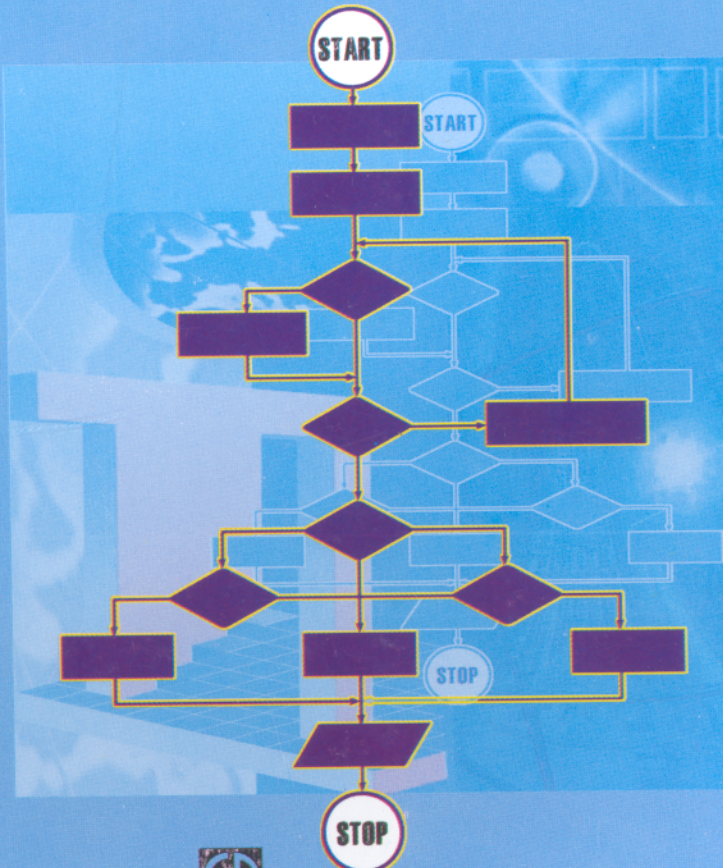


VỤ GIÁO DỤC CHUYÊN NGHIỆP

GIÁO TRÌNH NHẬP MÔN TIN HỌC

SÁCH DÙNG CHO CÁC TRƯỜNG ĐÀO TẠO HỆ TRUNG HỌC CHUYÊN NGHIỆP



NHÀ XUẤT BẢN GIÁO DỤC

TÔ VĂN NAM

Giáo trình

NHẬP MÔN TIN HỌC

(Sách dùng cho các trường Đào tạo hệ Trung học chuyên nghiệp)

(Tái bản lần thứ nhất)

NHÀ XUẤT BẢN GIÁO DỤC

Lời giới thiệu

Năm 2002, Vụ Giáo dục Chuyên nghiệp - Bộ Giáo dục và Đào tạo đã phối hợp với Nhà xuất bản Giáo dục xuất bản 21 giáo trình phục vụ cho đào tạo hệ THCN. Các giáo trình trên đã được nhiều trường sử dụng và hoan nghênh. Để tiếp tục bổ sung nguồn giáo trình đang còn thiếu, Vụ Giáo dục Chuyên nghiệp phối hợp cùng Nhà xuất bản Giáo dục tiếp tục biên soạn một số giáo trình, sách tham khảo phục vụ cho đào tạo ở các ngành: Điện - Điện tử, Tin học, Khai thác cơ khí. Những giáo trình này trước khi biên soạn, Vụ Giáo dục Chuyên nghiệp đã gửi đề cương về trên 20 trường và tổ chức hội thảo, lấy ý kiến đóng góp về nội dung đề cương các giáo trình nói trên. Trên cơ sở nghiên cứu ý kiến đóng góp của các trường, nhóm tác giả đã điều chỉnh nội dung các giáo trình cho phù hợp với yêu cầu thực tiễn hơn.

Với kinh nghiệm giảng dạy, kiến thức tích lũy qua nhiều năm, các tác giả đã cố gắng để những nội dung được trình bày là những kiến thức cơ bản nhất nhưng vẫn cập nhật được với những tiến bộ của khoa học kỹ thuật, với thực tế sản xuất. Nội dung của giáo trình còn tạo sự liên thông từ Dạy nghề lên THCN.

Các giáo trình được biên soạn theo hướng mở, kiến thức rộng và cố gắng chỉ ra tính ứng dụng của nội dung được trình bày. Trên cơ sở đó tạo điều kiện để các trường sử dụng một cách phù hợp với điều kiện cơ sở vật chất phục vụ thực hành, thực tập và đặc điểm của các ngành, chuyên ngành đào tạo.

Để việc đổi mới phương pháp dạy và học theo chỉ đạo của Bộ Giáo dục và Đào tạo nhằm nâng cao chất lượng dạy và học, các trường cần trang bị đủ sách cho thư viện và tạo điều kiện để giáo viên và học sinh có đủ sách theo ngành đào tạo. Những giáo trình này cũng là tài liệu tham khảo tốt cho học sinh đã tốt nghiệp cần đào tạo lại, nhân viên kỹ thuật đang trực tiếp sản xuất.

Các giáo trình đã xuất bản không thể tránh khỏi những sai sót. Rất mong các thầy, cô giáo, bạn đọc góp ý để lần xuất bản sau được tốt hơn. Mọi góp ý xin gửi về: Công ty Cổ phần sách Đại học - Dạy nghề, 25 Hàn Thuyên - Hà Nội.

VỤ GIÁO DỤC CHUYÊN NGHIỆP - NXB GIÁO DỤC

Lời nói đầu

Giáo trình **Nhập môn tin học** là tập hợp các bài giảng của tác giả trong khoảng thời gian hơn mười năm qua. Giáo trình bao gồm hai phần :

– Phần 1 : Đại cương

Phần này mang tính nhập môn, giới thiệu những kiến thức cơ bản về tin học như : thông tin, xử lý thông tin, thuật toán, hệ điều hành, mạng máy tính...

– Phần 2 : Ngôn ngữ lập trình **TURBO PASCAL**

Phần này trình bày những kiến thức cơ bản nhất về Turbo Pascal, một ngôn ngữ lập trình bậc cao có mặt trong hầu hết các chương trình đào tạo Tin học từ sơ cấp, trung cấp, cao đẳng đến đại học vì tính thân thiện dễ sử dụng, nhưng cũng đầy hiệu quả của nó.

Lần xuất bản này, thực hiện theo đơn đặt hàng của Vụ Giáo dục chuyên nghiệp, trước hết phục vụ các bạn sinh viên cũng như giáo viên các trường Trung học chuyên nghiệp, Cao đẳng và làm tài liệu tham khảo tốt cho những người muốn tìm hiểu về tin học.

Tác giả xin chân thành cảm ơn Vụ Giáo dục chuyên nghiệp, Nhà xuất bản Giáo dục đã tạo điều kiện cho cuốn sách sớm ra đời. Tác giả cũng đặc biệt cảm ơn các bạn đồng nghiệp trong Khoa công nghệ thông tin, Trường Đại học Bách khoa Hà Nội đã giúp đỡ tài liệu, góp ý kiến để cuốn sách được hoàn chỉnh hơn.

Cuốn sách chắc chắn vẫn không tránh khỏi những thiếu sót. Rất mong nhận được những ý kiến đóng góp của bạn đọc để lần tái bản sau được hoàn chỉnh hơn.

Mọi đóng góp xin gửi về theo địa chỉ :

Tô Văn Nam

Bộ môn HTTT, Khoa CNTT.

Đại học Bách khoa Hà Nội.

namtv@it-hut.edu.vn

TÁC GIẢ

Chương 1

NHỮNG KHÁI NIỆM CƠ BẢN VỀ TIN HỌC

1. KHÁI NIỆM VỀ THÔNG TIN VÀ XỬ LÍ THÔNG TIN

Khi xã hội ngày càng phát triển thì khối lượng thông tin cần xử lí càng nhiều. Đã có lúc loài người có thể rơi vào sự khủng hoảng thông tin vì thông tin nhiều tới mức vượt ra ngoài khả năng xử lí của con người nếu như chúng ta chỉ sử dụng các công cụ tính toán thô sơ như bàn tính gỗ, máy tính quay tay,...

Cũng có thể nói rằng nếu thông tin không được xử lí một cách nhanh chóng, kịp thời, đầy đủ và chính xác thì chắc chắn rằng không thể có được những tiến bộ vượt bậc trong khoa học, kĩ thuật, sản xuất và kinh doanh như hiện nay. Để có thể đạt được những thành tựu như vậy con người đã cố công tìm tòi, nghiên cứu, thiết kế và chế tạo ra được một công cụ tuyệt vời, mang tính cách mạng, đó chính là máy tính điện tử (MTĐT), hay còn gọi là máy điện toán.

Sự ra đời của MTĐT đã khai sinh ra một ngành khoa học mới, đầy sức hấp dẫn với mọi tầng lớp trong xã hội : từ các nhà nghiên cứu khoa học đến các nhà kinh doanh, từ người cao tuổi đến các cháu bé với các trò chơi điện tử.

1.1. Thông tin là một khái niệm để chỉ sự hiểu biết, nhận thức, mô tả về sự vật, sự việc, sự kiện và hiện tượng mà con người thu nhận được trong quá trình nghiên cứu, khảo sát, quan sát (trình bày, hình thức hoá được).

Một số tính chất của thông tin :

– Tính định hướng : Phản ánh mối quan hệ nguồn tin và nơi nhận tin (thông tin chỉ có nghĩa nếu nó làm giàu nơi nhận tin).

– Tính tương đối : Trong khoảng thời gian nhận tin, sự vật hiện tượng đã biến đổi khác trước.

– Tính cục bộ : Thông tin có thể đúng (hoặc có ý nghĩa) với hệ thống này mà không có ý nghĩa với hệ thống khác.

Dữ liệu có thể hiểu là một dạng biểu diễn của thông tin. Những ý nghĩa rút ra từ dữ liệu là thông tin ; người trao đổi với người hoặc với máy các bản tin bằng lời nói, chữ viết, hình vẽ,... Bên trong các máy tin học, thông tin được biểu diễn bằng các số nhị phân, thường gọi là các số liệu hoặc dữ liệu.

Dữ liệu thường được chia làm hai loại :

- Dữ liệu cơ bản như chữ số, chữ cái và các kí hiệu đặc biệt.
- Dữ liệu có cấu trúc được xây dựng lên từ các dữ liệu cơ bản.

Vật mang tin chính là hình thức cụ thể của thông tin (có thể là ngôn ngữ, chữ số, kí hiệu, xung điện,...). Tùy thuộc hình thức vật mang tin ta có thể chia chúng thành hai loại :

- Vật mang là các đại lượng vật lí liên tục – thông tin tương tự (ANALOG).
- Vật mang là các đại lượng vật lí rời rạc – thông tin số (DIGITAL).

Đơn vị đo thông tin

Đơn vị bé nhất để đo thông tin là **bit**

- 1 bit (*binary digit*) ghi được hai giá trị đại diện cho hai trạng thái phân biệt nhau. Ví dụ : đúng/sai, nhiễm từ/không nhiễm từ, có điện/không có điện,...

- byte : 1 byte được tổ hợp từ 8 bit xếp liên tiếp, ghi được $2^8 = 256$ giá trị khác nhau.

- KByte : 1 KByte = 2^{10} Byte = 1024 Byte

- MByte : 1 MByte = 2^{10} KByte = 1024 KByte

- GByte :

1.2. Xử lí thông tin

Muốn xử lí được thông tin bằng máy tính, thông tin phải thoả mãn các điều kiện sau :

- Khách quan : mang một ý nghĩa duy nhất không tùy thuộc vào suy nghĩ chủ quan.

- Đo được : xác định bằng một đại lượng đo cụ thể.

- Rời rạc : các giá trị kế cận của nó là rời nhau.

Các phép xử lí tin :

- Mã tin : là việc biểu diễn thông tin dưới dạng vật lí, không thời gian nhất định, thuận tiện cho các phép xử lí khác.

- Thu thập tin : phép lấy các mã số liệu/bản tin từ sự vật, sự việc khách quan.
- Lựa chọn, sắp xếp.
- Lưu trữ số liệu : ghi số liệu trên những vật mang giữ chuyên môn (bộ nhớ, đĩa từ, băng từ,...)
- Biến đổi số liệu : các phép tác động lên các số liệu vào để tạo nên số liệu sản phẩm ra (đây chính là phép xử lý quan trọng nhất).
- Truyền dẫn thông tin : gửi các bản tin từ nơi phát tới nơi nhận.
-

Tin học

Theo từ điển Larousse Tin học, thì *Tin học* (hay còn gọi là *Công nghệ thông tin*) là tập hợp các ngành khoa học, kĩ thuật, kinh tế-xã hội vận dụng vào việc xử lí thông tin và sự tự động hoá nó.

Để nói tới ngành khoa học mới này ở Mỹ người ta dùng thuật ngữ *Computer science*, ở Nhật là *Information technology*, còn ở Pháp *informatique* (*Informatique* = *Information* + *Automatique*).

2. ỨNG DỤNG CỦA CÔNG NGHỆ THÔNG TIN

- Tính toán khoa học kĩ thuật.
- Quản lí thông tin.
- Soạn thảo in ấn.
- Graphic + Trò chơi.
- Trí tuệ nhân tạo.
- ...

3. BIỂU DIỄN THÔNG TIN (HỆ ĐẾM, MÃ)

Hệ đếm là tập các kí hiệu và quy tắc xác định, dùng để biểu diễn và tính toán các giá trị số.

3.1. Các loại hệ đếm

3.1.1. Hệ đếm không định vị

Chúng ta chỉ đề cập đến hệ đếm không định vị thông dụng nhất còn tồn tại đến ngày nay, đó là hệ La-mã ; trong hệ đếm này mỗi kí hiệu biểu thị một giá trị hằng. Dạng mới nhất của những chữ số La-mã như sau :

$$I = 1, \quad V = 5, \quad X = 10, \quad D = 500, \quad M = 1000$$

Trước kia hình dạng của chúng hơi khác ; chẳng hạn, như số 1000 trước đây được biểu thị bằng dấu "(|)", còn số 500 bằng dấu "|)".

Về nguồn gốc của chữ số La-mã, không có ý kiến nào đáng tin cậy cả. Có thể là chữ số V thoát tiên được dùng để biểu thị bàn tay, còn chữ số X thì do hai chữ số năm hợp thành. Cũng như vậy dấu biểu thị 1000 có thể được tạo thành từ hai dấu biểu thị 500 (hoặc ngược lại).

Tất cả các số nguyên đều được viết bằng cách lặp lại những số trên, theo quy tắc sau :

- Nếu một chữ số lớn đứng trước một chữ số bé thì chúng cộng lại với nhau, còn nếu chữ số bé đứng trước chữ số lớn (trong trường hợp này chữ số bé không được lặp lại) thì số bé được trừ vào số lớn. Ví dụ :

$$VI = 5 + 1 = 6, \quad VII = 5 + 2 = 7, \quad IV = 5 - 1 = 4.$$

- Cùng một số không viết quá ba lần.

3.1.2. Hệ đếm định vị

Giá trị của mỗi chữ số không những phụ thuộc vào bản thân nó mà còn phụ thuộc vào vị trí của chúng trong con số. Ở đây sẽ giới thiệu một số hệ đếm điển hình nhất.

Về nguyên tắc các hệ đếm định vị đều biểu diễn một số nguyên hay một số thực bất kì theo dạng sau :

$$N = a_{n-1}a_{n-2}a_{n-3}... a_1a_0 = a_0.s^0 + a_1.s^1 + ... + a_{n-1}.s^{n-1}$$

$$R = a_{n-1}a_{n-2}a_{n-3}... a_1a_0a_{-1}... a_{-m+1}a_{-m} = \\ = a_0.s^0 + ... + a_{n-1}.s^{n-1} + a_{-1}.s^{-1} + ... + a_{-m}.s^{-m}$$

Trong đó N là một số nguyên có n chữ số còn R là số thực có (m+n) chữ số. Chữ số a_i tại vị trí i (i = 0,...,n-1, -1,..., -m) được gọi là trọng số ; giá trị s là cơ số của hệ đếm ; hệ đếm được đặt tên theo giá trị của cơ số. Chẳng hạn nếu s=10 ta có hệ đếm 10 (hay thập phân, hệ đếm cơ số 10), còn với s = 2 ta có hệ đếm 2 (hay hệ nhị phân, hệ đếm cơ số 2),...

Giá trị của s cũng cho ta biết số kí tự cần dùng để biểu diễn trị số (số lượng các kí tự được dùng gọi là cơ số của hệ đếm) ; điều này có nghĩa là :

- Với s = 2 hệ đếm cần hai kí tự 0, 1 để biểu diễn.
- Với s = 8 ta cần 8 kí tự 0, 1, 2, 3, 4, 5, 6, 7 để biểu diễn,...

– Một cách tổng quát đối với hệ đếm cơ số s ta dùng s kí tự $0, 1, 2, 3, \dots, s-1$ để biểu diễn.

• Hệ đếm thập phân (hệ đếm cơ số 10)

Trong hệ đếm này người ta dùng 10 kí tự để biểu diễn một số :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Quy ước :

– Số lượng các kí hiệu được gọi là cơ số của hệ đếm.

– Số mũ của cơ số xác định giá trị định lượng của mỗi đơn vị. Mỗi một đơn vị chiếm một vị trí xác định trong hàng số :

$$0,01 = 10^{-2}, 0,1 = 10^{-1} \dots$$

$$\text{Ví dụ : } 536,4 = 5 \cdot 10^2 + 3 \cdot 10^1 + 6 \cdot 10^0 + 4 \cdot 10^{-1}$$

Một cách tổng quát chúng ta có thể xây dựng hệ đếm b phân như sau (với $b > 1$, integer) :

Trong hệ đếm này sử dụng b kí hiệu để biểu diễn các giá trị : $0, 1, \dots, b-1$.

Quy tắc :

Giá trị của số N với biểu diễn :

$$N = d_{n-1}d_{n-2} \dots d_1d_0, d_{-1}d_{-2} \dots d_{-m}$$

có thể tính theo công thức :

$$d_{n-1}b^{n-1} + d_{n-2}b^{n-2} + \dots + d_1b^1 + d_0b^0 + d_{-1}b^{-1} + d_{-2}b^{-2} + \dots + d_{-m}b^{-m}$$

ở đây : $0 \leq d_i < b$, n bằng số lượng các chữ số bên trái còn m bằng số lượng các chữ số bên phải. dấu phẩy b phân.

• Hệ nhị phân (hệ đếm cơ số 2)

Hệ nhị phân hình thành từ cơ sở đại số Boole, xuất hiện từ cuối thế kỉ 19. Hệ đếm này thực sự tìm được sự ứng dụng rộng rãi từ khi có mạch điện logic hai trạng thái (đóng/mở).

Với cơ số $s = 2$, hệ này chỉ cần hai kí tự để biểu diễn một số nguyên bất kì ; mỗi kí tự (hay mỗi trị số) của hệ nhị phân được gọi là một bit (binary digit). Cách biểu diễn bit khá đa dạng ; thông thường người ta chọn hai kí tự "0" và "1" để biểu diễn giá trị không và một.

Đối với các mạch điện tử, mỗi bit có thể được biểu diễn bằng một hiệu điện thế (0 V(vôn) là 0 và 5 V là 1⁽¹⁾).

Ví dụ : 0, 1, 10011,....

• Hệ đếm cơ số 8

Hệ đếm cơ số 8, hay còn gọi là hệ bát phân, là một trong những hệ thường được sử dụng để biểu diễn số trong tin học ; để biểu diễn số, hệ này sử dụng tám chữ số là 0, 1, 2, 3, 4, 5, 6, 7.

Các số hệ cơ số 8 có dạng như sau : 345, 2, 231,....

• Hệ đếm cơ số 16 (hệ He xơ)

Trong hệ đếm này để biểu diễn các giá trị người ta sử dụng 16 kí tự sau :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Ví dụ : C69F, 10A7.

3.2. Chuyển đổi và biểu diễn số

3.2.1. Biến đổi số ở hệ đếm bất kì sang hệ đếm thập phân

Giả sử ta có số N trong hệ đếm cơ số b :

$$N_b = d_{n-1} d_{n-2} \dots d_1 d_0 d_{-1} d_{-2} \dots d_{-m}$$

Vậy N sẽ tương đương với số ở hệ 10 theo công thức :

$$N_{10} = d_{n-1}b^{n-1} + d_{n-2}b^{n-2} + \dots d_1b^1 + d_0b^0 + d_{-1}b^{-1} + d_{-2}b^{-2} + \dots d_{-m}b^{-m}$$

ở đây b là cơ số của hệ đếm.

Chẳng hạn số 37 có biểu diễn nhị phân là 100101 ta sẽ chuyển sang hệ đếm thập phân :

$$1.2^5 + 0.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = 37$$

3.2.2. Biến đổi số ở hệ đếm thập phân sang hệ đếm có cơ số bất kì

Ta thực hiện ba bước sau :

- Biến đổi phần nguyên
- Biến đổi phần phân
- Ghép phần nguyên và phần phân

$$N_{10} = N_{10} \cdot F_{10} = N_b = N_b \cdot F_b$$

(1) Trong các loại vi mạch thuộc thế hệ mới có thể là 3, 3 V ; 2, 8 V hay thấp hơn nữa.

1. Biến đổi phần nguyên

Chia $N_{10} = N_0$ cho b được N_1 dư d_0 ,

Chia N_1 cho b được N_2 dư d_1 ,

• • •

Chia N_n cho b được 0 dư d_{n-1} ,

lúc đó $N_b = d_{n-1} d_{n-2} \dots d_1 d_0$.

2. Biến đổi phần phân

Nhân $F_{10} = F_{-1}$ với b được nguyên d_{-1} và phần phân F_{-2}

Nhân F_{-3} với b được nguyên d_{-2} và phần phân F_{-3}

□ □ □ □

Nhân F_{-m} với b được nguyên d_{-m} và phần phân 0

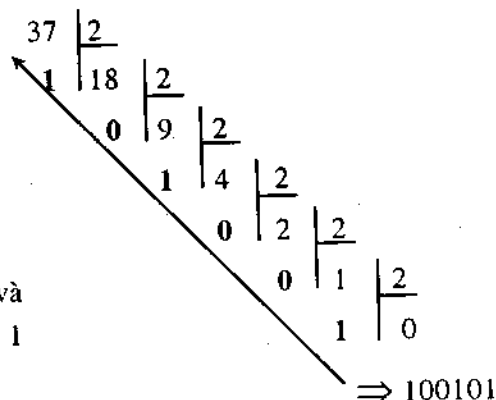
lúc đó $F_b = 0$, $d_{-1} \ d_{-2} \ \dots \ d_{-m}$

3. Ghép lại ta có :

$$N_{10} = N_b = N_b \cdot F_b$$

Ví dụ : Giả sử ta cần chuyển đổi 37. 25 sang biểu diễn nhị phân ta tiến hành như sau :

a) Biến đổi phần nguyên 37 : xem hình 1.4.



Hình 1.1

c) Ghép phần nguyên với phần phân :

$$37.25 = 100101.01$$

3.2.3. Số nhị phân có dấu kiểu mã bù 2 :

Quy tắc :

- Các bit vẫn có trọng số tuyệt đối như của mã nhị phân dương.
- Riêng bit cao nhất quy ước mang dấu âm, còn các bit thấp hơn đều mang dấu dương (mấy thường dùng 8 hoặc 16 bit để biểu diễn số).

Ví dụ : Với số 8 bit thì các giá trị tương ứng như sau :

Bit	7	6	5	4	3	2	1	0
Giá trị	$-2^7 = -128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$

Một số phép toán nhị phân cơ bản :

- Phép cộng hai số nhị phân :

$$0 + 0 = 0$$

$$0 + 1 = 1 + 0 = 1$$

$$1 + 1 = 0 \quad \text{nhớ 1 bit lên hàng đơn vị trên}$$

$$1 + 1 + 1 = 1 \quad \text{nhớ 1 bit lên hàng đơn vị trên}$$

- Quy tắc đổi dấu mã bù 2 :

- Bù tất cả các bit của mã ($1 \rightarrow 0, 0 \rightarrow 1$)

- Cộng thêm 1.

Ví dụ :

$$3_{10} = 0000\ 0011_2$$

+ Bù tất cả các bit của mã ta được : 1111 1100.

+ Cộng thêm 1 : 0000 0001.

+ Thực hiện phép cộng :

$$-3_{10} = 1111\ 1100 + 0000\ 0001 = 1111\ 1101_2.$$

- Phép trừ hai số :

$$A - B = A + (-B)$$

Cụ thể :

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$0 - 1 = 1 \text{ nhớ 1 bit lên hàng đơn vị trên.}$$

• Phép nhân hai số (phép dịch trái liên tiếp số bị nhân rồi cộng vào số nhân) :

$$0 * 0 = 0$$

$$0 * 1 = 0$$

$$1 * 0 = 0$$

$$1 * 1 = 1$$

• Phép chia hai số ngược lại phép nhân :

$$0/1 = 0$$

$$1/1 = 1.$$

3.3. Biểu diễn kí tự

Trong máy tính các kí tự cũng cần phải được chuyển thành chuỗi các số nhị phân. Có một số bộ mã kí tự thực hiện việc chuyển đổi này, tuy nhiên hiện nay có hai bộ mã được dùng phổ biến đó là ASCII và Unicode.

3.3.1. Bộ mã ASCII

Bộ mã kí tự ASCII (American Standard Code for Information Interchange) ; Bộ mã lúc ban đầu chỉ bao gồm 128 kí tự và được gọi là bộ mã ASCII chuẩn, bộ mã này chỉ được thiết kế cho nước Mỹ dùng để biểu diễn các chữ cái Latin, chữ số Ả-rập, các dấu đặc biệt, các lệnh và thông báo truyền tin giữa máy phát và máy nhận. Mã có độ dài cơ bản 7 bit (bao gồm 128 mã từ 0-127).

Về sau để tạo điều kiện cho các nước khác muốn đưa chữ viết của họ vào máy tính, các nhà chế tạo máy tính và các nhà phát triển phần mềm đã mở rộng bộ mã bằng cách sử dụng cả một byte (8 bit) để mã hóa kí tự ; như vậy mã ASCII mở rộng bao gồm 256 kí tự với mã từ 0 đến 255. Với bộ mã ASCII mở rộng này, phần nào yêu cầu của các hệ cụ thể đã được đáp ứng.

Bộ mã ASCII chuẩn mã hóa cho 128 kí tự có mã từ 0 đến 127, trong đó 33 mã dùng để điều khiển truyền thông, điều khiển màn hình, máy in và các mục đích khác ; các mã còn lại gán cho các kí tự thông dụng để hiển thị và in ấn bao gồm các chữ cái hoa và chữ thường của bảng chữ cái Latin, các chữ số, các dấu câu thông thường, các dấu phép toán cơ bản và dấu cách.

• Các kí tự hiển thị thông dụng :

Các mã từ 32 đến 126 dùng để mã hóa cho các kí tự hiển thị thông dụng ; người ta chia chúng thành 3 nhóm như sau :

– Các chữ cái Latin bao gồm :

+ 26 chữ cái thường từ a đến z có mã từ 97 đến 122 :

'a' có mã là 97,

'b' có mã là 98,

....,

'z' có mã là 122.

+ 26 chữ cái hoa từ A đến Z có mã từ 65 đến 90 :

'A' có mã là 65,

'B' có mã là 66,

....,

'Z' có mã là 90.

Như vậy mã của các chữ cái cùng loại được xếp tăng dần theo thứ tự trong bảng chữ cái, điều này rất cần thiết cho việc sắp xếp tuần tự theo ABC. Giữa chữ cái thường và chữ cái hoa tương ứng có mã chênh lệch nhau là 32.

– 10 chữ số thập phân từ 0 đến 9 được gán mã từ 48 đến 57 :

'0' có mã là 48,

'1' có mã là 49,

....,

'9' có mã là 57.

– Các kí tự khác như : các dấu chấm câu, các phép toán, một số kí tự thông dụng, dấu cách.

• Các kí tự điều khiển :

32 kí tự đầu tiên của bảng ASCII (có mã từ 0 đến 32) và một mã cuối cùng có mã là 127 dùng để mã hóa các thông tin điều khiển, vì vậy chúng có tên là các mã kí tự điều khiển ; các mã này dành cho việc chuyển những thông tin đến màn hình, máy in hay đến một máy tính khác..

• Các kí tự mở rộng : các kí tự này có mã từ 128 đến 255, đây là phần "tùy chọn" của các nhà chế tạo máy tính và phát triển phần mềm ; sau đây là một số bộ mã mở rộng :

– Bộ mã mở rộng của IBM dùng cho máy tính dòng IBM.

