

Giáo trình

Nhập môn Tin học

LỜI NÓI ĐẦU

Nhập môn Tin học là môn học quan trọng trong chương trình giáo dục đại cương. Tại hầu hết các trường Đại học và Cao đẳng ở nước ta hiện nay, môn học này là bắt buộc với sinh viên và nội dung ngày càng được nâng cao cả về lý thuyết và thực hành.

Cuốn Nhập môn Tin học này dành cho sinh viên hệ Đại học, Cao đẳng chuyên ngành Tin học và các ngành khác của trường Đại học Điện lực. Giáo trình không chỉ phù hợp cho người mới bắt đầu mà còn phù hợp cho những người cần tham khảo. Cấu trúc của giáo trình gồm các nội dung sau:

Chương 1: Các vấn đề cơ bản của Tin học

Chương 2: Sử dụng máy tính.

Chương 3: Giải thuật

Chương 4: Các yếu tố cơ bản của ngôn ngữ Pascal

Chương 5: Bước đầu xây dựng chương trình

Chương 6: Các câu lệnh có cấu trúc

Chương 7: Các kiểu dữ liệu có cấu trúc

Chương 8: Chương trình con

Khi biên soạn, chúng tôi đã tham khảo các giáo trình và tài liệu giảng dạy môn học này của một số trường đại học trong và ngoài nước để giáo trình vừa đạt yêu cầu cao về nội dung vừa thích hợp với đối tượng là sinh viên của trường Đại học Điện lực. Chúng tôi cũng đã nhận được sự đóng góp rất quý báu của GS Phạm Văn Ất, PGS Nguyễn Đình Hóa và một số đồng nghiệp khác.

Khi viết chúng tôi đã hết sức cố gắng để cuốn sách được hoàn chỉnh, song chắc chắn không tránh khỏi sai sót, vì vậy rất mong nhận được sự góp ý của độc giả.

Các tác giả

MỤC LỤC

LỜI NÓI ĐẦU	3
Chương 1:CÁC VẤN ĐỀ CƠ BẢN CỦA TIN HỌC	9
1.1. Thông tin	9
1.1.1 Thông tin là gì?.....	9
1.1.2. Mã hóa thông tin trên máy tính	10
1.1.3. Hệ đếm và biểu diễn số trong hệ đếm:	11
1.2. Kiến trúc chung hệ thống máy tính ^[2]	16
1.2.1. Bộ nhớ	16
1.2.2 Các thiết bị vào/ra.....	23
1.2.4. Quá trình thực hiện lệnh	27
1.3. Hệ điều hành (HĐH)	31
1.3.1. Khái niệm	31
1.3.2. Chức năng của hệ điều hành.....	31
1.4. Mạng máy tính (MMT)	34
1.4.1. Khái niệm	34
1.4.2. Phân loại mạng máy tính	34
1.5. Internet.....	36
1.5.1. Internet là gì?.....	36
1.5.2 Giao thức TCP/IP ^[2]	37
1.5.3. Các tài nguyên trên Internet.....	40
1.5.4. Các dịch vụ cơ bản trên Internet.....	40
1.5.5. Hệ thống tên miền:	41
1.5.6. Hệ thống định vị tài nguyên thống nhất URL (Uniform Resource Locator).....	42
1.5.7.Cấu trúc một mạng điển hình có nối với Internet.....	43
1.6. Một số vấn đề về tội phạm Tin học và đạo đức nghề nghiệp ^[2]	49
1.6.1 Tin tặc - một loại tội phạm kỹ thuật	49
1.6.2. Các tội phạm lạm dụng Internet vì những mục đích xấu.....	52
1.6.3. Sở hữu trí tuệ và bản quyền.....	52
1.6.4. Luật liên quan đến tội phạm tin học của Việt Nam	53
Chương 2:SỬ DỤNG MÁY TÍNH ^[2]	56
2.1. Hệ điều hành WINDOWS XP	56
2.1.1. Bắt đầu Windows XP	56
2.1.2. Một số khái niệm cơ bản trong Windows XP	57
2.1.3 Một số khái niệm cơ bản trên màn hình Windows XP.....	59
2.1.4 Thanh tác vụ của Windows XP	66
2.1.5 Thanh gọi chương trình nhanh (Quick Launch Bar)	66
2.1.6 khay hệ thống (System Tray).....	67

2.1.7 Sử dụng “Windows Explorer”	64
2.1.8 Sử dụng các dòng lệnh trong Windows (giống như DOS).....	68
2.2 Hệ điều hành LINUX	72
2.2.1 Giới thiệu về HĐH Linux	72
2.2.2 Linux - xu thế, giải pháp mới cho các hệ thống thông tin	72
2.2.3 Một số khái niệm cơ bản trong Linux	73
2.2.4 Môi trường đồ họa	75
Chương 3: THUẬT GIẢI	79
3.1. Khái niệm	79
3.2 Các đặc trưng của thuật giải	79
3.3 Các phương pháp biểu diễn thuật giải	80
3.3.1 Ngôn ngữ tự nhiên	80
3.3.2 Lưu đồ - sơ đồ khối	80
3.3.3 Mã giả	82
BÀI TẬP CHƯƠNG 3	84
Chương 4: CÁC YẾU TỐ CƠ SỞ CỦA NGÔN NGỮ PASCAL.....	85
4.1. Giới thiệu ngôn ngữ PASCAL	85
4.2. Các thành phần cơ bản của ngôn ngữ PASCAL	86
4.2.1 Bộ ký tự cơ bản.....	86
4.2.2 Từ khóa (key word)	86
4.2.3 Tên (identifier).....	86
4.2.4. Các dấu đặc biệt.....	87
4.3. Các kiểu dữ liệu đơn giản.....	87
4.3.1 Khái niệm	87
4.3.2. Phân loại các kiểu dữ liệu trong Turbo Pascal	88
4.3.3 Kiểu số nguyên	89
4.3.4 Kiểu số thực.....	90
4.3.5 Kiểu ký tự (CHAR)	92
4.3.6 Kiểu LÔGIC (BOOLEAN)	94
4.3.7. Một số kiểu dữ liệu đơn giản do người lập trình định nghĩa	95
4.4. Hằng, biến và biểu thức.....	98
4.4.1 Khái niệm về biến và hằng	98
4.4.2 Khai báo biến.....	98
4.4.3 Khai báo hằng.....	98
4.4.4 Biểu thức.....	99
Chương 5: BƯỚC ĐẦU XÂY DỰNG CHƯƠNG TRÌNH	101
5.1. Cấu trúc chung một chương trình Pascal.....	101

5.1.1 Chương trình Pascal	101
5.1.2. Phần tiêu đề chương trình.....	101
5.1.3. Phần khai báo	102
5.1.4. Phần thân chương trình.....	103
5.2. Câu lệnh trong chương trình Pascal	104
5.2.1 Phân loại câu lệnh.....	104
5.2.2. Lệnh gán	104
5.3. Các lệnh nhập, xuất dữ liệu	106
5.3.1 Lệnh xuất (in) dữ liệu ra màn hình	106
5.3.2 Lệnh nhập dữ liệu từ bàn phím.....	111
BÀI TẬP CHƯƠNG 5	115
 Chương 6:CÁC CÂU LỆNH CÓ CẤU TRÚC	117
6.1. Câu lệnh ghép (khối lệnh)	117
6.2. Các câu lệnh rẽ nhánh và lựa chọn.....	117
6.2.1. Lệnh rẽ nhánh IF	117
6.2.2 Câu lệnh lựa chọn CASE.....	119
6.3. Câu lệnh lặp xác định FOR	124
6.3.1. Ý nghĩa:	124
6.3.2 Câu lệnh FOR tiến (Dạng 1).....	124
6.3.3 Câu lệnh FOR lùi (Dạng 2)	125
6.4. Câu lệnh lặp không xác định WHILE và REPEAT	127
6.4.1 Ý nghĩa	127
6.4.2 Câu lệnh lặp không xác định kiểm tra điều kiện sau REPEAT	127
6.4.3 Câu lệnh lặp không xác định kiểm tra điều kiện trước WHILE	131
6.4.4. Một số câu lệnh kết thúc sớm vòng lặp hoặc chương trình.....	134
BÀI TẬP CHƯƠNG 6	137
 Chương 7:DỮ LIỆU CÓ CẤU TRÚC	139
7.1. Kiểu mảng	139
7.1.1 Khái niệm	139
7.1.2 Khai báo mảng một chiều.....	139
7.1.3. Khai báo mảng hai chiều	140
7.1.4. Các phép toán trên mảng	141
7.1.5. Nhập và in dữ liệu của mảng.....	142
7.1.6 Một số bài toán cơ bản về mảng.....	144
7.1.7. Một số ví dụ khác	147
7.2. Kiểu chuỗi (xâu) ký tự.....	149
7.2.1 Khái niệm	149
7.2.2. Khai báo xâu ký tự.....	149

7.2.3. Viết ra và đọc vào một xâu ký tự.....	150
7.2.4. Các phép toán trên xâu	151
7.2.5 Truy nhập vào từng phần tử của xâu	152
7.2.6 Các hàm xử lý xâu ký tự.....	153
7.2.7 Các thủ tục liên quan đến xâu.....	153
7.2.8 Các ví dụ về xâu	155
7.3. Kiểu bản ghi (Record)	157
7.3.1. Khái niệm	157
7.3.2 Khai báo kiểu bản ghi.....	157
7.3.3 Sử dụng bản ghi.....	158
7.3.4 Câu lệnh WITH	160
7.3.5 Mảng các bản ghi.....	161
7.3.6 Ví dụ về bản ghi	163
7.4. Kiểu tập hợp (Set of)	166
7.4.1. Khái niệm	166
7.4.2. Cú pháp.....	167
7.4.3. Một số tính chất.....	167
7.4.4. Các phép toán trên tập hợp	167
7.4.5. Viết và đọc dữ liệu kiểu tập hợp.....	168
7.5. Kiểu tệp (FILE)	170
7.5.1. Khái niệm	170
7.5.2. Cấu trúc và phân loại tệp	171
7.5.3. Tệp định kiểu.....	172
7.5.4. Tệp truy cập tuần tự.....	172
7.5.5. Mở tệp mới để ghi dữ liệu	173
7.5.6. Mở tệp đã tồn tại để đọc dữ liệu	174
7.5.7. Tệp truy cập trực tiếp.....	177
7.5.8. Các thao tác khác với tệp.....	179
7.5.9. Tệp văn bản	185
BÀI TẬP CHƯƠNG 7	195
Chương 8: CHƯƠNG TRÌNH CON.....	199
8.1. Các khái niệm.....	199
8.1.1. Khái niệm về chương trình con	199
8.1.2. Một số khái niệm	199
8.1.3. Sử dụng chương trình con	200
8.2. Thủ tục và hàm	202
8.2.1. Thủ tục (procedure)	202
8.2.2. Hàm (function)	203
8.3. Biến toàn cục và biến địa phương	204

8.4. Truyền tham số cho chương trình con.....	207
8.4.1. Vai trò của tham số.....	207
8.4.2. Truyền theo tham trị.....	207
8.4.3. Truyền theo tham biến.....	208
8.5. Tính đệ qui của chương trình con.....	210
8.5.1. Khái niệm về đệ qui.....	210
8.5.2. Cách dùng đệ qui.....	211
BÀI TẬP CHƯƠNG 8.....	216
PHỤ LỤC.....	218
TÀI LIỆU THAM KHẢO.....	221

Chương 1

CÁC VẤN ĐỀ CƠ BẢN CỦA TIN HỌC

1.1. Thông tin

1.1.1 Thông tin là gì?

Khái niệm thông tin (information) được sử dụng thường ngày. Con người có nhu cầu đọc báo, nghe đài, xem phim, video, đi tham quan, du lịch, tham khảo ý kiến người khác, ... để nhận được thêm thông tin mới. Thông tin mang lại cho con người sự hiểu biết, nhận thức tốt hơn về những đối tượng trong đời sống xã hội, trong thiên nhiên, ... giúp cho họ thực hiện hợp lý công việc cần làm để đạt tới mục đích một cách tốt nhất.



Khi tiếp nhận được thông tin, con người thường phải xử lý nó để tạo ra những thông tin mới, có ích hơn, từ đó có những phản ứng nhất định. Người tài xế chăm chú quan sát người, xe cộ đi lại trên đường, độ tốt xấu mặt đường, tính năng kỹ thuật cũng như vị trí của chiếc xe để quyết định, cần tăng tốc độ hay hãm phanh, cần rẽ trái hay sang phải... nhằm đảm bảo an toàn tối đa cho chuyến xe đi.

Thông tin có thể được phát sinh, được lưu trữ, được truyền, được tìm kiếm, được sao chép, được xử lý, nhân bản. Thông tin cũng có thể biến dạng, sai lệch hoặc bị phá hủy.

Mỗi tế bào sinh dục của những cá thể sinh vật mang thông tin di truyền quyết định những đặc trưng phát triển của cá thể đó. Gặp môi trường không thuận lợi, các thông tin di truyền đó có thể bị biến dạng, sai lệch dẫn đến sự hình thành những cá thể dị dạng. Ngược lại, bằng những tác động tốt của di truyền học chọn giống, ta có thể cấy hoặc làm thay đổi các thông tin di truyền theo hướng có lợi cho con người.

Thông tin được thể hiện dưới nhiều dạng thức khác nhau như sóng ánh sáng, sóng âm, điện từ, các ký hiệu viết trên giấy hoặc khắc trên gỗ, trên đá, trên các tấm kim loại ... Về nguyên tắc, bất kỳ cấu trúc vật chất nào hoặc bất kỳ dòng năng lượng nào cũng có thể mang thông tin. Chúng được gọi là những vật (giá) mang tin. Dữ liệu (data) là biểu diễn của thông tin và được thể hiện bằng các tín hiệu (signal) vật lý.

Thông tin chứa đựng ý nghĩa, còn dữ liệu là các dữ kiện không có cấu trúc và không có ý nghĩa rõ ràng nếu nó không được tổ chức và xử lý. Cùng một thông tin, có thể được biểu diễn bằng những dữ liệu khác nhau. Cùng biểu diễn một đơn vị, nhưng trong chữ số thập phân ta dùng ký hiệu 1, còn trong hệ đếm La Mã lại dùng ký hiệu I. Mỗi dữ liệu lại có thể được thể hiện bằng những ký hiệu vật lý khác nhau. Cũng là gập đầu, đối với nhiều dân tộc trên thế giới thì đó là tín hiệu thể hiện sự đồng tình; nhưng ngược lại, đối với người Hy Lạp, gập đầu để biểu lộ sự bất đồng. Cùng là ký hiệu I nhưng trong tiếng Anh có nghĩa là đại từ nhân xưng ngôi thứ nhất (tôi) còn trong toán học lại là chữ số La Mã có giá trị là 1. Mỗi tín hiệu có thể dùng để thể hiện các thông tin khác nhau. Chẳng hạn như trong máy tính điện tử (MTĐT), nhóm 8 chữ số 01000001, nếu là số sẽ thể hiện số 65, còn nếu là chữ sẽ là chữ “A”.

Như vậy, *Thông tin* là một khái niệm trừu tượng, tồn tại khách quan, có thể nhớ trong đối tượng, biến đổi trong đối tượng và áp dụng để điều khiển đối tượng. Thông tin làm tăng thêm hiểu biết của con người, là nguồn gốc của nhận thức. Thông tin về một đối tượng chính là một dữ kiện về đối tượng đó, chúng giúp ta nhận biết và hiểu được đối tượng.

Dữ liệu (data) là hình thức thể hiện của thông tin trong mục đích lưu trữ và xử lý nhất định. Khái niệm dữ liệu xuất hiện cùng với việc xử lý thông tin bằng máy tính. Vì thế trong nhiều tài liệu người ta định nghĩa dữ liệu là đối tượng xử lý của máy tính. Thông tin luôn mang một ý nghĩa xác định nhưng hình thức thể hiện của thông tin rõ ràng mang tính quy ước.

Tri thức (knowledge) có ý nghĩa khái quát hơn thông tin. Những nhận thức thu nhận được từ nhiều thông tin trong một lĩnh vực cụ thể nào đó, có tính hướng mục đích mới trở thành tri thức. Như vậy tri thức là mục đích của nhận thức trên cơ sở tiếp nhận thông tin. Quá trình xử lý thông tin chính là quá trình nhận thức để có tri thức.

1.1.2. Mã hóa thông tin trên máy tính

1.1.2.1. Mã hóa thông tin

Thông tin được chia làm hai loại là thông tin liên tục và thông tin không liên tục (thông tin rời rạc). *Thông tin liên tục* đặc trưng cho các đại lượng mà số lượng các giá trị có thể tiếp nhận được là vô hạn như độ dài dịch chuyển cơ học, điện áp, Còn *thông tin rời rạc* đặc trưng cho các đại lượng mà số lượng các giá trị có thể kể ra được như số trang sách của một cuốn sách, tên sinh viên trong lớp, địa chỉ của hộ gia đình trên phố, ...

Thông tin rời rạc có thể biểu diễn thông tin qua các bộ ký hiệu (mã ký tự) mà ta gọi là bảng chữ. Giả sử, ta có tập đối tượng X cần biểu diễn. Để làm điều này, ta chọn một tập hữu hạn A các ký hiệu làm bảng chữ mà mỗi ký hiệu là một chữ. Chúng ta sẽ gọi mỗi dãy hữu hạn các chữ là một từ trên A . Ví dụ nếu A là tập các chữ số thì mỗi từ chính là một số (cho bằng một dãy số). Mã hoá các thông tin rời rạc của một tập trên một bảng chữ A chính là cách gán cho mỗi phần tử x thuộc X , một từ y trên A . Phép gán mã phải đảm bảo tính chất: mã của hai đối tượng khác nhau phải khác nhau. Tính chất này đảm bảo khi biết mã có thể tìm được đối tượng tương ứng. Quá trình gán mã được gọi là phép lập mã. Quá trình ngược được gọi là phép giải mã. Ví dụ, nếu X là tập các thí sinh, chọn A là tập các chữ số thì mã của một thí sinh có thể lấy là số báo danh của thí sinh đó. Số báo danh phải cho phép chỉ định duy nhất một thí sinh.

Dữ liệu là hình thức biểu diễn thông tin với mục đích xử lý thông tin. Vậy mã hoá chính là con đường chuyển từ thông tin thành dữ liệu. Các thông tin dưới dạng số, văn bản, âm thanh, hình ảnh, ... đều phải chuyển dưới dạng mã phù hợp để máy tính có thể làm việc được.

1.1.1.2. Mã hóa nhị phân

Mã hóa trên bảng chữ cái ký hiệu được gọi là mã hóa nhị phân. Trong tin học, mã hóa nhị phân được sử dụng rất rộng rãi. Một trong nhiều lý do đó là cấu trúc bên trong máy tính bao gồm rất nhiều các mạch điện phức tạp. Tại mỗi thời điểm, một mạch điện chỉ nhận một trong hai trạng thái hoặc đóng hoặc mở. Thêm vào đó trong hệ nhị phân chỉ gồm hai chữ số 0 và 1 (tương ứng với bit 0 và bit 1), ta có bảng chữ nhị phân.

Trong mã hóa nhị phân, mỗi chữ số nhị phân (binary digit) mang một lượng tin nào đó về một trạng thái cần biểu diễn và được xem là một đơn vị thông tin. Ta gọi đơn vị đo thông tin đó là bit. Bit là chữ viết tắt của **BI**nary di**gi**T. Trong tin học, người ta thường sử dụng các đơn vị đo thông tin lớn hơn sau:

Tên gọi	Ký hiệu	Giá trị
Byte	B	8 bit
KiloByte	KB	2^{10} B = 1024 Bytes
MegaByte	MB	2^{20} B
GigaByte	GB	2^{30} B
TetraByte	TB	2^{40} B

Bảng 1.1. Bảng đơn vị đo thông tin

Như vậy, để có thể biểu diễn được thông tin trong máy tính thì cần biểu diễn các trạng thái hay chính là trạng thái các mạch điện trong máy tính. Người ta đã lựa chọn các bit 0/1 để biểu diễn thông tin trong máy tính. Mỗi một chuỗi bit 0/1 cho biết trạng thái một mạch điện, độ dài của chuỗi bit phụ thuộc vào độ phức tạp của mạch điện, chẳng hạn như sau:

Nếu sử dụng 1 bit thì ta biểu diễn được 2 (2^1) trạng thái là 0 và 1

Nếu sử dụng 2 bit thì ta biểu diễn được 4 (2^2) trạng thái là 00, 01, 10, 11

Nếu sử dụng 3 bit thì ta biểu diễn được 8 (2^3) trạng thái là 000, 001, 010, 011

100, 101, 110, 111

.....

Nếu sử dụng n bit thì ta biểu diễn được 2^n trạng thái.

Ngược lại, bất cứ một tập n trạng thái sẽ chỉ cần dùng không quá $\log_2 n + 1$ bit để tạo ra các mã đủ phân biệt n trạng thái.

1.1.3. Hệ đếm và biểu diễn số trong hệ đếm:

1.1.3.1. Hệ đếm

Hệ đếm là tập hợp các ký hiệu và qui tắc sử dụng tập ký hiệu đó để biểu diễn và xác định các giá trị các số. Mỗi hệ đếm có một số ký số hữu hạn. Tổng số ký số của mỗi hệ đếm được gọi là cơ số (base hay radix), ký hiệu là b.

Trong ngành toán - tin học hiện nay phổ biến 4 hệ đếm như sau:

Hệ đếm	Cơ số	Ký số và trị tuyệt đối
Hệ nhị phân	2	0, 1
Hệ bát phân	8	0, 1, 2, 3, 4, 5, 6, 7
Hệ thập phân	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hệ thập lục phân	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Hệ đếm phổ biến hiện nay là hệ đếm thập phân.

1.1.3.2. Biểu diễn số trong các hệ đếm

* Hệ đếm thập phân (decimal system)

Hệ đếm thập phân hay hệ đếm cơ số 10 là một trong những phát minh của người Ả rập cổ, bao gồm 10 ký số theo ký hiệu sau:

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

Qui tắc tính giá trị của hệ đếm này là mỗi đơn vị ở một hàng bất kỳ có giá trị bằng 10 đơn vị của hàng kế cận bên phải. Ở đây $b = 10$. Bất kỳ số nguyên dương trong hệ thập phân có thể thể hiện như là một tổng các chuỗi các ký số thập phân nhân cho 10 lũy thừa, trong đó số mũ lũy thừa được tăng thêm 1 đơn vị kể từ số mũ lũy thừa phía bên phải nó. Số mũ lũy thừa của hàng đơn vị trong hệ thập phân là 0.

Ví dụ 1.1: Số 2105 có thể được thể hiện như sau:

$$\begin{aligned} 2105 &= 2 \times 10^3 + 1 \times 10^2 + 6 \times 10^1 + 5 \times 10^0 \\ &= 2 \times 1000 + 1 \times 100 + 6 \times 10 + 5 \times 1 \end{aligned}$$

Thể hiện như trên gọi là ký hiệu mở rộng của số nguyên vì:

$$2105 = 2000 + 100 + 60 + 5$$

Như vậy, trong số 2105: ký số 5 trong số nguyên đại diện cho giá trị 5 đơn vị (1s), ký số 6 đại diện cho giá trị 6 chục (10s), ký số 1 đại diện cho giá trị 1 trăm (100s) và ký số 2 đại diện cho giá trị 2 nghìn (1000s). Nghĩa là, số lũy thừa của 10 tăng dần 1 đơn vị từ trái sang phải tương ứng với vị trí ký hiệu số,

$$10^0 = 1 \quad 10^1 = 10 \quad 10^2 = 100 \quad 10^3 = 1000 \quad 10^4 = 10000 \dots$$

Mỗi ký số ở thứ tự khác nhau trong số sẽ có giá trị khác nhau, ta gọi là giá trị vị trí (place value).

Phân phân số trong hệ thập phân sau dấu chấm phân cách (theo qui ước của Mỹ) thể hiện trong ký hiệu mở rộng bởi 10 lũy thừa âm tính từ phải sang trái kể từ dấu chấm phân cách:

$$\begin{aligned} \text{Ví dụ 1.2: } 2105.37 &= 2 \times 10^3 + 1 \times 10^2 + 6 \times 10^1 + 5 \times 10^0 + 3 \times 10^{-1} + 7 \times 10^{-2} \\ &= 2 \times 1000 + 1 \times 100 + 6 \times 10 + 5 \times 1 + 3 \times \frac{1}{10} + 7 \times \frac{1}{100} \\ &= 2000 + 100 + 60 + 5 + \frac{3}{10} + \frac{7}{100} \end{aligned}$$

Tổng quát, hệ đếm cơ số b ($b \geq 2$, b là số nguyên dương) mang tính chất sau

- Có b ký số để thể hiện giá trị số. Ký số nhỏ nhất là 0 và lớn nhất là $b-1$.
 - Giá trị vị trí thứ n trong một số của hệ đếm bằng cơ số b lũy thừa n : b
- Số $N(b)$ trong hệ đếm cơ số (b) thể hiện :

$$N_{(b)} = a_n a_{n-1} a_{n-2} \dots a_1 a_0 a_{-1} a_{-2} \dots a_{-m}$$

trong đó, số $N(b)$ có $n+1$ ký số ở phần nguyên và m ký số ở phần thập phân, sẽ có giá trị là :

$$N_{(b)} = a_n x b^n + a_{n-1} x b^{n-1} + a_{n-2} x b^{n-2} \dots a_1 x b^1 + a_0 x b^0 + a_{-1} x b^{-1} + a_{-2} x b^{-2} \dots a_{-m} x b^{-m}$$

Hay:
$$N_{(b)} = \sum_{i=-m}^n a_i b^i$$

* Hệ đếm nhị phân (binary number system)

Với $b=2$, chúng ta có hệ đếm nhị phân. Đây là hệ đếm đơn giản nhất với 2 chữ số là 0 và 1. Mỗi chữ số nhị phân gọi là BIT (viết tắt từ chữ BInary digiT). Hệ nhị phân tương ứng với 2 trạng thái của các linh kiện điện tử trong máy tính chỉ có: đóng (có điện hay có dòng điện đi qua) ký hiệu là 1 và tắt (không có điện hay không có dòng điện đi qua) ký hiệu là 0. Vì hệ nhị phân chỉ có 2 trị số là 0 và 1, nên khi muốn diễn tả một số lớn hơn, hoặc các ký tự phức tạp hơn thì cần kết hợp nhiều bit với nhau.

Ta có thể chuyển đổi hệ nhị phân theo hệ thập phân quen thuộc.

Ví dụ 1.3: Số $1110101_{(2)}$ sẽ tương đương với giá trị thập phân là: 117

Số nhị phân	1	1	1	0	1	0	1
Vị trí	6	5	4	3	2	1	0
Trị vị trí	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Hệ 10 là	64	32	16	8	4	2	1

Như vậy: $1110101_{(2)} = 1 \times 64 + 1 \times 32 + 1 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 117_{(10)}$

* Hệ đếm bát phân (Octal Number System)

Với $b=8=2^3$, ta được hệ đếm bát phân, là hệ đếm gồm tập hợp các ký số: 0, 1, 2, 3, 4, 5, 6, 7. Nếu trong hệ nhị phân, trị vị trí là lũy thừa của 2 thì trong hệ bát phân, trị vị trí là lũy thừa của 8.

Ví dụ 1.4: $165_{(8)} = 1 \times 8^2 + 6 \times 8^1 + 5 \times 8^0 = 117_{(10)}$

* Hệ đếm thập lục phân (hexa-decimal number system)

Hệ đếm thập lục phân là hệ cơ số $b = 16 = 2^4$, tương đương với tập 4 chữ số nhị phân (4 bit). Khi thể hiện ở dạng hexa-decimal, ta có 16 ký tự gồm 10 chữ số từ 0 đến 9, và 6 chữ in A, B, C, D, E, F để biểu diễn các giá trị số tương ứng là 10, 11, 12, 13, 14, 15. Với hệ thập lục phân, trị vị trí là lũy thừa của 16.

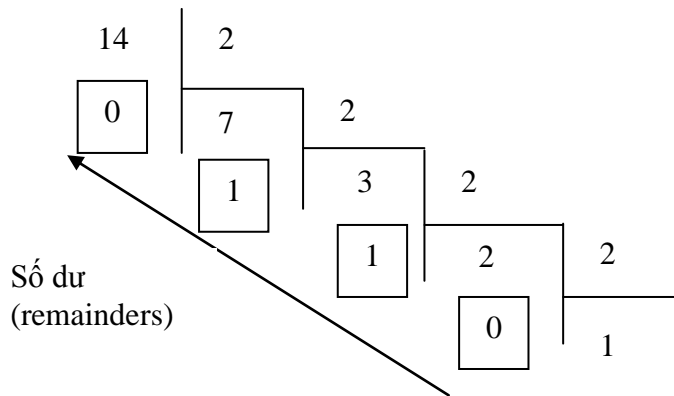
Ví dụ 1.5: $75_{(16)} = 7 \times 16^1 + 5 \times 16^0 = 117_{(10)}$
 $A2B_{(16)} = 10 \times 16^2 + 2 \times 16^1 + 11 \times 16^0 = 2603_{(10)}$

* Đổi một số nguyên từ hệ thập phân sang hệ b

Tổng quát: Lấy số nguyên thập phân $N(10)$ lần lượt chia cho b cho đến khi thương số bằng 0. Kết quả số chuyển đổi $N(b)$ là các dư số trong phép chia viết ra theo thứ tự ngược lại.

Ví dụ 1.6:

Số 14 trong hệ thập phân sẽ được biểu diễn như thế nào trong hệ nhị phân ($b=2$). Dùng phép chia 2 liên tiếp ta có các số dư như sau:



Ta được: $14_{(10)} = 01101_{(2)}$

1.1.3.3. Số học nhị phân:

Trong số học nhị phân chúng ta cũng có 4 phép toán cơ bản như trong số học thập phân là cộng, trừ, nhân và chia.

Qui tắc của 2 phép tính cơ bản cộng và nhân:

X	Y	X + Y	X * Y
0	0	0	0
0	1	1	0
1	0	1	0
1	1	10	1

Ghi chú: Với phép cộng trong hệ nhị phân, $1 + 1 = 10$, số 10 (đọc là một - không) chính là số 2 tương đương trong hệ thập phân. Viết 10 có thể hiểu là viết 0 nhớ 1. Một cách tổng quát, khi cộng 2 hay nhiều chữ số nếu giá trị tổng lớn hơn cơ số b thì ta viết phần lẻ và nhớ phần lớn hơn sang bên trái cạnh nó.

Ví dụ 1.7: Cộng 2 số $0101 + 1100 = ?$

$$\begin{array}{r}
 0110 \quad \text{Tương ứng với số 6 trong hệ 10} \\
 + 0011 \quad \text{Tương ứng với số 3 trong hệ 10} \\
 \hline
 1001 \quad \text{Tương ứng với số 9 trong hệ 10}
 \end{array}$$

Ví dụ 1.8: Nhân 2 số $0110 \times 0011 = ?$

$$\begin{array}{rcl}
 * & \begin{array}{r} 0110 \\ 0011 \\ \hline 0110 \end{array} & \begin{array}{l} \text{Tương ứng với số 6 trong hệ 10} \\ \text{Tương ứng với số 3 trong hệ 10} \end{array} \\
 + & \begin{array}{r} 0110 \\ 0000 \\ 0000 \\ \hline 0010010 \end{array} & \text{Tương ứng với số 18 trong hệ 10}
 \end{array}$$

Phép trừ và phép chia là các phép toán đặc biệt của phép cộng và phép nhân.

Ví dụ 1.9: Trừ hai số

$$\begin{array}{rcl}
 & 0110 & \text{Tương ứng với số 6 trong hệ 10} \\
 - & 0011 & \text{Tương ứng với số 3 trong hệ 10} \\
 \hline
 & 0011 & \text{Tương ứng với số 3 trong hệ 10}
 \end{array}$$

Chú ý: $0 - 1 = -1$ (viết 1 và mượn 1 ở hàng bên trái)

Ví dụ 1.10: Chia hai số

$ \begin{array}{r} - \begin{array}{r} 110 \\ 10 \\ \hline 010 \\ 10 \\ \hline 00 \end{array} \end{array} $	$ \begin{array}{r} 10 \\ \hline 11 \end{array} $	<p>Trong đó:</p> <p>110 tương ứng với số 6 trong hệ 10</p> <p>10 tương ứng với số 2 trong hệ 10</p> <p>11 tương ứng với số 3 trong hệ 10</p>
--	--	--

Qui tắc 1: Khi nhân một số nhị phân với 2^n ta thêm n số 0 vào bên phải số nhị phân đó.

Ví dụ 1.11: $1101 \times 2^2 = 110100$

Qui tắc 2: Khi chia một số nguyên nhị phân cho 2^n ta đặt dấu chấm ngăn ở vị trí n chữ số bên trái kể từ số cuối của số nguyên đó.

Ví dụ 1.12: $10010010 : 2^2 = 100100.10$

Bài đọc thêm: Ai là người đưa ra thuật ngữ “tin học” lần đầu tiên [2]

Môn “Máy tính điện tử” được đưa vào dạy trong chương trình đại học đầu tiên ở Việt Nam vào năm 1962. Người dạy môn học này đầu tiên là thầy Nguyễn Công Thuý, lúc đó là giảng viên Khoa Toán – Cơ thuộc Đại học Tổng hợp Hà Nội (nay là Đại học Khoa học Tự nhiên thuộc Đại học Quốc gia Hà Nội). Một trong những học sinh học môn học này thời đó là thầy Nguyễn Xuân My, người phụ trách lớp chuyên tin của Đại học Tổng hợp nhiều năm và cũng từng là trưởng đoàn các đội dự thi Olympic quốc tế tin học phổ thông nhiều năm của nước ta. Lúc bấy giờ nội dung của môn học rất đơn giản: một ít kiến thức về nguyên lý máy tính và một ít kiến thức về lập trình trên một ngôn ngữ quy ước có hình thức tương tự như hợp ngữ (assembly). Sinh

thời, cố Bộ trưởng Bộ Đại học và Trung học Chuyên nghiệp Tạ Quang Bửu là người rất quan tâm đến những lĩnh vực mới và thường khuyến khích các cán bộ trẻ đi vào các lĩnh vực đó. Ông là người đề nghị thầy Thủy dịch cuốn “Introduction à l’Informatique” vào năm 1971. Đây là một cuốn sách phổ biến khoa học của Pháp viết rất hay và đơn giản về các vấn đề về tin học. Thời đó các thuật ngữ khoa học dùng ở Đại học thường được chú ý Việt hóa. Thầy Thủy có trao đổi với các đồng nghiệp trong đó có thầy My và cho rằng nên dịch Informatique” là “Tin học”. Sợ rằng nếu dịch là Tin học nhiều người không hiểu sẽ không đọc nên thầy Thủy quyết định để nguyên từ Informatique. Cuốn “Mở đầu về Informatique” đã ra đời như vậy và được xuất bản thành tài liệu lưu hành nội bộ và có trong thư viện của Đại học Tổng hợp Hà Nội vào đầu những năm 70.

1.2. Kiến trúc chung hệ thống máy tính ^[2]

Hơn nửa thế kỷ qua, nhờ những tiến bộ khoa học kỹ thuật, tính năng của MTĐT đã được hoàn thiện không ngừng. Mặc dầu vậy, các nguyên lý hoạt động, cũng như cấu trúc cơ bản của MTĐT vẫn chưa có gì thay đổi đáng kể. Kiến trúc tổng quát của các hệ MTĐT đều bao gồm các khối chức năng chủ yếu sau đây:



- **Bộ nhớ (memory):** là nơi lưu trữ các dữ liệu. Bộ nhớ được phân cấp thành 2 loại. Bộ nhớ trong là bộ nhớ làm việc trong quá trình xử lý. Máy tính xử lý trực tiếp các thông tin trong bộ nhớ trong. Bộ nhớ ngoài có tốc độ làm việc chậm. Bù lại, thông tin trên bộ nhớ ngoài có thể lưu trữ lâu dài mà không cần nguồn nuôi. Tuy nhiên máy tính không thể xử lý trực tiếp các thông tin trên bộ nhớ ngoài mà trước khi xử lý phải chuyển chúng vào bộ nhớ trong.
- **Bộ số học và logic (Arithmetic Logic Unit - ALU)** là nơi thực hiện các xử lý như thực hiện các phép tính số học hay logic.
- **Bộ điều khiển (Control Unit)** là đơn vị chức năng đảm bảo cho máy tính thực hiện đúng theo chương trình đã định. Bộ điều khiển phải điều phối, đồng bộ hoá tất cả các thiết bị của máy để phục vụ yêu cầu xử lý do chương trình quy định. Do bộ điều khiển và bộ số học logic phải phối hợp hết sức chặt chẽ trong suốt quá trình thực hiện chương trình nên kể từ các máy tính thế hệ thứ 3, người ta thường chế tạo chúng trong một khối chức năng chung gọi là **bộ xử lý trung tâm (Central Processing Unit - CPU)**.
- **Thiết bị ngoại vi (Peripheral Device)** là các thiết bị giúp máy tính giao tiếp với môi trường bên ngoài kể cả với người sử dụng.

1.2.1. Bộ nhớ

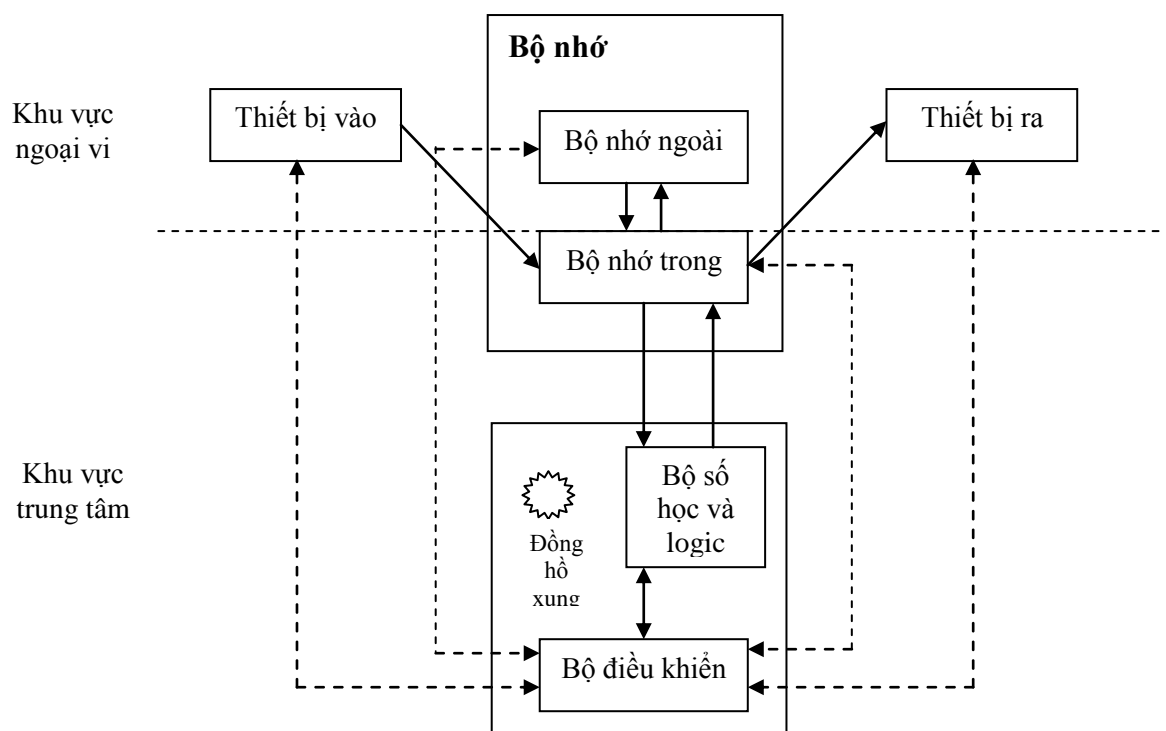
Bộ nhớ là thiết bị dùng để lưu trữ dữ liệu và chương trình. Tính năng của bộ nhớ đánh giá qua các đặc trưng chính sau:

- Thời gian truy cập (access time) là khoảng thời gian cần thiết kể từ khi phát tín hiệu điều khiển đọc/ghi đến khi việc đọc/ghi hoàn thành. Tốc độ truy cập là một yếu tố quyết định tốc độ chung của máy tính.

- Sức chứa bộ nhớ (memory capacity) chỉ khối lượng dữ liệu mà bộ nhớ có thể lưu trữ đồng thời.
- Độ tin cậy: đo bằng khoảng thời gian trung bình giữa hai lần lỗi.

1.2.1.1 Bộ nhớ trong (BNT)

Bộ nhớ trong là loại bộ nhớ có thời gian truy cập nhỏ. Nó được dùng để ghi chương trình và dữ liệu trong thời gian xử lý.



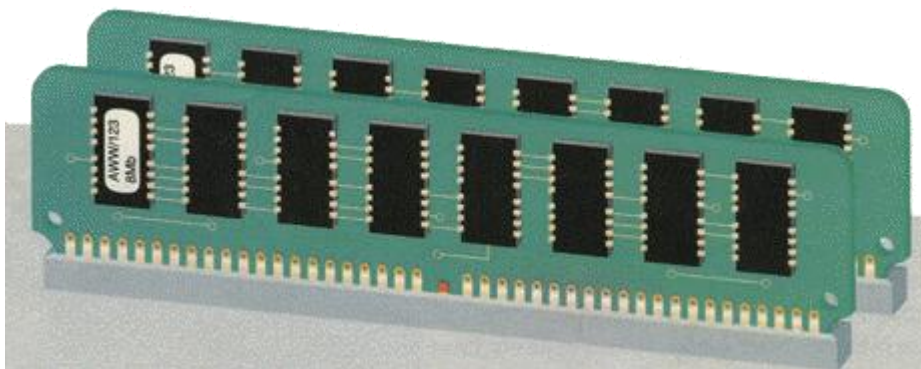
Hình 1.1: Sơ đồ cấu trúc logic của MTĐT.

Các mũi tên liền nét là đường chuyển dữ liệu, các đường đứt nét thể hiện các kênh điều khiển

BNT được cấu tạo từ các phần tử vật lý có hai trạng thái đối lập. Một trạng thái dùng để thể hiện bit 0 còn trạng thái kia thể hiện bit 1. Có nhiều kỹ thuật chế tạo các phần tử có hai trạng thái. Trong thập kỷ 60, 70 người ta thường dùng bộ nhớ từ tính như xuyên ferit hoặc màng mỏng từ và ghi nhớ các bit bằng chiều của từ thông. Sau này người ta dùng các bộ nhớ bán dẫn là các mạch bán dẫn điều khiển được có hai trạng thái đóng/mở để thể hiện các bit. Cần phân biệt thiết bị vật lý (ví dụ mạch điện) là phần cứng cố định còn trạng thái của thiết bị thì không cố định, dễ dàng thay đổi (ví dụ bằng cách đóng/mở mạch điện) để thể hiện các bit. Nhờ tiến bộ của công nghệ vi điện tử, các bộ nhớ bán dẫn có thể được chế tạo theo qui mô công nghiệp, giảm được giá thành. Thành phần chủ yếu của bộ nhớ MTĐT hiện đại là mạch tích hợp. Hiện nay, một vi mạch nhỏ cỡ vài cm^2 có sức nhớ tới vài trăm MB. Bộ nhớ bán dẫn được chia thành hai loại:

- **Bộ nhớ RAM (Random Access Memory)**

RAM là loại bộ nhớ có thể ghi và đọc dữ liệu. Chính vì vậy nó còn có một tên gọi khác là RWM (Read Write Memory). Dữ liệu phải nuôi bằng nguồn điện nên chúng sẽ bị xóa khi mất nguồn. Bản thân cụm từ “Random Access Memory” có nghĩa là bộ nhớ truy nhập ngẫu nhiên với ý nghĩa là thời gian truy nhập đến bất kỳ ô nhớ nào (ngẫu nhiên) cũng như nhau, và việc sao lưu hay xóa bỏ dữ liệu trên RAM phụ thuộc vào cách thức và trạng thái làm việc của hệ thống lúc đó.



Hình 1.2 : Bộ nhớ RAM

- **Bộ nhớ ROM (Read Only Memory)**

ROM là loại bộ nhớ cố định, chỉ cho phép người sử dụng đọc dữ liệu ra nhưng không cho phép ghi vào. Dữ liệu được ghi vào ROM trong lúc chế tạo hoặc bằng phương tiện chuyên dụng. Loại ROM có thể ghi lại được bằng phương tiện chuyên dụng gọi là EPROM (Erasable Programmable ROM).

Dữ liệu ghi trong ROM không cần nguồn nuôi. ROM thường dùng để lưu trữ các chương trình điều hành cơ sở của máy tính. Khi bật máy tính các chương trình này có thể thực hiện được ngay mà không cần nạp từ một nơi nào đó vào bộ nhớ trong.

Tổ chức bộ nhớ trong (BNT)

Ta có thể hình dung BNT như dãy liên tiếp các ô nhớ được đánh số. Chỉ số của một ô nhớ gọi là địa chỉ của ô nhớ đó. Địa chỉ được đánh số lần lượt từ 0, 1, 2, . . . Mỗi ô nhớ gồm nhiều ngăn, mỗi ngăn dùng để lưu một bit. Độ dài của ô nhớ là khác nhau theo từng loại máy. Trước đây khi máy tính dùng chủ yếu với mục đích khoa học kỹ thuật thì độ dài ô nhớ khá lớn. Ví dụ IBM/360 dùng ô nhớ 32 bit, chiếc máy tính đầu tiên dùng ở Việt nam, máy Minsk-22 của Liên xô dùng ở Việt Nam những năm 60 dùng ô nhớ 37 bit. . . Phần lớn các máy tính ngày nay dùng ô nhớ có độ dài 8 bit (một byte). Byte là đơn vị **Địa chỉ** thông tin thuận lợi cho xử lý dữ liệu chữ vì có thể chứa vừa đủ mã một chữ. Để thể hiện các dữ liệu dài hơn như số người ta sử dụng nhiều byte kế tiếp nhau ví dụ để lưu trữ một số nguyên lớn người ta có thể dùng 4 ô nhớ 1 byte kế nhau.

Hoạt động cơ sở của máy tính là thực hiện

một lệnh. Trong một lệnh, máy tính có thể xử lý cả một nhóm bit trong nhiều byte kế tiếp nhau. Dãy các bit nhớ dài nhất với tư cách một đơn vị dữ liệu mà CPU có thể xử lý trong một lệnh cơ bản gọi là một *từ máy* (memory word).

0	7	6	5	4	3	2	1	0
1								
2								
....								
n-1								

Hình 1.3: Hình ảnh địa chỉ hoá bộ nhớ n byte

Mỗi MTĐT có độ dài từ máy (số lượng các bit nhớ) xác định, thường là 8, 16, 32... bits (tương ứng một, hai, bốn... byte). Ví dụ từ máy của máy vi tính dùng bộ xử lý Intel 80286 là 16 bit, còn từ máy vi tính dùng bộ xử lý Pentium của Intel là 32 bit, từ máy của máy dùng bộ xử lý Alpha hay bộ vi xử lý Itanium mà Intel là 64 bit. Từ máy càng dài thể hiện mức song song hoá trong xử lý càng cao. Địa chỉ từ máy là địa chỉ byte đầu tiên của từ máy đó. Như vậy, mỗi ô nhớ có hai đặc trưng:

- *Địa chỉ là giá trị bằng số*, chỉ thứ tự của vị trí ô nhớ trong BNT. Địa chỉ của mỗi ô nhớ là cố định.
- *Nội dung là giá trị số dạng mã nhị phân*, được lưu trữ bằng các trạng thái vật lý trong ô nhớ. Nội dung ô nhớ có thể thay đổi.

Do mỗi ô nhớ có địa chỉ riêng của nó, nên có thể truy nhập tới dữ liệu trong từng ô nhớ không phụ thuộc vào các ô nhớ khác. Chính vì thế, BNT còn được gọi là bộ nhớ truy nhập trực tiếp. Dữ liệu truyền giữa CPU và bộ nhớ mỗi lần thường là một byte hay một từ.

Đọc/ ghi

Khi đọc bộ nhớ, nội dung chứa trong ô nhớ không thay đổi (tương tự như khi ta đọc sách thì chữ viết trong trang sách đó vẫn còn nguyên). Khi ghi vào bộ nhớ thì nội dung cũ có trong bộ nhớ đó bị xoá để lưu nội dung mới (tương tự như viết lên bảng, khi viết thì xoá nội dung trước đó). Để đọc/ghi với bộ nhớ trong, đầu tiên CPU gửi địa chỉ của vùng nhớ tới một mạch gọi là bộ giải mã địa chỉ, sau đó gửi một tín hiệu điều khiển tới kích hoạt bộ giải mã địa chỉ. Kết quả là bộ giải mã địa chỉ mở mạch nối trực tiếp với mạch lưu trạng thái của ô nhớ tương ứng rồi sao chép nội dung ra một vùng nhớ phụ nếu thao tác là đọc hoặc nội dung của vùng nhớ phụ được sao vào ô nhớ nếu thao tác là ghi. Vùng nhớ phụ này có tên là các thanh ghi - register mà ta sẽ nói kỹ hơn trong phần mô tả CPU. Do cơ chế địa chỉ hoá và phần nào đó do giá thành nên bộ nhớ trong thường có dung lượng không lớn lắm.

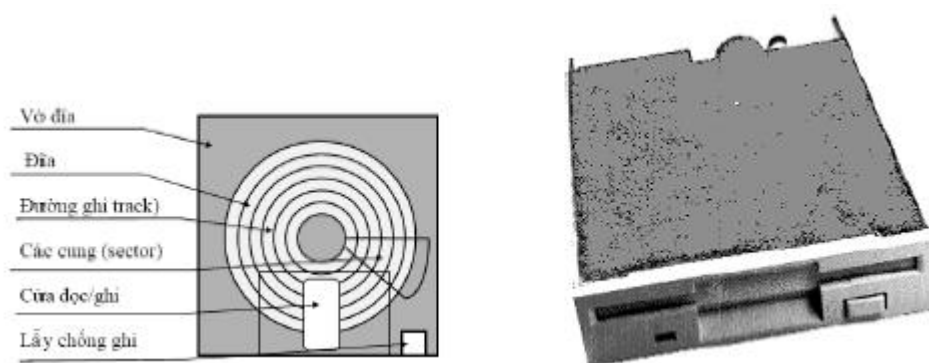
1.2.1.2 Bộ nhớ ngoài (BNN)

RAM chỉ dùng cho việc ghi dữ liệu khi xử lý, không dùng được khi không còn nguồn nuôi. Vì vậy, đối với các dữ liệu cần lưu giữ lâu dài, không thể để trên RAM được. Mặt khác tuy tốc độ truy nhập trên RAM là nhanh, nhưng dung lượng nhớ của nó nhỏ không cho phép lưu trữ lượng thông tin lớn. Để có thể lưu trữ thông tin lâu dài với khối lượng lớn, ta phải sử dụng bộ

nhớ ngoài. BNN thường làm bằng các vật liệu từ. Với BNN, tuy tốc độ khai thác chậm hơn, nhưng chi phí lưu trữ rẻ hơn và giữ được thông tin lâu dài không phụ thuộc vào nguồn. Có nhiều loại BNN. Cho đến nay chỉ còn sử dụng thông dụng một số loại là đĩa từ, băng từ và gần đây ta còn dùng đĩa quang (đọc bằng tia laser).

Ta mô tả một số loại BNN thông dụng

- **Đĩa mềm** (floppy disk) là một đĩa hình tròn làm bằng nhựa tổng hợp, trên đó có phủ lớp vật liệu có từ tính. Đĩa mềm được chứa trong vỏ bọc hình vuông để bảo vệ khỏi bụi và chỉ để mở ở hai chỗ, một chỗ cho đầu đọc/ghi tiếp xúc được với đĩa. Một chỗ gọi là lẫy bảo vệ đĩa mà khi ta cài lại thì việc ghi vào đĩa không thực hiện được. Biện pháp này giúp người sử dụng có thể bảo vệ thông tin ghi trên đĩa chống ghi nhầm hay xóa mất thông tin đang có trên đĩa. Dữ liệu được ghi trên một hoặc hai mặt của đĩa theo các đường tròn đồng tâm mà ta gọi là đường ghi (track). Để tiện định vị các dữ liệu trên các đường ghi, đường ghi được chia thành các cung (sector). Các cung được đánh số liên tiếp từ 0, 1, 2, . . .



Hình 1.4: Đĩa mềm (trái) và ổ đĩa mềm (phải).

Dữ liệu được định vị trên đĩa theo địa chỉ, được xác định thông qua tên đĩa, mặt dưới hay trên của đĩa, chỉ số đường ghi, chỉ số cung. Việc đọc/ghi thông tin với đĩa thực hiện theo các đơn vị vài cung gọi là liên cung (cluster) trên một đường ghi chứ không thực hiện theo từng byte. Thiết bị đọc ghi đĩa (mà sau đây ta sẽ gọi là ổ đĩa) hoạt động giống với bộ phận quay đĩa của máy hát. Ở tâm đĩa mềm có lỗ để bộ phận quay gắn vào đó và quay đĩa. Đầu từ đọc/ghi mặt đĩa qua cửa đọc/ghi. Khi có yêu cầu đọc/ghi, CPU gửi tín hiệu điều khiển đến ổ đĩa. Khi đó bộ phận quay gắn vào đĩa và quay đĩa còn đầu từ được di chuyển theo phương bán kính đến đường ghi cần thiết. Thời gian truy nhập đối đĩa bao gồm cả thời gian đặt đầu từ vào vùng đĩa chứa thông tin và cả thời gian đọc/ghi.

Có nhiều loại đĩa mềm có dung lượng và kích cỡ khác nhau. Đĩa mềm thông dụng nhất hiện nay là loại có đường kính 3.5 inch với sức chứa 1.44 MB.

- **Đĩa cứng** (Hard disk) thường là một bộ đĩa gồm nhiều đĩa xếp thành chồng, đồng trục. Các đĩa này là các đĩa hợp kim có phủ vật liệu từ trên mặt để ghi thông tin. Mỗi đĩa cũng quy định các đường ghi, các cung tương tự như đĩa mềm. Do có nhiều đĩa nên các đường ghi trên các đĩa có cùng một bán kính tạo nên một mặt trụ (cylinder). Khi nói tới *số trụ của đĩa cứng* ta hiểu đó chính là số thứ tự của đường ghi trên đĩa.

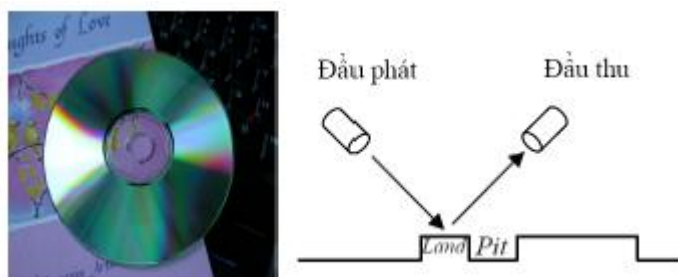


Hình 1.5: Bên trong và bên ngoài một đĩa cứng

Mỗi mặt đĩa có đầu đọc/ghi (head) riêng. Chúng được cố kết thành một chùm như một cái lược và di chuyển đồng thời. Khi có yêu cầu, bộ đọc/ghi chuyển đến một trụ và một đầu đọc được chọn để đọc/ghi trên mặt tương ứng. Có 3 tính năng thường được kể đến khi xem xét đĩa cứng. Đó là:

- *Sức chứa hay dung lượng tính theo MB hay GB.* Mật độ ghi trên đĩa cứng cũng cao hơn nhiều so với đĩa mềm. Những đĩa cứng ngày nay rất gọn và có thể có sức chứa tới nhiều chục GB. Các đĩa cứng dùng cho máy vi tính thông thường có sức chứa khoảng 20-40 GB. Tới năm 2002 đã xuất hiện các đĩa cứng có sức chứa tới 200 GB.
- *Thời gian truy nhập.* Thời gian này phụ thuộc cả tốc độ quay của đĩa và thời gian cần thiết để di chuyển đầu từ tới vị trí cần thiết. Tốc độ quay phổ biến hiện nay là 5400 vòng/phút hoặc 7200 vòng /phút. Cũng đã có các đĩa cứng có tốc độ quay đạt tới 10.000 vòng/phút. Thời gian trung bình để đặt được đầu từ vào vị trí đọc mất khoảng 10 ms. Trong các đĩa cứng hiện đại để cải thiện tốc độ giao tiếp người ta còn xây dựng những bộ nhớ đệm (cache) cho phép nạp trước những dữ liệu sẽ đọc vào bộ nhớ đệm. Sau đó CPU sẽ lấy dữ liệu từ bộ nhớ đệm để giảm thời gian chờ đọc từ đĩa cứng.
- *Độ tin cậy* thường tính bằng khoảng thời gian trung bình giữa hai lần lỗi. Bộ đĩa và bộ phận đọc/ghi được lắp đặt chung trong một hộp kín để tránh bụi. Khi hoạt động do tốc độ quay của đĩa rất nhanh nên dòng không khí tạo một lớp đệm tách đầu từ khỏi mặt đĩa, không làm cho đĩa cứng bị xước do những tiếp xúc cơ học như đối với đĩa mềm. Do vậy tuổi thọ đĩa cứng rất dài. Khoảng thời gian trung bình có một lỗi của đĩa cứng lên tới hàng chục nghìn giờ so với vài giờ của đĩa mềm.

• **Đĩa quang** hay đĩa compact (viết tắt là CD) làm bằng polycarbonate, có phủ một lớp phim nhôm có tính phản xạ và một lớp bảo vệ. Dữ liệu ghi trên đĩa bằng các vết lõm (trong tiếng Anh gọi là pit) và các vùng phản xạ hay còn gọi là vùng nổi (trong tiếng Anh gọi là land).



Hình 1.6: Đĩa quang và nguyên tắc đọc tín hiệu trên đĩa quang

Đĩa CD được đọc bằng tia laser, không có sự tiếp xúc cơ học nào giữa đầu đọc và mặt đĩa. Khi đọc, đầu đọc chiếu tia laser công suất thấp lên đĩa và phân tích tín hiệu phản hồi để nhận biết các điểm lõm và vùng nổi. Khi gặp các điểm lõm, tín hiệu phản hồi sẽ bị tán xạ. Còn khi gặp các vùng nổi, tia laser sẽ bị phản xạ lại.

Để ghi đĩa người ta dùng phương pháp ép khuôn nếu sản xuất hàng loạt. Các đĩa quang tạo dạng theo kiểu này không thể ghi lại được. Cũng vì thế mà ta thường gọi chúng là các đĩa CDRom (Compact Disk Read Only Memory). Một vài loại đĩa quang có nguyên tắc ghi khác như dùng chính tia laser để ghi, thậm chí có những loại có thể ghi được nhiều lần. Các ứng dụng đầu tiên của đĩa compact là các đĩa nhạc và đĩa hình. Tốc độ truy cập trên đĩa CD không nhanh bằng đĩa cứng. Người ta thường đo tốc độ đọc đĩa CD theo tỉ số tốc độ so với các đầu đọc đĩa nhạc số tiêu chuẩn. Ví dụ đầu đọc tốc độ 48 (kí hiệu là 48X) có tốc độ đọc nhanh gấp 48 lần đầu đọc đĩa nhạc (tốc độ khoảng 150 KB/s). Đĩa CD có sức chứa rất lớn. Các đĩa thông dụng hiện nay có sức chứa khoảng 650 MB. Các nhà sản xuất đã đưa ra một chuẩn đĩa khác là đĩa Video số (DVD) có sức chứa gấp 7 lần các đĩa CD ROM hiện nay. Với đĩa này có thể ghi một bộ phim kéo dài nhiều giờ. Người ta đã chuẩn bị đưa ra thị trường các đĩa quang có sức chứa tới 100 GB. Đối với đĩa DVD tốc độ 1X không chỉ là 150 KB/s như đĩa nhạc mà tính theo tốc độ đĩa hình là 1350 KB/s.

Đọc thông tin từ các loại đĩa (đĩa mềm, đĩa cứng hay đĩa quang) đều bắt đầu từ việc tính toán trước thông tin nằm ở vùng nào của đĩa (head, sector, track) sau đó mới đưa đầu từ vào đúng đường ghi (track, cylinder), sau đó kích hoạt việc đọc đối với đầu từ tương ứng (head) sau đó đợi cung (sector) tương ứng đi qua. Chính vì có thể tính được trước vùng chứa dữ liệu nên có thể đặt đầu đọc vào đúng vùng cần thiết mà các loại đĩa nói trên cũng được xem là bộ nhớ truy nhập trực tiếp (direct access) tương tự như bộ nhớ trong. Điều này khác hẳn với bộ nhớ kiểu băng từ mà ta nêu dưới đây.

• Thẻ nhớ

Vài năm gần đây xuất hiện một loại bộ nhớ ngoài mới (có dung lượng từ vài chục MB đến 1 GB). Loại bộ nhớ này dùng các mạch bán dẫn và công nghệ flash. Các bộ nhớ này rất gọn, khi cần dùng thì cắm vào cổng giao tiếp của máy tính (cổng USB). Giá thành của bộ nhớ này rất rẻ.



Hình 1.7: Một vài loại thẻ nhớ thông dụng và thẻ nhớ dùng cho máy ảnh số

• **Băng từ** được sử dụng rất rộng rãi trong thập kỷ 60 và 70. Ưu điểm chính của băng từ là giá rất rẻ. Tuy nhiên chế độ đọc/ghi với băng từ là tuần tự (sequential access). Nếu như để đọc một vùng nào đó trên đĩa người ta có thể tính toán để đặt chính xác đầu từ vào vùng đĩa cần đọc thì với băng từ phải duyệt tuần tự. Thời gian truy nhập đối với băng từ mất nhiều phút. Chính vì vậy

mà hiện nay băng từ vẫn còn được sử dụng với mục đích lưu trữ lâu dài, còn để lưu trữ với mục đích khai thác thường xuyên thì người ta không dùng băng từ mà dùng đĩa từ.

Chú ý rằng CPU chỉ xử lý trực tiếp các dữ liệu được lưu trữ ở BNT. Do vậy, trước khi được xử lý, các thông tin ở BNN cần được truyền vào BNT. Vì vậy BNT còn được gọi là bộ nhớ chính, BNN gọi là bộ nhớ phụ.

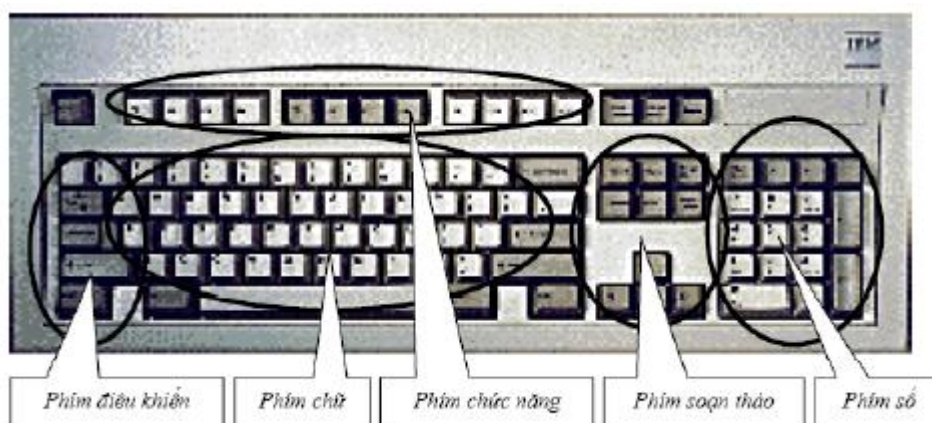
1.2.2 Các thiết bị vào/ra

Các thiết bị vào/ra (Input/Output Device) dùng để trao đổi dữ liệu giữa môi trường bên ngoài và MTĐT. Cụ thể hơn, qua các thiết bị vào có chức năng chuyển dữ liệu từ bên ngoài vào bộ nhớ trong còn các thiết bị ra dùng để chuyển thông tin từ bộ nhớ trong của MTĐT đưa ra môi trường ngoài.

1.2.2.1 Thiết bị vào

• **Bàn phím** (keyboard) là thiết bị dùng để đưa dữ liệu vào MTĐT trực tiếp, không qua giá mang tin. Tương tự như trên máy chữ, trên bàn phím có các phím chữ cái, chữ số và các phím kí tự đặc biệt. Các phím chia thành bốn nhóm sau:

- *Nhóm phím chữ*: bao gồm các phím tương tự như phím máy chữ để gõ vào các chữ, các chữ số, các dấu.
- *Nhóm phím chức năng*: để thực hiện nhanh một số yêu cầu nào đó. Thường các phần mềm tự quy định những thao tác tương ứng với các phím chức năng. Bàn phím của máy tính PC thường đề sẵn 12 phím chức năng.
- *Nhóm phím điều khiển*: xác định một số chức năng đặc biệt như thiết lập các chế độ khác nhau của bàn phím.
- *Nhóm phím soạn thảo*: Nhóm này đặc biệt quan trọng vì khoảng 80-90% thời gian làm việc trên máy là soạn thảo văn bản. Các phím soạn thảo hỗ trợ những công việc thông thường nhất trong soạn thảo.
- Nhóm phím cuối cùng là *các phím chữ số*.



Hình 1.8: Bàn phím

Khi ta ấn một phím, tín hiệu được truyền cho máy tính thông qua bộ lập mã, tương ứng với kí tự của phím được ấn đó. Bàn phím là thiết bị vào thông dụng của các máy tính hiện nay.

• **Con chuột** (mouse) là một vật nhỏ vừa gọn, mặt dưới có một viên bi lăn được trên mặt phẳng. Khi di chuyển con chuột trên mặt phẳng, chiều và độ dài lăn được của viên bi được truyền vào máy tính dưới dạng các xung điện. Một chương trình xử lý các dữ kiện này sẽ tạo ra một ảnh (thường thể hiện dưới dạng mũi tên hay là một ô chữ nhật gọi là con trỏ màn hình- cursor) trên màn hình. Khoảng cách và chiều di chuyển của con trỏ trên màn hình cũng tương tự như khoảng cách và chiều di chuyển của con chuột. Vì vậy ta có thể dùng con chuột để điều khiển con trỏ để chỉ định các đối tượng làm việc trên màn hình.



Hình 1.9: Con chuột

1.2.2.2. Thiết bị ra

Các thiết bị cho phép chuyển thông tin từ bộ nhớ trong ra một giá mang tin khác gọi là thiết bị ra. Có nhiều thiết bị ra như:

• **Màn hình** (display hoặc monitor) là thiết bị ra, giống như màn hình của máy thu hình. Mọi chữ hay ảnh trên màn hình mà ta thấy được đều tạo từ các điểm ảnh (pixel) thể hiện bởi một chấm nhỏ trên màn hình. Ngoài các tính năng giống như màn hình của máy thu hình thông thường cần

phải kể đến các tính năng kỹ thuật có liên quan đến đặc thù của máy tính. Một tính năng quan trọng của màn hình là độ phân giải (resolution) chỉ mật độ điểm ảnh trên màn hình - đo khả năng thể hiện tinh tế của màn hình. Một tính năng khác là khả năng thể hiện màu sắc. Thực ra cả hai tính năng trên không chỉ phụ thuộc vào chính màn hình mà còn phụ thuộc vào thiết bị điều khiển màn hình (bản mạch video).



Hình 1.10: Một máy scanner dùng với giấy khổ A4 có tốc độ đọc 900



Hình 1.11: Màn hình dùng đèn tia âm cực dùng cho máy tính để bàn (desktop) (bên trái) và màn hình tinh thể lỏng dùng với máy tính xách tay (laptop) (bên phải.)

