
Trình bày - Kỹ thuật vi tính:
THU HIỀN
Sửa bản in:
PHẠM THU TRANG

In 1.160 cuốn, khổ 17 x 24 cm, tại Công ty In Khoa học Kỹ thuật
101A Nguyễn Khuyến - Đống Đa - Hà Nội.
Số in: 94. Giấy phép xuất bản số: 199GT/290 CXB cấp ngày 14/3/2005.
In xong và nộp lưu chiểu tháng 4 năm 2005.

BỘ GIÁO TRÌNH XUẤT BẢN NĂM 2005
KHỐI TRƯỜNG TRUNG HỌC CÔNG NGHIỆP

1. AN TOÀN LAO ĐỘNG CHUNG
2. TIN HỌC ĐẠI CƯƠNG
3. AUTOCAD
4. VẼ KỸ THUẬT
5. VẬT LIỆU CƠ KHÍ
6. ĐO LƯỜNG KỸ THUẬT
7. CƠ KỸ THUẬT
8. NGUYÊN LÝ CẮT VÀ DỤNG CỤ CẮT

Giá: 26.500 đ