– Bộ mã mở rộng của Appel dùng cho máy tính dòng Machintosh.

– Các nhà tin học Việt Nam cũng đã thay đổi phần này để mã hóa cho các kí tự riêng của tiếng Việt, ví dụ như bộ mã TCVN5712.

3.3.2. Bộ mã Unicode

Ngày nay máy tính đã toàn cầu hóa, mà một hình ảnh cụ thể là mạng Internet, do vậy bộ mã ASCII đã bộc lộ khả năng mã hóa hạn chế của nó.

Để thống nhất một bộ mã trên toàn thế giới, các nhà sản xuất máy tính hàng đầu trên thế giới đã đề xuất bộ mã 16 bit mang tên Unicode. Vì dùng tới 16 bit để mã hóa (mã hóa được 2^{16} kí tự), vì vậy nó đủ lớn để đáp ứng cho việc mã hóa tất cả các ngôn ngữ trên toàn thế giới. Đặc điểm chính của mã Unicode là nó không chứa các kí tự điều khiển mà dành tất cả để mã hóa kí tự ; bảng sau đây cho chúng ta biết sơ bộ về cách phân bố mã chuẩn trong Unicode :

Mã thập phân	Kí tự
0 đến 8191	Chữ cái Anh, Latin1, Châu Âu, Latin mở rộng, chữ cái phiên âm, Hy Lạp, Nga, Armenia, Do thái, Ả rập, Ethiopi, Dvanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kanada, Malaixia, Thái, Lào, Miến điện, Khme, Tây tạng, Mông cổ, Georgi <i>Kí hiệu</i>
8192 – 12287	Chữ tượng hình, chữ cái Hàn, Nhật, Hàn
12288 – 16383	Chữ tượng hình Hàn, Nhật, Hàn
16384 – 59391	Dành cho người sử dụng
59392 – 65024	Vùng tương thích
65025 – 65536	Cho các mục đích trong tương lai

8192 giá trị đầu dành cho chữ cái chuẩn ;

4096 giá trị tiếp theo dành cho kí hiệu toán học, kĩ thuật,... ;

4096 giá trị sau nữa là dành cho chữ tượng hình ;

....

Unicode quy định các chữ cái có âm tiết trong tiếng Việt là các kí tự tổ hợp (composite character). Ví dụ chữ 'ă' là tổ hợp của hai chữ 'a' và '^' ; mỗi kí tự tổ hợp bao gồm nguyên âm cơ sở được nối tiếp bởi kí tự dấu thanh. Nguyên âm cơ sở và dấu thanh được đặt vào cùng một vị trí khi hiển thị. Nếu chữ cái được tổ hợp từ hai hay nhiều kí tự âm tiết (ví dụ 'ă') thứ tự các dấu không quan trọng nếu không có luật chính tả cụ thể.

Các kí tự được tổ hợp từ trước (precomposed character) như chữ 'd' chỉ dùng một mã duy nhất để mô tả.

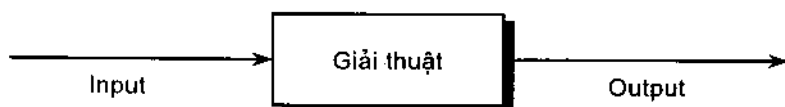
Để biểu diễn tiếng Việt ta cần :

- 33 chữ cái hoa ;
- 33 chữ cái thường ;
- 5 dấu thanh : huyền (`), hỏi (?), ngã (~) sắc ('), nặng (.).

4. GIẢI THUẬT, BIỂU DIỄN GIẢI THUẬT, CHƯƠNG TRÌNH

4.1. Khái niệm về giải thuật

Giải thuật là một hệ thống các quy tắc nhằm xác định một dãy các thao tác trên những đối tượng sao cho từ một bộ dữ liệu vào sau khi thực hiện lần lượt một số hữu hạn các thao tác xử lí ta đạt được mục tiêu định trước :



4.2. Các đặc trưng của giải thuật

Một giải thuật phải có 4 đặc trưng sau :

- Tính dừng : thuật toán phải dừng sau một số hữu hạn bước thực hiện.
- Tính xác định : mỗi bước phải hết sức rõ ràng, không nhập nhằng.
- Tính phổ dụng : có thể dùng để giải một lớp các bài toán.
- Tính hiệu quả : tiết kiệm tài nguyên (không gian, thời gian).

4.3. Một số cách biểu diễn giải thuật

Để hình dung ra trình tự chặt chẽ các bước thao tác, các công việc cụ thể phải làm, trước khi viết chương trình người ta phải mô tả thuật toán. Có hai phương pháp để mô tả thuật toán.

4.3.1. Ngôn ngữ mô phỏng

Trong phương pháp này người ta dùng ngôn ngữ tự nhiên để miêu tả giải thuật một cách chi tiết.

Ví dụ :

Thuật toán nấu cơm có thể được miêu tả như sau :

B1 : vo gạo, đổ nước

B2 : đun nước

B3 : nếu chưa sôi quay lại bước 2

B4 : cho gạo vào nồi

B5 : đun tiếp

B6 : nước chưa cạn quay lại bước 5

B7 : đun nhỏ lửa

B8 : nếu chưa chín quay lại bước 7

B9 : tắt bếp

4.3.2. Sơ đồ khối

Đây là một phương pháp hay được sử dụng để mô tả các giải thuật "tính toán". Ưu điểm nổi bật của sơ đồ khối là tính trực quan, nó cung cấp cho ta toàn cảnh của quá trình xử lý dữ liệu mà không cần giải thích bằng văn bản.

Sơ đồ khối biểu diễn giải thuật thông qua các khối hình học, các mũi tên chỉ đường đi ; nói một cách cụ thể hơn nó bao gồm các thành phần sau :

- Mỗi giải thuật luôn luôn có một điểm xuất phát và một điểm kết thúc thể hiện bằng hình ôvan sau :



- Các xử lý được đặt trong các hình chữ nhật :

$$D = b^2 - 4ac$$

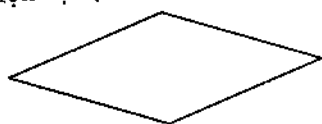
- Để chỉ hướng đi tiếp theo của giải thuật người ta dùng mũi tên :



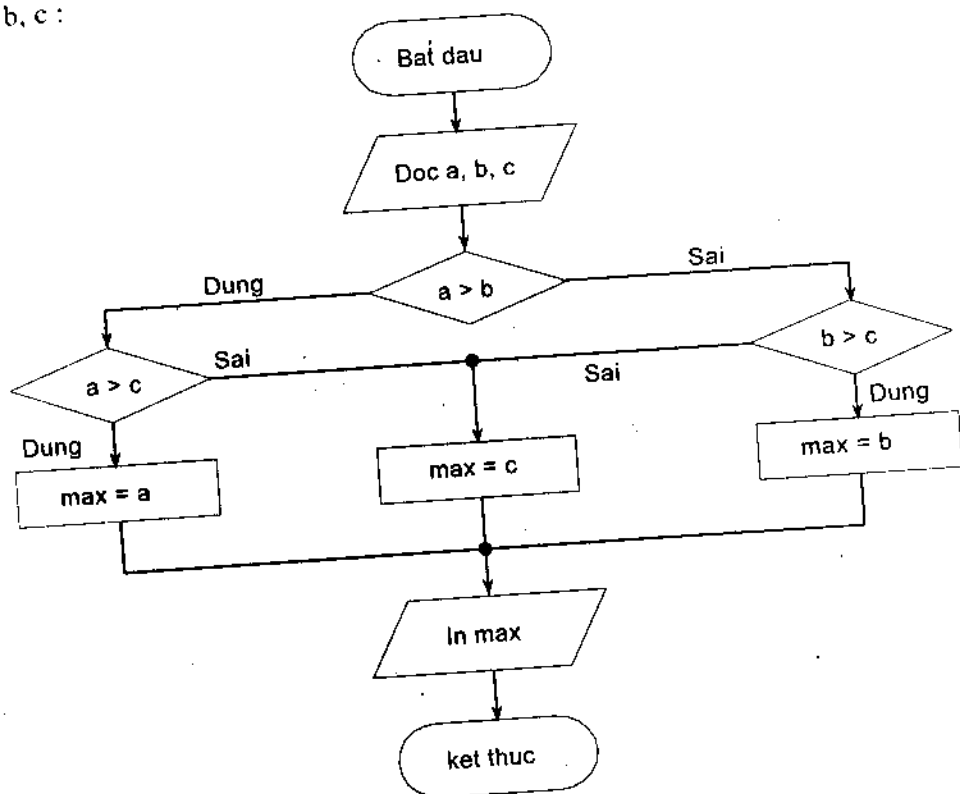
- Các thao tác vào ra dữ liệu được thể hiện bằng hình bình hành :



- Hình thoi để thể hiện sự lựa chọn



Ví dụ sơ đồ khối trên hình 1.2 mô tả giải thuật tìm giá trị lớn nhất của ba số a, b, c :



Hình 1.2. Tìm giá trị lớn nhất của ba số a, b, c

Chương trình là biểu diễn giải thuật trên một ngôn ngữ lập trình cụ thể

5. NGÔN NGỮ LẬP TRÌNH, CHƯƠNG TRÌNH DỊCH

Là hệ thống có một bộ các câu lệnh và một bộ các quy tắc về cú pháp từ đó có thể lập được các chương trình biểu diễn một giải thuật xác định.

Chương trình viết bằng ngôn ngữ cấp cao (ngôn ngữ lập trình) gọi là chương trình nguồn (CTN). Chương trình viết bằng ngôn ngữ máy gọi là chương trình đích (CTĐ).

5.1. Phân loại ngôn ngữ lập trình

Loại NN	Đối tượng xử lý	Lệnh xử lý	Đặc điểm
Assembler (Bậc thấp)	byte, bit,	Dãy số nhị phân	Máy hiểu, phụ thuộc máy, người khó nhớ
Bậc cao	Trừu tượng : số nguyên, thực chữ, file,...	Lệnh tiếng Anh	Máy không hiểu, độc lập máy, người dễ hiểu

5.2. Chương trình dịch (CTD)

Mỗi ngôn ngữ lập trình đều có một chương trình đặc biệt để thực hiện việc dịch chương trình nguồn viết bằng ngôn ngữ đó sang ngôn ngữ máy ; chương trình đặc biệt đó gọi là chương trình dịch (trình dịch). Có hai loại trình dịch là biên dịch và thông dịch:

	Biên dịch (compiler)	Thông dịch (Interpreter)
Cách thực hiện	Chương trình nguồn được nạp vào bộ nhớ trong CTD dịch CTN thành CTĐ, máy tính thực hiện CTĐ (giai đoạn dịch & thực hiện phân biệt) Sau khi dịch xong CTĐ có thể lưu lại (dịch 1 lần, chạy * n lần) Tốn nhiều bộ nhớ Chạy nhanh	Giai đoạn dịch và thực hiện tiến hành xen kẽ theo từng đơn vị cú pháp Mỗi lần chạy phải dịch lại : $n * (\text{dịch} + \text{chạy})$ Không cần bộ nhớ lưu trữ CTĐ Chạy chậm
Ví dụ	Dịch diễn văn	Thông ngôn

Bài tập

- Cho các số sau $X = 2002$ trong hệ thập phân, $Y = 010101111111$ trong hệ nhị phân, $Z = ABC$ trong hệ Hexa. Hãy chỉ ra cách sắp xếp đúng :
 - $X < Y < Z$
 - $Y < X < Z$
 - $Z < X < Y$
 - Cả a, b, c đều sai.

2. Số A được biểu diễn trong hệ đếm cơ số 8 có giá trị $A=0117$. Trong hệ thập phân giá trị của A là :

a) 91 ; b) 90 ; c) Cả a, b đều sai.

Hãy chọn cách trả lời đúng trong 3 cách trên.

3. Cho hai số nguyên không dấu A và B. A biểu diễn ở hệ 16 và có giá trị bằng AF3. B biểu diễn ở hệ 8 và có giá trị bằng 353. Hãy chọn cách trả lời đúng trong 3 cách trả lời sau :

a) $A < B$; b) $A = B$; c) $A > B$.

4. Số 7654321 có thể chấp nhận là số biểu diễn trong các hệ đếm :

a) 2 ; b) 8 ; c) 16

d) Các câu a, b, c đều sai

Hãy chọn cách trả lời đúng.

5. Cho hai số nguyên không dấu M và N. M biểu diễn ở hệ 16 và có giá trị bằng 555. N biểu diễn ở hệ 8 và có giá trị bằng 2002. Hãy chọn cách trả lời đúng trong 3 cách trả lời sau :

a) $M < N$

b) $M = N$

c) $M > N$.

6. Giả sử một số nguyên 16-bit có giá trị dưới dạng thập phân là 107. Hãy cho biết giá trị đó tương ứng với biểu diễn nào dưới đây :

a) 6A ở hệ đếm 16 ;

b) 006B ở hệ đếm 16 ;

c) 006C ở hệ đếm 16 ;

d) FF6B ở hệ đếm 16.

7. Theo bảng mã ASCII thì các biểu thức so sánh kí tự sau đây, biểu thức nào là đúng :

a) 'b' < 'Z' ;

b) 'a' > 'A' ;

c) '9' > 'A' ;

d) '4' > '5'.

3. Chọn phát biểu đúng nhất về bộ mã ASCII. 128 kí tự chuẩn của bộ mã ASCII bao gồm :

- a) Các kí tự điều khiển, các chữ cái tiếng Việt, các chữ số thập phân, các dấu câu, các kí hiệu toán thông dụng và một số kí tự thông dụng khác.
- b) Các chữ cái Latin, các chữ cái của các ngôn ngữ thông dụng ở châu Âu, các chữ số thập phân, các dấu câu, các kí hiệu toán thông dụng và một số kí tự thông dụng khác.
- c) Các kí tự điều khiển, các chữ cái Latin, các chữ số thập phân, các dấu câu, các kí hiệu toán thông dụng và một số kí tự thông dụng khác.
- d) Các kí tự điều khiển, các chữ cái Latin, các chữ số thập phân, các dấu câu, các chữ cái Hy-lạp.

9. Chọn câu trả lời đúng, BIT là kí hiệu viết tắt của :

- a) Binary Information Tranmission ;
- b) Binary Information Technology ;
- c) Binary Information uniT ;
- d) Binary Technology ;
- e) Cả a, b, c, d đều sai.

10. Cho các số sau $X=832$ trong hệ thập phân, $Y=010111111111$ trong hệ nhị phân, $Z=5AA$ trong hệ Hexa. Hãy chỉ ra cách sắp xếp đúng :

- a) $X < Z < Y$;
- b) $Y < X < Z$;
- c) $X < Y < Z$;
- d) Cả a, b, c đều sai.

11. Số 08032002 có thể chấp nhận là số biểu diễn trong các hệ đếm :

- a) Nhị phân (cơ số 2) ;
- b) Bát phân (cơ số 8) ;
- c) Hexa (cơ số 16) ;
- d) Thập phân (cơ số 10) ;
- e) Các câu a, b, c, d đều sai.

12. Dùng sơ đồ khối mô tả thuật toán cho phép tính giá trị của các biểu thức sau :

- a) $A = x^2 + y^2$;

b) $B = x + y + A$;

c) $C = xy + A - B^2$.

13. Vẽ sơ đồ khối mô tả thuật toán cho phép tính giá trị nhỏ nhất của ba số a, b, c .
14. Cho một dãy số thực bao gồm 120 phần tử, hãy mô tả thuật toán cho phép tìm giá trị lớn nhất của dãy đó.
15. Cho một dãy bao gồm 121 số nguyên, hãy mô tả thuật toán tìm ra số lượng số chẵn của dãy đó.
16. Hãy mô tả thuật toán cho phép tìm trung bình cộng của các phần tử dương trong một dãy số thực bao gồm 200 số thực.
17. Xét dãy số thực $x_1, x_2, \dots, x_{112}$, hãy tính số lượng số hạng đạt giá trị lớn nhất.
18. Cho một dãy số thực. Hãy mô tả các giải thuật :
- Sắp xếp lại dãy số đó theo thứ tự tăng dần.
 - Sắp xếp lại dãy số đó theo thứ tự giảm dần.
 - Sắp xếp các số âm lên đầu dãy, sau đó là các số còn lại.
 - Sắp xếp các số 0 lên đầu dãy, sau đó là các số âm, còn lại là các số dương.

Chương 2

MÁY TÍNH ĐIỆN TỬ

1. NGUYÊN LÝ VON NEUMANN

Nguyên lý này mang tên người phát minh ra nó, nhà toán học John Von Neumann gốc Hungari. Đây là một trong những phát minh vĩ đại của nhân loại trong lịch sử phát triển của máy tính. Lần đầu tiên khái niệm nhớ chương trình (*stored-program concept*) được công bố. Khái niệm nhớ chương trình có nghĩa là máy tính hoạt động theo chương trình được nhớ sẵn trong bộ nhớ của máy tính và khái niệm này đã trở thành nguyên tắc cơ bản của hầu hết các kiến trúc máy tính sau này.

Để có thể hiểu được nguyên lý Von Neumann, chúng ta hãy xây dựng một chương trình sử dụng những tấm bảng con tượng trưng cho các ô nhớ để thực hiện việc tính giá trị của biểu thức :

$$(a + b) * (c + d)$$

Trong đó a, b, c, d là những số cho trước, dấu * trong biểu thức là kí hiệu của phép toán nhân :

a

A

b

c

d

e

Hình 2.1

Cho $a = 26$, $b = 4$, $c = -10$ và $d = 5$; để tiện trình bày ta đặt tên cho các tấm bảng con chứa các giá trị tương ứng cũng là a , b , c , d . Ngoài ra chúng ta còn có thêm một tấm bảng A dùng để ghi các kết quả sau mỗi lần thực hiện một phép tính (xem hình 2.1).

Quá trình thực hiện biểu thức trên diễn ra như sau :

Thoạt tiên thực hiện phép tính $(a + b)$; kết quả của phép tính này được ghi tạm vào đâu đó, chẳng hạn vào e . Tiếp theo thực hiện phép toán $(c + d)$; kết quả của phép toán sau cùng này được lưu trữ trong A. Cuối cùng chỉ việc lấy giá trị trong A nhân với giá trị của e . Kết quả của biểu thức $(a + b) * (c + d)$ vừa thu được sẽ lưu trữ trong A. Đến lúc này ta có thể chuyển kết quả từ A sang một tấm bảng con nào đó, chẳng hạn sang chính tấm bảng con e .

Nội một cách chi tiết hơn nữa quá trình tính biểu thức trên diễn ra như sau :

Đọc a vào A : lấy nội dung của a ghi vào A ; lúc này A chứa 26.

Cộng A với b : Cộng số hiện có trong A(26) với số có trong tấm bảng $b(4)$; tổng số thu được ($26 + 4 = 30$) sẽ để trong A, lúc này A chứa 30.

Ghi A vào e : Ghi số hiện có trong A vào e , lúc này nội dung của e là 30.

Đọc c vào A : lấy nội dung của c ghi vào A ; lúc này nội dung cũ của A(30) sẽ mất đi thay vào đó là giá trị đọc từ $c(-10)$.

Cộng A với d : Cộng số hiện có trong A(-10) với số có trong tấm bảng $d(5)$; tổng số ($-10 + 5 = -5$) sẽ để trong A, lúc này A chứa -5.

Nhân A với e : Nhân số hiện có trong A(-5) với số hiện có trong e (30); tích số thu được ($-5 * 30 = -150$) để trong A.

Ghi A vào e : Ghi số hiện có trong A vào e .

Dùng việc tính toán.

Giá trị của biểu thức trên chính là nội dung của e và bằng -150.

Người ta nói quy trình tính toán trên được tổ chức theo câu lệnh một địa chỉ. Bảng A luôn luôn chứa các kết quả sau một lần thực hiện phép tính, đó chính là thanh ghi tổng, các bảng a , b , c , d và e là các ô nhớ.

Chúng ta hãy lược bỏ chi tiết để thu được nguyên lí làm việc của quá trình tính toán trên :

1. Đọc a (đọc giá trị của a vào A).
2. Cộng b (cộng nội dung của A với giá trị chứa trong b).
3. Ghi e (ghi nội dung của A vào e).

4. Đọc c (đọc giá trị của c vào A).
5. Cộng d (cộng nội dung của A với giá trị chứa trong d).
6. Nhân e (nhân nội dung chứa trong A với giá trị của e).
7. Ghi e (ghi nội dung của A vào e).
8. Dừng.

Người ta gọi dãy câu lệnh trên là chương trình mà một bộ xử lí nào đó phải thực hiện để tính giá trị của biểu thức $(a+b) * (c+d)$ và ghi giá trị đó vào e. Một chương trình như vậy có thể giao cho máy tính thực hiện. Ta có thể hình dung các ô nhớ của máy tính tựa như những tấm bảng con. Điểm khác nhau là các dữ liệu được ghi vào các ô nhớ theo nguyên tắc điện từ (tương tự như với các băng từ hoặc đĩa ghi âm hoặc ghi hình) và các từ trong bộ nhớ không gọi là a, b, c... mà gọi theo địa chỉ 0, 1, 2,...

Để CPU của máy tính có thể thực hiện tuần tự các câu lệnh của một chương trình ta phải sử dụng bàn phím và một số chương trình có sẵn để nạp chương trình của ta vào bộ nhớ một cách tuần tự.

Để minh họa ta lấy lại thí dụ trên : khi được nạp vào bộ nhớ mỗi dòng lệnh chiếm một ô nhớ, từ ô thứ 0 tới ô thứ 7 ; các giá trị của a, b, c và d cũng phải được nạp vào bộ nhớ, chẳng hạn ô thứ 8 (cho a), ô thứ 9 (cho b), 10 (cho c) và 11 (cho d) ; còn giá trị của e sẽ được ghi vào ô thứ 12.

Khi đó câu lệnh 1 (đọc a) sẽ được ghi trong máy là đọc 8 và được hiểu là yêu cầu CPU đọc nội dung chứa trong ô nhớ 8 vào thanh ghi A.

Ta có thể miêu tả hình trạng bộ nhớ sau khi nạp chương trình trên như hình 2.2.

0.	ĐỌC	8	Phân câu lệnh
1.	CỘNG	9	
2.	GHI	12	
3.	ĐỌC	10	
4.	CỘNG	11	
5.	NHÂN	12	
6.	GHI	12	
7.	DỪNG		Phân dữ liệu vào
8.		26	
9.		4	
10.		-10	
11.		5	
12.		0	Dữ liệu ra (lúc đầu là 0)

Hình 2.2

Các chương trình có thể rất lớn (chiếm nhiều ô nhớ), nhưng người lập trình hoàn toàn có thể bố trí bộ nhớ theo một trình tự nhất định nào đó ; bản thân máy tính có thể có từ vài chục vạn đến vài tỉ từ để lưu trữ chương trình phục vụ cho CPU.

Phần vừa trình bày minh họa cho nguyên lý hoạt động của MTĐT do Von Neumann đề xuất ; nội dung của nguyên lý này được phát biểu như sau : *"Các lệnh và dữ liệu cùng được lưu trữ trong bộ nhớ được tạo bởi các từ có địa chỉ"*.

Có thể thấy nguyên lý Von Neumann thể hiện hai yếu tố then chốt sau :

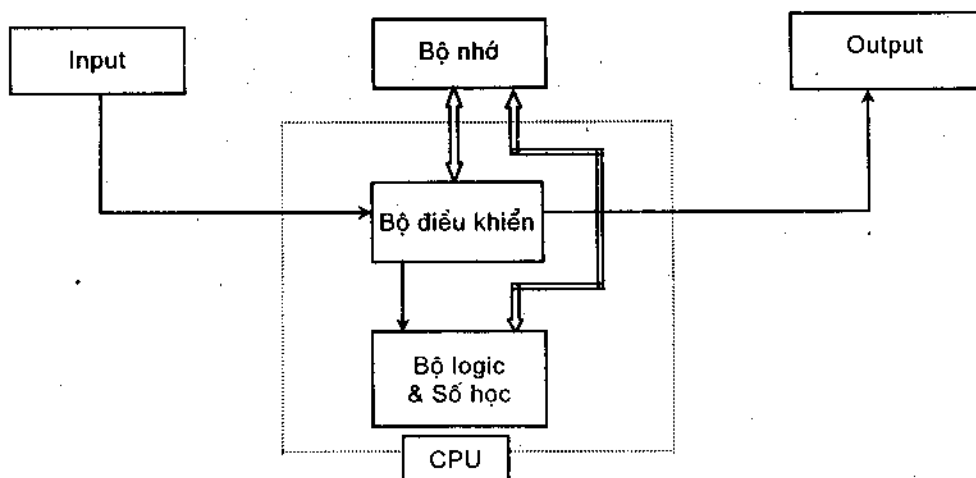
Điều khiển bằng chương trình : máy tính hoạt động theo sự chỉ dẫn, điều khiển của chương trình được lưu trữ trong bộ nhớ của nó ; các bước tác động được tiến hành theo các câu lệnh của chương trình ; chương trình chỉ dẫn cho máy tính biết phải làm gì và làm như thế nào.

Truy nhập theo địa chỉ : dữ liệu (dữ liệu vào, kết quả trung gian, kết quả cuối cùng, chương trình...) được lưu trữ trong RAM tại những vùng nhớ được định vị bằng các số thứ tự được gọi là địa chỉ. Dữ liệu được chỉ định và được truy nhập thông qua địa chỉ của ô nhớ chứa chúng.

Hai nội dung của nguyên lý Von Neumann chính là cơ sở đảm bảo cho máy tính thực hiện các chức năng xử lý thông tin một cách tự động.

2. KIẾN TRÚC MÁY TÍNH

Sơ đồ kiến trúc máy tính (hình 2.3):



Hình 2.3

• **Bộ xử lý trung tâm (CPU Central Processing Unit) :**

Điều khiển sự hoạt động của mọi khối khác.

Quá trình thực hiện chương trình là quá trình liên tiếp thực hiện từng lệnh :

- Đọc lệnh : CPU dùng PC (thanh đếm địa chỉ) ghi địa chỉ lệnh sắp thực hiện.
- Giải mã lệnh : căn cứ vào mã lệnh đọc được, để đọc nốt các thông tin cần thiết.
- Thực hiện lệnh : phát tín hiệu điều khiển Bộ số học và logic thực hiện phép toán.

Đặc trưng tốc độ xử lý của CPU là số nhịp đếm cơ sở cho việc thực hiện lệnh trong 1 giây (MHz).

Ví dụ về các CPU :

Do hãng Intel sản xuất : 8086, 80286, 80386, 80486 (DX, SX), Pentium, Pentium II, Pentium III, Pentium IV....

Ngoài ra các hãng như AMD, Cyrix, Motorola,... cũng đưa ra các sản phẩm tương đương.

• **Bộ điều khiển**

Khối này điều khiển mọi hoạt động của máy tính.

• **Bộ số học logic (ALU, Arithmetic & Logic Unit) :**

ALU thực hiện các phép toán số học và logic, phép điều khiển.

• **Bộ nhớ :**

Dùng để lưu trữ dữ liệu và chương trình. Bộ nhớ bao gồm bộ nhớ trong và bộ nhớ ngoài :

- Bộ nhớ trong : máy tính có thể truy nhập trực tiếp thông tin ghi trên nó. Chương trình muốn thực hiện phải được nạp vào bộ nhớ trong. Các loại vật liệu nhớ :

+ Vật liệu từ : dùng trạng thái từ tính để nhớ thông tin.

+ Bán dẫn : dùng trạng thái mạch điện để nhớ thông tin.

Bộ nhớ trong được tổ chức thành dãy liên tiếp các ô nhớ. Một ô thường có độ dài 8 bit = 1 byte. Các ô nhớ được gán địa chỉ và chứa các giá trị :

Loại	Đọc	Ghi	Xoá
ROM (Read Only Memory) :	*	0	0
RAM (Random Access Memory) :	*	*	*

- Bộ nhớ ngoài : dùng để lưu trữ dữ liệu và chương trình. Các thông tin này muốn dùng được phải được đọc vào bộ nhớ trong. Hiện nay bộ nhớ ngoài thường là vật liệu từ (băng từ, đĩa từ,...), đĩa quang học.