Các màn hình Super VGA thông thường hiện nay cho độ phân giải tới 768 x 1024 điểm ảnh với từ 2^8 đến 2^{24} sắc độ màu khác nhau. Một tính năng khác mà hầu hết các màn hình ngày nay đều phải tính đến là khả năng tiết kiệm năng lượng. Khi ngừng làm việc với máy một thời gian đủ dài, các màn hình có thể tạm thời ngừng hoạt động để khỏi tiêu hao năng lượng vô ích. Loại màn hình phổ biến nhất là đèn tia âm cực (đèn cathode) - chính là loại đèn hình dùng cho máy thu hình. Các điểm ảnh được tạo bởi các súng bắn điện tử trong đèn hình có phủ các vật liệu phát quang. Ngày nay người ta còn dùng các màn hình mỏng dùng công nghệ tinh thể lỏng hay

plasma. Các màn hình này thường dùng cho các máy tính xách tay (notebook) và bắt đầu dùng cho máy để bàn nhưng giá thành còn khá đắt.

• **Máy in (printer)** là thiết bị cho phép in ra các thông tin trên giấy in. Ta thường gặp một số loại máy in sau:

- *Máy in dòng (Line Printer)* có tốc độ in cực nhanh (từ 300 tới 1200 dòng/phút), nhờ sử dụng một trống kim loại hay một băng kim loại khảm các con chữ, chuyển động với tốc độ cao. Loại máy in này không in ảnh được vì các con chữ được tạo hình sẵn từ trước. Máy in dòng hay dùng ở những nơi cần in nhiều nhưng chỉ in chữ (ví dụ để in hoá đơn điện, nước, khí đốt hay ở các trung tâm máy tính của các đại học để in chương trình cho sinh viên...).

- *Máy in kim (dot printer)* là loại máy không dùng bộ chữ tạo dạng sẵn mà sử dụng một bộ các kim in. ảnh hay chữ được tạo bằng các chấm do kim in đập vào băng mực in vào giấy. Như vậy mỗi chữ được thể hiện qua một tổ hợp các điểm tách từ một ma trận điểm (khung chữ). Vì lý do này máy in kim còn gọi là máy in theo kiểu ma trận (matrix printer). Chất lượng của máy in kim có thể đánh giá qua tốc độ in (tính bằng số kí tự in được trong một giây) và mật độ điểm máy in có thể in được mà ta có thể đánh giá qua số đầu kim.

Tính năng thứ hai này đo độ tinh tế của ảnh và chữ được in ra. Máy in kiểu này khá rẻ và rất gọn nên hay được dùng phổ biến trong công tác văn phòng. Mặc dù chất lượng ảnh không thật cao nhưng do in bằng kim nên với việc in những bản in nhiều liên (như hoá đơn) thì chưa có máy in nào có thể thay được máy in kim.

- *Máy in laser (Laser Printer)* Máy in loại này dùng kỹ thuật laser để tạo ảnh từng trang một trên một trống tĩnh điện. Đầu tiên trống được đặt một điện tích âm có điện áp khoảng - 600 V. Sau đó người ta dùng chùm laser phối hợp với một chiếc gương quay chiếu lên trống theo dạng hình ảnh cần tạo để hạ điện áp những chỗ được chiếu xuống khoảng -100 V. Mực in (loại toner) được nạp điện áp thích hợp sẽ bị hút vào những chỗ có điện áp -100 V và tạo ra một bức ảnh “tĩnh điện” đúng như hình ảnh cần in. Khi trống áp vào giấy in những hạt mực sẽ dính trở lại giấy và được nung nóng để các hạt mực chảy ra thấm vào giấy. Ưu điểm của loại máy này là chất lượng ảnh rất cao. Ngày nay giá thành của máy in laser đã khá rẻ nên chúng đã



Hình 1.12: Một máy in laser

được sử dụng rộng rãi trong văn phòng.

- *Máy in phun mực (Inkjet Printer)*, thay vì dùng kim để tạo một điểm, máy này phun ra một tia mực siêu nhỏ. Công nghệ phổ biến nhất là dùng tinh thể áp điện để làm bơm mực. Một tinh thể áp điện sẽ co hay giãn tùy thuộc vào điện áp đặt vào hai mặt đối diện của tinh thể. Một nguyên lý khác cũng được dùng là đầu in có các ống phun mực nhỏ li ti, các ống này có thể làm nóng hầu như tức khắc. Khi ống bị nóng mực bị sôi tạo thành bong bóng siêu nhỏ bắn vào giấy. Máy in phun có chất lượng ảnh cao lại không ồn. Giá máy không đắt nhưng thiết bị phun mực nằm ngay trên hộp mực nên giá mực khá đắt.

• **Máy chiếu (Projector)**

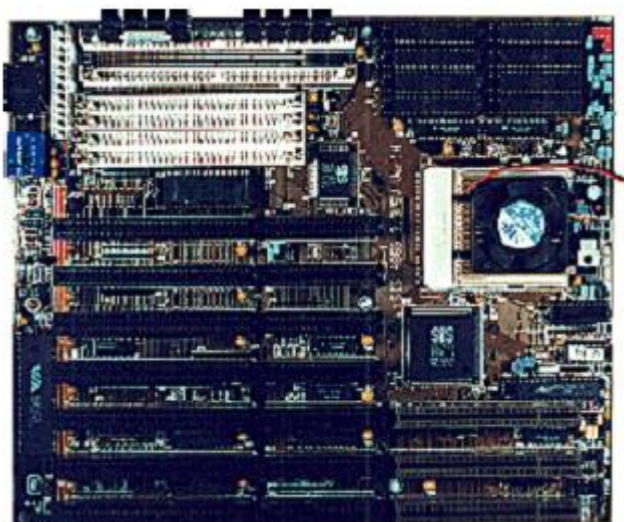
Máy chiếu là một thiết bị khá phổ biến hiện nay giúp chiếu màn hình của máy tính lên màn ảnh lớn. Máy chiếu rất hay được dùng để giảng bài, thuyết trình trong hội thảo với bài trình bày để sẵn trong máy tính. Hiện có hai loại máy chiếu phổ biến là máy chiếu dùng tinh thể lỏng (LCD) và máy chiếu DLP (Digital Light Processing) dùng vi gương. DLP được thực hiện bằng công nghệ nano, lần đầu tiên được Larry Hombeck (làm việc tại Texas Instrument) phát minh vào năm 1987, nhưng tới cuối những năm 90 mới được thương mại hoá. Để gửi ánh sáng, thiết bị tạo một chùm chớp sáng và dùng gương phản xạ chiếu lên màn ảnh. Gương trong DLP là linh kiện quang bán dẫn gọi là vi gương số (DMD – Digital Micro-mirror Device). Chíp DMD chứa đến hơn 2 triệu vi gương, mỗi cái chỉ lớn 16 micro mét vuông (nhỏ bằng 1/5 sợi tóc) và xếp cách nhau 1 phần triệu mét. Vi gương được điều khiển đồng bộ với các điểm ảnh đồ họa. Việc quay gương đi một góc nào đó sẽ giúp gương chiếu tia sáng vào màn ảnh hoặc chiếu tia sáng ra ngoài màn ảnh gây ra hiệu ứng tạo điểm ảnh hoặc tắt điểm ảnh trên màn ảnh. Gương có thể điều khiển thay đổi trạng thái tới 5000 lần/giây. Để tạo màu, người ta dùng các lọc màu theo ba màu cơ bản từ ánh sáng trắng để chiếu vào gương. Một số thiết bị có thể vừa là thiết bị vào vừa là thiết bị ra như:

- Các thiết bị đọc và ghi đĩa.
- Các modem để nối máy tính với nhau theo đường điện thoại. Tín hiệu số của máy tính qua modem sẽ biến thành tín hiệu tương tự (analog) để gửi theo đường điện thoại. Khi nhận, modem biến ngược trở lại từ tín hiệu tương tự ra tín hiệu số.

1.2.3. Bộ xử lý (CPU - Central Processing Unit)

CPU có chức năng điều khiển máy tính và xử lý thông tin theo chương trình đã được lưu trữ trong bộ nhớ. CPU gồm các thành phần:

- **Đồng hồ (clock)** tạo các xung điện áp chính xác, đều đặn để sinh ra các tín hiệu cơ bản để điều chế thông tin và đồng bộ hoá các thành phần khác của máy tính.
- **Các thanh ghi (registers)**. Ngoài bộ nhớ trong, CPU còn dùng các thanh ghi như là những bộ nhớ nhanh, chuyên dụng dùng trong khi thực hiện các lệnh. Các thanh ghi thường được dùng để ghi các lệnh đang được thực hiện, lưu trữ các dữ liệu phục vụ cho các lệnh, các kết quả trung gian, các địa chỉ, các thông tin dùng đến trong quá trình thực hiện một lệnh.
- **Khối số học và logic (ALU: arithmetic and logic unit)** là khối chức năng thực hiện các phép toán cơ sở của máy như các phép toán số học, các phép toán logic, phép tạo mã v.v. ALU bao gồm những mạch chức năng để thực hiện các phép toán đó.



Hình 1.13: Một bảng mạch chủ của máy vi tính. Trong đó có CPU, bộ nhớ và các mạch giao tiếp với ngoại vi.

Trong hình bên ta thấy một quạt điện nhỏ để làm nguội CPU, phía dưới quạt chính là CPU.

• **Khối điều khiển (CU: control unit)** là khối chức năng điều khiển sự hoạt động của MTĐT theo chương trình định sẵn. Nhờ công nghệ vi mạch, người ta có khả năng chế tạo toàn bộ bộ xử lý trong một chip (một mạch vi điện tử được đóng trong một vỏ duy nhất). Những bộ xử lý như vậy gọi là bộ vi xử lý (micro processor) viết tắt là μP .

1.2.4. Quá trình thực hiện lệnh

Để hiểu rõ quá trình này ta cần tìm hiểu thêm về lệnh máy. Mỗi lệnh máy là một yêu cầu ALU thực hiện một phép tính cơ sở (cộng, nhân, nhân logic, cộng logic, chọn lệnh cần thực hiện v. v). Các lệnh này phải chỉ ra đầy đủ các thông tin sau:

- Phép tính cần thực hiện. Trong lệnh máy nó cho bằng một số bit gọi là mã phép tính.
- Nơi đặt dữ liệu của lệnh. Thông tin này có thể là địa chỉ trong BNT hoặc là mã thanh ghi.
- Các thông tin liên quan đến kết quả thực hiện chẳng hạn địa chỉ của nơi để kết quả của phép toán.

Mã lệnh	Các thành phần địa chỉ
---------	------------------------

Hình 1.14: Cấu trúc lệnh.

Như vậy một lệnh có cấu trúc như Hình 1.14. Một chương trình máy là một dãy các lệnh. Do chương trình cũng nằm trong bộ nhớ nên chính các lệnh cũng có địa chỉ, đó chính là địa chỉ byte đầu tiên của lệnh. Quá trình thực hiện một chương trình là một quá trình thực hiện liên tiếp từng lệnh. Để quản lý thứ tự thực hiện các lệnh, CU sử dụng một thanh ghi gọi là thanh đếm địa chỉ (Program Counter - PC) ghi địa chỉ của lệnh sẽ thực hiện tiếp theo. Giá trị khởi tạo của PC là địa chỉ lệnh đầu tiên chương trình. MTĐT được điều khiển bởi các lệnh của chương trình. Chu kỳ thực hiện một lệnh bao gồm các bước sau:

- **Đọc lệnh:** Trong chu kỳ đọc lệnh, CU gửi nội dung PC vào bộ giải mã địa chỉ để đọc byte đầu tiên của lệnh lên một thanh ghi khác là thanh ghi lệnh. PC sẽ tăng lên một đơn vị để CU đọc byte tiếp theo. Độ dài các lệnh có thể khác nhau nhưng byte đầu tiên thường là nơi chứa mã lệnh.
- **Giải mã lệnh:** CU căn cứ vào mã lệnh để đọc nốt các thông tin địa chỉ của lệnh và hoàn thành việc đọc lệnh, PC tiếp tục tăng theo số lượng byte đã đọc vào.

- *Đọc dữ liệu:* Các địa chỉ dữ liệu được gửi vào bộ giải mã địa chỉ để đọc nội dung các đối tượng của lệnh gọi là các toán hạng (operand) vào các thanh ghi dữ liệu.
- *Thực hiện lệnh:* Phát tín hiệu điều khiển cho mạch chức năng của ALU thực hiện phép toán mà mã lệnh xác định. Sau đó quay lại chu kỳ đọc lệnh với nội dung mới của PC. Chú ý rằng nếu lệnh thực hiện là lệnh điều khiển thì giai đoạn thực hiện này sẽ đặt địa chỉ lệnh sẽ thực hiện tiếp theo vào PC.

Như vậy để thực hiện một lệnh nói chung phải đọc/ghi bộ nhớ nhiều lần. Có hai phương pháp tổ chức điều khiển:

- *Phương pháp điều khiển cứng.* Ứng với mỗi lệnh máy có một mạch điện thực hiện lệnh đã cho theo các tín hiệu điều khiển.
- *Phương pháp điều khiển vi chương trình.* Mỗi lệnh được thực hiện thông qua các lệnh sơ cấp hơn gọi là các vi lệnh. Khi đó người ta chỉ phải thiết kế phần cứng ở mức vi lệnh. Ví dụ, phép đọc một byte trong bộ nhớ có thể lấy làm một vi lệnh. Như vậy, một phép tính có thể thực hiện bằng cách thi hành một dãy vi lệnh gọi là vi chương trình. Do đó, khi thay đổi lệnh chỉ cần thay đổi vi chương trình tương ứng mà không cần thay đổi cấu trúc vật lý của MTĐT. Các CPU ngày nay không thực hiện lệnh theo kiểu tuần tự như trên mà thường thực hiện song song nhiều quá trình. Thông tin nạp từ bộ nhớ có thể là cả một khối lên một khối thanh ghi. Một số CPU có cả cơ chế xử lý thông minh để dự đoán các khối chương trình hay dữ liệu sắp dùng đến để tải trước lên thanh ghi. Trong khi đang thực hiện lệnh thứ nhất thì một thành phần khác giải mã lệnh thứ hai và một thành phần khác tải lệnh thứ 3 lên thanh ghi. Cách thức xử lý này gọi là pipeline. Nhờ phương thức này mà nhiều bộ xử lý có thể thực hiện nhiều lệnh đồng thời.

Bài đọc thêm: Các thế hệ máy tính điện tử

Các thế hệ máy tính có thể phân biệt theo công nghệ và hiệu năng. Người ta đã nói tới 6 thế hệ máy tính nhưng trên thực tế một số thế hệ vẫn chỉ là những dự án trong phòng thí nghiệm. Thế hệ thứ nhất mở đầu với sự ra đời của chiếc MTĐT đầu tiên (ENIAC). Về mặt công nghệ, chúng được chế tạo bằng đèn điện tử. Vì vậy các máy tính điện tử thế hệ đầu rất cồng kềnh, tiêu thụ nhiều năng lượng, tốc độ chậm (vài nghìn phép tính/giây) và khả



Hình 1.15: UNIVAC I, một máy tính thế hệ I

năng nhớ rất thấp (vài trăm cho đến vài nghìn từ). Chiếc máy tính đầu tiên ENIAC dùng tới 1900 bóng điện tử, nặng 30 tấn, chiếm diện tích làm việc tới 140 m², có công suất tiêu thụ tới 40KW và cần một hệ thống thông gió khổng lồ để làm mát máy. Nhược điểm lớn nhất của các máy tính thế hệ thứ nhất là độ tin cậy không cao. Một số máy phải thay thế tới 20% số đèn điện tử sau mỗi ngày làm việc. Những đại diện cho máy tính thế hệ thứ nhất có thể kể tới EDVAC, LEO, UNIVAC I.

Thế hệ thứ hai sử dụng công nghệ bán dẫn, ra đời vào khoảng đầu những năm 50. Về mọi phương diện (kích thước, năng lượng tiêu hao, tốc độ xử lý...) công nghệ bán dẫn đều tỏ ra ưu việt hơn dùng đèn điện tử. Các máy tính thế hệ hai bắt đầu sử dụng bộ nhớ xuyên ferit cho phép tăng tốc truy cập dữ liệu. Tốc độ trung bình của máy tính thế hệ hai đạt từ vài nghìn cho đến hàng trăm nghìn phép tính trong một giây, bộ nhớ trong khoảng vài chục nghìn từ máy. Những máy tính thế hệ thứ hai điển hình là ATLAS, họ IBM/7000. Chiếc MTĐT đầu tiên có ở Việt Nam (Minsk-22, năm 1967) là một máy tính thế hệ 2 có tốc độ tính theo phép nhân là 5000 phép tính/giây, bộ nhớ gồm 8192 từ 37 bit.



Hình 1.16: Máy tính PDP, một máy tính thế hệ 2 điển hình



Hình 1.17: Bàn điều khiển của máy tính Minsk-22

Thế hệ thứ ba khởi đầu với sự ra đời của họ máy tính nổi tiếng IBM/360 và CL/1900 vào năm 1964. Các máy IBM/360 được đưa vào Việt Nam từ năm 1968. Thế hệ thứ ba là các máy tính sử dụng công nghệ vi điện tử. Công nghệ vi điện tử cho phép chế tạo các mạch bán dẫn không phải từ các linh kiện rời mà chế tạo đồng thời cả một mạch chức năng cỡ lớn với các thành phần siêu nhỏ. Nhờ có độ tích hợp cao mà về mọi phương diện (kích thước, năng lượng tiêu hao, tốc độ xử lý) các máy tính thế hệ thứ 3 có đều tốt hơn rất nhiều so với máy tính thế hệ

thứ 2. Tốc độ các máy tính đã đạt từ vài trăm nghìn tới hàng triệu phép tính một giây. Lúc đầu các máy tính thế hệ 3 vẫn dùng bộ nhớ xuyên ferit, sau đó dùng bộ nhớ màng mỏng từ rồi bộ nhớ bán dẫn. Dung lượng bộ nhớ trong đạt khoảng vài trăm nghìn đến vài triệu byte. Một ưu điểm quan trọng khác của máy tính thế hệ 3 là tính mô đun cho phép có thể ghép nối hay mở rộng một cách dễ dàng.

Người ta thấy rằng mỗi thế hệ máy tính đều gắn liền với một cuộc cách mạng trong công nghệ chế tạo với chu kỳ khoảng 6-7 năm. Vì thế vào cuối những năm 60 người ta chờ đợi sự ra đời của thế hệ máy tính thứ tư. Thực tế đã không có một cuộc cách mạng trong công nghệ chế tạo vì vậy khó có thể nói đến các đặc trưng công nghệ của thế hệ này (Thậm chí ít thấy cả những cuộc tranh luận thế nào là máy tính thế hệ thứ 4). Tuy nhiên trong nhiều tài liệu, người ta xem những máy tính chế tạo trên cơ sở công nghệ mạch tích hợp mật độ cao VLSI (Very Large Scale Intergration) là các máy tính thế hệ thứ 4.



Hình 1.18: Máy tính IBM/360, dòng máy tính thế hệ 3 đầu tiên và rất nổi tiếng

Chúng ta ghi nhận hai xu hướng có vẻ đối nghịch cùng song song phát triển trong giai đoạn này: xây dựng những siêu máy tính (super computer) và xây dựng những máy tính cực nhỏ (micro computer). Các siêu máy tính thường được thiết kế dựa trên các kiến trúc song song, một máy tính có thể có nhiều bộ xử lý hoạt động cộng tác với một bộ nhớ chung. Những thành tựu mới của công nghệ vi điện tử cho phép chế tạo ra các máy tính rất mạnh. DeepBlue, máy tính đầu tiên đánh thắng nhà vô địch cờ thế giới Caxparov là một máy song song gồm 256 bộ xử lý PowerPC có khả năng phân tích 200 triệu nước cờ trong một giây. Trong bảng xếp hạng máy tính tính tới tháng 11 năm 2004, siêu máy tính số 1 là máy Blue Gen của công ty IBM bao gồm 65536 bộ xử lý, bộ nhớ 16TB (16 nghìn tỉ byte), đĩa cứng 400TB, tốc độ xử lý 70,72 Tflo (nghìn tỉ phép tính dấu phẩy động một giây). Máy tính được xếp hạng 1 tới tháng 6/2003, là chiếc ES (Earth Simulator) của hãng NEC (Nhật bản), được xây dựng từ 5104 bộ xử lý, với tốc độ tính toán 35,86 nghìn tỉ phép tính dấu phẩy động một giây, bộ nhớ 10 nghìn tỉ byte, đĩa cứng 700 nghìn tỉ byte. Máy này tới tháng 11/2004 đã lùi xuống hạng thứ 3. Song song với xu hướng trên là xu hướng thu nhỏ máy tính. Công nghệ vi điện tử đã cho phép chế tạo toàn bộ bộ xử lý trong một vi mạch duy nhất gọi là bộ vi xử lý (microprocessor). Bộ vi xử lý (BVXL) đầu tiên đưa ra thị trường là vi mạch 4004 của hãng Intel vào năm 1971 đã mở đầu cho kỷ nguyên máy vi tính. Các máy vi tính (micro computer) là các máy tính được xây dựng trên các bộ vi xử lý. Vào những năm 80 đã xuất hiện tới 300 loại máy vi tính trong đó có những máy có ảnh hưởng rất lớn đến tạo chuẩn cho máy vi tính. Máy PC của hãng IBM ra đời năm 1981 - đó là tiền thân của hầu hết các máy vi tính đang dùng ở Việt Nam hiện nay. Một dòng máy khác đã khai sinh ra dòng máy tính văn phòng với phong cách giao tiếp với người sử dụng rất thân thiện là máy Macintosh của hãng Apple.



Hình 1.19: Siêu máy tính Earth Simulator xếp hạng 1 tính đến 6/2003 và hạng 3 tính đến 11/2004



Hình 1.20:

Máy tính Macintosh (Apple), dòng máy tính văn phòng có giao diện đồ họa đầu tiên. Bên trái là giao diện của phần mềm MacPaint có từ năm 1986 trên Macintosh mà phần mềm PaintBrush của Microsoft Windows đã sử dụng ý tưởng. Với giá thành ngày càng rẻ với công suất ngày càng tăng, máy vi tính đã đến từng gia đình.

Có thể nói không thể có xã hội thông tin nếu không có máy vi tính. Một máy vi tính ngày nay có công suất xử lý gấp hàng trăm lần các máy tính gọi là lớn những năm 70. Trong khi người ta chưa hình dung máy tính thế hệ thứ tư sẽ như thế nào thì 1981, Nhật bản đã đưa ra một chương trình đầy tham vọng, cuốn hút các cường quốc máy tính vào một dự án chế tạo máy tính

thế hệ thứ năm. Theo dự án này thì máy tính thế hệ thứ năm sẽ là máy tính thông minh, có thể giao tiếp trên ngôn ngữ tự nhiên, có thể có các hoạt động mang tính sáng tạo dựa trên một cơ chế suy luận trên các tri thức và không hoàn toàn tuân theo nguyên lý Von Neumann. Tất nhiên những máy tính đó phải rất mạnh để thực hiện được rất nhiều lập luận trong một thời gian ngắn. Mặc dù mục tiêu đặt ra đã không đạt được nhưng người ta đã thu được rất nhiều các thành quả về công nghệ xử lý tri thức. Ngay khi việc nghiên cứu thế hệ thứ 5 đang triển khai thì người ta đã nghĩ đến máy tính thế hệ thứ 6 hoạt động theo nguyên lý sinh học. Đến nay người ta chưa hiểu nhiều về nguyên lý xử lý



thông tin của bộ não tuy vậy một mô hình xử lý dựa trên sự lan truyền tín hiệu của mạng neuron đã được xây dựng, một số thử nghiệm về các chất hữu cơ có hiệu ứng bán dẫn cũng đã được xem xét. Một số kết quả ban đầu về mạng neuron đã đưa vào ứng dụng như các máy y tế, các máy phát hiện chất nổ tại các sân bay, các thiết bị nhận dạng trong quân sự... Bây giờ còn quá sớm để có thể nói về tương lai của các máy tính phỏng sinh học này.

1.3. Hệ điều hành (HĐH)

1.3.1. Khái niệm

Máy tính là một thiết bị phức tạp với nhiều thành phần như CPU, bộ nhớ trong, bộ nhớ ngoài, thiết bị ngoại vi. Việc điều khiển máy vô cùng phức tạp. Trong thời kỳ đầu khi mới có máy tính, phương thức khai thác máy là trực tiếp theo đó chương trình được viết trên ngôn ngữ máy và người sử dụng có thể can thiệp trực tiếp vào mọi quá trình làm việc của máy. Phương thức làm việc như vậy rất kém hiệu quả. Tốt nhất là dùng máy tính để quản lý chính nó. Theo cách đó người ta lập các phần mềm hệ thống để quản lý tài nguyên của máy, quản lý công việc xử lý trên máy và giao tiếp với người điều khiển. Đó chính là hệ điều hành. Ngày nay máy tính rất phức tạp, không có hệ điều hành thì không thể điều khiển nổi máy tính. Như vậy hệ điều hành là hệ thống nằm giữa con người và máy giúp con người thực hiện công việc xử lý của họ (thể hiện bởi chương trình ứng dụng của họ) một cách hữu hiệu. Hoạt động của máy tính không thể tách rời khỏi hệ điều hành. Vì thế người ta coi máy tính và hệ điều hành là một máy ảo (virtual machine).

1.3.2. Chức năng của hệ điều hành

Hệ điều hành gồm có các chức năng sau:

- *Quản lý và điều phối các thiết bị của máy để phục vụ cho công việc xử lý.*
- *Quản lý thông tin ở bộ nhớ ngoài.* Ngoài việc quản lý thiết bị thì hệ điều hành còn phải quản lý các thông tin đã được lưu trữ để có thể dùng nhiều lần. Đây là một công việc rất phức tạp. Các thông tin ở bộ nhớ ngoài được tổ chức thành các đơn vị lưu trữ gọi là tệp. Vì thế, phân hệ thực hiện chức năng quản lý thông tin ở bộ nhớ ngoài gọi là hệ quản lý tệp (file management system). Các ứng dụng muốn tìm tệp hay lưu trữ thông tin lên bộ nhớ ngoài đều phải thông qua hệ quản lý

tệp. Chức năng này giải phóng các chương trình ứng dụng ra khỏi một công việc rất phức tạp và tỉ mỉ.

- *Quản lý các tiến trình (process management)*. Về cơ bản, một tiến trình là một chương trình đang thực hiện trên máy tính. Ngoài các chương trình của người ứng dụng, còn có các tiến trình hệ thống như quản lý vào ra, điều phối tài nguyên. . . Thực chất quản lý tiến trình là lập lịch thực hiện các tiến trình phù hợp với yêu cầu tài nguyên của mỗi tiến trình.
- *Cung cấp môi trường giao tiếp với người sử dụng* kể cả việc cung cấp các tiện ích cơ bản. Mỗi hệ điều hành thường cung cấp một ngôn ngữ giao tiếp với người sử dụng. Trước đây ngôn ngữ giao tiếp thường là các lệnh (command). Ngày nay một số hệ điều hành như WINDOWS hay UNIX cung cấp cả môi trường giao tiếp theo kiểu đồ họa theo đó người ta chỉ cần chỉ để chọn ra việc cần làm chứ không phải gõ các lệnh. Hệ điều hành phải được khởi động ngay trước khi máy tính làm việc với các chương trình khác. Chương trình điều hành phải luôn luôn thường trực cho tới khi máy ngừng hoạt động. Ngày nay, nói đến sử dụng máy tính thực chất là sử dụng hệ điều hành.

Bài đọc thêm. Hệ điều hành LINUX

LINUX là hệ điều hành kiểu UNIX viết cho máy PC, ra đời vào năm 1991. Tác giả của LINUX là Linus Torwalds, một người Phần lan. Linus Torwalds lưu ý mọi người cần phát âm tên hệ điều hành LINUX là “li-nu- x” chứ không phải phát âm theo kiểu Anh như nhiều người vẫn đọc là “lai-nơ-x”. LINUX đã trở thành một hiện tượng, một trung tâm chú ý của những người làm CNTT trong vài năm qua. Thập kỷ 90 vừa qua cũng chính là thập kỷ mà hệ điều hành Windows của Microsoft ngự trị. Các hãng viết phần mềm lớn đều phải tham gia phát triển trên môi trường Windows nếu muốn có chỗ đứng trên thị trường phần mềm. Người ta thấy rằng khó có một hệ điều hành nào có thể cạnh tranh với Windows vì Windows rất dễ sử dụng, số lượng phần mềm chạy trên môi trường Windows rất lớn và chi phí sở hữu phần mềm trên Windows nói chung là thấp hơn một phần mềm có tính năng tương đương trên UNIX khoảng 2-3 lần. Ai đã từng dùng UNIX trước kia đều có một cảm giác UNIX là một hệ điều hành của giới chuyên nghiệp và đầy rẫy các bí ẩn ở trong. Trong hoàn cảnh như vậy, tại sao LINUX được hàng chục triệu người quan tâm và được nhiều hãng phần mềm ủng hộ? Có một số lý do sau:

- Trước hết LINUX là một hệ điều hành mã mở. Linus cung cấp hệ điều hành LINUX cùng với mã nguồn viết trên ngôn ngữ C. Điều đó có nghĩa là người nào muốn biết LINUX chạy ra sao, thậm chí muốn sửa đổi, bổ sung chức năng đều được.
- Thứ hai là LINUX được cung cấp miễn phí, người dùng chỉ phải trả tiền tài liệu, tiền đĩa mà không phải trả tiền bản quyền. Nếu lấy từ Internet xuống thì không phải trả bất cứ một chi phí nào.
- Thứ ba, LINUX viết cho máy tính cá nhân họ PC, cộng đồng máy tính lớn nhất hiện nay, đó chính là lý do khiến nhiều người quan tâm và cùng đóng góp cho nó.
- Thứ tư, đối với người sử dụng LINUX tỏ ra không thuận lợi bằng Windows nhưng tính ổn định của hệ điều hành này cho phép xây dựng những máy chủ tin cậy đặc biệt là các máy chủ Internet. Với LINUX có rất nhiều phần mềm cho máy chủ Internet với giá rẻ.
- Cuối cùng, nhiều hãng máy tính kể cả phần cứng và phần mềm hiện đang hỗ trợ cho Windows nhưng đều sợ khả năng thao túng của cái gọi là “liên minh Wintel”. Microsoft là một người

không lồ cả về sức mạnh con người với tài quản lý và kỹ thuật và cả sức mạnh tài chính. Hệ điều hành Windows của Microsoft lần lượt đánh bại hầu hết các hệ điều hành khác như OS/2 của IBM, Mac/OS của Apple, SOLARIS của SUN, Novell của NETware. Các hãng phần mềm một mặt phải hỗ trợ Windows để giành thị phần nhưng vẫn muốn có một đối thủ của Windows để họ khỏi bị lệ thuộc. Đó là một lý do tại sao rất nhiều hãng phần mềm có tên tuổi như IBM, SUN đang xây dựng những phần mềm mạnh và miễn phí trên môi trường LINUX.

Bản địa hoá hệ điều hành LINUX

Về mặt an ninh, có những nghi ngờ liệu trong hệ điều hành Windows có ẩn giấu một điều khiển ngầm mà trong những hoàn cảnh nào đó điều khiển này được kích hoạt hay không. Mặc dù ít người tin điều này (vì nếu Windows làm như vậy mà sự việc bị vỡ lở thì Microsoft sẽ sụp đổ) nhưng những người phụ trách an ninh quốc gia không có quyền bỏ qua bất kỳ một nguy cơ tiềm ẩn nào. Với LINUX mã nguồn mở, có thể kiểm soát, thậm chí sửa đổi từng chi tiết nhỏ trong hệ điều hành, có thể đưa thêm các kiểm soát riêng vào hệ điều hành. Các nước nghèo còn một vấn đề khác là trả bản quyền phần mềm cho hệ điều hành. Người ta muốn xây dựng một hệ điều hành riêng trên cơ sở bản địa hoá (localization) LINUX. Việc bản địa hoá là một việc làm không tốn công lắm. Có hai việc chính cần làm trong bản địa hoá là thay đổi các thông báo trong mã của LINUX từ tiếng Anh chuyển sang tiếng địa phương và xây dựng các tiện ích giao tiếp với bản ngữ, ví dụ xây dựng bộ chữ Việt trên LINUX và bộ gõ bàn phím tiếng Việt. Thực ra vấn đề bản địa hoá nên đặt ra đối với chính các ứng dụng chạy trên LINUX hơn là với LINUX cũng như là công việc bản địa hoá trên môi trường WINDOWS. Microsoft đã từng thất bại trong việc Việt hoá WINDOWS/95 nên có thể nói khả năng thất bại của việc bản địa hoá LINUX rất cao do người dùng không cần đến sự bản địa hoá này. Không phải chỉ có nước nghèo tính đến việc dùng LINUX mà một số nước phát triển cũng đang xem xét kỹ lưỡng việc này. Việc lệ thuộc công nghệ hoàn toàn vào một công ty dù là công ty hùng mạnh đến mức nào cũng tiềm ẩn những thâm hoạ. ở Trung Quốc, công ty Hồng Kỳ đã phát triển một bản LINUX tiếng Hoa và chính phủ Trung quốc đã có quy định để tất cả các cơ quan chính phủ ở Bắc Kinh chỉ được dùng hệ điều hành LINUX. Gần đây để tránh mất thị trường chính MicroSoft đã phải tính đến mở mã Windows ở Trung Quốc cũng như họ đã phải làm ở Nga.

Dùng Windows hay LINUX ?

Theo nhiều chuyên gia thì lý do kinh tế, lựa chọn LINUX vì không phải trả tiền bản quyền là nguy hiểm. Người ta thấy rằng môi trường phát triển phần mềm trên LINUX hiện nay còn quá tồi so với môi trường phát triển phần mềm trên Windows. Chi phí phát triển phần mềm trên LINUX sẽ đắt gấp từ 3-5 lần chi phí phát triển phần mềm trên Windows, khiến cho giá sở hữu phần



mềm ứng dụng trên UNIX nói chung và LINUX nói riêng đắt hơn từ 2 - 3 lần giá của một phần mềm cùng tính năng trên Windows. Hơn nữa trên mỗi máy chỉ cần cài một hệ điều hành nhưng cần vài chục phần mềm ứng dụng. Nếu chỉ vì hệ điều hành rẻ hơn vài trăm USD mà mua phần mềm ứng dụng hoặc chi phí phát triển phần mềm ứng dụng đắt hơn nhiều nghìn USD thì rõ ràng hiện nay dùng LINUX chưa có lợi. Mặc dù biết UNIX là tốt nhưng các chuyên gia tin học sẽ chọn hệ điều hành nào cho chính máy tính của họ? Hiện nay hầu như chắc chắn sẽ là Windows.

Bricklin nói một cách hài hước về tình trạng này như sau: “Mọi người bỏ phiếu bằng một tay, còn tay kia thò vào túi đếm tiền”. Điều may mắn là LINUX nằm trong trào lưu GNU, trào lưu xây dựng quỹ phần mềm mã mở. Chúng ta có quyền hy vọng đến một lúc nào đó các hãng làm phần mềm làm ra nhiều phần mềm ứng dụng trên LINUX với giá cạnh tranh được với Windows. Tương lai của LINUX chính là ở chỗ này nhưng bây giờ thì phải đợi. Đối với môi trường đại học, đặc biệt trong các cơ sở đào tạo CNTT thì LINUX là một cơ may để những người làm tin học trong tương lai nắm được ngọn ngành của một phần mềm hệ thống phức tạp nhất. Ở Việt Nam, LINUX cũng được đông đảo những người làm tin học quan tâm. Hội thảo LINUX tháng 12/2000 tại Hà Nội có tới 300 người làm tin học tham dự. Chính phủ cũng đã đầu tư một khoản tiền không nhỏ để nghiên cứu và phát triển ứng dụng trên LINUX. Giải nhất Trí tuệ Việt Nam năm 2002 cũng đã được trao cho phần mềm LINUX tiếng Việt do nhóm ViệtKey thực hiện. Tháng 12/2002 một hội thảo lớn về phần mềm mã mở (chủ yếu là trên nền LINUX) cũng đã được tổ chức tại Hà Nội. Tháng 6/2003, Bộ Khoa học và Công nghệ đã có quyết định triển khai chương trình sử dụng các phần mềm mã nguồn mở và khuyến khích việc đưa việc giảng dạy sử dụng mã nguồn mở trong các trường đại học.

1.4. Mạng máy tính (MMT)

1.4.1. Khái niệm

Mạng máy tính là một tập hợp các máy tính nối với nhau bằng những đường truyền vật lý, theo một kiến trúc nhất định.

Khi nói về kiến trúc mạng ta muốn nói tới hai khía cạnh:

- *Tô pô của mạng (topology)*: đó là cách liên kết các máy với nhau về phương diện hình học.
- *Giao thức của mạng (protocol)*: đó là các quy ước truyền thông để các máy tính trong mạng có thể liên lạc, trao đổi thông tin với nhau.

Đường truyền vật lý dùng để chuyển các tín hiệu điện tử giữa các máy tính. Các tín hiệu điện tử đó biểu thị các giá trị dữ liệu dưới dạng các xung nhị phân (on/off). Tất cả các tín hiệu được truyền giữa các máy tính đều thuộc một dạng sóng điện từ (EM) nào đó, trải từ các tần số radio tới sóng cực ngắn (viba) và tia hồng ngoại. Tùy theo tần số của sóng điện từ có thể dùng các đường truyền vật lý khác nhau để truyền các tín hiệu.

Các máy tính được kết nối thành mạng máy tính nhằm đạt tới các mục tiêu chính dưới đây:

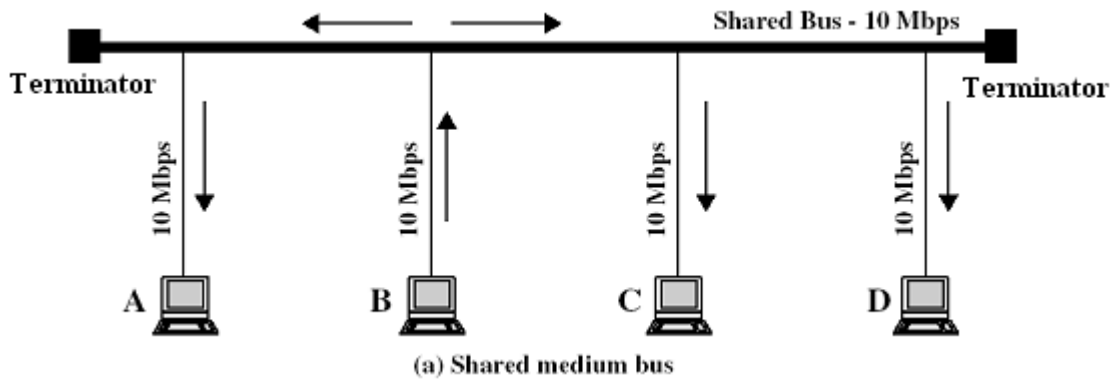
- Làm cho các tài nguyên có giá trị cao (thiết bị, chương trình, dữ liệu, ...) trở nên khả dụng với bất kỳ người sử dụng nào trên mạng.
- Tăng độ tin cậy của hệ thống nhờ khả năng thay thế khi xảy ra sự cố đối với một máy tính nào đó (đặc biệt quan trọng với các ứng dụng thời gian thực).

1.4.2. Phân loại mạng máy tính

Có nhiều cách phân loại mạng khác nhau tùy thuộc vào yếu tố chính được chọn để làm chỉ tiêu phân loại, chẳng hạn là “khoảng cách địa lý”, “kỹ thuật chuyển mạch”, “kiến trúc mạng”, ...

Nếu xét theo “khoảng cách địa lý” thì người ta chia thành các loại mạng sau:

- **Mạng cục bộ (LAN - Local Area Networks):** là mạng được cài đặt trong phạm vi tương đối nhỏ, chẳng hạn một tòa nhà, khu trường học, ...

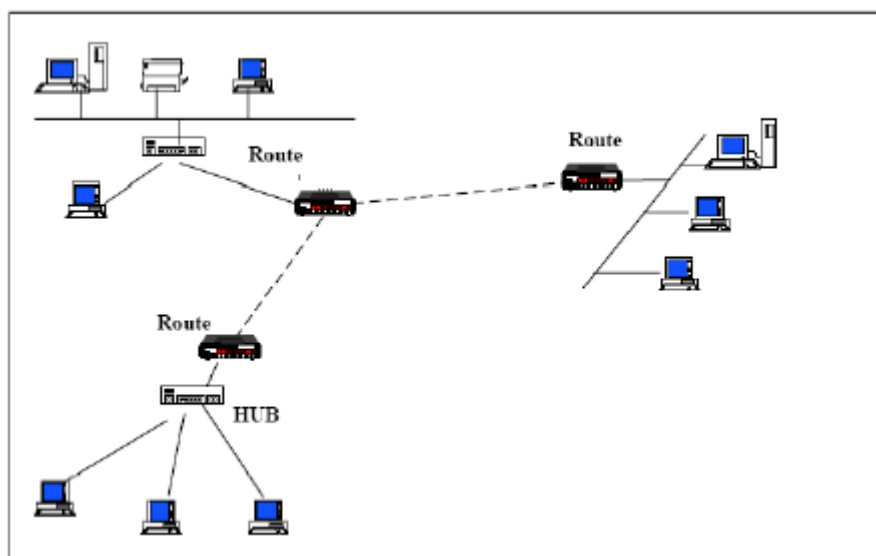


Hình 1.21: Mạng LAN theo tổ pô mạng Bus

Mạng cục bộ có những đặc trưng công nghệ sau:

- Kích thước nhỏ, thời gian truyền trong mạng bị giới hạn và phải biết trước.
- Tốc độ truyền cao, 1 Mbps - 100 Mbps
- Không có quan hệ Master-Slave, các máy tính cộng tác trong việc điều khiển truy cập đường truyền chung.
- Phương thức trao đổi: broadcast, connectionless,
- Tổ pô mạng: bus, ring, star, tree,

- **Mạng diện rộng (WAN - Wide Area Networks):** là mạng được cài đặt trong phạm vi rộng có thể vượt qua biên giới thậm chí cả lục địa. Tuy nhiên tốc độ truyền dữ liệu chậm và kém an toàn hơn so với LAN.



Hình 1.22a. Sơ đồ một WAN liên kết các LAN thông qua các bộ dẫn đường

Có thể xây dựng mạng rộng bằng cách liên kết các mạng cục bộ qua các đường truyền viễn thông (như cáp quang, các đường truyền riêng, vệ tinh. . .) thông qua các thiết bị kết nối. Các thiết bị này gọi là bộ dẫn đường hay định tuyến (router) có chức năng dẫn các luồng tin theo đúng hướng. Người ta sử dụng router để kết nối các LAN (để tạo nên những WAN) và để kết nối các WAN (để tạo nên các WAN lớn hơn).

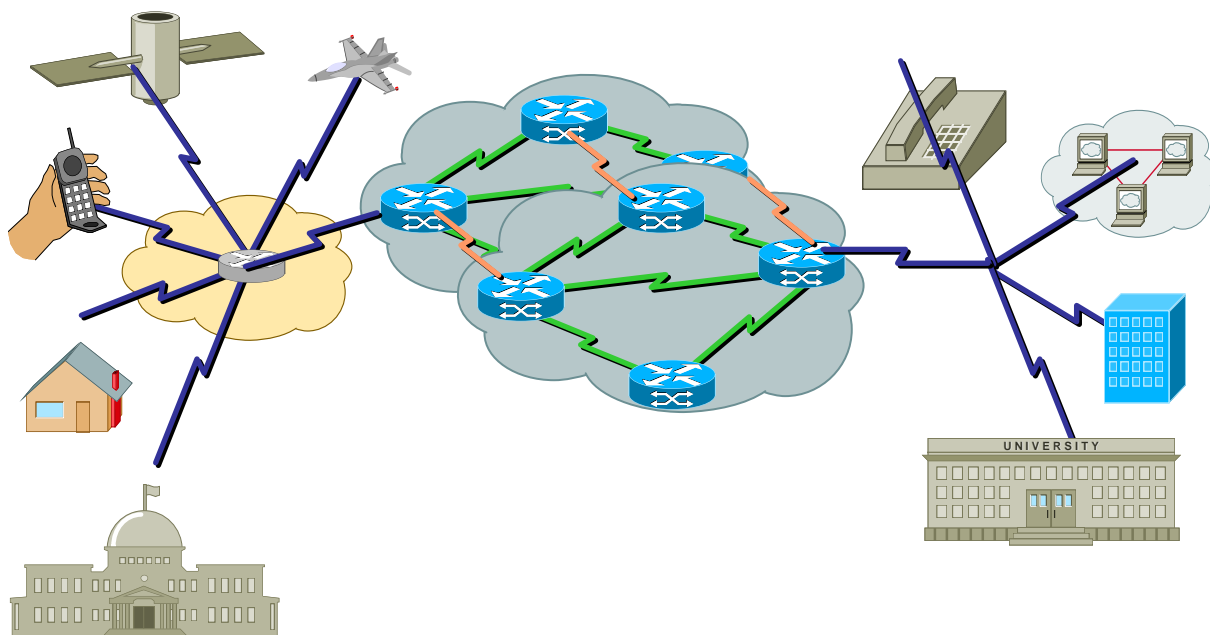
- *Mạng toàn cầu (GAN - Global Area Networks)*: Phạm vi mạng trải rộng khắp các lục địa của trái đất. Internet là một mạng toàn cầu.

1.5. Internet

1.5.1. Internet là gì?

Internet là một mạng kết nối hàng trăm mạng trên thế giới, liên kết các trường đại học, viện nghiên cứu, chính phủ, các doanh nghiệp thương mại, các tổ chức và các cá nhân khác nhau. Mạng Internet có mặt ở nhiều quốc gia, nhiều bang và có thể bao gồm nhiều mạng miền cùng hàng trăm mạng con của các trường học và những thư viện nghiên cứu, hàng trăm ngàn điểm thương mại. Internet có thể đến hầu như mọi miền trên thế giới, hệ máy tính lớn (mainframe), trạm làm việc (workstation), máy server, và máy vi tính cá nhân đều có thể nối được đến Internet.

Có thể nói Internet là mạng của các mạng vì Internet là sự kết nối của các mạng thông qua các bộ dẫn đường router. Người ta cũng nói Internet là mạng toàn cầu vì Internet hiện nay là một mạng rộng của vài trăm triệu máy tính trên phạm vi toàn cầu. Chính xác hơn, Internet là mạng toàn cầu sử dụng giao thức TCP/IP. Điểm khác với các mạng máy tính thông thường là ở chỗ Internet không có chủ và không có mạng nào điều hành mạng nào. Các mạng máy tính hay các máy đơn lẻ có thể tham gia tự nguyện vào Internet. Người tham gia mạng có khả năng khai thác tài nguyên (resources) thông tin thậm chí các thiết bị trên các mạng thành viên khác và cũng có nghĩa vụ tạo ra các nguồn tài nguyên cho người khác sử dụng. Chính vì vậy, về mặt thông tin Internet được xem là một kho tài nguyên thông tin toàn cầu. Dĩ nhiên để có thể làm điều này các máy tính tham gia InterNet phải có một địa chỉ nhất quán để các máy khác liên kết được. Mặc dù InterNet không có chủ nhưng vẫn có những tổ chức phi lợi nhuận, phi chính phủ (non profit, non governmental) quản lý việc cấp địa chỉ và nghiên cứu các chính sách cũng như công nghệ trên Internet.



Hình 1.22b

1.5.2 Giao thức TCP/IP ^[2]

Có hàng chục giao thức dùng với Internet trong đó có hai giao thức chính là giao thức định địa chỉ và chọn đường có tên là IP (Internet Protocol) và điều khiển việc truyền tin có tên là TCP (Transmission Control Protocol).

1.5.2.1. Địa chỉ IP

Để tham gia Internet, các thực thể truyền thông (máy tính và các thiết bị mạng có các hoạt động xử lý - trong tiếng Anh gọi là host) cần được cấp một địa chỉ gọi là địa chỉ IP. Mỗi địa chỉ IP là một số cho trong 4 byte. Người ta sử dụng cách viết gọi là dạng “dot decimal” để dễ đọc địa chỉ, theo đó giá trị trong mỗi byte được viết thành một số thập phân. Các số thập phân này tách nhau bởi dấu chấm ví dụ 192.13.23.120. Ta biết rằng số nguyên ghi trong một byte có giá trị nằm trong khoảng từ 0 đến 255.

Địa chỉ IP được chia làm nhiều lớp có kiểu là A, B, C, D. . . Sự khác nhau cơ bản giữa các lớp địa chỉ này là ở khả năng tổ chức các cấu trúc con của nó. Ví dụ một địa chỉ lớp B có thể cho tới 65535 địa chỉ của các máy trong mạng còn một địa chỉ lớp C chỉ có thể cho 255 địa chỉ.

Cấu trúc một địa chỉ IP 32 bit được mô tả trong hình 1.23 dưới đây:

Mã lớp	Địa chỉ mạng	Địa chỉ máy trên mạng (host)
Lớp A		
0	7 bit địa chỉ mạng	24 bit địa chỉ máy trên mạng
Như vậy	có 2^7 mạng địa chỉ lớp A và mỗi lớp A cho phép đánh địa chỉ cho 2^{24} máy	
Lớp B		
1 0	14 bit địa chỉ mạng	16 bit địa chỉ máy trên mạng

Như vậy có 2^7 mạng địa chỉ lớp B và mỗi lớp B cho phép đánh địa chỉ cho 2^{16} máy

Lớp C

110	21 bit địa chỉ mạng	8 bit địa chỉ máy trên mạng
-----	---------------------	-----------------------------

Như vậy có 2^7 mạng địa chỉ lớp C và mỗi lớp C cho phép đánh địa chỉ cho 2^8 máy

Hình 1.23: Cấu trúc địa chỉ IP của lớp A, B, C.

Cấu trúc của một gói tin trong giao thức IP

Thông tin truyền đi trên Internet không chỉ chính là nội dung cần truyền mà nó được đóng gói theo một cấu trúc được quy định, trong đó có một số thông tin kiểm soát việc truyền tin. Các thông tin này được ghi ở phía đầu gói tin gọi là header. Hình 1.24 minh họa cấu trúc của gói tin truyền đi trên Internet (trong tiếng Anh gọi là các IP packet). Phần header gồm ít nhất 20 byte chứa một số thông tin được minh họa trong 5 dòng đầu tiên.

Ver. 4 bit	Độ dài header, 4 bit	Dịch vụ, 8 bit	Độ dài gói tin, 16 bit	
Định danh gói tin (ID), 16 bit			Cờ, 3 bit	Offset phân đoạn, 13 bit
Thời gian sống, 8 bit	Giao thức, 8 bit		Tổng kiểm tra của header, 16 bit	
Địa chỉ IP của nơi gửi, 32 bit				
Địa chỉ IP của nơi nhận, 32 bit				
Vùng thông tin bổ sung nếu cần			Vùng đệm	
Dữ liệu				

Hình 1.24: Cấu trúc gói tin truyền đi trên Internet.

Phần dữ liệu là nội dung thông tin cần chuyển đi. Vùng tùy chọn không nhất thiết phải có. Nếu ta muốn mở rộng header để đưa thêm một số thông tin kiểm soát truyền thông thì sẽ lấy thêm phần tùy chọn. Ta nêu ý nghĩa của một số trường để hiểu rõ hơn cách kiểm soát truyền tin trên Internet.

- Độ dài header là chiều dài vùng mô tả vùng header của gói tin không kể phần dữ liệu tính theo đơn vị 4 byte. Bình thường giá trị đó sẽ là 5 (20 byte). Nếu phần tùy chọn được dùng thì số này có thể thay đổi.
- Độ dài gói tin là độ dài tính theo byte của toàn bộ gói tin kể cả phần header. Do phần này sử dụng 2 byte nên chiều dài cực đại có thể lên tới 65535 byte nhưng trên thực tế các gói tin có độ dài ngắn hơn nhiều – mặc định là 1518 byte cho phù hợp với độ dài các frame trong mạng cục bộ theo kiểu Ethernet.
- Định danh gói tin, cờ và offset phân đoạn có liên quan đến giao thức truyền TCP mà ta sẽ nói sau. Một bản tin dài sẽ được cắt thành nhiều gói có chung một định danh gói tin. Khi đó cờ sẽ cho biết các gói tin nào là gói tin cuối cùng, gói tin nào chưa phải cuối cùng. Còn offset cho biết gói tin này bắt đầu từ vị trí nào trong bản tin.
- Thời gian sống là số đo thời gian còn lại được phép tồn tại trên mạng trong quá trình lưu chuyển. Khi gói tin mới được tạo lập thời gian sống mặc định được đặt là 255 (giây). Các gói tin được chuyển tiếp qua các router, trước khi chuyển tiếp, thời gian sống được giảm đi thời gian đã chờ trên router, ít nhất cũng giảm đi 1. Nếu thời gian này giảm tới 0 thì gói tin này bị hủy bỏ và một thông báo được chuyển lại nơi phát gói tin này để thông báo.
- Trường giao thức dùng để thông báo giao thức nào đang được sử dụng đối với gói tin này, ví dụ TCP hay UDP.

- Tổng kiểm tra nhằm kiểm soát header được truyền đi có lỗi hay không. Khi nhận được gói tin, nơi nhận sẽ tiến hành tính lại tổng kiểm tra đối với header. Nếu số này không trùng với tổng kiểm tra gửi đi thì gói tin được xem là hỏng.
- Địa chỉ nguồn và địa chỉ đích là các địa chỉ IP của trạm nguồn và trạm đích.

Giao thức chọn đường tĩnh

Ta đã biết, các mạng liên kết với nhau thông qua các máy tính đặc biệt có chức năng dẫn đường gọi là router. Công việc của các router là khi nhận một gói tin nó chuyển gói tin đi đúng kênh cần thiết. Một router có thể có nhiều cổng nối với nhiều mạng khác. Người ta cài đặt sẵn một bảng ở router gọi là bảng chọn đường. Trong bảng chọn đường bao giờ cũng chỉ định một cổng gọi là cổng mặc định (default) sao cho nếu địa chỉ đích không có mặt trong bảng chọn đường thì gói tin sẽ được tự động chuyển theo đường mặc định. Giả sử ta có một mạng cục bộ với địa chỉ lớp C là 247.165.32.*. Ví dụ sau đây cho hình ảnh của một bảng chọn đường của một router có 5 cổng, 4 cổng nối ra ngoài gọi là cổng WAN và một cổng nối vào trong gọi là cổng LAN. Đương nhiên là chỉ có thể dẫn đường đi theo cổng WAN.

Cổng mặc định	Cổng WAN 1
203.195.16.*	Cổng WAN 2
162.34.*.*	Cổng WAN 3
176.15.*.*	Cổng WAN 4
247.165.32.*	Cổng LAN 5

Khi đó nếu router phát hiện thấy có gói tin gửi ra ngoài có địa chỉ đích là 162.34.56.123 sẽ được gửi theo cổng 3; gói tin có địa chỉ đích 203.195.16.234 sẽ được gửi theo cổng số 2 còn nếu gói tin không thuộc nhóm các địa chỉ nói trong cổng 2, 3, 4 và 5 thì sẽ gửi theo cổng mặc định - cổng 1. Thông thường cổng mặc định là cổng gửi lên mạng cấp trên.

Một tình trạng có thể xảy ra là các router thiết lập các đường mặc định thành một vòng kín. Khi đó liệu có một gói tin với địa chỉ vô thừa nhận có thể bị chạy mãi trên mạng hay không. Điều này không xảy ra vì sau một thời gian, thời gian sống của gói tin không còn và gói tin sẽ bị hủy bỏ. Bằng cơ chế trên các gói tin trên mạng Internet được hướng chính xác tới đích. Nhiều hệ thống sử dụng giao thức chọn đường động. Những gói tin mặc dù đến cùng đến một địa chỉ nhưng khi thì router gửi theo đường này, khi thì router gửi theo đường khác tùy theo chi phí và tình hình tại thời điểm xử lý gói tin. Ví dụ khi đường này đang quá tải thì có thể phải chọn đường khác. Ngay trong trường hợp không đường nào quá tải router vẫn có thể gửi đi theo nhiều đường để tăng tốc độ truyền. Vì thế khi ta gửi một thư điện tử trên mạng internet qua Mỹ, rất có thể một phần thư đi qua Australia theo đường vệ tinh, phần còn lại qua Hồng Kông theo cáp quang biển rồi nhập lại với nhau ở một máy tại Mỹ.

1.5.2.2. Giao thức TCP

Nếu IP là giao thức dùng để chuyển tin tức từ máy này đến máy kia (host –to – host) thì TCP cũng là một giao thức liên quan đến phương thức chuyển các gói tin từ một ứng dụng đang chạy trên máy này đến một ứng dụng đang chạy ở máy tính kia mạng kia (end – to – end). Khác với IP, TCP không quan tâm tới vấn đề đường đi (địa chỉ và dẫn đường) mà chỉ quan tâm tới đảm bảo chất lượng của việc truyền tin. Việc kiểm soát truyền giữa các mạng thực hiện ở router, còn việc kiểm soát chất lượng truyền tin thực hiện ở tại các máy tính tham gia truyền thông.

Một bản tin nếu lớn sẽ bị cắt thành các gói tin nhỏ hơn theo một định dạng nhất định. Chính TCP sẽ tạo định danh và đánh số gói tin (như đã thấy trong cấu trúc của gói tin IP) và được gửi tới địa chỉ đích. TCP sẽ kiểm soát luôn cả sự chính xác, nếu một gói tin nào bị hỏng TCP sẽ yêu cầu bên phát gửi lại thông qua chế độ xác nhận. TCP cũng đảm bảo các gói tin

không bị trùng lặp. Khi đủ các gói của bản tin, chính TCP sẽ lắp ráp các gói tin theo đúng thứ tự để khôi phục nguyên dạng bản tin. Ngoài ra TCP còn kiểm soát cả một số chế độ truyền tin như độ khẩn, thông báo về xác nhận, thiết lập kết nối.

1.5.3. Các tài nguyên trên Internet

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

Các dạng thông tin có thể lấy trên Internet bao gồm:

- Các văn bản dạng Text, các sách điện tử dạng file PDF
- Các hình ảnh
- Tài liệu lưu trữ dưới dạng âm thanh (bản nhạc, lời nói, ...)
- Phim video số
- Các phần mềm máy tính.
- Ngày nay việc lấy các tài liệu này được tích hợp qua kết nối trên các trang siêu văn bản (hypertext) trong đó có thể lồng ghép văn bản, ảnh, phim, sách điện tử, và các quá trình tương tác hai chiều như được thực hiện qua một ngôn lập trình web. Đó là tài liệu mà người ta gọi tắt là Web. Các báo điện tử thường là loại này.
- Các thiết bị phân cứng ...

1.5.4. Các dịch vụ cơ bản trên Internet

Có ba nhóm dịch vụ chính trên Internet đó là:

- Trao đổi thông tin giữa các thành viên trong cộng đồng thông qua thư điện tử, diễn đàn, hội thoại mạng, hội nghị từ xa, điện thoại Internet,...
- Khai thác các tài nguyên trên mạng như: tra cứu thông tin qua WEB, FTP, Video theo yêu cầu, ... sử dụng máy tính từ xa.
- Các dịch vụ khác nhờ sử dụng công nghệ Internet.

Dưới đây là chi tiết một số dịch vụ thông dụng:

- Thư điện tử (E mail). Người ta có thể gửi thư đến máy của người khác với giá rẻ hơn rất nhiều so với thư thường mà thời gian trễ không quá một vài phút. Dịch vụ này được sử dụng nhiều nhất trên Internet. Đặc điểm của kiểu trao đổi này là có đối tác cụ thể nhưng không cần có sự hiện diện đồng thời của người đối thoại trên mạng.
- Thâm nhập từ xa (Telnet). Người dùng trên hệ thống này có thể truy cập đến một hệ thống thông qua mạng chỉ cần dùng đến một số lệnh đơn giản. Người sử dụng có một account tại một vị trí trên Internet có thể vào kiểm tra những thư nhắn hoặc truy cập tập tin bằng một máy tính bất kỳ ở một nơi nào đó trên mạng.
- Nhóm tin (News Group) hay còn gọi là diễn đàn (Forum). Có hàng nghìn nhóm tin trên Internet với các chủ đề khác nhau. Nhóm tin là một trong các dịch vụ rất có ích với người dùng. Những ai có vấn đề cần trao đổi đều có thể đưa lên diễn đàn và cùng trao đổi với bất cứ ai tham gia diễn đàn.
- Hội thoại mạng (chatting). Có thể tổ chức một cuộc hội thoại trên mạng máy tính. Nhiều người ngồi tại máy của mình tại nhà mình mà vẫn có thể trao đổi, đàm thoại trực tiếp với

đối tác. Hình thức này mạnh hơn điện thoại ở chỗ: hàng chục người có thể trao đổi đồng thời không chỉ bằng chữ mà còn cả âm thanh và hình ảnh.

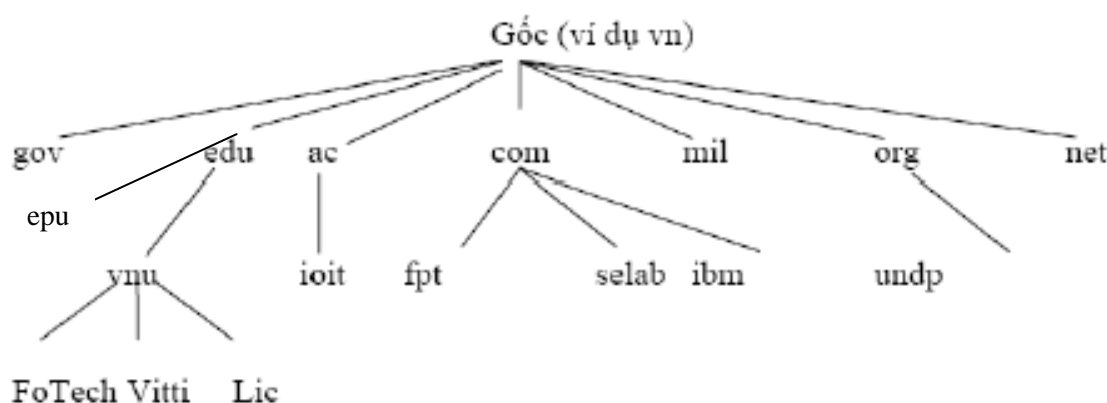
- Điện thoại Internet. Tiếng nói được số hóa và chuyển theo mạng Internet tới người đối thoại. theo cách này giá điện thoại trở nên vô cùng rẻ.
- Hội nghị từ xa (TeleConference) truyền hình ảnh động và âm thanh thu trực tiếp tương tự như cầu truyền hình.
- Thư viện file (Anonymous FTP). Có rất nhiều phần mềm miễn phí, phần mềm dùng chung, tranh ảnh, những tác phẩm văn học, những tập tin dữ liệu khác hoặc tin tức có sẵn trên mạng. Người dùng ở bất cứ nơi nào trên mạng cũng có thể xem những thư viện này và chép những tập tin về máy của mình.
- Phân nhóm theo loại thông tin quan tâm (Usenet). Internet là nơi có hàng nghìn nhóm người sử dụng có những mối quan tâm khác nhau cùng tham gia vào một hệ thống hội thoại từ xa và những nhóm thảo luận. Các nhóm này hình thành từ những quan điểm chung về đề nào đó, những vấn đề giáo dục, các loại nhạc được yêu thích, những sở thích thể thao, những mối quan tâm chính trị, ...tất cả đều phải thể hiện trên máy tính
- World Wide Web (WWW). Đây là dịch vụ quan trọng và tiện dụng nhất hiện nay. Dịch vụ này cho phép lấy các trang siêu văn bản trên mạng Internet. Một trang siêu văn bản có hình thức như một trang tài liệu bình thường gồm các chữ và hình ảnh. Điều khác là trong một số đoạn chữ và hình ảnh có ngầm chứa địa chỉ một trang siêu văn bản khác ở một máy tính khác trên mạng Internet hay một dịch vụ mạng khác. Địa chỉ liên kết đó gọi là siêu liên kết (hyperlink). Nếu ta đưa con trỏ tới đoạn chữ hay hình ảnh có siêu liên kết và nhấn chuột thì tài liệu hay dịch vụ liên kết sẽ được tải về và thực hiện. Cơ chế này cho phép ta tìm tài liệu hay dịch vụ theo các liên kết. Thuật ngữ World Wide Web nghĩa mạng nhện toàn cầu có hàm ý chỉ mỗi liên kết phức tạp này do WWW dễ sử dụng nên ngày càng nhiều dịch vụ Internet tiêu chuẩn như thư điện tử, thư viện file, thậm chí các ứng dụng riêng như các ứng dụng quản lý cũng được tích hợp trực tiếp trên các trang Web. Chính WWW đã gây ra vụ bùng nổ Internet lần thứ hai vì nó tạo cơ hội để ai cũng có thể sử dụng Internet.

1.5.5. Hệ thống tên miền:

Địa chỉ IP không thật thích hợp với người sử dụng vì rất khó nhớ. Người ta sử dụng một hệ thống đặt tên gọi là tên miền (domain name) để đặt tên cho các máy trên mạng. Mỗi tên có thể gồm nhiều trường phân cách nhau bởi một dấu chấm. Theo một quy ước tên được đặt theo một cây phân cấp mà trường đầu tiên là trường địa lý (thường là theo cách viết tắt của tên nước).

Ví dụ: *vn* chỉ Việt Nam,
 th để chỉ Thái lan,
 fr chỉ Pháp,
 jp chỉ Nhật. ...

Các tên miền không có lớp địa lý được hiểu ngầm là Mỹ.



Hình 1.25: Cây tên miền

Ví dụ: tên miền của một số trường như sau:

Trường Đại học Điện lực: `epu.edu.vn`

Khoa Công nghệ, Đại học Quốc gia Hà Nội: `fotech.vnu.edu.vn`

Đại học Bách khoa Hà Nội: `hut.edu.vn`

Trong tất cả các dịch vụ Internet, chỗ nào có địa chỉ IP đều có thể thay bằng tên miền. Trong khi định vị các máy tính trên mạng Internet, để có thể biết chính xác tên miền đó ứng với địa chỉ IP nào, cần có cơ chế giải mã tên miền. Trong các mạng người ta sử dụng một máy tính làm máy chủ thực hiện dịch vụ tên miền. Trong máy đó người ta đưa vào một bảng tương ứng giữa tên miền và địa chỉ IP.

Ví dụ:

www.vnu.edu.vn	172.16.0.168
www.vnn.vn	202.167.121.212

Đương nhiên chính máy chủ tên miền phải có một địa chỉ IP. Các dịch vụ nào muốn giải mã tên miền đều phải khai báo địa chỉ IP của máy chủ cung cấp dịch vụ tên miền này. Các máy chủ tên miền có thể được liên kết với nhau để khi máy chủ tên miền này không hiểu có thể yêu cầu một máy chủ tên miền khác trợ giúp.

Thực tế thì bảng thông tin trong các máy chủ cung cấp dịch vụ DNS phức tạp hơn nó có thể gồm tới 7 loại thông báo khác nhau, mỗi thông báo cũng có nhiều yếu tố chứ không chỉ tương ứng giữa tên miền và địa chỉ IP.

Việc cấp phát tên miền và địa chỉ do một tổ chức phi lợi nhuận, phi chính phủ là NIC (Network Information Center) quản lý. Đại diện của NIC tại khu vực Châu Á Thái Bình Dương là APNIC có trụ sở tại Tokyo. Hiện nay với sự phát triển của Internet không gian địa chỉ IP 4 byte đang cạn kiệt nhanh chóng. NIC đang chuẩn bị đưa ra địa chỉ IP mới 6 byte. Với không gian mới này, trung bình mỗi người dân trên trái đất sẽ có 4 địa chỉ IP. Mỗi quốc gia thường đều có tổ chức quản lý tên miền và địa chỉ của mình. Ở Việt Nam tổ chức đó là VNNIC.