Bộ nhớ ngoài là nơi lưu trữ tài nguyên phần mềm của máy tính. Bộ nhớ ngoài được ghép với máy tính thông qua mô-đun nối ghép vào ra ; như vậy về chức năng bộ nhớ ngoài thuộc về bộ nhớ, nhưng về cấu trúc bộ nhớ ngoài thuộc cấu trúc vào ra. Dung lượng bộ nhớ ngoài lớn hơn dung lượng bộ nhớ trong rất nhiều, song tốc độ lại chậm hơn. Bộ nhớ ngoài thông dụng nhất là đĩa từ và đĩa quang.

+ Đĩa từ được chế tạo từ chất dẻo, trên có phủ lớp vật liệu từ tính. Đĩa được phân chia thành các rãnh (track), xếp theo vòng tròn đồng tâm, các track lại được chia thành các cung sector (512 byte) ; đối với đĩa cứng có khái niệm trụ đĩa (cylinder). Có 2 loại đĩa từ :

* Đĩa mềm (floppy)⁽¹⁾ có hai loại :

- 3.5 inch : 740Kb, 1.4 Mb

- 5.25 inch : 360Kb, 1.2 Mb

Là loại đĩa có giá thành rẻ, dễ sử dụng, đặc biệt có thể đưa vào hay lấy ra khỏi ổ đĩa một cách dễ dàng cho nên rất thông dụng.

* Đĩa cứng (hard disk) : là thành phần rất quan trọng trong máy tính, có dung lượng tăng rất nhanh ; vào những năm đầu thập kỷ 90 của thế kỷ 20 các đĩa cứng thường có dung lượng cỡ một vài trăm Mbyte, sang đầu thế kỷ 21 (sau khoảng 10 năm) các ổ cứng thông dụng có dung lượng cỡ hàng chục Gbyte, tức là tăng lên hàng trăm lần.

+ Đĩa quang : thông tin được lưu trữ trên đĩa quang dưới dạng thay đổi tính chất quang của bề mặt đĩa ; khi ghi thông tin số lên vật liệu quang người ta sử dụng tia laser để đốt cháy một vị trí nào đó khi muốn ghi giá trị 1 và để trắng một vị trí nào đó khi muốn ghi giá trị 0.

Còn khi muốn đọc dữ liệu trên đĩa quang, đầu đọc cũng phát tia laser lên bề mặt đĩa, khi đó tia phản xạ tương ứng với hai loại vị trí ghi 1 và 0 là khác nhau ; đầu đọc sẽ thu lại tia phản xạ để nhận lại dữ liệu đã ghi ở trên đĩa. Có 4 loại đĩa quang khác nhau :

(1) Hiện nay trên thị trường hầu như chỉ còn bán loại đĩa mềm 3.5 inch, 1.4 Mb

* **CD-ROM** (*Compact Disk Read Only Memory*) thông tin được ghi lên đĩa khi sản xuất đĩa. Để đọc các đĩa này người ta dùng ổ đĩa CD-ROM.

* **CD-R** (*Recordable CD*) khi sản xuất ra các đĩa này còn trắng (chưa ghi thông tin) ; để ghi dữ liệu lên loại đĩa này (nhưng chỉ ghi được một lần) người ta dùng ổ đĩa CD-R.

* **CD-RW** (*rewritable CD*) là loại đĩa quang có thể ghi đi ghi lại nhiều lần bằng một ổ đĩa đặc biệt.

* **DVD** (*Digital Video Disk*) đây là đĩa quang có dung lượng lớn và tốc độ nhanh hơn các đĩa quang thông thường ; đĩa DVD lưu trữ thông tin trên cả hai mặt đĩa.

Việc lưu trữ thông tin ở bộ nhớ ngoài được tổ chức thành các file (tệp) :

Loại file	Tên	Nội dung
file dữ liệu	xxxxxxxx.xxx	Chứa dữ liệu I/O cho các chương trình
file chương trình	*.EXE, *.COM, *, BAT	Chứa các lệnh điều khiển

• **Các thiết bị ngoại vi** (in put, out put) :

Các thiết bị này dùng để trao đổi dữ liệu giữa người sử dụng và máy tính :

– Bìa đục lỗ (punched card), băng đục lỗ (punched tape).

– Máy in (printer) có các loại :

+ Laser ;

+ Phun mực ;

+ In kim.

– Bàn phím (keyboard) : các phím được gom nhóm thành các khu vực chính như phím số, phím chức năng, phím điều khiển thuận tiện cho sử dụng.

– Màn hình (display) bao gồm các loại :

+ Đơn sắc (monochrom) ;

+ CGA, EGA, VGA,... ;

+ Tinh thể lỏng (LCD), loại này hiện nay thường dùng cho máy tính xách tay (notebook).

– Con chuột (mouse) : điều khiển con trỏ dịch chuyển trên màn hình và nhận lệnh thông qua bấm nút (2 hoặc 3 nút bấm) (phổ biến cho môi trường WINDOWS).

– Ngoài ra còn một số thiết bị ngoại vi khác như : scanner, ploter, CD driver, terminal,...

• Các thế hệ máy tính⁽¹⁾

Năm	Linh kiện (cứng)	Ngoại vi	Tốc độ
1950–1958 (Thế hệ 1)	Đèn chân không Nhớ bằng xuyên từ	Vào số liệu bằng bia đục lỗ Điều khiển bằng tay	300–3000pht/s (pht/s : phép tính trên giây)
1958–1964 (Thế hệ 2)	Bán dẫn Nhớ bằng xuyên từ	Vào số liệu bằng bia & băng đục lỗ Có chương trình dịch OS đơn giản	10, 000–100, 000pht/s
1964–1974 : (Thế hệ 3)	Vi mạch (IC) Nhớ bằng xuyên từ, màng mỏng từ	OS đa chương trình đa người dùng, phân chia time nhớ ngoài bằng đĩa cứng	100'000–1triệupht/s
1974–đến nay : (Thế hệ 4)	Vi mạch lớn và rất lớn Đa xử lí	Nhớ ngoài bằng đĩa cứng, đĩa mềm	10 triệu pht/s
(Thế hệ 5)	Đang nghiên cứu trong những năm 80–90. Máy có trí khôn nhân tạo. Máy có thể học, thu thập, suy diễn, quản lí cơ sở tri thức để tìm ra cách giải quyết bài toán.		

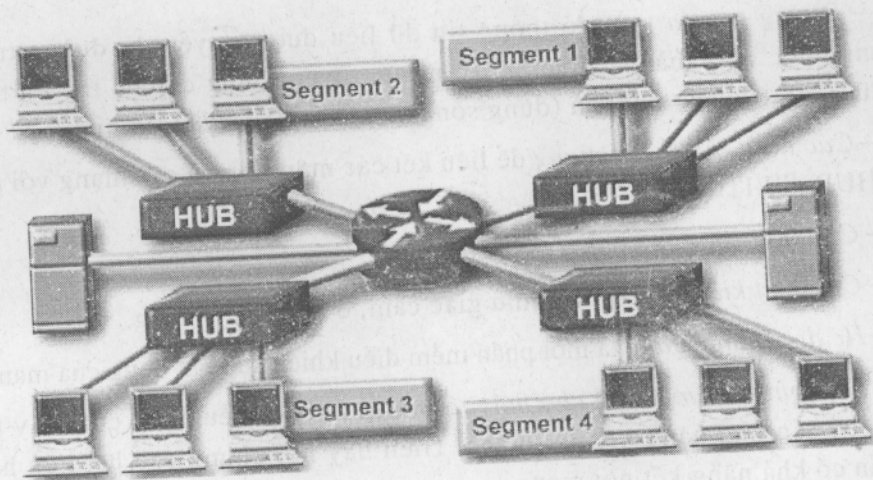
3. MẠNG MÁY TÍNH

3.1. Khái niệm về mạng

Ngày nay thuật ngữ *mạng* đã hiện diện và sẽ không bao giờ rời khỏi cuộc sống của chúng ta. *Mạng máy tính* hay *mạng* (*computer network, network*) là một tập hợp gồm hai hay nhiều máy tính hoặc thiết bị xử lí thông tin được kết nối với nhau qua các đường truyền và có sự trao đổi dữ liệu với nhau (xem hình 2.4).

Nhờ có mạng máy tính, thông tin dữ liệu từ một máy tính có thể được truyền sang máy tính khác. Có thể ví mạng máy tính như một hệ thống giao thông vận tải mà hàng hoá trên mạng là thông tin dữ liệu, máy tính là nhà máy lưu trữ xử lí dữ liệu, hệ thống đường truyền như là hệ thống đường xá giao thông.

(1) Việc phân chia các giai đoạn chỉ là tương đối



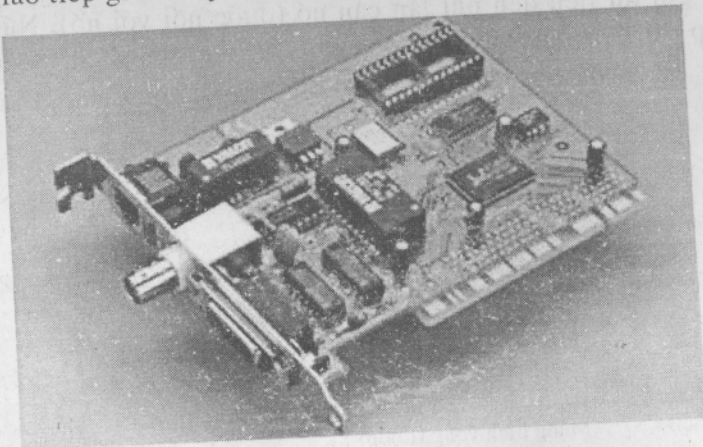
Hình 2.4. Mô hình một mạng máy tính

3.2. Các thành phần chính của một mạng máy tính

Một mạng máy tính có các thành phần chính sau :

- Các máy tính để xử lí, lưu trữ và trao đổi thông tin. Máy tính có thể phân làm hai loại là *máy chủ* (server) và *máy trạm* (workstation). Máy chủ dùng để quản trị mạng, quản lí tài nguyên dùng chung như file, máy in, Máy trạm là nơi người dùng (user) làm việc và truy cập vào mạng. Chú ý nếu là mạng bình đẳng (*peer-to-peer*) thì mọi máy đều là máy trạm. Trong thực tế một máy tính có thể vừa là máy chủ vừa là máy trạm. Ta cũng thường gọi mỗi máy tính trong mạng máy tính là một *nút* của mạng.

- *Vỉ mạng* (Network Interface Card, NIC) cho mỗi máy tính. Vì mạng có chức năng giao tiếp giữa máy tính và đường truyền (xem hình 2.5).



Hình 2.5. Vỉ mạng

– Đường truyền trên đó thông tin dữ liệu được truyền đi, đường truyền thường được chia thành 2 loại : truyền hữu tuyến (dùng cáp để truyền thông tin dữ liệu), truyền vô tuyến (dùng sóng vô tuyến để truyền tín hiệu).

– Các thiết bị kết nối mạng để liên kết các máy tính và các mạng với nhau như HUB, SWITCH.

– Các thiết bị đầu cuối (Terminal).

– Các phụ kiện mạng khác như giắc cắm, ổ cắm, dây mạng.

– Hệ điều hành mạng là một phần mềm điều khiển sự hoạt động của mạng.

– Các phần mềm mạng cho máy tính, cần khi hệ điều hành của máy tính không có sẵn khả năng kết nối mạng. Hiện nay nói chung các hệ điều hành đều sẵn có khả năng kết nối mạng.

– Các ứng dụng trên mạng. Ví dụ như Email, hệ quản trị cơ sở dữ liệu, ...

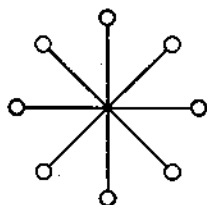
– Các tài nguyên dùng chung trên mạng như dữ liệu, các thiết bị (máy in, đĩa cứng, v. v...)

3.3. Kiến trúc mạng máy tính

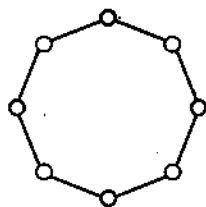
Kiến trúc mạng máy tính (network architecture) thể hiện cách kết nối máy tính với nhau và quy ước truyền dữ liệu giữa các máy tính như thế nào. Cách nối các máy tính với nhau gọi là hình trạng (topology) của mạng. Tập các quy ước truyền thông gọi là giao thức (protocol).

Có hai kiểu nối mạng chủ yếu là điểm–điểm (point to point) và quảng bá(broadcast).

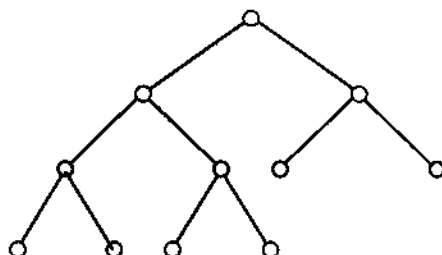
– Kiểu điểm–điểm các đường truyền nối các nút thành từng cặp. Như vậy một nút sẽ gửi dữ liệu đến nút lân cận nó (được nối với nó). Nút lân cận sẽ chuyển tiếp dữ liệu như vậy cho đến khi dữ liệu đến đích. Kiểu nối mạng điểm–điểm có ba dạng chính là : hình sao (star), vòng (loop) và cây (tree) (hình 2.6).



Mạng hình sao (Star)



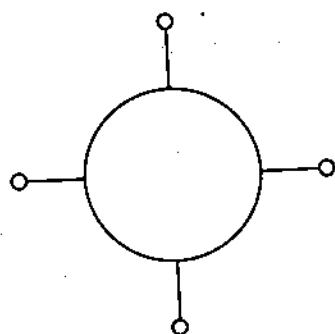
Mạng hình vòng (Loop)



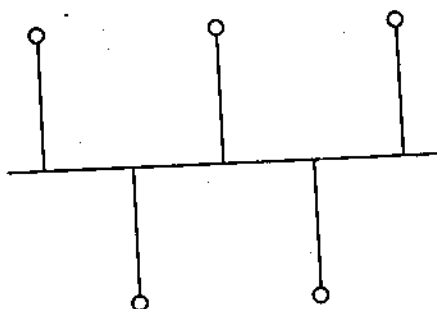
Mạng hình cây (Tree)

Hình 2.6. Một số topo mạng kiểu điểm–điểm

– Kiểu quang bá : trong kiểu này các nút nối vào đường truyền chung. Như vậy khi một nút gửi dữ liệu các nút còn lại đều nhận được. Do đó dữ liệu gửi đi cần có địa chỉ đích. Khi một nút nhận được dữ liệu nó sẽ kiểm tra địa chỉ đích xem có phải gửi cho mình không. Kiểu nối mạng quang bá có 2 dạng chính : dạng vòng tròn (ring) và xa lộ (bus) (hình 2.7).



Vòng tròn (RING)



Xa lộ (BUS)

Hình 2.7. Một số topo mạng kiểu quang bá

3.4. Các thiết bị mạng thông dụng

Các thiết bị kết nối để liên kết các máy tính và mạng với nhau. Sau đây là một số thiết bị mạng thông dụng :

- *Vì mạng* (Network Interface Card, NIC) dùng để giao tiếp giữa máy tính và đường truyền.
- *Bộ tập trung (HUB)* dùng để tập trung các dây mạng của các máy tính vào một điểm.
- *Bộ lặp (REPEATER)* dùng để tăng cường tín hiệu ở giữa các đoạn dây mạng khi dây quá dài, tín hiệu bị suy giảm.
- *Cầu (BRIDGE)* dùng để nối giữa hai mạng cùng kiểu.
- *Cổng (GATEWAY)* dùng để nối giữa hai mạng khác kiểu.
- *Bộ chọn đường (ROUTER)* dẫn đường cho các gói tin ở những điểm phân nhánh trên mạng.

– *Bộ chuyển mạch (SWITCH)* tập trung dây mạng, lọc và chọn đường cho các gói tin từ các máy tính hay mạng con. Khi một máy tính gửi một gói tin đến một máy tính, gói tin sẽ được gửi đến bộ chuyển mạch và bộ chuyển mạch căn cứ vào địa chỉ đích của gói tin để chuyển trực tiếp gói tin đó chỉ cho máy tính đích.

– *Modem* là thiết bị có chức năng kết nối một máy tính ở xa vào mạng thông qua đường điện thoại.

Trong thực tế một thiết bị có thể kết hợp nhiều chức năng, Ví dụ một thiết bị có thể có cả chức năng của HUB và REPEAT.

3.5. Phân loại mạng máy tính

Có nhiều cách phân loại mạng máy tính. Sau đây là một số cách phân loại thông dụng.

3.5.1. Phân loại theo mối quan hệ giữa các máy trong mạng

Có hai loại :

- *Mạng bình đẳng (peer-to-peer)* các máy có quan hệ ngang hàng, một máy có thể yêu cầu một máy khác phục vụ. Ví dụ như một mạng toàn windows 98.

- *Mạng khách/chủ (client/server)*. Một số máy là server (máy chủ) chuyên phục vụ các máy khác gọi là máy khách (client) hay máy trạm (workstation) khi có yêu cầu. Các dịch vụ có thể là cung cấp thông tin, tính toán hay các dịch vụ Internet. Ví dụ một mạng có thể bao gồm một máy chủ Window NT Server 4.0 và các máy trạm là Windows 98.

3.5.2. Phân loại theo quy mô địa lí

Có ba loại chính là :

- *LAN (Local Area Network)* : mạng cục bộ, ở trong phạm vi nhỏ, ví dụ bán kính khoảng 500m, số lượng máy tính không quá nhiều, mạng không quá phức tạp. Ví dụ mạng của khoa CNTT Trường Đại học Bách khoa Hà Nội.

- *WAN (Wide Area Network)* : mạng diện rộng, các máy tính có thể ở các thành phố khác nhau. Bán kính có thể 100-200 km. Ví dụ mạng của Tổng cục thuế.

Chú ý : nếu trong mạng LAN hoặc WAN có sử dụng công nghệ Internet thì chúng được gọi các mạng Intranet.

- *GAN (Global Area Network)* : mạng toàn cầu, máy tính ở nhiều nước khác nhau. Thường mạng toàn cầu là kết hợp của nhiều mạng con. Ví dụ như mạng Internet.

3.5.3. Phân loại theo hệ điều hành mạng (Network Operating System, NOS)

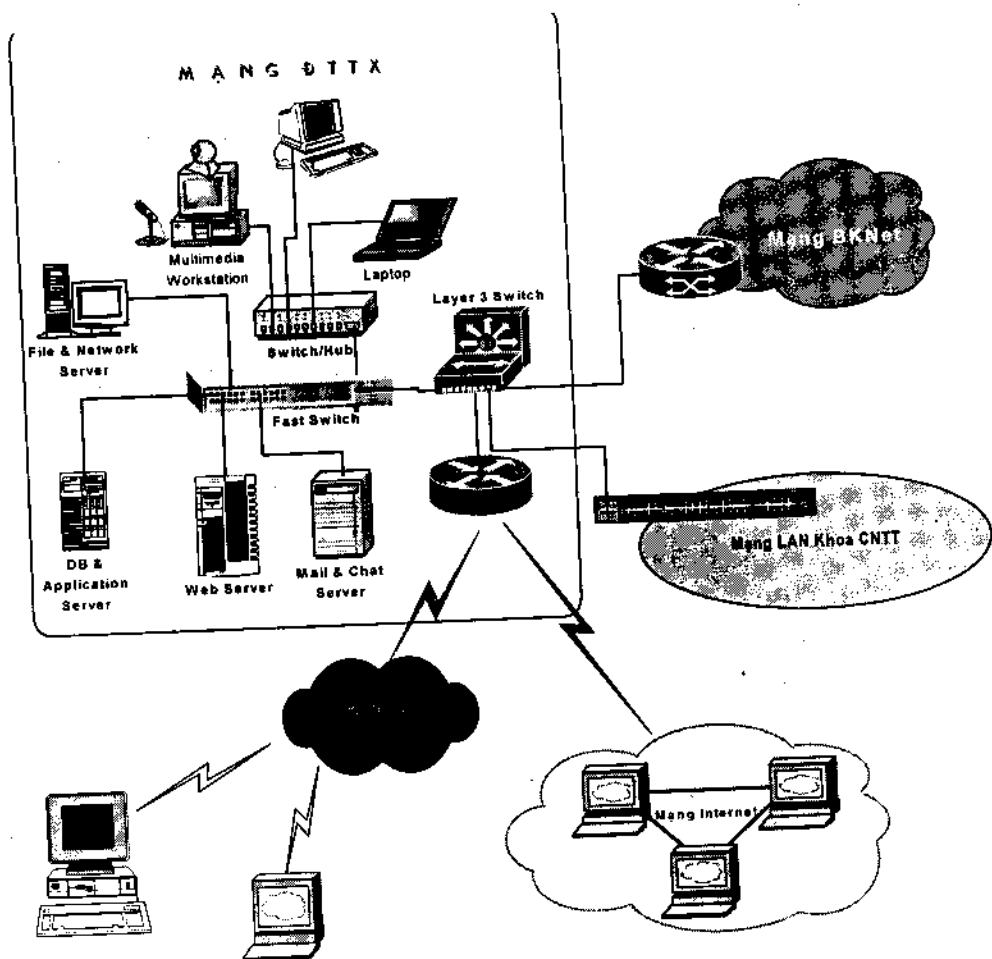
Hệ điều hành mạng : quản lí các công việc trên mạng. Thông dụng hiện nay có mạng UNIX, mạng Windows NT, mạng Netware, mạng LINUX. Chú ý trong thực tế một mạng có thể gồm nhiều mạng con khác kiểu được tích hợp với nhau thông qua các cổng (Gateway) ; ví dụ mạng Internet thực ra là một mạng gồm vô số mạng con đủ kiểu.

3.6. Internet

3.6.1. Internet là gì ?

Internet có nguồn gốc từ một dự án của Bộ quốc phòng Mỹ tên là ARPANET được thực hiện vào thập kỷ 70 của thế kỷ trước, trong thời chiến tranh lạnh dùng để đảm bảo liên lạc giữa các đơn vị quân đội. Sau đó phát triển thành mạng cho các trường đại học và viện nghiên cứu. Cuối cùng mạng có quy mô toàn cầu và trở thành mạng Internet. Dự án này đã hiện thực hóa ý tưởng liên mạng (là nguồn gốc của thuật ngữ Internet hiện nay).

Có thể nói Internet là **một mạng của các mạng máy tính** có quy mô toàn cầu (GAN), gồm rất nhiều mạng con và máy tính nối với nhau bằng nhiều loại phương tiện truyền tin (xem hình 2.8).



Hình 2.8

Điểm khác với các mạng máy tính thông thường là ở chỗ các mạng máy tính hay các máy đơn lẻ tham gia tự nguyện vào Internet và không có ai làm chủ Internet cả ; chỉ có các ủy ban điều phối và kĩ thuật giúp điều hành Internet.

3.6.2. Các tài nguyên và các dịch vụ chính của Internet

Khái niệm tài nguyên (resource) chỉ những nguồn lực tiềm tàng, sẵn sàng để khai thác. Tài nguyên trên Internet có thể là thiết bị nhưng chủ yếu là thông tin.

Qua Internet ta có thể sử dụng máy tính, máy in, thiết bị lưu trữ cách xa ta tới nửa vòng trái đất, miễn là người sử dụng được cấp quyền đó.

Các dạng thông tin trên Internet bao gồm :

- Các văn bản dạng text, poscript.
- Các ảnh.
- Các tài liệu lưu trữ dưới dạng âm thanh.
- Phim.
- ...

Ta có thể dùng Internet để thực hiện nhiều dịch vụ mạng. Các dịch vụ thông dụng nhất trên Internet hiện nay là :

1) Trao đổi đối thông tin : Có ba hình thức trao đổi thông tin giữa các thành viên trên Internet là :

- Thư điện tử (E-mail).
- Hội thoại mạng (chat).
- Điện thoại Internet, đây là một dịch vụ đặc biệt ; tiếng nói của người được số hóa và truyền trên Internet tới người đối thoại với giá rất rẻ.

2) Truyền thông tin (FTP, File Transfer Protocol).

3) Truy nhập máy tính từ xa (telnet).

4) World Wide Web (WWW) để xem (duyet) thông tin từ các kho thông tin có ở khắp mọi nơi trên trái đất⁽¹⁾ ; thông tin đưa tới máy của người sử dụng có thể là siêu văn bản (HTML), âm thanh, đồ họa, ...

....

(1) Chẳng hạn từ các Website

3.6.3. Vai trò của Internet

Trong thời đại của công nghệ thông tin hiện nay Internet có nhiều lợi ích như truyền tin, phổ biến tin, trao đổi tin một cách nhanh chóng thuận tiện rẻ tiền hơn so với các phương tiện truyền thông khác như điện thoại, fax, telex,

Internet ảnh hưởng sâu sắc đến toàn bộ thế giới về mọi mặt. Có thể nói rằng Internet đã trở thành một yếu tố quan trọng không thiếu được trong thời đại hiện nay, có mặt ở mọi nơi, mọi lúc, mọi lĩnh vực, mọi ngành.

3.6.4. Làm sao để có được các dịch vụ Internet

Để kết nối được Internet ta cần có :

- Máy tính có modem hoặc card mạng.
- Có phần mềm Internet thông dụng như Web browser để xem trang web, ví dụ IE, Netscape, ..., phần mềm Email để xem thư như Yahoo, Outlook, Messenger,....
- Có đường nối với Internet : qua mạng, qua đường điện thoại, đường thuê riêng của bưu điện. Thông thường hiện nay kết nối qua điện thoại.
- Có tài khoản Internet ở trên mạng hay ở một nhà cung cấp dịch vụ Internet (Internet Service Provider, ISP), ví dụ như VNN, FPT,....

Chi phí sử dụng Internet gồm có :

- Chi phí lắp đặt ban đầu.
- Thuê bao hàng tháng⁽¹⁾.

Bài tập

1. Chọn câu đúng nhất phát biểu về CPU. Các thành phần của CPU bao gồm :
 - a) Bus điều khiển (Control Bus), Khối điều khiển (Control Unit), Khối xử lý số học và logic (ALU), Tập các thanh ghi (Registers).
 - b) Bus bên trong CPU (Internal Bus), Khối điều khiển (Control Unit), Khối xử lý số học và logic (ALU), Tập các thanh ghi (Registers).
 - c) Bus dữ liệu (Data Bus), Khối điều khiển (Control Unit), Khối xử lý số học và logic (ALU), Tập các thanh ghi (Registers).

(1) Hoặc mua Internet card

2. Chọn câu đúng nhất về vai trò của MODEM trong các câu sau : Modem cho phép
- a) Sử dụng máy tính như máy điện thoại ;
 - b) Tăng tốc độ của máy tính ;
 - c) Kết nối máy tính vào mạng Internet qua mạng điện thoại ;
 - d) Tất cả các chức năng trên.
3. Chọn các câu đúng nhất : Các thiết bị sau thuộc về bộ nhớ ngoài :
- a) ROM, RAM ;
 - b) Đĩa cứng, đĩa mềm ;
 - c) Đĩa cứng, đĩa CDROM, ROM ;
 - d) Đĩa cứng, đĩa mềm, bộ nhớ cache.
4. Chọn các câu đúng nói về sự khác nhau của RAM và ROM.
- a) RAM là bộ nhớ trong, ROM là bộ nhớ ngoài ;
 - b) RAM không mất dữ liệu khi mất điện, ROM mất dữ liệu khi mất điện ;
 - c) RAM cho phép đọc/ghi được nhiều lần, ROM chỉ cho phép đọc ;
 - d) Tất cả đều sai.
5. Chọn câu trả lời đúng. Bộ nhớ trong so với bộ nhớ ngoài :
- a) Tốc độ nhanh hơn ;
 - b) Kích thước lớn hơn ;
 - c) Tất cả các ý trên đều sai.
6. Chọn các câu khẳng định đúng. Dữ liệu khi đưa vào bộ nhớ RAM máy vi tính :
- a) Phải được thông qua bàn phím.
 - b) Phải được lưu trên đĩa mềm.
 - c) Sẽ bị mất khi tắt máy vi tính.
 - d) Sẽ bị mất khi tắt màn hình.

Chương 3

PHẦN MỀM MÁY TÍNH

Phần cứng máy tính là vật vô tri, vô giác. Nó hoạt động được là nhờ phần mềm ; phần mềm máy tính có thể hiểu là các chương trình điều khiển mọi thao tác của máy tính. Khi máy tính đã được khởi động nó luôn luôn ở trong quá trình thực hiện các lệnh. Để có thể hiểu cặn kẽ về các thao tác của máy tính chúng ta cần nghiên cứu các lệnh, ...

Có bốn loại phần mềm chính như sau :

- Hệ điều hành (HĐH).
- Phần mềm ứng dụng.
- Phần mềm tiện ích.
- Các ngôn ngữ lập trình.