1.5.6. Hệ thống định vị tài nguyên thống nhất URL (Uniform Resource Locator)

Để tiện cho việc truy cập các tài nguyên thông tin trên mạng Internet, người ta quy ước một cách chỉ định nguồn tài nguyên một cách thống nhất viết tắt là URL có cấu trúc như sau:

<i>Giao_thức://</i>	<i>Tên miền hay địa chỉ máy cung cấp tài nguyên:[cổng dịch vụ]</i>	<i>/ Đường dẫn tới file</i>
---------------------	--	-----------------------------

Trong đó:

- Giao thức quy định cách giao tiếp để truy cập đến nguồn tài nguyên. Ví dụ giao thức để truy cập WEB là http (HyperText Transmission Protocol), giao thức truyền tệp là ftp, giao thức tìm tin theo kiểu thực đơn là gopher. . .
- Nơi cung cấp tài nguyên có thể cho bằng tên miền hay địa chỉ IP thực. Nếu cho bằng tên miền thì để truy cập đến máy chủ, hệ thống sẽ gọi dịch vụ tên miền để giải mã địa chỉ.
 - [cổng dịch vụ] là số hiệu cổng dịch vụ trao đổi thông tin.
- Đường dẫn đến tệp có cấu trúc tương tự như đường dẫn đến tệp trong các hệ điều hành. Nó được tính từ gốc theo quan niệm cài đặt các dịch vụ Internet. Trong một số trường hợp, đặc biệt trong tìm kiếm người ta có thể đưa vào các tham số tìm kiếm. Ví dụ để xem trang tin tức thể thao của mạng vnn ta có thể dùng URL

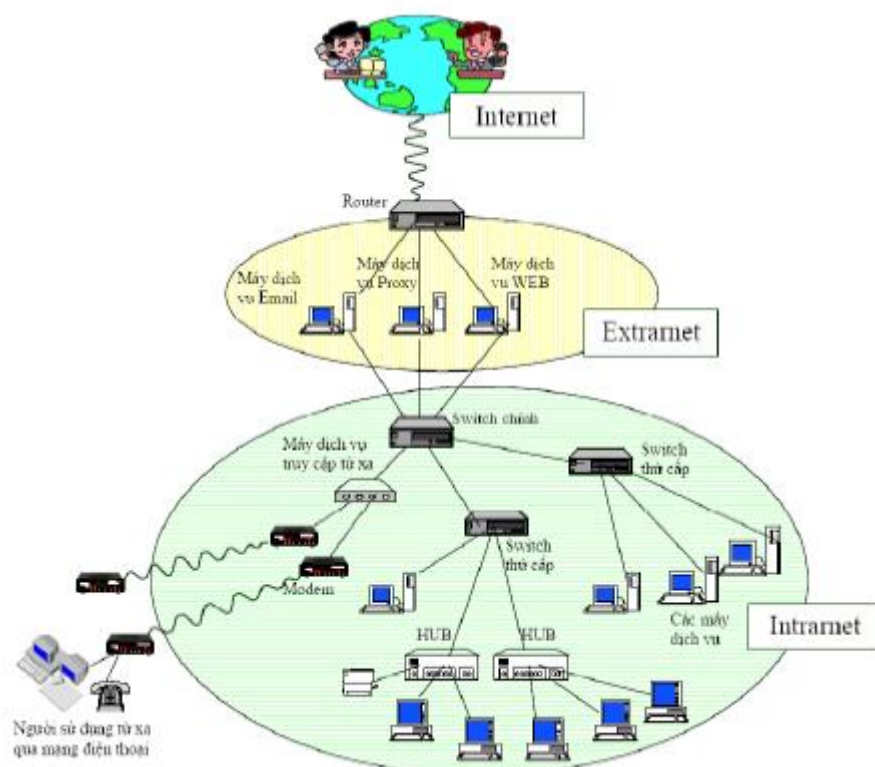
Ví dụ: <http://www.vnn.vn/tintuc/thethao>
 hoặc [http:// 202.167.121.212/tintuc/thethao](http://202.167.121.212/tintuc/thethao)

Để lấy các tệp tin antivirus trong thư viện phần mềm của trong mạng của khoa Công nghệ có thể dùng URL <ftp://www.fotech.vnu.vn/virus/>. Sau khi thư mục virus hiện ra ta có thể nhấp chuột vào biểu tượng tệp tương ứng để lấy về máy tính của mình.

1.5.7.Cấu trúc một mạng diễn hình có nối với Internet

Mạng riêng của một tổ chức nào đó kết nối với nhau theo giao thức TCP/IP gọi là mạng Intranet. Như vậy Intranet có thể là một mạng cục bộ, cũng có thể là mạng diện rộng. Đối với Intranet thì việc kết nối với Internet là việc không phức tạp vì chúng sử dụng chung giao thức truyền thông. Vấn đề được đặt ra đối với mạng Intranet là làm thế nào để kiểm soát được việc kết nối với Internet. Những vấn đề về an ninh mạng là các vấn đề rất phức tạp. Một khi kết nối với Internet, người sử dụng bên ngoài có thể truy nhập được vào trong mạng Intranet và có thể gây ra những hậu quả không mong muốn. Mặt khác cũng cần phải kiểm soát đối với những người ở trong mạng Intranet truy cập ra bên ngoài.

Một vấn đề khác được đặt ra là có một số dịch vụ vừa phải sử dụng trong Intranet, vừa cho phép sử dụng từ Internet. Các dịch vụ email hay dịch vụ WEB là hai trong số các ví dụ điển hình. Để giải quyết vấn đề này, người ta dùng một mạng có một số máy chủ cung cấp dịch vụ vừa nối với các máy bên trong, vừa nối với Internet. Các máy này đóng vai trò trung gian đảm bảo giao tiếp giữa Intranet và Internet cũng như cung cấp dịch vụ cho cả hai phía. Chúng hình thành một mạng đệm giữa hai khu vực và gọi là mạng Extranet.



Hình 1.26: Sơ đồ một mạng nội bộ (Intranet) điển hình có kết nối với Internet thông qua Extranet

Bài đọc thêm. Sự hình thành và phát triển của Internet

Thời kỳ phôi thai

Năm 1957 Liên Xô phóng vệ tinh nhân tạo đầu tiên Sputnik. Sự kiện này khiến Mỹ phải có đối sách để không bị lạc hậu trong lĩnh vực công nghệ cao phục vụ quốc phòng. Mỹ đã thành lập Cơ quan dự án nghiên cứu các vấn đề cao cấp (Advanced Research Projects Agency - ARPA) thuộc Bộ quốc phòng Mỹ (DOD) nhằm phát triển khoa học và công nghệ cao phục vụ cho quân sự.

Năm 1969 Bộ Quốc phòng Mỹ đã xây dựng dự án ARPANET để nghiên cứu lĩnh vực mạng, theo đó các máy tính được liên kết với nhau và có khả năng tự định đường truyền tin ngay khi một phần mạng đã bị phá hủy trong một cuộc chiến tranh. Năm 1972 diễn ra hội nghị quốc tế về truyền thông máy tính. Ở đó Bob Kahn đã trình diễn mạng ARPANET để liên kết 40 máy thông qua các bộ xử lý giao tiếp giữa các trạm cuối (Terminal Interface Processor - TIP). Cũng năm này nhóm InterNET Working Group (INWG) do Vinton Cerf làm chủ tịch ra đời nhằm đáp ứng nhu cầu thiết lập giao thức bắt tay (agreed-upon).

Năm 1972 cũng là năm Ray Tomlinson của BBN đã phát minh ra e-mail để gửi thông điệp trên mạng. Suốt 20 năm liền, Email là một trong những dịch vụ được dùng nhiều nhất.

Năm 1973, các đại học University College of London (Anh) và của Royal Radar Establishment (Na Uy) kết nối vào ARPANET. Cũng vào thời gian ở đại học Harvard, Bob Metcalfe đã phác họa ra ý tưởng về Ethernet (một giao thức trong mạng cục bộ) và Bob Kahn đưa ra vấn đề Internet, khởi đầu chương trình nghiên cứu liên mạng tại ARPA. Tháng 3/1973,

Vinton Cerf phác thảo ra cấu trúc gateway. Sau đó 6 tháng, tháng 9/1973 Vinton Cerf và Bob Kahn trình bày những ý tưởng cơ bản của Internet tại INWG ở University of Sussex, Brighton, Anh. Đó chính là những nét chính của giao thức TCP/IP. Năm 1974, BBN đã xây dựng giao thức ứng dụng Telnet cho phép sử dụng máy tính từ xa. Năm 1975, điều hành hoạt động của Internet được chuyển cho DCA (hiện nay là DISA).

Năm 1976, AT&T Labs phát minh ra dịch vụ truyền tệp qua mạng FTP.

Tháng 7/1977, lần đầu tiên trình diễn ARPANET/Packet Radio Net/SATNET theo giao thức Internet trên những gateway do BBN cung cấp.

Năm 1978, Tom Truscott và Steve Bellovin thiết lập mạng USENET dành cho những người sử dụng UNIX. Mạng USENET là một trong những mạng phát triển sớm nhất và thu hút nhiều người nhất. Trên USENET hình thành dịch vụ News Groups là một trong những dịch vụ phát triển mạnh của Internet sau này.

Năm 1979 ARPA thành lập Ban Kiểm soát Cấu hình Internet (Internet Configuration Control Board - ICCB).

Năm 1981 mạng CSNET (Computer Science NETwork) do nhiều nhà khoa học máy tính phối hợp với các trường đại học University of Delaware, Purdue, University of Wisconsin, công ty RAND và BBN lập nên nhờ tài trợ của NSF. CSNET cung cấp các dịch vụ mạng cho các nhà khoa học ở trường đại học mà không cần truy cập vào mạng ARPANET. CSNET sau này được xem như mạng phục vụ cho khoa học và máy tính.

Năm 1982 giao thức TCP (Transmission Control Protocol) và IP (Internet Protocol) được DAC và ARPA dùng đối với mạng ARPANET. Sau đó DOD tuyên bố chọn TCP/IP là giao thức chuẩn. Kết nối trực tiếp giữa các nước Hà Lan, Đan Mạch, Thụy Điển và Anh được thực hiện, tiếp theo đó truy nhập vào ARPANET được miễn phí.

Năm 1983 ARPANET được tách ra thành ARPANET và MILNET. MILNET tích hợp cùng với Mạng dữ liệu quốc phòng (Defense Data Network). ARPANET trở thành một mạng dân sự. Ban hoạt động Internet (Internet Activities Board - IAB) ra đời thay thế cho ICCB. Cũng vào năm này quỹ khoa học Quốc gia Mỹ NSF (National Science Foundation) tài trợ cho việc xây dựng NSFnet.

Năm 1984 giới thiệu Domain Name Server (DNS), số lượng máy chủ vượt quá 1000.

Thời kỳ bùng nổ lần thứ nhất của Internet

Năm 1986 mạng NSFnet chính thức được thiết lập (tốc độ của Backbone là 56 Kbps) kết nối 5 trung tâm máy tính toán cung cấp từ xa. Đây cũng là năm có sự bùng nổ kết nối, đặc biệt là ở trường đại học. Như vậy là NSF và ARPANET song song tồn tại theo cùng một giao thức, có kết nối với nhau.

Năm 1987 Version 2 của NSFnet ra đời với hơn 100.000 máy tính tham gia, 3400 trung tâm nghiên cứu được kết nối, tốc độ truyền 45 triệu bit/giây.

Năm 1988 một số vùng của Canada, Đan Mạch, Phần Lan, Pháp, NA Uy, Thụy Điển nối vào NSFnet.

Năm 1989 số lượng máy chủ vượt quá 100.000, mạng EUnet (Châu Âu) và AUSSIBnet (Úc) gia nhập mạng Internet. Đức, Israel, Ý, Nhật, Mexico, Hà Lan, New Zealand, Puerto Rico, U.K nối vào NSFnet, nhưng các doanh nghiệp không được sử dụng NSFnet tuy nhiên họ có thể tham gia mạng ARPANET.

Năm 1990, với tư cách là một dự án ARPANET ngừng hoạt động nhưng mạng do NSF và ARPANET tạo ra đã được sử dụng trong mục đích dân dụng, đó chính là tiền thân của mạng Internet ngày nay. Một số hãng lớn bắt đầu tổ chức kinh doanh đã tổ chức kinh doanh trên mạng, trong đó có mạng AOL (American On Line) nổi tiếng hiện nay. Các nước Argentina, Áo, Bỉ, Brazil, Chi Lê, Ấn Độ, Ireland, Hàn Quốc, Tây Ban Nha, Thụy Sĩ nối vào NSFnet. Đến lúc này đối tượng sử dụng Internet chủ yếu là những nhà nghiên cứu và dịch vụ phổ biến nhất là Email và FTP. Internet chưa phải là một phương tiện đại chúng. Có thể là có các lý do sau:

- Mặc dù máy vi tính đã ra đời nhưng chưa được phổ biến,
- Giao tiếp với Internet chưa thật thuận tiện,
- Các cơ sở dữ liệu trên Internet còn nghèo nàn.

Bùng nổ lần thứ 2 với sự xuất hiện của WWW

Năm 1991 Tim Berners Lee ở trung tâm nghiên cứu nguyên tử Châu Âu (CERN) hát minh ra World Wide Web(WWW) dựa theo một ý tưởng về siêu văn bản được Ted Nelson đưa ra từ năm 1985. Có thể nói đây là một cuộc cách mạng trên Internet vì người ta có thể tra cứu một cách dễ dàng các thông tin theo mối liên hệ. Cũng vào thời gian này NSFnet backbone được nâng cấp đạt tốc độ 44736 Mbps. NSFnet truyền một tỷ byte / tháng và 10 tỷ gói tin / tháng. Các nước Croatia, CH Séc, Hong kong, Hungary, Bồ Đào Nha, Singapore, Nam Phi, Đài Loan, Tunisia nối vào NSFnet.

Năm 1992 Internet Society bước vào hoạt động, số lượng máy chủ vượt quá con số 1000000. IAB trở thành một thành phần của Internet Society. Các nước Cameroon, Cyprus, Ecuador, Estonia, Kuwait, Latvia, Luxembourg, Malaysia, Slovakia, Slovenia, Thailand, Venezuela nối vào NSFnet.

Năm 1993 NSF cho ra đời InterNIC, cung cấp các dịch vụ Internet như: Dịch vụ về cơ sở dữ liệu và thư mục (AT & T), dịch vụ đăng ký (Network Solution Inc.) , dịch vụ thông tin (General Atomics/CERFnet), liên hiệp quốc trực tuyến (UN). Các nước Bungary, Costa Rica. . . nối vào mạng NSFnet nâng tổng số các nước tham gia mạng Internet lên hơn 59 quốc gia.

Năm 1994 là năm kỷ niệm lần thứ 25 ra đời ARPANET, NIST (The National Institute for Standards and Technology) đề nghị thống nhất TCP/IP và bỏ yêu cầu chỉ dùng chuẩn OSI. WWW đã trở thành dịch vụ phổ biến thứ hai sau dịch vụ FTP. Những hình ảnh Video đầu tiên được truyền đi trên mạng Internet.

Năm 1995 NSF kết thúc việc tài trợ và NSFnet thu lại thành một mạng nghiên cứu. Trong 3 tháng WWW vượt trội hơn FTP và trở thành một dịch vụ có sự lưu thông lớn nhất căn cứ trên số lượng gói tin truyền. Các hệ thống quay số trực tuyến truyền thống như Compuserve, AmericanOnline, Prodigy bắt đầu cung cấp khả năng tiếp cận mạng Internet. Số các máy tính kết nối với mạng Internet là khoảng 3, 2 triệu với 42 triệu người dùng từ 42000 mạng máy tính khác nhau trên 84 nước trên toàn thế giới. Triển lãm Internet 1996 World Exposition là triển lãm thế giới đầu tiên trên mạng Internet.

Tính đến 7/1997 đã có 171 nước tham gia Internet với 19.500.000 máy chủ kết nối vào mạng. Hiện nay có hơn 300 triệu người dùng Internet thường xuyên! Dự tính đến 2004 sẽ có 900 triệu người sử dụng Internet Chính Internet là môi trường thông tin mà xã hội thông tin chờ đợi. Điều mà kỹ thuật còn tiếp tục phải giải quyết là năng lực truyền thông của các mạng viễn thông

công cộng. Đó là nội dung kỹ thuật của cái gọi là “siêu xa lộ thông tin” (information superhighway) mà các nước phát triển đang đầu tư quyết liệt. Trong các nước phát triển, nhờ InterNet mà các dịch vụ tại nhà như giáo dục từ xa, mua hàng, tư vấn y tế, và rất nhiều điều khác đã trở thành hiện thực. Người ta thấy rằng InterNet nhân gấp bội hiệu quả lao động trí tuệ của con người. Vì vậy ở các nước phát triển không Đại học nào lại không tận dụng InterNet. Hầu hết các nước miễn phí cho sinh viên, cán bộ giảng dạy, các nhà khoa học (phí ở đây bao gồm chi phí đường truyền - thường là các công ty bưu điện là chủ và phí khai thác các cơ sở dữ liệu - vì phải có chi phí tạo ra thông tin đưa lên máy).

Internet vào Việt Nam như thế nào

Internet đặt chân đến Việt Nam phần nào nhờ vào sự tình cờ, nói đúng hơn là nhờ có sự nhiệt tình và tâm huyết của ông Rob Hurle cũng như các đồng nghiệp tại Đại học quốc gia Australia.

Tháng 3/2003 những người tham gia thiết lập kết nối với Internet đã gặp nhau tại Hà Nội để nhớ lại sự kiện 10 năm trước, Internet đã vào Việt Nam. Nhân một chuyến đi tới làm việc với khoa Sử của Đại học Tổng hợp Hà Nội năm 1992, ông Rob đã đến thăm một số sinh viên công nghệ thông tin (CNTT) của Việt Nam từng nghiên cứu tại Đại học Quốc gia Australia. Điều khiến ông Rob lưu ý nhất trong những cuộc gặp gỡ là sinh viên Việt Nam rất có nhu cầu được tiếp cận Internet. Khi còn học tại Đại học Quốc gia Australia, những sinh viên này sử dụng máy tính lớn để nghiên cứu, nhưng khi về nước, công việc của họ phải dừng lại. Điều này kích thích ông Rob tiến hành thử nghiệm kết nối. Khi về Australia, ông đã ghép nối một máy chủ với dây dẫn điện thoại và thử quay số kết nối tới Việt Nam, ông thấy nó hoạt động và những người sử dụng máy tính tại đầu dây ở Hà Nội có thể truy cập vào các máy tính lớn ở Australia. Trước khi nhóm nghiên cứu của ông Trần Bá Thái tại Viện CNTT Việt Nam bắt tay cùng ông Rob tiến hành thử nghiệm kết nối, Viện cũng từng nỗ lực thiết lập kết nối Internet với các trường đại học châu Âu, nhưng không thu được kết quả. Cuối năm 1992, một account được tạo ra tại Trung tâm tính toán của Đại học Quốc gia Australia (CCU), nơi có rất nhiều nhóm nghiên cứu về CNTT, trong đó Rob Hurle phụ trách một nhóm. Công việc “quá cảnh” cho các e-mail được tiến hành mỗi khi đường dây điện thoại được kết nối giữa Viện CNTT và Đại học Quốc gia Australia. Mọi cuộc gọi đều được thanh toán tại đầu dây Australia vì cước viễn thông tại Việt Nam đắt hơn nhiều lần. Chế độ callback được thiết lập để khi điện thoại gọi đến Australia, máy tại Australia lập tức cắt cuộc gọi và tiến hành gọi ngược lại để chính Đại học Quốc gia Australia trả tiền. Về sau, khi mạng VARENet ra đời, người sử dụng dịch vụ e-mail tại Việt Nam phải nộp phí, và khoản tiền đó được gửi trả lại Australia. Những người phụ trách giữa hai đầu kết nối là Rob Hurle và Trần Bá Thái. Khi e-mail được gửi tới máy chủ tại CCU, họ đã phải tiếp nhận, phân loại và đẩy những bức thư đó tới Việt Nam qua đường kết nối điện thoại. Tại Viện CNTT, Trần Bá Thái lặp lại quá trình tương tự. Chu trình hoạt động bán thủ công này diễn ra một thời gian cho tới tháng 9/1993, khi Công ty viễn thông Telstra của Australia để ý đến hoạt động này và tổ chức một cuộc hội thảo về kết nối máy tính tại Hà Nội. Kết quả của cuộc hội thảo, với sự có mặt của ông Rob, là quá trình kết nối được tự động hoá nhờ sử dụng các giao thức UUCP (Unix to Unix Copy). Giao thức UUCP tỏ ra rất hữu hiệu và mạnh, đặc biệt khi ứng dụng trên nền tảng cơ sở hạ tầng viễn thông yếu của Việt Nam lúc bấy giờ. Đồng thời, việc triển khai chủ đòi hỏi chi phí thấp và dễ dàng thực hiện.

Năm 1994, nhóm phát triển kết nối này nhận được khoản tài trợ của Bộ Giáo dục, khoa học và Đào tạo Australia. Vào tháng 4, Viện CNTT và Bộ Khoa học, Công nghệ và Môi trường đạt được thoả thuận đăng ký tên miền .vn với tổ chức quản lý tên miền Internet quốc tế. Nếu việc này không được tiến hành khẩn trương thì Việt Nam sẽ gặp những khó khăn về tên miền quốc gia giống như Philippines (tên miền .ph của Philippines đã bị người khác đăng ký mua trước).

Sau khi việc đăng ký tên miền hoàn tất, mạng VARENet (Vietnam Academic Research and Educational Network) ra đời với 9 đường điện thoại và các đầu dây dừng tại Đại học Quốc gia Australia. Trong thời gian đầu, e-mai được gửi 5 lần/ngày từ Australia về Hà Nội. Sau này, Viện CNTT thiết lập hệ thống phân phát thư tự động VARENet, bắt đầu tăng số lượng các đầu nút kết nối từ TP HCM đến Nha Trang, Huế và Hải Phòng. Đến năm 1996, có khoảng 300 điểm được kết nối với Viện CNTT tại Hà Nội nhờ sử dụng modem, mạng điện thoại công cộng và các máy tính có cài giao thức mạng UUCP.

Người sử dụng thuộc các tổ chức khác nhau, như Viện Toán học Hà Nội, Viện Vật lý, Đại học Quốc gia Hà Nội, Đại học Bách khoa Hà Nội và Ủy ban về Chất độc màu da cam. Các đầu mỗi được chia thành 5 loại: ac (nghiên cứu), edu (giáo dục), gov (chính phủ), org (các tổ chức) hoặc com (thương mại). Đến giữa 1998, VARENet đã thu hút gần 1.500 nhóm khách hàng và khoảng 4.000 người dùng cá nhân. Trên thực tế, các tên miền cấp hai trên (ac.vn, edu.vn, gov.vn và com.vn) được quản lý tại Australia. Mãi đến năm 2000, việc chuyển giao toàn bộ quyền quản lý tên miền cho Việt Nam mới được hoàn tất.

Internet ở Việt Nam

Việt Nam bắt đầu thử nghiệm kết nối với Internet từ 1992. Đến 1997 Việt Nam chính thức tham gia Internet. Chính phủ đã ra nghị định 21/CP về quy chế sử dụng Internet. Theo đó có 5 chủ thể tham gia Internet. IAP (Internet Access Provider) - người cung cấp dịch vụ đường truyền để kết nối với internet, quản lý cổng (gateway) nối với quốc tế.

Hiện nay đơn vị duy nhất được làm IAP là Công ty Dịch vụ truyền số liệu VDC thuộc Tổng Công ty Bưu chính Viễn thông. Các ISP (Internet Service Provider) - người cung cấp các dịch vụ Internet. Các ISP phải thuê đường và cổng của một IAP (hoặc của một IXP mà ta sẽ nói sau). Các ISP có quyền kinh doanh thông qua các hợp đồng cung cấp dịch vụ Internet cho các tổ chức và các cá nhân.

Tính đến 5/2002 ở Việt Nam đã có 12 ISP trong đó có:

- Công ty dịch vụ truyền số liệu VDC của Tổng Công ty Bưu chính Viễn thông, Đây là ISP lớn nhất.
- Công ty FPT thuộc Bộ Khoa học Công nghệ và Môi trường, công ty tin học lớn nhất của Việt Nam.
- NETNAM thuộc Viện Công nghệ Thông tin. Đây là tổ chức hoạt động có bề dày trong lĩnh vực Công nghệ Thông tin nói chung và Internet nói riêng với thiết bị và công nghệ hoàn hảo. NETNAM là nơi đầu tiên thực hiện kết nối Internet
- Saigon Postel là công ty cổ phần bưu điện của TP Hồ Chí Minh.
- Viễn thông quân đội VietTel.
- Viễn thông điện lực.

1.6. Một số vấn đề về tội phạm Tin học và đạo đức nghề nghiệp^[2]

1.6.1 Tin tặc - một loại tội phạm kỹ thuật

Từ nhiều năm nay, cùng với sự phát triển nhanh chóng của tin học, tội phạm tin học cũng gia tăng rất nhanh. Tác giả của tội phạm tin học về mặt kỹ thuật mà ta gọi là “tin tặc” (hacker) thường là những người rất giỏi về tin học. Sau đây là một số loại hình tội phạm tin học thường gặp.

1.6.1.2. Virus và sâu tin học

Virus được biết đến lần đầu tiên vào 1985 từ một nghiên cứu của Fred Cohen. Virus là những chương trình được viết theo một cơ chế đặc biệt có những tính năng như sau:

- Virus có khả năng lây lan, khi lọt vào một máy nó chiếm quyền điều khiển máy để tự nhân bản nhằm lây lan từ máy này sang máy khác. Chính vì tính năng tương tự với virus sinh học này mà người ta gọi các chương trình này là virus.
- Virus là các chương trình tương đối nhỏ, hiệu quả cao và thường có các cơ chế chống phát hiện.
- Cuối cùng, virus có mục đích gây nhiễu hoặc phá hoại. Những virus “hiền” thường chỉ gây nhiễu chứ không phá hủy dữ liệu, ví dụ virus Yankee Doodle, cứ đúng 17 giờ là tạm dừng máy để phát bản quốc ca Mỹ hay virus “Thứ 6 ngày 13” thì cứ đến thứ 6 hoặc ngày 13 (ngày nghỉ của những người theo đạo Hồi) thì không cho máy làm việc. Có những virus hiện lên dòng chữ “tôi đói” và ai đánh đúng chữ “Cookie” nghĩa là bánh bích quy thì nó cho máy tính làm việc tiếp. Những virus “dữ” thì làm hỏng các phần mềm khác trong máy hoặc làm hỏng các tệp dữ liệu. Có những virus tiến hành định dạng lại (format) đĩa cứng và huỷ toàn bộ thông tin có trên đĩa. Năm 1999 virus Chec-nô-bun của một sinh viên Đài loan đã gây tác hại cho hàng trăm ngàn máy tính trên thế giới gây thiệt hại nhiều tỉ đô la. Virus này không những format đĩa cứng mà còn xoá các chương trình điều khiển của máy tính trong bộ nhớ flash, khiến phải thay lại bảng mạch của máy tính hoặc nạp lại chương trình điều khiển. Virus “I love you” của một sinh viên Philippine năm 2000 lây lan qua đường thư điện tử cũng gây thiệt hại nhiều tỉ đô la. Năm 2001 người ta được chứng kiến những loại virus gây tê liệt những mạng lớn bằng cách gây quá tải như virus Code Red hay Nimda trong tháng 9/2001. Cơ chế lây lan được tin tặc tính toán rất kỹ. Cho tới nay, người ta biết tới bốn loại viurs, loại virus tệp nhiễm vào các tệp chương trình, loại virus boot nhiễm vào vùng khởi động của đĩa, virus macro lây qua các tệp tin văn bản và sâu (worm) một loại chương trình hoàn chỉnh thường lây qua mạng.
- Virus tệp: Với vius tệp, khi cho chạy chương trình đã nhiễm virus, virus sẽ phát tác. Thông thường virus sinh ra một đoạn mã thường trực trong bộ nhớ và chiếm lấy điều khiển tệp của hệ điều hành. Như vậy máy đã bị nhiễm virus. Khi chạy một chương trình chưa bị nhiễm virus, hệ điều hành do bị virus chiếm quyền trước sẽ không thi hành ngay chương trình mà ghép thêm đoạn mã virus vào chương trình đó, ghi lại lên đĩa sau đó mới cho thi hành. Như vậy chương trình vừa chạy đã bị nhiễm virus. Nếu ta mang chương trình đã bị nhiễm đem chạy ở máy khác thì virus sẽ lây tiếp sang máy khác. Virus tệp để lại một dấu vết dễ nhận là sau khi bị nhiễm, kích thước của tệp lớn thêm.

- Virus boot: Mỗi một đĩa (cứng hay mềm) đều dùng các sector đầu tiên để mô tả các thông số của đĩa và có một chương trình nhỏ giúp khởi động hệ điều hành. Khi đặt một đĩa vào ổ, máy tính sẽ đọc các thông tin đó và thi hành chương trình khởi động nếu máy trong trạng thái khởi động. Cơ chế này bị các tin tặc lợi dụng để phát tán virus. Khi đặt một đĩa đã bị nhiễm virus boot vào một máy tính rồi đọc, virus sẽ sinh ra một đoạn mã thường trực trong bộ nhớ và chiếm lấy điều khiển tập tương tự nhưng virus tập. Nếu ta đặt một đĩa mới vào máy, virus sẽ thay lại vùng đĩa khởi động của đĩa bằng một nội dung khác có mã của virus. Khi đem đĩa đến một máy khác virus sẽ được giải phóng để hoàn thành một chu kỳ lây lan.
- Virus macro: Trước đây ít khi người ta nghĩ đến khả năng các tài liệu văn bản cũng có thể là môi trường lây lan virus. Trong các tài liệu theo chuẩn của Microsoft có một cơ chế tự động thực hiện hàng loạt các công việc theo một kịch bản định sẵn gọi là macro. Ví dụ một người soạn tài liệu toán học, để đưa vào một dấu tích phân phải thực hiện khoảng 10 thao tác. Nếu họ tạo ra một kịch bản quy định nếu gõ phím Ctrl_I thì cả 10 thao tác đó sẽ được thực hiện một cách tự động thì rất tiện. Phương tiện tạo macro của Microsoft có thể cho máy học các kịch bản, sau đó có thể ghi lại kịch bản đó cùng với tài liệu và lưu lại trong máy để tự động dùng lại. Microsoft còn tạo ra cả một ngôn ngữ lập trình để xây dựng các kịch bản phức tạp, kể cả những hoạt động xóa hay sửa tệp. Chính vì thế macro được dân tin tặc tận dụng làm môi trường lây lan virus. Khi một tài liệu bị nhiễm đưa sang máy khác soạn thảo nó sẽ làm cho máy mới ghi lại kịch bản của virus. Virus macro đã hoành hành suốt một thời gian dài cho tới khi Microsoft đưa vào các phần mềm văn phòng của mình chức năng cảnh báo có macro để người dùng cảnh giác.
- Sâu (WORM): Gần đây tin tặc sử dụng Email để phát tán virus với một tốc độ và quy mô rất lớn. Virus được gửi kèm theo Email dưới dạng các tệp chương trình kèm theo được nguy trang. Các virus khi phát tác sẽ tìm trong hộp thư của máy bị nhiễm lấy danh sách địa chỉ của những người có trao đổi thư điện tử với đương sự và gửi lại những thư bức thư có mang virus. Về cơ bản worm là virus tệp được gắn với cơ chế phát tán tích cực mà email hay web chỉ là phương tiện phát tán. Nhiều người phân biệt một cách rạch ròi giữa WORM và virus dựa theo sự tương tự với cơ chế sinh học. Cũng giống như virus sinh học, virus tin học chỉ sống được nếu ký sinh được trên vật chủ. Các đoạn mã của virus tin học phải gắn vào trong chương trình hoặc vùng boot và phát tác khi chạy chương trình lây nhiễm hoặc đọc vùng boot của đĩa. Sâu tin học cũng như sâu sinh học là một thực thể hoàn chỉnh tự hoạt động mà không cần ký sinh vào một vật chủ nào mới có thể phát triển được. Sự nguy hiểm của virus hay sâu lây lan qua mạng là tốc độ lây lan với quy mô tăng theo hàm số mũ. Năm 2001, sâu Melissa với cơ chế phát tán qua email chỉ sau hai ngày đã lây lan ra toàn thế giới làm hàng chục triệu người khốn đốn. Sâu “Code Red” năm 2001 thì không dùng email mà tận dụng một lỗ hổng an ninh của phần mềm WEB server IIS để phát tán làm tê liệt rất nhiều hệ thống máy tính. Thường thì môi trường Windows được tin tặc quan tâm nhất vì cộng đồng người dùng Windows lớn nhất hiện nay. UNIX ít phổ cập hơn và dù sao UNIX cũng là môi trường chuyên nghiệp do đó dân tin tặc ít chọn UNIX làm môi trường hoạt động hơn Windows. Tuy vậy không phải là không có virus trên môi trường này. Chúng ta đã biết nhiều cuộc tấn công lớn nhằm vào những trung tâm máy tính rất nhạy cảm như Bộ Quốc phòng, Cơ quan hàng không vũ trụ Mỹ (NASA). Hệ thống máy chủ của các cơ quan

này chủ yếu là UNIX. Hiện nay UNIX đang được chuyển xuống PC mà LINUX đang là một hiện tượng. Đã bắt đầu xuất hiện virus tấn công LINUX.

1.6.1.2. Mạo danh, xâm nhập máy trái phép và đánh cắp và huỷ hoại thông tin

Các hệ thống máy tính thường được bảo vệ cẩn thận nhưng không có loại khoá nào có thể an toàn tuyệt đối trước sự tấn công của tin tặc. Thông thường các máy tính được bảo vệ tầng đầu tiên bằng tên đăng nhập và mật khẩu, chỉ khi gõ vào đúng tên và mật khẩu mới có thể đăng nhập hệ thống được.

Dân tin tặc cũng có rất nhiều cách để lấy trộm mật khẩu. Phương thức đơn giản nhất là chúng cho chạy một chương trình có màn hình giống hệt màn hình mời đăng nhập. Người sử dụng tưởng là đăng nhập gõ tên và mật khẩu. Chương trình này thu lại để ở một chỗ nào đó hoặc ngầm gửi thông tin ăn cắp được theo email đi, sau đó tự huỷ. Dĩ nhiên người sử dụng không biết là đang đăng nhập giả, họ chỉ nghĩ rằng có một trục trặc gì đó của máy tính và đăng nhập lại. Lần này thì mọi việc bình thường nhưng mật khẩu thì đã bị lộ. Một cách khác là dùng các chương trình sinh tên và mật khẩu ngẫu nhiên và thử một cách bền bỉ, khi nào thấy đăng nhập được thì ghi lại. Cũng có thể ăn trộm mật khẩu bằng cách bắt các gói tin của mạng để phân tích. Khi nào thấy gói tin liên quan đến trao đổi để xác thực trên mạng thì phân tích lấy ra mật khẩu. Việc này chỉ làm được đối với các hệ điều hành mạng yếu, không mã hoá tốt các giao dịch xác thực trên mạng, hoặc chính hệ điều hành đã có cơ chế mã hoá giao dịch nhưng người sử dụng không dùng.

Một cách khác khôn ngoan hơn là đưa virus vào máy bằng một cách nào đó. Sau khi xâm nhập được, các virus sẽ lấy mật khẩu gửi lại cho tin tặc bằng Email. Thời gian qua ở Việt Nam bằng cách này rất nhiều người bị lộ mật khẩu. Tin tặc (phần lớn là sinh viên CNTT) còn lập hẳn một WEBSITE trên mạng Internet đặt tại Mỹ để công bố các mật khẩu đánh cắp được. Nhiều thuê bao Internet ở Việt Nam chỉ trong một thời gian ngắn đã mất hàng chục triệu đồng tiền thuê bao Internet vì những kẻ mạo danh. Đối với các mạng riêng, sau khi lọt được vào mạng các tin tặc có thể lấy các thông tin bên trong và cũng có thể sửa đổi hoặc xoá tệp.

1.6.1.3. Tấn công gây tê liệt

Một loại hình tội phạm nữa là tấn công vô hiệu hoá hoạt động của hệ thống máy tính bằng cách làm hệ thống quá tải. Chúng lập một chương trình liên tiếp gửi các thông điệp đến hệ thống bắt hệ thống phải trả lời với một nhịp độ cao đến mức hệ thống không còn làm được bất kỳ việc gì khác nữa ngoài trả lời các thông điệp phá rối này. Có hai phương tiện mà các tin tặc thường dùng.

Trong các giao thức của Internet có giao thức ICMP, dùng để kiểm tra hai máy có giao tiếp được với nhau hay không. Khi một máy tính này gửi đến máy kia một gói tin của giao thức ICMP với hàm ý thử “có nhận được không” thì máy kia khi nhận được sẽ phát trả lời một gói tin trả lời với hàm ý “nhận được”. Dân tin tặc có thể viết một chương trình mỗi giây phát đi hàng vạn gói tin như thế tới một máy để máy này suốt ngày chỉ làm mỗi một việc là phát trả thông báo “đã nhận được” câu hỏi vô nghĩa kia. Cách thứ hai bọn tin tặc thường dùng là “dội bom” các hệ thư. Chúng cũng dùng các chương trình mỗi giây gửi hàng nghìn thư đến máy chủ thư. Các máy

chủ kiểm tra địa chỉ thư, nếu đúng thì phải mất thời gian ghi lại một thư vô nghĩa, nếu sai thì phải phúc đáp lại nơi phát rằng thư không có người nhận. Chỉ cần một PC không mạnh lắm đội bom cũng có thể vô hiệu hoá hoàn toàn một máy chủ cung cấp dịch vụ thư.

1.6.2. Các tội phạm lạm dụng Internet vì những mục đích xấu

Loại tội phạm này khác với tội phạm của tin tặc (là những kẻ phá hoại bằng kỹ thuật cao) mà liên quan tới nội dung thông tin. Có thể kể đến các loại sau.

1.6.2.1. Phát tán hoặc gieo rắc các tài liệu phản văn hoá, vi phạm an ninh quốc gia

Internet là môi trường công cộng, ai cũng có thể sử dụng. Một số người lợi dụng khả năng của Internet để phổ biến các tài liệu phản văn hoá như kích động bạo lực, phổ biến văn hoá đồi trụy, kích động bạo loạn, gây rối, kích động các xu hướng dân tộc hay tôn giáo cực đoan, hướng dẫn các phương pháp khủng bố. Trên Internet có tới hàng vạn WEBSITE có nội dung xấu kiểu này. Một số người 'lợi dụng' thư điện tử, chủ động gửi đến những tài liệu kiểu đó cho người khác. Trong thời gian vừa qua, nhiều người Việt Nam ở trong nước thường xuyên nhận được thư điện tử với nội dung xấu của các tổ chức phản động ở nước ngoài.

1.6.2.2. Vi phạm đời sống riêng tư của người khác

Có những người lạm dụng mạng để quấy rối, đe dọa, xúc phạm đến người khác. Có những người bị một kẻ khác mạo danh đưa ra các tuyên bố gây thiệt hại. Có nhiều công ty bằng cách nào đó lấy được địa chỉ thư, liên tục gửi đến các thư quảng cáo sản phẩm hay dịch vụ của công ty mình. Điều này gây ra rất nhiều phiền toái, mỗi khi mở thư điện tử phải xoá hàng trăm thư quảng cáo và những thư không mong đợi. Việc lạm dụng thư điện tử để quảng cáo gọi là “nhồi thư” (spamming). Nhiều nước đang xem xét những đạo luật liên quan đến spamming có được phép hay không.

1.6.3. Sở hữu trí tuệ và bản quyền

1.6.3.1. Tình trạng vi phạm bản quyền phần mềm

Tình trạng dùng phần mềm sao chép không có bản quyền rất phổ biến không chỉ riêng ở các nước đang phát triển. Ngay ở Mỹ cũng có đến 1/3 số phần mềm được dùng không có bản quyền. Nếu tình trạng này không kiểm soát được thì các công ty làm phần mềm không thể bán được sản phẩm và không thể tái đầu tư được. Theo thống kê của các tổ chức có trách nhiệm tình trạng dùng phần mềm không có bản quyền đã gây thiệt hại cho những người làm phần mềm nhiều tỷ đô la mỗi năm. Ở Việt Nam, nhiều công ty đầu tư hàng trăm triệu, mất hàng năm để làm ra một phần mềm, chỉ sau khi phát hành vài ngày, sản phẩm của họ đã bị sao chép bán khắp nơi với giá từ 10- 15.000 đồng trên đĩa CD. Nếu ai mua máy tính, các cửa hàng bán máy tính sẵn sàng cài đặt miễn phí các phần mềm. Các nhà sản xuất phần mềm đã tìm các phương pháp chống sao chép nhưng “không lại” được với dân tin tặc. Cho đến nay, chưa một phần mềm nào của Việt Nam chống được nạn bẻ khoá. Tình trạng dùng không có bản quyền như vậy làm cho những người sản xuất phần mềm đóng gói không dám đầu tư.

1.6.3.2. Sở hữu trí tuệ trong tin học

Bản quyền chỉ là một trong các yếu tố về sở hữu trí tuệ và bị vi phạm nhiều hơn cả. Sở hữu trí tuệ còn các vấn đề khác nữa như kiểu dáng công nghiệp, nhãn hiệu, xuất xứ sản phẩm, giải pháp hữu ích. Một vấn đề ít được đề ý tới là tình trạng vi phạm sở hữu trí tuệ. Thông thường trong khi xây dựng các phần mềm hoặc rộng hơn là xây dựng các hệ thống thông tin có các “bí quyết công nghệ”. Thậm chí chỉ cần có một ý tưởng tốt đã là rất quan trọng. Điều này dẫn đến tình trạng nhiều người tìm cách lấy cắp bí quyết dưới nhiều hình thức. Thậm chí nhân viên các công ty có thể bỏ việc và làm rò rỉ những bí quyết công nghệ.

Luật pháp nhiều nước bảo hộ phương pháp giải quyết vấn đề, nhãn mác sản phẩm, chứ không bảo hộ ý tưởng. Để đảm bảo sở hữu trí tuệ cần đăng ký giải pháp muốn bảo hộ và có ký hợp đồng có tính pháp lý với những người tham gia. Tuy nhiên giải pháp này không giải quyết được hoàn toàn vấn đề. Chỉ có thể giải quyết được nếu nâng cao được đạo đức và ý thức pháp luật của mỗi người làm tin học.

1.6.4. Luật liên quan đến tội phạm tin học của Việt Nam

Bất cứ một nước phát triển nào cũng phải có quy định dưới dạng các văn bản pháp luật để chống lại các tội phạm tin học. Ở Việt Nam, nhận thức được tính nghiêm trọng của các tội phạm tin học, Quốc hội Cộng hoà Xã hội Chủ nghĩa Việt Nam đã ban hành một số điều luật chống tội phạm tin học trong bộ luật hình sự (13/1/2000).

Điều 224. Tội tạo ra và lan truyền, phát tán các chương trình vi - rút tin học

1. Người nào tạo ra và cố ý lan truyền, phát tán các chương trình vi-rút qua mạng máy tính hoặc bằng các phương thức khác gây rối loạn hoạt động, phong toả hoặc làm biến dạng, làm huỷ hoại các dữ liệu của máy tính hoặc đã bị xử lý kỷ luật, xử phạt hành chính về hành vi này mà còn vi phạm, thì bị phạt tiền từ năm triệu đồng đến một trăm triệu đồng hoặc phạt tù từ sáu tháng đến ba năm.

2. Phạm tội gây hậu quả rất nghiêm trọng hoặc đặc biệt nghiêm trọng, thì bị phạt tù từ hai năm đến bảy năm.

3. Người phạm tội còn có thể bị phạt tiền từ năm triệu đồng đến năm mươi triệu đồng, cấm đảm nhiệm chức vụ, cấm hành nghề hoặc làm công việc nhất định từ một năm đến năm năm.

Điều 225. Tội vi phạm các quy định về vận hành, khai thác và sử dụng mạng máy tính điện tử

1. Người nào được sử dụng mạng máy tính mà vi phạm các quy định về vận hành, khai thác và sử dụng mạng máy tính gây rối loạn hoạt động, phong toả hoặc làm biến dạng, làm huỷ hoại các dữ liệu của máy tính hoặc đã bị xử lý kỷ luật, xử phạt hành chính về hành vi này mà còn vi phạm, thì bị phạt tiền từ năm triệu đồng đến một trăm triệu đồng, cải tạo không giam giữ đến ba năm hoặc phạt tù từ một năm đến ba năm.

2. Phạm tội thuộc một trong các trường hợp sau đây, thì bị phạt tù từ hai năm đến năm năm:

a) Có tổ chức;

b) Gây hậu quả rất nghiêm trọng hoặc đặc biệt nghiêm trọng.

3. Người phạm tội còn có thể bị phạt tiền từ năm triệu đồng đến năm mươi triệu đồng, cấm đảm nhiệm chức vụ, cấm hành nghề hoặc làm công việc nhất định từ một năm đến năm năm.

Điều 226. Tội sử dụng trái phép thông tin trên mạng và trong máy tính

1. Người nào sử dụng trái phép thông tin trên mạng và trong máy tính, cũng như đưa vào mạng máy tính những thông tin trái với quy định của pháp luật gây hậu quả nghiêm trọng, đã bị xử lý kỷ luật, xử phạt hành chính mà còn vi phạm, thì bị phạt tiền từ năm triệu đồng đến năm mươi triệu đồng, cải tạo không giam giữ đến ba năm hoặc bị phạt tù từ sáu tháng đến ba năm.

2. Phạm tội thuộc một trong các trường hợp sau đây, thì bị phạt tù từ hai năm đến năm năm:

a) Có tổ chức;

b) Gây hậu quả rất nghiêm trọng hoặc đặc biệt nghiêm trọng.

3. Người phạm tội còn có thể bị phạt tiền từ ba triệu đồng đến ba mươi triệu đồng, cấm đảm nhiệm chức vụ, cấm hành nghề hoặc làm công việc nhất định từ một năm đến năm năm.

Trong tháng 7/2001 tại thành phố HCM, đã xử phạt hai trường hợp đầu tiên hai trường hợp chiếm đoạt mật khẩu, truy nhập trái phép Internet, gây thiệt hại kinh tế cho người thuê bao Internet.

Nghị định 55/2001/NĐ-CP

Ngày 23/8/2001 Chính phủ ban hành nghị định 55/2001/NĐ-CP quy định một số mức xử phạt các vi phạm khi sử dụng Internet.

Điều 41 khoản 2 quy định:

“Phạt tiền từ 200.00 đồng đến 1.000.000 đồng đối với một trong các hành vi vi phạm sau đây:

a) Sử dụng mật khẩu, khoá mật mã, thông tin riêng của người khác để truy nhập, sử dụng dịch vụ Internet trái phép

b) Sử dụng các công cụ phần mềm để truy nhập, sử dụng dịch vụ Internet trái phép.”

Điều 41 khoản 5 quy định:

“Phạt từ 10.000.000 đồng đến 20.000.000 triệu đồng đối với một trong các hành vi vi phạm sau đây:

....

g) Sử dụng Internet để nhằm mục đích đe dọa, quấy rối, xúc phạm đến danh dự, nhân phẩm của người khác mà chưa đến mức truy cứu trách nhiệm hình sự.

h) Đưa vào Internet hoặc lợi dụng Internet để truyền bá các thông tin, hình ảnh đồi trụy, hoặc những thông tin khác trái với quy định của pháp luật về nội dung thông tin trên Internet mà chưa đến mức truy cứu trách nhiệm hình sự.

i) Danh cấp mật khẩu, khoá mật mã, thông tin riêng của tổ chức, cá nhân và phổ biến cho người khác sử dụng .

k) Vi phạm các quy định về vận hành, khai thác và sử dụng máy tính gây rối loạn hoạt động, phong toả hoặc làm biến dạng, làm huỷ hoại các dữ liệu trên Internet mà chưa đến mức truy cứu trách nhiệm hình sự.”

Điều 41 khoản 6 quy định:

“Phạt tiền từ 20.000.000 đồng đến 50.000.000 đồng đối với một trong các hành vi vi phạm sau đây:

a) ...

b) Tạo ra và cố ý lan truyền, phát tán các chương trình vi rút trên Internet mà chưa đến mức truy cứu trách nhiệm hình sự.”

Chương 2

SỬ DỤNG MÁY TÍNH ^[2]

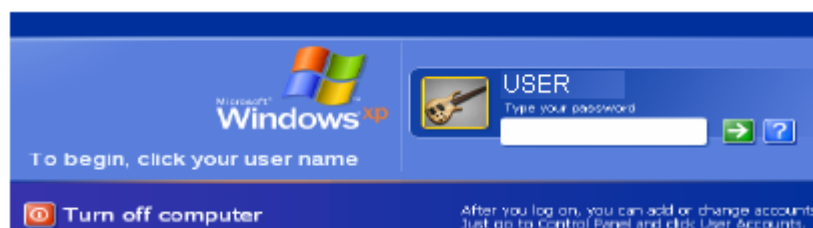
Việc sử dụng một hệ điều hành là việc rất cần thiết trước khi sử dụng bất cứ ứng dụng nào trên hệ điều hành đó. Có làm chủ được hệ điều hành thì mới làm chủ được các ứng dụng. Trong phần này, chúng tôi xin giới thiệu hai hệ điều hành khá thông dụng tại Việt Nam - Hệ điều hành Windows (tiêu biểu là hệ điều hành Windows XP) và Linux.

2.1. Hệ điều hành WINDOWS XP

2.1.1. Bắt đầu Windows XP

2.1.1.1. Đăng nhập vào Windows XP

Sau khi khởi động máy tính, và Windows XP được nạp, một màn hình đăng nhập sẽ hiện ra. Thông thường, trên một máy, có thể có một số người dùng. Trên màn hình đăng nhập, chúng ta có thể thấy danh sách những người dùng. Để có thể bắt đầu sử dụng Windows XP, chúng ta phải đăng nhập vào bằng cách chọn đúng tên trong danh sách người dùng và sau đó gõ mật khẩu (password) vào.

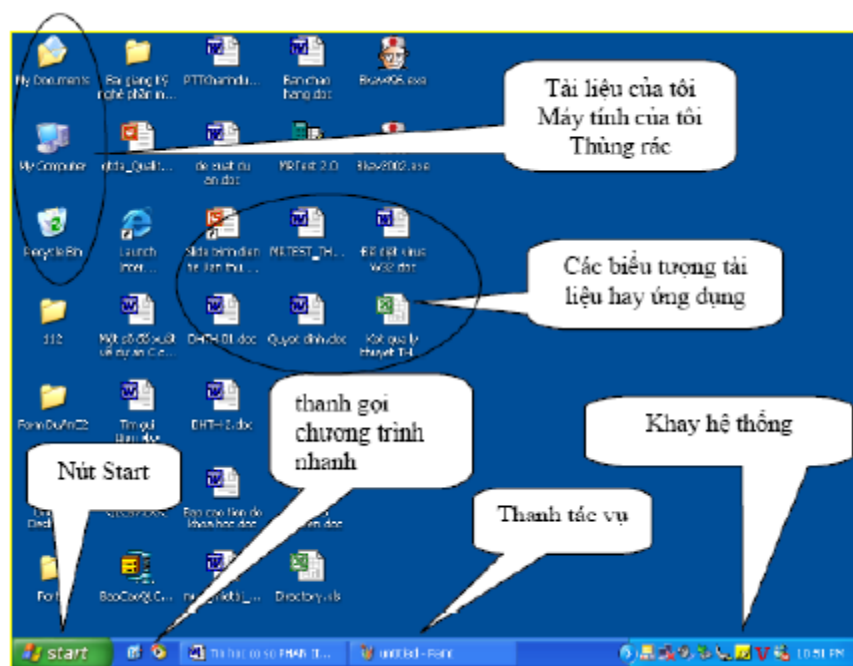


Hình 2.1. Màn hình đăng nhập vào Windows XP

2.1.1.2. Màn hình nền (desktop) Windows XP

Trong Hình 2.2, chúng ta có thể thấy một ví dụ về màn hình nền (desktop) của Windows XP. Màn hình desktop nền này xuất hiện sau khi chúng ta đã đăng nhập vào Windows XP.

Trên màn hình desktop, chúng ta có thể thấy các biểu tượng, nút Start, thanh tác vụ (taskbar), thanh gọi chương trình nhanh (quick launch bar) và khay hệ thống (system tray). Chúng ta có thể tùy chỉnh màn hình desktop nền của chúng ta bằng cách thêm các ảnh nền, thay màu nền, thêm các biểu tượng, ...



Hình 2.2: Desktop với các biểu tượng, thanh tác vụ, thanh gọi chương trình nhanh và khay hệ thống

2.1.2. Một số khái niệm cơ bản trong Windows XP

2.1.2.1. Tổ chức thông tin trên đĩa từ

Mọi thông tin trên máy tính đều chỉ là tập hợp các con số 0 hoặc 1. Các số này được lưu trữ dưới các trạng thái khác nhau trong máy tính: Tín hiệu từ, tín hiệu điện, tín hiệu quang. . . để khi cần thiết, máy tính có thể đọc lại chúng và chuyển thành các con số 0, 1 để xử lý. Các thiết bị trong máy tính có khả năng lưu trữ những thông tin này được gọi là các thiết bị nhớ. Windows lưu trữ và tổ chức thông tin theo cấu trúc cây. Có bốn khái niệm chính được Windows sử dụng trong cấu trúc này:

- Ổ đĩa (driver)
- Thư mục (directory)
- Tập (file)
- Đường dẫn (path)

Thông tin được lưu trữ trong các tệp khác nhau. Những tệp này được đặt trong các thư mục, các thư mục có thể được đặt trong các thư mục khác, cuối cùng là thư mục được đặt trong ổ đĩa.

2.1.2.2. Ổ đĩa

Ổ đĩa là đơn vị logic cấp cao nhất mà Windows sử dụng trong cấu trúc cây để lưu trữ và tổ chức thông tin. Mỗi ổ đĩa tương ứng với một gốc cây, do đó không có khả năng ổ đĩa chứa ổ đĩa. Với Windows, mỗi máy tính có thể có một hay nhiều ổ đĩa. Mỗi ổ đĩa được đại diện bởi một ký tự trong bảng chữ cái A – Z (không phân biệt hoa - thường) và ký tự hai chấm “.”. Do đó, mỗi máy tính chỉ có thể có tối đa là 26 ổ đĩa.

Bên trong mỗi ổ đĩa là các tệp và các thư mục. Các tệp và thư mục chứa trong ổ đĩa này có thể trùng tên với các tệp và thư mục trong ổ đĩa khác. Nhưng không bao giờ có hai ổ đĩa cùng

tên. Thông thường các ổ đĩa A, B là các ổ đĩa mềm, các ổ đĩa C, D, . . . Z là các ổ đĩa cứng, đĩa mạng, đĩa CD(DVD) hay đĩa ảo, đĩa RAM.

2.1.2.3. Thư mục

Thư mục là khái niệm logic của Windows dùng để quản lý thông tin trên đĩa. Mỗi thư mục tương ứng với một cành trong cấu trúc cây của Windows. Do đó, mỗi thư mục có thể chứa hoặc không chứa một hay nhiều thư mục con hay các tệp (khi đó nó được gọi là thư mục cha). Các thư mục con của thư mục này lại có thể chứa các thư mục con hoặc các tệp khác. Mỗi thư mục đều được đặt tên, quy tắc đặt tên thư mục cũng giống như quy tắc đặt tên tệp sẽ được nêu ở mục sau). Các thư mục con có chung một thư mục cha thì không được phép trùng tên. Ngoài ra, Windows còn có ba thư mục đặc biệt được biểu diễn riêng bởi các ký hiệu trong câu lệnh:

- Thư mục gốc (root directory): Ký hiệu “\”. Đây là thư mục duy nhất không có thư mục cha. Thư mục này phụ thuộc trực tiếp vào ổ đĩa. Mỗi ổ đĩa chỉ có một thư mục gốc duy nhất. Mọi thư mục khác trong ổ đĩa đều là thư mục con của thư mục gốc hoặc là con của các thư mục khác.
- Thư mục hiện hành (current directory): Ký hiệu “.”. Giống như khái niệm ổ đĩa, tại mỗi thời điểm, Windows chỉ làm việc với một thư mục duy nhất trên các ổ đĩa (còn gọi là đang mở thư mục nào). Thư mục mà Windows đang làm việc gọi là thư mục hiện hành. Nếu máy tính có nhiều ổ đĩa thì mỗi ổ đĩa có một thư mục hiện hành riêng (Mặc định khi mới khởi động là thư mục gốc của mỗi ổ đĩa). Tuy nhiên khi chỉ nói thư mục hiện hành tức là nói đến thư mục đang làm việc của ổ đĩa hiện hành.

Ví dụ: Giả sử máy tính của bạn có hai ổ đĩa A và C. Tại ổ A, thư mục đang mở là thư mục gốc “A:\” còn tại ổ C thư mục đang mở là thư mục WINDOWS nằm trong thư mục gốc “C:\WINDOWS”. Ổ đĩa hiện hành là ổ đĩa C. Khi đó ta có các phát biểu sau:

- Thư mục hiện hành của ổ đĩa C là: C:\WINDOWS
- Thư mục hiện hành của ổ đĩa A là: A:\
- Thư mục hiện hành là: C:\WINDOWS

- Thư mục cha (parent directory): Ký hiệu “..”. Ngoại trừ các thư mục gốc “\”, mọi thư mục khác trên hệ thống đều nằm trong một thư mục cha của nó. Có thể dễ dàng xác định được thư mục cha của những thư mục này thông qua tên của chúng.

Ví dụ:

- Thư mục cha của thư mục C:\WINDOWS\TEMP là thư mục C:\WINDOWS
- Thư mục cha của thư mục C:\WINDOWS là thư mục C:\
- Thư mục C:\ không có thư mục cha

Chính bởi khả năng chứa lẫn nhau của các thư mục nên cấu trúc lưu trữ thông tin của Windows có dạng cây (còn gọi là cây thư mục). Mỗi một máy tính (sử dụng Windows) có thể có một hay nhiều cây phụ thuộc vào số lượng ổ đĩa có trong máy.

2.1.2.4. Tệp (File)

Tệp là khái niệm logic cơ sở của Windows trong việc lưu trữ thông tin. Mỗi tệp là một tập hợp các thông tin có quan hệ với nhau. Trong hệ thống cây thư mục, các tệp tương ứng với các lá cây. Không có khái niệm tệp này chứa tệp kia. Mỗi tệp đều phải được chứa trong một và chỉ một thư mục duy nhất. Cũng giống như thư mục: Các tệp và cả các thư mục có cùng một thư mục cha

thì đều không được phép trùng tên (Nếu điều này xảy ra tại máy tính nào có nghĩa là hệ thống lưu trữ thông tin của máy tính đó đã có vấn đề).

2.1.2.5. Đường dẫn (Path)

Đường dẫn là khái niệm được Windows sử dụng để định vị các tệp, thư mục trong hệ thống cây thư mục của Windows. Mỗi tệp hoặc thư mục (trừ thư mục gốc) đều được đặt trong một thư mục cha xác định, các thư mục cha đó lại được đặt trong các thư mục cha khác. Quá trình trên cứ tiếp diễn như vậy cho đến thư mục gốc. Mỗi thư mục gốc lại được đặt trong một ổ đĩa xác định. Mỗi quá trình đi từ một tệp hoặc thư mục xác định đến ổ đĩa chứa chúng (Xét cả hai chiều) tạo ra một con đường xác định ta gọi là đường dẫn.

Trong một máy tính, không có hai tệp hoặc thư mục nào có chung một con đường như vậy. Một đường dẫn đầy đủ bao gồm tên ổ đĩa (có dấu hai chấm), thư mục gốc, các thư mục cha đến một tệp hoặc một thư mục con được gọi là đường dẫn tuyệt đối. Nếu thiếu bất kỳ một thành phần nào trong các thành phần nêu trên thì đường dẫn được gọi là đường dẫn tương đối. Các thành phần trên được nối với nhau bởi các dấu “\” (Ngoại trừ các phần nối tới thư mục gốc).

Ví dụ: - Đường dẫn tuyệt đối

C:\WINDOWS\COMMAND\ANSI.SYS

- Đường dẫn tương đối

ANSI.SYS

..\COMMAND\ANSI.SYS




C:ANSI.SYS

\WINDOWS\COMMAND\ANSI.SYS

2.1.3 Một số khái niệm cơ bản trên màn hình Windows XP

2.1.3.1. Biểu tượng (Icon)

Biểu tượng là các hình ảnh nhỏ thể hiện các chương trình máy tính, các tệp, thư mục, máy in, . . . Để kích hoạt các chương trình/tệp/thư mục/ . . . mà các biểu tượng thể hiện, chúng ta chỉ cần nhấp chuột trái kép vào biểu tượng.

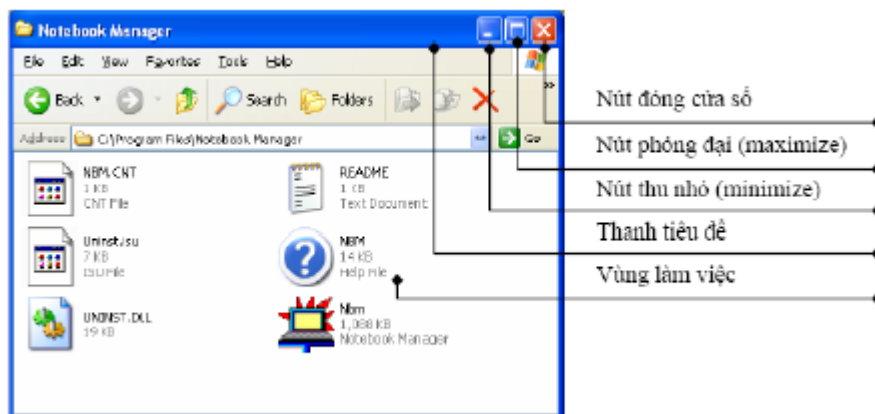
	My computer, từ đây ta thực hiện truy cập tới mọi tài nguyên trên máy tính.
	My document, là nơi mặc định chứa tài liệu như văn bản, bảng tính, hình ảnh, âm thanh, . . .
	Recycle Bin (thùng rác), là nơi chứa tài liệu khi ta xóa nó. Điều này cho phép ta có thể thực hiện khôi phục lại các tài liệu này nếu chúng ta lỡ xóa nhầm. Các tài liệu này sẽ thực sự bị xóa khi ta xóa nó trong thùng rác.

Bảng 2.1: Một số biểu tượng đáng chú ý trên desktop

Chúng ta có thể đổi tên của các biểu tượng bằng cách nhấp chuột phải vào chúng và chọn “rename”, hoặc tương tự nhấp chuột phải và chọn “delete” để xóa chúng. Trên màn hình desktop có 3 biểu tượng rất đáng chú ý, được chỉ ra trong Bảng 2.1.

2.1.3.2. Cửa sổ (Window)

Khi một chương trình trong Windows XP được kích hoạt, giao diện làm việc của nó được thể hiện là một vùng hình chữ nhật trên màn hình, được gọi là cửa sổ (window).



Hình 2.3: Cửa sổ

Hình 2.3 là một ví dụ của cửa sổ. Trên mỗi cửa sổ là một thanh tiêu đề chứa tiêu đề của cửa sổ, thường là tên thư mục hoặc tên chương trình. Phía trên bên trái của cửa sổ là bảng chọn hệ thống của cửa sổ bao gồm các chức năng:

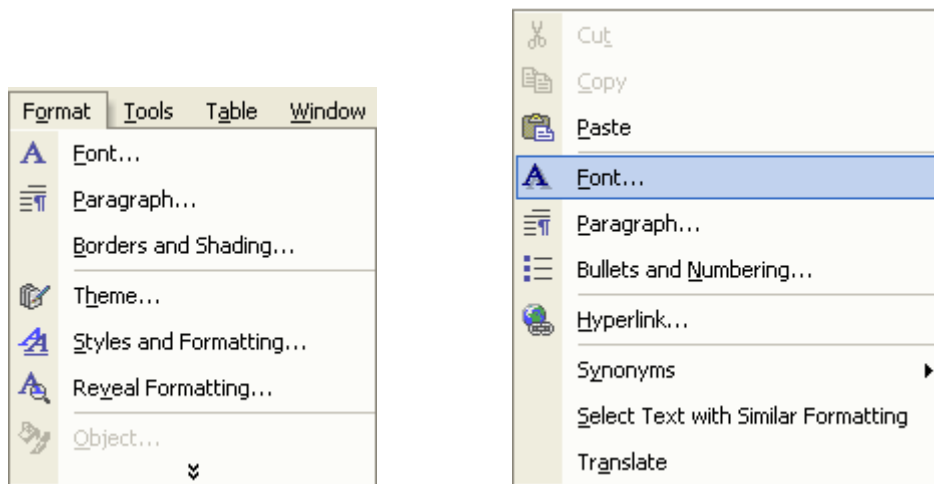
- Khôi phục (Restore): Đưa về chế độ mặc định của cửa sổ.
- Di chuyển vị trí cửa sổ (Move): (Chỉ thực hiện được khi ở chế độ mặc định, chứ không phải chế độ Maximize và Minimize) Di chuyển cửa sổ. Chúng ta cũng có thể di chuyển cửa sổ bằng cách nhấp chuột và giữ vào thanh tiêu đề của cửa sổ, di chuyển chuột để kéo cửa sổ đến chỗ cần đặt và thả chuột.
- Thay đổi kích thước (Size): (Chỉ thực hiện được khi ở chế độ mặc định, chứ không phải chế độ Maximize và Minimize): Thay đổi kích thước cửa sổ. Chúng ta cũng có thể thay đổi kích thước của cửa sổ bằng cách nhấp chuột và giữ vào các đường biên của cửa sổ, di chuyển chuột để thay đổi kích thước.
- Thu nhỏ (Minimize): Thu nhỏ về chế độ nhỏ nhất (không nhìn thấy cửa sổ, chỉ nhìn thấy trên thanh tác vụ).
- Phóng đại (Maximize): Phóng to cực đại (chiếm toàn bộ màn hình).
- Đóng (Close): Đóng cửa sổ, đồng thời kết thúc chương trình. Ngoài ra, phía trên bên phải cửa sổ có ba nút, lần lượt từ trái sang phải: thu nhỏ cửa sổ; phóng đại/khôi phục về chế độ mặc định hoặc chuyển đổi giữa hai chế độ này; và đóng cửa sổ.

2.1.3.3. Bảng chọn (Menu) (hay còn gọi là thực đơn)

Bảng chọn là một danh sách các lệnh giúp cho người dùng không phải nhớ mà chỉ cần lựa chọn. Bảng chọn xuất phát từ kết quả nghiên cứu về khả năng xử lý thông tin của con người – con người nhận dạng tốt hơn nhớ lại. Một mục trong bảng chọn được chọn khi chúng ta nhấp

chuột trái vào mục đó (có trường hợp một mục trong bảng chọn có thể được chọn nhanh bằng cách nhấn một tổ hợp phím tắt, ví dụ Ctrl+O để mở một tệp). Có hai loại bảng chọn chính:

- *Bảng chọn kiểu thả xuống (drop down menu)*: Loại bảng chọn này thường nằm phía trên của cửa sổ. Một mục của bảng chọn sẽ hiện ra khi ta nhấp chuột trái vào mục đó. Bảng chọn loại này có thể có nhiều tầng, giúp cho việc tổ chức bảng chọn một cách dễ dàng và cho phép người dùng tìm kiếm một mục dễ hơn.