1. HỆ ĐIỀU HÀNH

Hệ điều hành được dịch ra từ thuật ngữ tiếng Anh : Operation System(OS)

Hệ điều hành điều khiển và đồng bộ việc sử dụng phần cứng của các chương trình ứng dụng phục vụ nhiều người sử dụng khác nhau với các mục đích phong phú đa dạng.

Ta có thể hiểu HĐH là hệ thống các chương trình đảm bảo các chức năng giao tiếp người, máy và quản lí tài nguyên hệ thống tính toán, hệ điều hành là phần mềm không thể thiếu đối với một máy tính.

1.1. Giới thiệu

OS là một hệ thống các chương trình :

- Giám sát quá trình thực hiện các chương trình của người dùng.
- Quản lí và phân chia các tài nguyên (bộ nhớ, file dữ liệu, thiết bị ngoại vi, vi xử lí...)
- Giúp cho việc sử dụng máy tính được thuận lợi, hiệu quả nhất.

1.2. Khái niệm về Hệ điều hành MS-DOS

- Do hãng Microsoft sản xuất.
- Các version : MS-DOS 1.0, 1.1, ..., 6.0, 6.20, 6.22⁽¹⁾.

Chức năng cơ bản :

- Tạo giao diện người/máy.
- So sánh, sao chép, đưa ra, xóa, đổi tên các tệp trên đĩa.
- Tổ chức phân chia đĩa (khởi tạo đĩa).
- Thực hiện các chương trình (SYS & USER).
- Soạn thảo một tệp.
- Đưa ra thiết bị nhận và truyền xa thay cho việc đưa ra máy in.
- Cứu tệp trên đĩa bị hỏng.
- In đồ thị trên màn hình màu ra giấy.
- In nội dung các tệp ra giấy.
- ...

1.3. Các thành phần của MS-DOS

a) ROM BIOS (*Basic Input Output System*) :

Nằm : tại ROM, chung cho mọi OS trên PC, gồm nhiều chương trình con.

Chức năng :

- Kiểm tra bộ nhớ & các thiết bị ngoại vi.
- Khởi động hệ điều hành (kiểm tra ổ đĩa đã sẵn sàng chưa, đọc "Boot record" từ DOS, chuyển điều khiển cho boot record để đọc các modul...).
- Các chương trình điều khiển thiết bị ngoại vi.

b) Boot record :

Nằm : tại sector đầu tiên của DOS, cố định trên rãnh đầu tiên.

Chức năng : Khởi động chương trình nạp 2 modul (IO.SYS & MSDOS.SYS).

c) IO.SYS :

Nằm : trên đĩa DOS được Boot record nạp vào bộ nhớ (file ẩn).

(1) Hãng IBM có các bản 6.1 và 6.3 tương ứng với 6.20 và 6.22 của MS

Chức năng :

- Bổ sung và cung cấp những mở rộng (hoặc chữa lỗi) cho ROMBIOS.
- Điều khiển đĩa cứng và thiết bị ngoại vi.
- Dễ thay đổi nội dung.

d) MSDOS.SYS

Nằm : trên đĩa DOS được Boot record nạp vào bộ nhớ (file ẩn).

Chức năng :

- Chứa các chương trình phục vụ không liên quan tới các thiết bị I/O.
- Các chức năng khác : sắp xếp các file, đóng mở file, tạo xóa file, ...

e) Command.com

Nằm : trên đĩa DOS được nạp khi cần thiết (file hiện).

Chức năng :

- Phân tích và xử lý các lệnh do người dùng đánh vào.
- Thực hiện file có đuôi BAT.
- Chứa các lệnh thường trú (internal command).

f) External command

Đó chính là các file chương trình dạng : *.COM, *.EXE.

Chức năng : thực hiện các dịch vụ khi được yêu cầu.

Từ 2.0 trở đi có *.BAT là file chứa gộp các lệnh.

1.4. Một số khái niệm cơ bản về MS-DOS

1.4.1. Nạp DOS hay khởi động máy

Để có thể yêu cầu máy thực hiện các ứng dụng (chương trình), thao tác đầu tiên ta cần thực hiện là khởi động máy. Khi khởi động máy HĐH được nạp vào bộ nhớ của máy. Sau khi nạp HĐH chúng ta có thể khai thác các khả năng của máy để phục vụ cho mọi mục đích.

Muốn khởi động máy ta phải có một **đĩa hệ thống** hay còn gọi là đĩa khởi động ; để gọi là đĩa hệ thống, trên đĩa đó cần có tối thiểu ba tệp tin **io.sys**, **msdos.sys** và **command.com** ; trong đó hai tệp tin đầu cần chiếm một vị trí đặc biệt trên đĩa.

Có hai cách thường dùng để khởi động máy :

1. Khởi động từ đĩa A ta tiến hành hai thao tác sau :

- Chèn đĩa hệ thống vào ổ A,

- Bật công tắc CPU và công tắc màn hình,

chúng ta đợi cho đến khi xuất hiện trên màn hình xâu kí tự A : \>⁽¹⁾.

2. Khởi động từ đĩa C, ta cần chắc chắn rằng trong ổ A không có đĩa, sau đó bật công tắc CPU và công tắc màn hình, quá trình khởi động kết thúc khi nhìn thấy xâu kí tự C : \>.

a) Dấu nhắc hệ thống :

d : \>

ở đây d có thể là : A, B, C,... tên ổ đĩa hiện thời⁽²⁾ (trong một thời điểm chỉ có một ổ đĩa hiện thời).

b) Khi có thông báo chuẩn hệ thống, mới có thể đánh lệnh DOS. Sau khi thực hiện xong thì lại xuất hiện thông báo chuẩn.

1.4.2. Tổ chức đĩa từ

1. Tên ổ đĩa

Hệ điều hành nhận diện ra ổ đĩa mềm hay ổ đĩa cứng thông qua tên của chúng. Ổ đĩa mềm có tên là A hay B⁽³⁾; ổ cứng được đặt tên là C, nếu có ổ cứng thứ hai ổ này sẽ có tên là D,...tên của ổ đĩa quang (nếu có) là kí tự ngay sau tên của ổ cứng cuối cùng.

Để thay đổi ổ đĩa làm việc ta chỉ cần gõ tên của ổ đĩa cần chuyển đến, dấu ':' (hai chấm) rồi gõ phím Enter⁽⁴⁾ (↵); chẳng hạn giả sử rằng chúng ta đang ở ổ C để chuyển sang ổ D ta thực hiện các thao tác sau :

C : \> d : (↵)

2. Thư mục và tệp tin (directory và file)

a) Tệp tin : Tất cả các dữ liệu và chương trình lưu lại trên đĩa đều được nhóm lại thành các tệp tin ; để phân biệt các tệp tin người ta đặt tên cho chúng. MSDOS quy định tên tệp bao gồm hai phần : tên chính và tên phụ (đuôi, kéo dài), tên chính có tối đa 8 kí tự, tên phụ có tối đa 3 kí tự ; giữa chúng có dấu '.' (chấm) :

XXXXXXXX.XXX

(1) Người ta gọi là dấu nhắc hệ thống.

(2) Còn gọi là ổ đĩa hiện hành hay ổ chủ.

(3) Trên thực tế hiện nay máy PC chỉ có một ổ đĩa mềm A.

(4) Chú ý nếu ổ đĩa cần chuyển đến là ổ mềm thì cần chắc chắn là trong ổ đã có một đĩa còn tốt.

Ta chỉ có thể dùng một số loại kí tự sau để đặt tên : chữ cái, chữ số, dấu gạch chân, dấu \$.

Ta lưu ý tới một số kí tự đặc biệt :

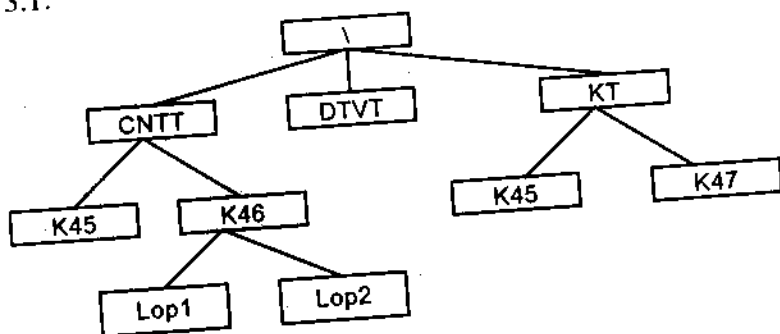
- ? thay thế một kí tự bất kì ;
- * thay thế một nhóm kí tự bất kì.

Chú ý : - Tên phụ không bắt buộc phải có.

- Tên nên có tính diễn nghĩa.
- Càng ngắn càng tốt.

b) Thư mục : Để dễ dàng quản lí các tệp tin, người ta gộp các tệp có chung một đặc tính nào đó (ví dụ của cùng một người, cùng một công việc,...) vào từng nhóm gọi là thư mục (directory) ; đến lượt chúng mỗi thư mục lại có thể chứa các thư mục khác gọi là thư mục con (subdirectory),...

Chẳng hạn trên một ổ đĩa (thí dụ C), người ta muốn lưu trữ các thông tin liên quan tới trường Đại học Bách khoa Hà Nội ; một giải pháp tối ưu sẽ là : tất cả các thông tin về khoa Công nghệ Thông tin sẽ để trong thư mục CNTT, các thông tin về khoa Điện tử _ Viễn thông sẽ chứa trong thư mục DTVT,... đến lượt chúng các thư mục CNTT, DTVT,... lại có thể chứa các thư mục khác gọi là thư mục con : K45, K46,... ; một cấu trúc lưu trữ như thế có thể miêu tả như hình 3.1.



Hình 3.1. Cây lưu trữ

Một cấu trúc lưu trữ có hình ảnh như vậy gọi là cấu trúc cây (tree).

Nhờ sử dụng thư mục nên việc quản lí các tệp trở nên dễ dàng vì khi cần tìm kiếm một tệp chỉ cần tìm nó trong thư mục chứa các tệp có cùng tính chất chứ không cần tìm trên toàn bộ đĩa từ. Rõ ràng việc này sẽ đẩy nhanh tốc độ tìm kiếm vì nói chung số lượng tệp trong một thư mục thường không nhiều lắm.

Trong tất cả các thư mục có một thư mục chính gọi là thư mục gốc ; đây là một thư mục đặc biệt có tên là " \ " ; mỗi thư mục đều có tên, việc đặt tên

ho thư mục giống như đặt tên cho tệp, ngoại trừ một việc tên thư mục không có đuôi.

Lưu ý : trong cùng một thư mục không có hai tệp, hay hai thư mục con hoặc một thư mục con và một tệp trùng tên nhau.

- Tạm thời dừng các lệnh bằng phím Pause ; tiếp tục bằng phím bất kì.
- Để dừng hẳn một lệnh ta gõ Ctrl-Break để trở về dấu nhắc hệ thống.

3. Đường dẫn

Theo định nghĩa đường dẫn là một dãy các tên thư mục, mỗi tên thư mục cách nhau bởi dấu "\" ; *Ví dụ :*

\CNTTK46\Lop1 hay \KTK47 ...

4. Cú pháp các lệnh DOS

Các lệnh dos có dạng như sau :

tên_lệnh [tham số1 [tham số2 ...]]

ở đây tên_lệnh và các tham số phải cách nhau bởi ít nhất một dấu cách, các tham số có thể chia làm hai loại :

- Bắt buộc.
- Không bắt buộc (được đặt trong cặp []).

5. Tên các thiết bị cơ bản :

DOS dành riêng một số tên làm tên thiết bị ngoại vi của hệ thống :

Tên	Thiết bị
CON	Bàn phím + màn hình
COM1	Cổng nối tiếp (truyền xa)
COM2	Cổng nối tiếp
LPT1 (PRN)	Cổng song song 1
LPT2	Cổng song song 2
LPT3	Cổng song song 3
NUL	Thiết bị rỗng

Quy ước :

- Không được đặt tên file trùng.
- Có thể được sử dụng như một tên file.
- Khi sử dụng phải đảm bảo thiết bị đã sẵn sàng.

1.5. Quá trình làm việc

- Khởi động máy hay còn gọi là nạp MS-DOS.
- Chạy các ứng dụng.
- Tắt máy.

1.6. Các lệnh thông dụng của DOS

1.6.1. Các lệnh nội trú

Lệnh nội trú là những lệnh được tải tự động vào bộ nhớ trong khi khởi động máy và có mặt thường trực ở đó.

1. Lệnh xoá màn hình CLS.

Công dụng : xoá màn hình và đưa con trỏ màn hình⁽¹⁾ về góc trên bên trái.

Cú pháp :

cls (↵)

2. Chuyển ổ đĩa hiện thời

Cú pháp :

d : (↵)

Ở đây d là tên ổ đĩa cần chuyển tới.

Ví dụ :

Trước khi thực hiện lệnh :

A :> b : (↵)

B :> b : (↵)

B :> c : (↵)

sau khi thực hiện lệnh :

B :>

B :>

C :>

3. Lệnh DIR

Công dụng : xem nội dung của một thư mục được chỉ ra ; điều này có nghĩa là lệnh này làm hiển thị lên màn hình danh sách các tệp và các thư mục con nằm trong một thư mục nào đó.

Cú pháp :

DIR [d :][path][file] [/p]/[w]⁽²⁾

(1) Con trỏ màn hình là dấu hiệu chỉ vị trí vào ra các thông tin.

(2) Nhắc lại một lần nữa cặp dấu [] chỉ ra rằng các tham số nằm trong đó có thể vắng mặt khi viết lệnh, khi đó hệ thống hoạt động theo chế độ ngầm định.

ở đây : dir là từ khoá ;

d là tên ổ đĩa ;

path là đường dẫn,

file là tên tệp cần hiển thị thông tin ; nếu không tìm thấy máy hiện thị thông báo " file not found" ;

/p tham số này đưa vào khi ta muốn xem từng trang màn hình một ;

/w được dùng khi ta chỉ muốn xem tên tệp và tên thư mục.

Ví dụ : Giả sử ổ chủ hiện thời là C (lúc đó dấu nhắc hệ thống có dạng C:\>)

dir a :\ làm hiển thị danh mục thư mục gốc của ổ a lên màn hình

dir c :\CNTT làm hiển thị danh mục thư mục CNTT của ổ C.

dir c :\config.sys làm hiển thị thông tin về tệp config.sys lên màn hình.

4. Lệnh MD

Công dụng : tạo ra một thư mục con.

Cú pháp :

md [d :][path]\tên_tm (↵).

Ví dụ :

Lệnh md c :\CNTT sẽ tạo ra thư mục CNTT trong thư mục gốc của ổ C.

Lệnh md c :\CNTTK45 sẽ tạo ra thư mục con K45.

Chú ý : trong trường hợp tại thư mục được chỉ ra đã tồn tại một tệp hoặc một thư mục con có tên trùng với tên_tm, lúc đó hệ thống sẽ báo lỗi " can't create subdirectory".

5. Lệnh CD

Công dụng : chuyển thư mục làm việc sang một thư mục khác trong cùng ổ đĩa.

Cú pháp :

cd path (↵)

Ví dụ :

Trước khi thực hiện lệnh

C :\>cd \CNTT (↵)

C :\CNTT>cd \KTK47 (↵)

Sau khi thực hiện lệnh

C :\CNTT>

C :CNTTKTK47>

Chú ý : có hai dạng đặc biệt của lệnh cd, đó là :

- 1) cd.. chuyển về thư mục mẹ của thư mục hiện hành.
- 2) cd\ chuyển về thư mục gốc của ổ đĩa hiện hành.

6. Lệnh RD

Công dụng : xoá bỏ một thư mục được chỉ ra ở cuối đường dẫn.

Cú pháp :

rd [d :][path]

Ví dụ :

rd c:\CNTTK45 (↵) sẽ loại bỏ thư mục con K45.

Chú ý : muốn xoá được một thư mục cần phải có hai điều kiện :

- Thư mục cần xoá phải rỗng.
- Thư mục đó không phải là thư mục hiện hành hoặc chứa thư mục hiện hành ; chẳng hạn lệnh sau sẽ bị báo lỗi :

C:\CNTTK46>rd K46 (↵) hoặc :

C:\CNTTK46>rd C:\CNTT (↵)

Sau đây là một số lệnh liên quan tới file :

7. Lệnh COPY

Công dụng : sao chép một hay nhiều tệp tin từ nơi này qua nơi khác

Cú pháp :

COPY [d1 :][path1]file1 [d2 :][path2][file2].

ở đây : d1 là tên ổ đĩa nơi chứa tệp cần sao di(mặc định ổ đĩa hiện thời) ;

path1 chỉ đường dẫn tới tệp cần sao ;

file1 là tên tệp cần sao di ; nếu không tìm thấy hệ thống sẽ báo lỗi ;

file2 là tên file đích (mặc định file1) ;

d2 là tên ổ đĩa sẽ chứa file đích (mặc định ổ đĩa hiện thời).

Ví dụ :

Lệnh :

copy c:\tp55\turbo.exe a:\tp55 (↵)

sẽ sao tệp turbo.exe từ thư mục tp55 ở ổ C sang thư mục tp55 của ổ A.

Để sao nhiều tệp tin một lần, người ta sử dụng các kí tự đại diện * và ? ;

ví dụ lệnh :

copy c:\tp55\turbo.* a:\tp55 (↵)

sẽ sao các tệp có tên chính là turbo tên phụ là gì cũng được từ thư mục tp55 ở ổ C sang thư mục tp55 của ổ A ; chẳng hạn các tệp turbo.exe, turbo.tpl, turbo.tp sẽ được sao đi. Còn lệnh :

copy c : \tp55\td?.pas a : \ (↵)

sẽ sao các tệp có tên phụ là pas, tên chính có 3 ký tự, trong đó hai ký tự đầu là t, d còn ký tự thứ ba là gì cũng được ; chẳng hạn các tệp có dạng sau đây sẽ được sao đi : td1.pas, td2.pas, tdc.pas, td.pas,...

Chú ý : Người ta có thể dùng lệnh copy để tạo ra một tệp văn bản qua bàn phím bằng các thao tác như sau :

- Từ dấu nhắc hệ thống gõ lệnh : copy con [d :][path]tên_tệp (↵) ;
- Gõ nội dung tệp (bắt buộc phải gõ nội dung này ; dùng phím enter khi xuống dòng) ;
- Gõ f6 (hay ctrl_z).

8. Lệnh DEL

Công dụng : xoá một hay nhiều tệp tin.

Cú pháp :

DEL [d :][path]file

ở đây :

file là tên tệp cần xoá (nếu không có sẽ báo lỗi) ;

d là tên ổ đĩa chứa tệp (mặc định ổ đĩa hiện thời).

path là đường dẫn tới nơi chứa tệp.

Ví dụ :

Lệnh :

del c : \tp55\thi_du.pas (↵)

sẽ xoá tệp tin thi_du.pas trong thư mục tp55 của ổ C.

Chú ý : để xoá nhiều tệp tin ta sử dụng các ký tự đại diện, giống như trong trường hợp lệnh copy.

9. Lệnh TYPE

Công dụng : hiển thị nội dung của tệp văn bản lên màn hình.

Cú pháp :

TYPE [d :][path]file

ở đây : file là tên tệp văn bản cần xem nội dung (nếu không có sẽ báo lỗi) ;

d là tên ổ đĩa chứa file (mặc định ổ đĩa hiện thời).

Ví dụ :

type c:\config.sys

Sẽ hiện thị nội dung của tệp config.sys lên màn hình.

Chú ý : Có thể dùng lệnh này để in

Cú pháp :

TYPE [d :][path]\file > PRN

10. Lệnh Date

Công dụng : cho xem và nếu cần thì điều chỉnh ngày tháng năm của hệ thống.

Cú pháp :

date (↵)

khi thực hiện lệnh ta thu được thông báo sau :

current date is mm-dd-yy

Enter new date (mm-dd-yy) :

Lúc này ta có thể gõ ngày tháng mới hay ấn Enter để chấp nhận các giá trị hiện thời của hệ thống.

11. Lệnh time

Công dụng : cho xem và nếu cần thì điều chỉnh lại các giá trị thời gian của hệ thống.

Cú pháp :

time (↵)

khi thực hiện lệnh ta thu được thông báo sau :

current time is hh:mm:ss.xx

Enter new time (hh:mm:ss) :

Lúc này ta có thể gõ giá trị mới của thời gian hay ấn Enter để chấp nhận các giá trị hiện thời của hệ thống.

1.6.2. Các lệnh ngoại trú

Lệnh ngoại trú là những lệnh không nằm thường trực ở bộ nhớ trong mà nằm trên đĩa dưới dạng những tệp tin.

1. Lệnh định dạng đĩa.

Công dụng : tạo (hoặc tạo lại) khuôn dạng cho đĩa.

Cú pháp :

[d1 :][path]FORMAT [d2 :][/s] (-)

ở đây : d1 là tên ổ chứa tệp format.com ;

path là đường dẫn tới nơi chứa tệp format.com ;

d2 là tên ổ đĩa chứa đĩa muốn format⁽¹⁾ ;

/s nếu ta muốn tạo đĩa hệ thống.

Trước khi format máy ra thông báo đề nghị người dùng cho đĩa cần format vào ổ đĩa.

2. Lệnh SYS.

Công dụng : tạo ra đĩa hệ thống.

Cú pháp :

[d1 :][path]sys d : (-)

ở đây : d là tên ổ đĩa nơi chứa đĩa mà ta muốn trở thành đĩa hệ thống.

3. Lệnh ATTRIB.

Công dụng : đặt thuộc tính cho tệp

Để nắm được ý nghĩa của lệnh này ta đề cập đôi chút về khái niệm thuộc tính của một tệp tin ; một tệp có thể có 4 thuộc tính sau đây :

archive (A) (lưu trữ) : thuộc tính này cho phép các tệp có thể lưu trữ, cập nhật, xoá, thay đổi nội dung và tên gọi bất cứ lúc nào.

read only (R) (chỉ đọc) : các tệp mang thuộc tính này chỉ cho phép người sử dụng đọc nội dung hoặc thực hiện chương trình ; không cho phép xoá hay thay đổi nội dung của tệp này, nhưng cho phép sao hoặc đổi tên.

Hidden (H) (ẩn) : các tệp có thuộc tính này bị làm "ẩn" và do đó sẽ không có tác dụng đối với các lệnh dir, del, copy, ren.

System (S) (hệ thống) : các tệp có thuộc tính này thường là các tệp hệ thống, đồng thời chúng có thêm luôn các thuộc tính Hidden và Read only.

Các tệp trên đĩa khi được tạo ra đều có thuộc tính mặc định là archive.

Cú pháp :

[d:][path]Attrib [+a/-a] [+r/-r] [+h/-h] [+s/-s] file

ở đây các thuộc tính được gán hoặc bỏ bằng kí hiệu + hoặc - ; thí dụ lệnh :

Attrib +r nc.exe

Lệnh này gán thuộc tính chỉ đọc cho tệp nc.exe.

(1) Thường là ổ A

4. Lệnh DISKCOPY :

Công dụng : tạo ra một bản sao giống hệt như đĩa nguồn.

Cú pháp :

[d1 :][path]\diskcopy d2 : d3 :⁽¹⁾ (↵)

ở đây : d2 là ổ đĩa chứa đĩa nguồn

d3 là ổ đĩa chứa đĩa đích

5. Lệnh PATH

Như đã trình bày, các tệp chương trình có thể thực hiện trong DOS là các tệp có đuôi EXE hay COM. Các tệp này hoặc là các thành phần của DOS hoặc là các ứng dụng viết trong môi trường DOS nhằm giải quyết một bài toán nào đó. Để thực hiện chúng trước tiên ta phải chuyển tới thư mục chứa chúng rồi mới gõ lệnh để thực hiện chúng ; tuy nhiên việc sử dụng lệnh cd nhiều khi gây ra những phiền toái nhất định. Lệnh PATH cho phép máy tự động tìm kiếm vị trí của các tệp chương trình này và thực hiện chúng mà không nhất thiết phải định vị tại thư mục chứa chúng.

Cú pháp :

Path path₁ ;path₂ ;... ;path_n

Sau khi thực hiện lệnh PATH danh sách các đường dẫn được chỉ ra trong lệnh sẽ được HDH "ghi nhớ" lại ; Khi ta phát lệnh thực hiện một chương trình nào đó, DOS sẽ tìm tệp đó trước tiên trong thư mục ngầm định, nếu không tìm thấy HDH sẽ tìm tệp tại các đường dẫn đã được ghi nhớ trong lệnh PATH.

1.7. Các tệp hệ thống

1.7.1. Các tệp hệ thống chính yếu

Để trở thành đĩa hệ thống (đĩa khởi động) trên đĩa phải có ít nhất 3 tệp sau :

1. COMMAND.COM

1. IO.SYS

2. MSDOS.SYS

trong đó : hai tệp sau luôn có thuộc tính ẩn, lệnh dir dạng thông thường không làm hiện chúng.

(1) Trên thực tế hiện nay người ta chỉ sử dụng dạng : [d1 :][path]\diskcopy a : a :

Chú ý : – Không thể dùng lệnh copy để sao chép 3 tệp hệ thống nói trên vào một đĩa mới để tạo ra một đĩa hệ thống.

– Ngoài các tệp tối thiểu, bắt buộc trên, các tệp sau đây cũng có những ý nghĩa quan trọng : config.sys, autoexec.bat, himem.sys,...

1.7.2. Tệp CONFIG.SYS

Tệp config.sys là tệp văn bản chứa các thông số chính của hệ thống. Các thông số thường gặp là :

Files = 99

ở đây 99 là số các tệp được mở đồng thời, nên đặt trong khoảng từ 20 đến 50.

Buffers = 99

đặt bộ nhớ đệm bàn phím ; nên đặt từ 10 tới 40.

Device = device_name

đặt thông số cho các thiết bị ngoại vi hoặc bộ nhớ mở rộng ; device_name thường là các tệp. Thí dụ :

Device = himem.sys

đặt trình xử lý bộ nhớ mở rộng.

Device = ramdrive.sys 99

Thiết lập ổ đĩa ảo.

...

Tệp config.sys phải đặt ở thư mục gốc của đĩa khởi động.

1.7.3. Tệp Autoexec.bat

Tệp này là tệp văn bản chứa các lệnh DOS. Các lệnh này sẽ được thực hiện ngay khi khởi động máy ; tệp autoexec.bat cũng được đặt ở thư mục gốc.

2. PHẦN MỀM ỨNG DỤNG

Phần mềm ứng dụng là các chương trình phục vụ cho các ứng dụng cụ thể. Chúng ta có thể liệt kê một loại như sau :

– Soạn thảo văn bản như : WORD, BKED, VietRes, VNI, ...

Đây là các chương trình cho phép biên soạn văn bản bao gồm nhiều thao tác thuận tiện : nhập văn bản, xoá văn bản, căn lề, trình bày các kiểu chữ, ...

– Bảng tính điện tử như : EXCEL, LOTUS, ...

Bảng tính điện tử giúp ta thực hiện các thao tác dựa trên việc trình bày dữ liệu theo ô, mỗi ô có thể diễn số hay văn bản, ...

- Thư tín điện tử (email) như : yahoo, Outlook, ...

Các chương trình này giúp chúng ta nhận và gửi thư với khoảng cách nửa vòng trái đất chỉ trong vài giây, ...

- ...

3. PHẦN MỀM TIỆN ÍCH

Là các chương trình nhỏ hỗ trợ thêm cho hệ điều hành bằng cách cung cấp một số dịch vụ mà hệ điều hành chưa có hoặc chưa thực hiện tốt ; thí dụ tối ưu hoá đĩa cứng, khôi phục các thông tin bị xoá hay bị lỗi, tạo dạng đĩa, ...Hiện nay trên thị trường có một số loại như : NU, NC, PCTool, ...

4. CÁC NGÔN NGỮ LẬP TRÌNH

Ngôn ngữ lập trình là các chương trình giúp người sử dụng lập ra các chương trình của chính họ ; đối với ba loại phần mềm đã nói ở trên người sử dụng chỉ có thể thực hiện các thao tác sẵn có để khai thác một cách thụ động, trong khi đó với ngôn ngữ lập trình họ có thể sáng tác ra các phần mềm riêng cho mình,

Ngôn ngữ lập trình gồm ngôn ngữ máy, hợp ngữ và các ngôn ngữ cấp cao như : PASCAL, C, C⁺⁺, JAVA, ...Trong giáo trình này chúng tôi sẽ giới thiệu ngôn ngữ lập trình TURBO PASCAL.

Bài tập

1. Chọn các câu phát biểu đúng về vai trò của hệ điều hành
 - a) Quản lí các thiết bị ngoại vi, theo dõi và cấp phát bộ nhớ trong của máy tính.
 - b) Giao diện với người sử dụng.
 - c) Thực hiện Test các phần cứng và RAM khi bật máy tính.
 - d) Tất cả (a), (b), (c) đều sai.
2. Lệnh nào trong các lệnh MSDOS sau cho phép sao chép các tệp VD1.DAT, VD2.DAT có trong thư mục C:\VIDU vào thư mục C:\PASCAL nếu thư mục hiện thời là C:\PASCAL
 - a) COPY..\VIDU\VIDU*.DAT ;
 - b) COPY VIDU\VIDU*.DAT ;
 - c) Copy C:\VIDU\VD*.DAT ;
 - d) Không lệnh nào đúng cả.

3. Chọn câu trả lời đúng. Một đĩa khởi động của hệ điều hành MS-DOS phải chứa tối thiểu những tệp :
 - a) MSDOS.SYS, IO.SYS, COMMAND.COM ;
 - b) MSDOS.SYS, IO.SYS, COMMAND.COM, CONFIG.SYS, AUTOEXEC.BAT ;
 - c) MSDOS.SYS, IO.SYS ;
 - d) Cả a, b, c đều sai.
4. Những lệnh sau, lệnh nào là lệnh MSDOS đúng :
 - a) copy A : *.* C : ;
 - b) mkdir *.* ;
 - c) cd *.*.
5. Các chức năng không phải của Hệ điều hành (DOS) là :
 - a) Đọc ghi ổ đĩa ;
 - b) Đọc, ghi bộ nhớ trong ;
 - c) Giải PT bậc hai. ;
 - d) Dịch chương trình PASCAL.
6. Những tên tệp nào sau đây là hợp lệ trong hệ điều hành MS-DOS :
 - a) HOP\DONG.VNS ;
 - b) 2HOPDONG ;
 - c) HOP+DONG.VBS ;
 - d) ALX.
7. Chọn các câu phát biểu đúng :
 - a) Máy tính dùng để giải quyết các bài toán mà con người không làm được.
 - b) Chương trình dịch là một chương trình dùng để dịch ngôn ngữ cấp cao ra ngôn ngữ máy.
 - c) Bộ nhớ ngoài bao gồm ROM và CD-ROM.
8. Chọn ra các phát biểu đúng sau đây :
 - a) UNIX, DOS là hệ điều hành ;
 - b) Multimedia là hệ điều hành ;
 - c) Internet là hệ điều hành ;
 - d) NC là hệ điều hành.
9. Chọn ra các phát biểu đúng sau đây :
 - a) PASCAL, C, PROLOG là các ngôn ngữ cấp cao.
 - b) NC là một ngôn ngữ cấp cao.
 - c) BKED và BKAV là những ngôn ngữ cấp cao.

Chương 4

SOẠN THẢO VĂN BẢN

Để phục vụ cho công việc lập trình, chúng tôi giới thiệu một trình soạn thảo văn bản của TURBO PASCAL. Nếu các bạn nắm vững các thao tác của hệ soạn thảo này các bạn chỉ cần "để tâm" hơn một chút nữa là có thể làm chủ bất kì hệ soạn thảo văn bản nào khác.

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 :

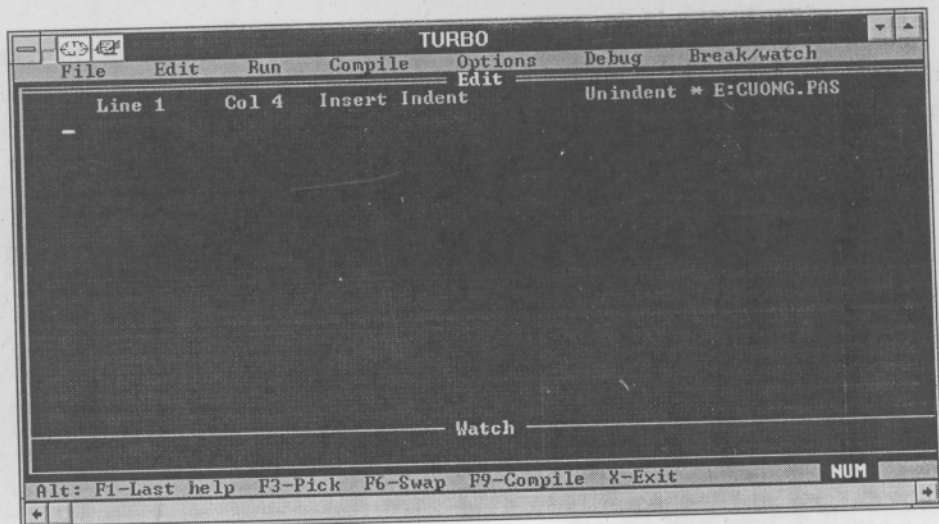
1. KHỞI ĐỘNG TRÌNH SOẠN THẢO

Giả sử chúng ta có trên đĩa tối thiểu 2 tệp :

TURBO.EXE

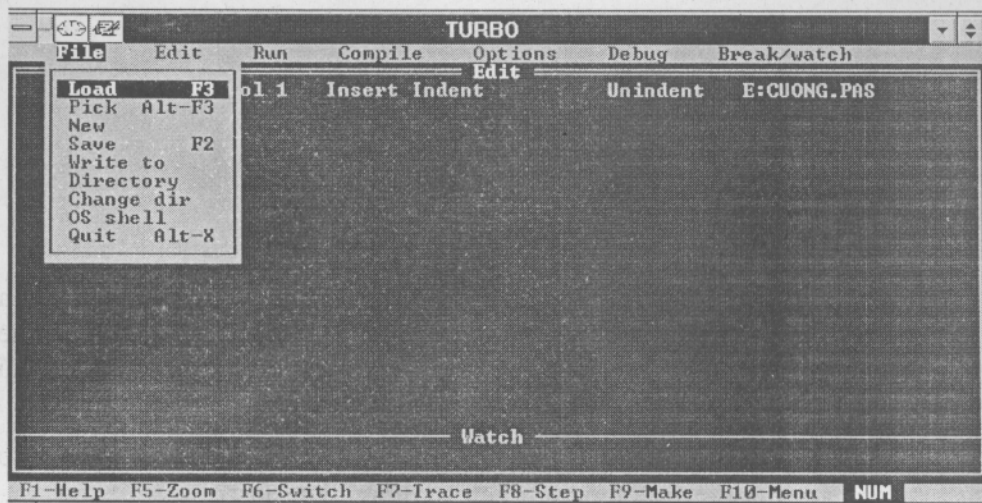
TURBO.TPL.

Lúc đó ta có thể bắt đầu làm việc với trình soạn thảo TURBO PASCAL 5.5. Sau khi khởi động, ta thu được màn hình như hình 4.1 :



Hình 4.1

Dòng trên cùng là tên các chức năng của thực đơn⁽¹⁾ chính. Mỗi chức năng có thể là một lệnh hay lại là một thực đơn có nhiều chức năng tiếp theo khác.



Hình 4.2

Trên màn hình ta thấy 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 kí tự đại diện bởi vì nếu ta gõ một trong các kí tự đại diện đó thì :

- Hoặc một lệnh được thực hiện (chẳng hạn gõ E kích hoạt lệnh Edit).
- Hoặc một thực đơn khác sẽ hiện ra (xem hình 4.2), 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 là 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 Unindent 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.

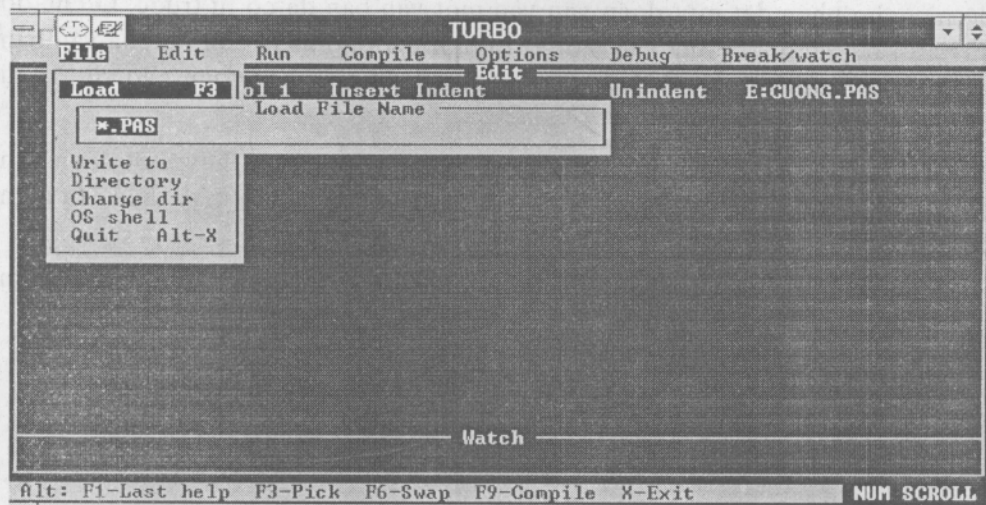
(1) Còn gọi là Menu

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 phím **E**, con trỏ màn hình sẽ xuất hiện ở cửa sổ soạn thảo, lúc này ta có thể soạn thảo văn bản của chương trình vào máy thông qua bàn phím giống như gõ máy chữ.

2. Cách thứ hai ta gõ **F**, trên màn hình xuất hiện thực đơn con (xem hình 4.3) ; 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 soạn thảo văn bản mới vào máy thông qua bàn phím. Trong thực đơn File ta còn thấy một số lệnh hay dùng sau (xem hình 4.2) :



Hình 4.3

- *New* được dùng khi ta muốn soạn thảo một tệp mới.
 - *Save* ghi tệp đang trong cửa sổ soạn thảo lên đĩa.
 - *Os shell* được sử dụng khi ta muốn tạm thời quay về DOS mà không thoát khỏi PASCAL ; muốn trở lại ta chỉ cần gõ *exit*.
 - *Quit* được sử dụng khi ta muốn thoát khỏi TURBO PASCAL.
- Các lệnh biên tập hay được dùng có thể chia 2 thành nhóm :

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

3. 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 kí 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 kí 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 kí tự đưa vào sẽ thay thế kí 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 gõ tổ hợp *ctrl-V*).

1. Một số phím và tổ hợp phím dùng để xoá

- **BackSpace** để xoá một kí tự bên trái con trỏ.
- **Del** để xoá kí 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.

2. 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.

Phần hai. LẬP TRÌNH BẰNG NGÔN NGỮ PASCAL

Chương 5

MỞ ĐẦU

PASCAL là ngôn ngữ lập trình cấp cao do giáo sư Niklaus Wirth thiết kế vào năm 1970,

Trước khi PASCAL ra đời, FORTRAN thường được dùng để giảng dạy và lập trình cho các bài toán khoa học kỹ thuật; nhưng ngôn ngữ này không có cấu trúc nên khi lập trình thường hay mắc nhiều lỗi, mất nhiều thời gian để phát hiện và sửa lỗi.

Mục đích ban đầu của tác giả là nhằm tạo ra một ngôn ngữ nhằm thay thế cho FORTRAN trong giảng dạy, quá trình sử dụng PASCAL trong giảng dạy cũng như trong công việc lập trình ứng dụng người ta đã dần dần hoàn thiện, bổ sung để PASCAL ngày càng đáp ứng được nhiều ứng dụng khoa học quản lý,... chính vì vậy mà có rất nhiều phiên bản của PASCAL xuất hiện.

TURBO PASCAL (TP) là một trong những phiên bản nổi tiếng nhất do hãng BORLAND đề xuất, nó không chỉ được những người "nhập môn" ưa thích mà còn là một công cụ mạnh cho những nhà lập trình chuyên nghiệp. Bản thân TURBO PASCAL cũng trải qua nhiều thế hệ, trong thời điểm hiện nay các thế hệ TURBO PASCAL 5.5, TURBO PASCAL 6.0 và TURBO PASCAL 7.0 là những phiên bản hay được sử dụng nhất.

1. CÁC THÀNH PHẦN CƠ BẢN

1.1. Các kí hiệu cơ bản

Cũng giống như mọi ngôn ngữ khác TURBO PASCAL có bộ chữ cái riêng cho mình chia làm 2 nhóm như sau :

- Các kí tự chữ và số :
 - + 26 chữ cái a, b, c, ..., z⁽¹⁾.
 - + 10 chữ số thập phân 0, 1, 2, ..., 9.
- Các kí tự đặc biệt :
 - + Kí tự trống
 - + Dấu bằng =
 - + Dấu cộng và dấu trừ +, -
 - + Dấu nhân và dấu chia *, /
 - + Dấu lớn hơn và dấu nhỏ hơn >, <
 - + Dấu lớn hơn hay bằng và dấu nhỏ hơn hay bằng >=, <=
 - + Dấu đô la \$
 - + ...

Đây là tập hợp các kí tự hợp lệ được dùng để viết các chương trình, không được dùng các kí hiệu khác ngoài các kí hiệu nói trên chẳng hạn như : ∂ , Σ , ω , ψ , β , α , φ , π , ...

Các kí 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.

1.2. Các từ khoá

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 thông dụng :

(1) TURBO PASCAL không phân biệt chữ hoa và chữ thường

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

1.3. Tên

Tên theo định nghĩa là một dãy kí 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 kí 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 kí tự, nếu ta đặt một tên với độ dài lớn hơn 63 kí tự thì chỉ có 63 kí tự đầu tiên có nghĩa.

TURBO PASCAL không phân biệt chữ hoa với chữ thường, ví dụ 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.

1.4. 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	abs
exp	false	boolean
char	odd	copy

...

1.5. Các dòng chương trình

Độ dài tối đa của một dòng chương trình là 127 kí tự.

2. 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 ứng 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 này :

2.1. Kiểu integer

Còn gọi là kiểu nguyên, trong TURBO PASCAL một biến kiểu integer có thể lấy các giá trị nguyên nằm trong đoạn $[-32768, 32767]$. Mỗi giá trị integer chiếm 2 byte 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 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 quá 32767.

Vì kiểu integer chỉ biểu diễn được các số nguyên khá hạn hẹp nên từ TURBO PASCAL 4.0 trở đi, người ta định nghĩa thêm các kiểu nguyên khác :

longInt	4 byte	$[-2\ 147\ 483\ 648, 2\ 147\ 483\ 647]$
word	2 byte	$[0, 65535]$
byte	1 byte	$[0, 255]$
shortInt	1 byte	$[-128, 127]$

2.2. Dữ liệu kiểu real

Một biến kiểu real có thể lấy các giá trị thực nằm trong khoảng $[2.9 \times 10^{-39}, 1.7 \times 10^{38}]$, 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.

2.3. 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.

2.4. Kiểu char

Còn gọi là kiểu kí tự, bao gồm 256 kí tự trong bảng mã ASCII. Một kí tự được viết trong hai dấu nháy đơn ; Ví dụ : 'I', 'a', 'H',.... Một giá trị kiểu char chiếm 1 byte bộ nhớ. 128 kí tự đầu tiên (có mã ASCII từ 0 đến 127) gọi là kí tự chuẩn, 128 kí tự còn lại (có mã ASCII từ 128 đến 255), được sử dụng để mã hoá các kí tự riêng của từng ngôn ngữ, gọi là các kí tự mở rộng.

Trong các ngôn ngữ lập trình người ta thường sử dụng 128 kí tự chuẩn (còn gọi là mã ASCII chuẩn), chúng được phân chia như sau :

- Các kí tự từ 0 đến 31 là các kí tự điều khiển, không in ra màn hình hoặc máy in được. Chúng được dùng để điều khiển các thiết bị ngoại vi, các thủ tục trao đổi thông tin ; Ví dụ :

- + Số 7 phát ra tiếng bip (BELL)

- + Số 27 thoát khỏi (ESC)

- + ...

- Các kí tự từ 32 đến 126 là các chữ, số, kí hiệu ... ví dụ kí tự '0' có mã 50, 'A' có mã 65, 'a' có mã 97,...

- Kí tự có mã 127 lại là kí tự điều khiển có tác dụng xoá.

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

Các kí tự có thể so sánh với nhau. Sự so sánh được tiến hành theo giá trị mã ASCII của chúng, chẳng hạn :

'A' < 'B' vì 65 < 66.

Trong TURBO PASCAL có một số hàm :

- ord() : cho ta giá trị tương ứng của kí tự trong bộ mã ASCII, chẳng hạn ord('A') = 65.

- chr(i) : hàm này cho ta kí tự tương ứng với vị trí i trong bộ mã ASCII, chẳng hạn chr(65) = 'A', trong TURBO PASCAL còn một cách biểu diễn tương đương là dùng dấu # (dấu thăng) trước số i cũng cho kết quả tương tự, như vậy :

chr(i) là tương đương với #i

- Hàm pred(ch) cho kết quả là kí tự đứng trước kí tự ch, ví dụ pred('C')='B'

- Hàm succ(ch) cho kết quả là kí tự đứng sau kí tự ch, ví dụ succ('C')='D'

- ...

3. HẲNG

Hằng là đại lượng mà giá trị của nó không thay đổi trong thời gian thực hiện chương trình. Trong ngôn ngữ C có các loại hằng :

Hằng nguyên :

Có giá trị từ -32768 đến 32767. Có thể biểu diễn một hằng nguyên dưới dạng thập phân, bát phân (cơ số 8) hay thập lục phân (cơ số 16).

Hằng thực :

Có hai cách viết giá trị của một hằng thực :

- Trong cách viết thông thường (dấu phẩy tĩnh), hằng được viết giống như trong thực tế nhưng thay dấu phẩy thập phân bằng dấu chấm.

Ví dụ : 359.12, -415.16.

- Trong cách viết theo kí pháp khoa học (dấu phẩy động) hằng được viết gồm hai phần : phần định trị mmmmm.nnn và phần mũ ;

Ví dụ : 12.3E+3 giá trị của hằng này bằng :

$$12.3 * 10^3 = 12300.0 ;$$

phần định trị là số thực, phần mũ là số nguyên. Bởi vì chúng ta có thể tăng giảm giá trị của phần mũ và tương ứng dịch chuyển dấu chấm thập phân trong phần định trị nên số thực được viết dưới dạng này còn có tên gọi là số thực dấu phẩy động, còn cách viết trên tương ứng với tên gọi số thực dấu phẩy tĩnh.

Hằng kí tự :

Là một kí hiệu trong bảng mã ASCII được đặt giữa hai dấu nháy đơn (').

Ví dụ :

Hằng 'A'

Hằng chuỗi kí tự :

Là một dãy các kí tự được đặt trong cặp dấu nháy đơn (').

Ví dụ :

'SEAGAMES 22'

4. BIỂU THỨC

Biểu thức bao gồm các toán hạng (biến số, hằng số, lời gọi hàm) kết hợp với nhau bằng các toán tử để biểu diễn một công thức nào đó.

Các toán tử trong TURBO PASCAL được chia thành 5 lớp có thứ tự ưu tiên thực hiện các phép toán trong biểu thức 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	chia 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. Lưu ý : 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 :

Toán tử**Phép toán**

=	so sánh bằng
>	so sánh lớn hơn
<	so sánh nhỏ hơn
>=	so sánh lớn hơn hay bằng
<=	so sánh nhỏ hơn hay bằng
<>	so sánh khác

6. Các hàm toán học thông dụng

PASCAL cài đặt sẵn một số hàm, sau đây là một số hàm hay dùng :

TOÁN HỌC**TURBO PASCAL**

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

5. CẤU TRÚC CỦA MỘT CHƯƠNG TRÌNH PASCAL

Một chương trình PASCAL ở dạng đầy đủ nhất bao gồm 3 phần :

1. Phần tiêu đề của chương trình.
2. Phần khai báo dữ liệu.
3. Phần thân chương trình chứa các lệnh mà máy phải thực hiện.

Để minh họa chúng ta xét một ví dụ đơn giản sau :

Ví dụ 5.1 :

```

Program tinh_dt ;
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 một chương trình.

5.1. Phần tiêu đề

Phần này kể từ bản 4.0 không nhất thiết phải có, nhưng nó chứa tên chương trình, thông thường tên được đặt sao cho có thể gọi nhớ đến nội dung của chương trình, do đó phần này giúp ta phân biệt được các chương trình khác nhau. Phần tiêu đề bắt đầu bằng từ khoá **program**, tiếp theo là tên chương trình. Trong ví dụ phần tiêu đề là *tinh - dt* ;

5.2. 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 chương trình con, các hằng, các unit cần 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 khai báo có thể gồm 7 loại khai báo, mỗi loại bắt đầu bởi một từ khoá :

- *Uses* : khai báo các unit ;
- *Const* : khai báo các hằng ;
- *Var* : khai báo các biến ;
- *Type* : mô tả kiểu dữ liệu ;
- *Label* : Khai báo các nhãn của các lệnh trong chương trình. Hiện nay với kỹ thuật lập trình cấu trúc, mục này là không cần thiết ;
- *Procedure* : khai báo các thủ tục của người sử dụng ;
- *Function* : khai báo các hàm của người sử dụng.

Ta lần lượt xét tất cả các mục này :

1. *Phân khai báo Unit* đây là một nét mới tạo sức mạnh của TURBO PASCAL từ thế hệ thứ 4 trở đi, nó thể hiện ở chỗ cho chúng ta chia chương trình thành nhiều mô đun (1 mô đun hay còn gọi là 1 Unit được hiểu 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 mô đun, 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.

Phân khai báo unit nếu có thì phải là mục đầu tiên trong phân khai báo và bắt đầu bằng từ khoá *uses* tiếp theo là danh sách các unit cần khai báo, Ví dụ :

uses graph, crt, dos ;

Có hai loại unit :

- 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 hệ điều hành 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 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 các hàm và thủ tục đồ hoạ, muốn vậy ta khai báo như sau :

uses graph ;

2. *Phân 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.

3. *Phần khai báo biến* : biến là một đại lượng có thể thay đổi giá trị giống như khái niệm biến của 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 ta đặt, còn các *ki* là các kiểu dữ liệu có sẵn hoặc là các kiểu dữ liệu vừa được định nghĩa trước đó trong mục **type**.

Chú ý : nếu có nhiều biến cùng kiểu, ta có thể khai báo chung, chẳng hạn :

```
var  
    r, s : real ;
```

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(etc) mà chúng ta sẽ nói kĩ ở phần sau.

Như vậy phần khai báo bao gồm nhiều mục, tuy nhiên tùy 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. Chẳng hạn chương trình sau không có bất cứ khai báo nào, đồng thời cũng không cần tới tiêu đề :

```
Begin  
    Writeln('Trong dam gi dep bang sen,') ;  
    Writeln('La xanh bong trang lai chen nhi vang,') ;  
    Writeln('Nhi vang bong trang la xanh') ;  
    Writeln('Gan bun ma chang hoi tanh mui bun') ;  
    Readln  
  
End.
```

5.3. Thân chương trình

Phần này nằm trọn trong cặp từ khoá **begin...end** đầu tiên, dùng dấu ";" để ngăn cách 2 câu lệnh Pascal, dấu "." sau **end** để báo điểm kết thúc chương trình.

Bài tập

1. Hãy tìm sai lầm trong việc đặt tên các biến sau :

Ky so	\$coso	6so
do	repeat	βk

2. Đưa chương trình sau vào máy :

```
program tho_con_coc ;  
begin  
    writeln ('Con coc trong hang') ;  
    writeln ('Con coc nhảy ra') ;  
    writeln ('Con coc ngồi đây') ;  
    write ('Con coc nhảy đi') ;  
    readln ;  
end.
```

rồi thực hiện các công việc sau :

- Lưu chương trình trên vào tệp với tên là 'THO.PAS'
- Ấn tổ hợp ctrl_F9 để chạy chương trình trên.
- Chủ động tạo ra lỗi, sau đó ấn tổ hợp alt_F9 để quan sát thông báo lỗi.

3. Gõ chương trình sau đây vào máy :

```
Program So_hoc ;  
Uses crt ;  
Var  
    so1, so2, tong, hieu, tich :Integer ;  
    thuong :Real ;  
Begin
```

```

Clrscr ;
Write ('-Nhap so thu nhat = ');
Readln (so1) ;
Write ('-Nhap so thu hai = ');
Readln (so2) ;
tong := so1 + so2 ;
hieu := so1 - so2 ;
tich := so1 * so2 ;
thuong := so1 / so2 ;
Writeln ('*Tong của hai số', so1, 'và', so2, '=', tong) ;
Writeln ('*Hieu của hai số', so1, 'và', so2, '=', hieu) ;
Writeln ('*Tich của hai số', so1, 'và', so2, '=', tich) ;
Writeln ('*Thuong của hai số', so1, 'và', so2, '=', thuong : 6 : 2) ;
Readln

```

End.

Thực hiện các công việc sau :

- Lưu chương trình trên vào tệp 's_hoc.pas' ;
- Chạy chương trình trên.
- Thử xóa khai báo *uses crt* ; rồi quan sát thông báo lỗi.

4. Viết chương trình tính lương của một nhân viên theo công thức

Tiền lương = bậc lương/ngày công hệ số + phụ cấp*

Chương 6

CÁC LỆNH CỦA TURBO PASCAL

Như đã biết các từ khoá liên kết với nhau theo những quy tắc cần phải tuân theo (gọi là cú pháp) để tạo thành các câu lệnh, sau đây ta xét một số lệnh :

1. 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 toán tử " := ".

Cách viết :

biến := *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 :

z := 3.4 ; {đọc là gán 3.4 cho *z* hay *z* nhận giá trị 3.4}

x := *x* + 1 ; {đọc là *x* nhận giá trị mới bằng giá trị cũ cộng thêm 1}

y := *x* ; {đọc là gán *x* cho *y*}

Việc thực hiện câu lệnh gán gồm hai bước :

- 1) Tính giá trị biểu thức bên phải dấu gán, ví dụ tính *x* + 1.
- 2) Gán giá trị tính được cho biến bên trái dấu gán, ví dụ gán cho biến *x*.

2. CÁC THỦ TỤC VÀO RA DỮ LIỆU

2.1. Thủ tục vào dữ liệu qua 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 có thể gặp nhiều bất tiện. Bởi vậy nên để vào dữ liệu người ta còn dùng lệnh read hoặc readln. Có 3 dạng viết như sau :

1. `read(b1, b2, ..., bn) ;`
2. `readln(b1, b2, ..., bn) ;`
3. `readln ;`

trong đó các *bi* nhất thiết phải là các biến.

Dạng 1 và 2 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 ↵ ; hệ thống sẽ gán 5 cho *x*, 3.45 cho *y* và 12.4 cho *z*.

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 dạng 1 và 2 chỉ khác nhau chút ít. Đối với dạng 1 sau khi gõ ↵ (phím return) con trỏ màn hình *không* nhảy xuống dòng dưới ; trong khi đó đối với dạng 2 sau khi gõ ↵ (phím return) con trỏ màn hình *nhảy* xuống đầu dòng dưới.

Lệnh `readln` ở dạng 3 chỉ có tác dụng dừng màn hình để ta xem kết quả thực hiện chương trình.

2.2. Thủ tục hiển thị dữ liệu ra màn hình

Có ba dạng viết như sau :

1. `write(mucl, muc2, ..., muck) ;`
2. `writeln(mucl, muc2, ..., muck) ;`
3. `writeln ;`

Trong đó *mucl* (*i* = 1, 2, ..., *k*) là các *biểu thức* mà giá trị của nó cần hiển thị ra màn hình. Chẳng hạn trong ví dụ 5.1 của chương 5, nếu ta muốn hiển thị giá trị của diện tích ra màn hình ta cần thêm vào lệnh `write(s) ;`

Program tinh_dt ;

const

pi = 3.14 ;

var

s, r : real ;

Begin

$r := 5 ;$

$s := \pi * r * r ;$

write(s)

End.

Khi thực hiện chương trình này ta thu được kết quả :

7.853981634E+01.

Sự khác nhau giữa lệnh 1 và 2 ở chỗ vị trí con trỏ màn hình sau khi kết thúc lệnh : đối với lệnh 2, sau khi hiển thị các giá trị, con trỏ màn hình sẽ chuyển xuống đầu dòng tiếp theo ; còn đối với lệnh 1, sau khi hiển thị các giá trị, con trỏ màn hình sẽ không chuyển xuống dòng tiếp theo, mà sẽ ở vị trí ngay sau mục k.

Lệnh 3 sẽ chỉ làm một động tác đơn giản là đưa con trỏ màn hình xuống dòng tiếp theo.

Hiển thị có quy cách :

- Đối với các giá trị thực

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(mucl :ml :nl, muc2 :m2 :n2, ..., muck :mk :nk) ;

hoặc :

writeln(mucl :ml :nl, muc2 :m2 :n2, ..., muck :mk :nk) ;

Ở đây mk và nk là các số nguyên dương ; tác dụng của lệnh trên như sau : hiển thị giá trị của biểu thức *muck* ra màn hình trên tất cả *mk* cột, trong đó có *nk* cột cho phần thập phân. Ví dụ nếu ta chứa lệnh *write(s)* trong ví dụ 5.1 thành *write(s :5 :2)* ; ta thu được kết quả như sau trên màn hình :

78.54

- Đối với các giá trị nguyên

Cách hiển thị có quy cách được viết như sau :

write(btn :m) ;

hoặc :

writeln(btn :m) ;

trong đó : btn là biểu thức có giá trị nguyên, còn m là một số nguyên ; lúc đó máy sẽ dành m vị trí (cột) để hiển thị giá trị của btn , nếu còn thừa chỗ máy tính sẽ chèn thêm các kí tự trống ở bên trái.

2.3. Kết hợp giữa *write* và *readln* tạo hội thoại người-máy

Trong các ví dụ dùng *read* và *readln* vừa xét thể hiện một yếu điểm 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 phạm phải sai lầm.

Ta hãy xét một vài ví dụ liên quan tới các vấn đề này :

Ví dụ 2.1 : Viết chương trình thực hiện bốn phép tính số học.

Program So_hoc ;

Var

so1, so2, tong, hieu, tich :Integer ;

thuong :Real ;

Begin

Write('—Nhập số thu nhất = ');

Readln(so1);

Write('—Nhập số thu hai = ');

Readln(so2);

tong := so1 + so2 ;

hieu := so1 - so2 ;

*tich := so1 * so2 ;*

thuong := so1 / so2 ;

Writeln('Tong cua hai so ', so1, ' va ', so2, ' = ', tong) ;

Writeln('Hieu cua hai so ', so1, ' va ', so2, ' = ', hieu) ;

Writeln('Tich cua hai so ', so1, ' va ', so2, ' = ', tich) ;

Writeln('Thuong cua hai so ', so1, ' va ', so2, ' = ', thuong :6 :2) ;

Readln

End.

Ví dụ 2.2 : Viết chương trình tính lương của một nhân viên theo công thức :

$$\text{Tiền lương} = \text{bậc lương/ngày công} * \text{hệ số} + \text{phụ cấp}$$

Program tinh_luong ;

Var

Ten :String[24] ;

bl, nc, pc, thang :Integer ;

tl, hs :Real ;

Begin

Writeln('CHUONG TRINH TINH LUONG') ;

Writeln('-----') ;

Writeln ;

Write('— Cho biet thang : ') ; Readln(thang) ;

Write('— Cho biet ho ten : ') ; Readln(ten) ;

Write('— Cho biet bac luong : ') ; Readln(bl) ;

Write('— Cho biet ngay cong : ') ; Readln(nc) ;

Write('— Cho biet he so : ') ; Readln(hs) ;

Write('— Cho biet phu cap : ') ; Readln(pc) ;

Writeln ;

*tl := ((bl/30)*nc*hs+pc) ;*

Writeln('Tien luong thang :', thang :2, ' của Ông/Bà :', ten, ' là =', tl :8 :2) ;

Writeln('Bam phim <Enter> de ve cua so soan thao') ;

Readln

End.

2.4. Một số hàm và thủ tục trình bày màn hình trong Turbo Pascal

Các hàm và thủ tục này chứa ở trong unit CRT. Unit CRT bao gồm các hằng, hàm và thủ tục để ấn định các chế độ của màn hình như : xoá màn hình, định vị con trỏ, xác lập màu của màn hình, định đặt cửa sổ làm việc trên màn hình, vì vậy để sử dụng nó trước hết chúng ta phải có khai báo sau :

uses CRT ;

Sau đây là một số hàm và 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 bên trái được đánh số 1 và dòng trên cùng 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).

Cách đặt màu nền và màu chữ

Để xác định màu nền ta dùng thủ tục :

textbackground(color : byte) ;

Để xác định màu chữ ta dùng thủ tục :

textcolor(color : byte) ;

Trong cả hai trường hợp biến color chứa mã của 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 họa 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 trên trái có toạ độ (10, 5), góc dưới phải 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 màu tím ta viết tiếp hai lệnh sau :

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

3. CÂU LỆNH ĐIỀU KIỆN

3.1. Câu lệnh IF... THEN... ELSE

- Dạng không đầy đủ :