Hình 2.4: Bảng chọn kiểu thả xuống (trái) và kiểu bật lên (phải)






- *Bảng chọn kiểu bật lên (pop-up menu)*: Loại bảng chọn này được kích hoạt khi chúng ta nhấp chuột phải vào hầu hết các vị trí trên màn hình. Tùy theo vị trí của chuột, loại bảng chọn tương ứng sẽ hiện lên. Ví dụ, khi chúng ta đang soạn thảo văn bản, nhấp chuột phải trong khu vực soạn thảo sẽ hiện ra bảng chọn chứa một số chức năng soạn thảo chính như chép (copy), cắt (cut), dán (paste). Loại bảng chọn này được thiết kế nhằm giảm bớt số thao tác của người dùng cho một số chức năng hay dùng. Người dùng không phải di chuột và đưa mắt nhìn lên phía trên cửa sổ để dùng bảng chọn kiểu thả xuống. Kiểu bảng chọn này cũng một phần giúp giảm bớt các tác hại nghề nghiệp khi sử dụng máy tính.

2.1.3.4. Con trỏ chuột

Có thể dễ dàng nhận thấy được hình dạng của con trỏ chuột thay đổi khi chúng ta di chuyển con chuột đến các phần khác nhau của màn hình hoặc khi máy tính đang xử lý. Mỗi hình dạng của con trỏ chuột có mục đích riêng của nó. Chúng có thể cung cấp chúng ta một số thông tin quan trọng. Có một số hình dạng con trỏ chuột chính chúng ta phải lưu ý, được chỉ ra trong Bảng 2.2 và Bảng 2.3.

	Con trỏ bình thường: Con trỏ bình thường được dùng để làm các động tác chọn lựa.
	Tại góc của một cửa sổ: Con trỏ kích thước này dùng để thay đổi kích thước của cửa sổ bằng cách nhấp chuột và kéo. Con trỏ này làm việc trên cả hai cạnh của cửa sổ kể với góc của cửa sổ nó đang trỏ đến.
	Tại cạnh trái hay phải của cửa sổ: Con trỏ kích thước này dùng để thay đổi cạnh trái hay phải của cửa sổ bằng cách nhấp chuột và kéo.
	Tại cạnh dưới hay trên của cửa sổ: Con trỏ kích thước này dùng để thay đổi cạnh dưới hay trên của cửa sổ bằng cách nhấp chuột và kéo.

Bảng 2.2: Một số con trỏ chuột đáng chú ý (phần 1).

	Tại bất cứ đâu: Khi Windows đang bận và không nhận dữ liệu vào.
	Tại bất cứ đâu: Khi Windows đang bận nhưng vẫn có thể nhận dữ liệu vào.
	Tại vị trí nhập dữ liệu văn bản (text): con trỏ này được gọi là I-beam và xuất hiện tại vị trí nhập dữ liệu văn bản, thường là trong một hộp nhập liệu.
	Trên một từ kèm theo một ý nghĩa ẩn: Con trỏ này xuất hiện khi con chuột nằm trên một từ có kèm theo một ý nghĩa ẩn. Ví dụ trong Internet Explorer, bàn tay xuất hiện có nghĩa là có một liên kết đến từ đang trỏ đến.
	Khi không thể thao tác được: con trỏ này có nghĩa là tại vị trí hiện tại, không thể thao tác chuột được.

Bảng 2.3: Một số con trỏ chuột đáng chú ý (phần 2).

2.1.3.5. Sử dụng chuột (Mouse) trên nền Windows

Chuột dùng điều khiển con trỏ chuột tương tác với những đối tượng trên màn hình.

Các phím chuột, Một con chuột có một phím chính và một phím phụ. Phím chính dùng để chọn và nhấp vào các đối tượng, đặt vị trí của con trỏ trong một văn bản, và để kéo các đối tượng. Phím chính thường được đặt là phím trái. Phím phụ thường dùng để hiện ra một bảng chọn liên quan đến đối tượng đang trỏ đến. Bảng chọn này giúp hoàn thành một số công việc liên quan đến đối tượng đang trỏ đến nhanh hơn. Nhấp phím phụ được gọi là nhấp chuột phải (right-clicking) do phím phụ thường được đặt là phím phải. Chúng ta có thể thay đổi chức năng của hai phím này (ví dụ cho những người thuận tay trái). Chuột thường có 2 nút:

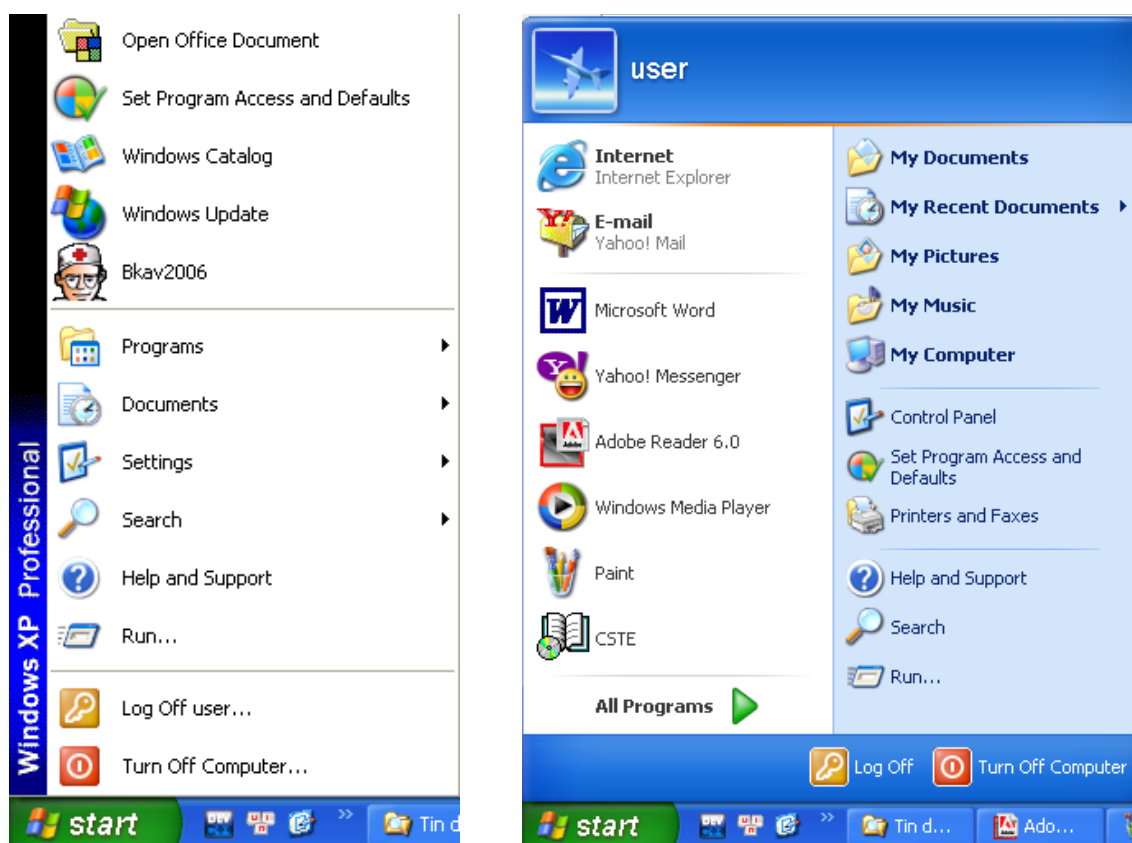
- Nút trái thường dùng để chọn đối tượng; rê đối tượng...
- Nút phải thường dùng để hiển thị một menu công việc. Nội dung Menu công việc thay đổi tùy thuộc con trỏ chuột đang nằm trên đối tượng nào.

Các hành động mà chuột thực hiện

Trỏ đối tượng	Rà chuột trên mặt phẳng bàn để di chuyển con trỏ chuột trên màn hình trỏ đến đối tượng cần xử lý.
Click trái	Thường dùng để chọn một đối tượng, bằng cách trỏ đến đối tượng, nhấn nhanh và thả mắt trái chuột.
Rê/Kéo (Drag)	Dùng di chuyển đối tượng hoặc quét chọn nhiều đối tượng ... bằng cách trỏ đến đối tượng, nhấn và giữ mắt trái chuột, di chuyển chuột để dời con trỏ chuột đến vị trí khác, sau đó thả mắt trái chuột.
Click phải	Thường dùng hiển thị một menu công việc liên quan đến mục được chọn, bằng cách trỏ đến đối tượng, nhấn nhanh và thả mắt phải chuột.
Bấm đúp (Double click)	Thường dùng để kích hoạt chương trình được hiển thị dưới dạng một biểu tượng trên màn hình, bằng cách trỏ đến đối tượng, nhấn nhanh và thả mắt trái chuột 2 lần.

2.1.3.6. Nút “Start” của Windows XP

Nút “Start” (bắt đầu) là một phần rất quan trọng của Windows XP. Khi chúng ta nhấp chuột vào nút này, menu chính sẽ hiện ra cho phép chúng ta truy cập vào các chương trình (xem Hình 2.5), cài đặt, máy in, . . .



Hình 2.5: Bảng chọn Start menu (bên phải) và Classical Start menu (bên trái)

Sau đây là các chức năng trên bảng chọn khi chúng ta nhấp chuột vào nút Start:

Log off - cho phép người dùng hiện thời rời hệ thống.

Turn Off Computer - tắt máy, khởi động lại hoặc chuyển sang chế độ nghỉ ngơi. **All Programs** - Truy cập vào các chương trình đã được cài.

Run - Dùng để bắt đầu các tệp thực hiện (các chương trình) một cách thủ công.

Search - Tìm trên máy tính tệp và thư mục.

Help and Support - Mở phần trợ giúp sẵn có của Windows, trong đó có các chủ đề trợ giúp khác nhau, cập nhật Windows và cho phép yêu cầu trợ giúp từ một người bạn hay đồng nghiệp ở xa thông qua Internet hoặc mạng nội bộ.

Printers and Faxes - Truy cập vào thư mục máy in và máy fax, cho phép chúng ta thêm/xóa và cấu hình các máy in và máy fax.

Control Panel - Cho phép chúng ta cấu hình các thiết lập khác nhau của Windows XP.

My Computer - Cho phép chúng ta truy cập vào các ổ đĩa của máy tính và các tệp.

My Music - Liên kết đến một thư mục tạo bởi Windows XP chuyển để lưu trữ các tệp âm nhạc trên ổ cứng.

My Pictures - Liên kết đến một thư mục tạo bởi Windows XP chuyển để lưu trữ các tệp ảnh trên ổ cứng.

My Recent Documents - Thư mục này chứa các tài liệu được mở gần đây.

My Documents - Cho phép truy cập đến một thư mục tạo sẵn bởi Windows XP cho phép chúng ta lưu trữ tài liệu.

Tour Windows XP - Bắt đầu một cuộc dạo qua các tính năng của Windows XP.

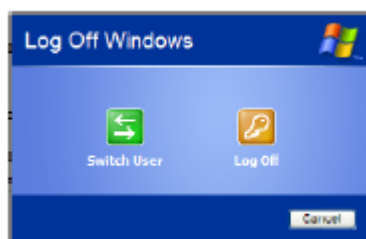
Windows Movie Maker - Mở chương trình sửa phim của Windows XP.

Outlook Express - Mở chương trình đọc thư Outlook Express.

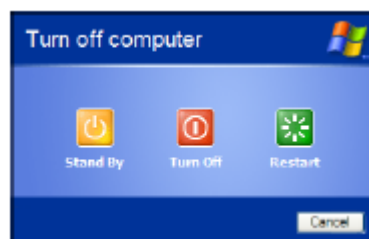
Files and Settings Transfer Wizard - Cho phép chúng ta nhập hoặc xuất các tệp từ hoặc đến các máy khác có cài Windows XP.

2.1.3.7. Bảng chọn đăng thoát (Log off)

Khi bạn nhấp chuột vào Log off trong menu Start, bạn sẽ thấy một hộp hội thoại xuất hiện như trong Hình 2.6a. Trong hộp hội thoại đó có hai sự lựa chọn:



Hình 2.6a: Bảng chọn “Log off”



Hình 2.6b: Bảng chọn “Turn off computer”

Log off: Đăng thoát khỏi người dùng hiện thời, quay trở về màn hình đăng nhập

Switch user: Không đăng thoát khỏi người dùng hiện thời, tuy nhiên quay trở về màn hình đăng nhập để cho người khác có thể sử dụng. Các ứng dụng của người dùng hiện thời vẫn được chạy.

2.1.3.8. Bảng chọn Tắt máy (Turn off computer)

Khi bạn nhấp chuột vào Turn Off Computer trong menu bắt đầu, bạn sẽ thấy một hộp hội thoại xuất hiện như trong Hình 2.6b. Trong hộp hội thoại đó có ba sự lựa chọn:

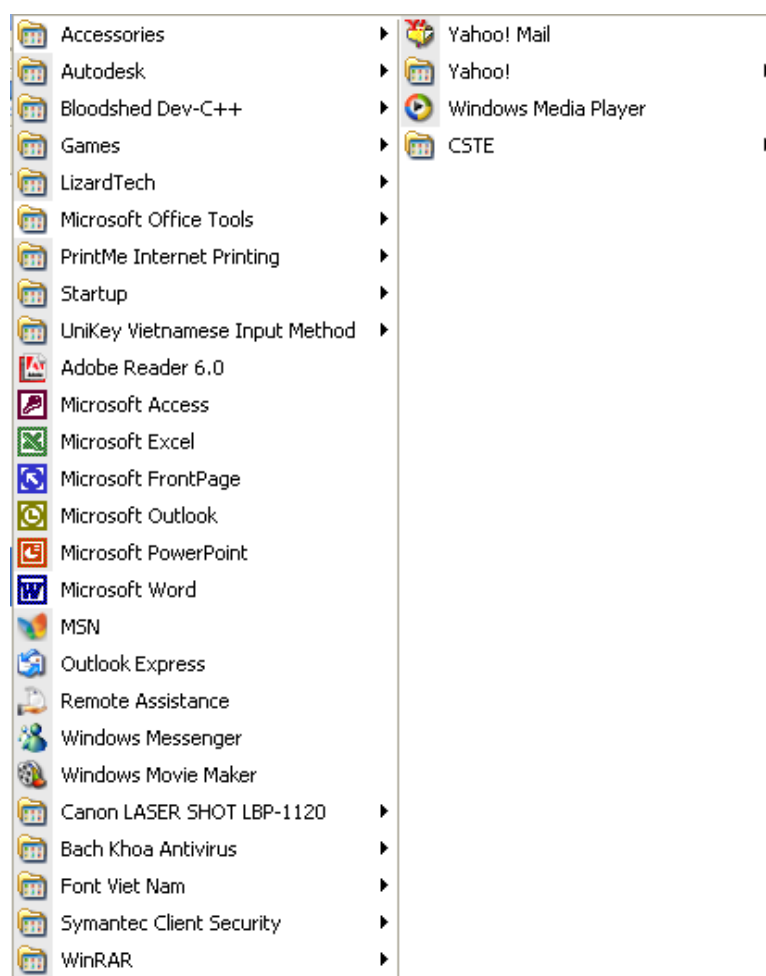
Standby - đưa máy tính vào chế độ nghỉ ngơi (standby). Về cơ bản, chế độ này sẽ tắt một số thành phần của máy tính như màn hình và đĩa cứng để tiết kiệm năng lượng.

Turn Off - Tắt máy, chúng ta phải luôn luôn sử dụng lựa chọn này để tắt máy.

Restart - khởi động lại máy.

2.1.3.9. Bảng chọn “All Programs”

Khi chúng ta di chuột vào “All Programs”, một menu sẽ hiện ra như trong Hình 2.7. Từ đây chúng ta có thể truy cập vào những chương trình đã được cài trên máy. Menu này ở trên các máy khác nhau có thể khác nhau do cài đặt những chương trình khác nhau.



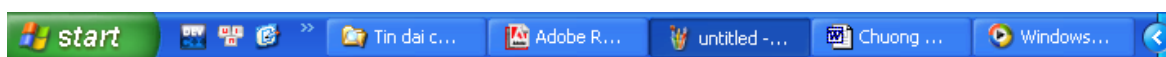
Hình 2.7: Bảng chọn “All Programs”.

Để mở một chương trình chỉ cần nhấp chuột vào chương trình đó. Một số mục của menu có một mũi tên nhỏ màu đen chỉ sang phải - điều này nói lên rằng nó chứa một menu con khác.

Để mở menu con đó, chúng ta phải di chuột vào mục chứa menu con đó. Chúng ta có thể thay đổi tên của bất cứ chương trình nào trên menu bằng cách nhấp chuột phải vào và chọn rename. Để xóa một chương trình, chúng ta có thể nhấp chuột phải vào và chọn delete. Tuy nhiên cách này chỉ xóa liên kết từ menu đến chương trình chứ không thực sự xóa chương trình. Để thực sự xóa chương trình, chúng ta nên dùng chức năng uninstall.

2.1.4 Thanh tác vụ của Windows XP

Thanh tác vụ là một phần quan trọng khác của hệ điều hành Windows XP. Một trong những chức năng chính của nó là để chuyển đổi giữa các chương trình.



Hình 2.8: Thanh tác vụ.

Trong hình 2.8 là một thanh tác vụ tiêu biểu của Windows XP. Về thanh tác vụ này, có hai điểm đáng lưu ý:

- *Window Tabs* - Windows tabs được dùng để chuyển giữa bất cứ cửa sổ nào đang được mở (chương trình, thư mục, v.v.). Chúng ta có thể chuyển giữa các cửa sổ bằng cách nhấp chuột vào tab cửa sổ tương ứng. Nhấp chuột vào một tab cửa sổ đang được chọn sẽ thu nhỏ (minimize)/ hoặc phóng to cửa sổ đó.
- *Grouping Window Tabs* - Một chức năng rất hay của Windows XP (không giống như các phiên bản trước) là khả năng ghép các tệp/cửa sổ của cùng một chương trình vào một tab khi phần Window Tabs đã đầy. Ví dụ, khi chúng ta có 2 văn bản Word được mở ở 2 cửa sổ khác nhau với 2 tab, 2 tab đó sẽ được ghép thành một khi Windows Tab đầy. Chúng ta có thể truy cập một trong hai cửa sổ/văn bản này bằng cách nhấp chuột trái vào tab chung, một menu sẽ hiện ra cho phép chúng ta chọn tệp/cửa sổ cần chọn vào. Chúng ta có thể dễ dàng tắt chức năng này bằng cách nhấp chuột phải vào thanh tác vụ, chọn Properties và xóa bỏ đánh dấu ở phần “Group similar taskbar buttons”.

2.1.5 Thanh gọi chương trình nhanh (Quick Launch Bar)

Windows XP có một thanh gọi chương trình nhanh ngầm định và ẩn. Khi muốn thanh gọi chương trình nhanh này hiện, chúng ta chỉ cần nhấp chuột phải vào thanh tác vụ, đưa chuột đến “toolbars”, và đánh dấu vào ô “Quick Launch”. Thanh gọi chương trình nhanh cho phép chúng ta truy cập trực tiếp vào các chương trình, tệp hoặc thư mục chỉ bằng một lần nhấp chuột.



Hình 2.9: Thanh gọi chương trình nhanh.

Khi chúng ta mới kích hoạt thanh gọi chương trình nhanh, thường có sẵn một số biểu tượng trên thanh đó:

- *Biểu tượng “Show Desktop”* (là biểu tượng bên phải nhất trong hình 2.9): khi nhấp chuột vào, mọi cửa sổ đang mở sẽ được thu nhỏ để màn hình nền của Windows XP hiện ra, nhấp chuột vào đó một lần nữa sẽ khôi phục lại tất cả các cửa sổ về trạng thái cũ.

- Biểu tượng “Internet Explorer” (biểu tượng bên trái trong hình 2.9): dùng để mở trình duyệt Internet Explorer của Microsoft.

Để thêm biểu tượng vào thanh gọi chương trình nhanh, chỉ cần kéo (nhấp chuột vào biểu tượng, giữ và di chuột) các biểu tượng của các chương trình/tệp/thư mục vào thanh này và thả ra (thả nút chuột). Cũng rất dễ để xóa một biểu tượng từ thanh gọi chương trình nhanh. Để làm được việc đó, chỉ cần nhấp chuột phải vào biểu tượng cần xóa và chọn “delete”. Lưu ý rằng xóa một biểu tượng không xóa chương trình khỏi máy tính.

2.1.6 Khay hệ thống (System Tray)

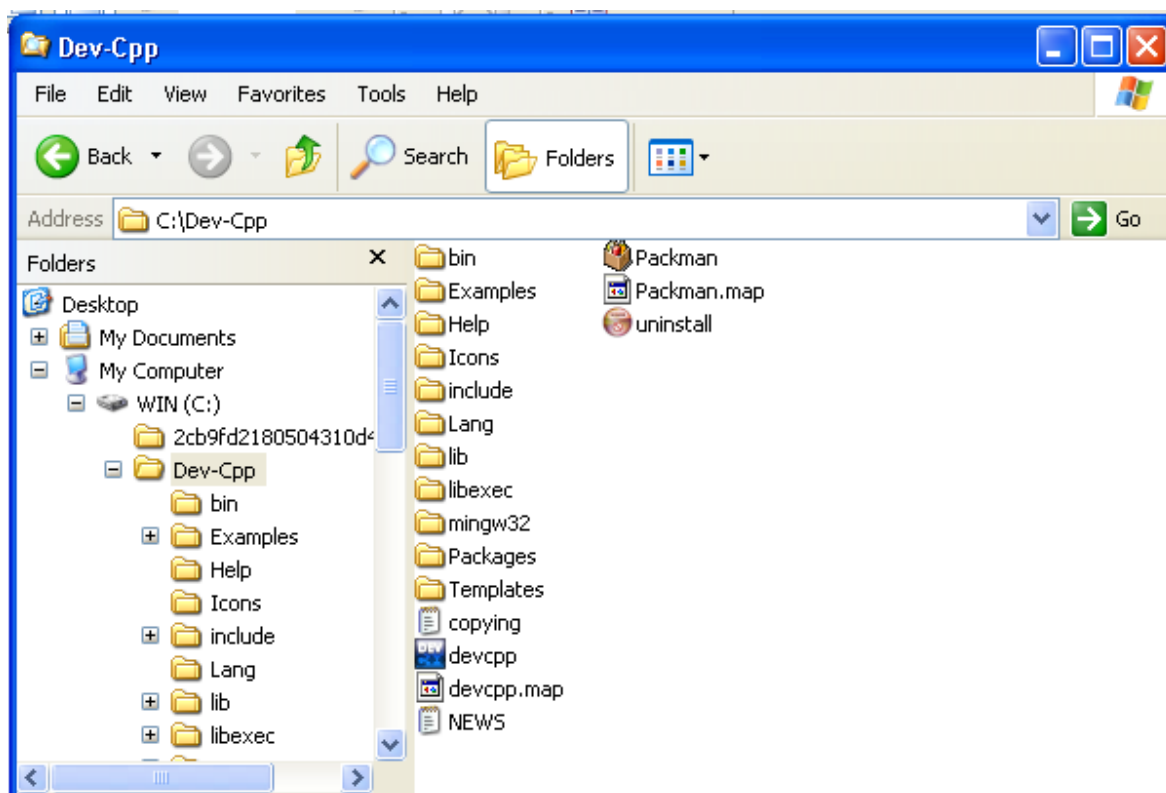
Khay hệ thống hiển thị những biểu tượng (icon) của một số chương trình đã được nạp vào bộ nhớ. Đây thường là những chương trình chạy thường trú trên nền của Windows. Ngoài ra trên khay hệ thống còn hiển thị giờ, và khi chúng ta di chuyển chuột lên trên chỗ hiện giờ, chúng ta sẽ biết được ngày tháng. Để thay đổi thời gian (ngày tháng và giờ) chỉ cần nhấp chuột vào chỗ hiện giờ.



Hình 2.10: Khay hệ thống.

2.1.7 Sử dụng “Windows Explorer”

Một phần quan trọng của khi xử dụng máy tính chính là việc quản lý tệp và thư mục, ví dụ như tạo hoặc xóa tệp/thư mục. “Windows Explorer” (có từ Windows 95) đã là một công cụ rất hữu ích để quản lý tệp và thư mục.



Hình 2.11: Chương trình Windows Explorer.

Để mở Windows Explorer, có bốn cách:

Cách 1: Nhấp chuột vào nút Start, di chuột vào All Programs, rồi Accessories, và nhấp chuột vào Windows Explorer.

Cách 2: Giữ nút Start và bấm phím E.

Cách 3: Nhấp chuột vào nút Start, chọn run, rồi gõ explorer vào nhấn Enter.

Cách 4: Mở mục My Computer và nhấp vào nút Folders ở phía trên. Sau khi kích hoạt Windows Explorer, một cửa sổ tương tự như Hình 2.11 sẽ hiện ra. Chúng ta có thể nhìn thấy cửa sổ Windows Explorer chia làm hai phần: phần bên trái được gọi là danh sách thư mục, phần bên phải chứa các tệp và thư mục trong thư mục hiện thời. Trong danh sách thư mục, một số thư mục có dấu cộng (+) bên cạnh, nói lên rằng trong thư mục đó có chứa một số thư mục khác.

2.1.8 Sử dụng các dòng lệnh trong Windows (giống như DOS)

Không phải lúc nào việc sử dụng cửa sổ, con chuột và biểu tượng cũng thuận tiện hơn chế độ dòng lệnh giống như DOS. Trong một số trường hợp, ví dụ như xóa tất cả các tệp trong thư mục hiện thời bắt đầu bằng abc, việc sử dụng dòng lệnh tiện lợi hơn rất nhiều. Để có thể sử dụng các dòng lệnh trong Windows, chọn Start, rồi All Programs, rồi Accessories, rồi Command Prompt (hoặc Start rồi Run, rồi gõ cmd). Một cửa sổ hiện ra (giống màn hình của DOS) cho phép chúng ta thực hiện các dòng lệnh.

2.1.8.1. Lệnh chuyển ổ đĩa hiện hành

Cú pháp:

<Tên ổ đĩa>:

Ví dụ:

a:→Chuyển ổ đĩa hiện hành sang ổ A.

c:→Chuyển ổ đĩa hiện hành sang ổ C.

2.1.8.2. Lệnh liệt kê nội dung thư mục

Cú pháp:

DIR [Tên thư mục]

Mô tả:

Lệnh DIR với tham số là tên thư mục sẽ hiển thị nội dung của thư mục được chỉ định trong phần tham số (Nếu không chỉ định thì Windows sẽ tự hiểu là làm việc với thư mục hiện hành). Kết quả trả về được hiển thị trên màn hình bao gồm các thông tin: Tên ổ đĩa, tên thư mục liệt kê, các thư mục con và các tệp tin bên trong thư mục được chỉ định, . . . (Thư mục con được Windows đánh dấu bằng <DIR>).

Ví dụ: Giả sử thư mục hiện hành là C:\WINDOWS. Để liệt kê nội dung thư mục hiện hành ta có thể sử dụng các lệnh sau:

DIR DIR .

DIR C:\WINDOWS

Để liệt kê nội dung thư mục C:\ ta có thể sử dụng các lệnh sau:

DIR C:\ DIR \ DIR ..

2.1.8.3. Lệnh chuyển thư mục hiện hành

Cú pháp:

CD <Tên thư mục cần chuyển>

hoặc CHDIR <Tên thư mục cần chuyển>

Mô tả:

Lệnh CD (viết tắt của CHDIR) được sử dụng khi người dùng muốn thay đổi thư mục hiện hành. Với mục đích này, tên thư mục cần chuyển là tham số bắt buộc phải có. Nếu thư mục cần chuyển có thật và nằm trong ổ đĩa hiện hành, thư mục hiện hành sẽ được thay đổi ngay đến thư mục này, dấu nhắc lệnh cũng được thay đổi theo. Nếu thư mục cần chuyển có thật nhưng không nằm trong ổ đĩa hiện hành thì thư mục hiện hành của ổ đĩa đó cũng được thay đổi ngay nhưng dấu nhắc lệnh vẫn không đổi (Nghĩa là ổ đĩa hiện hành và thư mục hiện hành cũng không đổi). Muốn thấy hiệu lực của sự thay đổi này, ta cần thay đổi ổ đĩa hiện hành (tức là thư mục hiện hành cũng thay đổi theo) bằng lệnh chuyển ổ đĩa.

Ví dụ:

Giả sử máy tính có hai ổ đĩa là A và C. Thư mục hiện hành trên ổ đĩa A là A:\NC và thư mục hiện hành trên ổ đĩa C là C:\WINDOWS\COMMAND.

Khi đó lệnh: CD.. sẽ thay đổi dấu nhắc hệ thống từ C:\WINDOWS\COMMAND thành C:\WINDOWS.

Lệnh A: sẽ thay đổi dấu nhắc hệ thống từ C:\WINDOWS thành A:\NC Lệnh CD C:\ sẽ thay đổi thư mục hiện hành của ổ đĩa C thành C:\ tuy nhiên dấu nhắc hệ thống sẽ không thay đổi gì cả.

2.1.8.4. Lệnh tạo thư mục

Cú pháp:

MD <Tên thư mục mới>

hoặc MKDIR <Tên thư mục mới>

Mô tả:

Lệnh MD (viết tắt của MKDIR) cần một tham số đầu vào là tên của thư mục cần tạo. Nếu chưa có thư mục này, thì Windows sẽ tạo ra một thư mục mới tương ứng.

Ví dụ: MD TEST

Lệnh trên sẽ tạo ra thư mục mới có tên là TEST là thư mục con của thư mục hiện hành.

MD C:\TEST

Lệnh trên sẽ tạo ra thư mục mới tên là TEST là thư mục con của thư mục gốc tại ổ đĩa C, bất kể thư mục hiện hành là thư mục nào.

2.1.8.5. Lệnh đổi tên thư mục

Cú pháp:

REN <Tên thư mục cần đổi tên> <Tên mới>

hoặc RENAME <Tên thư mục cần đổi tên> <Tên mới>

Mô tả:

Lệnh REN (viết tắt của RENAME) cần hai tham số đầu vào. Tham số thứ nhất là tên của thư mục cần đổi tên. Tham số thứ hai là tên mới của thư mục này (tên mới phải là thư mục con của thư mục chứa thư mục cần chuyển tên). Nếu thư mục cần đổi tên tồn tại và tên mới không

trùng với một thư mục khác đã có sẵn (nhưng có thể chính là tên cũ của thư mục cần đổi tên) thì Windows sẽ thi hành lệnh đổi tên thư mục.

Ví dụ:

REN C:\TEST C:\TEST1

Lệnh trên đổi tên thư mục TEST thành TEST1 tại thư mục gốc của ổ đĩa C.

2.1.8.6. Lệnh xóa thư mục

Cú pháp:

RD <Tên thư mục cần xóa>

hoặc RMDIR <Tên thư mục cần xóa>

Mô tả:

Lệnh RD (viết tắt của RMDIR) cần một tham số đầu vào cho biết tên thư mục cần xóa. Nếu thư mục này tồn tại và thỏa mãn các điều kiện: Rỗng (không chứa bất kỳ thư mục con hoặc tập tin nào), Không trùng với thư mục hiện hành và các thuộc tính được thiết lập hợp lý, thì Windows sẽ xóa thư mục được yêu cầu.

Ví dụ: RD C:\TEST1

Lệnh trên sẽ xóa thư mục TEST1 tại thư mục gốc của ổ đĩa C mà ta đã tạo ra và sử dụng từ các ví dụ trước.

2.1.8.7. Lệnh tạo tập tin mới

Cú pháp:

COPY CON <Tên tập tin mới>

Mô tả:

Lệnh COPY CON cần một tham số là tên tập tin mới cần tạo. Windows sẽ kiểm tra tính hợp lệ của tên mới. Nếu tên này trùng với tên một thư mục đã tồn tại thì Windows sẽ từ chối thi hành lệnh. Nếu tên mới trùng với tên một tập tin đã tồn tại thì Windows sẽ hỏi là có ghi đè lên tập tin cũ không. Nhấn Y (hoặc A) và Enter nếu muốn (toàn bộ nội dung tập tin cũ sẽ bị xóa) hoặc N và Enter nếu không muốn (Windows sẽ ngừng thi hành lệnh và trả lại dấu nhắc lệnh). Nếu tên mới không có vấn đề gì hoặc người dùng chấp nhận ghi đè lên tập tin cũ thì Windows sẽ chuẩn bị môi trường soạn thảo nội dung tập tin này. Dấu nhắc lệnh mới sẽ không được hiện ra, chỉ còn con trỏ nhấp nháy đứng ngay dưới dấu nhắc lệnh cũ. Lúc này người dùng có toàn quyền gõ vào từ bàn phím nội dung cho tập tin mới sẽ tạo ra. Sau khi gõ xong, để ghi lại: Nhấn phím F6 hoặc tổ hợp phím Ctrl+Z để trên màn hình hiện lên hai ký tự ^Z rồi nhấn Enter. Lúc này Windows mới ghi toàn bộ nội dung người dùng vừa gõ vào tập tin được chỉ định và thông báo: 1 file(s) copied. (Trong khi soạn thảo nếu bạn muốn Windows hủy bỏ việc thi hành lệnh hãy nhấn tổ hợp phím Ctrl+C).

Ví dụ:

Hãy tự nghĩ ra một tên tập tin bạn thích rồi thực hành theo chỉ dẫn. Sau khi gõ xong nội dung và ghi lại, tập tin mới sẽ có ích cho việc thực hành lệnh sau.

2.1.8.8. Lệnh xem nội dung tập tin

Cú pháp:

TYPE <Tên tập tin cần xem>

Mô tả:

Lệnh TYPE sẽ liệt kê nội dung tập tin cần xem lên màn hình. Lưu ý là có thể nội dung của tập tin quá dài, không thể hiện ra hết trong một trang màn hình thì có thể thêm vào cuối lệnh tham số sau: |MORE. (Có nhiều tập tin chứa những thông tin đã được mã hóa, nếu xem nội dung bằng lệnh TYPE sẽ chỉ thấy những ký tự loằng ngoằng chứ không hiểu được tập tin này chứa đựng thông tin gì).

Ví dụ:

Hãy thực hành lệnh với chính tập tin đã tạo ra từ lệnh trước. Sau đó hãy dùng lệnh liệt kê nội dung thư mục để biết tên các tập tin khác và thử lệnh TYPE với các tập tin đó.

2.1.8.9. Lệnh đổi tên tập tin

Cú pháp:

REN <Tên tập tin cũ> <Tên tập tin mới>
hoặc RENAME <Tên tập tin cũ> <Tên tập tin mới>

Mô tả:

Lệnh đổi tên tập tin hoạt động giống như lệnh đổi tên thư mục.

Ví dụ: Hãy tự thực hành với các tập tin mà bạn đã thực hành từ những lệnh trước. Mọi thứ phụ thuộc vào trí tưởng tượng của bạn.

2.1.8.10. Lệnh sao chép tập tin

Cú pháp:

COPY <Từ nguồn> [Đến đích]

Mô tả:

Lệnh COPY sẽ tìm tập tin được chỉ định trong tham số thứ nhất, sau đó sẽ chép tập tin này đến nơi được chỉ định trong tham số thứ hai. Nếu tham số thứ hai không được sử dụng thì Windows sẽ tự hiểu đích là thư mục hiện hành. Sao khi sao chép ta được một tập tin mới giống hệt tập tin cũ. Nếu trong tham số thứ hai có chỉ h dẫn ra một tên tập tin mới, thì tập tin cũ sau khi được sao chép sẽ được đổi lại tên.

Ví dụ: Hãy tự thực hành với các tập tin mà bạn thích.

2.1.8.11. Lệnh xóa tập tin

Cú pháp:

DEL <Tên tập tin cần xóa>
hoặc ERASE <Tên tập tin cần xóa>

Mô tả:

Lệnh DEL (hoặc ERASE) sẽ kiểm tra tập tin được chỉ định trong tham số thứ nhất (lệnh sẽ không xét tham số thứ hai để xóa). Nếu tập tin này tồn tại và được thiết lập thuộc tính hợp lý thì nó sẽ bị xóa.

Ví dụ:

Hãy tự tạo các tập tin của riêng bạn và xóa chúng. Không xóa các tập tin khác trước khi bạn biết chúng được tạo ra để làm gì.

2.2 Hệ điều hành LINUX

2.2.1 Giới thiệu về HĐH Linux

Phiên bản Linux đầu tiên do Linus Torwald, một người Phần Lan phát triển vào năm 1991 dựa trên mã nguồn hệ điều hành Unix được cung cấp bởi phòng thí nghiệm trường đại học công nghệ MIT (Massachusetts Institute of Technology). Thông qua mạng Internet toàn cầu - những kết quả ban đầu với Linux đã được truyền tải nhanh chóng tới rất nhiều tổ chức, cá nhân trên toàn thế giới. Hàng ngàn, hàng chục ngàn lập trình viên “tự do” đã miệt mài hoàn thiện, bổ sung thêm rất nhiều tính năng, cũng như ứng dụng mới cho Linux. Các hãng đi đầu trong phát triển Linux gồm có: RedHat, Caldera, SuSE, Mandrake. Và cho tới thời điểm năm 1995/1996, Linux đã mang dáng dấp của một hệ điều hành Unix thương mại chuyên nghiệp - thừa kế mọi điểm mạnh của Unix như:

- là hệ điều hành hướng mạng
(các dịch vụ internet/intranet: eMail, Web, FTP, . . .),
- thực sự đa nhiệm,
- an toàn, bảo mật cao,
- tính thông minh trong quản lý tài nguyên hệ thống (file, CPU, bộ nhớ).

Hệ điều hành Linux còn có đặc tính nổi bật:

- là sản phẩm miễn phí, chạy được trên các máy tính cá nhân PC - dòng Intel (sử dụng bộ vi xử lý Intel),
- với đặc thù là sản phẩm “mã nguồn mở”, kiểm soát được, từ đó tạo khả năng xây dựng những chức năng phục vụ các yêu cầu riêng biệt - đặc biệt đối với các hệ thống thông tin đòi hỏi tính ổn định và bảo mật cao.

2.2.2 Linux - xu thế, giải pháp mới cho các hệ thống thông tin

Cũng trong những năm gần đây, những tiến bộ công nghệ đã đem lại cho máy tính cá nhân PC một vị thế quan trọng không ngờ (giá thành ngày càng hạ, tính năng ngày càng được nâng cao. Tại rất nhiều nơi PC cao cấp đã thoát hẳn khỏi vai trò là một máy tính đơn lẻ thuần túy, đảm nhiệm vai trò máy chủ tài nguyên, hoặc máy chủ nghiệp vụ cho cả một tổ chức, một cơ quan). Như vậy, giờ đây việc xây dựng một hệ thống thông tin chuyên nghiệp (ổn định, an toàn bảo mật, các dịch vụ hướng mạng: eMail, Web, . . .) với giá thành hạ là điều hoàn toàn khả thi với Linux.

Ví dụ về một hệ thống thông tin hoàn chỉnh xây dựng trên Linux:

- **Samba:** Dịch vụ chia sẻ tài nguyên file.
- **Dhcpd:** Dịch vụ cấp phát địa chỉ IP tự động cho máy trạm.
- **Named:** Dịch vụ DNS - Giải nghĩa tên máy, tên miền -> địa chỉ IP và ngược lại.
- **OpenLDAP:** Dịch vụ sổ địa chỉ phức hợp – cung cấp tới người dùng cuối danh sách địa chỉ eMail, thông tin cá nhân về các thành viên khác trong một tổ chức, một cơ quan.
- **Sendmail:** Dịch vụ gửi, chuyển tiếp thư điện tử (internet eMail).
- **IPOP3d:** Dịch vụ cung cấp hộp thư POP3.
- **Apache:** Dịch vụ Web.

- **Squid:** Máy chủ Proxy – cho phép các máy trạm truy nhập Internet thông qua một máy tính có địa chỉ internet (IP) hợp lệ duy nhất.
- **MySQL, PostgreSQL:** Xây dựng cơ sở dữ liệu.
- **Iptables:** Firewall - Bức tường lửa bảo vệ hệ thống.

Song song với những bước đi vững chắc khi đóng vai trò máy chủ trong các hệ thống thông tin - cũng đã có nhiều biến chuyển tích cực đối với mảng máy tính văn phòng. Ngày càng có nhiều ứng dụng chất lượng được phát triển trên Linux. Có thể kể đến: môi trường giao diện đồ họa mỹ thuật KDE, GNOME, các ứng dụng soạn thảo, lập bảng tính đơn giản, trình soạn thảo văn bản Abiword, bộ ứng dụng văn phòng Koffice, bộ ứng dụng duyệt, thiết kế WEB, gửi/nhận eMail - Netscape Communicator, . . . và đặc biệt là bộ ứng dụng văn phòng OpenOffice - được đánh giá là tương thích hoàn toàn với Ms Office97/2000, và sẽ là một đối thủ nặng ký đối với Microsoft. Sự phong phú của các sản phẩm nguồn mở đã đem tới người dùng một cách nhìn mới về thế giới phần mềm nguồn mở. Giờ đây người dùng có thể lựa chọn cho mình các sản phẩm phù hợp nhất, và với chi phí thấp nhất có thể. (So sánh với chính sách độc quyền sản phẩm, giá thành cao trong trường hợp của sản phẩm phần mềm thương mại - với đại diện tiêu biểu là Microsoft). Ví dụ về một trạm làm việc văn phòng xây dựng trên Linux:

- **GNOME, KDE:** Môi trường desktop đồ họa.
- **Mc, Gmc:** Trình quản lý tệp, thư mục.
- **Mozilla/Netscape Communicator:** Bộ ứng dụng duyệt Web, gửi/nhận eMail, soạn/ thiết kế trang Web, xây dựng sổ địa chỉ cá nhân.
- **Kppp:** Trình quay số, thiết lập kết nối internet.
- **OpenOffice:** Bộ ứng dụng văn phòng tương thích MsOffice97/2000/XP (soạn thảo văn bản, lập bảng tính, soạn bản trình diễn, thiết kế đồ họa, . . .).
- **Kdat:** Tiện ích sao lưu dữ liệu.
- **Fsck:** Tiện ích kiểm tra, khắc phục lỗi xuất hiện với hệ thống tệp, thư mục.
- **Glade, KDevelop:** Môi trường phát triển ứng dụng với ngôn ngữ C/C++.
- **Gcc:** Trình biên dịch C/C++.
- **Kylix:** Môi trường phát triển ứng dụng với ngôn ngữ Pascal (Tương đương với môi trường Delphi phát triển cho Windows). Phù hợp với xu thế của thế giới thông tin đa ngôn ngữ - các phiên bản Linux mới nhất, các ứng dụng chạy trên Linux cũng đã được bổ sung những khả năng hỗ trợ Unicode/UTF-8 ngày càng hoàn thiện hơn. Những lợi thế do Linux cũng như các sản phẩm phần mềm nguồn mở đem lại đã kéo theo sự quan tâm với mức độ ngày càng tăng - từ mức chính phủ, tới các đại gia trong ngành công nghệ thông tin, cho tới các doanh nghiệp vừa và nhỏ. Chính phủ nhiều nước như Trung Quốc, Đài Loan, Đức, Pháp đã cân nhắc phần mềm nguồn mở như là giải pháp an toàn và kinh tế nhất cho hệ thống thông tin của họ. Những người khổng lồ như: IBM, Intel, Compaq, HP, Oracle, và SunMicrosystem đã có những cam kết hỗ trợ, và gia tăng đầu tư trong lĩnh vực phần mềm nguồn mở.

2.2.3 Một số khái niệm cơ bản trong Linux

2.2.3.1. Cấu trúc thư mục

Hệ thống tệp của Linux, với tên gọi ext2, là hệ thống thư mục dạng cây. Thư mục gốc (ký hiệu là /), là gốc của toàn bộ hệ thống tệp. Khái niệm “ổ” (ổ C, ổ A, . .) của Windows không sử

dụng trong hệ thống tệp của Linux. Thay vào đó, các thiết bị như ổ mềm sẽ được ánh xạ vào một thư mục nào đó nằm dưới thư mục gốc của toàn bộ hệ thống. Cũng giống như trong Windows, các thư mục có thể chứa thư mục con và các tệp. Tên tệp trong Linux có thể tùy ý và có thể chứa cả chữ số, dấu cách, dấu gạch, . . . Tệp trong Linux phân biệt chữ hoa và chữ thường.

```

[rooth@maria Mail]# cd ..
[rooth@maria /root]# ls -la
total 25
drwxr-xr-x 3 root root 1024 Apr 23 18:38 .
drwxr-xr-x 15 root root 1024 Apr 24 1998 ..
-rw-r--r-- 1 root root 349 Apr 23 18:38 .FIMM2-errors
-rw-r--r-- 1 root root 1126 Aug 24 1996 .Xdefaults
-rw-r--r-- 1 root root 2301 Apr 23 18:38 .bash_history
-rw-r--r-- 1 root root 24 Jul 14 1994 .bash_logout
-rw-r--r-- 1 root root 238 Aug 24 1996 .bash_profile
-rw-r--r-- 1 root root 176 Aug 24 1996 .bashrc
-rw-r--r-- 1 root root 182 Oct 8 1998 .cshrc
drwxr-xr-x 2 root root 1024 Apr 3 12:11 .elm
drwxr-xr-x 2 root root 1024 Apr 2 2011 .gnome
drwxr-xr-x 2 root root 1024 Apr 2 2011 .gnome_private
drwxr-xr-x 4 root root 1024 Apr 3 12:09 .netscape
-rw-r--r-- 1 root root 166 Mar 4 1996 .tcshrc
-rw-r--r-- 1 root root 7 Apr 23 18:38 .vim_style
drwxr-xr-x 2 root root 1024 Apr 3 12:11 Mail
-rw-r--r-- 1 root root 315 Apr 23 18:38 Xrootenv.0
drwxr-xr-x 2 root root 1024 Apr 23 07:26 abc
-rw-r--r-- 1 root root 51 Apr 3 12:16 backally.rm
drwxr-xr-x 2 root root 1024 Apr 2 06:28 ossail
[rooth@maria /root]#
  
```

Hình 2.12: Kết quả của lệnh “ls -la”.

Tên đường dẫn (pathname) có thể là tuyệt đối hoặc tương đối. Tên đường dẫn tuyệt đối bắt đầu bằng “/” và tại thư mục gốc (Windows sử dụng “\”). Các thư mục con kế tiếp sau đó cũng được phân cách bằng “/”. Cũng giống như hệ thống tệp của Windows, hệ thống tệp của Linux lưu giữ thông tin về các tệp và thư mục.

Các thông tin này bao gồm quyền đối với tệp/thư mục, người tạo tệp/thư mục, nhóm (group) của tệp/thư mục đó. Người dùng có thể xem quyền truy cập của mỗi tệp bằng cách sử dụng lệnh “ls -l”. Quyền truy cập được chia làm ba lớp người dùng: người tạo ra, nhóm làm việc mà nó thuộc về, và tất cả các người dùng khác. Quyền cho mỗi loại người dùng được chia làm ba loại: quyền đọc (r), quyền viết (w) và quyền thực hiện (x). Cách phân quyền của thư mục khác với tệp. Với một thư mục, quyền thực hiện có nghĩa là người dùng có truy cập vào sâu trong thư mục đó. Quyền đọc của thư mục có nghĩa là người dùng có thể xem tên các tệp/thư mục trong thư mục đó. Quyền viết đối với thư mục có nghĩa là người dùng có thể tạo ra một tệp/thư mục trong thư mục đó.

Ví dụ, nếu ta gõ “ls -l”, thông tin hiện ra như sau:

```

drwxr-xr-x 2 userid users 1024 May 3 11 : 24 temp
rw-rw-rw- 1 userid users 5400 Jan 8 2001 thcs.doc
  
```

Trong ví dụ này, thư mục hiện thời chứa một thư mục con và một tệp. Người dùng có quyền đọc và quyền truy cập vào trong thư mục con, còn đối với tệp thì chỉ người tạo ra nó mới có thể đọc và viết vào tệp đó. Ngoài ra không ai có quyền gì với tệp đó.

2.2.3.2. Ký hiệu tắt trong Linux

- “.” cho biết đó là thư mục làm việc;
- “..” chỉ thư mục cao hơn một cấp (thường gọi là thư mục mẹ);
- “ ” chỉ thư mục riêng (home directory).

Những ký hiệu này có thể được sử dụng cùng với nhau.

Ví dụ: “/.” có nghĩa là thư mục mẹ của thư mục riêng.

Hoặc bạn có thể dùng “../” để chỉ một thư mục cao hơn thư mục mẹ.

2.2.3.3. Thư mục tiêu biểu của Linux

Một tên đường dẫn tương đối bắt đầu với thư mục mà bạn đang ở đó (gọi là thư mục làm việc; bạn nhập “pwd” và nhấn <Enter> nếu không biết đang ở thư mục nào) và chuyển xuống thư mục thấp hơn. Những tên đường dẫn tương đối bắt đầu với tên thư mục nằm dưới thư mục làm việc. Mỗi thư mục cấp thấp hơn phải được thể hiện với dấu “/” trước đó. Ví dụ: có một thư mục /usr với cài đặt chuẩn của RHL, dưới đó bạn sẽ tìm thấy nhiều thư mục con khác, bao gồm cả thư mục /bin (dành cho các tệp nhị phân). Tên đường dẫn tuyệt đối cho bin là /usr/bin. Nhưng do đang nằm trong /usr nên thay vì /usr/bin bạn chỉ cần gõ bin trong đối số của dòng lệnh. Sau khi đăng nhập, bạn sẽ được vào thư mục chứa các tệp của mình, gọi là thư mục riêng của bạn (home directory). Thông thường, một thư mục riêng của người dùng Linux có dạng /usr/user (user là username của bạn), nhưng root lại có thư mục riêng là /root (trong RHL). Linux, giống như Unix, nhạy cảm với chữ thường và chữ hoa trong hầu hết mọi sự kiện, kể cả khi bạn nhập username, password hay câu lệnh. Khi đã đăng nhập, hệ thống đưa ra dấu nhắc lệnh, và bạn có thể yêu cầu máy tính thực hiện những gì bạn muốn - bằng cách nhập dòng lệnh. Điều này làm cho người dùng Windows, vốn quen với phong cách trỏ-và-nhấn, cảm thấy không thoải mái trong lần dùng đầu tiên. Nhưng khi đã quen, bạn sẽ nhận thấy dòng lệnh tỏ ra hiệu quả hơn nhiều đối với người dùng có kinh nghiệm, và chỉ cần sau một ngày, bạn có thể sử dụng Linux một cách thành thạo. Dấu nhắc lệnh được đưa ra bởi một tiện ích shell (trình giao diện), khởi động mỗi khi bạn đăng nhập.

Các lệnh trong Unix thường bắt đầu bằng tên lệnh (command name), sau đó là cờ (flag) và đối số (argument).

Cú pháp tổng quát là:

command [flag] argument1 argument2 . . .

Những gì nằm giữa hai dấu ngoặc vuông - []- có nghĩa là tùy chọn. Nếu đã từng sử dụng Windows, bạn sẽ thấy nó cũng có cú pháp gần giống như vậy. Tuy nhiên, vẫn có vài khác biệt. Các lệnh Windows có tham số (hay flag) đứng sau ký tự “/”; trong khi Linux sử dụng ký tự “-”. Ví dụ, trong Windows bạn gõ “dir/a /o:d”, còn trong Linux lại là “ls -la” để thực hiện cùng một mục tiêu.

2.2.4 Môi trường đồ họa

Môi trường desktop đồ họa cho Linux cũng giống như môi trường desktop của Windows hay Mac OS. Chúng ta có thể chạy một chương trình bằng cách nhấp vào biểu tượng của nó hoặc chọn từ bảng chọn. Hai môi trường đồ họa rất phổ biến của Linux là KDE và GNOME. Nút bắt đầu (Start) của KDE hay GNOME nằm bên trái dưới của màn hình (xem Hình 2.13). Khi nhấp chuột vào đó, bảng chọn chính sẽ hiện ra cho phép chúng ta lựa chọn và chạy chương trình.

passwd: thay đổi mật khẩu.
cp: copy tệp (giống lệnh copy của Windows).
mv: đổi tên tệp.
rm: xóa tệp.
pwd: in ra đường dẫn của thư mục hiện thời.
ls: liệt kê nội dung thư mục.
cd: đổi thư mục.
mkdir: tạo thư mục.
rmdir: xóa thư mục.

2.2.4.5. Chuyển đổi màn hình nền (Desktop)

Không giống như hệ điều hành Windows hay Mac, môi trường đồ họa của Linux cho phép chúng ta cấu hình nhiều desktop ảo. Với mỗi desktop, chúng ta có một tập các cửa sổ khác nhau. Theo mặc định, môi trường đồ họa của Linux có bốn desktop ảo. Chúng ta có thể chuyển đổi giữa các desktop bằng cách nhấp chuột vào một trong bốn nút đại diện cho bốn desktop. Bốn nút này nằm ở thanh công cụ phía dưới màn hình.

2.2.4.6. Thao tác với cửa sổ (thu nhỏ, phóng đại và đóng)

Cách dễ nhất thực hiện những hành động trên với cửa sổ là thao tác với những nút ở góc trên phải của cửa sổ.

- Nhấp chuột vào phím X để đóng cửa sổ.
- Nhấp chuột vào bình vuông để phóng đại cửa sổ.
- Nhấp chuột vào nút chấm nhỏ để thu nhỏ cửa sổ.

Về cơ bản, trên môi trường Desktop đồ họa, việc sử dụng Linux cũng tương tự như sử dụng Windows.

Bài đọc thêm. Linux, các sản phẩm phần mềm nguồn mở tại Việt nam

Phù hợp với xu thế chung, chúng ta đã có những chuyển biến tích cực trong việc quảng bá cũng như thúc đẩy phong trào Linux nói riêng cũng như phần mềm nguồn mở nói chung tại Việt nam. Sau hội thảo về Linux (lần đầu tại Việt nam) tháng 12/2000, số lượng người dùng Linux (người dùng cuối, các tổ chức, cơ quan), số lượng đơn vị tham gia nghiên cứu, phát triển Linux, ứng dụng trên Linux đã tăng lên rõ rệt. Và điều đáng mừng là: các sản phẩm ra đời trong thời gian ngắn vừa qua (từ năm 2000 đến nay) đều thực sự là các sản phẩm có giá trị sử dụng cao (không chỉ dừng ở mức nghiên cứu, thử nghiệm), khẳng định được hiệu quả trong sử dụng thực tế (chất lượng/giá thành). Những kết quả khích lệ này cũng đã tác động tích cực tới những tổ chức/cá nhân tham gia công tác hoạch định phương hướng phát triển ngành công nghệ thông tin nước nhà - đặc biệt khi Việt nam xúc tiến thực thi các hiệp định thương mại, các hiệp định về bảo vệ quyền tác giả (thể hiện rõ nét nhất khi sử dụng các phần mềm thương mại) với Hoa Kỳ. Các đơn vị tham gia nghiên cứu, ứng dụng, và phát triển Linux cũng như phần mềm nguồn mở tại Việt nam

Từ năm 1999 trong nước đã có nhiều nhóm nghiên cứu và tiến hành bản địa hoá hệ điều hành Linux:

- Công ty máy tính truyền thông CMC
- Tổng công ty điện tử tin học Việt Nam – VEIC
- Công ty Việt Khang
- Công ty phát triển phần mềm VISC
- Trường Đại học Bách khoa Hà nội
- Trường Đại học Bách khoa Thành phố Hồ Chí Minh
- Trung tâm tin học Bộ quốc phòng

Các hệ điều hành Việt hoá cho đến năm 2001 thường dùng bản Redhat 6.2. Đến thời điểm hiện nay, đáng kể nhất có Hệ điều hành tiếng Việt của công ty CMC, và Hệ điều hành Vietkey Linux của Vietkey Group – VEIC. Cả 2 hệ điều hành này đều đã hỗ trợ tiếng Việt Unicode dựng sẵn, tuân thủ TCVN 6909:2001 và sẵn sàng theo Quyết định số 72 của Thủ tướng Chính phủ về việc chuyển đổi sang bộ mã tiếng Việt Unicode. Một thực tế là bên cạnh vai trò đã được khẳng định trong lĩnh vực máy chủ, trong lĩnh vực xây dựng các hệ thống thông tin - nếu chúng ta muốn thúc đẩy sử dụng hệ điều hành Linux, cũng như các ứng dụng nguồn mở trên phạm vi rộng rãi và rất quan trọng là “môi trường làm việc văn phòng”, khi đó chúng ta sẽ nhận thức được tầm quan trọng của việc bản địa hoá các sản phẩm phần mềm nguồn mở.

Chương 3

THUẬT GIẢI

3.1. Khái niệm

Khi viết một chương trình máy tính, ta thường cài đặt một phương pháp đã được nghĩ ra trước đó để giải quyết một vấn đề. Phương pháp này thường là độc lập với một máy tính cụ thể sẽ được dùng, nó hầu như thích hợp như nhau cho nhiều máy tính. Trong bất kỳ trường hợp nào thì phương pháp phải được nghiên cứu để làm thế nào đi sâu vào bài toán. Từ "Thuật giải" được dùng trong khoa học máy tính để mô tả một phương pháp giải bài toán. Thuật toán là "Chất liệu" của khoa học máy tính, chúng là những đối tượng nghiên cứu trung tâm trong nhiều, nếu không nói là hầu hết các lĩnh vực của Tin học. Thuật giải nổi tiếng nhất có từ thời cổ Hy Lạp là thuật giải Euclid, thuật giải tìm ước số chung lớn nhất của 2 số m và n .

Thuật toán này được mô tả như sau:

- Dữ liệu vào (Input): m, n nguyên dương, giả sử $m > n$
- Dữ liệu ra (Output): U , ước số chung lớn nhất của m và n
- Phương pháp giải (Method):
 - +Bước 1: Tìm phần dư r của phép chia m cho n ($r = m \bmod n$)
 - +Bước 2: Nếu $r=0$ thì gán giá trị của n cho U ($U=n$) và dừng lại. Trong trường hợp ngược lại ($r \neq 0$) thì $m = n$, $n = r$ và quay lại bước 1.

Chúng ta có thể quan niệm các bước cần thực hiện để nấu một nồi cơm là một thuật giải.

Chúng ta đưa ra định nghĩa không hình thức về thuật giải như sau:

Thuật giải là một dãy hữu hạn các bước, mỗi bước mô tả chính xác các phép toán hoặc hành động cần thực hiện, để giải quyết một vấn đề.

Với một vấn đề đặt ra, có thể có nhiều thuật toán giải. Một vấn đề có thuật toán giải gọi là vấn đề giải được (bằng thuật toán). Chẳng hạn vấn đề tìm nghiệm của hệ phương trình tuyến tính là vấn đề giải được. Một vấn đề không tồn tại thuật toán giải gọi là vấn đề không giải được (bằng thuật toán).

Trên đây chúng ta đã trình bày định nghĩa không hình thức về thuật giải. Có thể xác định khái niệm thuật giải một cách chính xác bằng cách sử dụng các hệ hình thức như: máy Turing, hệ thuật toán Markôp, văn phạm Chomsky dạng 0,... Song vấn đề này không thuộc phạm vi chúng ta quan tâm. Đối với chúng ta, chỉ sự hiểu biết trực quan, không hình thức về khái niệm thuật giải là đủ.

3.2 Các đặc trưng của thuật giải

Knuth – tác giả của bộ sách nổi tiếng “Nghệ thuật lập trình” đã đưa ra 5 đặc trưng sau đây của thuật toán:

1. *Input*: Mỗi thuật giải cần có một số (có thể bằng không) dữ liệu vào. Đó là giá trị cần đưa vào khi thuật giải bắt đầu làm việc. Chẳng hạn trong thuật giải Euclid trên thì m và n là các dữ liệu vào

2. *Output*: Mỗi thuật giải cần có một hoặc nhiều dữ liệu ra. Đó là các giá trị có quan hệ xác định với giá trị vào và là kết quả của sự thực hiện thuật giải. Trong thuật giải Euclid thì dữ liệu ra là U.
3. *Tính xác định*: Mỗi bước của thuật giải cần phải được mô tả một cách chính xác, chỉ có một cách hiểu duy nhất. Bởi vì, nếu một bước có thể hiểu theo nhiều cách khác nhau, thì cùng một dữ liệu vào, những người thực hiện thuật giải khác nhau có thể cho ra kết quả khác nhau. Nếu ta mô tả thuật giải bằng ngôn ngữ tự nhiên, không có gì đảm bảo người đọc hiểu đúng ý của người viết thuật giải, bởi ngôn ngữ tự nhiên thường là đa nghĩa. Để đảm bảo đòi hỏi này, thuật giải cần được mô tả trong một ngôn ngữ lập trình, chẳng hạn một ngôn ngữ lập trình (Pascal, C...). trong các ngôn ngữ này, các mệnh đề được tạo thành theo các qui tắc cú pháp nghiêm ngặt và chỉ có một ý nghĩa duy nhất.
4. *Tính khả thi*: Tất cả các phép giải có mặt trong các bước của thuật giải phải đủ đơn giản. Điều đó có nghĩa là các phép giải phải sao cho, ít nhất về nguyên tắc có thể thực hiện được bởi con người trong một khoảng thời gian hữu hạn. Chẳng hạn trong thuật giải Euclid, ta chỉ cần thực hiện các phép chia các số nguyên, các phép gán và các phép so sánh.
5. *Tính dừng*: Với mọi bộ dữ liệu vào thoả mãn điều kiện qui định, thuật giải phải dừng lại sau một số hữu hạn bước thực hiện. Chẳng hạn đối với thuật giải Euclid là thuật giải thoả mãn điều kiện này. Bởi vì số dư r luôn nhỏ hơn n , nếu $r \neq 0$ thì giá trị của n ở bước sau là giá trị của r ở bước trước, ta có $n > r = n_1 > r_1 = n_2 > r_2 \dots$. Dãy số nguyên dương giảm dần cần phải kết thúc ở 0, do đó sau một số bước giá trị của $r = 0$ thuật giải dừng.

3.3 Các phương pháp biểu diễn thuật giải

3.3.1 Ngôn ngữ tự nhiên

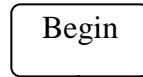
Trong cách biểu diễn thuật giải theo ngôn ngữ tự nhiên, người ta sử dụng ngôn ngữ thường ngày để liệt kê các bước của thuật giải (Các ví dụ về thuật giải trong mục 1 của chương sử dụng ngôn ngữ tự nhiên). Phương pháp biểu diễn này không yêu cầu người viết thuật giải cũng như người đọc thuật giải phải nắm các quy tắc. Tuy vậy, cách biểu diễn này thường dài dòng, không thể hiện rõ cấu trúc của thuật giải, đôi lúc gây hiểu lầm hoặc khó hiểu cho người đọc. Gần như không có một quy tắc cố định nào trong việc thể hiện thuật giải bằng ngôn ngữ tự nhiên. Tuy vậy, để dễ đọc, ta nên viết các bước con lù vào bên phải và đánh số bước theo quy tắc phân cấp như 1, 1.1, 1.1.1, ... Bạn có thể tham khảo lại ba ví dụ trong mục 1 của chương để hiểu cách biểu diễn thuật giải theo ngôn ngữ tự nhiên.

3.3.2 Lưu đồ - sơ đồ khối

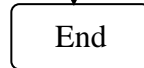
Lưu đồ hay sơ đồ khối là một công cụ trực quan để diễn đạt các thuật giải. Biểu diễn thuật giải bằng lưu đồ sẽ giúp người đọc theo dõi được sự phân cấp các trường hợp và quá trình xử lý của thuật giải. Phương pháp lưu đồ thường được dùng trong những thuật giải có tính rắc rối, khó theo dõi được quá trình xử lý.

Vẽ lưu đồ hoặc sơ đồ khối, mỗi khối mô tả một bước. Sử dụng mũi tên nối khối này với khối kia để chỉ trình tự thực hiện các bước. Để dễ dàng nhận biết chức năng của từng khối, mỗi khối được thể hiện ở một dạng như sau:

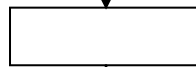
- Bắt đầu



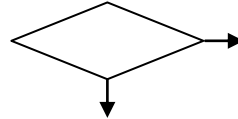
- Kết thúc



- Thực hiện công việc



- Điều kiện

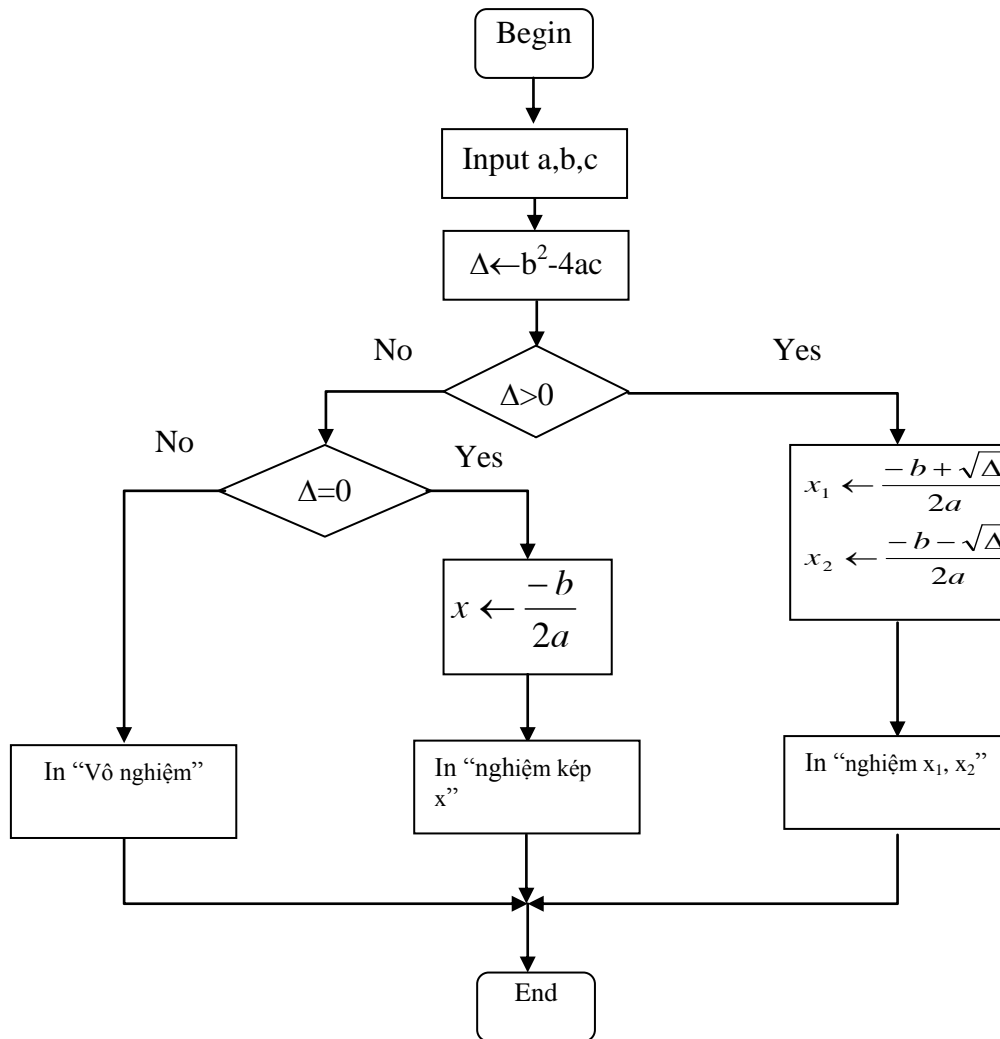


Ví dụ 3.1: Tìm nghiệm thực của phương trình bậc hai $ax^2+bx+c=0$. Thuật toán được dạy gồm các bước sau:

- + Nhập dữ liệu a, b, c.
- + Tính biệt số $\Delta = b^2-4ac$.
- + So sánh Δ với 0. Bước tiếp theo sẽ phụ thuộc vào giá trị của Δ .
- + Nếu $\Delta < 0$, đáp số của bài toán là “Vô nghiệm”.
- + Nếu $\Delta = 0$, tính và ghi đáp số trị của nghiệm kép $x = -\frac{b}{2a}$.
- + Nếu $\Delta > 0$, tính và ghi đáp số trị của hai nghiệm phân biệt:

$$x_1 = \frac{-b + \sqrt{\Delta}}{2a} \quad x_2 = \frac{-b - \sqrt{\Delta}}{2a}.$$

Lưu đồ biểu diễn thuật toán này được thể hiện như sau:



3.3.3. Mã giả

Tuy sơ đồ khối thể hiện rõ quá trình xử lý và sự phân cấp các trường hợp của thuật toán nhưng lại cồng kềnh. Để mô tả một thuật toán nhỏ ta phải dùng một không gian rất lớn.

Khi thể hiện thuật toán bằng mã giả, ta sẽ vay mượn các cú pháp của một ngôn ngữ lập trình nào đó để thể hiện thuật toán. Tất nhiên, mọi ngôn ngữ lập trình đều có những thao tác cơ bản là xử lý, rẽ nhánh và lặp. Dùng mã giả vừa tận dụng được các khái niệm trong ngôn ngữ lập trình, vừa giúp người cài đặt dễ dàng hiểu chính xác thuật toán. Tất nhiên là trong mã giả ta vẫn dùng một phần ngôn ngữ tự nhiên. Một khi đã vay mượn cú pháp và khái niệm của ngôn ngữ lập trình thì chắc chắn mã giả sẽ bị phụ thuộc vào ngôn ngữ lập trình đó. Chính vì lý do này, chúng ta chưa vội tìm hiểu về mã giả trong bài này (vì chúng ta chưa biết gì về ngôn ngữ lập trình!). Sau khi tìm hiểu xong bài về thủ tục - hàm bạn sẽ hiểu mã giả là gì !

Một đoạn mã giả của thuật toán giải phương trình bậc hai:

```

if Delta > 0 then
    begin
        x1 = (-b + sqrt(delta)) / (2*a)
    
```

```
        x2=(-b-sqrt(delta))/(2*a)
        xuất kết quả : phương trình có hai nghiệm là x1 và x2
    end
else
    if delta = 0 then
        xuất kết quả : phương trình có nghiệm kép là -b/(2*a)
    else {trường hợp delta < 0 }
        xuất kết quả : phương trình vô nghiệm
```

BÀI TẬP CHƯƠNG 3

Bài 3.1. Cho tam giác ABC có góc vuông A và cho biết cạnh a và góc B . Hãy viết thuật toán để tính góc C , cạnh b và cạnh c .

Bài 3.2. Trình bày tính chất xác định của thuật toán và nêu rõ nghĩa của tính chất này.

Bài 3.3*. Hãy phát biểu thuật toán để giải bài toán sau: “Có một số quả táo. Dùng cân hai đĩa (không có quả cân) để xác định quả táo nặng nhất”.

Bài 3.4. Căn cứ vào các đặc trưng của thuật toán, hãy xem quy tắc sau đây có phải là một thuật toán hay không: “Giả sử cho trước một cuốn sách tiếng Việt.

- a) Hãy mở một trang có chữ số tận cùng bằng 5;
- b) Hãy lấy từ đầu tiên của trang đó;
- c) Hãy xem chữ cái đầu tiên của từ đó nếu chữ cái đó là một chữ từ A đến H thì bạn đi dạo, nếu không thì ở nhà”.

Bài 3.5. Xác định input và output cho các thuật toán sau đây:

- a) Rút gọn một phân số.
- b) Kiểm tra xem ba số cho trước a , b và c có thể là độ dài ba cạnh của một tam giác hay không?
- c) Tính trung bình cộng của hai số.
- d) Dùng một cốc phụ để tráo nước ở hai cốc cho trước.
- e) Tìm chu vi và diện tích của hình tròn có bán kính cho trước.

Bài 3.6. Chỉ dùng phép cộng, viết thuật toán để từ số tự nhiên n , tính số n^2 .

Chương 4

CÁC YẾU TỐ CƠ SỞ CỦA NGÔN NGỮ PASCAL

4.1. Giới thiệu ngôn ngữ PASCAL

PASCAL là ngôn ngữ lập trình cấp cao được giáo sư Niklaus Wirth ở trường đại học Kỹ thuật Zurich (Thụy sĩ) thiết kế và công bố vào năm 1971. Ông đặt tên cho ngôn ngữ của mình là Pascal để tưởng nhớ nhà toán học nổi tiếng người Pháp ở thế kỷ 17: Blaise Pascal, người đã sáng chế ra chiếc máy tính cơ khí đầu tiên của nhân loại. Qua thời gian sử dụng, Pascal ngày càng được đông đảo người dùng đánh giá cao, và trở thành một trong các ngôn ngữ lập trình phổ biến nhất hiện nay.

Thành công của ngôn ngữ Pascal là ở chỗ: nó là ngôn ngữ đầu tiên đưa ra và thể hiện được khái niệm lập trình có cấu trúc. Ý tưởng về một chương trình có cấu trúc xuất phát từ suy nghĩ cho rằng có thể chia một bài toán lớn, phức tạp thành nhiều bài toán nhỏ, đơn giản hơn. Nếu mỗi bài toán nhỏ được giải quyết bằng một chương trình con, thì khi liên kết các chương trình con này lại sẽ tạo nên một chương trình lớn giải quyết được bài toán ban đầu.

Bằng cách chia một chương trình thành các chương trình con như vậy, người lập trình có thể lập trình để giải quyết riêng lẻ từng phần một, từng khối một, hoặc có thể tổ chức để nhiều người cùng tham gia, mỗi người phụ trách một vài khối. Đặc biệt khi phải thay đổi hay sửa chữa trong một khối thì điều đó sẽ ít ảnh hưởng đến các khối khác.

Tính cấu trúc của ngôn ngữ Pascal còn thể hiện trong việc tổ chức các câu lệnh và tổ chức dữ liệu. Từ các lệnh đã có, người lập trình có thể nhóm chúng lại với nhau và đặt giữa hai từ khóa Begin và End tạo thành một câu lệnh mới phức tạp hơn gọi là câu lệnh ghép. Đến lượt mình, hai hay nhiều lệnh ghép lại có thể được nhóm lại để tạo thành một câu lệnh ghép phức tạp hơn nữa, v.v. Tương tự như thế, ngôn ngữ Pascal cũng cho phép xây dựng các kiểu dữ liệu phức tạp hơn từ các kiểu dữ liệu đã có.

Pascal là một ngôn ngữ không chỉ chặt chẽ về mặt cú pháp mà còn chặt chẽ về mặt dữ liệu. Mỗi biến, mỗi hằng tham gia trong chương trình luôn có một kiểu dữ liệu xác định và chỉ nhận những giá trị có cùng kiểu dữ liệu với nó. Điều này buộc người lập trình phải nắm chắc cú pháp và luôn chú ý đến tính tương thích của các biểu thức về mặt kiểu dữ liệu. Chính vì thế, lập trình bằng ngôn ngữ Pascal là một cơ hội tốt không chỉ rèn luyện tư duy mà còn rèn luyện tính cẩn thận và chính xác.

Ngày nay, ngôn ngữ Pascal được dùng để viết các chương trình ứng dụng trong nhiều lĩnh vực. Với văn phạm sáng sủa, dễ hiểu, với khả năng đủ mạnh, Pascal được xem là ngôn ngữ thích hợp nhất để giảng dạy ở các trường phổ thông và đại học.

4.2. Các thành phần cơ bản của ngôn ngữ PASCAL

4.2.1 Bộ ký tự cơ bản

Mỗi ngôn ngữ đều được xây dựng từ một tập ký tự nào đó. Nhiều ký tự nhóm lại với nhau tạo nên các từ. Nhiều từ liên kết với nhau theo một qui tắc ngữ pháp nhất định (gọi là văn phạm) thì tạo nên các mệnh đề. Trong các ngôn ngữ lập trình, mệnh đề còn được gọi là câu lệnh. Một tập hợp các câu lệnh được sắp xếp theo một trật tự nhất định nhằm chỉ thị cho máy các thao tác phải thực hiện tạo thành một chương trình. Các chương trình được soạn thảo bởi người lập trình và được lưu trữ trên đĩa dưới dạng các tập tin.

Ngôn ngữ Pascal được xây dựng trên bộ ký tự cơ bản, gồm:

- Các chữ cái la tinh: A, B, C,...,Z, a, b, c,..., z
- Các chữ số : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Các ký hiệu đặc biệt: +, -, *, /, =, <, {, }, [,], %, \$, &, #, ...
- Ký tự gạch nối '_' và ký tự trắng ' ' (space)
- Các chữ ả rập: α , β , δ , ... không thuộc bộ ký tự của Pascal.

4.2.2 Từ khóa (key word)

Có một số từ được Pascal dành riêng cho việc xây dựng các câu lệnh, các khai báo, các phép tính,... gọi là từ khóa. Việc sử dụng các từ khóa đòi hỏi phải tuân thủ đúng quy tắc đề ra, và đặc biệt là người lập trình không được đặt một tên mới (tên biến, tên hằng, tên hàm, tên thủ tục,...) trùng với một trong các từ khóa. Dưới đây là danh sách các từ khóa của Pascal :

absolute, and, array, begin, case, const, div, do, downto, else, end, file, for, forward, function, goto, if, implementation, in, inline, interface, interrupt, label, mod, nil, not, of, or, packed, procedure, program, record, repeat, set, shl, shr, string, then, to, type, unit, until, uses, var, while, with, xor

*Các từ khóa có thể viết dưới dạng chữ hoa hay chữ thường hay xen kẽ chữ hoa với chữ thường đều được. Ví dụ viết **begin** hay **Begin** hay **BEGIN** là như nhau.*

4.2.3 Tên (identifier)

Các biến, các hằng, các hàm, các thủ tục, ... được sử dụng trong chương trình đều cần phải đặt tên, còn gọi là định danh hay danh hiệu.

Các tên này do người lập trình tự đặt và phải đảm bảo đúng quy tắc:

- Tên chỉ gồm chữ số, chữ cái và dấu gạch nối (gạch dưới '_')
- Không bắt đầu bởi một chữ số
- Không trùng với từ khóa
- Chiều dài của tên tối đa là 127 ký tự.

- Thông thường tên nên đặt ngắn gọn và có tính gợi nhớ.

Ví dụ về các tên được đặt đúng:

Delta, X1, X2, i, j, Chuc_vu, Luong, So_luong, Don_gia.

Các tên được đặt sai:

3ABC, In, Chu vi, Ma-so

vì :

3ABC: bắt đầu bằng số

Chu vi: có chứa ký tự trắng

Ma-so : ký tự '-' là dấu trừ chứ không phải gạch nối.

In : trùng với từ khóa In

Cũng giống như từ khóa, Tên không phân biệt viết hoa hay viết thường.

Ví dụ viết X1 hay x1 cũng chỉ là một tên thôi.

Trong Pascal có một số tên đã được đặt sẵn rồi, gọi là tên chuẩn, chẳng hạn:

Abs, Arctan, Boolean, Byte, Char, Cos, Copy, Delete, Eof, False, Longint, Ord, Integer, Real, Readln, Writeln, True, Text, ...

Mặc dù người lập trình có thể đặt một tên mới trùng với một trong các tên chuẩn, song, để đỡ nhầm lẫn, chúng ta nên tránh điều này.

4.2.4. Các dấu đặc biệt

- Dấu kép: cặp các dấu viết liền nhau
ví dụ: <=; >=; :=; ...
- Dấu “,” ngăn cách các câu lệnh.
- Dấu chú thích sử dụng bao các câu là lời giải thích cho các đoạn của chương trình và các câu chú thích được chương trình dịch bỏ qua trong quá trình dịch. Có thể viết dòng chú thích tại bất cứ vị trí nào trong chương trình mà có thể để dấu cách và theo cú pháp:
(* lời chú thích *) hoặc { lời chú thích }

4.3. Các kiểu dữ liệu đơn giản

4.3.1 Khái niệm

Chức năng của máy điện toán là xử lý các thông tin. Các thông tin được nhập và lưu trữ trong bộ nhớ của máy dưới các dạng khác nhau: có thể là số, là chữ, có thể là hình ảnh, âm thanh, v.v. mà thuật ngữ tin học gọi chung là dữ liệu. Tính đa dạng của dữ liệu đòi hỏi phải tổ chức và phân phối bộ nhớ thích hợp để lưu trữ và xử lý tốt các dữ liệu. Ngôn ngữ lập trình chia các dữ liệu thành từng nhóm riêng trên đó xây dựng một số phép toán tạo nên các kiểu dữ liệu khác nhau, mỗi kiểu dữ liệu là một tập hợp các giá trị mà một biến thuộc kiểu đó có thể nhận. Khi một biến được khai báo thuộc kiểu dữ liệu nào thì máy sẽ dành cho biến đó một dung lượng thích hợp trong bộ nhớ để có thể lưu trữ các giá trị thuộc kiểu dữ liệu đó.

4.3.2. Phân loại các kiểu dữ liệu trong Turbo Pascal

Trong Pascal, ngôn ngữ lập trình cấu trúc nên có thể chia các kiểu dữ liệu thành hai loại là:

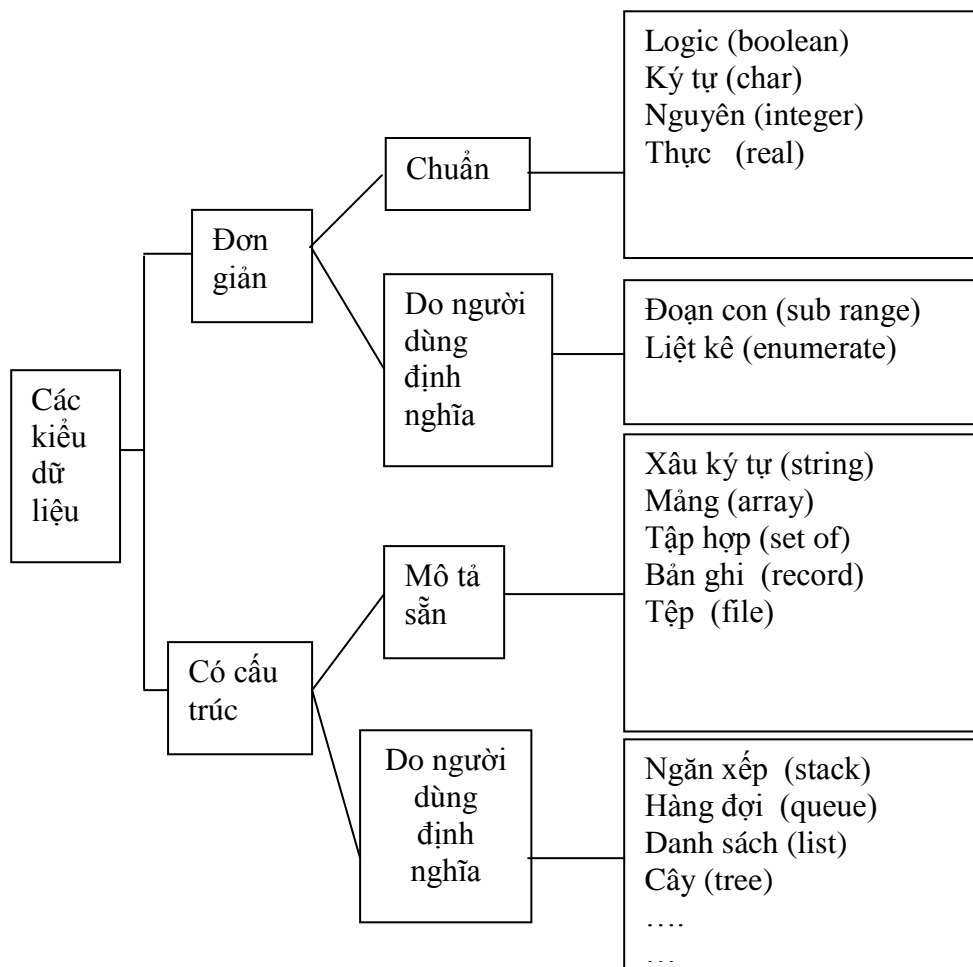
- **Kiểu dữ liệu đơn giản** là những kiểu dữ liệu cơ sở nhất, không thể phân chia nhỏ hơn nữa thành một số thành phần có nghĩa.
- **Kiểu dữ liệu có cấu trúc** là những kiểu dữ liệu được xây dựng bằng cách ghép một số dữ liệu đơn giản theo cấu trúc cú pháp xác định.

Với mỗi kiểu dữ liệu đơn giản hay cấu trúc lại có thể phân thành những kiểu khác nhau đó là

- **Kiểu dữ liệu có sẵn:** là những kiểu đã được xây dựng sẵn với các tính chất hoàn toàn xác định, ta có thể sử dụng ngay mà không cần phải bổ xung, sửa chữa gì. Ta có thể gọi các kiểu dữ liệu đó là các kiểu dữ liệu chuẩn.

Ví dụ như các kiểu nguyên, các kiểu thực, kiểu ký tự, kiểu logic,

- **Kiểu dữ liệu chưa có sẵn,** những kiểu dữ liệu mà khi cần sử dụng tới là người lập trình phải tự xây dựng tùy theo mục đích của người sử dụng.



4.3.3 Kiểu số nguyên

4.3.3.1. Khái niệm: Kiểu nguyên là một kiểu số hữu hạn và đếm được có miền giá trị phụ thuộc vào số byte được cấp phát.

4.3.3.2 Các kiểu số nguyên

Tên kiểu	Phạm vi giá trị	Số byte	Giải thích
ShortInt	-128 .. 127	1	1 bit chứa dấu, 7 bit chứa giá trị
Byte	0..255	1	8 bit chứa giá trị
Integer	-32768 .. 32767	2	1 bit chứa dấu, 15 bit chứa giá trị
Word	0 .. 65535	2	16 bit chứa giá trị
LongInt	-2147483648 .. 2147483647	4	1 bit chứa dấu, 31 chứa giá trị

Bảng 3.1. các kiểu số nguyên

Ngoài kiểu Integer là thông dụng nhất, các số nguyên còn được chia ra thành 4 kiểu nữa đó là: Byte, Word, ShortInt và LongInt. Bảng 3.1 liệt kê chi tiết về tên gọi, phạm vi giá trị và độ dài tính theo đơn vị byte của từng kiểu nguyên.

Các biến nguyên chỉ có thể nhận các giá trị là các số nguyên nằm trong phạm vi giá trị của biến đó. Khi gán cho một biến một số nguyên nằm ngoài phạm vi của biến thì máy sẽ báo lỗi: "Const out of range".

Ví dụ 4.1: cho khai báo :

```
Var i : Byte;
```

```
      N : Integer;
```

thì các lệnh dưới đây là đúng:

```
i:= 200;
```

```
N:= -1500;
```

còn các lệnh dưới đây là bị lỗi :

```
i:= -5;
```

```
N:= 50000;
```

Đặc biệt không thể gán một số thực cho một biến nguyên. Câu lệnh sau là sai :

```
N:= 12.5 ;
```

Khi gặp tình huống này, máy sẽ báo lỗi "Type mismatch".

Chú ý: Các số nguyên hệ thập lục phân (hệ 16) được biểu diễn bằng cách viết thêm dấu \$ ở trước số, ví dụ ba số dưới đây :

\$A , \$FF và \$10

là các số nguyên viết trong hệ 16. Chúng có giá trị tương ứng trong hệ 10 là:

10 , 255 và 16

4.3.3.3 Các phép toán số học trên số nguyên

Phép cộng và trừ : ký hiệu + và - như thường lệ.

Phép nhân : ký hiệu bằng dấu *, ví dụ $4*2$ cho kết quả là 8.

Phép chia : ký hiệu bằng dấu / , ví dụ $6/4$ cho kết quả là 1.5.

Phép chia lấy phần nguyên : ký hiệu bằng từ khóa DIV.

Phép lấy phần dư nguyên của phép chia: ký hiệu bằng từ khóa MOD.

Ví dụ 4.2: $15 \text{ DIV } 6$ cho kết quả là 2.

$15 \text{ MOD } 6$ cho kết quả là 3.

Các phép toán trên đều cho kết quả là các số nguyên, trừ ra phép chia (/) luôn cho kết quả là một số thực. Vì thế nếu N là một biến nguyên, mà gán : $N := 20/5$;

thì máy sẽ báo lỗi, bởi vì phải có giá trị kiểu thực (=4.0) mặc dù phần lẻ bằng 0.

Nhận xét: số nguyên N là chẵn nếu $N \bmod 2 = 0$ (tức N chia hết cho 2),

ngược lại, là lẻ nếu $N \bmod 2 \neq 0$. (dấu \neq trong Pascal có nghĩa là khác nhau).

Thứ tự thực hiện các phép toán cũng giống như thường lệ:

- Các biểu thức trong (...) được tính trước tiên
- Kế đến là *, /, div, mod
- Sau cùng là +, -
- Đối với các phép toán cùng thứ tự mà đứng liền nhau thì phép toán nào đứng trước được làm trước.

Ví dụ 4.3: tính biểu thức sau :

$$\begin{aligned} & 15 \bmod (2 + 4) * 20 \text{ div } (10 \text{ div } 4) + 40 \bmod (5 * 3) \\ &= 15 \bmod 6 * 20 \text{ div } 2 + 40 \bmod 15 \\ &= 3 * 20 \text{ div } 2 + 10 \\ &= 60 \text{ div } 2 + 10 \\ &= 30 + 10 \\ &= 40 \end{aligned}$$

4.3.4 Kiểu số thực

4.3.4.1 Kiểu Real và các kiểu mở rộng

Kiểu Real là kiểu số thực thông dụng nhất dùng để biểu diễn các số thực x có trị tuyệt đối $|x|$ nằm trong khoảng từ $2.9*10^{-39}$ đến $1.7*10^{+38}$. Nếu $|x| > 1.7*10^{+38}$ thì không biểu diễn x trong máy được, còn nếu $|x| < 2.9*10^{-39}$ thì x được coi là bằng 0.

Có hai cách biểu diễn các số thực:

- Cách 1: Viết bình thường, trong đó dấu phẩy thập phân được thay bằng dấu chấm thập phân, ví dụ như: $45.0 - 256.45 + 122.08$
- Cách 2: Viết số dưới dạng khoa học :

$$1.257\text{E}+01 \text{ (có giá trị } = 1.257*10^1 = 12.57 \text{)}$$

$$1257.0\text{E}-02 \text{ (có giá trị } = 1257*10^{-2} = 12.57 \text{)}$$

Trong dạng này số gồm có hai phần, phần đứng trước E gọi là phần định trị, được viết theo cách 1, phần đứng sau E gọi là phần bậc, gồm dấu cộng hoặc trừ, tiếp đến là một số nguyên.

Số viết theo cách 1 còn gọi là số có dấu chấm thập phân cố định, số viết theo cách 2 còn gọi là số có dấu chấm thập phân di động hay số dạng khoa học (Scientific).

Ví dụ 4.4: Muốn khai báo hai biến x, y kiểu real, ta viết:

Var x, y : Real;

Ngoài kiểu Real ra, các số thực còn có 4 kiểu mở rộng nữa là Single, Double, Extended và Comp. Bảng 3.2 nêu chi tiết về phạm vi giá trị và số byte dùng để lưu trữ trong bộ nhớ của từng kiểu số thực.

Tên kiểu	Phạm vi giá trị	Số byte
Real	$2.9 \cdot 10^{-39} \dots 1.7 \cdot 10^{38}$	6
Single	$1.5 \cdot 10^{-45} \dots 3.4 \cdot 10^{38}$	4
Double	$5.0 \cdot 10^{-324} \dots 1.7 \cdot 10^{308}$	8
Extended	$3.4 \cdot 10^{-4932} \dots 1.1 \cdot 10^{4932}$	10
Comp	$-9.2 \cdot 10^{18} \dots 9.2 \cdot 10^{18}$	8

Bảng 3.2: các kiểu số thực

Chú ý: Turbo Pascal thường chỉ làm việc với một kiểu Real. Muốn dùng 4 kiểu thực còn lại, phải chuyển sang mode 8087 bằng cách viết chỉ thị {\$N+} ở ngay đầu chương trình.

4.3.4.2 Các phép toán trên số thực

Có 4 phép toán số học là nhân (*), chia (/), cộng (+) và trừ (-). Khi một trong các số hạng tham gia tính toán là kiểu thực thì kết quả của phép toán cũng là một số thực. Phép toán DIV, MOD không dùng cho các số thực.

Ví dụ 4.5: với hai biến x, y kiểu thực thì lệnh sau là bị lỗi vì biểu thức về phải không hợp lệ: $y := x \bmod 10$;

Các phép toán so sánh (=, <, >, <=, >=) cũng dùng được cho các số hạng là thực hay nguyên.

4.3.4.3 Các hàm có đối số nguyên hoặc thực

- **Hàm inc(i,h);** là hàm tương đương với câu lệnh gán $i := i + h$, với i, h là các số nguyên và kết quả trả lại có kiểu nguyên.

Nếu h=1 thì có thể viết tắt là inc(i);

- **Hàm ABS(x):** tính trị tuyệt đối của x. Kiểu dữ liệu của kết quả cùng kiểu với đối số. Nếu x nguyên thì ABS(x) cũng nguyên, nếu x là số thực thì ABS(x) cũng là số thực.

Ví dụ 4.6: Abs(5 - 8) = 3

- **Hàm SQR(x):** tính bình phương của x. Kiểu dữ liệu của kết quả cùng kiểu với đối số.

Ví dụ 4.7: Sqr(4.0) = 16.0

Sqr(7 div 3) = 4

- Trong các hàm dưới đây, đối số x có thể là nguyên hay thực, nhưng giá trị trả về luôn luôn là kiểu thực:

Hàm **SQRT(x)**: tính căn bậc 2 của x ($x \geq 0$), kết quả trả lại là một số thực

Hàm **EXP(x)**: tính e mũ x , kết quả trả lại là một số thực

Hàm **LN(x)**: tính $\ln x$, ($x > 0$), kết quả trả lại là một số thực

Các hàm **SIN(x)**, **COS(x)**, và **ARCTAN(x)**: tính $\sin x$, $\cos x$ và $\arctan x$.

Hàm **INT(x)**: cho số thực bằng phần nguyên của x .

Ví dụ 4.8: $\text{Int}(12.55) = 12.0$

$\text{Int}(1+10/3)=4.0$

Hàm **FRAC(x)**: cho số thực bằng phần lẻ của x .

Ví dụ 4.9: $\text{Frac}(12.55) = 0.55$

Hai hàm đặc biệt dưới đây cho kết quả là số nguyên:

Hàm **TRUNC(x)**: cho số nguyên là phần nguyên của x .

Ví dụ 4.10: $\text{Trunc}(12.55) = 12$

$\text{Trunc}(-2.98) = -2$

Hàm **ROUND(x)**: cho số nguyên bằng cách làm tròn x .

Ví dụ 4.11: $\text{Round}(12.45) = 12$

$\text{Round}(-2.98) = -3$

Chú ý rằng hàm $\text{Int}(x)$ và hàm $\text{Trunc}(x)$ cùng cho phần nguyên của x , chúng chỉ khác nhau về kiểu dữ liệu của giá trị trả về. $\text{Int}(4.5) = 4.0$ còn $\text{Trunc}(4.5) = 4$ (viết 4 thì hiểu đó là số nguyên, còn viết 4.0 thì hiểu đó là số thực).

4.3.5 Kiểu ký tự (CHAR)

4.3.5.1 Ký tự và biến kiểu ký tự

Các ký tự dùng trong máy tính điện tử được liệt kê đầy đủ trong bảng mã ASCII (xem phụ lục 1) gồm 256 ký tự khác nhau và được đánh số thứ tự từ 0 đến 255. Số thứ tự của mỗi ký tự còn gọi là mã ASCII của ký tự đó.

Tuy có 256 ký tự khác nhau song chỉ có 128 ký tự đầu tiên là hay dùng, còn lại là các ký tự mở rộng. Các ký tự có mã từ 0 đến 31 gọi là các ký tự điều khiển, không in ra được, được dùng để điều khiển các thiết bị ngoại vi, chẳng hạn ký tự có mã là 7 dùng để tạo một tiếng kêu bip, ký tự có mã là 13 dùng để chuyển con trỏ màn hình xuống đầu dòng dưới...

Mỗi ký tự trong bảng mã ASCII gọi là một hằng ký tự, chiếm độ dài 1 byte, và khi viết trong Pascal phải được đặt trong cặp nháy đơn: '0', '1', 'A', 'B', '\$', ...

Giữa các ký tự, có một thứ tự mặc nhiên theo nguyên tắc: ký tự có mã nhỏ hơn thì nhỏ hơn. Tức là:

Ký tự trắng < '0' < '1' < ... < '9' < 'A' < 'B' < ... < 'Z' < 'a' < 'b' < ... < 'z'

Biến nhận giá trị là các hằng ký tự gọi là biến kiểu ký tự, chúng được khai báo nhờ từ khóa CHAR, chẳng hạn như khai báo hai biến ch và $ch1$ dưới đây:

Var $ch, ch1$: Char;

Khi đó có thể gán:

ch:='A';

ch1:='\$';

Ký tự 'A' gọi là giá trị của biến ch, còn '\$' là giá trị của biến ch1.

Nhận xét: Từ bảng mã của các chữ cái ta suy ra:

Mã chữ thường = Mã chữ hoa tương ứng + 32. (1)

4.3.5.2 Các hàm liên quan đến ký tự

- Hàm **PRED(ch)**: cho ký tự đứng ngay trước ký tự ch trong bảng mã.

Ví dụ 4.12: Pred('B')='A'

- Hàm **SUCC(ch)**: cho ký tự đứng ngay sau ký tự ch trong bảng mã.

Ví dụ 4.13: Succ('A')='B'.

- Hàm **UpCase(ch)**: đổi ký tự ch thành chữ hoa tương ứng.

Ví dụ 4.14: Upcase('a')='A', Upcase('b')='B', Upcase('A')='A'.

- Hàm **ORD(ch)**: cho mã của ký tự ch.

Ví dụ 4.15: Ord('A')=65, Ord('a')=97.

- Hàm **CHR(k)**: đổi số k nguyên, $0 \leq k \leq 255$, cho ký tự có mã bằng k.

Ví dụ 4.16 Chr(65)='A',

Chr(97)='a',

Chr(32) là ký tự trắng.

Có một số ký tự không có trên bàn phím, để viết chúng lên màn hình ta phải dùng lệnh Write và hàm CHR.

Ví dụ 4.17 Lệnh `Writeln(Chr(201));` in ra ký tự: ¶

Lệnh `Writeln(Chr(187));` in ra ký tự: ¶

Ký tự có mã là 7 gọi là ký tự BEL (chuông), và lệnh:

`Write(Chr(7));` hay `Write(#7);` {phát ra một tiếng kêu bip}

Chú ý: Turbo Pascal (TP) cho phép viết gọn Chr(k) thành #k nếu k là hằng số. Ví dụ, hai lệnh sau cùng in lên màn hình chữ A:

`Write(#65);`

`Write(Chr(65));`

Trong TP không có hàm đổi chữ hoa ra chữ thường, nhưng có thể làm việc này nhờ công thức (1) và hai hàm Ord và Chr:

Chữ thường := Chr (Ord(chữ hoa) + 32)

Tuy nhiên trước khi thực hiện thì cần phải kiểm tra xem ký tự cần chuyển có phải chữ hoa hay không. Nếu là chữ hoa thì mới thực hiện đổi, ngược lại thì không nên đổi vì sẽ cho ra kết quả không đúng.

4.3.6 Kiểu LÔGIC (BOOLEAN)

Kiểu Boolean chỉ có hai giá trị là TRUE (đúng) và FALSE (sai), không phân biệt chữ hoa hay chữ thường. Về quan hệ thứ tự thì $FALSE < TRUE$. Mỗi giá trị boolean chiếm một byte bộ nhớ.

Các phép toán logic gồm có: NOT, AND, OR và XOR. Nếu A và B là hai đại lượng logic thì NOT A, A and B, A or B và A xor B cũng là những đại lượng logic có kết quả được cho ở bảng dưới đây:

A	not A
True	False
False	True

A	B	A and B	A or B	A xor B
True	True	True	True	False
True	False	False	True	True
False	True	False	True	True
False	False	False	False	False

Bảng 3.5: Phép toán logic

Cũng từ bảng này ta rút ra các nhận xét :

- A and B là đúng khi và chỉ khi A và B đồng thời đúng. (Do đó chỉ cần một trong hai biến A hoặc B sai thì A and B sẽ sai).
- A or B là sai khi và chỉ khi A và B đồng thời sai. (Do đó chỉ cần một trong hai biến A hoặc B đúng thì A or B sẽ đúng).
- A xor B là đúng khi và chỉ khi A khác B.

Thứ tự thực hiện các phép toán logic là như sau: NOT tính trước, kế đến AND, sau cùng là OR, XOR.

Ví dụ 4.18: sau khi thực hiện lệnh:

$A = \text{Not } (2*3=5) \text{ or } ('A' < 'B') \text{ and not } (4/2=2) \text{ xor } (\text{Sqrt}(2) > 1);$

thì giá trị của A= FALSE, thật vậy :

$$\begin{aligned}
 & \text{Not } (2*3=5) \text{ or } ('A' < 'B') \text{ and not } (4/2=2) \text{ xor } (\text{Sqrt}(2) > 1) \\
 &= \text{TRUE or TRUE and FALSE xor TRUE} \\
 &= \text{TRUE or FALSE xor TRUE} \\
 &= \text{TRUE xor TRUE} \\
 &= \text{FALSE}
 \end{aligned}$$

Biến chỉ nhận giá trị là TRUE hoặc FALSE gọi là biến kiểu logic. Khi khai báo biến kiểu logic ta dùng từ khóa Boolean.

Ví dụ 4.19:


```
Var      A, B : Boolean;
```

Trong chương trình ta có thể gán :

```
A:= true;
```

```
B:=2*2 < 3;
```

Giá trị của biến B sẽ là False vì biểu thức $2*2 < 3$ là sai.

Về thứ tự tính toán, các phép so sánh thì ngang cấp nhau và được tính sau tất cả các phép toán khác.

Ví dụ 4.20: tính biểu thức :

```
5+7 div 2 < -7 mod 3 + 5*2 =  
= 5 + 3 < -1 + 10  
= 8 < 9  
= TRUE
```

Do đó, khi trong một biểu thức mà có các phép toán logic xen kẽ với các biểu thức so sánh thì các biểu thức so sánh phải để trong ngoặc đơn.

Chẳng hạn, biểu thức sau là sai quy cách:

```
N > 0 and N < 10
```

Cần sửa đúng thành : $(N > 0) \text{ and } (N < 10)$

4.3.7. Một số kiểu dữ liệu đơn giản do người lập trình định nghĩa

Ngoài các kiểu dữ liệu chuẩn thì Turbo pascal chấp nhận các kiểu dữ liệu khác do người dùng lập trình định nghĩa. Việc khai báo kiểu dữ liệu cho phép chúng ta xây dựng các kiểu dữ liệu mới.

Các kiểu dữ liệu mới được khai báo trong phần khai báo **TYPE** với cú pháp như sau:

```
TYPE <tên kiểu dữ liệu mới> = <mô tả kiểu dữ liệu mới>;
```

Ví dụ 4.21:

```
TYPE kieu_nguyen = Integer;
```

```
KieuSV = Record  
    masv: int;  
    ht: string[30];  
    diem:float;  
End;
```

```
Var    a: kieu_nguyen;  
        c: integer;  
        sv: KieuSV;
```

4.3.7.1 Kiểu liệt kê (enumerated type)

* Cách khai báo

Ngoài các kiểu dữ liệu đã có sẵn như kiểu nguyên, thực, ký tự, logic và kiểu chuỗi, Turbo Pascal còn cho phép người sử dụng có thể tự xây dựng các kiểu dữ liệu mới.

Kiểu liệt kê được định nghĩa bằng cách sử dụng từ khóa **TYPE** và liệt kê ra tất cả các giá trị của kiểu, theo mẫu sau:

```
Type Tênkiểu = (tên1, tên2, ..., tênN) ;
```

trong đó tên1, tên2,..., tênN là các tên tự đặt theo đúng quy ước về đặt tên.

Ví dụ 4.22 :

```
Type    Phai=(nam, nu) ;  
        Ten_mau = (den, trang, xanh, vang, tim, nau);
```

Theo khai báo này thì *Phai* là một kiểu dữ liệu liệt kê chỉ có hai giá trị là nam và nu, *Ten_mau* cũng là kiểu dữ liệu liệt kê và có sáu giá trị là : den, trang, xanh, vang, tim, nau.

Khi một kiểu liệt kê đã được định nghĩa thì có thể khai báo các biến thuộc kiểu liệt kê này bằng từ khóa Var.

Ví dụ 4.23: Var Ph1, Ph2 : Phai;
 M1, M2 : Ten_mau ;

Trong chương trình, ta có thể gán :

```
Ph1:=nam;  
Ph2:=nu;  
M1:=den;  
M2:=trang;
```

Pascal còn cho phép khai báo trực tiếp biến kiểu liệt kê không cần qua giai đoạn định nghĩa Type bằng cách liệt kê các giá trị mà biến có thể nhận.

Ví dụ 4.24: các biến Ph1, Ph2, M1, M2 nói trên có thể khai báo trực tiếp như sau:

```
Var Ph1, Ph2 : (nam, nu) ;  
    M1, M2 : ( den, trang, xanh, vang, tim, nau);
```

*** Các hàm liên quan đến kiểu liệt kê**

- **Hàm ORD(tên):** Trả về số thứ tự của tên trong kiểu liệt kê. Các giá trị liệt kê được đánh số thứ tự bắt đầu từ 0. Ví dụ:

```
Ord(nam)=0,  
Ord(xanh)=2
```

Thông qua hàm Ord, các giá trị liệt kê có thể so sánh với nhau theo quy tắc: giá trị nào có số thứ tự nhỏ hơn thì nhỏ hơn:

```
den < trang < xanh < vang < tim < nau
```

- **Hàm PRED(tên) và hàm SUCC(tên):** trả về giá trị đứng ngay trước và ngay sau tên trong kiểu liệt kê tương ứng.

Ví dụ 4.25: Pred(nu)=nam

```
Pred(nau)=tim  
Succ(den)=trang
```

- **Hàm Tênkiểu(k):** trả về giá trị liệt kê có số thứ tự là k trong Tênkiểu, ví dụ:

```
Phai(0)=nam  
Ten_mau(2)= xanh
```

Hàm này là hàm ngược của hàm Ord.

*** Nhập , xuất kiểu liệt kê**

Các giá trị liệt kê không thể nhập, xuất trực tiếp bằng lệnh Readln và Write được. Đây là hạn chế của kiểu liệt kê, khiến nó không thông dụng.

Khi muốn nhập hay xuất kiểu liệt kê, ta có thể dùng một biến trung gian St kiểu chuỗi. Chẳng hạn, muốn nhập màu xanh cho biến M1, ta dùng hai lệnh:

```
Readln(St);  
If St='xanh' then M1:=xanh;  
Tương tự, muốn in màu xanh lên màn hình, ta dùng lệnh :  
If M1= xanh then Writeln('xanh');
```

4.3.7.2 Kiểu đoạn con (Subrange type)

Kiểu đoạn con được mô tả bằng cách chỉ ra phạm vi giá trị mà các biến thuộc kiểu đó có thể nhận :

TYPE Tênkiểu = hằng1..hằng2;

VAR Tênbiến : Tênkiểu;

hoặc khai báo trực tiếp :

VAR Tênbiến : hằng1..hằng2;

Trong đó, hằng1 < hằng2 là hai hằng thuộc cùng một kiểu dữ liệu. Kiểu dữ liệu của hằng1 và hằng2 chỉ có thể là kiểu nguyên, ký tự, lôgic, hay liệt kê

Ví dụ 4.26: Type Chu_Hoa = 'A' .. 'Z' ;

Tuoi= 0..200;

Var Ch : Chu_hoa;

T: Tuoi;

Theo khai báo này thì ch là một biến kiểu đoạn con, có thể nhận các giá trị là các ký tự từ 'A' đến 'Z', tương tự, biến T có thể nhận các giá trị là các số nguyên từ 0 đến 200.

Cũng có thể khai báo hai biến Ch và T trực tiếp theo cách sau:

Var Ch : 'A' .. 'Z' ;

T : 0..200;

Trong nhiều trường hợp, việc khai báo đoạn con có tác dụng tiết kiệm bộ nhớ. Tùy theo phạm vi hằng1..hằng2 mà Turbo Pascal sẽ cấp phát cho biến một số byte tối thiểu. Trong ví dụ trên, mỗi biến Ch hay T sẽ được chứa trong 1 byte.

Kiểu đoạn con còn cho phép kiểm soát được giá trị của biến có vượt ra ngoài phạm vi của nó hay không. Ví dụ, nếu đối với biến T mà gán: T:=201; thì máy sẽ báo lỗi "const out of range". Ngoài ra khi chạy chương trình trong mode {\$R+}, chương trình sẽ dừng ngay nếu biến nhận giá trị vượt khỏi phạm vi.

Kiểu liệt kê và kiểu đoạn con thuộc loại đơn giản và đếm được.

4.4. Hằng, biến và biểu thức

4.4.1 Khái niệm về biến và hằng

Trong phần trước ta đã biết mỗi kiểu dữ liệu có một tập các giá trị tương ứng. Các giá trị của kiểu nguyên hay kiểu thực là các số, như 40 hay 5.72, các giá trị của kiểu ký tự là các ký tự như 'A' hay 'a', còn kiểu logic thì chỉ có hai giá trị là True và False, ...

Quá trình xử lý trong máy tính đòi hỏi mỗi giá trị phải được lưu trữ ở một ô nhớ nào đó trong bộ nhớ của máy, và ô nhớ này được đặt một cái tên để gọi. Khi đó mọi việc tính toán hay xử lý liên quan đến mỗi giá trị được thực hiện gián tiếp thông qua tên của ô nhớ chứa giá trị đó. Ví dụ, nếu số 5.72 được lưu trong ô nhớ có tên là x, thì biểu thức $5.72 * 2$ có thể được viết là $x * 2$. Việc dùng tên x để nhớ và tiện hơn nhiều so với việc dùng và nhớ số 5.72.

Như vậy, khi một ô nhớ được đặt tên thì tên này đồng nhất với giá trị của nó. Trong một chương trình, mỗi ô nhớ có một tên duy nhất nhưng giá trị của nó thì có thể thay đổi hoặc không. Nếu giá trị của ô nhớ có thể thay đổi được thì ô nhớ này là một biến, tên của ô nhớ là tên biến, ngược lại, nếu giá trị của ô nhớ không thể thay đổi, thì ô nhớ là một hằng, tên của ô nhớ là tên hằng.

Các biến và hằng tham gia trong chương trình đều phải được khai báo. Việc khai báo có tác dụng báo trước cho máy dành sẵn các ô nhớ thích hợp trong bộ nhớ để sẵn sàng chứa dữ liệu.

4.4.2 Khai báo biến

+ Khái niệm : Biến là đại lượng có giá trị và giá trị này có thể thay đổi được trong chương trình.

+ Khai báo :

Var Danh sách tên biến : Tên Kiểu Dữ liệu ;

Trong đó : Tên biến là do người lập trình đặt theo đúng quy tắc của một tên.

Ví dụ 4.27: Var i, j : Integer;
 x, y : Real;

Theo khai báo trên, ta có hai biến i và j cùng kiểu số nguyên (Integer), và hai biến x, y cùng kiểu số thực (Real).

4.4.3 Khai báo hằng

+ Định nghĩa : Hằng là một đại lượng có giá trị không đổi trong chương trình.

+ Việc khai báo hằng bằng tên có 3 ưu điểm:

- Chương trình dễ đọc
- Dễ thay đổi chỉnh sửa
- Tiết kiệm bộ nhớ

+ Khai báo :

Const Tên_hằng = giá trị ;

Trong đó: Tên hằng là do người lập trình đặt theo đúng quy tắc của một tên.

Ví dụ 4.28: const N = 10;
 SoPi = 3.1416;

Turbo Pascal có sẵn một số hằng chuẩn cho phép sử dụng mà không phải khai báo, như : Pi, MaxInt . Hằng Pi có giá trị bằng số p , còn MaxInt = 32767, là số Integer lớn nhất. Chẳng hạn, có thể dùng các lệnh sau:

```
Writeln('Dien tich hinh tron ban kinh 5 la: ', Pi*5*5:8:3);
Writeln('So Integer lon nhat = ', MaxInt);
```

4.4.4 Biểu thức

Biểu thức là một công thức gồm có một hay nhiều thành phần được kết nối với nhau bởi các phép toán. Mỗi thành phần (hay toán hạng) có thể là hằng, là biến hay là hàm. Khi các phép toán trong biểu thức được thực hiện thì ta nhận được một giá trị gọi là kết quả của biểu thức. Kiểu dữ liệu của kết quả gọi là kiểu dữ liệu của biểu thức.

Ví dụ 4.29: $3 * 5 \text{ div } 2 + 7 \text{ mod } 4$ là biểu thức nguyên, có kết quả là 10

$2 + \sin(\pi/2)$ là biểu thức thực, có kết quả là 3.0

$\text{Chr}(\text{ord}('a') - 32)$ là biểu thức ký tự, có kết quả là 'A'

$(4+2=6) \text{ and } ('B' <> 'b')$ là biểu thức logic, có kết quả là True

$'AB' + 'CD'$ là biểu thức chuỗi, có kết quả là 'ABCD'

Các thành phần trong biểu thức cần phải có kiểu dữ liệu phù hợp để cho các phép toán thực hiện được, nếu không máy sẽ báo lỗi. Ví dụ, biểu thức sau :

$5 + 'A'$ là sai vì ta không thể cộng một số nguyên với một ký tự.

Một biểu thức có thể chứa nhiều phép toán. Thứ tự thực hiện các phép toán được cho trong bảng dưới đây.

Cấp ưu tiên	Phép toán
1	biểu thức trong ngoặc đơn (...)
2	Các hàm
3	NOT, - (phép lấy dấu âm)
4	*, /, DIV, MOD, AND
5	Shl, Shr
6	+, - (trừ), OR, XOR
7	=, <>, <, <=, >, >=, IN

Bảng 3.6: Bảng ưu tiên các phép toán

Việc tính toán một biểu thức dựa theo hai quy tắc :

Quy tắc 1:

Phép toán có cấp ưu tiên nhỏ thì được tính trước, phép toán có cấp ưu tiên lớn thì được tính sau.

Quy tắc 2:

Đối với các phép toán đứng liền nhau và có cùng cấp ưu tiên, thì cái nào đứng trước được tính trước.

Ví dụ 4.30:

+ *Tính biểu thức số học :*

$$\begin{aligned} & (4+5)*2 \operatorname{div} 7 + \sin(\pi/6) \\ = & 9 * 2 \operatorname{div} 7 + 0.5 \\ = & 18 \operatorname{div} 7 + 0.5 \\ = & 2 + 0.5 \\ = & 2.5 \end{aligned}$$

+ *Tính biểu thức logic :*

$$\begin{aligned} & (2 > 4 \operatorname{div} 2) \text{ or Not } (49.25 + 2 < 50) \\ = & (2 > 2) \text{ or Not } (51.25 < 50) \\ = & \text{FALSE or Not FALSE} \\ = & \text{FALSE or TRUE} \\ = & \text{TRUE} \end{aligned}$$

Chương 5

BUƯỚC ĐẦU XÂY DỰNG CHƯƠNG TRÌNH

5.1. Cấu trúc chung một chương trình Pascal

5.1.1 Chương trình Pascal

Chương trình là một dãy các câu lệnh chỉ thị cho máy các công việc phải thực hiện. Một chương trình Pascal đầy đủ gồm ba phần chính:

Phần tiêu đề

Phần khai báo

Phần thân chương trình chính

```
Program Tên_tự_đặt ; { Phần tiêu đề}

{ Phần khai báo}
Uses ... {khai báo sử dụng thư viện chuẩn}
Label ... {khai báo nhãn}
Const ... {khai báo hằng}
Type ... {khai báo kiểu dữ liệu}
Var ... { khai báo biến}
Function ... { khai báo hàm}
Procedure ... {khai báo thủ tục }

{ Phần thân chương trình chính}
Begin
{ Các lệnh }
End.
```

Hình 4.1: Cấu trúc của chương trình Pascal

5.1.2. Phần tiêu đề chương trình

Phần này bắt đầu bằng từ khóa Program, sau đó ít nhất là một khoảng trắng và một tên do người dùng tự đặt, cuối cùng kết thúc bằng dấu chấm phẩy ‘;’.

Cú pháp: **Program Tên_tự_đặt;**

Ví dụ 5.1: Program Btap1;

 hoặc: Program Giai_pt_bac2;

Phần tiêu đề còn gọi là phần đầu của chương trình, nó có thể không có cũng được.

Tên_tự_đặt phải tuân theo quy tắc đặt tên.

5.1.3. Phần khai báo

Phần khai báo có nhiệm vụ giới thiệu và mô tả các đối tượng, các đại lượng sẽ tham gia trong chương trình. Nó gồm có:

- **Khai báo sử dụng thư viện chuẩn:**

Turbo Pascal có sẵn một số hàm và thủ tục chuẩn, chúng được phân thành từng nhóm theo chức năng mang các tên đặc trưng, gọi là các thư viện hay đơn vị chương trình (Unit), như : Crt, Graph, Dos, Printer, .v.v. . Muốn sử dụng các hàm hay thủ tục của thư viện nào, ta phải khai báo có sử dụng thư viện đó, lời khai báo phải để ở ngay sau phần tiêu đề của chương trình theo cú pháp :

Uses *danh_sách_các_thư_viện_chuẩn*;

Trong đó các thư viện cách nhau dấu phẩy (,).

Ví dụ 5.2: do thủ tục Clrscr nằm trong thư viện CRT, nên nếu trong chương trình mà có dùng lệnh Clrscr, thì phải khai báo :

Uses CRT ;

Muốn sử dụng cả hai thư viện CRT và GRAPH, ta khai báo :

Uses CRT, GRAPH ;

- **Khai báo nhãn:**

Label *Danh_sách_các_tên_nhãn*;

Khi sử dụng trong chương trình thì sử dụng theo cú pháp:

***Tên_nhãn*: <các câu lệnh>;**

Các nhãn thường đi với câu lệnh **goto *tên_nhãn*;**

- **Khai báo hằng** theo cú pháp:

Const *Tên_hằng* = *giá_trị*;

- **Khai báo kiểu dữ liệu mới:**

Ngoài các kiểu dữ liệu mà bản thân ngôn ngữ đã có sẵn như kiểu thực, kiểu nguyên, kiểu ký tự, kiểu logic, .v.v. người dùng có thể tự xây dựng các kiểu dữ liệu mới phục vụ cho chương trình của mình, nhưng phải mô tả sau từ khóa TYPE. Khi đã định nghĩa một kiểu dữ liệu mới, ta có thể khai báo các biến thuộc kiểu dữ liệu này theo cú pháp sau:

Type *Tên_kiểu_dữ_liệu_mới* = *mô_tả_cho_kiểu_dữ_liệu_mới*

Ví dụ 5.3: ta định nghĩa một kiểu dữ liệu mới có tên là KieuMang :

Type KieuMang = Array[1..10] of Real;

Bây giờ có thể khai báo hai biến A và B có kiểu dữ liệu là kiểu Mang :

Var A, B : KieuMang ;

- **Khai báo biến** theo cú pháp:

Var *Danh_sách_các_biến_kiểu_dữ_liệu_cho_biến*;

Trong đó các biến cách nhau bởi dấu phẩy (,).

Ví dụ 5.3: Const N=10 ;
 Var x, y : Real ;
 i, k : Integer;

- *Khai báo các thủ tục và hàm:* được dùng khi có nhu cầu thiết kế các chương trình lớn, phức tạp. Đối với các bài toán nhỏ, đơn giản, việc sử dụng chương trình con là chưa cần thiết. Chi tiết về phần này sẽ được trình bày kỹ trong chương 8.

5.1.4. Phần thân chương trình

Đây là phần chủ yếu nhất của một chương trình, bắt buộc phải có.

Thân chương trình bắt đầu bằng từ khóa **BEGIN** và kết thúc bằng **END**. (có dấu chấm ở cuối). Giữa BEGIN và END là các lệnh. Mỗi lệnh phải kết thúc bằng dấu chấm phẩy ';'. Một lệnh, nếu dài, thì có thể viết trên hai hay nhiều dòng, ví dụ:

Writeln(' Phương trình có hai nghiệm là X1= ', X1:8:2, ' và X2= ', X2:8:2) ;

Ngược lại, một dòng có thể viết nhiều lệnh miễn là có dấu ';' để phân cách các lệnh đó, chẳng hạn :

Write(' Nhập A, B, C: ') ; Readln(A,B,C) ;

Thông thường mỗi dòng chỉ nên viết một lệnh để dễ đọc, dễ kiểm tra lỗi.

Ví dụ 5.4: 1 chương trình đơn giản bằng ngôn ngữ Pascal, để nhập vào một số có hai chữ số và in ra số hàng chục và hàng đơn vị của nó.

Tiêu đề	{	PROGRAM IN_CHU_SO;
Khai báo		VAR n:integer; { khai bao cac bien}
Thân chương trình	{	BEGIN
		Write('Nhập số n = ');
		Readln(n);
		If (10<= n) and (n<=99) then
		Begin
		writeln('hàng chục = ',n div 10);
		writeln(' hàng đơn vị = ',n mod 10);
		end
		Else
		Writeln(' Ko phải là số có 2 chữ số');
		readln;
		END.

5.2. Câu lệnh trong chương trình Pascal

5.2.1 Phân loại câu lệnh

Câu lệnh là một dãy các ký tự cơ bản được xây dựng theo một quy tắc nhất định (gọi là cú pháp) nhằm chỉ thị cho máy thực hiện một công việc xác định. Các câu lệnh được chia ra hai loại: câu lệnh đơn giản và câu lệnh có cấu trúc.

Lệnh gán và lời gọi thủ tục được xếp vào loại đơn giản.

Ví dụ 5.5:

```
k := 20;  
Clrscr ;  
Writeln(k) ;
```

Các lệnh rẽ nhánh và lệnh lặp được xếp vào loại có cấu trúc, chúng được xây dựng từ các lệnh đơn giản.

Ví dụ 5.6:

```
If k>=0 then Writeln(k)  
else  
    Writeln( -k) ;
```

Hai hay nhiều lệnh đơn giản được gom lại và đặt giữa hai từ khóa BEGIN và END tạo thành một câu lệnh ghép, câu lệnh ghép cũng là lệnh có cấu trúc.

Ví dụ 5.7:

```
Begin  
    Write(' nhập k :');  
    Readln(k) ;  
End;
```

Từ các lệnh đơn giản và các lệnh có cấu trúc đã có lại có thể xây dựng được các lệnh có cấu trúc phức tạp hơn.

Ví dụ 5.8:

```
If k>= 0 then Writeln(k)  
else  
    Begin  
        Writeln(' k âm, xin nhập lại : ');  
        Readln(k) ;  
    End;
```

Sau đây sẽ trình bày kỹ về một lệnh đơn giản và thông dụng : lệnh gán.

5.2.2. Lệnh gán

+ Lệnh gán có cú pháp như sau :

Tên_Biến := Biểu_thức ;

+ Ý nghĩa : tính toán biểu thức bên phải, rồi lưu kết quả tính được vào biến ở vế trái.

- Biểu thức bên vế phải có thể là một biến, một hằng, một hàm, hay một biểu thức,

Ví dụ 5.9: cho khai báo :

```
Var    A, B : Real;  
        K : Integer;
```

Khi dùng lệnh các lệnh:

```
K := 10 ;  
B := K* 3+5.5;
```

thì biến K có giá trị là 10, biến B có giá trị là 35.5.

Nếu thực hiện tiếp lệnh gán :

```
B:= 17/2;
```

thì giá trị của B bây giờ sẽ là 8.5.

→ Như vậy nếu một biến được gán nhiều lần thì nó sẽ lấy giá trị của lần gán sau cùng, tính đến thời điểm đang xét.

Đặc biệt, lệnh: `B:=B +1;`

có tác dụng tăng giá trị của biến B lên 1 đơn vị, kết quả là B có giá trị bằng 9.5.

Cách thực hiện lệnh `B:=B+1` là như sau: lấy giá trị hiện thời của biến B (là 8.5) cộng thêm 1 (được 9.5), rồi đem kết quả gán cho chính biến B.

Tương tự, lệnh `B:=B-1;` có tác dụng giảm B đi 1 đơn vị.

Yêu cầu để cho lệnh gán thực hiện được là kiểu dữ liệu của biểu thức ở vế phải phải phù hợp với kiểu dữ liệu của biến ở vế trái, nếu không phù hợp thì khi dịch (Compile) chương trình, Turbo Pascal sẽ thông báo lỗi : "Error 26 : Type mismatch". Ví dụ, lệnh gán dưới đây là sai vì vế trái là kiểu thực còn vế phải là kiểu chuỗi :

```
A:='Pascal';
```

Chú ý rằng một số nguyên có thể gán cho một biến thực, (chẳng hạn lệnh `A:=10;` là đúng), nhưng một số thực không thể gán cho một biến nguyên. Ví dụ lệnh `K:=10/4;` là sai vì biến K có kiểu nguyên, còn vế phải cho kết quả là một số thực (=2.5).

Xét thêm ví dụ về các kiểu dữ liệu khác :

Ví dụ 5.10: Cho khai báo

```
Var    Ch : Char ;  
        St: String[20];
```

Khi đó:

Lệnh `St:='A';` là đúng.

Lệnh `St:='100';` là đúng.

Lệnh `Ch:='ABCD';` là sai vì vế phải là một chuỗi.

Lệnh `St:= 100;` là sai vì vế phải là một số.

Lệnh `Ch:='1' ;` là đúng.

Lệnh `Ch:=St ;` là sai vì vế phải là một chuỗi.

Lệnh `St:=Ch;` là đúng

5.3. Các lệnh nhập, xuất dữ liệu

5.3.1 Lệnh xuất (in) dữ liệu ra màn hình

5.3.1.1 Ý nghĩa

Để in thông báo, lời mời nhập dữ liệu, kết quả của chương trình... ta dùng các lệnh xuất (lệnh in) dữ liệu. Có nhiều thiết bị để xuất thông tin ra, song trong phần này chúng ta chỉ nghiên cứu các lệnh đưa dữ liệu ra màn hình - thiết bị xuất chuẩn thông tin.

5.3.1.2 Cú pháp

WRITE (mục1, mục2, mục3,...,mụcn); (1)

WRITELN (mục1, mục2, mục3,...,mụcn); (2)

WRITELN; (3)

5.3.1.3 Sự thực hiện

Dạng (1): In lên màn hình các mục ($i=1,n$) tại vị trí hiện thời của con trỏ màn hình theo thứ tự liệt kê trong lệnh, từ trái sang phải. Sau đó, con trỏ màn hình đứng ở vị trí sau mục n .

Dạng (2): In lên màn hình các mục ($i=1,n$) tại vị trí hiện thời của con trỏ màn hình theo thứ tự liệt kê trong lệnh, từ trái sang phải. Sau đó, con trỏ màn hình được đưa xuống đầu dòng tiếp theo trên màn hình

Dạng (3): Đưa con trỏ màn hình ở vị trí hiện thời về đầu dòng tiếp theo.

5.3.1.4 Chú ý

- Các mục: mục1, mục2, mục3,...,mụcn có thể là:
 - Xâu ký tự
 - Biến
 - Biểu thức
 - Hàm
 - Hằng
 - Các giá trị thuộc kiểu dữ liệu đơn giản chuẩn
 - Các giá trị thuộc kiểu dữ liệu có cấu trúc, như: kiểu mảng, kiểu xâu
- Các mục có thể in ra ở dạng không qui cách hoặc có qui cách.
- Các mục phải đặt cách nhau dấu phẩy (,)

5.3.1.5 Các ví dụ

Ví dụ 5.11: Giả sử có các khai báo hằng, biến sau:

```
CONST kq='Ket qua cua chuong trinh';  
so=189.64;  
VAR a, b: Integer;  
x: Real;
```

```
MT: Array[1..10, 1..10] of Real;
```

- Có các phép gán:

```
a:=5;
```

```
b:=17;
```

```
x:=0.061;
```

- Có thể viết một số lệnh in ra màn hình như sau:

```
WRITELN('Truong Cao dang Dien luc');
```

```
WRITELN(a);
```

```
WRITELN(x);
```

```
WRITELN(a*x+b);
```

```
WRITELN(SQRT(2*a*x));
```

```
WRITELN(so);
```

```
WRITELN(kq);
```

```
WRITELN(PI);
```

```
WRITELN(1999);
```

```
WRITELN(MT[3,2]);
```

```
WRITELN('So be nhat ',a,' so lon nhat ',b);
```

...

5.3.1.6 In dữ liệu ra màn hình có qui cách

a) Đối với số nguyên

- In không qui cách:

Ví dụ 5.12: Xét đoạn chương trình sau:

```
a:=5;
```

```
b:=164;
```

```
WRITELN(a);
```

```
WRITELN(b);
```

```
WRITELN(MAXINT);
```

- Kết quả thu được trên màn hình:

```
5
```

```
164
```

```
32767
```

- Nhận xét:

Các số nguyên được in mỗi số trên một dòng nhờ câu lệnh WRITELN. Các số nguyên in không qui cách sẽ căn theo lề bên trái, vì vậy sẽ là in chưa hợp lý nếu cần bố trí in dữ liệu theo dạng cột.

- In có qui cách:

```
write(I:m); hoặc writeln(I:m);
```

Trong đó:

- I là một biến kiểu số nguyên
- m là số vị trí trên màn hình dành để in số nguyên I, m là một giá trị nguyên.

Ví dụ 5.13: Xét đoạn chương trình sau:

```
a:=5;  
b:=164;  
WRITELN(a:6);  
WRITELN(b:6);  
WRITELN(MAXINT:6);
```

- Kết quả thu được trên màn hình:

```
      5  
     164  
    32767
```

- Nhận xét:

Các số nguyên in có qui cách sẽ căn theo lề bên phải, căn cứ vào số chỗ dành để in chúng đã được chỉ ra trong câu lệnh. Nếu thừa chỗ sẽ để trống phía trước (bên trái) số nguyên được in ra. Vì vậy muốn in dữ liệu kiểu nguyên theo dạng cột thì phải in chúng với cùng một qui cách.

b) Đối với số thực

- In không qui cách:

Ví dụ 5.14: Xét đoạn trình sau:

```
x:=93.61;  
y:=-0.0011;  
WRITELN(x);  
WRITELN(y);  
WRITELN(PI);
```

- Kết quả thu được trên màn hình:

```
9.36100000000E+01  
-1.10000000000E-03  
3.1415926536E+00
```

- Nhận xét:

Các số thực được in ra theo kiểu dấu phẩy động (còn gọi là ký pháp khoa học) theo qui định sau: mỗi số in trong phạm vi 14 vị trí trên màn hình: 1 vị trí dành để in dấu - nếu số thực đó là số âm, nếu không sẽ bỏ trống; tiếp theo là 1 vị trí dành để in phần nguyên của số thực; 10 vị trí tiếp theo sẽ dành để in phần thập phân, nếu không có sẽ in số 0; chữ E được in ở vị trí tiếp theo, sau đó là 1 vị trí cho dấu của phần mũ và 2 vị trí cuối cùng dành để in phần mũ.

Số thực được in như vậy sẽ làm chúng ta khó xem kết quả của chương trình nếu chưa quen.

• In có qui cách:

write(R:m:n);

hay ***writeln(R:m:n)***

Trong đó:

- R là biến kiểu số thực
- m là tổng số vị trí trên màn hình dành để in R
- n là số vị trí dành để in phần thập phân của R hay là số chữ số sau dấu thập phân.
- m, n là các giá trị nguyên; $m > n$, nếu bạn để $m \leq n$ thì việc viết theo quy cách là không có ý nghĩa.

Ví dụ 5.15: Xét đoạn chương trình sau:

`x:=93.61;`

`y:=-0.0011;`

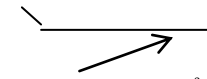
`WRITELN(x:10:4);`

`WRITELN(y:10:4);`


`WRITELN(PI:10:4);`

- Kết quả thu được trên màn hình:

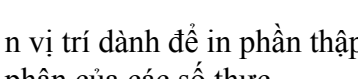
9 3 . 6 1 0 0	
- 0 . 0 0 1 1	
3 . 1 4 1 6	



m vị trí dành để
in các số thực



1 vị trí dành để
in dấu chấm
thập phân



n vị trí dành để in phần thập
phân của các số thực

- Nhận xét:

Các số thực được in ra ở dạng dấu phẩy tĩnh (còn gọi là ký pháp thông thường), so đều phải theo đúng qui cách đã chỉ ra trong câu lệnh. Phần thập phân nếu không sử dụng hết n vị trí sẽ được điền vào đó các số 0; nếu n vị trí không đủ để in hết phần thập phân của số thì phần thập phân sẽ được làm tròn.

Nếu m vị trí không được sử dụng hết thì sẽ để trống phía bên trái số thực được in ra.

Vậy, in các số thực có qui cách sẽ làm cho kết quả trở nên dễ đọc hơn và dữ liệu sẽ được in thành cột nếu mọi số thực đều in với cùng một qui cách.

- **Chú ý:**

Nếu in số thực theo qui cách của số nguyên (R:m) thì kết quả vẫn in ở dạng dấu phẩy động nhưng gọn hơn. Lưu ý rằng trong trường hợp này m phải được chọn ≥ 8 , nếu chọn $m < 8$ thì số thực in ra vẫn chiếm 8 vị trí trên màn hình.

Ví dụ 5.16:

- Xét đoạn trình sau:
 `x:=93.61;`
 `y:=-0.0011;`
 `WRITELN(x:10);`
 `WRITELN(y:10);`
- Kết quả thu được trên màn hình:

```
      9 . 3 6 1 E + 0 2  
-    1 . 1 0 0 E - 0 3
```

c) Đối với dữ liệu là ký tự, chuỗi ký tự, logic

- Viết không qui cách:

Kết quả thu được: dữ liệu sẽ in từ trái sang phải; mỗi ký tự chiếm 1 vị trí trên màn hình; in hết dữ liệu thuộc biến thì dừng.

- Viết có qui cách:

Mẫu

CH : m

ST : m

BL : m

Trong đó:

- CH là biến kiểu ký tự
- ST là biến kiểu chuỗi ký tự
- BL là biến kiểu logic
- m là số vị trí trên màn hình dành để in các biến CH, ST, BL
- Kết quả thu được:

Dữ liệu được in từ phải sang trái căn cứ vào số vị trí dành để in biến theo qui cách đã chỉ ra trong lệnh. Nếu dữ liệu in ra không sử dụng hết m vị trí theo qui cách đã chỉ ra, thì ký tự trống sẽ được điền vào phía trái của dữ liệu được in. Nếu số vị trí m được chỉ ra trong qui cách lại nhỏ hơn số vị trí thực sự để in dữ liệu thì dữ liệu vẫn được in ra đầy đủ.