```
if <bt> then <lệnh> ;
```

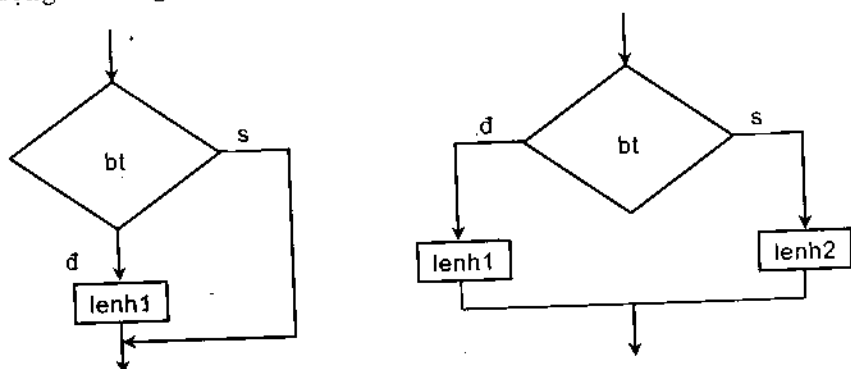
• Dạng đầy đủ :

```
if <bt> then <lenh1>
    else <lenh2> ;
```

trong đó : *if*, *then*, *else* là các từ khoá ; *bt* là một biểu thức boolean, còn *<lenh>*, *<lenh1>*, *<lenh2>* là các lệnh hoặc khối lệnh⁽¹⁾ của PASCAL

Bộ vi xử lý sẽ hoạt động như sau khi gặp lệnh này :

- Đối với dạng không đầy đủ nếu *<bt>* có giá trị đúng thì thực hiện *<lenh>* ; trong trường hợp ngược lại thì bỏ qua không thực hiện *<lenh>*.
- Đối với dạng đầy đủ nếu *<bt>* có giá trị đúng thì thực hiện *<lenh1>* ; trong trường hợp ngược lại thì thực hiện *<lenh2>*. Ta có thể minh hoạ các hoạt động đó bằng các sơ đồ khối trên hình 2.1 :



Hình 2.1

Bởi vì phần **else** của lệnh **if ...then... else** là phần tùy chọn nên có thể có xảy ra nhập nhằng khi một **else** bị bỏ qua trong một dãy các câu lệnh **if** lồng nhau. Điều đó được giải quyết theo cách thông thường : **else** luôn luôn kẹp đôi với **if** gần nhất chưa có **else**. Chẳng hạn, trong câu lệnh :

```
if <bt> then if <bt2> then <lenh1>
    else <lenh2> ;
```

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

```
if <bt> then
begin
    if <bt2> then <lenh1>
    else <lenh2> ;
end ;
```

(1) khối lệnh được hiểu là một nhóm lệnh của PASCAL được đặt trong cặp từ khoá *begin... end* :

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

Ví dụ 3.1 :

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

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) ;
```

Bạn đọc hãy viết hoàn chỉnh một chương trình cho phép tính giá trị của hàm trên.

Ví dụ 3.2 : Viết một chương trình cho phép tìm được giá trị lớn nhất trong ba giá trị được 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 max := c
```

```
  else if b > c then max := b
```

```
    else max := c ;
```

```
  write(' Gia tri lon nhat la : ', max :8 :2) ;
```

```
  readln ;
```

```
end.
```

Chú ý : Về kí pháp viết "lùi vào" khi viết một chương trình (cách viết có cấu trúc nhô ra lùi vào) ; kí pháp này giúp 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 nên thẳng hàng.

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 :

```
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 ;
tuyệt đối không nên dùng !

3.2. 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 <bi> of

hằng1 : <lệnh1> ;

hằng2 : <lệnh2> ;

.....

hằngn : <lệnhn>

end ;

Dạng 2

case <bi> of

hằng1 : <lệnh1> ;

hằng2 : <lệnh2> ;

.....

hằngn : <lệnhn> ;

else <lệnh n+1>

end ;

Trong đó <bt> không chỉ là kiểu logic mà còn có thể là kiểu : integer, byte, char, miền con, ... ; nhưng không được là kiểu thực, còn các hằng i ($i=1, \dots, n$) là các hằng có cùng kiểu với <bt>.

Chú ý : trong trường hợp có nhiều hằng i ứng với cùng một lệnh, 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 3.3 : Viết chương trình tính số ngày của một tháng :

```
var
    so_ngay, thg, nam : integer ;
begin
    write(' Hay vào một tháng : ' ) ; readln(thg) ;
    write(' Hay vào một năm : ' ) ; 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(' Tháng ', thg, ' có : ', so_ngay, ' ngày ' ) ;
    readln ;
end.
```

4. CÁC CÂU LỆNH 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.

4.1. 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 :

1. `for <bdk> := <gtd> to <gtc> do <lenh> ;`
2. `for <bdk> := <gtd> downto <gtc> do <lenh> ;`

Trong đó : `for`, `to`, `do`, `downto` là các từ khóa.

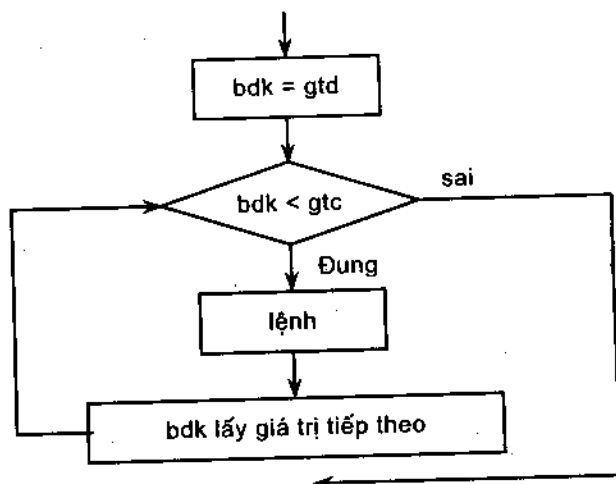
`<bdk>` là biến điều khiển vòng lặp có kiểu nguyên, miền con hay logic, ... tức là những kiểu đếm được, rời rạc, hữu hạn nhưng không được là kiểu thực.

`<gtd>`, `<gtc>` là các biểu thức có kiểu giống như biến điều khiển.

`<lenh>` là câu lệnh đơn hoặc khối lệnh của PASCAL.

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

Sơ đồ của dạng 1 như trên hình 2.2.



Hình 2.2

Ví dụ 4.1 :

Hãy viết một chương trình cho phép tính giá trị của tổng sau :

$$S = 1 + 1/2 + 1/3 + \dots + 1/n.$$

Với `n` đọc vào máy qua bàn phím.

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(' giá trị tính được là : ', s :6 :2) ;

readln ;

END.

Ta lưu ý mấy điểm sau :

1) *<gtd>* và *<gtc>* chỉ tính một lần.

2) Đối với dạng 1 nếu *<gtd>* > *<gtc>* lệnh FOR không thực hiện.

Đối với dạng 2 nếu *<gtd>* < *<gtc>* lệnh FOR không thực hiện

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

4) Câu lệnh FOR có thể lồng nhau.

Ví dụ 4.2 : Xét bài toán vui như sau :

Trăm trâu trăm cỏ

Con đứng ăn năm

Con nằm ăn ba

Ba con già ăn một bó.

Tìm số trâu mỗi loại ?

Ta lập luận như sau : gọi số trâu đứng, trâu nằm, trâu già lần lượt là *trau_dung*, *trau_nam*, *trau_gia* ; ba đại lượng này phải thoả mãn các điều kiện sau :

- $\text{trau_dung} + \text{trau_nam} + \text{trau_gia} = 100$
- $5 * \text{trau_dung} + 3 * \text{trau_nam} + (\text{trau_gia} \text{ div } 3) = 100$ {tổng số cỏ}
- Số bó cỏ là nguyên cho nên số trâu già phải chia hết cho 3 :

$$\text{trau_gia} \bmod 3 = 0 ;$$

Chương trình như sau :

```
program so_trau ;
```

```
var
```

```
    trau_dung, trau_nam, trau_gia, soco, sotrau : Integer ;
```

```
begin
```

```
    for trau_dung := 0 to 20 do
```

```
        for trau_nam := 0 to 33-trau_dung do
```

```
            for trau_gia := 0 to 100 - trau_dung - trau_nam do
```

```
                begin
```

```
                    soco := 5*trau_dung + 3*trau_nam + (trau_gia div 3) ;
```

```
                    sotrau := trau_dung + trau_nam + trau_gia ;
```

```
                    if (trau_gia mod 3 = 0) and (sotrau = 100) and (soco = 100)
                    then
```

```
                        writeln('Trau dung : ', trau_dung, ' Trau nam : ', trau_nam,
                                ' Trau gia : ', trau_gia) ;
```

```
                    end ;
```

```
                readln
```

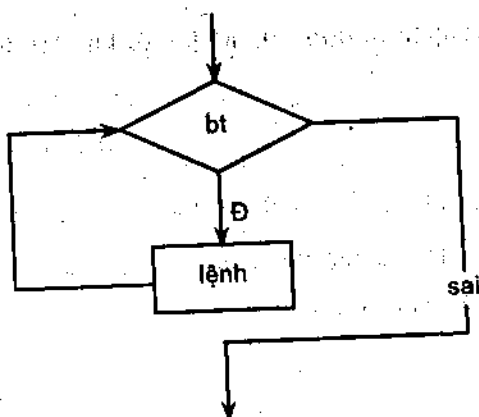
```
            end.
```

4.2. 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 sẽ 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 2.3).



Hình 2.3

Sau đây là chương trình tính tổng $S = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ đã nói đến ở trên, nhưng sử dụng lệnh WHILE :

```
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 4.3 : Tính hàm số $\sin x$ theo công thức sau :

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

số phân 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 $T_0, T_1, \dots, T_n$ ta có :

$$T_0 = x$$

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

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 := 0 ; T := x ; n := 0 ;

while abs(T) >= eps do

begin

s := s+T ; n := n+1 ;

*T := T*sqr(x) / (2*n) / (2*n+1) ;*

end ;

writeln('Gia tri ham sin tai diem', x :6 :2, ' la :', s :8 :6) ;

readln ;

end.

Ví dụ 4.4 : 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 write(' Không xét!')

else begin

i := 2

while x mod i <> 0 do i := i+1 ;

if i = x then writeln(' x là số nguyên tố')

else writeln(' x không là số nguyên tố')

end ;

end.

4.3. Câu lệnh REPEAT

Lệnh này có dạng như sau :

REPEAT

<lenh1> ;

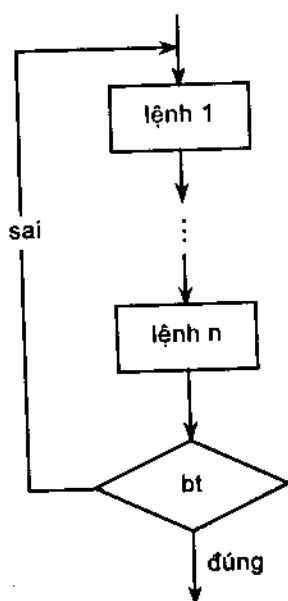
.....

<lenhn> ;

UNTIL <bt> ;

Trong đó REPEAT, UNTIL là các từ khoá, <lenhi> 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 nên đúng. Bộ xử lý sẽ hoạt động như sơ đồ hình 4.3 khi gặp lệnh này.

Ví dụ ta tính lại tổng $S = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ ở trên nhưng lần này ta dùng lệnh repeat :



Hình 2.4

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 4.5. Tính $n! = 1*2*...*n$

var gt : longint ; *i : integer ;*

begin

$gt := 1 ; i := 0 ;$

repeat

$i := i + 1 ; \{ \text{mỗi lần tăng } i \text{ lên } 1 \}$

$gt := gt * i ; \{ \text{nhân dồn } i \text{ vào } gt \}$

until $i = n ; \{ \text{nếu } i < n \text{ thì quay ngược lên ; nếu } i = n \text{ thì chấm dứt} \}$

write(gt) ;

readln ;

end.

Bài tập

1. Giải biện luận bất phương trình $ax + b > 0$.
2. Giải biện luận bất phương trình $ax^2 + bx + c > 0$ ($a \neq 0$).
3. Nhập một dãy bao gồm N số bất kì, N nhập từ bàn phím. Tính trung bình cộng các số âm và đếm số các số lẻ. Đưa kết quả ra màn hình.
4. Viết chương trình nhập vào hai cạnh góc vuông của tam giác vuông ; rồi tính diện tích và cạnh huyền của tam giác này.
5. Nạp và chạy chương trình sau :

Program Chu_so ;

Var

$n : \text{Integer} ;$

Begin

Writeln('XUAT CAC CHU SO CUA SO NGUYEN DUONG N') ;

Write('—Cho biet so nguyen : ') ;

Readln(n) ;

Writeln('So nguyen : ', n : 5, ' Co cac chu so sau') ;

Write(' ', n DIV 10000) ;

$n := n \text{ MOD } 10000 ;$

Write(' ', n DIV 1000) ;

$n := n \text{ MOD } 1000 ;$

Write(' ', n DIV 100) ;

$n := n \text{ MOD } 100 ;$

Write(' ', n DIV 10) ;

n := n MOD 10 ;

Write(' ', n) ;

Writeln ;

Writeln(' Bam phim <Enter> de ket thuc') ;

Readln

End.

6. Viết chương trình chuyển từ độ Fahrenheit (F) sang độ Celcius (C) ; biết rằng công thức đổi từ độ F sang độ C như sau :

$$C = ((F - 32) * 5) / 9.$$

7. Viết chương trình tính điểm trung bình 5 môn học của học sinh với :

- Văn hệ số 3
- Toán, lí hệ số 2
- Sinh vật, ngoại ngữ hệ số 1.

8. Viết một chương trình thực hiện các công việc sau :

- Đọc vào máy một số nguyên.
- Tính và hiển thị lên màn hình số lượng chữ số của số nguyên đó.

9. Viết chương trình nhập vào ba toạ độ x, y, z của một vectơ 3 chiều ; hãy tính và in ra màn hình độ dài của véc tơ này.

10. Hãy viết chương trình tính chu vi, diện tích của hình tròn khi nhập bán kính từ bàn phím.

11. Viết chương trình chuyển từ Mile (dặm Anh) sang km, biết rằng :

$$1 \text{ Mile} = 1.609344 \text{ km}.$$

12. Viết chương trình nhập 2 số nguyên dương a, b ; hãy tính a^b .

13. Viết chương trình nhập 3 số nguyên dương a, b, c. Hãy in ra màn hình giá trị trung bình cộng S và trung bình nhân P.

14. Viết chương trình nhập hai số nguyên không quá 3 chữ số, rồi in kết quả phép tính nhân hai số đó theo dạng :

$$\begin{array}{r} 567 \\ \times 14 \\ \hline 7938 \end{array}$$

15. Viết một chương trình nhập vào tổng số giây, rồi hiển thị lên màn hình giờ, phút, giây tương ứng.

16. Hãy viết một chương trình nhập vào hai số nguyên có 3 chữ số, rồi in ra kết quả phép tính nhân theo dạng :

$$\begin{array}{r} 457 \\ \times 345 \\ \hline 2285 \\ 1828 \\ 1371 \\ \hline 157665 \end{array}$$

17. Viết chương trình đổi chỗ hai số nguyên a, b ; sau đó tính giá trị của a^b .

18. Viết một chương trình thực hiện các công việc sau :

- Nhập một kí tự rồi hiển thị mã ASCII của nó lên màn hình.
- Nhập mã ASCII rồi in ra kí tự tương ứng.

19. Viết chương trình :

- Nhập một kí tự viết thường rồi hiển thị nó dưới dạng viết hoa.
- Nhập một kí tự viết hoa rồi hiển thị nó dưới dạng viết thường.

20. Cho một dãy số nguyên $x_1, x_2, \dots, x_{100}$. Viết một chương trình thực hiện các công việc sau :

- Nhập dãy trên vào máy thông qua bàn phím.
- Tính trung bình cộng của các số lẻ ; hiển thị giá trị đó lên màn hình.
- Đọc một số nguyên c vào máy qua bàn phím, rồi kiểm tra xem có bao nhiêu số trong dãy bằng số c vừa đọc vào.

21. Cho 2 dãy số thực x_i ($i=1, \dots, 10$) và y_i ($i=1, \dots, 10$). Viết một chương trình thực hiện các công việc sau :

- Đọc hai dãy trên vào máy thông qua bàn phím.

- Tính tổng : $S = \sum_{i=1}^{10} x_i^2 \cdot y_i$.

22. Viết một chương trình thực hiện các công việc sau :

- Đọc vào máy một số nguyên n.
- Tìm tất cả các ước số của số vừa đọc vào.

23. Hãy viết một chương trình in ra mã ASCII của các kí tự từ 'A' đến 'Z' và từ 'a' đến 'z'.
24. Biết rằng số dân trong một thành phố là 6000000 người, tỷ lệ tăng dân số hàng năm là 2% ; hãy tính dân số của thành phố đó 5 năm sau.
25. Tìm tất cả các số nguyên khác nhau a, b, c, d trong khoảng từ 0 tới 10 thỏa mãn điều kiện $a*d*d = b*c*c*c$.
26. Giải bài toán vui sau đây :
- Vừa gà vừa chó
Bó lại cho tròn
Ba mươi sáu con
Đủ một trăm chân
Hỏi có bao nhiêu gà, bao nhiêu chó.

27. Tính số e với độ chính xác ε cho trước theo công thức :

$$e = \sum_{k=0}^{\infty} \frac{1}{k!}.$$

Chương 7

CÁC KIỂU DỮ LIỆU MỞ RỘNG

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

1. 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 :

1.1. Định nghĩa kiểu

Các kiểu dữ liệu đếm được như integer, char, byte, boolean, liệt kê bao gồm các giá trị được sắp theo một thứ tự nhất định ; trong một số trường hợp nhất định chúng ta không cần sử dụng tất cả mà chỉ cần một khoảng (một số) giá trị nào đó trong số giá trị này, ta sẽ khai báo kiểu miền con ; cú pháp 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, kí 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 đoạn từ *can_duoi* đến *can_tren*.

Ví dụ :

type

tuoi_thanh_nien = 15..28 ;

toc_do_o_to = 0..300 ;

1.2. Khai báo biến

Để khai báo biến kiểu miền con ta có thể tiến hành theo hai cách :

•Gián tiếp :

Trước hết chúng ta khai báo kiểu miền con, sau đó sẽ khai báo biến, cú pháp như sau :

```
type
    ten_kieu = can_duoi..can_tren ;
var
    bien : ten_kieu ;
```

thí 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 `tui := 12` hoặc `tui := 31` là không hợp lệ ;

•Trực tiếp :

Chúng ta cũng có thể khai báo biến và kiểu một cách đồng thời như sau :

```
var
    bien : can_duoi..can_tren ;
```

chẳng hạn :

```
var
    tuoi : 15..28 ;
    van_toc : 0..300 ;
```

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⁽¹⁾.

2. 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...

(1) Cần phải khai báo chỉ thị chương trình dịch {SR+}

Kiểu mảng có một số đặc trưng sau :

– Các phần tử của mảng phải có cùng kiểu gọi là kiểu cơ sở hay kiểu thành phần.

– Các phần tử trong mảng có chỉ số, tức là số thứ tự của chúng trong mảng ; kiểu của chỉ số phải là kiểu rời rạc, chẳng hạn kiểu miền con, kiểu nguyên, kí tự, ... và có thể âm.

– Các chỉ số là các biểu thức nằm trong cặp dấu ngoặc vuông [].

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 [] ; chẳng hạn $x[5]$, $a[2, 3]$ v.v..

2.1. Mảng một chiều

Mảng là kiểu dữ liệu có cấu trúc được đề xuất sớm nhất trong các ngôn ngữ lập trình ; nó được dùng để chỉ tới một nhóm đối tượng có cùng một tính chất nào đó. Chẳng hạn vec-tơ là một nhóm các số mà mỗi số để xác định ta chỉ cần biết chỉ số. Một khai báo mảng phải được đặt bởi hai khía cạnh của mảng : kiểu các phần tử của mảng và kiểu của chỉ số.

2.1.1. Khai báo kiểu mảng một chiều

Khai báo kiểu mảng có dạng sau :

Type

$ten_k = \text{array}[kcs] \text{ of } kft ;$

trong đó : type, array, of là các từ khoá ;

kcs là kiểu của chỉ số (các kiểu có thể là integer, miền con, kí tự, char, ..., nhưng không được là kiểu real) ;

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

2.1.2. Khai báo biến mảng

Có hai cách khai báo biến mảng như sau :

- Cách thứ nhất trước hết ta khai báo kiểu mảng, sau đó khai báo biến mảng :

Type

$ten_k = \text{array}[kcs] \text{ of } kft ;$

var

bien : ten_k ;

- Cách thứ hai ta khai báo trực tiếp :

var

bien : array[kcs] of kft ;

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

type

km = array[1..20] of integer ;

var

a : km ;

Hoặc ngắn gọn hơn :

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

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] ; ...

2.1.3. Cách gán giá trị cho biến kiểu 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 := 1 to 100 do

begin

write(' So hạng thu ', i, ' là : ') ; readln(a[i]) ;

end ;

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

Ví dụ 7.1 : 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(' So hang thu ', i, ' la : ' ) ;  
            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 7.2 : Cho một dãy số nguyên $x_1, x_2, \dots, x_{100}$. Viết một chương trình thực hiện các công việc sau :

- Nhập dãy trên vào máy thông qua bàn phím.
- Tính trung bình cộng của các số lẻ. Hiển thị giá trị đó lên màn hình.

program giai 3 ;

```
var    x : array[1..100] of integer ;  
        i, tong, c, d : integer ; tbc : real ;  
begin  
    for i := 1 to 100 do
```

```

begin
    write(' vào số thứ ', i) ; readln(x[i]) ;
end ;
tong := 0 ; d := 0 ;
for i := 1 to 100 do
    if odd(x[i]) then
        begin
            tong := tong + x[i] ; d := d + 1 ;
        end ;
    if d <> 0 then
        begin
            tbc := tong / d ; writeln(' trung bình cộng là : ', tbc : 10 : 4) ;
        end
        else writeln(' Không có số lẻ') ;
    readln ;
end.

```

Ví dụ 7.4 : Cho một dãy số thực $x_1, \dots, x_{100}$; hãy sắp xếp lại 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 100 do
        if x[i] > x[j] then
            begin
                tg := x[i] ; x[i] := x[j] ; x[j] := tg ;
            end ;

```

Bạn đọc hãy tự hoàn thiện chương trình.

2.2. 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.

2.2.1. Khai báo kiểu mảng hai chiều

Một kiểu mảng hai chiều được khai báo như sau :

type

ten_kieu = array[kcs1, kcs2] of kft ;

kcs1, kcs2 : là kiểu của chỉ số (integer, miễn con, kí tự...)