Ví dụ 5.17: Hãy viết 1 chương trình in ra 2 dòng tùy ý; gán tên, tuổi, ngành học vào các biến tương ứng sau đó in các thông tin đó ra màn hình.

```
Program VIDU1;  
Uses Crt; {Khai báo dùng unit chuẩn CRT}  
Var Ten, Nganhhoc: String[25];  
    Tuoi: Byte;
```



```
BEGIN
    ClrScr;  {Xoá màn hình}
    WriteLn('Chương trình minh hoa');
    WriteLn('*****');
    WriteLn;
    {Gán dữ liệu cho các biến}
    Ten := 'Nguyen Thanh Lam';
    Tuổi := 20;
    Nganhhoc := 'Tin hoc';
    {In thông tin trong các biến ra màn hình}
    WriteLn('Tentoi la:', Ten, ' nam nay toi', Tuổi:3, ' tuoi');
    WriteLn('Toi đang học ngành ', Nganhhoc);
    WriteLn;
    WriteLn(' Xin chào các bạn. Hẹn gặp lại');
    WriteLn(' Bạn hãy nhấn phím ENTER để kết thúc!');
    ReadLn;
END.
```

5.3.2 Lệnh nhập dữ liệu từ bàn phím

5.3.2.1 Ý nghĩa

Để đưa 1 giá trị vào 1 biến, ngoài phép gán rất thuận tiện và dễ sử dụng nhưng phải gán sẵn trong chương trình, ta còn có thể dùng các thủ tục READ và READLN để nhập dữ liệu từ các thiết bị nhập của máy tính. Trong phạm vi chương này, chúng ta chỉ xét việc dùng các lệnh nêu trên để nhập dữ liệu từ bàn phím - thiết bị nhập chuẩn.

5.3.2.2 Cú pháp

READ(biến1, biến2, ..., biếnn); (1)

READLN(biến1, biến2, ..., biếnn); (2)

READLN; (3)

5.3.2.3 Sự thực hiện

Dạng (1): Khi thực hiện chương trình, gặp lệnh này máy tính dừng lại chờ người sử dụng gõ dữ liệu (từ bàn phím) lần lượt cho các biến theo thứ tự đã liệt kê trong câu lệnh. Kết thúc việc nhập, người sử dụng gõ phím ENTER. Với dạng (1) nếu số lượng dữ liệu nhập vào nhiều hơn số lượng biến được chỉ ra trong câu lệnh thì những dữ liệu dư ra vẫn được lưu lại trong vùng đệm và sẽ tự động chuyển vào các biến trong các lệnh READ hoặc READLN tiếp theo trong chương trình.

Dạng (2): Khi thực hiện chương trình, gặp lệnh này máy tính dừng lại chờ người sử dụng gõ dữ liệu (từ bàn phím) lần lượt cho các biến theo thứ tự đã liệt kê trong câu lệnh. Kết thúc việc nhập, người sử dụng gõ phím ENTER. Dạng (2) khác với dạng (1) ở chỗ: nếu số lượng dữ liệu

nhập vào nhiều hơn số lượng biến được chỉ ra trong câu lệnh thì những dữ liệu dư ra sẽ bị loại bỏ mà không được lưu lại trong vùng đệm, nên không gây ảnh hưởng gì tới các lệnh READ hoặc READLN tiếp theo trong chương trình.

Dạng (3): Tạm dừng chương trình, chờ người sử dụng gõ phím ENTER để tiếp tục thực hiện các lệnh tiếp theo trong chương trình.

5.3.2.4 Chú ý

- Các tham số của các lệnh nhập dữ liệu đặt trong dấu ngoặc () chỉ cho phép là các biến mà thôi;
- Khi nhập dữ liệu từ bàn phím lưu ý dùng ít nhất một dấu cách (khoảng trống) để phân cách các giá trị cần nhập cho các biến khác nhau, không hạn chế số dấu cách.
- Trong quá trình nhập dữ liệu, các lệnh nhập đã nêu sẽ tự kiểm tra tính tương thích giữa dữ liệu nhập vào với kiểu của biến đã được khai báo. Nếu không phù hợp về kiểu, máy tính sẽ thông báo lỗi và ngay lập tức cho dừng việc nhập dữ liệu.
- Biến kiểu Boolean không nhập được bằng hình thức này.
- Để nhập dữ liệu từ bàn phím người ta thường dùng dạng (2) để tránh việc dữ liệu còn lưu ở vùng đệm tự động chuyển vào biến khi người sử dụng không kiểm soát được.

Ví dụ 5.18 :

```
VAR    a, b, c: INTEGER;  
        x: REAL;
```

Nếu dùng dạng (1) ta viết các câu lệnh:

```
READ (a, b) ;  
READ (x) ;
```

và nhập các số từ bàn phím như sau:

```
164 188 2 (↵)  
5.5 (↵)
```

ta thu được dữ liệu trong các biến là:

```
a = 164  
b = 188  
x = 2
```

Vì dữ liệu nhập thừa từ lệnh nhập phía trên đã tự động chuyển vào lệnh nhập ngay tiếp sau, do đó đôi khi ta nhận được dữ liệu không hoàn toàn theo ý muốn.

Trái lại, nếu dùng dạng (2) ta viết các câu lệnh:

```
READLN (a, b) ;  
READLN (x) ;
```

và nhập các số từ bàn phím như sau:

```
164 188 2 (↵)  
5.5 (↵)
```

ta thu được dữ liệu trong các biến là:

a=164

b=188

x=5.5

Vì các dữ liệu nhập thừa từ lệnh nhập trên không còn tác dụng khi lệnh đó kết thúc.

5.3.2.5 Ví dụ

Ví dụ 5.19: Lập chương trình nhập từ bàn phím kích thước 2 cạnh của một hình chữ nhật. Tính chu vi, diện tích của hình chữ nhật đó. In kết quả ra màn hình.

```
PROGRAM vidu_5_19;
USES Crt;
VAR    a,b,cv,dt:Real;
      {a: cạnh dài, b: cạnh ngắn, cv: chu vi, dt: diện tích}
BEGIN
  CLRSCR;
  READLN(a,b);
  cv:=2*(a+b);
  dt:=a*b;
  WRITELN('Chu vi hình chữ nhật = ',cv:8:3);
  WRITELN('Diện tích hình chữ nhật = ',dt:8:3);
  READLN;
END.
```

5.3.3 Kết hợp WRITE và READLN khi nhập dữ liệu

5.3.3.1 Ý nghĩa

Trong chương trình nếu chỉ dùng lệnh READLN để nhập dữ liệu từ bàn phím vào biến sẽ có nhược điểm là không có sự chỉ dẫn cho người sử dụng biết cần nhập loại dữ liệu gì, cho biến nào. Để khắc phục nhược điểm trên, ta có thể kết hợp 2 thủ tục WRITE - in ra lời chỉ dẫn và READLN để nhập dữ liệu từ bàn phím vào biến, như vậy sẽ làm cho chương trình trở nên dễ sử dụng và đẹp mắt hơn.

5.3.3.2 Cú pháp

WRITE(" Lời chỉ dẫn.....");

READLN(biến1, biến2,..., biếnn);

5.3.3.3 Chú ý

Không nên dùng WRITELN để in ra lời chỉ dẫn nhập dữ liệu vì sau khi in xong lời chỉ dẫn con trỏ màn hình sẽ bị dời xuống đầu dòng tiếp theo.

5.3.3.4 Ví dụ

Ví dụ 5.20: Xét đoạn chương trình 1:

```
WRITE(' Chiều dài của hình chu nhật  a= ');  
READLN(a);  
WRITE(' Chiều rộng của hình chu nhật b= ');  
READLN(b);
```

.....

- Kết quả khi chạy chương trình:

```
Chiều dài của hình chu nhật  a= _ {Chờ NSD nhập dữ liệu cho biến a}  
Chiều rộng của hình chu nhật b= _ {Chờ NSD nhập dữ liệu cho biến b}
```

- Xét đoạn chương trình 2:

```
WRITELN(' Chiều dài của hình chu nhật  a= ');  
READLN(a);  
WRITELN(' Chiều rộng của hình chu nhật b= ');  
READLN(b);
```

.....

- Kết quả khi chạy chương trình:

```
Chiều dài của hình chu nhật  a=  
_ {Chờ NSD nhập dữ liệu cho biến a}  
Chiều rộng của hình chu nhật b=  
_ {Chờ NSD nhập dữ liệu cho biến b}
```

- Nhận xét:

Cả 2 cách thực hiện đều đúng nhưng nên viết theo đoạn chương trình 1 sẽ hợp lý hơn.

BÀI TẬP CHƯƠNG 5

Bài 5.1. Trong một chương trình có các khai báo sau:

```
Const  max  = 100;
Var    x, y: real;
       m, n, : integer;
       a: char;
       b: boolean;
```

Cho biết câu lệnh nào sau đây là sai? Vì sao?

x := max * n;	b := m > n;	readln(m+n, x, y);
y := -0.1;	b := a + m > n;	readln(a);
max := m+n;	a := a + 1;	read(b);
m+n := max;	m := m + 1;	readln(max, x, y, m, n);
m := n;	inc(m);	writeln(b);
m := x;	inc(x);	writeln(m+n);
x := n;	readln(x);	writeln(x+m>y+n);
m := m mod x;	readln(a);	writeln(a:8:2);
m := max + n;	readln(m, n, x, y);	write(x:8:2, m+n:8);

Bài 5.2. In ra màn hình các dòng sau:

```
*****
*   Truong:  DAI HOC DIEN LUC           *
*   Khoa   :  Cong nghe thong tin       *
*   Ho ten:  Nguyen Thanh Lam           *
*****
```

Bài 5.3. Lập trình in họ tên, ngày tháng năm sinh của một sinh viên bất kỳ và hiển thị các thông tin đó lên màn hình theo mẫu:

```
Sinh vien:  ....
Sinh ngay:  ....  thang ....  nam .....
```

Bài 5.4. Giả sử a là biến thực nhận giá trị 21.0547 và b là biến thực nhận giá trị là -16.05308. Cho biết các câu lệnh in có định dạng sau sẽ in ra a và b như thế nào.

```
Writeln(a:8:2);
Writeln(a:5)
Writeln(b:10:3);
Writeln(a:7:0)
```

Bài 5.5. Nhập từ bàn phím 2 số thực bất kỳ, tính tổng, hiệu, tích của chúng và hiển thị kết quả lên màn hình theo mẫu:

So thu nhất:

So thu hai:

Tong cua chung la:

Hieu cua chung la:

Tich cua chung la:

Bài 5.6. Nhập x từ bàn phím. Tính và hiển thị lên màn hình giá trị biểu thức sau:

$$Y = (a^3 + 2a) / \ln(b^2 + 21)$$

Trong đó: $a = e^x + 1$

$$b = |x| + 1$$

Bài 5.7. Nhập vào từ bàn phím 2 biến X,Y. Hãy hoán vị X,Y và in kết quả trước và sau khi hoán vị.

Bài 5.8. Một quả bóng được ném theo phương ngang với vận tốc ban đầu là V. Thời gian kể từ khi ném tới khi bóng chạm đất là T.

Lập trình nhập V (m/s) và T (s) từ bàn phím. Tính tầm xa S(m) và độ cao H(m) mà bóng đạt được. Chạy thử với V=25; T=3;

$$\text{Gợi ý: } S = V.T; H = gt^2/2.$$

Bài 5.9. Lãi suất tiết kiệm hàng tháng của ngân hàng A là $k = 1.25\%$. Tính số tiền có được sau t tháng gửi nếu biết số tiền ban đầu là x .

Bài 5.10. Cho ba điểm A(x_a, y_a), B(x_b, y_b), C(x_c, y_c) tạo thành một tam giác. Tính chu vi và diện tích tam giác ABC.

Chương 6

CÁC CÂU LỆNH CÓ CẤU TRÚC

6.1. Câu lệnh ghép (khối lệnh)

Khối lệnh là một nhóm các lệnh bắt đầu bởi từ khóa BEGIN và kết thúc bởi từ khóa END; nhằm thực hiện một công việc chung nào đó.

Cú pháp: BEGIN

Câu lệnh 1;

Câu lệnh 2;

... ..

Câu lệnh n;

END;

Thực hiện: Các câu lệnh bên trong vẫn thực hiện tuần tự như thông thường. Tuy nhiên câu lệnh ghép có tác dụng nhóm nhiều lệnh lại thành một khối lệnh (được coi như một câu lệnh duy nhất) để phù hợp với các cú pháp của các cấu trúc điều khiển sau này.

6.2. Các câu lệnh rẽ nhánh và lựa chọn

6.2.1. Lệnh rẽ nhánh IF

6.2.1.1 Ý nghĩa

Lệnh rẽ nhánh IF được dùng để giải quyết tình huống các công việc có được thực hiện hay không là tùy thuộc vào kết quả của biểu thức logic (biểu thức điều kiện).

6.2.1.2 Cú pháp

Turbo Pascal cung cấp 2 mẫu câu lệnh rẽ nhánh sau:

Dạng khuyết

If <điều kiện> **then**
<Công việc>;

Dạng đủ

If <điều kiện> **then**

<Công việc1>

Else

<Công việc2>;

Trong đó:

- <Điều kiện> là biểu thức logic
- <Công việc> có thể là 1 lệnh đơn giản hoặc 1 khối lệnh. Nếu 1 khối lệnh phải được đặt trong Begin..End;

6.2.1.3 Sự hoạt động

Dạng khuyết

Nếu <Điều kiện> có giá trị TRUE thì thực hiện <Công việc>;

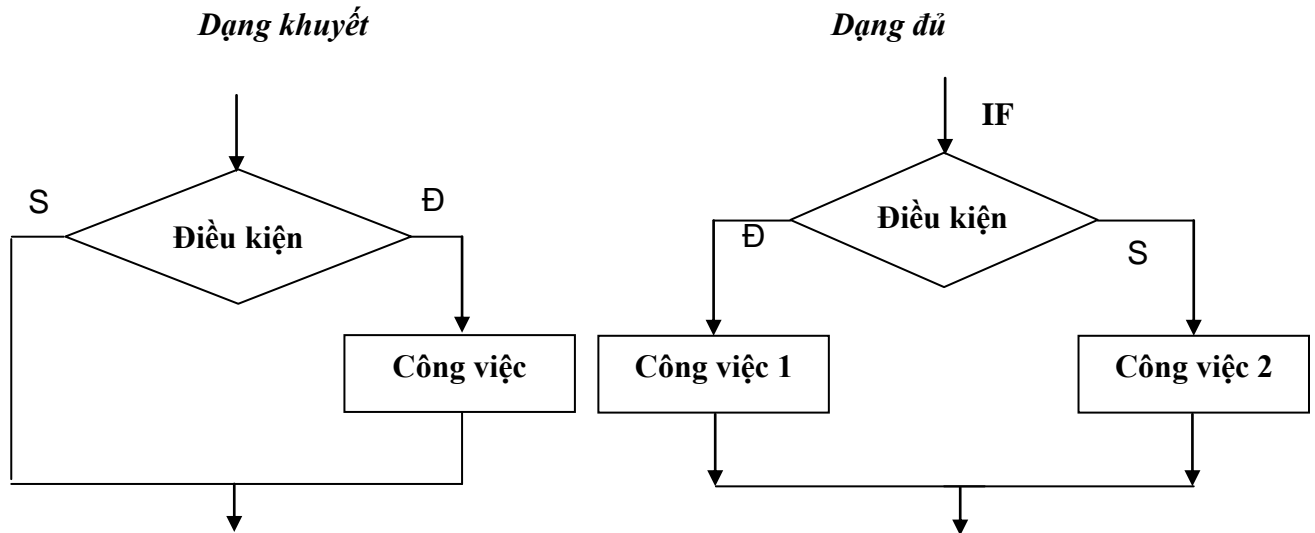
Dạng đủ

Nếu <Điều kiện> có giá trị TRUE thì thực hiện <Công việc 1>;

Nếu <Điều kiện> có giá trị FALSE thì ra khỏi lệnh IF mà không làm gì.

Nếu <Điều kiện> có giá trị FALSE thì thực hiện <Công việc 2>;

6.2.1.4 Lưu đồ minh họa câu lệnh



6.2.1.5 Ví dụ:

Ví dụ 6.1: Viết chương trình giải phương trình bậc nhất $ax+b=0$ với các hệ số a, b nhập từ bàn phím.

```
Program    giai_phuong_trinh_bac_nhat;
Uses Crt;
Var  a, b, x: Real;
BEGIN
    ClrScr;
    Write('Nhap he so a= '); ReadLn(a);
    Write('Nhap he so b= '); ReadLn(b);
    If a<>0 then
        Begin
            x:=-b/a;
            WriteLn('Phuong trinh co nghiem x= ',x:8:3);
        End
    Else {a = 0}
        If b=0 then
            WriteLn('Phuong trinh co vo so nghiem')
        Else {a = 0, b <> 0}
            WriteLn('Phuong trinh vo nghiem ');
        ReadLn;
    END.
```


6.2.1.6 Chú ý

- Câu lệnh ngay trước từ khoá ELSE của lệnh IF không có dấu ;
- Trong trường hợp dùng các lệnh IF lồng nhau, nên viết IF ... THEN ... ELSE thành khối để tiện kiểm tra khi có lỗi logic.

Chẳng hạn ta có thể viết một lệnh IF lồng nhau theo cấu trúc sau:

```
IF <Điều kiện 1> THEN
    <Công việc 1>
ELSE
    IF <Điều kiện 2> THEN
        <Công việc 2>
    ELSE <Công việc 3>;
```

- Cần nắm được qui ước sau đây để hiểu ELSE đi với IF nào “ELSE gắn với IF gần nhất với nó ngoài cặp BEGIN .. END sát nó.
- Nên trình bày chương trình khoa học, các lệnh tương ứng ngang hàng nhau, lệnh nào nằm trong điều kiện thì nên lui vào một chút so với câu điều kiện của nó, bạn sẽ dễ đọc chương trình và chỉnh sửa lỗi.

6.2.2 Câu lệnh lựa chọn CASE

6.2.2.1 Ý nghĩa

Câu lệnh IF chỉ thực hiện rẽ 2 nhánh tương ứng với giá trị của biểu thức <Điều kiện>. Khi cần lựa chọn để thực hiện một trong nhiều khả năng khác nhau ta phải dùng rất nhiều lệnh IF. Câu lệnh CASE sẽ giới thiệu sau đây là lệnh rẽ nhánh theo giá trị cho phép lựa chọn để thực hiện 1 trong nhiều công việc.

6.2.2.2 Cú pháp

<i>Dạng khuyết</i>	<i>Dạng đủ</i>
Case <Biểu thức> of	Case <Biểu thức> of
Tập hằng 1: <Công việc 1>;	Tập hằng 1: <Công việc 1>;
Tập hằng 2: <Công việc 2>;	Tập hằng 2: <Công việc 2>;
....
Tập hằng i: <Công việc i>;	Tập hằng i: <Công việc i>;
....
Tập hằng n: <Công việc n>;	Tập hằng n: <Công việc n>;
End;	Else <Công việc n+1>;
	End;

Trong đó:

- <Điều kiện> là biểu thức số học, cho kết quả là một kiểu vô hướng đếm được (nguyên, ký tự, logic, liệt kê).
- <Tập hằng i> (i=1,...,n) có thể là các hằng và các đoạn hằng, ví dụ:

3: <Công việc i>;

5, 10..15: <Công việc i>;

2, 4, 6: <Công việc i>;

<5 : <Công việc i>;

'a'..'z': <Công việc i>;

- <Công việc i> có thể là lệnh đơn giản hoặc khối lệnh. Nếu 1 khối lệnh phải được đặt trong Begin..End;

Giá trị của <Tập hằng i> phải cùng kiểu với kiểu của <Biểu thức> và cũng phải là kiểu vô hướng liệt kê.

6.2.2.3 Sự hoạt động

Sự thực hiện của lệnh CASE ..OF phụ thuộc vào giá trị của <Biểu thức>

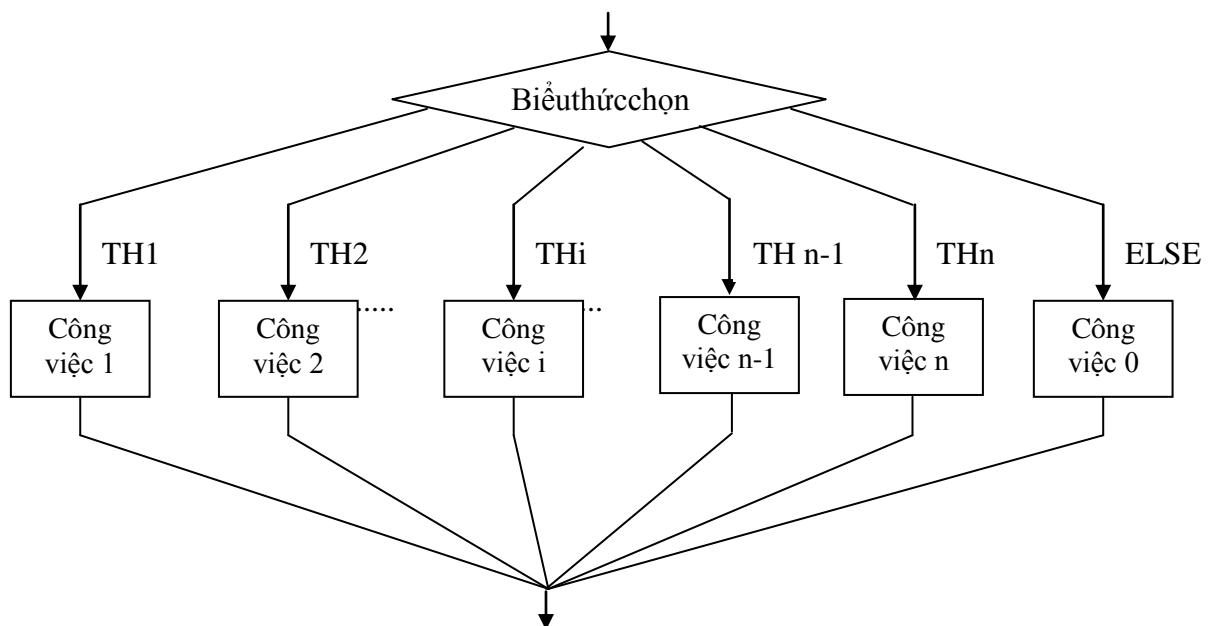
Dạng khuyết

Nếu giá trị của <Biểu thức> thuộc <tập hằng i> thì <Công việc i> được thực hiện và ra khỏi lệnh CASE; Nếu giá trị của <Biểu thức> không thuộc tập hằng nào thì thoát ra khỏi lệnh CASE mà không làm gì.

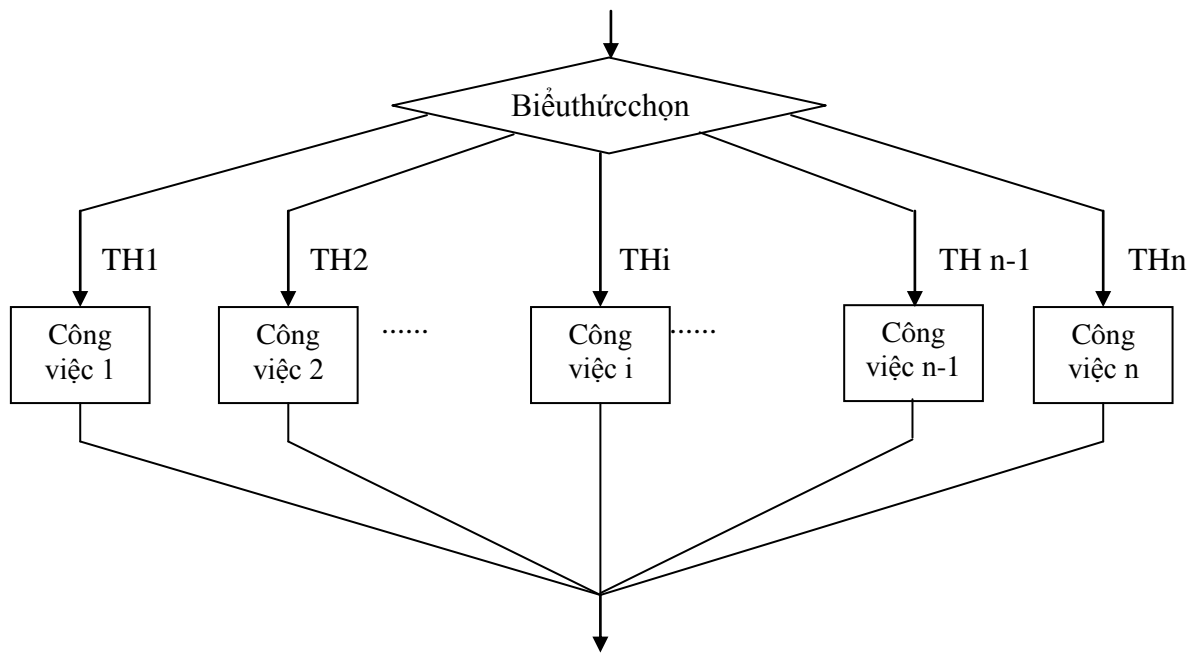
Dạng đủ

Nếu giá trị của <Biểu thức> thuộc <tập hằng i> thì <Công việc i> được thực hiện và ra khỏi lệnh CASE; Nếu giá trị của <Biểu thức> không thuộc tập hằng nào thì thực hiện <Công việc n+1> rồi thoát ra khỏi lệnh CASE .

Có thể biểu diễn câu lệnh theo sơ đồ hoạt động sau:



Hình 6.2a: Sơ đồ hoạt động của câu lệnh CASE dạng đủ.



Hình 6.2b: Sơ đồ hoạt động của câu lệnh CASE dạng khuyết.

6.2.2.4 Lệnh IF .. THEN .. ELSE có thể chuyển thành CASE .. OF như sau

IF <Điều kiện> THEN
 <Công việc1>
 ELSE
 <Công việc2>;

Case <Điều kiện> of
 TRUE : <Công việc 1>;
 FALSE: <Công việc 2>;
 End;

Ví dụ 6.2: Viết chương trình nhập vào tháng (và năm nếu cần), máy sẽ đưa ra số ngày của tháng đó.

```

Program VIDU_6_2;
Uses Crt;
Var songay, thang: Byte;
    nam: Integer;
BEGIN
  ClrScr;
  Write('Cho biet thang (dang so): '); ReadLn(thang);
  Write('Cho biet nam '); ReadLn(nam);
  Case thang of
    4,6,9,11: songay:=30;
    2: If nam mod 4 = 0 then
        if (nam mod 100 = 0) and (nam mod 400 <> 0) then
            songay:=28
        else songay:=29;
      Else songay:=28;
  
```

```
Else { else của CASE}
    songay:=31;
End; {của Case}
{In kết quả}
WriteLn('Thang ', thang, '/', nam, ' có: ', songay, ' ngày')
ReadLn;
END.
```

Ví dụ 6.3: Nhập vào hai số x, y. Tính giá trị biểu thức sau: $F = \begin{cases} x + y & \text{Nếu } x > y \\ |x + y| & \text{Nếu } x \leq y \end{cases}$

Với bài này ta có thể giải bằng cách sử dụng câu lệnh IF hoặc CASE

Cách 1: sử dụng câu lệnh IF

```
Program vidu_6_3_IF;
Var x, y, F: real;
Begin
    Write(' Nhập x = '); readln(x);
    Write(' Nhập y = '); readln(y);
    If x > y then
        F := x + y
    Else {ngược lại của x > y tức là x ≤ y}
        F := abs(x + y);
    Writeln(' Giá trị của biểu thức F = ', F:8:2);
    Readln;
End.
```

Cách 2: sử dụng câu lệnh CASE

```
Program vidu_6_3_CASE;
Var x, y, F: real;
Begin
    Write(' Nhập x = '); readln(x);
    Write(' Nhập y = '); readln(y);
    Case x > y of
        True: F := x + y;
        False: F := abs(x + y);
    End;
    Writeln(' Giá trị của biểu thức F = ', F:8:2);
    Readln;
End.
```

Ví dụ 6.4: Lập chương trình tính chiều dài đoạn thẳng AB khi đã biết tọa độ hai đầu mút (Xa, Ya) và (Xb, Yb).

```
Program VIDU_6_4;
```

```
Uses  Crt;
Var   Xa, Ya, Xb, Yb, d: Real;
BEGIN
    ClrScr;
    Write('Nhap toa do x cua diem A Xa= '); ReadLn(Xa);
    Write('Nhap toa do y cua diem A Ya= '); ReadLn(Ya);
    Write('Nhap toa do x cua diem B Xb= '); ReadLn(Xb);
    Write('Nhap toa do y cua diem B Yb= '); ReadLn(Yb);
    { Tinh do dai doan thang AB }
    d := SQRT(SQR(Xa-Xb)+SQR(Ya-Yb));
    { In ket qua }
    WriteLn('Do dai doan thang AB = ', d:10:4);
    ReadLn;
END.
```

Ví dụ 6.5: Lập chương trình biện luận nghiệm của phương trình bậc hai $ax^2+bx+c=0$ với các hệ số a, b nhập vào từ bàn phím.

```
Program  VIDU_6_5;
Uses     Crt;
Var      a, b, c, x1, x2, delta: Real;
BEGIN
    ClrScr;
    Write('Nhap he so thu nhat a= '); ReadLn(a);
    Write('Nhap he so thu hai  b= '); ReadLn(b);
    Write('Nhap he so thu ba   c= '); ReadLn(c);
    If a=0 then
        WriteLn('Phuong trinh bac hai suy bien, khong xet!')
    Else
        Begin
            delta := SQR(b)-4*a*c; { Tinh delta }
            { Xet delta }
            If delta < 0 then WriteLn('Phuong trinh vo nghiem')
            Else
                If Delta = 0 then
                    Begin
                        WriteLn('Phuong trinh co nghiem kep');
                        x1:=-b/(2*a);
                        WriteLn('PT co nghiem kep x1=x2= ',x1:8:3);
                    end
                Else
                    Begin
```

```
WriteLn('PT co hai nghiệm riêng biet');  
x1:=(-b+SQRT(Delta))/(2*a);  
x2:=(-b-SQRT(Delta))/(2*a);  
WriteLn('Nghiệm thu nhat cua PT la x1= ',x1:8:3);  
WriteLn('Nghiệm thu hai cua PT la x2= ',x2:8:3);  
End;  
End;  
ReadLn;  
END.
```

6.3. Câu lệnh lặp xác định FOR

6.3.1. Ý nghĩa:

Câu lệnh lặp là câu lệnh mà cho phép thực hiện lặp đi lặp lại nhiều lần một số công việc. Câu lệnh FOR là câu lệnh cho phép lặp với số lần lặp biết trước.

6.3.2 Câu lệnh FOR tiên (Dạng 1)

6.3.2.1 Cú pháp

For Biến_điều_khiển:=<Giá trị đầu> **to** <Giá trị cuối> **do** <Công việc>;

Trong đó:

- Biến_điều_khiển, <Giá trị đầu>, <Giá trị cuối> phải là dữ liệu kiểu vô hướng đếm được như kiểu số nguyên, kiểu ký tự, kiểu logic...
- <Công việc> có thể là một lệnh hoặc một khối lệnh.

6.3.1.2 Sự hoạt động

Bước 1: Biến_điều_khiển được khởi tạo là <Giá trị đầu>

Bước 2: Kiểm tra điều kiện Biến_điều_khiển có nhỏ hơn hay bằng (\leq) Giá_trị_cuối hay không,

Nếu đúng thực hiện bước 3, nếu sai thực hiện bước 4.

Bước 3:

Bước 3.1: Thực hiện <Công việc>

Bước 3.2: Gán giá trị ngay sau cho Biến_điều_khiển bằng succ(của biến điều khiển ở bước trước)

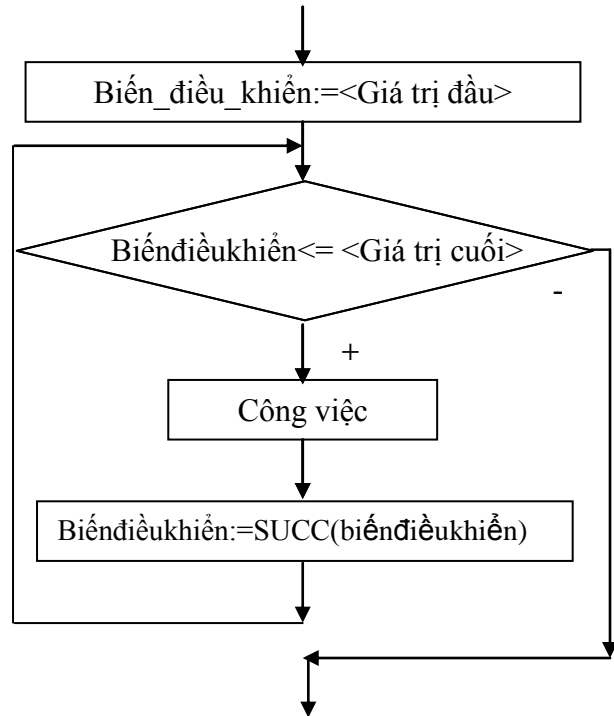
Bước 3.3: Quay lại bước 2.

Bước 4: Ra khỏi vòng lặp For

Chú ý: Giá trị đầu thường là \leq giá trị cuối, tuy nhiên nếu giá trị đầu $>$ giá trị cuối thì vòng lặp sẽ dừng ngay khi kiểm tra điều kiện đầu tiên.

6.3.2.3 Lưu đồ thể hiện sự hoạt động

Hình 6.4. Lưu đồ hoạt động của vòng lặp xác định FOR tiến



6.3.2.4. Ví dụ

Ví dụ 6.6: Tính tổng $S=1+1/2+1/3+\dots+1/n$. Biết rằng n là một số nguyên dương được đọc vào từ bàn phím.

```

Program VIDU_6_6; { Chương trình sử dụng FOR tiến }
Uses Crt;
Var i, n: Integer;
    s: Real;
BEGIN
  ClrScr;
  Write('Nhập n= '); ReadLn(n);
  s:=0;
  For i:=1 to n do
    s:=s+1/i;
  WriteLn('Tổng S= ', s:8:3);
  ReadLn;
END.
  
```

6.3.3 Câu lệnh FOR lùi (Dạng 2)

6.3.3.1 Cú pháp

For Biến_điều_khiển:=<Giá trị đầu> **to** <Giá trị cuối> **do** <Công việc>;

Trong đó:

- Biến_điều_khiển, <Giá trị đầu>, <Giá trị cuối> phải là dữ liệu kiểu vô hướng đếm được như kiểu số nguyên, kiểu ký tự, kiểu logic...

- <Công việc> có thể là một lệnh hoặc một khối lệnh. Nếu là một khối lệnh thì phải được đặt trong **Begin ... End**;
- Thường thì giá_trị_đầu \geq giá_trị_cuối.

6.3.3.2 Sự hoạt động

Bước 1: Biến_điều_khiển được khởi tạo bằng <Giá trị đầu>

Bước 2: Kiểm tra điều kiện Biến_điều_khiển lớn hơn hay bằng (\geq) <Giá trị cuối>

Nếu đúng thực hiện bước 3, nếu sai thực hiện bước 4.

Bước 3:

Bước 3.1: Thực hiện <Công việc>

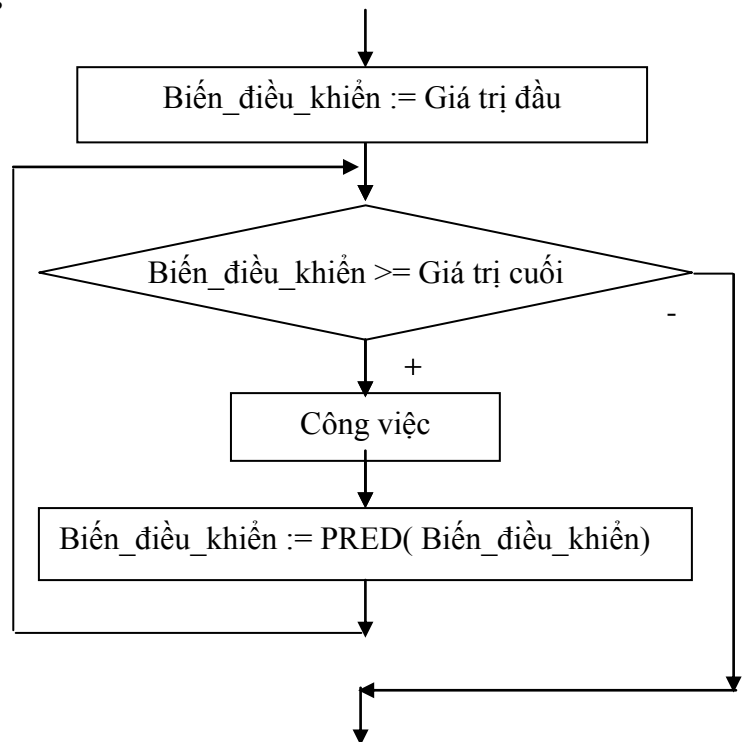
Bước 3.2: Gán giá trị cho Biến_điều_khiển bởi pred (biến điều khiển ở bước trước)

Bước 3.3: Quay lại bước 2.

Bước 4: Ra khỏi vòng lặp For

6.3.3.3 Lưu đồ thể hiện sự hoạt động

Hình 6.5. Lưu đồ hoạt động của vòng lặp xác định FOR lùi



Ví dụ 6.7: Xét ví dụ 6.6 nhưng sử dụng câu lệnh FOR lùi

```
Program VIDU_6_7;  
Uses Crt;  
Var i, n: Integer;  
    s: Real;  
BEGIN  
    ClrScr;  
    Write('Nhập n= '); ReadLn(n);  
    s:=0;  
    For i:=n downto 1 do s:=s+1/i;
```



```
WriteLn('Tong S= ',s:8:3);  
ReadLn;  
END.
```

Ví dụ 6.8: Minh họa cho việc sử dụng biến điều khiển có kiểu CHAR: Viết chương trình in ra màn hình 2 dòng: dòng thứ nhất in các chữ thường a ÷z, dòng thứ hai in các chữ IN HOA từ Z ÷A.

```
Program vidu_6_8;  
Uses Crt;  
Var ch: Char;  
BEGIN  
    ClrScr;  
    WriteLn('In cac ky tu viet thuong tu a ==> z');  
    For ch:='a' to 'z' do Write(ch:3);  
    WriteLn;  
    WriteLn;  
    WriteLn('In cac ky tu viet IN HOA tu Z ==> A');  
    For ch:='Z' downto 'A' do Write(ch:3);  
    WriteLn;  
    ReadLn;  
END.
```

6.4. Câu lệnh lặp không xác định WHILE và REPEAT

6.4.1 Ý nghĩa

Rất nhiều trường hợp bài toán cần phải lặp đi lặp lại một hoặc một số công việc song lại không thể xác định được là phải lặp bao nhiêu lần, do vậy không thể dùng lệnh FOR được. Pascal cung cấp 2 lệnh là REPEAT và WHILE để lặp cho những trường hợp không biết trước số lần lặp.

6.4.2 Câu lệnh lặp không xác định kiểm tra điều kiện sau REPEAT

Câu lệnh REPEAT là câu lệnh lặp thực hiện công việc trước, kiểm tra điều kiện sau. Câu lệnh này thường được sử dụng trong những bài toán mà ta chỉ biết một điều kiện kiểm tra, cần lặp đi lặp lại công việc cho đến khi điều kiện này được thỏa mãn. Những trường hợp như tính toán thỏa mãn sai số cho trước, chạy thử một chương trình nhiều lần để kiểm tra mà chỉ cần ấn Ctrl+F9 một lần, chương trình chỉ dừng khi ta chọn K hoặc k cho câu hỏi “Co tiếp tục nữa không? (C/K)” hay nhấn phím ESC cho câu hỏi “Co tiếp tục nữa không? (nhấn ESC để dừng),... Thực hiện câu lệnh REPEAT - UNTIL như sau:

6.4.2.1 Cú pháp

Repeat

<Công việc>;

Until <Điều kiện>;

Trong đó:

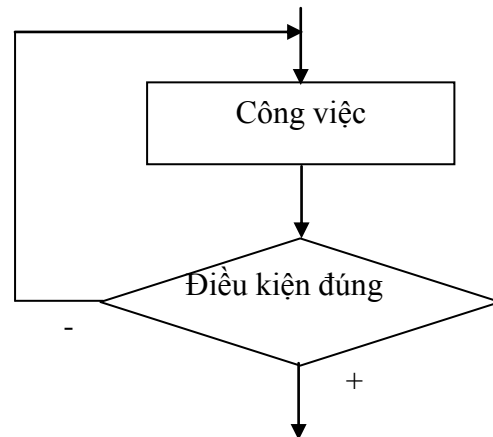
- <Điều kiện> là một biểu thức logic, trả về giá trị TRUE hoặc FALSE
- <Công việc> Có thể là một lệnh hoặc một khối lệnh

6.4.2.2 Sự hoạt động

- i) Thực hiện công việc Công_việc
- ii) Tính giá trị của biểu thức logic điều_kiện,
- iii) Nếu sai, quay về bước i),
- iv) Nếu đúng, dừng và kết thúc vòng lặp.

6.4.2.3 Lưu đồ thể hiện sự hoạt động

Hình 6.6. Lưu đồ hoạt động của câu lệnh Repeat



6.4.2.4 Chú ý

- <Công việc> đặt giữa 2 từ khoá Repeat và Until có thể là một lệnh hoặc một khối lệnh song không cần phải đặt trong Begin...End;
- Trong khi thực hiện <Công việc> phải có một lệnh làm thay đổi giá trị của một biến thuộc biểu thức logic <Điều kiện> để làm cho biểu thức logic tiến đến TRUE nhằm kết thúc vòng lặp.
- <Công việc> được thực hiện trước, <Điều kiện> được kiểm tra sau nên ít nhất <Công việc> cũng được thực hiện một lần, ngay cả khi <Điều kiện> đã có giá trị TRUE.
- Câu lệnh lặp Repeat ... Until cũng dùng được trong trường hợp đã biết trước số lần lặp.

6.4.2.5 Ví dụ

Ví dụ 6.9: Tính tổng $S=1+1/2+1/3+...+1/n$. Biết rằng n là một số nguyên dương được đọc vào từ bàn phím.

```
Program VIDU_6_9;  
Uses Crt;  
Var i, n: Integer;  
    s: Real;  
  
BEGIN  
    ClrScr;  
    Write('Nhập n= '); ReadLn(n);  
    s:=0; i:=1;  
    Repeat  
        s:=s+1/i;
```

```
        i:=i+1;
Until i>n;
WriteLn('Tong S= ',s:8:3);
ReadLn;
END.
```

6.4.2.6 Ứng dụng của câu lệnh lặp Repeat ... Until

a) Mẫu 1: dùng để lặp lại việc nhập dữ liệu cho đến khi dữ liệu nhập là hợp lý.

```
.....
{ Nhập dữ liệu }
Repeat
    ClrScr;
    Write('Nhap diem thi mon Tin hoc dai cuong DIEM= ');
    ReadLn(DIEM);
    If (DIEM<0) or (DIEM>10) then
        Begin
            WriteLn('#7, 'Du lieu sai, moi nhap lai');
            ReadLn;
        End;
Until (DIEM>=0) or (DIEM<=10);
....
```

b) Mẫu 2: dùng để quay vòng một đoạn hoặc cả một chương trình theo ý muốn.

```
...
Var
    ...
    TL: Char;
BEGIN
    Repeat
        ClrScr;
        { Nhập dữ liệu }
        ....
        { Xử lý }
        ...
        { In kết quả }
        Write('Co thuc hien chuong trinh lan nua khong (C/K)? ');
        ReadLn(TL);
    Until (TL='k') or (TL='K');
END.
```

Ví dụ 6.10: Viết chương trình giải phương trình bậc nhất $ax+b=0$ với các hệ số a, b nhập từ bàn phím có quay vòng.

```
Program    VIDU_6_10;
Uses      Crt;
Var       a, b, x: Real;
          TL: Char;
BEGIN
  Repeat
    ClrScr;
    Write('Nhap he so a= '); ReadLn(a);
    Write('Nhap he so b= '); ReadLn(b);

    If a<>0 then
      Begin
        x:=-b/a;
        WriteLn('Phuong trinh co nghiem x= ',x:8:3);
      End
    Else
      If b=0 then WriteLn('Phuong trinh co vo so nghiem')
      Else WriteLn('Phuong trinh vo nghiem ');
      {hoi de lap lai chuong trinh}
      Write('Co thuc hien chuong trinh lan nua khong (C/K)? ');
      ReadLn(TL);
  Until (TL='k') or (TL='K');
END.
```

c) Mẫu 3: Để xây dựng chương trình điều khiển bằng chọn

Ví dụ 6.11: viết chương trình nhập vào ba số a, b, c cho đến khi chúng thỏa mãn là độ dài của ba cạnh trong một tam giác. Xây dựng menu chọn công việc thực hiện

Công việc 1: tính chu vi tam giác

Công việc 2: tính diện tích tam giác

```
Program    vidu_6_11;
Var a, b, c, S, CV: real;
    ch:char;
Begin
  Repeat
    Write(' Nhap ba so a, b, c : ');readln(a,b,c);
  Until (a+b>c) and (a+c>b) and (b+c>a) and (a>0) and (b>0) and (c>0);
  Repeat
    Writeln(' Cac lua chon: ');
    Writeln(' 0: Thoat khoi chuong trinh!');
    Writeln(' 1: Tinh chu vi tam giac, ');
```

```

Writeln(' 2: Tinh dien tich tam giac,');
Write(' Lua chon cua ban la : ');readln(ch);
Case ch of
  '0': readln;
  '1': begin
        CV:=a+b+c;
        Writeln(' Chu vi tam giac la CV = ',CV:8:2);
      end;
  '2': begin
        p:=(a+b+c)/2;
        S:=sqrt(p*(p-a)*(p-b)*(p-c));
        Writeln(' dien tich tam giac la S =', S:8:2);
      end;
Until (ch:='0');
End.

```

6.4.3 Câu lệnh lặp không xác định kiểm tra điều kiện trước WHILE

Câu lệnh WHILE cũng là câu lệnh lặp với số lần lặp không biết trước giống như câu lệnh REPEAT - UNTIL. Điểm khác của câu lệnh WHILE là thực hiện kiểm tra điều kiện trước, nếu thỏa mãn (điều kiện vẫn sai) thì thực hiện công việc và nếu không thỏa mãn (điều kiện đúng) thì dừng và kết thúc vòng lặp.

6.4.3.1 Cú pháp

While <Điều kiện> **do** <Công việc>;

Trong đó:

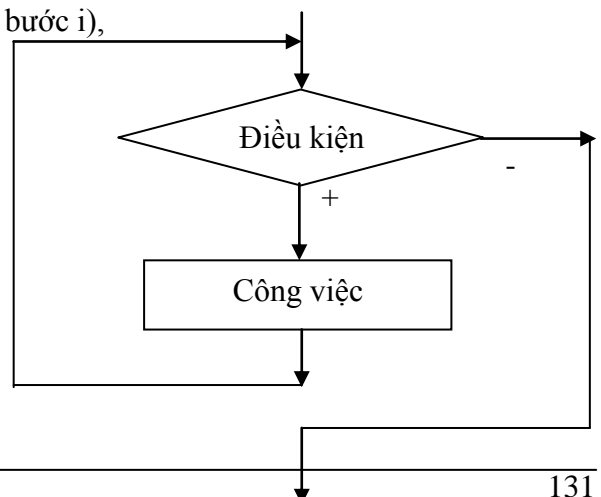
- <Điều kiện> là một biểu thức logic, trả về giá trị TRUE hoặc FALSE
- <Công việc> Có thể là một lệnh hoặc một khối lệnh, nếu là một khối lệnh thì phải được đặt trong **Begin ... End**;

6.4.3.2 Sự hoạt động

- Tính giá trị của biểu thức logic Điều_kiện,
- Nếu đúng, thực hiện công việc và quay lại bước i),
- Nếu sai, kết thúc câu lệnh While.

6.4.3.3 Lưu đồ thể hiện sự hoạt động

Hình 6.6. Lưu đồ hoạt động của câu lệnh While.



6.4.3.4. Chú ý

- Trong khi thực hiện <Công việc> phải có một lệnh làm thay đổi giá trị của một biến thuộc biểu thức logic <Điều kiện> để làm cho biểu thức logic tiến đến FALSE nhằm kết thúc vòng lặp.
- <Công việc> được thực hiện sau khi kiểm tra <Điều kiện> nên có thể <Công việc> không được thực hiện một lần nào vì ngay từ đầu <Điều kiện> đã có giá trị FALSE.
- Câu lệnh lặp **While ... do** cũng dùng được trong trường hợp đã biết trước số lần lặp.

Ví dụ 6.12: Tính tổng $S=1+1/2+1/3+\dots+1/n$. Biết rằng n là một số nguyên dương được đọc vào từ bàn phím.

```
Program VIDU_6_12;
Uses Crt;
Var i, n: Integer;
    s: Real;
BEGIN
    ClrScr;
    Write('Nhập n= '); ReadLn(n);
    s:=0; i:=1;
    While i<=n do
        Begin
            s:=s+1/i;
            i:=i+1;
        End;
    WriteLn('Tổng S= ', s:8:3);
    ReadLn;
END.
```

6.4.3.5. Xây dựng cấu trúc lặp với câu lệnh không xác định WHILE

Dưới đây giới thiệu một số bài toán thường sử dụng lặp và phương pháp giải.

a) Điều khiển vòng lặp bằng giá trị canh chừng

Ta xét ví dụ sau

Ví dụ 6.13: viết chương trình nhập vào số thực a . Với n lớn nhất bằng bao nhiêu để tổng $S =$

$1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ nhỏ hơn số a cho trước.

Với ví dụ này ta không biết trước số lần lặp vì giá trị a là giá trị bất kỳ được nhập từ bàn phím.

Ta **lấy** giá trị a là giá trị canh chừng để dừng vòng lặp. Thực hiện như sau:

- i) `Dữ_liệu_vào := giá_trị_đầu_tiên;`
- ii) **while** `dữ_liệu_vào <= giá_trị_canh_chừng_a` **do**
begin

....

Dữ_liệu_vào:=giá_trị_tiếp_theo;

end;

Chương trình chi tiết:

Program vidu_6_13;

Var a, S: real;

n, i: integer;

Begin

write(' Nhập a = ');readln(a);

S:=0; i:=1; {khởi gán giá trị ban đầu cho S và i}

while S<=a do {chúng ta sẽ chạy vòng lặp khi S còn nhỏ hơn hoặc bằng a}

Begin

S:=S+1/i;

i:=i+1;

End;

n:=i-1; {n=i-1 vì khi kết thúc vòng lặp,
giá trị của i làm S>a}

writeln(' Giá trị n lớn nhất thỏa mãn là n = ', n);

readln;

End.

b) Điều khiển vòng lặp bằng cờ báo:

Xây dựng cấu trúc lặp mà được điều khiển bằng cờ báo thường áp dụng cho các bài toán tìm kiếm sự tồn tại hay phần tử có tính chất nào đó. Với bài toán này thường sử dụng một biến logic, ta thường khởi tạo cho biến cờ báo này một giá trị là FALSE, lặp lại nhiều lần phép kiểm tra cho đến khi điều kiện được thỏa mãn tức tìm thấy, khi đó ta gán cho biến cờ báo giá trị là TRUE và vòng lặp kết thúc. Cụ thể như sau:

(i) Cờ_báo:=false;

(ii) **while not** cờ_báo **do**

begin

.....

if tìm_thấy **then** cờ_báo:=true;

end;

Ta xét ví dụ sau:

Ví dụ 6.14: Viết chương trình tìm số chính phương nhỏ nhất lớn hơn m với m nhập từ bàn phím, in kết quả lên màn hình.

Program vidu_6_14;

Var i, m: integer;

Found:boolean;

Begin

```
i:=m;
found:=false;
while found =false do
  if (frac(sqrt(i))=0) and (i mod 2 =0) then
    found:=true;
  else i:=i+1;
writeln(' So chinh phuong chan nhỏ nhất mà lớn
        hơn ',m,' là ', i);
readln;
End.
```

6.4.4. Một số câu lệnh kết thúc sớm vòng lặp hoặc chương trình

6.4.4.1. Lệnh nhảy không điều kiện (Goto) là câu lệnh nhảy không điều kiện, cho phép nhảy từ bất kì nơi nào bên trong chương trình hay chương trình con đến vị trí đã đánh dấu bằng **nhãn**. **Nhãn** là một tên, như tên biến hoặc là một số nguyên, sử dụng như sau:

Cú pháp: **Goto** nhãn;

Cách thực hiện:

1) Khai báo **nhãn**: phải khai báo nhãn tại phần đầu khai báo (xem cấu trúc chung một chương trình Pascal) theo cú pháp:

label nhãn1, nhãn2, ..., nhãnN;

2) Đánh dấu đích: Trong thân chương trình vị trí đích sẽ nhảy đến bằng lệnh Goto cần đánh dấu trước, theo cú pháp sau:

nhãn: các_câu_lệnh_đích;

3) Viết câu lệnh: **Goto nhãn**;

* Chú ý với lệnh nhảy vô điều kiện Goto:

- Có thể nhảy từ trong vòng lặp ra ngoài,
- Không cho phép nhảy từ ngoài vào trong vòng lặp, từ ngoài vào trong chương trình con.
- Nên hạn chế dùng câu lệnh nhảy vô điều kiện Goto vì nó phá vỡ tính cấu trúc của câu lệnh điều khiển, khó theo dõi.

Ví dụ 6.15: Viết chương trình nhập vào một số nguyên n, In ra màn hình số chính phương lớn nhất mà nhỏ hơn n.

```
Program vidu_6_15;
Label in_ket_qua;
var k,i:integer;
begin
  write('Nhap so can kiem tra :');readln(k);
  for i:=k downto 1 do
    if frac(sqrt(i)) = 0 then
```



```
        goto in_ket_qua;
in_ket_qua: writeln(' So chinh phuong do la : ',i);
readln;
end.
```

6.4.4.2. Lệnh chấm dứt sớm vòng lặp (Break) là câu lệnh có tác dụng chấm dứt giữa chừng một vòng lặp dù chưa kết thúc, theo cú pháp sau:

Cú pháp: Break;

Ví dụ 6.16: xét lại ví dụ 6.16 trên nhưng sử dụng câu lệnh Break

```
Program vidu_6_16;
var k,i:integer;
begin
    write('Nhap so can kiem tra :');readln(k);
    for i:=k downto 1 do
        if frac(sqrt(i)) = 0 then
            begin
                writeln(' So chinh phuong do la : ',i);
                break;
            end;
        readln;
    end.
```

*** Chú ý:**

- Lệnh Break cho phép thoát khỏi mọi kiểu vòng lặp For, While hay Repeat - Until.
- Nếu có nhiều vòng lặp lồng nhau thì câu lệnh Break cho phép thoát khỏi vòng lặp bên trong nhất chứa nó. Các vòng lặp bên ngoài vẫn hoạt động bình thường.

6.4.4.3. Lệnh thoát khỏi chương trình con (Exit) là lệnh kết thúc và thoát khỏi chương trình con, theo cú pháp sau:

Cú pháp: Exit;

Thực hiện như sau:

- Nếu ở bên trong chương trình con thì lệnh này chấm dứt chương trình con (mặc dù chưa đến câu lệnh cuối) và trở về chương trình bên ngoài đã gọi chương trình con.
 - Nếu ở chương trình chính thì lệnh này chấm dứt chương trình chính và dừng lại.
- câu lệnh Exit chỉ nên sử dụng trong các chương trình con

Ví dụ 6.17: Viết chương trình nhập vào số k, kiểm tra số k có phải là số nguyên tố hay không?

```
Program vidu_6_17;
var k,i:integer;
function lsnt(k:integer):boolean;
var i:integer;
begin
```

```
    lsnt:=true;
    for i:=2 to k-1 do
        if k mod i = 0 then
            begin
                lsnt:=false;
                exit;
            end;
    end;
end;
begin
    write('Nhap so can kiem tra :');readln(k);
    if lsnt(k) then    writeln(k,' la so nguyen to')
    else    writeln(k,' khong la so nguyen to');
    readln;
end.
```

6.4.4.4. Lệnh dừng chương trình bất thường (Halt) là lệnh dừng chương trình chính. Lệnh này được sử dụng khi xảy ra lỗi, sai sót nghiêm trọng, việc tiếp tục chương trình là không có ý nghĩa.

Cú pháp: Halt;

Ví dụ 6.18:

```
Program  vidu_6_18;
var    F,x,y:real;
begin
    writeln(' Nhap x = ');readln(x);
    writeln(' Nhap y = ');readln(y);
    if y <>0 then    F:=x/y
    Else halt; {Chương trình sẽ dừng lại ngay khi thay y =0}
    writeln(' F = ',F:8:3);
    readln;
end.
```

BÀI TẬP CHƯƠNG 6

Bài 6.1: Nhập từ bàn phím 3 số a,b,c. Kiểm tra xem a,b,c có phải là 3 cạnh của tam giác hay không.

- Nếu thỏa mãn hãy tính chu vi, diện tích của tam giác và hiển thị kết quả lên màn hình
- Nếu không thỏa mãn in dòng thông báo: a,b,c không là ba cạnh của tam giác

Bài 6.2: Nhập từ bàn phím 3 số a, b, c cho đến khi thỏa mãn là độ dài 3 cạnh của tam giác ABC rồi tính đường tròn ngoại tiếp tam giác theo công thức:

$$R = \frac{abc}{4S} \quad \text{với } S \text{ là diện tích tam giác được tính theo công thức Hêrông:}$$

$$S = \sqrt{p(p-a)(p-b)(p-c)} \quad \text{trong đó } p \text{ là nửa chu vi}$$

Bài 6.3: Lập chương trình giải phương trình bậc 2: $ax^2+bx+c=0$ ($a \neq 0$)

Bài 6.4: Nhập từ bàn phím 3 số a,b,c. Tìm số lớn nhất và nhỏ nhất trong 3 số đó.

Bài 6.5: Viết chương trình in ra thông báo gõ phím loại nào: Dấu phép toán +, -, *, /, <, >, =; dấu chính tả: dấu chấm (.), dấu phẩy (,), chấm phẩy (;), chấm than (!), dấu hỏi (?), dấu ba chấm (...); các chữ số: 0, 1, ..., 9; các chữ cái thường: a, b, ..., z; các chữ cái in hoa: A, B, ..., Z; hay các ký hiệu khác.

Bài 6.6: Nhập từ bàn phím 2 số a,b và tính giá trị của biểu thức: $F(x)=4x^2+5x+1$

$$x = \begin{cases} \frac{a+b}{2} & \text{Nếu } a > b \\ 152.48 & \text{Nếu } a = b \\ \frac{a+b}{b^2} & \text{Nếu } a < b \end{cases}$$

Bài 6.7: Nhập vào 1 số nguyên n ($n < 20$). In ra màn hình n dòng, mỗi dòng n ký tự *.

Bài 6.8: Nhập từ bàn phím trị n nguyên, dương, sau đó tính tổng S và in kết quả ra màn hình

- a) $S = 1 - 1/2 + 1/3 - 1/4 + \dots + 1/n$
- b) $S = 1 + 3 + 5 + \dots + (2n+1)$

Bài 6.9: Nhập n từ bàn phím và tính n!. In kết quả lên màn hình.

Bài 6.10: Nhập x,n từ bàn phím rồi tính:

- a) $S := 1 + x + 2x + \dots + nx$;
- b) $S := 1 + x + x^2 + \dots + x^n$
- c) $S = 1 + x + x^2/2! + x^3/3! + \dots + x^n/n!$
- d) $S = \sin x / \cos x + \sin 2x / \cos 2x + \dots + \sin nx / \cos nx$.

Bài 6.11: Tìm 3 số nguyên dương a,b,c thỏa mãn: $a^2 + b^2 + c^2 = abc$

Bài 6.12: Tìm số m lớn nhất thỏa mãn bất đẳng thức: $m^3 + m^2 + m + 1 < 1999$

Bài 6.13: Nhập n nguyên dương từ bàn phím và đếm các số lẻ trong đoạn từ 1 tới n .

Bài 6.14: Nhập n nguyên dương từ bàn phím và tính tích các số chẵn trong đoạn từ 1 tới n .

Bài 6.15: In lên màn hình tam giác vuông, tam giác cân bằng các dấu *, biết chiều cao của tam giác là h được nhập vào từ bàn phím.

Bài 6.16: Kiểm tra việc nhập từ bàn phím một ngày tháng hợp lệ. Yêu cầu gõ lại cho đến khi đúng.

Bài 6.17: Viết chương trình lập công việc sau cho đến khi nhấn phím ESC.

Tính cước điện thoại biết đầu vào thời gian bắt đầu cuộc gọi là T_s và độ dài của cuộc gọi T (phút). Có giảm giá 1/3 nếu ngoài giờ làm việc (trước 8h00 và sau 16h00), giảm giá một nửa nếu từ 23h00 đến 6h00 sáng. Giá chuẩn là x đồng/1 phút.

Bài 6.18: Viết chương trình nhập vào một số và kiểm tra xem số đó có là số nguyên tố hay không.

Chương 7

DỮ LIỆU CÓ CẤU TRÚC

7.1. Kiểu mảng

7.1.1 Khái niệm

Chúng ta đã làm quen với các kiểu dữ liệu đơn giản là các kiểu dữ liệu vô hướng (integer, real, char, string, boolean). Trong Pascal tồn tại các kiểu dữ liệu có cấu trúc, chúng được tạo từ các kiểu dữ liệu đơn giản theo một qui tắc nào đó. Dữ liệu có cấu trúc được đặc trưng bằng kiểu dữ liệu của các phần tử, phương pháp cấu thành kiểu dữ liệu.

Khi một số hữu hạn các phần tử có cùng kiểu, được đặt tên chung và phân biệt với nhau nhờ thứ tự cách sắp xếp thì ta gọi là dữ liệu có cấu trúc kiểu mảng (array). Số phần tử của mảng được xác định ngay từ khi định nghĩa ra mảng, các phần tử của mảng được truy xuất thông qua tên mảng và chỉ số được để giữa 2 ngoặc vuông [].

Trong thực tế, mảng có ứng dụng rất lớn nhất là đối với các bài toán kỹ thuật, như lưu trữ một dãy tài liệu của các trạm đo, lưu giá trị các phần tử trên các nút mạng trong bài toán sai phân...

*** Kiểu dữ liệu kiểu mảng có những đặc trưng sau:**

- Các phần tử mảng phải cùng kiểu dữ liệu với nhau.
- Mỗi phần tử trong mảng được gắn với một chỉ số, đó chính là vị trí của phần tử trong mảng hay có thể truy cập trực tiếp tới từng phần tử của mảng thông qua biến mảng và chỉ số.
- Các phần tử trong mảng được sắp thứ tự trước sau thông qua chỉ số của nó
 - + mảng một chiều: $A[i]$ đứng trước $A[j]$ nếu $i < j$
 - + mảng hai chiều: $A[i_1, j_1]$ đứng trước $A[i_2, j_2]$ nếu $i_1 < i_2$
hoặc nếu $i_1 = i_2$ và $j_1 < j_2$.
- Các chỉ số có của các phần tử trong cùng một mảng phải cùng kiểu dữ liệu với nhau gọi chung là *kiểu chỉ số*, kiểu chỉ số phải là kiểu hữu hạn đếm được.

Dưới đây sẽ trình bày về khai báo mảng một chiều và mảng hai chiều:

7.1.2 Khai báo mảng một chiều

Có thể khai báo mảng bằng hai cách: khai báo trực tiếp và khai báo thông qua định nghĩa kiểu TYPE.

- Khai báo trực tiếp:

VAR Tên_biến_mảng: ARRAY[*kiểu chỉ số*] OF *kiểu phần tử*;

Trong đó:

- *kiểu chỉ số*: là cách tổ chức các phần tử của mảng, cách truy nhập vào các phần tử của mảng, nó có thể là các kiểu dữ liệu đơn giản vô hướng đếm được, hữu hạn giá trị.
- *kiểu phần tử*: là kiểu dữ liệu của các phần tử của mảng, có thể là bất kỳ kiểu dữ liệu nào.

Ví dụ 7.1: Khai báo mảng trực tiếp:

VAR X : array[1..15] of integer;

```
HT: array[1..50] of string[30];
```

Trong khai báo trên:

- X là mảng một chiều có thể được xem như là một dãy số có tối đa 15 phần tử, các phần tử được đánh số từ 1 đến 15 và có cùng kiểu nguyên.

- HT là mảng một chiều có thể được xem như là một dãy có tối đa 50 phần tử, các phần tử được đánh số từ 1 đến 15 và có cùng kiểu xâu có tối đa 30 ký tự.

- Khai báo gián tiếp:

```
TYPE Tên_kiểu_mảng = ARRAY[kiểu chỉ số] OF kiểu phần tử;
```

```
VAR Tên_biến_mảng : Tên_kiểu_mảng;
```

Trong các khai báo trên, chỉ số có thể là kiểu miền con, hoặc kiểu liệt kê vô hướng

Ví dụ 7.2: Khai báo mảng gián tiếp: khai báo mảng trực tiếp như trong ví dụ 5.1 tương đương với cách khai báo gián tiếp như sau:

```
TYPE      M1 = array[1..15] of integer;
          M2 = array[1..50] of string[30];
VAR       X  : M1;
          HT : M2;
```

- Truy cập đến từng phần tử của mảng 1 chiều theo cú pháp:

Tên_biến_mảng[chỉ_số]

Ví dụ 7.3: với khai báo mảng trong ví dụ 7.2

M1[1] là chỉ phần tử thứ nhất trong mảng M1

M1[i] là chỉ phần tử thứ i trong mảng M1

7.1.3. Khai báo mảng hai chiều

- **Khai báo trực tiếp:**

```
VAR Tên_biến_mảng : ARRAY[chỉ_số_hàng, chỉ_số_cột] OF kiểu phần tử;
```

Trong đó:

- *chỉ_số_hàng*, *chỉ_số_cột*: là cách tổ chức các phần tử của mảng, cách truy nhập vào các phần tử của mảng, nó có thể là các kiểu dữ liệu đơn giản vô hướng đếm được, hữu hạn giá trị.

- *kiểu phần tử*: là kiểu dữ liệu của các phần tử của mảng, có thể là bất kỳ kiểu dữ liệu nào.

Ví dụ 7.4: Khai báo mảng trực tiếp:

```
VAR X : array[1..15, 1..5] of integer;
    HT: array[1..50, 1..50] of string[30];
```

Trong khai báo trên:

- X là mảng hai chiều có thể được xem như là một ma trận có tối đa 75 (tối đa là 15 hàng và tối đa 5 cột), các phần tử có kiểu nguyên.

- HT là mảng hai chiều có thể được xem như là một ma trận có tối đa 50 hàng và 50 cột, các phần tử có kiểu xâu có tối đa 30 ký tự.

- **Khai báo gián tiếp:**

TYPE Tên_kiểu_mảng = ARRAY[*chỉ_số_hàng*, *chỉ_số_cột*] OF *kiểu_phần_tử*;
VAR Tên_biến_mảng : Tên_kiểu_mảng;

* **Chú ý:** Trong các khai báo trên, chỉ số có thể là kiểu miền con, hoặc kiểu liệt kê vô hướng

Ví dụ 7.5: Khai báo mảng gián tiếp: khai báo mảng trực tiếp như trong ví dụ 5.1 tương đương với cách khai báo gián tiếp như sau:

```
TYPE      M1 = array[1..15,1..5] of integer;  
          M2 = array[1..50,1..50] of string[30];  
VAR X : M1;  
    HT: M2;
```

- Truy cập đến từng phần tử của mảng 2 chiều theo cú pháp:

Tên_biến_mảng[chỉ_số_hàng, chỉ_số_cột]

Ví dụ 7.6: với khai báo mảng trong ví dụ 7.5

M1[1,1] là chỉ phần tử ở hàng 1, cột 1 trong mảng M1

M1[i,j] là chỉ phần tử ở hàng i, cột j trong mảng M1

* **Chú ý:** Thực chất mảng hai chiều là mảng một chiều mà các phần tử của nó là một mảng một chiều.

7.1.4. Các phép toán trên mảng

- **Phép gán:**

- Có thể thực hiện gán hai biến mảng cho nhau nếu chúng cùng kiểu dữ liệu với nhau.

Ví dụ 7.7: Với khai báo mảng sau:

```
TYPE dayso = array[1..50] of integer;  
VAR a,b:dayso; {a và b được gọi là cùng kiểu dữ liệu với nhau}  
    c,d: array[1..50] of integer;  
{c và d cùng kiểu dữ liệu với nhau nhưng không cùng kiểu với a và b}
```

Ta có thể thực hiện các phép gán sau:

a:=b; b:=a; c:=d; d:=c;

- Các phần tử trong một mảng sẽ có các phép toán của kiểu dữ liệu của nó.