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

2.2.2. Khai báo biến mảng

Cũng giống như trong trường hợp mảng một chiều, ta có hai cách khai báo :

– Cách thứ nhất trước hết ta khai báo kiểu mảng, sau đó khai báo biến

mảng :

Type

ten_k = array[kcs1, kcs2] of kft ;

var

bien : *ten_k* ;

– Cách thứ hai ta khai báo trực tiếp :

var

bien : array[kcs1, kcs2] of kft ;

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 7.5 : Viết chương trình thực hiện việc cộng hai 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*

for j := 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] ;
        writeln('Ma tran tong :') ;
    for i := 1 to 4 do
        begin
            for j := 1 to 5 do write(c[i, j] :6 :2) ;
            writeln
        end ;
    end.

```

3. KIỂU XÂU KÍ TỰ

3.1. Khai báo kiểu

Khai báo một kiểu xâu như sau :

```

type
    ten_k = string[n] ;

```

trong đó : *ten_k* là một tên do người lập trình đặt ra.

string là từ khoá, còn *n* báo cho máy biết số ô nhớ (cụ thể là byte) tối đa cần dành sẵn cho mỗi biến thuộc kiểu này.

Ví dụ :

```

type
    str10 = string[10] ;

```

3.2. Khai báo biến

Để khai báo biến xâu ta có thể thực hiện theo hai cách :

– Cách thứ nhất trước hết ta khai báo kiểu xâu, sau đó sẽ khai báo biến xâu, cú pháp như sau :

```

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

```

– Cách thứ hai ta có thể khai báo 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 dùng 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 := 'TIN YEU' ;

thì *s1* có dạng như sau :

#8	T	I	N			Y	E	U		
----	---	---	---	--	--	---	---	---	--	--

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 := 'TIN YEU'* thì

write(s1[1]) sẽ in ra ký tự 'T'

còn *s1[4] := 'H'* làm cho *s1 = 'TINH YEU'*

Muốn tính chiều dài của xâu ta có thể tính *ord(s1[0])*.

3.3. Biểu thức đối với string

– Phép nối string ký hiệu bởi phép cộng (+), dùng để nối nhiều string thành một. Ví dụ :

'khoa'+ '40' có trị là *'khoa40'*

– 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'*

– Phép gán :

Có dạng *biến := biểu thức* ; ví dụ *s1 := 'a'+ 'n'* ;

3.4. Các hàm và thủ tục xử lý xâu

a) Hàm *length(x)* cho độ dài thực tế của xâu *x*.

b) Hàm *copy(x, m, n)* cho ta *n* ký tự của xâu *x* kể từ ký tự thứ *m*.

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

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

e) Thủ tục `delete(x, m, n)` : xoá `n` kí tự của `x` kể từ kí tự thứ `m`.

f) Thủ tục `insert(y, x, n)` : chèn xâu `y` vào xâu `x` từ kí tự thứ `n`.

g) Thủ tục `val(x, i, ma)` : Thủ tục này 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 kí tự đầu tiên không chuyển đổi được.

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

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

Ta xét một vài ví dụ minh hoạ cho công việc xử lí xâu :

Ví dụ 7.6 : Lập trình thực hiện các công việc sau :

1. Nhập vào một xâu kí 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 ;
    write(c, ' ', D) ; readln ;
end ;
readln ;
end.

```

Ví dụ 7.7 : Lập chương trình thực hiện các công việc sau :

- Đọc qua bàn phím họ và tên của một người
- Hiển thị tên của người đó lên màn hình

Chương trình như sau :

```

program trích_ten ;
var
    ht : string[20] ; ten : string[10] ;
    i : integer ;
begin
    write(' hay nhap ho va ten :') ;
    readln(ht) ;
    ten := '' ; i := length(ht) ;
    while ht[i] <> '' do
        begin
            ten := ht[i] + ten ;
            i := i -1 ;
        end ;
    write(' Ten nguoi do la : ', ten) ;
end.

```

Bài tập

1. Hãy viết chương trình thực hiện yêu cầu sau :

- Nhập một dãy N số nguyên từ bàn phím, với N=10. In dãy số vừa nhập ra màn hình.
- Tính trung bình các số chẵn và chia hết cho 3.

Cho biết trong dãy số đã nhập có bao nhiêu số có giá trị > giá trị trung bình vừa tính được ở trên.

2. Viết một chương trình sắp xếp lại một dãy số theo thứ tự tăng dần trong dấu giá trị tuyệt đối.
3. Viết một chương trình thực hiện các công việc sau :
 - Đọc vào máy một dãy số thực có 300 phần tử từ bàn phím.
 - Xoá khỏi dãy này những phần tử bằng với c (c đọc vào máy qua bàn phím).
 - Hiển thị dãy trước và sau khi xử lí.
4. Cho một dãy số thực $x_1, x_2, x_3, \dots, x_n$ ($n \leq 200$), hãy viết một chương trình thực hiện các công việc sau :
 - Đọc n và các phần tử của dãy vào máy qua bàn phím.
 - Sắp xếp lại dãy trên theo nguyên tắc sau : các phần tử bằng 0 đứng đầu, tiếp đến các phần tử âm, sau cùng là các phần tử dương.
 - Hiển thị dãy trước và sau khi sắp xếp.
5. Viết một chương trình thực hiện các công việc sau :
 - Đọc các phần tử của dãy số thực có 20 phần tử vào máy qua bàn phím.
 - Đếm xem trong dãy đó có bao nhiêu phần tử đạt giá trị lớn nhất.
6. Cho một dãy số nguyên $x_1, x_2, x_3, \dots, x_n$ ($n \leq 300$), hãy viết một chương trình thực hiện các công việc sau :
 - Đọc n và các phần tử của dãy vào máy qua bàn phím.
 - Sắp xếp lại dãy trên theo nguyên tắc sau : các phần tử chẵn đứng đầu, tiếp đến các phần tử lẻ.
 - Hiển thị dãy trước và sau khi sắp xếp.
7. Viết một chương trình thực hiện các công việc sau :
 - Đọc vào máy một xâu kí tự.
 - Xác định xem có tất cả bao nhiêu kí tự khác nhau xuất hiện trong xâu trên và chúng là những kí tự nào.
8. Viết một chương trình thực hiện các công việc sau :
 - Nhập vào một số nguyên bất kỳ.
 - Hiển thị lên màn hình số vừa nhập vào có bao nhiêu chữ số.
9. Viết một chương trình thực hiện các công việc sau :
 - Đọc vào máy một xâu kí tự.
 - Tạo một xâu mới bao gồm toàn các kí số của xâu vừa nhập vào.

10. Viết chương trình tìm tập hợp các kí tự không xuất hiện trong một xâu kí tự, biết rằng xâu đó được đọc vào máy qua bàn phím.
11. Viết một chương trình thực hiện các công việc sau :
 - Đọc vào máy một câu văn.
 - Tính xem trong câu vừa đọc vào có bao nhiêu từ (từ là một nhóm kí tự không chứa dấu trống)
12. Viết chương trình đổi một số nguyên hệ thập phân sang hệ nhị phân.
13. Viết chương trình đổi một số nguyên hệ thập phân sang hệ bát phân.
14. Viết chương trình đổi một số nguyên hệ thập phân sang hệ thập lục phân.
15. Viết một chương trình thực hiện các công việc sau :
 - Đọc vào máy một xâu kí tự.
 - Hiển thị lên màn hình xâu vừa đọc vào theo thứ tự ngược lại.

Chương 8

BẢN GHI (RECORD)

1. KHÁI NIỆM RECORD (BẢN GHI)

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 thực tế xuất hiện nhiều đối tượng cần xử lý mà chúng ta phải dùng nhiều kiểu dữ liệu để mô tả chúng.

Chẳng hạn để lưu giữ những thông tin liên quan đến một đối tượng nhân viên, ta cần có một biến nào đó có khả năng lưu trữ được cả tên, địa chỉ, ngày sinh lẫn mã số nhân viên, mức lương, v...v... để có thể xử lý biến này như một phần tử thống nhất, thể hiện thông tin của một nhân viên cụ thể. Ngôn ngữ PASCAL cho phép chúng ta tự xây dựng những kiểu dữ liệu phức hợp như vậy và sử dụng những kiểu dữ liệu này để khai báo cho các biến sử dụng sau đó. Chúng ta gọi những kiểu dữ liệu như vậy là kiểu bản ghi.

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.

2. KHAI BÁO RECORD

2.1. Khai báo kiểu

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 ; *k1*, *k2*, ..., *kn* là các kiểu dữ liệu sẵn có của PASCAL hoặc là một kiểu dữ liệu đã giới thiệu ở chương trước.

Ví dụ :

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

Ta dễ thấy rằng 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 ;
```

end ;

2.2. Khai báo biến bản ghi

2.2.1. Gián tiếp

Trước hết ta khai báo kiểu, sau đó khai báo biến :

```
type
  tk_bg = record
    -
    ...
  end ;
var
  bien : tk_bg ;
```

Ví dụ :

```
type
  dia_chi = record
    so_nha : integer ;
    pho, quan, th_pho : string[15] ;
  end ;
  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 ;
  end ;
var
  ng1, ng2 : ly_lich ;
```

2.2.2. Trục tiếp

Chúng ta có thể khai báo biến đồng thời với định nghĩa kiểu một cách tường minh như sau :

```
var
    bien : record
    -
    ...
end ;
```

Ví dụ :

```
var
    ng1, ng2 : record
        ht : string[25] ;
        cv : string[15] ;
        lg : real ;
        cho_o : record
            so_nha : integer ;
            pho, quan, th_pho : string[15] ;
        end ;
        n_sinh : record
            ngay : 1..31 ;
            thang : 1..12 ;
            nam : integer ;
        end ;
    end ;
```

2.3. Truy nhập vào các trường của một record

Các thủ tục nhập, xuất dữ liệu áp dụng cho các kiểu dữ liệu mà ta đã biết từ trước tới nay *không thể* áp dụng cho record được ; nói rõ hơn là không thể truy xuất thông qua các biến bản ghi mà phải thông qua các trường.

Thực vậy ta thử chạy chương trình dưới đây sẽ thấy ngay vấn đề :

Chương trình sau sẽ nhập từ bàn phím và hiển thị lên màn hình tọa độ của điểm **M** trong hệ trục tọa độ đề_các :

type

toa_do = record

x, y : integer ;

end ;

var

M : toa_do ;

begin

write('Nhap hai toa do cua M : ');

readln(M) ;

wite(' Hien thi hai toa do : ', M) ;

readln ;

end.

Khi chạy sẽ xuất hiện thông báo "*Cannot Read or write variable of this type*".

Chương trình phải sửa như sau :

type

toa_do = record

x, y : integer ;

end ;

var

M : toa_do ;

begin

write('Nhap hai toa do cua M : ');

readln(M.x, M.y) ;

wite(' Hien thi hai toa do : ', M.x, ', ', M.y) ;

readln ;

end.

Như vậy ta thấy 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_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 ta tránh được một loạt lệnh gán như sau :

ng1.ht := ng2.ht ;

ng1.cv := ng2.cv ;

....v...v

2.4. Lệnh *with ... do...*

Ta thấy để truy nhập tới một trường ta không những chỉ ra tên của trường mà còn chỉ ra tên các bản ghi chứa trường này ; điều này gây ra sự lãng phí công sức và tẻ nhạt trong lập trình. Để khắc phục nhược điểm này ta sử dụng lệnh *with* :

With ten_ban_ghi do

begin

....

end ;

trong khoảng *begin ...end* ta chỉ cần nêu tên các trường cần truy cập

Ví dụ :

```
with ngl 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 ngl, 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.

3. MẢNG CÁC BẢN GHI

Giả sử chúng ta cần xử lí lí 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 cũng chính là một biến (ngl chẳng hạn). 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 ;
```

Sau đây ta xét một số ví dụ :

Ví dụ 1 : Một lớp có tối đa 60 học sinh. Với mỗi học sinh cần quản lí 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'.

Hãy viết một chương trình thực hiện các công việc sau :

1. Đọc các thông tin về các học sinh vào máy qua bàn phím.
2. Hiển thị các học sinh giỏi lên màn hình.

Lời giải :

```
program ql_hs ;
```

```
type
```

```
hs = record
```

```
ht :string[30] ;
```

```
tuoi :byte ;
```

```
dc :string[35] ;
```

```
t, v :real ;
```

```
pl :string[10] ;
```

```
end ;
```

```
var
```

```
ds : array[1.. 60] of hs ;
```

```
n, i, j : byte ;
```

begin

repeat

write(' Đọc số lượng học sinh : ') ; readln(n) ;

until (n>0) and (n<=60) ;

for i :=1 to n do

begin

writeln('Cac so lieu ve hoc sinh thu ', i) ;

with ds[i] do

begin

write(' Ho và ten : ') ; 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 binh' ;

if (t+v >= 14) and (t+v < 18) then pl := 'kha' ;

if (t+v>=18) then pl := 'gioi' ;

end ;

end ;

writeln(' Danh sach hoc sinh gioi : ') ;

for i :=1 to n do

with ds[i] do

if pl = 'gioi' then writeln(ht) ;

readln

end.

Ví dụ 2 : Viết một chương trình nhập 5 câu hỏi trắc nghiệm, mỗi câu gồm :

- Câu hỏi ;
- Các nghĩa của câu hỏi A, B, C, D ;

- Đáp án ;

- Bạn chọn nghĩa nào đúng thì gõ nghĩa đó vào,
máy sẽ cho thống kê các câu trả lời đúng.

Sau đây là một lời giải :

```
program thi_trac_nghiem ;
```

```
type
```

```
    trac_ngh = record
```

```
        cau_hoi : string ;
```

```
        A, B, C, D : string ;
```

```
        da : char ; {đáp án}
```

```
    end ;
```

```
var
```

```
    A : array[1..5] of trac_ngh ;
```

```
    i, dung : integer ;
```

```
    tl : char ; {trả lời}
```

```
begin
```

```
    writeln(' CAU HOI TRAC NGHIEM :') ;
```

```
    writeln(' Nhap 5 cau hoi, y nghĩa và đáp án :') ;
```

```
    for i := 1 to 5 do
```

```
        with a[i] do
```

```
            begin
```

```
                write('Cau hoi thu ', i, ' :') ; readln(cau_hoi) ;
```

```
                write(' Nghia A : ') ; readln(A) ;
```

```
                write(' Nghia B : ') ; readln(B) ;
```

```
                write(' Nghia C : ') ; readln(C) ;
```

```
                write(' Nghia D : ') ; readln(D) ;
```

```
                write(' Đáp án : ') ; readln(da) ;
```

```
            end ;
```

```
    dung = 0 ;
```

```

writeln('Hay chon nghĩa cho tung cau hoi : ');
for i := 1 to 5 do
begin
    writeln(i, a[i].cau_hoi) ;
    writeln('A) ', a[i].A) ;
    writeln('B) ', a[i].B) ;
    writeln('C) ', a[i].C) ;
    writeln('D) ', a[i].D) ;
    write(' Hay lua bang cach go nghĩa cau do : ');
    readln(tl) ;
    if upcase(tl) = upcase(a[i].da) then dung = dung + 1 ;
end ;
writeln(' Ban da tra loi dung ', dung, ' cau') ;
readln
end.

```

4. PHONG CÁCH VIẾT CHƯƠNG TRÌNH

Tới đây chúng ta đã nắm bắt được khá nhiều kỹ năng lập trình ; bởi vậy để tạo thuận cho các bước tiếp theo chúng ta hãy cùng nhau đề cập một chút về phương diện hình thức trình bày một chương trình.

Như ta đã biết các trình biên dịch không bị ảnh hưởng bởi phong cách viết, một chương trình được trình bày với phong cách tồi hay tốt đều được biên dịch như nhau ; thế nhưng một phong cách tốt làm chương trình dễ đọc, dễ hiểu và dễ sửa. Một chương trình với phong cách viết tốt phải có những đặc điểm sau :

- Chú thích đầy đủ.
- Sử dụng kí pháp thụt đầu hàng, khoảng trắng, dòng trắng giúp cho chương trình dễ đọc.
- Các tên gọi trong chương trình là các tên có tính diễn nghĩa.

4.1. Chú thích

Chú thích được bao trong dấu ngoặc như sau :

(* Đây là chú thích *)

hoặc : { Đây là chú thích }

Chú thích thường xuất hiện ở đầu chương trình hay ở đầu chương trình con và thân chương trình.

Chú thích ở đầu chương trình giải thích mục đích của chương trình như : mô tả ý tưởng của giải thuật, liệt kê các dữ liệu vào, kiểu của chúng và cách nhập,... điều này rất quan trọng cho người sử dụng chương trình hay cho chính tác giả sau một thời gian nhất định.

Chú thích trong thân chương trình thường ngắn gọn không quá một vài hàng ; Ví dụ :

```
{ hiển thị giá trị lớn nhất lên màn hình }  
write('Giá trị lớn nhất là : ', max:8:3) ;
```

4.2. Dùng khoảng trống và thụt đầu hàng

Chúng ta hãy hình dung một cuốn sách không có biên, lề, không có xuống dòng, không có hàng trống,... cuốn sách đó sẽ khó đọc, khó hiểu. Với chương trình cũng thế, mặc dù là khoảng trống và thụt đầu hàng không ảnh hưởng gì đến trình dịch nhưng chúng làm chương trình trở nên dễ chịu (thân thiện) hơn cho người đọc.

Thụt đầu hàng thường được dùng để 'tách' các cấu trúc điều kiện và vòng lặp. Một lệnh if có thể viết :

```
if x <> y then tong := tong + x else write(x, ' là số trung') ;
```

nhưng cũng có thể trình bày :

```
if x <> y then tong := tong + x  
else write(x, ' là số trung') ;
```

hay :

```
if x <> y then  
tong := tong + x  
else write(x, ' là số trung') ;
```

Cả 3 cách viết đều chính xác nhưng rõ ràng là hai cách sau tỏ ra sáng sủa hơn, người ta cũng nói rằng hai cách sau có "cấu trúc" hơn.

Việc chen vào các khoảng trống giữa các lệnh hoặc giữa lệnh và các biến trong đa số các trường hợp giúp cho chương trình dễ đọc ; chẳng hạn sau đây là một vài lệnh không có khoảng trống :

```
readln(a,b,c) ;
```

```
thanhtien :=dongia*soluong ;
```

và có khoảng trống :

```
readln(a, b, c) ;
```

```
thanhtien := dongia * soluong ;
```

Các dòng trống thường để phân cách các phần khác nhau trong một chương trình. Ví dụ giữa các chương trình con, giữa phần nhập dữ liệu và phần xử lý dữ liệu,...

4.3. Dùng tên có tính diễn nghĩa

Dùng tên biến, hằng,... có nghĩa phù hợp với đại lượng mà chúng biểu diễn cũng là một yếu tố quan trọng góp phần làm cho chương trình dễ đọc và dễ hiểu. Ta có thể so sánh các lệnh :

```
thanh_tien := don_gia * so_luong ;
```

```
t := g * s ;
```

rõ ràng cách viết đầu dễ hiểu và dễ kiểm tra hơn.

Bài tập

1. Viết chương trình thực hiện các công việc sau :

- Nhập vào máy điểm văn, điểm toán của một lớp học có không quá 100 học sinh.
- Hiển thị các thông tin trên lên màn hình.

2. Viết chương trình nhập họ tên, ngày tháng năm sinh của nhân viên trong một xí nghiệp, sau đó sắp xếp lại danh sách trên theo thứ tự tăng dần của tuổi ; hiển thị danh sách đó trước và sau khi sắp xếp.

3. Viết chương trình thực hiện các công việc sau đây :

- Nhập họ lót, tên và điểm của từng học sinh trong một lớp học có sĩ số nhiều nhất là 100.
- Sắp xếp lại danh sách trên theo thứ tự từ điển.
- Hiển thị danh sách trên trước và sau khi sắp xếp.

4. Viết chương trình nhập hồ sơ cán bộ của một cơ quan bao gồm : Họ tên, tuổi, bậc lương. Hiển thị lên màn hình người có tuổi lớn nhất, người có tuổi nhỏ nhất ; người có lương lớn nhất, người có lương thấp nhất.

5. Viết chương trình thực hiện các công việc sau đây :

Nhập tối đa 100 hoá đơn bán hàng ; mỗi hoá đơn bao gồm các thông tin như sau : người mua, mã hàng, đơn giá, số lượng. Tính tổng số tiền bán được theo từng mã hàng.

6. Viết chương trình tạo một từ điển gồm 6 từ Anh – Việt, lưu mỗi từ vào một bản ghi gồm hai trường nghĩa tiếng Anh và Việt tương ứng ; sau đó nhập một từ tiếng Anh rồi đưa ra màn hình nghĩa tiếng Việt tương ứng hoặc thông báo nếu từ đó không có trong từ điển.

7. Viết chương trình nhập 5 câu hỏi trắc nghiệm, mỗi câu gồm :

– Câu hỏi.

– Các lựa chọn A, B, C, D.

– Đáp án.

sau đó trắc nghiệm trên máy rồi thống kê số câu trả lời đúng.

Chương 9

TỆP TIN (FILE)

1. KHÁI NIỆM

Ta đã thấy các chương trình khi chạy thường sản sinh ra những khối lượng dữ liệu lớn mà sau này chúng ta thường có thể sẽ sử dụng lại nhiều lần, vì vậy việc lưu trữ chúng là hoàn toàn cần thiết ; muốn vậy người ta tổ chức chúng thành từng đơn vị để ghi (hay còn gọi là lưu trữ) chúng vào thiết bị mang tin (bộ nhớ ngoài) như ổ đĩa, băng từ..., các đơn vị này gọi là 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 lượng phần tử là không hạn chế.

Bên cạnh đó như đã giới thiệu trong các chương trước ta đã thấy có 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 chương này chúng ta sẽ học cách để vượt qua tổn thất trên nhờ cách tạo lập các *file* (tệp tin).

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 kì phần tử nào trong file.

2. KHAI BÁO TẬP

2.1. Khai báo kiểu

Kiểu tập được định nghĩa trong mục khai báo *type* :

type

ten_kieu = file of kft ;

trong đó : *file*, *of* là các từ khoá ; *kft* là kiểu dữ liệu của các phần tử của tập ;

Ví dụ :

type

T = file of integer ;

R = file of real ;

Chú ý : *kft* có thể là kiểu do chính người lập trình định nghĩa, chẳng hạn :

type

HocSinh = record

HoTen : string[20] ;

Van, Toan, Ly, Hoa : integer ;

end ;

ths = file of HocSinh ;

2.2. Khai báo biến tập

Biến tập 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 ;

nhân_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 ;

3. CÁC THAO TÁC TRÊN FILE

3.1. Mở 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 như :

assign(f1, 'NGUYEN.DAT') ;

rewrite(f1) ;

Lệnh đầu gán tệp có tên là NGUYEN (đuôi là DAT) cho biến tệp ; sau các lệnh này, 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 một 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, x+1) ;
```

```
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 nhật tệp để phản ánh trạng thái mới của tệp.

Ví dụ 1 : 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.

```
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.
```

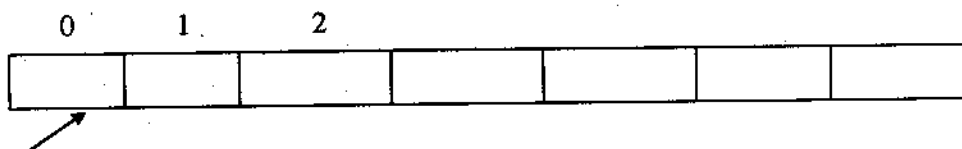
3.2. Mở tệp để đọc

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 giải nghĩa ; lệnh thứ hai có ý nghĩa như sau : tệp trên đĩa (có tên là ten_tep) vừa gán cho biến tệp bt được mở ra để sẵn sàng chờ xử lí ; 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 biến thuộc kiểu thành phần của file, các biến này đặt cách nhau bởi một dấu phẩy ; 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 chưa? Đây là một hàm kiểu boolean cho kết quả là 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.

Ví dụ 2 : Chương trình sau đây sử dụng hàm *EOF* để đọc tệp 'NGUYEN.DAT' và tính tổng của tất cả các phần tử của nó :

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+i
end ;
close(f) ;
END.

```

3.3. 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 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.

Ví dụ 3 : 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.

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, j) ; 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.

```

3.4. Các hàm và thủ tục xử lý tệp

1) Các hàm

– *filesize(bt)* hàm này cho ta số lượng phần tử trong tệp ; khi tệp rỗng cho ta giá trị 0.

– *filepos(bt)* cho ta vị trí hiện tại của con trỏ tệp.

2) Các thủ tục

– *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* ;

– *erase*

Cách viết : *erase(bt)*.

Công dụng : xoá tệp đang kết hợp với *bt*.

Để kết thúc ta xét một Ví dụ : 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 ; việc 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'.

Hãy viết một chương trình lưu trữ các thông tin trên vào một tệp với tên là 'LOP.DAT'

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(' Doc so luong hoc sinh : ' ) ; readln(n) ;
  until (n>0) and (n<=60) ;
  assign(f, 'LOP.DAT') ; rewrite(f) ;
  for i :=1 to n do
    begin
      writeln(' Cac so lieu ve hoc sinh thu ', i) ;
      with x do
        begin
          write(' Ho va ten : ' ) ; readln(ht) ;
          write(' Tuoi : ' ) ; readln(tuoi) ;
          write(' Dia chi : ' ) ; readln(dc) ;
          write(' Diem toan : ' ) ; readln(t)
          write(' Diem van : ' ) ; readln(v) ;
```

```

        if  $t+v < 10$  then  $pl := 'kem'$  ;
        if  $(t+v \geq 10)$  and  $(t+v < 14)$  then  $pl := 'trung bình'$  ;
        if  $(t+v \geq 14)$  and  $(t+v < 18)$  then  $pl := 'kha'$  ;
        if  $(t+v \geq 18)$  then  $pl := 'giỏi'$  ;
    end ;
    write(f, x) ;
end ;
close(f) ;
end.

```

4. TẬP VĂN BẢN

Tập văn bản là tập mà các phần tử của tập là các kí 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 kí 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 :

Tập văn bản
Rất hay dùng

được bố trí như sau trong bộ nhớ ngoài :

Tập văn bản	CR LF	Rất hay dùng	EOF
-------------	-------	--------------	-----

Chú ý : • Tập 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 tập văn bản chứa các kí 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.

4.1. Khai báo tập văn bản

Các biến tập văn bản được định nghĩa theo mẫu sau :

```

var
    bt_vb : text ;

```

Trong đó : *bt_vb* là một tên, còn *text* là từ khoá.

4.2. Ghi vào tệp văn bản

Để ghi vào tệp văn bản, 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 *bi* là các giá trị cần ghi vào tệp văn bản chúng có thể là các biểu thức kiểu integer, kiểu real, kiểu char, 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 là sau khi đưa các giá trị vào tệp nó chèn thêm kí hiệu *eoln* vào tệp.

Dạng thứ ba chỉ thực hiện việc đưa dấu hết dòng vào tệp.

Ví dụ 4 : Ta có thể viết lại chương trình ở tiết trước bằng cách ghi dữ liệu lên một tệp văn bản :

```
program ql_lop ;
```

```
type
```

```
hs = record
```

```
    ht : string[30] ;
```

```
    tuoi : byte ;
```

```
    dc : string[35] ;
```

```
    t, v : real ;
```

```
    pl : string[10] ;
```

```
end ;
```

```
var
```

```
    f : text ;
```

```
    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 := 'kha' ;
                if (t+v>=18) then pl := 'gioi' ;
            end ;
        writeln(f, ht :30, tuoi :3, dc :35, t :4 :1, v :4 :1, pl :10) ;
    end ;
close(f) ;
end.

```

4.3. 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 kí tự, kiểu integer, kiểu real.

Lệnh dạng 1 : sau khi đọc hết các biến con trở tệp không chuyển xuống dòng tiếp theo.

Lệnh dạng 2 : sau khi đọc hết các biến con trở tệp chuyển xuống dòng tiếp theo.

Lệnh dạng 3 : chuyển con trở tệp xuống dòng tiếp theo.

Người ta xây dựng hai hàm kiểu boolean để kiểm tra việc 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 ...

Để minh hoạ phương pháp đọc một tệp văn bản ta xét ví dụ sau.

Ví dụ 5 : Giả sử rằng người ta muốn lưu trữ thông tin về một lớp học sinh trong một tệp văn bản '*danh_sach.txt*' tại thư mục gốc của ổ C, thông tin về mỗi học sinh được lưu trữ trên một dòng với quy cách *họ và tên* chiếm 24 vị trí, *điểm* chiếm 4 vị trí ; hãy viết chương trình đọc tệp văn bản trên rồi hiển thị danh sách đó lên màn hình. Chương trình như sau :

type

hs = record

ht :string[24] ;

d :integer ;

end ;

var

tro : hs ;ch : char ;

f : text ;ten : string[24] ;

begin

assign (f, 'c : \danh_sach.txt') ;reset (f) ; ch := '-' ;

while not eof (f) do

begin

readln (f, tro.ht, tro.d) ;

```
writeln(tro.ht, ch : (24-length(tro.ht)), tro.d : 4)
```

```
end ;
```

```
readln ;
```

```
close(f) ;
```

```
end.
```

4.4. 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) ;
```

Cập lệnh này mở tệp tin đã có theo lối ghi và định vị con trỏ tệp ở cuối tệp, khi đó lần dùng lệnh write (hay writeln) sẽ thêm văn bản vào cuối tệp.

Ví dụ giả sử ta có tệp văn bản 'SO.TXT' chứa 50 số nguyên từ 1 đến 50 ; c 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.
```

5. CÁC TỆP 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 : Pascal coi các thiết bị vào ra như bàn phím, màn hình, máy in,... các cổng như tệp văn bản, nghĩa là việc xuất nhập được thực hiện bằng các thao tác như đối với tệp văn bản. Chúng ta có thể sử dụng các thiết bị chuẩn đó mà không cần phải khai báo.