Ví dụ 7.8: Với khai báo mảng ở ví dụ 7.7 ta có

Các phần tử a[i] có kiểu dữ liệu là kiểu integer nên nó có mọi tính chất như một biến có kiểu integer.

- **Phép so sánh**

- Không được sử dụng bất kỳ phép so sánh nào với biến có kiểu dữ liệu là kiểu mảng.
- Nếu kiểu phần tử của mảng có thực hiện được các phép so sánh thì có thể thực hiện các phép so sánh giữa các phần tử có cùng kiểu dữ liệu với nhau.

7.1.5. Nhập và in dữ liệu của mảng

7.1.5.1. Nhập dữ liệu cho mảng

Để nhập dữ liệu cho mảng ta cũng sử dụng câu lệnh *read* hay *readln*, tuy nhiên không được phép sử dụng các lệnh nhập dữ liệu cho biến mảng mà chỉ sử dụng cho từng phần tử trong mảng. Do vậy **để nhập dữ liệu cho mảng ta phải nhập dữ liệu cho từng phần tử trong mảng.**

- Nhập dữ liệu cho mảng một chiều ta sử dụng một vòng FOR như sau:

Ví dụ cho mảng A được khai báo như sau:

```
Var A:array[1..100] of integer;
```

Nhập dữ liệu cho mảng A như sau:

```
For i:=1 to 100 do  
    Readln(A[i]);
```

Sử dụng vòng lặp FOR để duyệt hết mảng A, ứng với mỗi giá trị của i ta thực hiện nhập dữ liệu cho phần tử ở vị trí thứ i trong mảng A.

- Nhập dữ liệu cho mảng hai chiều ta sử dụng hai vòng FOR lồng nhau như sau:

Ví dụ cho mảng A được khai báo như sau:

```
Var A:array[1..50,1..50] of integer;
```

Nhập dữ liệu cho mảng A như sau:

```
For i:=1 to 50 do  
    For j:=1 to 50 do  
        Readln(A[i,j]);
```

Sử dụng hai vòng lặp FOR để duyệt hết mảng A, thực hiện duyệt hết theo từng hàng một, ứng với mỗi giá trị của i ta thực hiện nhập dữ liệu cho phần tử trên hàng i, ứng với mỗi giá trị i và j ta thực hiện nhập dữ liệu cho phần tử ở vị trí hàng i, cột j trong mảng A.

7.1.5.1. In dữ liệu mảng lên màn hình

Để in dữ liệu mảng ta cũng sử dụng câu lệnh *write* hay *writeln*, tuy nhiên không được phép sử dụng các lệnh in dữ liệu cho biến mảng mà chỉ sử dụng cho từng phần tử trong mảng. Do vậy **để in dữ liệu mảng lên màn hình ta phải in dữ liệu từng phần tử trong mảng.**

- In dữ liệu mảng một chiều ta sử dụng một vòng FOR như sau:

Ví dụ cho mảng A được khai báo như sau:

```
Var A:array[1..100] of integer;
```

In dữ liệu mảng A như sau:

```
For i:=1 to 100 do  
    write(A[i]:5);
```


Sử dụng vòng lặp FOR để duyệt hết mảng A, ứng với mỗi giá trị của i ta thực hiện in dữ liệu phần tử ở vị trí thứ i trong mảng A.

- In dữ liệu mảng hai chiều ta sử dụng hai vòng FOR lồng nhau như sau:

Ví dụ cho mảng A được khai báo như sau:

```
Var A:array[1..50,1..50] of integer;
```

In dữ liệu mảng A theo dạng bảng như sau:

```
For i:=1 to 50 do
  begin
    For j:=1 to 50 do
      write(A[i,j]);
      writeln; {xuống dòng}
    end;
```

Sử dụng hai vòng lặp FOR để duyệt hết mảng A, thực hiện duyệt hết theo từng hàng một, ứng với mỗi giá trị của i ta thực hiện in dữ liệu từng phần tử trên hàng i, in hết một hàng sẽ xuống dòng để in tiếp dòng tiếp theo, ứng với mỗi giá trị i và j ta thực hiện in dữ liệu phần tử ở vị trí hàng i, cột j lên màn hình.

Ví dụ 7.9: Viết chương trình nhập vào một dãy số có n phần tử nguyên ($1 \leq n \leq 100$).

- In dãy số đó lên màn hình theo hàng.
- Tính tổng và trung bình cộng các phần tử trong dãy số đó. In kết quả lên màn hình.

Program vidu_7_9;

```
Var A: array[1..100] of integer;
```

```
S, i, n: integer;
```

```
TB:real;
```

```
Begin
```

```
Write(' Nhập số phần tử của dãy số n = '); readln(n);
```

```
Writeln(' Nhập từng phần tử trong dãy số :');
```

```
For i:=1 to n do
```

```
  Begin
```

```
    Write('a[',i,',']= '); readln(a[i]);
```

```
  End;
```

```
Writeln(' In dãy số vừa nhập vào là ');
```

```
For i:=1 to n do
```

```
  Write(a[i]:5);
```

```
{Tính tổng và trung bình cộng}
```

```
S:=0;
```

```
For i:=1 to n do
```

```
  S:=S+a[i];
```

```
TB:=S/n;
```

```
Writeln(' Tổng các phần tử dãy số S = ',S);
```

```
Writeln(' Trung bình cộng TB = ',TB:8:2);
```

```
Readln;
```

```
End.
```

Ví dụ 7.10: Viết chương trình nhập vào một ma trận nguyên cỡ $m \times n$ ($1 \leq n, m \leq 20$).

- In ma trận đó lên màn hình theo bảng.
- Tính tổng các phần tử âm của ma trận.

```
Program vidu_7_10;
Var  A: array[1..20,1..20] of  integer;
     S, i, n: integer;
Begin
    Write(' Nhập số hàng m = '); readln(m);
    Write(' Nhập số cột      n = '); readln(n);
    Writeln(' Nhập từng phần tử trong ma trận :');
    For i:=1 to m do
        For j:=1 to n do
            Begin
                Write('a[',i,',',j,']= '); readln(a[i,j]);
            End;
        Writeln(' In ma tran vua nhap theo dang bang: ');
        For i:=1 to m do
            Begin
                For j:=1 to n do
                    Write(a[i,j]:5);
                Writeln;
            End;
        {Tính tổng các số âm trong ma trận}
        S:=0;
        For i:=1 to m do
            For j:=1 to n do
                If a[i,j]<0 then
                    S:=S+a[i,j];
        Writeln(' Tổng các phần tử âm là S = ',S);
        Readln;
    End.
```

7.1.6 Một số bài toán cơ bản về mảng

7.1.6.1. Bài toán tìm kiếm trên mảng:

Tìm kiếm là một trong những bài toán cơ sở trong xử lý thông tin. Có nhiều bài toán tìm kiếm, tuy nhiên có thể quy về tìm kiếm trên mảng như sau: Cho mảng và phần tử x có kiểu dữ liệu cùng với kiểu dữ liệu của phần tử mảng. Hãy tìm xem có phần tử mảng nào có giá trị bằng x hay không.

Để giải quyết bài toán dạng này có thể sử dụng phương pháp tìm kiếm tuần tự hay tìm kiếm nhị phân.

Thủ tục tìm kiếm tuần tự là duyệt lần lượt các phần tử mảng, so sánh giá trị của nó với x. Việc tìm kiếm dừng khi tìm thấy hoặc đã hết mảng, tức là phát hiện ra là mảng không có giá trị khóa nào bằng x.

Có thể triển khai thủ tục tìm kiếm trên mảng bằng một vòng lặp while, dùng cờ báo là biến **found** kiểu Boolean.

Ví dụ 7.11: Viết chương trình nhập vào một dãy số thực có n phần tử và một số thực x. Kiểm tra xem x có xuất hiện trong dãy số đó không?

```
Program vidu_7_11;
Var  A:array[1..100] of real;
     i,n:integer;
     x:real;
     found:boolean;
Begin
    write(' Nhập số phần tử của dãy số n = ');readln(n);
    writeln(' Nhập từng phần tử của dãy số:');
    For i:=1 to n do
        begin
            write('A[' ,i, ']= ');readln(A[i]);
        end;
    writeln('In dãy số vừa nhập:');
    For i:=1 to n do
        write(A[i]:7:2);
        {Tim x trong dãy số A}
    writeln;
    write(' Nhập số cần tìm x = ');readln(x);
    found:=false;
    For i:=1 to n do
        if A[i]=x then
            begin
                found:=true;
                break; {dừng vòng lặp lại}
            end;
    if found then
        writeln(x:5:2,' có xuất hiện trong dãy số')
    else  writeln(x:5:2,' không xuất hiện trong dãy số');
    readln;
end.
```

7.1.6.2. Bài toán sắp xếp dãy số tăng (giảm)

Sắp xếp cũng là một trong những bài toán cơ sở trong xử lý thông tin. Từ dãy dữ liệu ban đầu chưa có thứ tự, cần tiến hành sắp xếp để nhận được dãy kết quả có thứ tự tăng hoặc giảm dần theo một khóa cho trước nào đó. Bài toán quy về sắp xếp một mảng số nguyên.

Dưới đây sẽ trình bày phương pháp sắp xếp chọn trực tiếp hay còn được gọi là phương pháp nổi bọt. Không mất tính tổng quát ta xét bài toán sắp xếp tăng, cụ thể như sau:

Mô tả

- Chọn phần tử nhỏ nhất trong dãy nguồn, xếp nó vào vị trí đầu tiên trong dãy đích. Đây là phần tử thứ nhất và cũng là phần tử cuối cùng của dãy đích.
- Chọn phần tử nhỏ nhất trong dãy nguồn còn lại, đây là phần tử nhỏ thứ hai, xếp nó vào vị trí thứ hai và cũng là vị trí cuối của dãy đích vào lúc này.
- Lặp lại việc này cho đến khi hết dãy nguồn.

Để tiết kiệm chỗ, ta cũng xếp dãy đích và dãy nguồn liền nhau trong mảng. Do đó, ở bước thứ i , thao tác "xếp vào cuối dãy đích" chính là "đổi chỗ cho $a[i]$ ".

Mình họa:

Giả sử cần sắp xếp dãy số sau:

44 55 12 42 94 18 06 67

Sự thay đổi của dãy số qua từng bước sắp xếp như sau:

Xuất phát: 44 55 12 42 94 18 06 67

Bước 1: 06 / 55 12 42 94 18 44 67

Bước 2: 06 12 / 55 42 94 18 44 67

Bước 3: 06 12 18 / 42 94 55 44 67

Bước 4: 06 12 18 42 94 55 44 67

Bước 5: 06 12 18 42 44 / 55 94 67

Bước 6: 06 12 18 42 44 55 / 94 67

Bước 7: 06 12 18 42 44 55 67 / 94

Bước 8: 06 12 18 42 44 55 67 94

Thủ tục chi tiết

```

Var i,j,k: integer;
    x: real;
Begin
    for i:=1 to n-1 do
        for j:=i+1 to n do
            if a[j]<a[i] then
                "đổi chỗ a[i] và a[j] cho nhau"
End;
```

Ví dụ 7.12: Viết chương trình nhập vào một dãy số thực có n phần tử ($1 \leq n \leq 100$). Sắp xếp dãy số theo chiều tăng dần. In lên màn hình dãy số trước và sau khi sắp xếp.

```
Program vidu_7_12;
Type kieuDayso = array[1..100] of real;
Var A: kieuDayso;
    i, j, n: integer;
    tg: real;
Begin
    Write('Nhap so phan tu cua day so n = ');readln(n);
    writeln('Nhap tung phan tu cua day so:');
    For i:=1 to n do
        begin
            write('A[' ,i, ']= ');readln(A[i]);
        end;
    writeln('In day so vua nhap:');
    For i:=1 to n do
        write(A[i]:7:2);
    { Sắp xếp dãy số}
    For i:=1 to n-1 do
        For j:=i+1 to n do
            If A[i]< A[j] then
                Begin {hoán vị A[i] và A[j] cho nhau}
                    tg:=A[i];
                    A[i]:=A[j];
                    A[j]:=tg;
                End;
    Writeln(' Day so sau khi sap xep la: ');
    For i:=1 to n do
        write(A[i]:7:2);
    readln;
End.
```

7.1.7. Một số ví dụ khác

• **Ví dụ 7.13:** Viết đoạn chương trình chèn một số vào dãy đã sắp tăng mà không thay đổi thứ tự sắp xếp của dãy:

```
Write('Nhap so can chen k =');
Readln(k);
i:=1;
While (k>a[i]) and (i<=n) do i:=i+1;
Writeln('Vi tri chen i=',i:2);
If i>n then
    Begin
```

```
        a[i]:=k; n:=n+1;
    End
Else
    Begin
        For j:=n downto i do
            a[j+1]:=a[j];
        a[i]:=k;
        n:=n+1;
    End;
```

• **Ví dụ 7.14:** Viết chương trình tính tổng các phần tử của ma trận, nhập vào hàng k và tính trung bình cộng các phần tử trên hàng k. Hiển thị ma trận nhập dạng bảng và các kết quả tính:

```
Program Vidu14;
Uses Crt;
Var    i,j,m,n,k :integer;
      a: array[1..10,1..10] of real;
      S,Sk,TBC   :real;
BEGIN
    Clrscr;
    Write('Cho so hang cua ma tran m= ');Readln(m);
    Write('Cho so cot cua ma tran  n= ');Readln(n);
    Writeln('Nhap tri cac phan tu cua ma tran:');
    For i:=1 to m do
        For j:=1 to n do
            Begin
                Write('a[' ,i ,',' ,j ,']= ');
                Readln(a[i,j]);
            End;
        Writeln('Ma tran nhap:');
        For i:=1 to m do
            Begin
                For j:=1 to n do
                    Write(a[i,j]:7:2);
                Writeln;
            End;
        S:=0;
        For i:=1 to m do
            For j:=1 to n do
                S:=S+a[i,j];
            Writeln('Tong cac pt cua ma tran  S= ',S:6:2);
```

```
Write('Nhap hang k :');Readln(k);
Sk:=0;
For j:=1 to n do
    Sk:=Sk+a[k,j];
TBC:=Sk/n;
Writeln('TBC cac phan tu hang',k:2,' la: ',TBC:6:2);
Readln;
END.
```

7.2. Kiểu chuỗi (xâu) ký tự

7.2.1 Khái niệm

- Xâu ký tự** là một kiểu dữ liệu nhận giá trị là dãy các ký tự trong bảng mã ASCII, dãy ký tự này được đặt trong một cặp dấu nháy đơn (' ')

Ví dụ 7.15:

```
St1 = 'NGON NGU PASCAL'
St2 = 'Tin hoc nam 2002'
St3 = '123456'
St4 = 'Truong Dai hoc Dien Luc'
```

- Độ dài của một xâu** là số ký tự của xâu đó. Xét ví dụ 7.20:

```
độ dài của st1 là 15,
độ dài của st2 là 16,
độ dài của st3 là 6,
độ dài của st4 là 23,
```

- Xâu rỗng** là xâu không chứa ký tự nào, nó có độ dài là 0, được ký hiệu là ''. Xâu rỗng khác với xâu chứa ký tự trống ' '. Xâu chỉ chứa 1 ký tự trống có độ dài là 1.

- Trong bộ nhớ của máy, một biến xâu sẽ chiếm một số byte bằng độ dài tối đa của nó cộng thêm 1. Byte đầu tiên, gọi là byte 0, chứa một ký tự có mã bằng độ dài thực của xâu, mỗi byte còn lại chứa một ký tự. Cấu trúc của biến St1 nói trên có dạng:

Xâu ký tự		N	G	O	N		N	G	U		P	A	S	C	A	L
Số thứ tự	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Độ dài N (=15) của biến St1 và ký tự trong byte 0 (ký hiệu là St[0]) liên quan với nhau như sau:

```
N = Ord ( St[0] )
St[0]= Chr( N )
```

7.2.2. Khai báo xâu ký tự

7.2.2.1. Khai báo kiểu dữ liệu xâu:

TYPE Tên_kiểu_xâu = string; {mặc định là xâu dài tối đa 256 ký tự}

Hoặc

TYPE Tên_kiểu_xâu = string[độ dài tối đa của xâu];

VAR biến_xâu : Tên_kiểu_xâu;

Trong đó độ dài tối đa của xâu là một số nguyên dương nằm trong đoạn [0..256].

Ví dụ 7.16: Khai báo kiểu họ tên là kiểu xâu ký tự có độ dài tối đa là 40 ký tự.

```
TYPE kieu_hoten = string[40];
```

```
VAR ht: kieu_hoten; {khai báo biến ht có kiểu là kieu_hoten}
```

***Chú ý:** Khi một biến xâu được dùng làm đối số của hàm hay thủ tục thì nó cần phải được khai báo theo cách này (trừ các biến xâu có kiểu String).

7.2.2.2. Khai báo biến kiểu dữ liệu xâu:

VAR tên_biến_xâu: string;

Hoặc

VAR tên_biến_xâu: string[độ dài tối đa của xâu];

Ví dụ 7.17:

```
VAR st1: string; {khai báo biến xâu st mặc định có độ dài tối đa là 256}
```

```
st2: string[50]; {khai báo biến xâu st có độ dài tối đa là 50}
```

*** Chú ý:** Nếu xâu st có độ dài tối đa là N mà ta lại gán cho nó xâu có độ dài lớn hơn N thì xâu st chỉ nhận N ký tự đầu tiên.

Ví dụ 7.18:

```
- Var st: string[15];
```

Sau lệnh gán st:=’Truong Dai hoc Dien Luc’; thì st sẽ nhận giá trị là: ‘Truong Dai hoc ’

```
- Var st1: string[50];
```

```
st1 := ‘C5CNTD’
```

*** Chú ý:** Cần phân biệt độ dài với độ dài tối đa của biến xâu: độ dài tối đa được xác định ngay khi khai báo là khả năng có thể chứa của biến xâu, còn độ dài của xâu là số ký tự đang thực có trong xâu. Như ví dụ 7.23, xâu st1 có độ dài tối đa là 50 nhưng độ dài thực tế của xâu là 6.

7.2.3. Viết ra và đọc vào một xâu ký tự

• Viết ra

Có thể in một xâu ký tự ra màn hình, máy in hay tệp văn bản bằng các lệnh *write*

Ví dụ 7.19:

```
- Writeln(st); in ra nguyên vẹn xâu st lên màn hình,
```

```
- Writeln(‘Turbo Pascal’); in lên màn hình dòng chữ ‘Turbo Pascal’, viết xong con trỏ ở đầu dòng tiếp theo.
```


- Write('Turbo Pascal'); in lên màn hình dòng chữ 'Turbo Pascal', viết xong con trỏ ở cuối dòng vừa in.

- **Đọc vào**

Có thể đọc vào một giá trị cho một biến kiểu xâu bằng các lệnh *read*, xem lại sử dụng các lệnh *read* trong phần nhập xuất dữ liệu.

Ví dụ 7.20:

```
- readln(st) ; hay read(st) ;    nhập dữ liệu cho xâu st
- readln(st1, st2) ;            nhập dữ liệu cho biến xâu st1 và st2
```

7.2.4. Các phép toán trên xâu

7.2.4.1. Phép gán

Cũng tương tự như các kiểu dữ liệu đơn giản đã học, ta hoàn toàn có thể thực hiện gán giá trị hai biến xâu cho nhau.

Ví dụ 7.21: với khai báo

```
Var  st1, st2: string[10];
      st:string;
```

Thì có thể thực hiện các lệnh gán:

```
(1)    St1:=st2; St2:=st1; st:=st1; st1:= st;
(2)    St1 := '123456' ...
(3)    St2 := '1234567890abcdef' ;
```

* **Chú ý:** khi thực hiện phép gán xâu st2 có độ dài L2 cho biến xâu st1 có độ dài L1 mà L2>L1 thì st1 chỉ nhận L1 ký tự đầu tiên của xâu st2.

Như ví dụ 7.28, sau lệnh gán (3) thì st2 = '1234567890'.

7.2.4.2. Phép so sánh

Có thể thực hiện các phép toán so sánh với kiểu xâu ký tự.

Quy tắc so sánh:

- Thực hiện so sánh giữa các cặp ký tự tương ứng từ trái qua phải. Khi phát hiện có một cặp ký tự khác nhau thì xâu nào chứa ký tự nhỏ hơn sẽ nhỏ hơn.
- Nếu nội dung của hai xâu giống nhau từ đầu đến hết chiều dài của xâu ngắn hơn thì xâu ngắn hơn là nhỏ hơn

Ví dụ 7.22: 'hoa' > 'Hoa'

'Pascal' > 'Pas'

'hai' < 'hao'

'Hoa'> 'Hao'

- Hai xâu bằng nhau nếu chúng dài bằng nhau và mọi cặp ký tự ở các vị trí tương ứng đều giống nhau.

Ví dụ 7.23:

Biểu thức 'Pascal' = 'Pascal' cho kết quả là đúng

Biểu thức 'Pascal' = 'Pascal' cho kết quả là sai.

Ví dụ 7.24: Viết chương trình nhập vào hai chuỗi ký tự st1 và st2.

- In lên màn hình hai chuỗi, mỗi chuỗi một dòng.
- Cho biết chuỗi nào có giá trị lớn hơn, in chuỗi đó lên màn hình.

```
Program vidu_7_24;  
Var st1,st2 : string;  
Begin  
    Write(' Nhập vào chuỗi st1 = ');  
    Readln(st1); {nhập dữ liệu cho chuỗi st1}  
    Write(' Nhập vào chuỗi st2 = ');  
    Readln(st2); {nhập dữ liệu cho chuỗi st2}  
    {So sánh hai chuỗi}  
    If st1>st2 then  
        Writeln(' Chuỗi lớn hơn là chuỗi ', st1)  
    Else  
        if st1<st2 then  
            Writeln(' Chuỗi lớn hơn là chuỗi ', st2)  
        else writeln(' hai chuỗi bằng nhau');  
    Readln;  
End.
```

7.2.4.3. Phép cộng chuỗi

Phép toán cộng (+) hai chuỗi st1 và st2 (ký hiệu là st1+st2) là phép nối chuỗi st2 vào ngay sau chuỗi st1.

Ví dụ 7.25: st1 := 'Truong';
st2 := 'Dai hoc';
st3 := st1 + st2; thì st3 = 'Truong Dai hoc';
st4 := st3 + ' Dien luc';
thì st4 = 'Truong Dai hoc Dien luc';
st5 := 'Turbo' + ' Pascal';
thì st5 = 'Turbo Pascal';

7.2.5 Truy nhập vào từng phần tử của chuỗi

Giống như mảng, mỗi phần tử của chuỗi được truy nhập thông qua tên chuỗi và chỉ số của phần tử.

Gọi N=Length(St), khi đó ký tự thứ i (i=1, 2, ..., N) của St được ký hiệu là St[i].

Ví dụ 7.26: cho: St := 'ABC';

thì $N=3$ và $St[1]='A'$, $St[2]='B'$, $St[3]='C'$.

Lệnh $St[1]:='a'$; sẽ biến đổi St thành $St='aBC'$.

Như vậy mỗi ký tự $St[i]$ được dùng như một biến kiểu ký tự, và xâu có thể xem là một mảng các ký tự. Chẳng hạn để in xâu ta có thể in từng ký tự như sau:

```
For i:=1 to Length(St) do write(St[i]);
```

Điều này cho thấy xâu là một kiểu dữ liệu có tính cấu trúc.

Nhưng mặt khác, mỗi xâu lại có thể xem là một giá trị duy nhất, vì có thể nhập và in xâu trực tiếp bằng các lệnh:

```
Readln(St);
```

```
Write(St);
```

Đặc điểm này cho thấy xâu còn là một kiểu dữ liệu có tính đơn giản.

7.2.6 Các hàm xử lý xâu ký tự

- **Hàm Length(St):** cho kết quả là một số nguyên bằng độ dài của xâu St.

Ví dụ 7.27: $Length('ABCD')=4$ vì xâu 'ABCD' có 4 ký tự.

Xâu rỗng có độ dài bằng 0.

- **Hàm Pos(S, St):** Cho vị trí đầu tiên tìm thấy xâu S trong xâu St, nếu không tìm thấy thì hàm cho kết quả bằng 0.

Ví dụ 7.28:

$Pos('Ab', 'cdAb3Abm') = 3,$

$Pos('Ab', '1bA3b') = 0.$

- **Hàm Copy(St, k, m) :** kết quả trả lại là một xâu gồm m ký tự liên tiếp của St tính từ vị trí k.

Ví dụ 7.29: $Copy('ABCDEF', 4, 2) = 'DE'$.

- Nếu $k > Length(St)$ thì kết quả sẽ là một xâu rỗng

- Nếu $m > \text{số ký tự đứng sau kể từ vị trí } k$ thì hàm Copy chỉ lấy các ký tự từ vị trí k đến hết chiều dài của St, ví dụ :

$Copy('ABCD', 3, 10) = 'CD'$

- **Hàm Concat(St1, St2, ..., Stn) :** Ghép nối các xâu St1, St2, ..., Stn theo thứ tự đó thành một xâu duy nhất. Vậy :

$Concat(St1, St2, ..., Stn) = St1+St2+...+Stn.$

Ví dụ 7.30: $st1:='C5'$; $st2:='Cong Nghe'$; $st3:='Tu Dong'$

$St := concat(st1,st2,st3);$

Thì $st = 'C5 Cong Nghe Tu Dong'$

7.2.7 Các thủ tục liên quan đến xâu

- **Thủ tục Delete(St, k, m) :** Xóa m ký tự trong biến xâu St bắt đầu từ vị trí thứ k. Ví dụ, sau khi thực hiện các lệnh:

Ví dụ 7.31: $St:=' TurboPascal';$

$Delete(St, 1, 5);$ {xóa đi 5 ký tự trong xâu st tính từ ký tự đầu tiên}

thì giá trị của $St = \text{'Pascal'}$ vì 5 ký tự đầu đã bị xóa.

- Nếu $k > \text{Length}(St)$ thì không xóa gì cả.

- Nếu $m > \text{số ký tự đứng sau kể từ vị trí } k$ thì xóa hết từ vị trí k đến cuối xâu.

Ví dụ 7.32: Sau khi thực hiện ba lệnh :

```
St := 'Turbo Pascal' ;  
Delete(St, 10, 20) ;  
Write(St) ;
```

thì in ra chữ Turbo Pas vì St đã bị xóa đi 3 ký tự cuối nên chỉ còn $St = \text{'Turbo Pas'}$.

• **Thủ tục Insert(S, St, k) :** Chèn xâu S vào biến xâu St tại vị trí k .

Ví dụ 7.33: $St = \text{'ABCD'}$;

Sau khi thực hiện lệnh:

```
Insert('**', St, 3);
```

thì St bị biến đổi thành $St = \text{'AB**CD'}$.

- Nếu $k > \text{Length}(St)$ thì S được nối vào cuối của St . Ví dụ, sau khi thực hiện hai lệnh:

```
St := 'XYZ';
```

```
Insert('ABC', St, 6);
```

thì $St = \text{'XYZABC'}$.

• **Thủ tục Str(x, St):** Biến đổi số nguyên hay thực x thành kiểu xâu và gán cho biến xâu St .

Ví dụ 7.34: Sau khi thực hiện lệnh :

```
Str(4752, St);
```

thì kết quả là $St = \text{'4752'}$.

Số x có thể được định dạng như khi in ra màn hình.

Lệnh $\text{Str}(4752 : 6, St)$;

cho kết quả $St = \text{' 4752'}$ (trước số 4752 có 2 ký tự trắng) .

- Nếu x là biến thực và giá trị của $x = 34.95$ thì lệnh :

```
Str(x : 7:3, St);
```

cho kết quả $St = \text{' 34.950'}$ (trước số 34.950 có 1 ký tự trắng) .

• **Thủ tục Val(St, x, k) :** Biến đổi xâu số St thành số nguyên hay thực và gán cho biến nguyên hay thực x . Số nguyên k dùng để phát hiện lỗi: nếu đổi được thì $k = 0$, ngược lại, giá trị của k là vị trí có lỗi trong xâu St .

Ví dụ 7.35: cho ba biến n, k, j kiểu nguyên và biến x kiểu thực, sau khi thực hiện các lệnh

```
St := '385' ;
```

```
Val(St, n, j) ;
```

```
Val('12.59', x, k) ;
```

thì $n = 385, j = 0, x = 12.59$ và $k = 0$.

Nếu gán $St := \text{'3a7'}$; và thực hiện lệnh: $\text{Val}(St, n, k)$;

thì giá trị của n không xác định còn $k = 2$ là vị trí của chữ a trong xâu St , tại đó không đổi ra số được.

7.2.8 Các ví dụ về xâu

Ví dụ 7.36: Đổi một xâu ra chữ hoa hay chữ thường.

Để đổi cả xâu St thành chữ hoa, ta đổi từng ký tự của xâu đó ra chữ hoa, tức là :

```
For i:=1 to Length(St) do
    St[i]:=Uppcase(St[i]);
```

Tương tự, để đổi cả xâu St thành chữ thường, ta cũng đổi từng ký tự của xâu St ra chữ thường:

```
For i:=1 to Length(St) do
    if ( St[i]>='A') and (St[i]<='Z') then
        St[i]:=Chr( Ord(St[i]) + 32) ;
```

Ví dụ 7.37: Chuẩn hóa một xâu ký tự .

Cho xâu St có nhiều ký tự trắng thừa ở đầu, ở cuối và giữa các từ, như St=' nguyên van tuan '. Chuẩn hóa xâu St là xóa hết các ký tự trắng thừa ở đầu và ở cuối, và giữa hai từ chỉ giữ lại đúng một ký tự trắng, như St='nguyên van tuan'.

a) Xóa các ký tự trắng ở đầu xâu :

Để xóa một ký tự trắng ở đầu của xâu St, ta dùng lệnh:

```
If St[1]=#32 then Delete(St,1,1);
```

Muốn xóa hết các ký tự trắng ở đầu xâu ta dùng lệnh:

```
While St[1]=#32 do Delete(St,1,1);
```

Diễn giải: chừng nào ký tự đầu tiên của St vẫn còn là ký tự trắng thì cứ xóa nó đi cho đến khi ký tự đầu tiên là khác trắng.

Sở dĩ phải dùng vòng lặp While là vì số ký tự trắng ở đầu xâu là không biết trước.

b) Xóa các ký tự trắng ở cuối xâu :

Tương tự, muốn xóa tất cả các ký tự trắng ở cuối của xâu St, ta dùng lệnh:

```
While St[ length(St) ]=#32 do Delete(St, length(St), 1);
```

Diễn giải: chừng nào ký tự cuối cùng của St còn là khoảng trắng thì cứ xóa nó đi cho đến khi ký tự cuối cùng là khác trắng.

c) Xóa các ký tự trắng thừa ở giữa hai từ trong xâu :

Muốn xóa các ký tự trắng thừa để giữa hai từ chỉ còn đúng một ký tự trắng ta làm như sau: tìm trong St chỗ nào có hai ký tự trắng thì xóa đi một, và lặp lại thao tác trên cho đến khi trong St không còn chỗ nào có hai ký tự trắng liên tiếp. Tức là :

```
k:=Pos(' ', St); { ' ' là 2 ký tự trắng }
While k > 0 do
    begin
        Delete(St, k, 1);
        k:=Pos(' ', St);
    end;
```

Ví dụ 7.38 : đếm trong xâu St có bao nhiêu chữ pascal.

Vì chữ pascal có 6 ký tự, nên ta so sánh từng cụm 6 ký tự của St với xâu pascal, bắt đầu từ vị trí 1:

```
Dem:=0;
For i:=1 to Length(St) do
  if Copy (St, i, 6) ='pascal' then
    Inc(Dem);
Writeln(' Số chữ pascal là ', Dem);
```

Ví dụ 7.39: Tìm kiếm và thay thế.

Tìm trong xâu St xem có chứa chữ 'basic' không, nếu có thì thay bằng chữ 'pascal', nếu không có thì in câu 'không có'. Ví dụ St='ngon ngu basic duoc dung pho bien', sau khi thay thế ta được St = 'ngon ngu pascal duoc dung pho bien'.

Ta dùng hàm Pos để tìm xem trong St có chứa chữ 'basic' không. Thủ tục Delete sẽ xóa xâu 'basic' khỏi St, và thủ tục Insert sẽ chèn xâu 'pascal' vào St tại vị trí đang xét:

```
k:= Pos('basic' , St);
while k> 0 do
  begin
    Delete(St, k, 5); { xóa chữ basic }
    Insert('pascal' , St, k); { chèn chữ pascal }
    k:= Pos('basic' , St);
  end;
```

Ví dụ 7.40: Tính tổng các bình phương của các chữ số của một số tự nhiên N.

Ví dụ N= 325 thì T=32+22+52 = 38.

```
Str(N,St); { Đổi số N ra xâu gán vào St }
T:=0;
For i:=1 to Length(St) do
  begin
    Val(St[i],j,k); {Đổi St[i] ra số gán vào j}
    T:=T+ j*j;
  end;
```

Ví dụ 7.41: Tách mỗi từ của xâu in riêng trên một dòng:

Cho St=' ngon ngu pascal ', cần in ra :

```
ngon
ngu
pascal
```

Phương pháp:

Bước 1: - Chuẩn hóa xâu St thành St='ngon ngu pascal'.

- Thêm một ký tự trắng vào cuối để St='ngon ngu pascal '.

Bước 2: -Tìm k là vị trí của ký tự trắng đầu tiên, in k-1 ký tự đầu tiên, đó chính là từ thứ nhất, xóa k ký tự đầu tiên, kết quả St='ngu pascal '.

Lặp lại quá trình trên cho đến khi trong St không còn ký tự trắng nào nữa.

Chương trình cụ thể như sau:

```
St:=St + #32; { thêm 1 ký tự trắng vào cuối St}
k:=Pos(#32, St);
While k>0 do
begin
    Tu:=Copy(St, 1, k-1);
    Writeln(Tu);
    Delete(St, 1, k);
    k:=Pos(#32, St);
end;
```

7.3. Kiểu bản ghi (Record)

7.3.1. Khái niệm

Các phần trình bày trước cho thấy ngôn ngữ Pascal rất mạnh trong việc giải quyết các bài toán thiên về tính toán. Trong phần này chúng ta sẽ thấy thêm một khả năng mạnh mẽ nữa của ngôn ngữ Pascal trong lĩnh vực quản lý: quản lý nhân sự, quản lý vật tư, quản lý tài chính, v.v. Hàng ngày chúng ta rất quen thuộc với một danh sách sinh viên gồm các cột: Số thứ tự, Họ và tên, ngày sinh, điểm học tập, ... ví dụ như bảng sau:

STT	Họ tên	Ngày sinh	Nơi sinh	Điểm TB
1	Đỗ Tuấn Anh	16/05/1988	Hà Nội	8.0
2	Nguyễn Mai Anh	12/12/1988	Hà Nội	7.8
3	Nguyễn Cường	27/06/1988	Quảng Ninh	9.3
4	Nguyễn Đức Dũng	15/11/1988	Thái Bình	7.3
..

Mỗi dòng liệt kê các dữ liệu về một người, mỗi cột là một dữ liệu thành phần cung cấp thông tin về một thuộc tính cụ thể của những người đó.

Trong ngôn ngữ Pascal, mỗi dòng được gọi là một RECORD (dịch là bản ghi hay thẻ ghi), mỗi cột là một FIELD (dịch là trường hay thành phần, hay thuộc tính cho sát với thực tế). Nói tổng quát, mỗi bản ghi là một tập gồm nhiều trường (field), các trường có thể có kiểu dữ liệu khác nhau.

7.3.2 Khai báo kiểu bản ghi

Cú pháp khai báo:

TYPE Tênkiểu = RECORD

 Têntrường1 : Kiểudữ liệu1;

 Têntrường2 : Kiểudữ liệu2;

 ...

 Têntrườngk : Kiểudữ liệuk;

End;

Ví dụ 7.42: Ta định nghĩa một kiểu KSVIEN như sau:

```
TYPE KSVIEN = RECORD
```

```
Hoten:String[20];
```

```
    Maso : String[8];
```

```
    Toan, Ly, DTB: Real;
```

```
END;
```

Ví dụ 7.43: Ta mô tả thời gian là kiểu KDATE có ba trường ngày , tháng, năm như sau:

```
TYPE KDATE = RECORD
```

```
    Ngày : 1..31;
```

```
    Thang : 1..12;
```

```
    Nam : Integer;
```

```
END;
```

Ví dụ 7.44: Để quản lý các sách trong một thư viện, ta xây dựng một kiểu bản ghi KSACH như sau:

```
TYPE KSACH = RECORD
```

```
    Ma_so_sach: String[6];
```

```
    Ten_doc_gia: String[20];
```

```
    Nam_xban :Integer;
```

```
    Gia_tien: Real;
```

```
    Ngày_muon : KDATE;
```

```
END;
```

Kiểu KSACH là một bản ghi có 5 trường mô tả 5 thuộc tính của sách là: mã số sách, tên độc giả, năm xuất bản, giá tiền và ngày mượn.

Ví dụ này cho thấy các bản ghi có thể được mô tả lồng nhau: kiểu dữ liệu của một trường của bản ghi này lại có thể là một kiểu bản ghi khác đã được định nghĩa trước đó. Trong bản ghi KSACH, Ngày_muon là một trường có kiểu dữ liệu là một bản ghi kiểu KDATE.

Mỗi đối tượng cần quản lý có thể có nhiều trường, song tùy yêu cầu quản lý mà ta chỉ lựa chọn và khai báo những trường thật sự cần thiết. Khai báo thừa thì hao phí bộ nhớ, nhưng nếu thiếu thì công tác quản lý sẽ khó khăn do thiếu thông tin. Vì vậy, nên khai báo các trường với số lượng ít nhất nhưng đủ dùng.

7.3.3 Sử dụng bản ghi

7.3.3.1. Truy cập từng thành phần

Kiểu bản ghi sau khi đã được định nghĩa có thể dùng khai báo cho các biến.

Ví dụ 7.45: `Var X, Y, Z : KSVIEN;`

Trong đó KSVIEN là bản ghi đã mô tả ở phần trên.

Theo khai báo này, X, Y và Z là ba biến kiểu bản ghi KSVIEN, mỗi biến đều có 5 trường Hoten, Maso, Toan, Ly và DTB.

Để thâm nhập vào một trường của bản ghi ta viết tên biến kiểu bản ghi, sau đó là dấu chấm '.' và tên trường, tức là:

Tên biến . Tên trường

Các lệnh dưới đây gán giá trị cho từng trường của biến X :

`X.Hoten := 'Nguyen Thanh Lam' ;`

`X.Maso := '1973208' ;`

`X.Toan := 8.0 ;`

`X.Ly := 7.0 ;`

`X.DTB := (X.Toan + X.Ly) / 2 ;`

Để nhập dữ liệu cho trường Hoten của biến Y, ta viết:

`Readln(Y.Hoten) ;`

Sở dĩ phải viết tên biến bản ghi ở trước tên trường là để xác định trường đó là của biến bản ghi nào. Mỗi biến X, Y, Z đều có trường Hoten, nên nếu chỉ viết Hoten thôi thì không biết đó là Hoten của ai: X, Y hay Z ?. Còn viết X.Hoten là chỉ rõ đây là Hoten của biến X.

Như vậy, mỗi trường của biến bản ghi có thể thâm nhập và sử dụng như một biến bình thường. X.Hoten là biến kiểu String[20], X.Maso là biến kiểu String[8], ..., X.DTB là biến kiểu Real.

Đối với các bản ghi lồng nhau, cách truy xuất đến từng trường cũng tương tự.

Ví dụ 7.46: Cho khai báo biến S kiểu KSACH:

`Var S : KSACH ;`

Để truy nhập đến các trường Ngay, Thang, Nam của Ngay_muon ta viết :

`S.Ngay_muon.Tên trường`

Chẳng hạn gán :

`S.Ngay_muon.Ngay := 2 ;`

`S.Ngay_muon.Thang := 9 ;`

`S.Ngay_muon.Nam := 1999 ;`

7.3.3.2. Các phép toán với bản ghi:

a) Phép gán: Hai biến bản ghi cùng kiểu có thể gán cho nhau.

Ví dụ 7.47: với khai báo ở ví dụ 7.50, lệnh : `Y:=X;`

gán giá trị của từng trường của biến X cho trường tương ứng của biến Y. Vậy lệnh trên tương đương với khối lệnh sau :

`begin`

```
Y.Hoten :=X.Hoten;
Y.Maso :=X.Maso;
Y.Toan :=X.Toan;
Y.Ly :=X.Ly;
Y.DTB :=X.DTB;
```

end;

b) Phép so sánh bằng hoặc khác:

Các bản ghi có thể so sánh bằng nhau hoặc khác nhau:

Ví dụ 7.48:

```
If X=Y then
    writeln(' X và Y là một người ');
If X<>Y then
    writeln(' X khác Y ');
```

Tuy nhiên **không có phép so sánh** <, <=, >, >= cho các bản ghi.

Hai bản ghi có thể hoán đổi giá trị cho nhau theo nghĩa hoán đổi từng cặp giá trị của các trường tương ứng.

Giống như các biến đơn giản, để hoán đổi hai bản ghi X và Y ta dùng ba lệnh:

```
Z:=X; X:=Y; Y:=Z;
```

trong đó Z là biến trung gian cùng kiểu bản ghi với X và Y.

Ví dụ 7.49: Nếu biến X và Y có các trường tương ứng là :

X.Hoten ='Nguyen Thanh Lam'	Y.Hoten ='Nguyen Tung Lam'
X.Maso ='1973208'	Y.Maso ='1974564'
X.Toan =8.0	Y.Toan =5.0
X.Ly =7.0	Y.Ly =8.0
X.DTB =7.5	Y.DTB =6.5

thì sau khi hoán đổi, ta được :

X.Hoten ='Nguyen Tung Lam'	Y.Hoten =' Nguyen Thanh Lam'
X.Maso ='1974564'	Y.Maso ='1973208'
X.Toan =5.0	Y.Toan =8.0
X.Ly =8.0	Y.Ly =7.0
X.DTB =6.5	Y.DTB =7.5

7.3.4 Câu lệnh WITH

Khi làm việc với nhiều trường của một biến bản ghi thì cách thâm nhập ở trên tỏ ra rườm rà vì phải viết nhiều lần tên biến trước tên trường.

Để đơn giản cách viết, Pascal đưa ra câu lệnh :

WITH Tên_biến DO LệnhP;

Trong đó **Tên_biến** thuộc kiểu bản ghi.

Nếu LệnhP có truy xuất đến các trường của Tên biến thì không cần phải viết Tên biến và dấu chấm trước các tên trường.

Ví dụ 7.50: thay vì viết:

```
X.Hoten:= 'Nguyen Van A';
```

ta có thể viết :

```
WITH X DO
```

```
    Hoten:= 'Nguyen Van A';
```

Để in các trường của biến X lên màn hình, ta viết:

```
WITH X DO
```

```
    Begin
```

```
        Writeln(' Ho va ten :' , Hoten);
```

```
        Writeln(' Ma so      :' , Maso);
```

```
        Writeln(' Diem Toan :' , Toan: 4:1);
```

```
        Writeln(' Diem Ly   :' , Ly: 4:1);
```

```
        Writeln(' Diem TB   : ' , DTB :4:1);
```

```
    End;
```

Tất cả các tên trường nằm trong khối begin và end được hiểu là các trường của biến X (nếu không ghi rõ tên biến nào khác).

Các lệnh sau gán các giá trị cho các trường của biến S kiểu KSACH là một bản ghi lỏng nhau:

```
WITH S DO
```

```
    begin
```

```
        Ma_so_sach:='TH-435';
```

```
        Ten_doc_gia:='Nguyen van Mai';
```

```
        Nam_xban :=1999;
```

```
        Gia_tien:= 15000;
```

```
        WITH Ngay_muon DO
```

```
            begin
```

```
                Ngay:=2;
```

```
                Thang:=9;
```

```
                Nam:=1999;
```

```
            end;
```

```
    end;
```

7.3.5 Mảng các bản ghi

Trong thực tế, ta thường phải quản lý một danh sách sinh viên của một lớp, một danh sách các quyển sách trong thư viện. Vì thế mảng các bản ghi được dùng rất phổ biến.

Ví dụ 7.51: Khi khai báo :

```
VAR DS : Array[1..50] of KSVIEN;
```

Hoặc :

```
TYPE    KMang = Array[1..50] of KSVIEN;  
VAR     DS: KMang;
```

thì ta có một mảng DS gồm 50 phần tử DS[1],..., DS[50] cùng kiểu bản ghi KSVIEN. Mỗi DS[i], i=1,..., 50 là một bản ghi có 5 trường Hoten, Maso, Toan, Ly, DTB lưu trữ các thông tin về sinh viên thứ i trong danh sách.

Để nhập dữ liệu (Hoten, Maso, Toan, Ly) rồi tính Điểm trung bình cho 50 sinh viên này, ta dùng vòng lặp For phối hợp với câu lệnh WITH :

```
For i:=1 to 50 do  
  WITH DS[i] DO  
    begin  
      Write('Nhap ho ten sinh vien thu ',i,' : ');  
      Readln(Hoten);  
      Write('Nhap ma so sinh vien thu ',i,' : ');  
      Readln(Maso);  
      Write('Nhap điểm Toan, Ly sinh vien thu ',i,' : ');  
      Readln(Toan, Ly);  
      DTB:=(Toan + Ly)/2;  
    end;
```

Để sắp xếp danh sách sinh viên theo trật tự giảm của DTB sao cho người có DTB cao thì đứng trước, người có DTB thấp thì đứng sau, ta có thể áp dụng phương pháp sắp xếp mảng, song chỉ lấy trường DTB làm tiêu chuẩn so sánh:

```
For i:=1 to 49 do  
  For j:=i+1 to 50 do  
    If DS[i].DTB < DS[j].DTB then  
      begin { Đổi chỗ DS[i] và DS[j] }  
        Z:=DS[i];  
        DS[i]:=DS[j];  
        DS[j]:=Z;  
      end;
```

Ở đây Z là biến bản ghi cùng kiểu với các phần tử DS[i].

Khi sắp xếp theo DTB tăng hoặc giảm thì trường DTB được gọi là "khóa" (key) sắp xếp. Một cách tương tự, có thể sắp xếp danh sách sinh viên theo khóa là điểm Toan, hoặc điểm Ly, hoặc Maso, hoặc theo Hoten.

Để in danh sách đã sắp xếp lên màn hình, ta dùng lệnh :

```
For i:=1 to 50 do  
  WITH DS[i] DO
```

```
Writeln(i:2, Hoten:25, Maso:9 , Toan:5:1, Ly:5:1, DTB:5:1);
```

7.3.6 Ví dụ về bản ghi

Ví dụ 7.52: Nhập một danh sách N ($1 \leq N \leq 50$) sinh viên gồm các trường: Họ tên, Mã số, các điểm Toán, Lý.

Đối với mỗi sinh viên, hãy tính điểm trung bình :

$DTB = (Toan + Ly) / 2$, và phân loại như sau:

Loại= Giỏi nếu $DTB \leq 9$,

Khá nếu $7 \leq DTB < 9$,

Bình nếu $5 \leq DTB < 7$,

Kém nếu $DTB < 5$.

Sắp xếp danh sách theo trật tự giảm của DTB, in danh sách lên màn hình, mỗi người trên một dòng gồm các mục: Số thứ tự, Họ tên, Mã số, điểm Toán, Lý, DTB và phân loại.

Cho biết điểm Toán cao nhất là mấy ?. Có bao nhiêu người được điểm cao nhất đó?

Điểm trung bình môn Toán cho cả danh sách là bao nhiêu ?.

Bài giải :

Trong chương trình dưới đây, chỗ nào có ký hiệu ' ' thì hãy thay bằng một ký tự trắng.

```
PROGRAM VIDU_7_52;
Uses CRT;
Type    KSVIEN = RECORD
                Hoten:String[18];
                Maso, Loai : String[8];
                Toan, Ly, DTB : Real;
            End;
Var  DS : Array[1..50] of KSVIEN;
     Z : KSVIEN;
     N, i, j, Dem : Integer;
     Max, TBToan: Real;
BEGIN
    CLRSCR;
    Repeat
        Write(' Nhập số sinh viên N= ') ;
        Readln(N) ;
    Until ( N>0) and ( N < 51);
    { Nhập danh sach, tính DTB va phân loại }
    For i:=1 to N do WITH DS[i] DO
        begin
            Write('Nhap ho ten sinh vien thu ',i,' : ');
```

```
Readln(Hoten);
Write('Nhap ma so sinh vien thu ',i,' : ');
Readln(Maso);
Write('Nhap điểm Toán, Lý sinh vien thu ',i,' : ');
Readln(Toan, Ly);
DTB:=( Toan + Ly)/2;

If DTB >=9 then Loai:='Gioi'
else
    if DTB >=7 then Loai:='Kha'
    else
        if DTB >=5 then Loai:='Binh'
        else
            Loai:='Kem';
    end;
{ Sắp xếp giảm theo DTB }
For i:=1 to N-1 do
    For j:=i+1 to N do
        If DS[i].DTB < DS[j].DTB then
            begin
                Z:=DS[i]; DS[i]:=DS[j]; DS[j]:=Z;
            end;
Writeln(' In danh sach len man hinh ');
Writeln('STT HO VA TEN MA SO TOAN LY DTB LOAI');
For i:=1 to N do
    WITH DS[i] DO
        Writeln(i:2,#32,Hoten,#32:19-Length(Hoten),
            Maso:8,Toan:4:1,Ly:4:1,DTB:4:1,Loai:5);
{ Tìm điểm Toán cao nhất }
Max:=DS[1].Toan;
For i:=1 to N do
    if Max< DS[i].Toan then
        Max:=DS[i].Toan;
Writeln('Diem Toán cao nhat = ', Max:4:1);
{ Đếm số em có điểm Toán =Max}
Dem:=0;
For i:=1 to N do
    if DS[i].Toan =Max then
        Dem:=Dem+1;
Writeln(' Có ', Dem, ' em có điểm Toán= ', Max:4:1);
{ Tính điểm trung bình môn Toán cho cả danh sach }
```

```
TBToan:=0; { Lấy tổng điểm môn Toán}
For i:=1 to N do
    TBToan := TBToan + DS[i].Toan;
TBToan:=TBToan/N ;
Writeln('Diem trung binh mon Toan= ' , TBToan:6:2);
Readln;
END.
```

Ví dụ 7.53 Nhập một danh sách N ($1 \leq N \leq 50$) đầu sách gồm các thông tin về Tên sách, Số lượng (SL) và Đơn giá (DG).

Với mỗi đầu sách, tính Giá tiền như sau:

$$\begin{aligned} GT &= SL * DG \text{ nếu } SL < 10 \\ &= SL * DG - 5\% * SL * DG \text{ nếu } 10 \leq SL < 30 \\ &= SL * DG - 8\% * SL * DG \text{ nếu } 30 \leq SL < 50 \\ &= SL * DG - 10\% * SL * DG \text{ nếu } SL \geq 50 \end{aligned}$$

(Tức là nếu mua từ 10 đến 29 cuốn thì được giảm 5% đơn giá, mua từ 30 đến 49 cuốn thì được giảm 8% đơn giá, mua từ 50 cuốn trở lên thì được giảm 10% đơn giá).

Tính Tổng số tiền phải chi cho việc mua sách, kể cả 10% thuế giá trị gia tăng đánh vào tổng giá tiền.

Tìm xem trong danh sách đó có cuốn "TIN HOC DAI CUONG" không. Nếu có thì cho biết có bao nhiêu cuốn và đơn giá của nó.

Bài giải:

```
PROGRAM VIDU_7_53;
Uses CRT;
Type KSACH = RECORD
    Tensach : String[20];
    SL : Integer;
    DG, GT: Real;
End;
Mang = Array[1..50] of KSACH;
Var S : Mang;
    Z : KSACH;
    N, i : Integer;
    Tongtien : Real;
BEGIN
Repeat
    Write(' Nhập N: '); Readln( N );
Until (N>0) and ( N<51);
{ Nhập N sách và tính tiền}
For i:=1 to N do
```

```
With S[i] do
begin
    Write('Nhap ten sach thu: ',i,':'); Readln(Tensach);
    Repeat
        Write('Nhap Số lượng : '); Readln(SL);
    Until SL>0;
    Write('Nhap đơn giá : '); Readln(DG);
    Case SL of
        1..9 : GT:=SL*DG;
        10..29 : GT:=SL*DG*(1-0.05);
        30..49 : GT:=SL*DG*(1-0.08);
        else GT:=SL*DG*(1-0.1);
    end; { Hết Case}
end; { Hết For}

{Tính tổng giá thành các đầu sách}
Tongtien:=0;
For i:=1 to N do
    Tongtien:=Tongtien+S[i].GT;

{Cộng thêm thuế 10% giá trị gia tăng}
Tongtien:=Tongtien + 0.1*Tongtien ;
Writeln('Tổng tiền phải chi là: ', Tongtien :6:2);

{Tìm cuốn 'TIN HOC DAI CUONG'}
i:=1;
While (i<=N) and (S[i].Tensach<> 'TIN HOC DAI CUONG') do
    i:=i+1;
If i> N then
    writeln (' Không có cuốn TIN HOC DAI CUONG! ')
else
    WITH S[i] DO
        writeln(' Co ', SL:4, ' cuon, don gia= ',DG:6:2);
    Readln;
END.
```

7.4. Kiểu tập hợp (Set of)

7.4.1. Khái niệm

Kiểu tập hợp là một kiểu dữ liệu có cấu trúc nhằm thể hiện mô hình dữ liệu tập hợp trong toán học. Nó thể hiện loại dữ liệu có thể xác nhận giá trị không phải chỉ là một phần tử đơn lẻ

trong miền xác định không phải chỉ là một phần tử đơn lẻ trong miền xác định mà là một tập hợp nhiều phần tử cùng thuộc một kiểu dữ liệu vô hướng đếm được nào đó.

7.4.2. Cú pháp

Phần mô tả khai báo kiểu tập hợp có cú pháp như sau:

TYPE *tên_kiểu_tập_hợp* = **SET OF** *tên_kiểu_dữ_liệu_cơ_sở*;

Var *tên_biến* : *tên_kiểu_tập_hợp*;

Trong đó: *tên_kiểu_dữ_liệu_cơ_sở* có thể là một kiểu dữ liệu vô hướng đếm được bất kỳ, kể cả kiểu đoạn con, kiểu liệt kê.

Chú ý: Turbo Pascal hạn chế số phần tử có thể của mỗi giá trị kiểu tập hợp là 256

7.4.3. Một số tính chất

- Một giá trị của dữ liệu kiểu tập hợp được thể hiện bằng cách liệt kê mọi phần tử của tập, cách nhau bởi dấu phẩy, đóng trong hai dấu ngoặc vuông.
- Thứ tự liệt kê các phần tử không quan trọng.
- Có thể gán giá trị cho biến tập hợp như thông thường.
- Tập rỗng ký hiệu là [].

7.4.4. Các phép toán trên tập hợp

7.4.4.1. Phép toán quan hệ

Để thực hiện các phép toán quan hệ thì các toán hạng toán hạng phải cùng kiểu. và thực hiện như sau:

- So sánh bằng nhau:

$A = B$ cho kết quả là TRUE nếu A và B hoàn toàn như nhau,

- So sánh khác nhau:

$A \neq B$ cho kết quả là FALSE nếu A và B hoàn toàn như nhau

- So sánh nhỏ hơn hay bằng:

$A \leq B$ là TRUE nếu A là tập con của B

- So sánh lớn hơn hay bằng:

$A \geq B$ là TRUE nếu B là tập con của A.

* **Chú ý:** không có so sánh nhỏ hơn chặt (<), lớn hơn chặt (>) đối với kiểu tập hợp. Muốn thể hiện quan hệ “là tập con thực sự” ta phải kết hợp hai điều kiện:

$(A \leq B) \text{ and } (A \neq B)$

7.4.4.2. Phép tìm kiếm (thuộc về)

Để diễn tả một phần tử **x** có thuộc một tập hợp **A** hay không ta sử dụng cú pháp:

x in A cho kết quả là TRUE nếu **x** là một phần tử của **A**

ngược lại cho kết quả là FALSE

7.4.4.3. Phép hợp, Giao, Hiệu

Để thực hiện các phép hợp, giao, hiệu thì hai toán hạng cũng phải cùng kiểu.

- Phép hợp được ký hiệu bằng dấu +:

$$A+B = \{x_i \mid (x_i \text{ in } A) \text{ or } (x_i \text{ in } B)\}$$

- Phép giao được ký hiệu bằng dấu *:

$$A*B = \{x_i \mid (x_i \text{ in } A) \text{ and } (x_i \text{ in } B)\}$$

- Phép lấy hiệu được ký hiệu bằng dấu -:

$$A - B = \{x_i \mid (x_i \text{ in } A) \text{ and not}(x_i \text{ in } B)\}$$

7.4.5. Viết và đọc dữ liệu kiểu tập hợp

Nói chung ta không thể dùng các thủ tục WRITE (WRITELN) và READ (READLN) để viết ra và đọc vào các dữ liệu kiểu tập hợp. Tuy nhiên chúng ta có thể lập trình để thực hiện các thao tác này khi mà các phần tử của tập hợp là các số nguyên hoặc ký tự.

Ví dụ 7.54: Nhập vào n ký tự rồi lựa chọn trong đó các chữ hoa để thành lập một tập hợp và cuối cùng là in ra các ký tự trong tập vừa tạo.

```
Program vidu_7_54;
VAR  Tap_chu_hoa : SET OF 'A'..'Z' ;
     i, n :Integer;
     Ch :Char;
BEGIN
    WRITE(' Nhập số n = '); READLN(n);
    Tap_Chua_Hoa :=[]; {Khoi tao Tap_chu_hoa la tap rong }
    {Để đọc vào n ký tự bất kỳ }
    FOR i :=1 TO n DO
        BEGIN
            WRITE('Nhập Ký tự thứ ', i , ' : ' ); READLN(Ch);
            {Neu ch la ky tu hoa thi cho no vao Tap_Chua_Hoa}
            IF ch=UPCASE(ch) THEN
                Tap_Chua_hoa :=Tap_Chua_hoa + [Ch];
        END;
    {in cac ky tu trong Tap_Chua_hoa ra man hinh }
    FOR Ch :='A' TO 'Z' DO
        IF Ch IN Tap_Chua_Hoa THEN
            WRITE(ch : 4) ;
    Writeln;
    READLN;
END.
```

Ví dụ 7.55: Đếm số kí tự là nguyên âm, là phụ âm trong một dãy kí tự. Để đơn giản ta dùng kí tự số 0 để đánh dấu kết thúc dãy.

Dễ thấy rằng cần phải có hai tập hợp chữ cái là các nguyên âm và các phụ âm để phục vụ cho việc kiểm tra từng kí tự đọc vào. Chương trình cần phải định nghĩa kiểu dữ liệu *TapKiTu* và hai biến *NguyenAm*, *PhuAm* có kiểu là tập hợp bộ chữ cái như đã nêu trên.

```
Program vidu_7.55;
Type TapKiTu = Set of Char;
Var NguyenAm, PhuAm: TapKiTu;
    ch: char;
    So_NguyenAm, So_PhuAm: byte;
Begin
    So_NguyenAm:= 0; So_PhuAm:= 0;
    NguyenAm:= ['a','e','i','o','u'];
    PhuAm:= ['a'..'z'] - NguyenAm;
    Write(' cho day ki tu, ket thuc bang so 0: ');
    read(ch);
    while ch <> '0' do
        begin
            if ch in NguyenAm then
                So_NguyenAm:= So_NguyenAm + 1;
            if ch in PhuAm then
                So_PhuAm:= So_PhuAm + 1;
            read(ch);
        end;
    Writeln('So nguyen am la: ', So_NguyenAm);
    Writeln('So phu am la: ', So_PhuAm);
end.
```

Ví dụ 7.56: Tìm các số nguyên tố trong khoảng 1..N

Thuật toán sàng Eratosthene.

Phân tích:

Sàng ban đầu chứa toàn bộ dãy số 2 .. N. Bắt đầu từ số nhỏ nhất là 2, ta loại bỏ khỏi sàng tất cả các số là bội số của 2. Số nhỏ nhất còn lại trên sàng sẽ là số nguyên tố thứ nhất là 3. Bỏ số này ra khỏi sàng, ghi nó vào tập số nguyên tố.

Tiếp tục loại bỏ khỏi sàng các số là bội số của các số nguyên tố vừa chọn được. Số nhỏ nhất còn lại trên sàng sẽ là số nguyên tố tiếp theo.

Lặp lại việc này ta sàng dần được hết các số nguyên tố.

```
Program Sang_Eratosthene ;
Const N=100;
Type Nguyen = 1.. N;
```

```
Var NguyenTo, Sang: Set of Nguyen ;
    Number: Nguyen ;
    I: Integer ;
BEGIN
    NguyenTo:= [ ] ;
    Sang:= [ 2..N] ;
    Number:= 2;
    Repeat
        While not (Number IN Sang) do
            Number:= Number + 1;
        NguyenTo:= NguyenTo + [Number] ;
        Write ( Number , ' ' );
        i:= Number ;
        While i <= N do
            Begin
                Sang:= sang - [i] ;
                I:= i + Number ;
            End.
        Until Sang = [ ] ;
    END.
```

7.5. Kiểu tệp (FILE)

7.5.1. Khái niệm

Nhập và xuất dữ liệu là hai công việc rất phổ biến khi thực hiện một chương trình. Cho đến nay, ta mới chỉ nhập dữ liệu từ bàn phím và xuất dữ liệu ra màn hình. Các dữ liệu này được tổ chức trong bộ nhớ của máy, chúng tồn tại khi chương trình đang chạy và bị xóa khi chương trình kết thúc. Muốn lưu trữ các dữ liệu lâu dài để sử dụng nhiều lần thì phải ghi chúng lên đĩa thành các tệp.

Tệp (file) trong Pascal là một kiểu dữ liệu có cấu trúc. Mỗi tệp là một tập hợp các phần tử có cùng chung một kiểu dữ liệu được nhóm lại thành một dãy và được ghi trên đĩa dưới một cái tên chung. Khái niệm tệp và mảng có những điểm rất gần nhau. Song tệp khác mảng ở những điểm sau đây:

Mảng được tổ chức trong bộ nhớ còn tệp chủ yếu được tổ chức trên đĩa.

Số phần tử của mảng được xác định ngay khi khai báo, còn số phần tử của tệp thì không. Các tệp được kết thúc bằng một dấu hiệu đặc biệt gọi là EOF (End Of File).

Các phần tử của mảng được truy xuất thông qua chỉ số. Các phần tử của tệp được truy xuất nhờ một biến trung gian chỉ điểm vào vị trí của chúng trên đĩa, gọi là con trỏ tệp. Tại mỗi thời điểm, con trỏ sẽ chỉ vào một vị trí nào đó trong tệp, gọi là vị trí hiện thời.

Cách khai báo kiểu tệp như sau:

Type *Tên_kiểu_Tệp* = **File of Kiểu_phần_tử** ;

Ví dụ 7.57:

```
Type    Ksvien = Record
                                Ten: String[20];
                                Namsinh : Integer;
                                DTB  : Real;
                                end;
KieuT1 = File of Integer;
KieuT2 = File of String[20];
KieuT3 = File of Ksvien ;
```

Theo khai báo trên thì KieuT1 là tệp có các phần tử kiểu nguyên (Integer), KieuT2 là tệp có các phần tử là các chuỗi ký tự (String[20]), còn KieuT3 là tệp có các phần tử là các bản ghi kiểu Ksvien.

Khi đã có kiểu tệp, ta có thể khai báo các biến tệp :

```
Var    F1 : KieuT1;
       F2 : KieuT2;
       F3 : KieuT3;
```

F1, F2, F3 là các biến kiểu tệp, một loại biến đặc biệt, không dùng để gán giá trị như các biến nguyên, thực hay chuỗi. Mỗi biến này đại diện cho một tệp mà thông qua các biến đó ta có thể thực hiện các thao tác trên tệp như : tạo tệp, mở, đóng, xóa tệp, ghi dữ liệu vào tệp và đọc dữ liệu từ tệp, ...

Ngoài cách khai báo các biến F1, F2, F3 thông qua việc định nghĩa các kiểu dữ liệu mới như trên, Pascal còn cho phép khai báo trực tiếp các biến tệp như sau:

```
Var    TênbiếnTệp : File of Kiểuphầntử ;
```

Ví dụ 7.58: có thể khai báo ba biến F1, F2, F3 nói trên theo cách sau :

```
Type    Ksvien = Record
                                Ten: String[20];
                                Namsinh : Integer;
                                DTB  : Real;
                                end;
Var    F1 : File of Integer;
       F2 : File of String[20];
       F3 : File of Ksvien ;
```

7.5.2. Cấu trúc và phân loại tệp

Các phần tử của tệp không có tên nên truy cập không thể tùy tiện được. Các phần tử của tệp được sắp xếp thành một dãy và mỗi thời điểm chương trình chỉ có thể truy cập tệp thông qua một biến đếm. Biến đếm dùng để đánh dấu vị trí truy cập tệp hay còn gọi là cửa sổ tệp. Mỗi tệp

đều có một ký hiệu kết thúc gọi là EOF(F) - End of File F. Việc phân loại tệp dựa trên việc bố trí các phần tử của tệp và cách truy cập tệp: truy cập tệp tuần tự và truy cập tệp trực tiếp.

7.5.3. Tệp định kiểu

Như đã nêu ở phần trên, tệp là cấu trúc để lưu trữ nhiều dữ liệu. Nếu tệp lưu các phần tử dữ liệu cùng kiểu thì ta có tệp định kiểu.

- Tệp định kiểu là tập hợp các phần tử dữ liệu *cùng kiểu, xếp liền nhau*;
- Số phần tử hay kích thước của tệp *có thể thay đổi* trong thời gian thực hiện chương trình ;
- Kích thước tệp không được xác định khi khai báo.

Cú pháp:

Type *tên kiểu tệp* = **File of** *kiểu phần tử* ;

Trong đó kiểu phần tử của tệp gọi là *kiểu cơ sở*, có thể là bất kì kiểu gì, trừ kiểu tệp.

Ví dụ 7.59:

```
TYPE    TepSoThuc    = FILE OF    Real ;  
         TepSoNguyen    =    FILE OF    Integer ;  
         TepHoSo    =    FILE    OF    LyLich ;  
  
VAR    F1: TepSoThuc ;  
       F2: TepSoNguyen ;  
       F3: TepHoSo ;
```

Cũng có thể khai báo tắt, kết hợp khai báo biến với mô tả kiểu tệp:

```
F1: FILE    OF    Real ;  
F2: FILE    OF    Integer;  
F3: FILE    OF    LyLich ;
```

Biến kiểu tệp là biến đại diện cho tệp. Mọi việc truy cập tệp đều phải thông qua biến tệp.

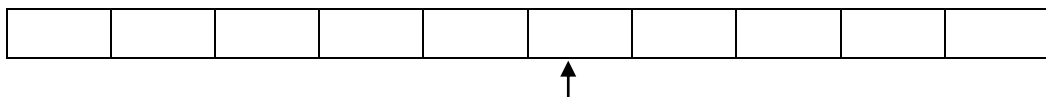
7.5.4. Tệp truy cập tuần tự

Mô tả cấu trúc

Theo định nghĩa, tệp là tập hợp các phần tử dữ liệu *cùng kiểu, xếp liền nhau*, do đó có thể coi tệp như một mảng với kích thước cố định được.

Cấu trúc của mảng có khuôn mẫu cố định nên có thể truy cập trực tiếp vào các phần tử qua chỉ số mảng. Trái lại, các phần tử làm nên tệp không được đánh chỉ số, vì có thể bị di chuyển do các thao tác thêm bớt vào nội dung.