Sau đây là bảng tên các thiết bị trong TURBO PASCAL :

Tên	Chức năng	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

Ví dụ 6 : Chương trình sau đây cho phép ta in tất cả các số nguyên từ 1 tới 50 :

```
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) ; {dẩy giấy khỏi máy in}
```

```
close(may_in) ;
```

```
END.
```

Ví dụ 7 : Về cách sao chép tệp tin

Program Sao_Chep_Tep_Tin ;

Var

Dich, Nguon : File of byte ;

Tep_dich, Tep_nguon : String[50] ; {Tên tệp đích, tên tệp nguồn}

ch : byte ;

Begin

Writeln('SAO CHEP TU TEP TIN NGUON SANG TEP TIN DICH') ;

Write('Tep tin nguon la : ') ;

Readln(Tep_nguon) ;

Write('Tep tin dich la : ') ;

Readln(Tep_dich) ;

Assign(Nguon, Tep_nguon) ;

Reset(Nguon) ;

Assign(Dich, Tep_dich) ;

Rewrite(Dich) ;

Writeln ;

Writeln('Dang chep Tep tin : ', Tep_nguon, ', Filesize(Nguon), ' bytes') ;

Writeln ;

while not eof(Nguon) do

begin

read(Nguon, ch) ;

write(Dich, ch)

end ;

Write('Da sao chep xong, bam <Enter>... ') ;

Readln ;

Close(Nguon) ;

Close(Dich) ;

End.

Bài tập

- Viết chương trình nhập dữ liệu và ghi vào tệp KETQUA.DAT gồm các nội dung sau :

Họ và tên ;

Điểm toán, điểm văn, điểm lí ;

Điểm trung bình (tính theo công thức $((\text{toán} \times 2) + (\text{văn} \times 2) + \text{li}) / 5$).

quá trình nhập sẽ kết thúc khi gặp họ tên là một chuỗi rỗng.

- Viết chương trình nhập thêm dữ liệu vào tệp KETQUA.DAT đã có ở trên.
- Viết chương trình tạo hai tệp định kiểu lưu trữ các số thực ; sau đó ghép hai tệp này thành một tệp mới.
- Viết chương trình tạo một danh bạ điện thoại điện tử để lưu trữ họ tên, số điện thoại, địa chỉ vào một tệp ; khi muốn tìm số điện thoại và địa chỉ của ai thì gõ tên của người đó qua bàn phím.
- Viết chương trình lưu trữ vào tệp tin LUONG.DAT gồm các thành phần :
Họ và tên ;
Chức vụ ;
Bậc lương.
khi không muốn nhập nữa thì gõ phím ESC.
- Viết chương trình đọc một tệp tin văn bản, tên tệp này đọc từ bàn phím. Tìm và hiển thị lên màn hình các dòng ngắn nhất và dài nhất.
- Viết chương trình đếm các chữ cái viết hoa trong một tệp văn bản ; tên tệp được đọc từ bàn phím.
- Viết chương trình nhập vào máy một từ (ví dụ "bk46"), sau đó đọc một tệp văn bản và kiểm tra xem từ đó có xuất hiện trong tệp văn bản đó không? Nếu có thể hiện rõ, nó xuất hiện bao nhiêu lần? trong trường hợp ngược lại cho hiển thị thông báo thích hợp.
- Viết chương trình đếm các chữ số trong một chuỗi (String) được đọc từ bàn phím.

Chương 10

CHƯƠNG TRÌNH CON

1. KHÁI NIỆM

Chương trình con trong PASCAL bao gồm hai loại *thủ tục (procedure)* và *hàm (function)*, do nhà thiết kế ra ngôn ngữ hoặc người lập trình tạo ra để khi cần đến chỉ cần gọi chương trình con đó ra thực hiện mà không cần viết lại nữa, thông qua đó không những tiết kiệm được thời gian và công sức lập trình mà còn nâng cao được tốc độ thực hiện chương trình.

Sở dĩ như vậy là vì trong chương trình chúng ta thường thấy 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. Rõ ràng nên 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 phần 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.

Như đã nói trong PASCAL có hai loại chương trình con : *function* và *procedure*, chúng được khai báo cuối cùng trong phần khai báo.

2. HÀM (FUNCTION)

2.1. Cấu trúc của một hàm

Cấu trúc của một hàm như sau :

function ten_ham (danh sách các tham số) : kiểu ;

$\left. \begin{array}{l} - \\ \dots \\ - \end{array} \right\}$ phần khai báo

Begin

$\left. \begin{array}{l} - \\ \dots \\ - \end{array} \right\}$ các lệnh của chương trình con

End ;

Nhận xét : về phương diện cấu trúc giống như chương trình chính

Ví dụ 1 :

Program Uoc_so_chung_lon_nhat ;

Var

so1, so2, s :Integer ;

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 ;

BEGIN

Write ('-Nhap so thu nhât : ');

Readln(so1) ;

Write ('-Nhap so thu hai : ');

Readln(so2) ;

S := USCLN(so1, so2) ;

Writeln ;

Writeln ('+Uoc so chung lon nhât cua ', so1, ' va ', so2, ' = ', s) ;

```

Writeln ;
Write(' Bam phim <Enter> de ket thuc ');
Readln
END.

```

Ví dụ 2 : ta có thể "mô tả" cách tính $n!$ bằng một hàm có dạng như sau

```

Program G_thua ;
Var
  i :Integer ;k :real ;
Function giai_thua(m :Integer) :real ;
Begin
  i := 0 ;
  k := 1 ;
  while i <= m do
    begin
      i := i+1 ;
      k := k*i
    end ;
  giai_thua := k ;
End ;
BEGIN
  Write('-Nhap N = ');
  Readln(n) ;
  Writeln('+Giai thua cua ', n, ' = ', gt(n) :8 :0) ;
  Writeln ;
  Write(' Bam phim <Enter> de ket thuc ');
  Readln
End.

```

2.2. 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 ;
```

Để kết thúc ta xét một ví dụ có tính tổng hợp :

Ví dụ 3 :

Xét một khai báo :

type

```
mang = array[1..45] of integer ;
```

lập một chương trình thực hiện các công việc sau :

a) Viết một hàm để đếm số phần tử của mảng B có giá trị bằng một số M cho trước ; hàm này được khai báo như sau :

```
function DEMB(B : mang ; M : integer ; var Minindex : integer) : integer ;
```

trong đó : M là số cần so sánh ; hàm DEMB trả về số phần tử trong B có giá trị bằng M.

b) Chương trình chính thực hiện các công việc sau :

– Nhập các phần tử của một dãy số có 45 phần tử nguyên.

– Nhập một số nguyên từ bàn phím.

– Hiển thị lên màn hình số phần tử trong dãy trên có giá trị bằng số nguyên vừa nhập vào và chỉ số của phần tử đầu tiên bằng số nguyên đó ; nếu như không có phần tử nào trong B có giá trị bằng M thì Minindex = -1.

Sau đây là chương trình :

```
program dem ;
  type
    mang = array[1..45] of integer ;
  var
    a : mang ; i, k, min : integer ;
function DEMB(B : mang ; M : integer ; var Minindex : integer) : integer ;
  var
    i, tg ; cs : integer ;
  begin
    tg := 0 ; cs := -1 ;
    for i := 1 to 45 do if B[i] = m then
      begin
        tg := tg + 1 ;
        if cs = -1 then Minindex := i ;
      end ;
    DEMB := tg ;
  end ;
begin
  write('Doc vao mot so :') ; readln(k)
  for i := 1 to 45 do read(a[i]) ;
  writeln('So lan xuat hien gia tri ', k, ' la : ', DEMB(a, k, min)) ;
  write('chi so nho nhat la : ', min) ;
end.
```

3. THỦ TỤC (PROCEDURE)

Thủ tục có dạng :

procedure tên_thu_tuc(danh sách các tham số) ;

$\left. \begin{array}{l} - \\ \dots \\ - \end{array} \right\} \text{phân khai báo}$

Begin

$\left. \begin{array}{l} - \\ \dots \\ - \end{array} \right\} \text{các lệnh của thủ tục}$

End ;

trong đó *tên_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 phẩy ' ; ', trong một nhóm các tham số có cùng kiểu cách nhau bởi dấu phẩy ' , ', 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 vao(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 cần 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) ;
```

Để kết thúc mục này ta xét một ví dụ sau :

Ví dụ 5 :

Giả sử có khai báo :

```
type  
  mang = array[1..50] of real ;
```

hãy viết một chương trình thực hiện các công việc sau :

a) Lập một thủ tục cho phép sắp xếp một dãy số thực theo thứ tự tăng dần ;
thủ tục được khai báo như sau :

```
sap_xep(var b : mang) ;
```

b) Chương trình chính thực hiện các công việc :

- Đọc từ bàn phím một dãy số bao gồm 50 số thực ;
- Dùng thủ tục vừa viết ở trên sắp xếp lại dãy đó theo thứ tự tăng dần.
- Hiển thị dãy số trước và sau khi sắp xếp.

Sau đây là lời giải :

```
program sx ;  
type  
  mang = array[1..50] of real ;  
var  
  a : mang ;
```

```

    i : integer ;
    procedure sap_xep(var b : mang) ;
        var i, j : integer ; tg : real ;
    begin
        for i := 1 to 49 do
            for j := i+1 to 50 do
                if b[i] > b[j] then
                    begin
                        tg := b[i] ; b[i] := b[j] ; b[j] := tg ;
                    end ;
            end ;
        end ;
    begin
        for i := 1 to 50 do read(a[i]) ;
        writeln('Day vua doc vao la :') ;
        for i := 1 to 50 do write(a[i] :6 :2) ;
        sap_xep(a) ;
        writeln('Day vua sap xep la :') ;
        for i := 1 to 50 do write(a[i] :6 :2) ;
        readln
    end.

```

4. 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 các chương trình con thường không có lệnh read(), readln() vì các 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 6 :

```
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(' Không có nghiệm thực ' ) ;
end ;
begin
    delta := b*b - 4*a*c ;
    if delta >= 0 then delta_khong_am
    else delta_am ;
end ;
BEGIN
    readln(x, y, z) ;
    cach_giai(x, y, z) ;
    readln ;
END.
```

Nhận xét : Ta thấy trong chương trình trên có tất cả ba chương trình con, cơ chế hoạt động như sau : chương trình chính gọi chương trình con *cach_giai*, đến lượt chương trình con *cach_giai* lại gọi các chương trình con của nó là *delta_am* hay *delta_khong_am* tùy thuộc vào các giá trị của biệt thức *delta*.

Sau đây là nguyên tắc hoạt động của cơ chế "gọi" : chương trình con được ở lệnh thứ *n*, sau khi chương trình con thực hiện xong nhiệm vụ của nó, điều khiển chương trình lại quay về lệnh thứ *n+1*.

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* (lệnh thứ *n*) 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 tùy 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, hệ thống trở lại lệnh sau lệnh gọi *cach_giai* (lệnh thứ *n+1*), gặp lệnh *readln*, lệnh này dừng màn hình cho xem kết quả, để kết thúc gõ phím *enter*.

5. 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.

```

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 tùy 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 :

– Các tham số cùng kiểu được khai báo trong cùng nhóm, cách nhau bởi một dấu phẩy.

– Các nhóm tham số hình thức cách nhau bởi dấu chấm phẩy.

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 :

- Truyền bằng trị thông qua tham trị.
- Truyền bằng biến thông qua tham biến.

Sau đây chúng ta lần lượt khảo sát hai cơ chế này :

1) Truyền bằng tham 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).

2) Truyền bằng tham 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 *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 với chương trình trên ta thấy : ở lần gọi đầu : $so_sanh(n1, n2, l)$ 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 $so_sanh(a, b, \text{max} : \text{real})?$

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 : Về cách truyền tham số

```
program cach_truyen ;  
  var  
    x, y : integer ;  
  procedure vi_du(x1 : integer ; var x2 : 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.

5.1. 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.

Chú ý :

- Sau khi chương trình con được thực hiện xong, mọi biến cục bộ (kể cả tham trị) đều bị xoá bỏ trong bộ nhớ.

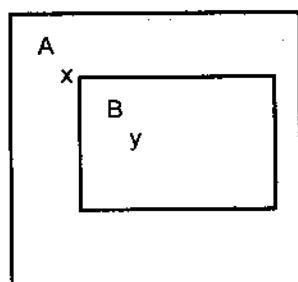
- Khi cần truyền tham số một cấu trúc dữ liệu lớn (mảng chẳng hạn) cho một chương trình con chúng ta nên sử dụng kỹ thuật theo biến, vì nếu truyền theo trị ta sẽ tốn thêm bộ nhớ để tạo bản sao và thời gian cần thiết để tạo bản sao đó.

5.2. 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 (nhân, 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 được nhận biết ở những nơi nào? Ví dụ ta có khối A chứa khối B và x là biến của A còn y là biến của B lúc đó :



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 khối 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.

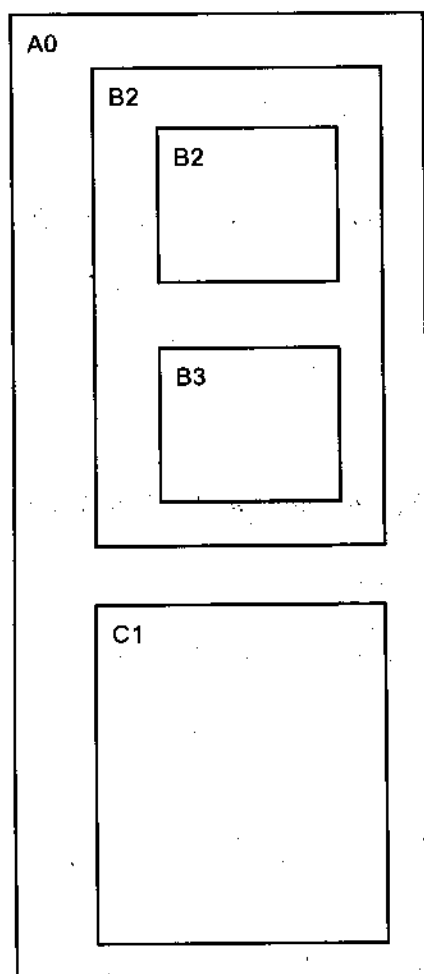
Sau đây ta xét một vài trường hợp cụ thể để làm rõ vấn đề này :

5.2.1. 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 lại có thể khai báo nhiều chương trình con khác ; vậy đối với một chương trình con 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.

Ví dụ 6.8 :



```

program A0 ;
  var
  ...
  procedure B1 ;
    var x, y : integer ;
    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 của 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 :

- Từ A0 có thể gọi B1.
- Từ C1 có thể gọi B1.
- Từ 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...

5.2.2. Vấn đề tầm vực của biến ; biến toàn cục, biến cục bộ, biến không cục bộ

Trở lại với ví dụ bên 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.

p2 ;

writeln(x) ;

END.

Khi chạy VD2 giá trị 1 sẽ được in ra màn hình.

Bạn đọc tự giải thích.

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 không cục bộ trong thân chương trình con gọi là hiệu ứng lè ; hiệu ứng lè làm chương trình kém trong sáng ; khiến ta khó quản lí các biến và làm cho chương trình chạy sai, nhất là khi ta dùng biến không cục bộ làm biến 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? Bạn đọc tự giải thích.

6. ĐỆ QUY

Chương trình con đệ quy là một chương trình mà trong thân nó lại gọi đến chính nó, PASCAL cho phép xây dựng chương trình con đệ quy.

Việc sử dụng kỹ thuật đệ quy khi viết chương trình cho ta một cách nhìn sáng sủa hơn đối với bài toán cần giải quyết.

Để bắt đầu chúng ta xét một vài thí dụ :

Ví dụ 12 : bài toán tính giai thừa.

Chúng ta có thể định nghĩa $n!$ theo hai cách như sau :

Cách đệ quy :

$$n! = \begin{cases} 1 & \text{nếu } n = 0 \\ n.(n-1) & \text{nếu } n > 0 \end{cases}$$

Cách lặp :

$$n! = 1.2.3.4...n$$

Hàm đệ quy có dạng như sau :

```

function giai_thua(n : integer) : integer ;
begin
    if n = 0 then giai_thua := 1
    else giai_thua := n*giai_thua(n-1) ;
end ;

```

Hàm tính giai thừa viết bằng kỹ thuật lặp đề nghị bạn đọc xem như bài tập.

Ví dụ 13 : Dãy Fibonacci theo định nghĩa mà các phần tử của nó được tính theo công thức truy hồi sau :

$$F_1 = F_2 = 1$$

$$F_n = F_{n-2} + F_{n-1}$$

Hàm đệ quy để tính số Fibonacci được viết như sau :

```
function fibo(n : integer) : integer ;
begin
    if n < 3 then fibo := 1
    else fibo := fibo(n-2) + fibo(n-1) ;
end ;
```

Ví dụ 14 : Tháp Hà Nội

Bài toán này phát biểu như sau : có n đĩa kích thước từ nhỏ tới lớn được sắp xếp sao cho đĩa nhỏ nằm trên đĩa lớn ; người ta cần chuyển n đĩa này từ vị trí ban đầu đến vị trí khác theo 3 nguyên tắc :

1. Mỗi lần chỉ chuyển một đĩa ;
2. Trong quá trình chuyển không được để đĩa lớn lên trên đĩa bé ;
3. Được phép dùng một vị trí trung gian thứ 3 ;

Để giải quyết bài toán trên ta dùng phương pháp quy nạp :

1. Với 1 đĩa bài toán được giải quyết ngay.
2. Giả sử bài toán đã được giải quyết cho trường hợp (n-1) đĩa, khi đó lời giải với n đĩa như sau :

- Chuyển (n-1) đĩa từ vị trí ban đầu đến vị trí trung gian ;
- Chuyển đĩa lớn nhất từ vị trí ban đầu đến vị trí tập kết ;
- Chuyển (n-1) đĩa từ vị trí trung gian đến vị trí tập kết.

Như vậy thủ tục đệ quy có dạng như sau :

```
procedure cd(n, t1, t2, t3 : integer) ; {chuyển n đĩa từ t1 sang t2 nhờ t3}
begin
    if n = 1 then write(t1, '→', t2)
    else
        begin
```

```

        cd(n-1, t1, t3, t2) ;
        cd(1, t1, t2, t3) ;
        cd(n-1, t2, t1, t3) ;
    end ;
end ;

```

Bài tập

- Viết một hàm cho phép xác định một số nguyên có phải là số nguyên tố hay không.
- Viết một thủ tục cho phép tính tổng của hai ma trận có cùng kích thước.
- Xét chương trình sau :

```

program kt ;
    var a : integer ;
    function f(var x : integer) : integer ;
        begin
            x := x+1 ;
            f := x ;
        end ;
    begin
        a := 5 ;
        write(f(a)*f(a)) ;
        readln ;
    end.

```

Khi chạy chương trình kết quả hiện lên là 42(1) ; hãy giả thích tại sao ?

- Hãy viết một hàm tính số Fibonacci bằng lệnh lặp.
- Viết một thủ tục cho phép sắp xếp một dãy số theo thứ tự tăng dần.
- Xét chương trình

```

program td ;
    var
        tu, mau, d : integer ;
    function USCLN(var a, b : integer) : integer ;

```

begin

while a <> b do

if a > b then a := a - b

else b := b - a ;

end ;

begin

write(' Doc vao tu và mau :') ; readln(tu, mau) ;

d := USCLN(tu, mau) ;

write('Dang toi gian của phân thức đã cho là :', tu div d, '/', mau div d) ;

readln ;

end.

Khi chạy thu được kết quả là 1/1 ; Hãy giải thích tại sao?

7. Viết chương trình con dạng thủ tục cho phép tính giá trị lớn nhất của một dãy số.
8. Viết một chương trình con dạng hàm cho phép trích tên trong xâu họ và tên ; chẳng hạn nếu họ và tên là 'Tran van Chau' thì hàm này trả lại xâu 'Chau' ;
9. Giả sử ta có định nghĩa sau : theo định nghĩa một câu gọi là câu *chuẩn* nếu câu đó là một xâu (string) bắt đầu và kết thúc bằng kí tự khác kí tự trống ; các từ trong câu phân biệt với nhau bởi chỉ *một* kí tự trống. Hãy viết một chương trình con dạng hàm để chuẩn hoá một câu bất kì.
10. Viết một hàm để đếm số lần cụm kí tự "kt" xuất hiện trong một xâu.
11. Viết một hàm để kiểm tra xem cụm kí tự "kh" có xuất hiện trong một xâu hay không ?

TÀI LIỆU THAM KHẢO

1. *Tô Văn Nam*, Tin học đại cương, NXB KH&KT (2002).
2. *Hồ Sĩ Đàm, Nguyễn Thanh Tùng*, Tin học 10, NXBGD (1995).
3. *Wirth, N.* Cấu trúc dữ liệu + giải thuật = chương trình, NXB Đại học và THCN (1991).
4. *Jensen, K. and Wirth, N.* PASCAL, User Manual and Report, Springer – Verlag (1974).
5. *Wirth, N.*, The Programming Language PASCAL, Acta informatica, 1, No 1 (1974).
6. *Le Beux, P. et Tavernier*, PASCAL par la pratique, Cybex (1999).
7. *Boussard et Mahl*, Programmation avancée, Erolles (1998).
8. *Nguyễn Thanh Thủy,...*, Ngôn ngữ lập trình C, NXB Khoa học và kỹ thuật (1999).
9. *Nyhoff, L. và Leedstma, S.*, Lập trình nâng cao bằng PASCAL với cấu trúc dữ liệu, NXB Đà Nẵng (1997).
10. *Nguyễn Đình Tế,...*, Giáo trình Pascal, NXB GD (1999).
11. *Quách Tuấn Ngọc*, Ngôn ngữ lập trình Pascal, NXB giáo dục (1996).
12. Các bài giảng của đồng nghiệp tại Trường ĐHBK HN

MỤC LỤC

Trang

Phần một **ĐẠI CƯƠNG**

<i>Chương 1. NHỮNG KHÁI NIỆM CƠ BẢN VỀ TIN HỌC</i>	5
1. Khái niệm về thông tin và xử lí thông tin	5
2. Ứng dụng của công nghệ thông tin	7
3. Biểu diễn thông tin (hệ đếm, mã)	7
4. Giải thuật, biểu diễn giải thuật, chương trình	16
5. Ngôn ngữ lập trình, chương trình dịch	18
<i>Chương 2. MÁY TÍNH ĐIỆN TỬ</i>	23
1. Nguyên lí Von Neumann	23
2. Kiến trúc máy tính	26
3. Mạng máy tính	30
<i>Chương 3. PHẦN MỀM MÁY TÍNH</i>	39
1. Hệ điều hành	39
2. Phần mềm ứng dụng	52
3. Phần mềm tiện ích	53
4. Các ngôn ngữ lập trình	53
<i>Chương 4. SOẠN THẢO VĂN BẢN</i>	55
1. Khởi động trình soạn thảo	55
2. Các lệnh dịch chuyển con trỏ màn hình	58
3. Các lệnh chèn, xóa	58

Phần hai **LẬP TRÌNH BẰNG NGÔN NGỮ PASCAL**

<i>Chương 5. MỞ ĐẦU</i>	59
1. Các thành phần cơ bản	62
2. Các kiểu dữ liệu chuẩn	64
3. Hằng	64
4. Biểu thức	66
5. Cấu trúc của một chương trình Pascal	66

Chương 6. CÁC LỆNH CỦA TURBO PASCAL

1. Lệnh gán	72
2. Các thủ tục vào ra dữ liệu	72
3. Câu lệnh điều kiện	78
4. Các câu lệnh lặp	82

Chương 7. CÁC KIỂU DỮ LIỆU MỞ RỘNG

1. Kiểu dữ liệu miền con (khoảng con)	93
2. Dữ liệu kiểu mảng	94
3. Kiểu xâu kí tự	100

Chương 8. BẢN GHI (RECORD)

1. Khái niệm Record (bản ghi)	106
2. Khai báo Record	106
3. Mảng các bản ghi	112
4. Phong cách viết chương trình	116

Chương 9. TẬP TIN (FILE)

1. Khái niệm	120
2. Khai báo tệp	121
3. Các thao tác trên file	122
4. Tệp văn bản	128
5. Các tệp tin thiết bị của DOS	132

Chương 10. CHƯƠNG TRÌNH CON

1. Khái niệm	136
2. Hàm (Function)	136
3. Thủ tục (Procedure)	141
4. Lưu ý về phong cách lập trình	143
5. Cách truyền tham số	145
6. Đề quy	153
Tài liệu tham khảo	157

Chịu trách nhiệm xuất bản :

Chủ tịch HĐQT kiêm Tổng Giám đốc NGÔ TRẦN ÁI

Phó Tổng Giám đốc kiêm Tổng biên tập VŨ DƯƠNG THỤY

Biên tập lần đầu :

TRẦN VĂN THẮNG

Biên tập tái bản :

BÙI MINH HIỀN

Trình bày bìa :

TÀO HUYỀN

Sửa bản in :

MINH HUYỀN

Chế bản :

CHÚC LINH

GIÁO TRÌNH NHẬP MÔN TIN HỌC

Mã số: 6H155T5-DAI

In 2.000 bản, khổ 16 x 24cm tại Công ty Cổ phần in Bắc Giang. Số in: 10
Số xuất bản: 21/243-05. In xong và nộp lưu chiểu tháng 2 năm 2005.



CÔNG TY CỔ PHẦN SÁCH ĐẠI HỌC - DẠY NGHỀ
HEVOBCO

Địa chỉ : 25 Hàn Thuyên, Hà Nội



NGÔI SÀO BẠCH KIM
 CHẤT LƯỢNG
 QUỐC TẾ

TÌM ĐỌC GIÁO TRÌNH DÙNG CHO CÁC TRƯỞNG
ĐÀO TẠO HỆ TRUNG HỌC CHUYÊN NGHIỆP - DẠY NGHỀ
CỦA NHÀ XUẤT BẢN GIÁO DỤC
(NGÀNH ĐIỆN TỬ - TIN HỌC)

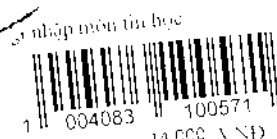
- | | |
|---|---------------------------------------|
| 1. Linh kiện điện tử và ứng dụng | TS. Nguyễn Viết Nguyên |
| 2. Điện tử dân dụng | ThS. Nguyễn Thanh Trà |
| 3. Điện tử công suất | Trần Trọng Minh |
| 4. Mạch điện tử | TS. Đặng Văn Chuyết |
| 5. Kỹ thuật số | TS. Nguyễn Viết Nguyên |
| 7. Kỹ thuật điều khiển | .Vũ Quang Hải |
| 8. Kỹ thuật xung - số | TS. Lương Ngọc Hải |
| 9. Điện tử công nghiệp | Vũ Quang Hải |
| 10. Toán ứng dụng trong tin học | PGS. TS. Bùi Minh Trí |
| 11. Nhập môn tin học | Tô Văn Nam |
| 12. Cấu trúc máy vi tính và vi xử lý | Lê Hải Sâm - Phạm Thanh Liêm |
| 13. Hệ các chương trình ứng dụng
(Window, Word, Excel) | GVC. Trần Viết Thường - Tô Văn Nam |
| 14. Cơ sở dữ liệu | Tô Văn Nam |
| 15. Lập trình C | GVC Tiêu Kim Cương |
| 16. Cấu trúc dữ liệu và giải thuật | PGS.TS. Đỗ Xuân Lôi |
| 17. Cài đặt và điều hành mạng | TS. Nguyễn Vũ Sơn |
| 18. Phân tích thiết kế hệ thống | GVC. Tô Văn Nam |
| 19. ACCESS và ứng dụng | TS. Huỳnh Quyết Thắng |
| 20. Sử dụng Corel Draw | Nguyễn Phú Quảng |
| 21. Bảo trì và quản lý phòng máy tính | Phạm Thanh Liêm |
| 22. Kinh tế và quản trị doanh nghiệp
(kinh tế và TCQLSX) | TS. Ngô Xuân Bình - TS. Hoàng Văn Hải |

Bạn đọc có thể tìm mua tại các Công ty Sách - Thiết bị trường học ở các địa phương hoặc các Cửa hàng sách của Nhà xuất bản Giáo dục:

Tại Hà Nội : 25 Hàn Thuyên, 81 Trần Hưng Đạo, 187 Giảng Võ,
 23 Tràng Tiền.

Tại Đà Nẵng : 15 Nguyễn Chí Thanh.

Tại Thành phố Hồ Chí Minh : 104 Mai Thị Lựu, Qu



Giá: 14.000đ