Tại một thời điểm chỉ có thể truy cập vào một phần tử thông qua giá trị của một biến đếm.



Có thể minh họa biến đếm như một cửa sổ di động dọc theo tệp để nhìn vào các phần tử tệp. Tại một thời điểm của sổ này nằm ở một vị trí xác định.

Cửa sổ sẽ tự động dịch đến phần tử kế tiếp sau mỗi lần truy cập.

Có dấu hiệu đặc biệt *EOF* tại vị trí kết thúc tệp và hàm chuẩn *Eof(F)* cho biết cửa sổ tệp của tệp *F* đã ở phần tử cuối tệp hay chưa. *Eof(F) = True* nếu đã ở vị trí cuối tệp, *Eof(F) = False* nếu trái lại.

Tệp được tổ chức như trên gọi là **tệp truy cập tuần tự - *Sequential access***.

Với tệp truy cập tuần tự, muốn đọc một phần tử nào đó phải đi qua các phần tử đứng trước. Muốn thêm một phần tử phải đưa cửa sổ về cuối tệp.

Hình ảnh của tệp truy cập tuần tự là băng từ.

Tóm lại, tệp truy cập tuần tự có cấu trúc đơn giản và dễ xây dựng nhưng kém linh hoạt.

Pascal chuẩn chỉ định nghĩa tệp truy cập tuần tự. TurboPascal có cung cấp kiểu tệp truy cập trực tiếp, cho phép đặt cửa sổ tệp vào một vị trí bất kì trong tệp và truy cập các phần tử tệp tương tự như truy cập mảng.

7.5.5. Mở tệp mới để ghi dữ liệu

Chương trình chỉ có thể ghi dữ liệu vào tệp sau khi ta đã làm thủ tục mở tệp. Việc mở tệp được tiến hành nhờ hai thủ tục sau:

ASSIGN(biểntệp, têntệp);

REWRITE(biểntệp);

Ví dụ 7.60: Gán tên tệp cho biến tệp, ở đây tên tệp là một biểu thức kiểu chuỗi tên thực sự của tệp.

```
Assign(F1, 'DLIEU.DAT');
```

khởi tạo tệp DLIEU.DAT nhờ thủ tục

```
Rewrite(F1);
```

```
Assign(F3, 'QLSV.DAT');
```

```
Rewrite(F3); {khởi tạo tệp QLSV.DAT}
```

Sau hai lệnh này máy khởi tạo tệp mới, nếu tệp đã có trên đĩa thì nó xóa đi và tạo mới, biến *F1* đồng nhất với tệp DLIEU.DAT, mọi thao tác trên biến *F1* chính là thao tác trên tệp DLIEU.DAT. Tương tự, biến *F3* đồng nhất với tệp QLSV.DAT

Sau khi mở tệp xong, tệp sẽ rỗng vì chưa có phần tử nào, cửa sổ của tệp sẽ không có giá trị xác định vì nó trở vào cuối tệp.

***Chú ý:**

- Nếu muốn tạo ra tệp trong một thư mục khác thì tên tệp phải bao gồm cả đường dẫn.
- Nếu trên đĩa đã có sẵn một tệp trùng tên thì nội dung của tệp này sẽ bị xoá. Tệp trở thành rỗng.

*** Ghi dữ liệu vào tệp với thủ tục *WRITE***

Thủ tục *WRITE* sẽ ghi các giá trị mới vào tệp

WRITE(biểntệp, b1, b2, ..., bN);

Tuần tự ghi vào tệp các giá trị của các biến *b1, b2, ..., bN*. Các biến *b1, ..., bN* phải cùng kiểu dữ liệu với các phần tử của tệp.

Ví dụ 7.61:

- Cho i, j, k là các biến kiểu Integer và i=10, j=20, k=100, khi đó lệnh :

```
Write(F1, i, j, k);
```

sẽ ghi lần lượt các giá trị 10, 20, 100 vào tệp DLIEU.DAT

- Cho khai báo:

```
Var X : Ksvien;
```

Các lệnh sau gán giá trị cho X và ghi X vào tệp QLSV.DAT:

```
X.Ten:='Nguyen Thanh Lam';
```

```
X.Namsinh :=2002;
```

```
X.DTB :=6.5;
```

```
Write(F3, X);
```

Sau khi ghi X vào tệp QLSV.DAT, con trỏ tệp tự động dời đến vị trí của phần tử tiếp theo.

*** Đóng tệp**

Bước cuối cùng của việc ghi dữ liệu vào tệp là đóng tệp lại bằng thủ tục CLOSE để đảm bảo thông tin trên tệp là đầy đủ và tin cậy.

CLOSE(biếntệp);

Ví dụ 7.62: Chương trình sau ghi 10 số nguyên từ 1 đến 10 vào tệp SoNguyen.DAT

```
Program Ghi_Tep;
```

```
Var I : Integer;
```

```
f : File Of Integer;
```

```
Begin
```

```
Writeln('THU TUC WRITE, GHI DU LIEU VAO TEP TREN DIA');
```

```
Writeln('-----');
```

```
Writeln;
```

```
Assign(f, 'SoNguyen.DAT');
```

```
Rewrite(f);
```

```
For i := 1 To 10 Do
```

```
Write(f, i);
```

```
Writeln;
```

```
Writeln('Da ghi vao tep SONGUYEN');
```

```
Writeln(' Bam <Enter>...');
```

```
Readln;
```

```
Close(f);
```

```
End.
```

7.5.6. Mở tệp đã tồn tại để đọc dữ liệu

Đối với các tệp tuần tự, ta không **thể** vừa ghi vừa đọc được cùng một lúc. Sau khi ghi dữ liệu vào tệp và đóng lại, ta có thể đọc lại các giá trị đã ghi trong tệp.

Một chương trình muốn sử dụng các dữ liệu đã chứa trong một tệp, đầu tiên phải mở nó ra để đọc.

Ta có hai thủ tục sau:

ASSIGN(biểntệp, tên tệp);

RESET(biểntệp);

Sau lệnh RESET con trỏ tệp trở vào phần tử đầu tiên (có số thứ tự là 0) của tệp.

Ví dụ 7.63: Assign(F3, 'QLSV.DAT');

Reset (F3); mở tệp QLSV.DAT

***đọc dữ liệu từ một tệp đã có**

Để đọc dữ liệu từ tệp ta dùng thủ tục READ có cú pháp như sau:

READ(biểntệp, b1, b2, ..., bN);

Đọc tuần tự các phần tử của tệp từ vị trí hiện thời của con trỏ tệp và gán cho các biến b1, b2, ..., bN. Kiểu dữ liệu của các biến b1, b2, ..., bN phải cùng kiểu với các phần tử của tệp. Mỗi khi đọc xong một phần tử, con trỏ tệp tự động dời đến phần tử tiếp theo.

Ví dụ 7.64:

Lệnh Read(F1, i, j); đọc hai số nguyên trong tệp DLIEU.DAT (kể từ vị trí hiện thời) và gán cho các biến nguyên i, j .

Lệnh Read(F3, X); đọc bản ghi hiện thời của tệp QLSV.DAT và gán cho biến bản ghi X.

Chương trình sau sẽ đọc các phần tử của một tệp có tên được nhập từ bàn phím (kiểm tra sự tồn tại của tệp).

```
Program Doc_Tep;  
Label tt;  
Var  
    I,k : Integer;  
    Ten : String;  
    f   : File Of Integer;  
Begin  
    Writeln('THU TUC READ, DOC DU LIEU TU TEP TREN DIA');  
    Writeln('-----');  
    tt:Writeln;  
    k := 0;  
    Write('-Cho biet ten tap tin: ');  
    Readln(Ten);  
    Assign(f,Ten);  
    {$I-}  
    Reset(f);  
    If IOResult <> 0 Then  
        {ham IOResult =0 tuc tep f co ton tai}  
        Begin
```

```
        Writeln;
        Writeln(' Không có tệp này');
        Write(' Bam <Enter> để tìm lại: ');
        Readln;
        Goto tt;
    End;
While Not EOF(f) Do
    Begin
        Read(f,i);
        Writeln(i);
        k := k+1;
    End;
Writeln;
Writeln('Tệp: ',ten,' có: ',k:3,' phần tử');
Write(' Bam <Enter>...');
Readln;
Close(f);
End.
```

Ví dụ 7.65:

Nhập một danh sách sinh viên gồm Họ tên, điểm Toán, điểm Lý, tính điểm trung bình rồi lưu vào tệp HOSO.DAT. Sau đó đọc dữ liệu tệp này và in ra màn hình.

```
PROGRAM VIDU_7_65;
Uses CRT;
Const Tenttin = 'HOSO.DAT';
Type Ksvien= Record
    Hoten: String[20];
    Toan, Ly, Dtb : Real;
end;
KieuTtin = File of Ksvien ;
Var F : KieuTtin ;
    X : Ksvien ;
    i: Integer;
BEGIN
    Clrscr;
    Assign(F, Tenttin);
    Rewrite(F);
    i:=0;
    Repeat
        Clrscr;
```

```
Gotoxy(10,4);
Write('NHAP SV THU ',i,':(Enter để kết thúc)');
With X do
  begin
    Gotoxy(10,6); Write('Ho va ten:');
    Gotoxy(10,8); Write('Diem Toan:');
    Gotoxy(10,10); Write('Diem Ly :');
    Gotoxy(20,6); Readln(Hoten);
    If Hoten<>' ' then
      begin
        Gotoxy(20,8); Readln(Toan);
        Gotoxy(20,10); Readln(Ly);
        DTB:=(Toan+Ly)/2;
      end;
    end;
  If X.Hoten<>' ' then
    Write(F,X);
  i:=i+1;
Until X.Hoten=' ';
Close(F);
Writeln(#32:5,'Ho va ten',#32:6,'TOAN~~~LY~~~DTB');
Reset(F);
While Not Eof(F) do
  begin
    Read(F, X);
    Writeln(X.Hoten, #32:20-Length(X.Hoten),
      X.Toan:4:1,#32:3, X.Ly:4:1, #32:3, X.DTb:4:1);
  end;
Close(F);
Readln;
END.
```

***Chú ý:** Nếu tham số trong chương trình con là tệp thì nó phải là tham số biến, không thể là tham số trị.

7.5.7. Tệp truy cập trực tiếp

PASCAL chuẩn chỉ định nghĩa một kiểu tệp truy nhập tuần tự. Tuy nhiên các bộ nhớ ngoài như đĩa từ, đĩa quang, . . có thể cho phép tính toán tọa độ của một phần tử bất kỳ trong tệp (vì độ dài các phần tử là bằng nhau), do đó có thể truy cập trực tiếp vào một phần tử mặc dù cấu tạo logic của tệp vẫn là dạng tuần tự. Trong TURBO PASCAL để truy nhập trực tiếp vào phần tử của tệp ta dùng thủ tục SEEK, cú pháp:

SEEK(biểntệp, k);

Đặt con trỏ tệp vào phần tử thứ k trong tệp. Thủ tục này cho phép truy xuất trực tiếp một phần tử của tệp mà không phải thực hiện tuần tự từ đầu tệp.

Ví dụ 7.66: đọc phần tử thứ 10 của tệp DLIEU.DAT và gán cho biến nguyên i rồi in giá trị của i:

```
Seek(F1, 10);  
Read(F1, i);  
Write(i);
```

Chương trình sau minh họa thủ tục SEEK

```
Program Vidu_7_66_Seek;  
Var I,N : Integer;  
    Ch : Char;  
    f : File Of Integer;  
Begin  
    Writeln('DI CHUYEN CON TRO TEP DEN PHAN TU THU N');  
    Writeln('-----');  
    Writeln;  
    Assign(f, 'SoNguyen.DAT');  
    Reset(f);  
    Write('-Can tim phan tu thu: ');  
    Readln(n);  
    Seek(f, (n-1));  
    Read(f,i);  
    Writeln;  
    Writeln('-Phan tu thu: ',n,' co tri = ',i);  
    Writeln;  
    Write('-Co can sua khong ? (C/K) ');  
    Readln(Ch);  
    If Ch in ['C','c'] Then  
        Begin  
            Seek(f,n-1);  
            Write('-Nhap so can sua: ');  
            Readln(i);  
            Write(f,i);  
        End;  
End;
```

```

Writeln;
Write('Da sua va ghi lai vao tep, bam <Enter> de xem lai');
Readln;
Seek(f,0);
While Not EOF(f) Do
  Begin
    Read(f,i);
    Writeln(i);
  End;
Writeln;
Seek(f,0);
Read(f,i);
Writeln('-Phan tu dau tien co tri= ',i);
Writeln;
Seek(f,Filesize(f)-1);
Read(f,i);
Writeln('-Phan tu cuoi cung co tri = ',i);
Writeln;
Writeln('  Bam <Enter>...');
Readln;
Close(f);
End.

```

7.5.8. Các thao tác khác với tệp

* Các hàm và thủ tục xử lý tệp

THỦ TỤC VÀ HÀM	Ý NGHĨA
Hàm FileSize (Biểntệp)	Cho số phần tử của biến tệp . Hàm sẽ cho giá trị 0 khi tệp rỗng
Hàm FilePos (Biểntệp)	Cho biết vị trí hiện tại của con trỏ tệp (phần tử đầu tiên của tệp được đánh số là 0)
Hàm EOF (Biểntệp)	Cho giá trị là True nếu con trỏ tệp ở vị trí cuối của tệp, trái lại là False. Hàm này thường dùng để kiểm tra xem đã đọc hết dữ liệu trong tệp chưa ?

Hàm EOLn (Biếntệp)	Cho giá trị là True nếu con trỏ tệp ở vị trí cuối dòng của tệp văn bản, trái lại là False. Hàm này thường dùng để kiểm tra xem đã đọc hết dòng văn bản trong tệp chưa ? Hàm này chỉ dùng với tệp văn bản
Truncate (Biếntệp);	Cắt tệp từ vị trí hiện tại của tệp (cửa sổ tệp).
Erase (Biếntệp)	Thủ tục này dùng để xóa tệp trên đĩa có tên đã gán vào biến tệp. Không được xóa tệp khi tệp đang mở .
Rename (Biếntệp,Str)	Thủ tục này dùng để đổi tên biến tệp trên đĩa thành tệp có tên mới kiểu String chứa trong Xâu kí tự Str . Không được đổi tên tệp khi tệp đang mở và tên mới không được trùng với tên tệp đã có trên đĩa đang làm việc . Ví dụ đổi tên tệp có tên là NTL02.pas thành NTL02.sl : Assign(Biến tệp,'NTL02.pas'); Rename(Biến tệp,'NTL02.sl');

Ví dụ 7.67: Tạo tệp TEST.INT có 10 số nguyên, sau đó hiển thị ra màn hình kích thước tệp, các phần tử của tệp.

```

Program Ham_Tep;
Var      f : File Of Integer;
         i,j,k,v : Integer;
Begin
    Writeln('SU DUNG CAC HAM TEP ');
    Writeln('-----');
    Assign(f,'TEST.INT');
    Rewrite(f);
    For i := 1 To 10 Do
        Write(f,i);
    Writeln;
    Writeln('Da tao tep TEST.INT co 10 so nguyen');
    Write('Bam <Enter> de xem noi dung tep TEST');
    Readln;
    Reset(f);
    While Not EOF(f) Do
        Begin

```

```
        Read(f,i);
        Writeln(i);
    End;
Writeln;
k:= Filesize(f);
Writeln('-Kich thuoc tap tin= ',k,' phan tu');
Write('Bam <Enter> de di chuyen den phan tu thu 5 ');
Readln;
Seek(f,5-1);
Read(f,i);
Writeln;
Writeln('-Phan tu thu 5 co tri = ',i);
v := Filepos(f);
Writeln;
Writeln('-Vi tri con tro tep hien gio o phan tu thu: ',v);
Writeln;
Write('Bam <Enter> de xem cac phan tu con lai ');
Readln;
While Not EOF(f) Do
    Begin
        Read(f,i);
        Writeln(i);
    End;
Writeln;
If EOF(f) Then
    Writeln('-Da ket thuc tep')
Else
    Writeln('-Chua ket thuc tep');
Writeln;
Write('    Bam <Enter> de ket thuc ');
Readln;
Close(f);
Erase(f);
End.
```

Ví dụ 7.68: Chương trình cắt tệp TEST.INT

```
Program Cat_Tep;
Var  f : File Of Integer;
     i,j : Integer;
Begin
    Writeln('CAT TEP BANG THU TUC TRUNCATE');
    Writeln('-----');
    Writeln;
    Assign(f, 'TEST.INT');
    Rewrite(f);
    For i := 1 To 10 Do
        Write(f,i);
    Writeln;
    Writeln('Da tao tep TEST.INT co 10 so nguyen');
    Write('Bam <Enter> de xem noi dung tep TEST');
    Readln;
    Reset(f);
    While Not EOF(f) Do
        Begin
            Read(f,i);
            Writeln(i);
        End;
    Reset(f);
    Writeln;
    Write('  Bam <Enter> de cat 5 phan tu ');
    Readln;
    Reset(f);
    For i := 1 To 5 Do
        Read(f,i);
    Truncate(f);
    Writeln;
    Writeln(' Da cat 5 phan tu cua tep');
    Write('Bam <Enter> de xem noi dung tep con lai ');
    Readln;
```



```
Reset(f);
While Not EOF(f) Do
  Begin
    Read(f,i);
    Writeln(i);
  End;
Writeln;
Write('  Bam <Enter> de ket thuc ');
Readln;
Close(f);
Erase(f);
End.
```

Ví dụ 7.68: Chương trình minh họa thủ tục xóa tệp

```
Program Xoa_Tep;
Var f : File Of String;
    Ten : String;
Begin
  Writeln('THU TUC ERASE, XOA TEP');
  Write('-Cho biet ten tap tin can xoa: ');
  Readln(Ten);
  Assign(f,Ten);
  {$I-}
  Reset(f);
  If IOResult <> 0 Then
    Writeln('  Khong co tep: ',Ten,' tren dia')
  Else
    begin
      Erase(f);
      Writeln;
      Write('  Da xoa tep, bam <Enter>. . . ');
    End;
  Readln;
  Close(f);
End.
```

*** Bẫy lỗi khi mở tệp:**

Đặt vấn đề

Khi thực hiện thao tác mở tệp để đọc / ghi có thể bị lỗi nếu:

- mở tệp chưa tồn tại trên đĩa, tên sai ...
- hết chỗ trên đĩa để ghi thêm vào tệp.

TurboPascal cung cấp chỉ thị để trình biên dịch thêm các mã lệnh thực hiện kiểm tra lỗi trong quá trình thực hiện đọc / ghi tệp mà dưới đây sẽ gọi ngắn gọn là chỉ thị kiểm tra I/O.

{ \$ I + } : bật (mở) việc kiểm tra I/O. Nếu gặp lỗi I/O chương trình sẽ báo lỗi và dừng lại. Đây là chế độ mặc định.

{ \$ I - } : tắt việc kiểm tra I/O. Chương trình không dừng dù có lỗi I/O. Tuy nhiên nó sẽ bỏ qua (treo, tạm dừng) tất cả các thủ tục vào/ra sau đó cho đến khi gặp lời gọi hàm *IOResult*. *IOResult* là hàm trả về mã lỗi khi thực hiện các thao tác đọc ghi ra đĩa. Nếu công việc thành công thì mã lỗi là 0. Trái lại, mã lỗi là một số khác không.

Sử dụng cơ chế kiểm tra I/O có thể thực hiện việc bẫy lỗi khi mở tệp, nhất là tiến hành *mở tệp mới an toàn*, tránh việc vô tình xóa mất tệp cùng tên khi mở tệp mới để viết vào bằng thủ tục *Rewrite*.

Các bước để mở tệp an toàn.

Tắt chế độ kiểm tra và thử mở tệp ra để đọc.

Gọi hàm *IOResult*.

Nếu *IOResult* bằng 0, nghĩa là mở tệp thành công tức là đã có tệp cùng tên. Cần thông báo cho người dùng biết để xử lý, nhập lại tên khác chẳng hạn.

Nếu *IOResult* khác không nghĩa là chưa có tệp nào có tên như thế. Có thể yên tâm tạo tệp bằng lệnh *Rewrite*.

Ví dụ 7.70:

```
Assign( biến tệp , tên tệp );
```

```
{ $ I - }
```

```
Reset( biến tệp );
```

```
{ $ I + }
```

```
If IOResult <> 0 then {không có tệp nào trùng tên}
```

```
Begin
```

```
Writeln(' Ban muon tao tep moi ? Enter=Yes ' );
```

```
if ReadKey = #13 then Rewrite( biến tệp )
```

```
else exit;
```

```
end;
```

```
Else
```

```
Begin
    Writeln(' Da co tep trung ten '); ....
End;
```

Việc tắt chế độ kiểm tra I/O bằng chỉ thị $\{ \$I - \}$ trước câu lệnh *Reset* là để tránh báo lỗi và dừng chương trình theo mặc định khi mở tệp chưa có trên đĩa.

7.5.9. Tệp văn bản

7.5.9.1. Ý nghĩa

Tệp văn bản là một kiểu tệp phổ biến, đã được định nghĩa sẵn, để lưu trữ xử lý các "văn bản". Khái niệm "văn bản" ở đây khác với khái niệm văn bản thông thường trong các phần mềm xử lý văn bản. "Văn bản" ở đây chỉ gồm các kí tự ASCII và được phân thành các dòng. Ngoài ra, không có bất cứ định dạng nào khác về trình bày như cỡ chữ, kiểu chữ, màu sắc, v.v.

7.5.9.2. Cú pháp

Để khai báo F là biến tệp văn bản ta viết :

Var F: Text;

Ví dụ 7.71: VAR F1, F2 : Text ;

7.5.9.3. Cấu trúc

Thành phần chính của tệp văn bản là các kí tự ASCII. Tuy nhiên, tệp văn bản được chia thành các dòng. Đánh dấu hết dòng bằng dấu *Eoln* (*end of line*). Đó là cặp kí tự CR (*carriage return*) - nhảy về đầu dòng, mã ASCII là 13 - và LF (*line feed*) - nhảy xuống dòng dưới, mã ASCII là 10.

Dấu hết tệp *Eof* đối với tệp văn bản là *Ctrl-Z* có mã ASCII là 26.

Trong môi trường DOS có thể dùng lệnh *Type* để hiển thị nội dung của tệp văn bản lên màn hình.

Tệp văn bản không phải là tệp các kí tự *File of Char*. *File of Char* khác với tệp kiểu *Text* ở chỗ nó không nhận biết cặp kí tự CR, LF như dấu hết dòng mà coi như hai kí tự ASCII bình thường.

Ví dụ 7.72: đoạn văn bản sau :

Nguyen Thanh Lam

08-08-2002

Het

được chứa trong tệp văn bản thành một dãy :

Nguyen Thanh Lam	CR LF	08-08-2002	CR LF	Het	Eof
------------------	----------	------------	----------	-----	-----

Các thủ tục **Assign**, **Rewrite**, **Reset**, **Write**, **Read**, **Close**, **Erase**, **Rename** đều dùng được cho tệp văn bản. Ngoài ra còn có thêm thủ tục **Append(biến tệp)** dùng để mở tệp văn bản và cho phép ghi thêm dữ liệu vào cuối tệp.

Đối với tệp văn bản, không thể đồng thời vừa ghi vừa đọc dữ liệu như tệp có định kiểu.

Để ghi dữ liệu, trước tiên phải khởi tạo tệp bằng lệnh **Rewrite** hay mở tệp và đưa trở về cuối tệp bằng lệnh **Append**. Sau đó ghi dữ liệu vào tệp bằng thủ tục **Write** hay **Writeln**.

Để đọc dữ liệu một tệp đã có, trước tiên ta phải mở tệp bằng lệnh **Reset**. Sau đó đọc dữ liệu bằng thủ tục **Read** hay **Readln**.

Nếu mở tệp bằng **Rewrite** hoặc **Append** thì không thể đọc được bằng **Read** và **Readln**. Nếu mở tệp bằng **Reset** thì không thể ghi được bằng **Write** hay **Writeln**.

7.5.9.4. Ghi dữ liệu vào tệp văn bản

Có thể ghi các giá trị kiểu *nguyên*, *thực*, *logic*, *xâu kí tự* ra tệp văn bản bằng các lệnh *Write*. Các thủ tục *Write* viết ra tệp văn bản sẽ tự động tính các biểu thức và chuyển đổi các giá trị sang dạng biểu diễn xâu kí tự thông thường để có thể ghi vào tệp văn bản.

Write (*biến tệp* , *biểu thức 1* , *biểu thức 2* , ... , *biểu thức n*) ;

Writeln (*biến tệp* , *biểu thức 1* , *biểu thức 2* , ... , *biểu thức n*) ;

Writeln (*biến tệp*) ;

Sau các biểu thức có thể kèm phần định dạng (quy cách) in ra.

Tác dụng của các câu lệnh trên hoàn toàn tương tự như trong trường hợp xuất ra màn hình đã trình bày ở Chương 3. Chỉ khác là ở đây không phải in ra màn hình mà là viết ra tệp văn bản ứng với *biến tệp*. Về bản chất, màn hình cũng giống như một tệp văn bản.

Ví dụ 7.73: Write(F, 3, 10:4, 'a':2, 'Text', 4.5:6:2);

sẽ ghi vào tệp thành dãy như sau (Dấu ~ hiểu là một ký tự trắng):

3~~10~aText~~4.50

Chương trình dưới đây sẽ tạo tệp văn bản T1.TXT :

```
Program VIDU_7_73;  
Var   F: Text;  
      A : Integer;  
      B : Real;  
  
Begin  
      A:=100;  
      B:=1234.5;  
      Assign(F, 'T1.TXT') ;  
      Rewrite(F) ;
```

```
Write(F, 'Ket qua=' :10, A:5, B:7:2);  
Close(F);
```

End.

Nội dung của tệp T1.TXT là :

~~Ket qua==100~123.45

Như vậy, cách ghi dữ liệu vào tệp văn bản hoàn toàn giống như khi in dữ liệu lên màn hình.

Thủ tục WRITELN cũng có công dụng như WRITE, nhưng ghi xong dữ liệu thì đưa con trỏ tệp xuống dòng dưới. Đặc biệt, lệnh Writeln(F); không ghi gì cả, chỉ đưa con trỏ tệp xuống dòng.

Nội dung của các tệp văn bản tạo bằng Pascal hoàn toàn có thể xem được bằng lệnh Type của MSDOS, bằng Norton hay bằng chính Turbo Pascal, ...

7.5.9.5. Đọc dữ liệu của tệp văn bản

Các thủ tục *Read* quen biết có thể đọc từ tệp văn bản rồi gán cho các biến không những các kí tự mà cả các kiểu dữ liệu khác nữa: số nguyên, thực, logic Thủ tục *Read* sẽ tự động chuyển đổi các xâu kí tự biểu diễn dữ liệu thành giá trị có kiểu tương ứng của biến sẽ được gán

Read (*biến tệp* , *biến 1* , *biến 2* , ... , *biến n*);

Readln (*biến tệp* , *biến 1* , *biến 2* , ... , *biến n*);

Readln (*biến tệp*);

Tác dụng: hoàn toàn giống như các lệnh đọc dữ liệu từ bàn phím đã quen biết, chỉ khác là ở đây không phải nhập dữ liệu từ bàn phím mà các mục dữ liệu đã viết sẵn trong tệp văn bản ứng với *biến tệp*.

Tuỳ theo kiểu của *biến i* mà lệnh *Read* thực hiện đọc một dãy kí tự từ tệp văn bản, chuyển đổi nó thành một *giá trị trực tiếp* thuộc kiểu dữ liệu của *biến i* và gán cho biến này.

Quy định về phân cách các mục dữ liệu kiểu số, kiểu kí tự, kiểu xâu kí tự đã được trình bày chi tiết trong Chương 3.

Chú ý: Các biến1, biến2, ..., biếnN phải có kiểu dữ liệu phù hợp với dữ liệu cần đọc tại vị trí tương ứng trong tệp.

Ví dụ 7.74:

Nếu tệp T1.TXT có nội dung như sau:

~~Ket qua==100~123.45

thì để đọc lại các dữ liệu này, ta phải khai báo:

```
Var St :String[10];  
i: Integer ; Z : Real;
```

Và dùng các lệnh:

```
Reset(F);  
Read(F, St, i, Z);
```

Giá trị của St, i, Z sẽ là: St='~~Ket qua=', i=100, Z=123.45.

Nếu khai báo St có kiểu String[9] thì sẽ bị lỗi vì sau khi đọc xong 9 ký tự đầu, máy sẽ đọc tiếp giá trị ==100 cho biến nguyên i, nhưng vì giá trị này bắt đầu là dấu = nên không đổi ra số nguyên được.

Thủ tục READLN(biếntệp, biến1, biến2, ..., biếnN) đọc dữ liệu cho các biến xong sẽ đưa trở tệp xuống đầu dòng dưới.

Đặc biệt, lệnh READLN(biếntệp); không đọc gì cả, chỉ đưa con trỏ tệp xuống dòng.

7.5.9.6. Các hàm, thủ tục chuẩn khác cho tệp văn bản

Các thủ tục *Seek*, *FileSize*, *FilePos* không áp dụng được cho tệp văn bản vì tệp văn bản có cấu trúc là dãy các dòng kích thước (độ dài) khác nhau.

Dưới đây là các hàm thủ tục áp dụng cho tệp *Text*.

- **Eof**(*biến tệp*): boolean ;

Kiểm tra đã ở cuối tệp chưa.

- **Eoln** (*biến tệp*): boolean ;

Kiểm tra đã ở cuối dòng chưa.

- **Append** (*biến tệp*);

Mở tệp văn bản để viết vào, của sổ tệp đặt tại cuối tệp.

- **SeekEoln**(*biến tệp*): boolean ;

Tương tự như *Eoln* nhưng trước khi kiểm tra nó nhảy qua các dấu cách và dấu Tab. *SeekEoln* cho kết quả *True* nếu trên phần còn lại của dòng, kể từ vị trí hiện tại, không còn mục dữ liệu nào nữa.

- **SeekEof**(*biến tệp*): boolean ;

Tương tự như *Eof* nhưng trước khi kiểm tra nó nhảy qua các dấu cách, dấu Tab và dấu xuống dòng. *SeekEof* cho kết quả *True* nếu từ vị trí hiện tại đến cuối tệp không còn mục dữ liệu nào nữa.

- **Flush**(*biến tệp*);

Xả hết nội dung bộ đệm của một tệp kiểu văn bản đang được mở để viết vào.

F là một biến kiểu tệp văn bản *Text*, được mở bằng *Rewrite* hoặc *Append* để viết ra. Khi đó lời gọi *Flush* sẽ xả hết phần nội dung còn đọng trong bộ đệm, viết nốt ra tệp F. Thủ tục này được dùng để đảm bảo rằng mọi lệnh viết ra tệp F đều được thực hiện trọn vẹn, không bỏ sót dữ liệu.

Flush không có tác dụng nếu tệp F được mở để đọc.

- **SetTextBuf**(*biến tệp*, *vùng đệm*, *kích thước*);

Gán một vùng đệm I/O cho tệp văn bản. Tham biến *Vùng đệm* là một vùng trong bộ nhớ dùng làm trung gian, tạm chứa dữ liệu trong các thao tác truy cập đọc ra hoặc viết vào tệp trên ổ đĩa. Tham trị *kích thước* là tùy chọn, có thể không có mặt.

Thủ tục này thường dùng để tăng kích thước vùng đệm nhằm đẩy nhanh tốc độ trao đổi dữ liệu với ổ đĩa. Ta đã biết tốc độ truy cập đĩa chậm hơn nhiều so với truy cập bộ nhớ trong. Nếu có một vùng đệm thì các thao tác xử lý đọc / viết tạm ra đây. Khi đầy vùng đệm mới tiến hành thao tác đọc / viết thực sự ra đĩa, không làm lắt nhắt nhiều lần.

SetTextBuf không nên gọi cho tệp đang mở và đã có thao tác trao đổi dữ liệu. Nó có thể được gọi ngay sau *Reset*, *Rewrite*, hay *Append*. Nếu gọi *SetTextBuf* cho tệp đang mở thì khi có thao tác I/O, có thể bị mất dữ liệu do kích thước của vùng đệm thay đổi.

Ví dụ 7.75: Minh họa cách dùng *SetTextBuf*

```
Var   F: Text;
      Ch: Char;
      Bodem: array[1..4095] of Char; { 4K buffer }
begin
  {Lấy tên tệp từ tham số dòng lệnh}
  Assign(F, ParamStr(1));
  {Tăng bộ đệm lớn để đọc nhanh}
  SetTextBuf(F, Bodem);
  Reset(F);
  {Đưa toàn bộ nội dung tệp văn bản ra màn hình}
  while not Eof(f) do
    begin
      Read(F, Ch);
      Write(Ch);
    end;
end.
```

Chú ý: Thủ tục *Seek* và các hàm *FileSize*, *FilePos* không dùng cho tệp văn bản.

7.5.9.7. Các tệp văn bản ngầm định

Trong Pascal có hai biến tệp văn bản đã được khai báo sẵn là **Input** và **Output**, tức là máy đã ngầm khai báo:

```
Var   Input, Output: Text;
```

Input thường là bàn phím còn Output thường là màn hình.

Lệnh *Readln*(Input, x); được viết tắt thành *Readln*(x);

Lệnh *Writeln*(Output, x); được viết tắt thành *Writeln*(x);

Máy in cũng là một tệp văn bản, được ngầm khai báo với tên là *LST*. Để in các biểu thức *bt1*, *bt2*, ..., *btN* ra máy in, ta phải khai báo sử dụng thư viện chuẩn *PRINTER*, và dùng lệnh:

```
Write(LST, bt1, bt2, ... , btN);
```

Ví dụ 7.76: Cho tệp văn bản tên là T2.TXT và có nội dung là:

```
dong=6
cot =7
0 8 8 -2 6 11 1
8 0 2 0 7 0 2
8 2 0 11 12 9 3
-2 0 11 0 -7 9 4
6 7 12 -7 0 6 5
11 0 9 9 6 0 6
```

Hãy đọc tệp này đưa vào một ma trận A và in ma trận A lên màn hình. Số dòng và số cột của ma trận A được ghi ở hai dòng đầu tiên trong tệp T2.TXT.

```
Program VIDU_7_76;
{ vi du ve File Van ban }
uses crt;
Type MANG = array[1..20,1..20] of integer ;
Var A : MANG;
    N, M : integer;
    F : Text;
Procedure Nhap;
Var i,j : Byte;
    st: string[5];
Begin
    Assign(F, 'T2.TXT');
    Reset(F);
    Readln(F, St, N);
    Readln(F, St, M);
    For i:=1 to N do
        begin
            For j:=1 to M do
                Read(F, A[i,j]);
            Readln(F);
        end;
    Close(F);
End;
```



```
Procedure    InMatran;
Var  i, j : Integer;
Begin
    For i:=1 to N do
        begin
            for j:=1 to M do
                write( A[i,j]:4);
            writeln;
        end;
    End;
BEGIN
    Clrscr;
    Nhap;
    InMatran;
    Readln;
END.
```

Ví dụ 7.77: Lập bảng lượng giác từ 1^0 đến 89^0 và ghi ra tệp BLGIAC.DAT

```
Program  Bang_Luong_Giac;
CONST   g='|';
Var  F : Text;
      k : Integer;
      Rad,S,C,T,CT : Real;
{-----}
Function  taobansao(Dong:Char; T:Integer):String;
{Chương trình con này tạo đoạn thẳng bởi các dấu bằng =}
Var j : Integer;
      Tam :String[80];
Begin
    Tam := ' ';
    For j := 1 To T Do
        Tam := Tam + Dong;
    Replicate := Tam;
End;
{-----}
```

```
BEGIN
    Assign(f, 'BLGIAC.DAT');
    Rewrite(f);
    Writeln(f, '                * BANG LUONG GIAC *');
    Writeln(f);
    Writeln(f, taobansao(#61,58));
    Writeln(f,g, ' DO ',g, ' RADIAN ',g, ' SIN ',g, ' COSIN ',
        g, ' TAN ',g, ' COTANG ',g);
    Writeln(f, taobansao(#61,58)); {tạo đoạn thẳng}
    For k := 1 To 89 Do
        Begin
            Rad := k * Pi /180;
            S := Sin(Rad);
            C := Cos(Rad);
            T := S/C;
            CT := 1/t;
            writeln(f,g,k:2,g,Rad:10:8,g,S:10:8,g,C:10:8,g,
T:10:6,g,CT:10:6,g);
        End;
    Writeln(f, taobansao(#61,58));
    Close(f);
END.
```

{Hãy mở tệp 'BLGIAC.DAT' trong thư mục hiện hành để xem kết quả hoặc để tiện quan sát bạn có thể thêm phần ghi ra màn hình}

- **Tệp không định kiểu**

- * **Tệp không định kiểu:**

Tệp không định kiểu - *Untyped File* - là tệp không định rõ kiểu của phần tử khi khai báo, coi dữ liệu chính là các mã nhị phân 0,1 không cần biết nó biểu thị cái gì. Đơn vị xử lý là từng khối bit, có kích thước do ta quy định, gọi là các Record. Lưu ý rằng record ở đây không phải là kiểu bản ghi như đã biết.

Tệp không định kiểu chủ yếu dùng để tăng tốc độ thao tác vào ra dữ liệu.

- * **Cú pháp**

Khai báo: dùng từ khoá FILE

Var biến tệp : **File** ;

Truy cập: Các thủ tục *Reset*, *Rewrite* đối với tệp không định kiểu cần có thêm tham trị *kích thước Rec*. Tham trị *kích thước Rec* là một số nguyên (kiểu *Word*) để ấn định kích thước của khối dữ liệu đơn vị (record) trong các thao tác tệp. Kích thước này tính theo byte.

Reset (*biến tệp* , *kích thước Rec*);

Rewrite (*biến tệp* , *kích thước Rec*);

Ví dụ 7.78:

Reset(F,4); đọc ra từng khối 4 byte một.

Rewrite(F,2); viết vào từng khối 2 byte một.

Nếu tham số này không có mặt thì record có kích thước mặc định là 128 byte.

*** Các hàm, thủ tục khác.**

Ngoài ra, để tăng nhanh tốc độ xuất nhập dữ liệu, còn có hai thủ tục dành riêng cho tệp không định kiểu là **BlockRead** và **BlockWrite**.

BlockRead(*biến tệp* , *vùng đệm* , *số Rec* , *kết quả*);

Vùng đệm: vùng đệm trong bộ nhớ, là một tham biến, nơi chứa dữ liệu đọc vào

Số Rec : tham trị kiểu *Word*, là số khối dự định đọc.

Kết quả : tham biến kiểu *Word*, là số khối thực đọc được.

Tác dụng.

BlockRead đọc một lượng *số Rec* khối bit từ tệp ứng với *biến tệp* rồi ghi vào bộ nhớ, bắt đầu từ byte thứ nhất của *Vùng đệm*. Số khối bit thực sự đọc được được trả về trong biến *kết quả*. Lưu ý rằng số khối bit thực sự đọc được có thể bé hơn *số Rec* dự định đọc nếu đã đến cuối tệp.

Nếu không có tham biến *kết quả* thì khi số khối thực sự đọc được không bằng *số Rec* sẽ có lỗi I/O xảy ra.

Số byte đọc được nhiều nhất là *số Rec * kích thước Rec*, ở đây *kích thước Rec* là kích thước của khối. Kích thước khối là số lượng byte trong một khối, được ấn định khi mở tệp như đã trình bày ở trên.

Nếu *số Rec * kích thước Rec* > 65,535 (64K) thì sẽ có lỗi.

BlockWrite(*biến tệp* , *vùng đệm* , *số Rec* , *kết quả*);

Vùng đệm: vùng đệm trong bộ nhớ, là một tham biến, nơi chứa dữ liệu để xuất ra.

Số Rec : tham trị kiểu *Word*, là số khối dự định xuất ra.

Kết quả : tham biến kiểu *Word*, là số khối thực sự xuất ra được.

Ý nghĩa của các tham số và công dụng của **BlockWrite** cũng tương tự như **BlockRead** ở trên. Chỉ có thao tác truy cập ở đây là *viết từ bộ nhớ ra tệp*.

Ví dụ 7.79:

Chương trình dưới đây minh họa cách dùng *BlockRead*, *BlockWrite*. Nó thực hiện sao chép tệp nhanh, đơn giản không kiểm tra lỗi.

```
Program CopyFile;
Var   Fnguồn, Fdịch: file;
      Byte_doc, byte_ghi: Word;
      Buf: array[1..2048] of Char;
Begin
    Assign(Fnguồn, 'nguồn.txt');
    Reset(Fnguồn, 1);    { đọc ra từng khối mỗi lần 1 byte }
    Assign(Fdịch, 'dịch.txt');
    Rewrite(Fdịch, 1);   { ghi ra từng khối mỗi lần 1 byte }
    Writeln('Copy ', FileSize(Fnguồn), '
bytes từ file nguồn.txt sang file dịch.txt');
    Repeat
        BlockRead(Fnguồn, Buf, SizeOf(Buf), byte_doc);
        BlockWrite(Fdịch, Buf, byte_doc, byte_ghi);
    Until (byte_doc = 0) or (byte_ghi <> byte_doc);
{Việc đọc ghi sẽ dừng nếu file nguồn.txt rỗng, tức byte_doc=0
hoặc số byte đọc và ghi là khác nhau (NumWritten <> NumRead)}
    Close(Fnguồn);
    Close(Fdịch);
    Readln;
End.
```

BÀI TẬP CHƯƠNG 7

Bài 7.1: Cho dãy số gồm n phần tử ($n \leq 15$), giá trị từng phần tử là số thực. Thực hiện các yêu cầu:

1. Nhập từ bàn phím số phần tử và giá trị của từng phần tử;
2. Tính tổng và trung bình cộng các phần tử của dãy;
3. In dãy đã cho ra màn hình và kết quả tìm được.

Bài 7.2: Cho dãy số gồm n phần tử ($n \leq 20$), giá trị từng phần tử là số nguyên. Thực hiện các yêu cầu sau:

1. Nhập từ bàn phím số phần tử và giá trị của từng phần tử;
2. Tính tổng các phần tử âm của dãy;
3. In ra màn hình dãy số nhập, dãy chứa toàn số âm và các kết quả tìm được.

Bài 7.3: Cho dãy số gồm n phần tử ($n \leq 20$), giá trị từng phần tử là số nguyên. Thực hiện các yêu cầu sau:

1. Nhập từ bàn phím số phần tử và giá trị của từng phần tử;
2. Tính tổng và trung bình cộng các phần tử chẵn của dãy;
3. In dãy đã cho, dãy chứa các phần tử chẵn ra màn hình và kết quả tìm được.

Bài 7.4: Cho dãy số gồm n phần tử ($n \leq 15$), giá trị từng phần tử là số thực. Thực hiện các yêu cầu sau:

1. Nhập từ bàn phím số phần tử và giá trị của từng phần tử;
2. Tìm phần tử có giá trị lớn nhất và vị trí xuất hiện đầu tiên của nó trong dãy;
3. In dãy đã cho ra màn hình và các kết quả tìm được.

Bài 7.5: Cho dãy số gồm n phần tử ($n \leq 15$), giá trị từng phần tử là số thực. Thực hiện các yêu cầu sau:

1. Nhập từ bàn phím số phần tử và giá trị của từng phần tử;
2. Kiểm tra điều kiện n ($0 < n \leq 15$), nếu không thỏa mãn yêu cầu nhập lại n cho đến khi có n thỏa mãn điều kiện.
2. Tìm phần tử có giá trị tuyệt đối bé nhất và vị trí xuất hiện cuối cùng của nó trong dãy;
3. In dãy đã cho ra màn hình và các kết quả tìm được.

Bài 7.6: Cho dãy số gồm n phần tử ($n \leq 15$), giá trị từng phần tử là số thực. Thực hiện các yêu cầu sau:

1. Nhập từ bàn phím số phần tử và giá trị của từng phần tử;
2. Sắp xếp dãy đã cho theo chiều tăng dần về giá trị của từng phần tử;
3. In dãy số nhập vào và dãy sau khi sắp ra màn hình.

Bài 7.7: Cho dãy số gồm n phần tử ($n \leq 20$), giá trị từng phần tử là số thực. Thực hiện các yêu cầu sau:

1. Nhập từ bàn phím số phần tử và giá trị của từng phần tử;
2. Tìm phần tử nhỏ nhất trong dãy và chỉ ra vị trí của tất cả các phần tử đạt giá trị nhỏ nhất (nếu có);
3. In dãy số nhập ra màn hình và các kết quả tìm được.
4. Thêm vào đoạn chương trình câu hỏi đáp: Có tiếp không (C/K) ?
Nếu gõ C hoặc c thì chương trình thực hiện lại từ đầu

Bài 7.8: Cho một ma trận chữ nhật $m \times n$ phần tử ($m, n \leq 5$), giá trị từng phần tử là số thực. Thực hiện các yêu cầu sau:

1. Nhập từ bàn phím số hàng, số cột và giá trị từng phần tử của ma trận
2. Tính tổng các phần tử của ma trận
3. Tìm phần tử lớn nhất trong ma trận.
4. In ma trận đã cho dưới dạng bảng ra màn hình và các kết quả tìm được.

Bài 7.9: Cho một ma trận chữ nhật $m \times n$ phần tử ($1 \leq m, n \leq 5$), các phần tử có giá trị nguyên. Thực hiện các yêu cầu sau:

1. Nhập từ bàn phím số hàng, số cột và giá trị từng phần tử của ma trận
2. Tính tổng các phần tử âm và trung bình cộng của chúng
3. In ma trận nhập và các kết quả tính.
4. Kiểm tra xem ma trận có phải là ma trận vuông hay không. Nếu là ma trận vuông hãy tính tổng các phần tử trên đường chéo chính. Nếu không phải là ma trận vuông hãy in dòng thông báo: Ma trận nhập không phải là ma trận vuông.

Bài 7.10: Cho ma trận vuông $n \times n$ phần tử ($1 \leq n \leq 5$), các phần tử có giá trị nguyên. Thực hiện các yêu cầu sau:

1. Nhập từ bàn phím số hàng, cột và giá trị từng phần tử của ma trận
2. Tìm phần tử có giá trị nhỏ nhất về giá trị tuyệt đối của ma trận và vị trí của nó.
3. Tính tổng các phần tử trên đường chéo chính của ma trận
4. In ma trận đã cho dưới dạng bảng ra màn hình và các kết quả tìm được.

Bài 7.11: Cho ma trận vuông $n \times n$ phần tử ($1 \leq n \leq 5$), các phần tử có giá trị nguyên. Thực hiện các yêu cầu sau:

1. Nhập từ bàn phím số hàng, cột và giá trị từng phần tử của ma trận
2. Tìm phần tử nhỏ nhất trên hàng k của ma trận và vị trí của nó - k được đọc vào từ bàn phím
3. Tìm phần tử có giá trị tuyệt đối nhỏ nhất trên đường chéo chính của ma trận.
4. In ma trận đã cho dưới dạng bảng ra màn hình và các kết quả tìm được.

Bài 7.12: Cho một ma trận chữ nhật $m \times n$ phần tử ($m, n \leq 5$), giá trị từng phần tử là số thực. Thực hiện các yêu cầu sau:

1. Nhập từ bàn phím số hàng, số cột và giá trị từng phần tử của ma trận
2. Đếm số phần tử chẵn âm của ma trận
3. In ma trận đã cho dưới dạng bảng ra màn hình và các kết quả tìm được.
4. Kiểm tra xem ma trận nhập vào có phải là ma trận đơn vị hay không?

Bài 7.13: Viết chương trình quản lý điểm của một lớp học gồm có các chức năng sau:

1. Nhập hồ sơ của mỗi học sinh gồm: Họ và tên, năm sinh, điểm trung bình học kỳ một, điểm trung bình học kỳ hai.
2. In danh sách các học sinh của lớp có điểm trung bình cả năm từ 5 điểm trở lên và theo thứ tự giảm dần của điểm trung bình cả năm.
3. In danh sách các học sinh phải thi lại (điểm trung bình cả năm < 5) theo thứ tự abc của tên.

Bài 7.14: Viết chương trình nhập vào từ bàn phím Họ và tên, sau đó in phần tên lên màn hình. Ví dụ: nhập 'Nguyễn Thanh Lam' và in ra 'Lam'.

Bài 7.15: Nhập vào một xâu ký tự gồm các từ. Viết chương trình xóa bỏ bớt các dấu trống giữa các từ sao cho các từ chỉ cách nhau ít nhất một dấu trống. , ví dụ nhập 'Cao đang Dien luc' thì in ra 'Cao đang Dien luc'

Bài 7.16: Viết chương trình thực hiện trò chơi sau:

Người chơi nhập một số k trong phạm vi từ 1 đến 9

Tạo một tập S gồm ba số ngẫu nhiên trong phạm vi từ 1 đến 9

Kiểm tra xem k có thuộc tập S không?. Nếu thuộc thì người chơi thắng, ngược lại là thua. In k và tập S lên màn hình.

Hướng dẫn: Trong thư viện CRT có hàm $Random(n)$ trả về một số ngẫu nhiên j thuộc phạm vi: $0 \leq j < n$.

Bài 7.17: Mỗi phân số được mô tả như sau :

Type PhanSo = Record

tu, mau : Integer;

end;

Nhập hai phân số từ bàn phím, tạo một tệp chứa hai phân số đó và hai phân số là tổng, hiệu của chúng. Đọc bốn phân số đó từ tệp và in lên màn hình.

Bài 7.18: Nhập số nguyên dương N ($0 < N < 20$) và một dãy N số nguyên : A_1, A_2, \dots, A_N . Ghi dãy số đó vào tệp DL.DAT. Đếm trong tệp DL.DAT có bao nhiêu số chẵn. Đọc các số lẻ từ tệp DL.DAT và in lên màn hình.

Bài 7.19: Để quản lý Họ tên, các điểm Toán, Lý và Điểm trung bình của sinh viên, ta mô tả kiểu Ksvien như sau :

Type Ksvien= Record

```
Hoten: String[20];  
Toan, Ly, Dtb : Real;  
end;
```

Hãy nhập một danh sách sinh viên gồm Họ tên, điểm Toán và điểm Lý, rồi tính Điểm trung bình: $Dtb = (Toan + Ly) / 2$, lưu vào tệp QLY.DAT. Quá trình nhập kết thúc khi nhập Họ tên là rỗng (tức không nhập gì cả, cứ Enter).

Đọc danh sách sinh viên từ tệp QLY.DAT và in danh sách lên màn hình.

Chép danh sách sinh viên vào tệp QLY.IDX sao cho các phần tử của QLY.IDX được sắp xếp theo trật tự giảm của điểm trung bình.

Đọc dữ liệu của tệp QLY.IDX và ghi vào tệp văn bản tên là QLY.TXT theo dạng:

STT	Họ và tên	Điểm trung bình
1	Nguyen Thao Nguyen	8.5
2	Nguyen Thanh Lam	8.0
..

Bài 7.20: Dùng hệ soạn thảo của Turbo Pascal để tạo một tệp văn bản có tên là MT.DAT chứa hai ma trận vuông cấp 3 là A và B có các phần tử là các số nguyên. Lấy dữ liệu từ tệp MT.DAT để tính ma trận $C = A + B$. Ghi ma trận C vào cuối tệp MT.DAT. Đọc các ma trận A, B, C từ tệp MT.DAT và in lên màn hình.

Chương 8

CHƯƠNG TRÌNH CON

8.1. Các khái niệm

8.1.1. Khái niệm về chương trình con

Chương trình là một dãy các lệnh được xây dựng cho máy tính theo một trật tự xác định nhằm hoàn thành một công việc nào đó. Công việc này có thể chia ra thành nhiều công việc nhỏ và mỗi công việc nhỏ này lại được tổ chức như những chương trình. Mỗi chương trình này cũng có thể được chia nhỏ tiếp, ta gọi đó là các chương trình con.

Chương trình con là một đoạn chương trình thực hiện một nhiệm vụ riêng được khai báo trước khi sử dụng. Mỗi một chương trình con có thể là chương trình con của chương trình con khác. Chương trình ở mức ngoài cùng (được gọi bởi người sử dụng) là chương trình chính.

Khi một chương trình con được khai báo thì nó có thể được sử dụng nhiều lần thông qua các lời gọi chương trình con.

Có hai loại chương trình con đó là hàm (function) và thủ tục (procedure).

Vậy sử dụng chương trình con có ưu điểm gì?

Khi viết những đoạn chương trình nhỏ thì chương trình con không thực sự hữu ích nhưng khi viết các chương trình lớn ta sẽ thấy có những đoạn trình lặp đi lặp lại nhiều lần với các dữ liệu đầu vào khác nhau. Chương trình con sẽ giúp ta tránh được việc viết lặp nhiều lần bằng cách tổ chức chương trình thành nhiều chương trình con.

Đặc biệt là xây dựng những chương trình hoàn chỉnh kích thước lớn và phức tạp thì tất yếu sẽ chia thành nhiều mô đun nhỏ độc lập nhau, giao cho mỗi nhóm phát triển độc lập. Sau đó phối hợp các chương trình con đó lại sẽ cho ta một chương trình hoàn chỉnh.

Việc tổ chức chương trình thành nhiều chương trình con sẽ giúp chương trình dễ hiểu hơn, cấu trúc rõ ràng nhờ đó việc quản lý, sửa lỗi, bảo trì cũng dễ dàng hơn.

Trong Pascal có một số chương trình con đã được xây dựng sẵn và được tổ chức thành thư viện các chương trình con. Một số hàm mẫu như: abs, sqrt, sqr, sin, cos, ... hay một số thủ tục như: clrscr, exit, break, Người lập trình có thể xây dựng thêm các chương trình con khác, kế thừa kết quả trước đây, giảm chi phí và giảm công sức trong việc viết chương trình.

8.1.2. Một số khái niệm

a) Biến toàn cục (global variable) hay còn gọi là biến chung:

Biến toàn cục là biến được khai báo ở đầu chương trình, nó được sử dụng bên trong chương trình chính và cả bên trong chương trình con. Biến toàn cục sẽ tồn tại trong suốt quá trình thực hiện của chương trình.

b) Biến địa phương (local variable) hay còn gọi là biến riêng:

Biến địa phương là biến khai báo ở đầu chương trình con, và nó chỉ được sử dụng bên trong thân chương trình con hoặc bên trong chương trình con khác nằm trong nó.

Biến địa phương chỉ tồn tại khi chương trình con đang hoạt động, nghĩa là biến địa phương sẽ được cấp phát một vùng nhớ khi chương trình con được thi hành, và sẽ giải phóng vùng nhớ đó ngay sau khi chương trình con kết thúc.

**Chú ý:* Khi trong một chương trình tồn tại biến địa phương và biến toàn cục trùng tên nhau thì khi thực hiện chương trình con, biến địa phương của chương trình con đó sẽ được ưu tiên.

c) Tham số thực sự (actual parameter) là một tham số mà nó có thể là một biến toàn cục, một biểu thức hoặc một giá trị số (cũng có thể là biến cục bộ khi sử dụng chương trình con lồng nhau) mà ta dùng chúng khi truyền giá trị cho các tham số hình thức tương ứng của chương trình con.

d) Tham số hình thức (formal parameter) là các biến được khai báo ngay sau Tên chương trình con, nó dùng để nhận giá trị các tham số thực truyền đến.

8.1.3. Sử dụng chương trình con

8.1.3.1. Khai báo chương trình con

Các chương trình con cần phải được khai báo trước khi có thể sử dụng. Trong Pascal các chương trình con còn phải được triển khai đầy đủ khi khai báo. Vị trí khai báo chương trình con là ở phần sau phần khai báo thư viện, nhãn, hằng kiểu, biến và ngay trước phần thân chương trình chính.

Cấu trúc chung một chương trình con gồm các phần sau:

- (i) Phần tiêu đề chương trình con: nhằm khai báo
 - Loại chương trình con: hàm (function) hay thủ tục (procedure)
 - Tên chương trình con: rất quan trọng và nó tuân theo quy tắc đặt tên.
- Các tham số hình thức: là khai báo các đối tượng hình thức sẽ tham gia vào chương trình con
 - Kiểu kết quả trả lại nếu là hàm.
- (ii) Phần khai báo cho chương trình con: nhằm khai báo thư viện các nhãn, các hằng, kiểu các biến chỉ sử dụng trong chương trình con đó. Quy cách khai báo giống như khai báo trong chương trình chính.

(iii) Phần thân chương trình con: bao gồm các lệnh bao trong cặp từ khóa

(begin ..end;)

Begin

<các câu lệnh>

End;

Ví dụ 8.1: Viết chương trình tính giá trị biểu thức $F := \begin{cases} a & \text{nếu } a \geq 0 \\ |a| & \text{nếu } a < 0 \end{cases}$

với a được nhập từ bàn phím.

```
Program tinh_F_a;
Uses crt;
Var a, F: real;
Procedure GT(a: real): real;      {Khai báo chương trình con (hàm)}
Begin
    If a>=0 then GT:=a
    Else      GT:= abs(a);
End;                               {Bắt đầu thân chương trình chính}
BEGIN
    Clrscr;                        {Lời gọi hàm, giá trị hàm được gán
    write(' Nhập a = ');readln(a); cho biến F}
    F:=GT(a);

    writeln('Gia tri F = ',F:8:2);
    readln;
END.
```

Ta cũng có thể sử dụng chương trình thủ tục để tính giá trị F như sau:

```
Program tinh_F_a;
Uses crt;
Var a, F: real;
Procedure tinh(var F:real;a: real); {Khai báo thủ tục}
Begin
    If a>=0 then F:=a
    Else      F:= abs(a);
End;                               {Bắt đầu thân chương trình chính}
BEGIN
    Write(' Nhập a = ');readln(a);
    tinh(F,a);                     {Lời gọi thủ tục}
    writeln(' Gia tri F = ', F:8:2);
    readln;
END.
```

8.1.3.2. Lời gọi chương trình con (thủ tục và hàm)

Để chương trình con được thi hành, ta phải có lời gọi đến chương trình con, lời gọi chương trình con thông qua tên chương trình con và danh sách tham số tương ứng (nếu có) theo các quy tắc:

- Trong thân chương trình chính hoặc thân chương trình con, ta chỉ có thể gọi tới các chương trình con trực thuộc nó.
- Trong chương trình con, ta có thể gọi các chương trình con ngang cấp đã được thiết lập trước đó hoặc cũng có thể gọi lại chính nó (chương trình con đệ qui).

Trong lời gọi hàm, thủ tục thì các danh sách tham số thực sự phải tương đương ứng một - một với danh sách tham số hình thức: đúng số lượng, thứ tự và kiểu dữ liệu.

Tại một thời điểm trong chương trình chính, khi gặp lời gọi chương trình con thì chương trình chính tạm dừng. các tham số hình thức sẽ được tạm gán các giá trị hiện tại của tham số thực sự và các lệnh xử lý trong chương trình con được tiến hành. Khi kết thúc chương trình con, điều khiển sẽ được trả về cho chương trình chính và công việc của chương trình chính tiếp tục lại từ chỗ vừa tạm dừng để thực hiện chương trình con.

8.2. Thủ tục và hàm

8.2.1. Thủ tục (procedure)

- Thủ tục là một chương trình con thực hiện một nhiệm vụ nào đó.
- Khai báo thủ tục theo cấu trúc sau:

Procedure <tên thủ tục>[(danh sách các tham số hình thức)];

[khai báo các đối tượng chính thức sẽ tham gia trong thủ tục]

Begin

<tập các lệnh>

End;

Trong đó:

- Các mục viết trong cặp dấu [] có thể có hoặc không tùy theo yêu cầu của thủ tục
- Các khai báo được khai báo (nếu cần) theo cú pháp như ngoài chương trình chính.
- Nếu khai báo thủ tục không có các tham số hình thức thì ta được thủ tục không tham số, còn nếu có ta gọi là thủ tục có tham số.

Ví dụ 8.2: Viết chương trình nhập vào 3 số thực a, b, c. Tìm và in lên màn hình số nhỏ nhất trong 3 số đó?

```
Program min_a_b_c;
```

```
Var a,b,c: real;
```

{Khai báo các biến toàn cục}

```
Procedure min_abc(a,b,c: real);           {Khái báo tiêu đề thủ tục}
var min:real;                             {Khai báo biến min (là biến địa
                                          phương của thủ tục)}

begin
    min:=a;
    if min>b then min:=b;
    if min>c then min:=c;
    writeln(' Gia tri min= ',min:8:2);
end;

Begin                                   {Bắt đầu chương trình chính}
    write(' Nhap a = ');readln(a);
    write(' Nhap b = ');readln(b);
    write(' Nhap c = ');readln(c);
    min_abc(a,b,c);                       {Lời gọi thủ tục}
    readln;

End.
```

*** Nhận xét**

- Thủ tục min_abc được khai báo và triển khai đầy đủ trước khi nó được truy xuất,
- Các biến a, b, c được khai báo là biến toàn cục và được nhập ở chương trình chính, biến min được khai báo là biến địa phương trong thủ tục.

8.2.2. Hàm (function)

Hàm là một chương trình con thực hiện một nhiệm vụ nào đó và trả lại giá trị thông qua tên hàm.

Cấu trúc một hàm như sau:

Function <tên hàm>[(danh sách các tham số hình thức)]:<kiểu dữ liệu kết quả>;

[khai báo các đối tượng chính thức sẽ tham gia trong hàm]

Begin

<tập các lệnh>

End;

Trong đó:

- Các mục viết trong cặp dấu [] có thể có hoặc không tùy theo yêu cầu của hàm
- Nếu khai báo hàm không có các tham số hình thức thì ta được hàm không tham số, còn nếu có ta gọi là hàm có tham số.

Ví dụ 8.3: Viết chương trình nhập vào 3 số thực a, b, c. Tìm và in lên màn hình số nhỏ nhất trong 3 số đó?

```
Program min_a_b_c;
Var   a,b,c: real;                               {Khai báo các biến toàn cục}
function min(a,b,c:real): real;                 {Khái báo tiêu đề hàm}
var   m:real;                                     {m là biến địa phương}
begin
    m:=a;
    if m>b then m:=b;
    if m>c then m:=c;
    min:=m;                                       {lệnh gán giá trị thông qua tên hàm}
end;                                             {Bắt đầu chương trình chính}
Begin
    write(' Nhập a = ');readln(a);
    write(' Nhập b = ');readln(b);              {Lời gọi hàm}
    write(' Nhập c = ');readln(c);
    writeln('min= ',min(a,b,c):8:2);
    readln;
End.
```

***Chú ý:**

- Hàm cũng tương tự như thủ tục nhưng trong chương trình con hàm cần có ít nhất một câu lệnh gán giá trị tính được cho tên hàm:

<Tên hàm> :=<biểu thức hoặc giá trị>

Giá trị hay biểu thức mà được gán cho hàm cần phải có kiểu dữ liệu cùng kiểu với <kiểu dữ liệu kết quả> của hàm đó.

8.3. Biến toàn cục và biến địa phương

Với việc sử dụng chương trình con, một chương trình lớn có thể được chia thành nhiều chương trình con nhỏ khác nhau. Chương trình con này có thể lại là chương trình con trực tiếp của chương trình chính và cũng có thể là chương trình con của chương trình con khác. Khai báo các biến ở những vị trí khác nhau sẽ cho ta các biến làm việc trong những phạm vi, quy định khác nhau.

Những biến được khai báo trong chương trình chính được dùng trong toàn bộ khối chương trình và các khối con của nó được gọi là biến toàn cục (biến chung). Những biến được khai báo

trong chương trình con và chỉ được sử dụng trong chương trình con đó và các chương trình con của nó thì được gọi là biến địa phương (biến riêng). Khái niệm biến địa phương hay toàn cục có tính chất tương đối: một biến là toàn cục của chương trình con này nhưng lại là biến địa phương của các chương trình con khác.

Việc cấp phát bộ nhớ cho các biến thuộc các chương trình khác nhau là độc lập, hai đối tượng ở hai chương trình con khác nhau có thể trùng tên nhau, chúng sẽ được cấp phát ở những vùng nhớ khác nhau. Điều này đảm bảo tính độc lập và toàn vẹn dữ liệu khi xây dựng chương trình. Ta xét ví dụ dưới đây:

Ví dụ 8.4:

```
Program      vidu_8_4;
Var  t,k: integer;                                {Khai báo biến của chương trình chính: 2
                                                    biến t, k là hai biến toàn chương trình chính}

Procedure   test;                                {Khai báo biến của chương trình con: t là
                                                    biến riêng của thủ tục test;}
Var  t: integer;

Begin
    k:=k+5;
    t:=t+5;
End;                                                {In ra giá trị k = 5 và t = 5}
Begin                                              {Lời gọi thủ tục test}
    k:=5; t:=5;                                    {In ra màn hình k = 10 và t = 5}
    writeln('k=',k,'va t = ',t);
    test;
    writeln('k=',k,'va t = ',t);
    readln;
End.
```

Nhận xét:

Trong chương trình trên mặc dù k, t đều cùng nhận giá trị ban đầu là 5 và cùng thực hiện cộng thêm 5 trong chương trình con. Tuy nhiên kết quả in ra màn hình chỉ có biến k nhận giá trị mới là 10 còn biến t vẫn nhận giá trị là 5. Trong chương trình này thì thủ tục test không làm ảnh hưởng gì đến sự thay đổi giá trị của biến toàn cục. Vì trong chương trình con cũng được khai báo một biến địa phương cùng tên t, nên khi gọi tới chương trình con thực hiện thì biến địa phương được ưu tiên.

Trong một chương trình có biến địa phương và biến toàn cục cùng tên thì khi gọi chương trình con biến địa phương của chương trình con đó sẽ được cấp phát một vùng nhớ khác và thực hiện các lệnh trong chương trình con đối với biến này trên vùng nhớ mới mà không tác động gì

đến biến toàn cục cùng tên nên không bị thay đổi bởi chương trình con. Khi thoát khỏi chương trình con thì vùng nhớ của biến địa phương đó sẽ được giải phóng ngay, và khi trở lại chương trình chính thì chương trình dịch lại làm việc với biến toàn cục.

Khi viết các chương trình có chương trình con cần chú ý đặc biệt đến việc khai báo biến toàn cục và địa phương. Việc này tuy khó xác định đối với các bạn mới học song nó lại rất có lợi:

- Hạn chế thói quen lập trình tùy tiện,
- Nâng cao tính ổn định của chương trình
- Hạn chế những lỗi logic tiềm ẩn chưa được phát hiện
- Dễ sửa chữa hoặc phát triển chương trình đặc biệt là với các chương trình lớn, cần nhiều người tham gia.

Xét ví dụ dưới đây:

Ví dụ 8.5: Viết chương trình nhập vào 2 chuỗi st1, st. Kiểm tra xem bao nhiêu ký tự trong chuỗi st1 xuất hiện trong chuỗi st.

```
Program vidu_8_5;
var st1,st: xau50;
    i,j:byte;
function timthay(c: char; st:xau50): boolean;
(*)
begin
    timthay:=false;
    i:=1;
    while (i<=length(st))and(c<>st[i]) do
        i:=i+1;
    timthay:=i<=length(st);
end;
begin
    write(' Nhập xau st = ');readln(st);
    write(' Nhập xau st1= ');readln(st1);
    j:=0;
    for i:=1 to length(st1) do
        if timthay(st1[i],st) then
            j:=j+1;
    writeln(' Số ký tự của xau ',st1,' tìm thấy
```



```
    trong_xau ',st,' la: ',j);  
    readln;  
end.
```

Nhận xét: Chương trình này chạy cho kết quả ko chính xác. Vì biến *i* được sử dụng trong hàm là biến *i* toàn cục của chương trình chính nên nó sẽ bị thay đổi giá trị. Hơn nữa biến *i* lại là biến điều khiển của vòng lặp *for* trong chương trình chính. Vì vậy sẽ làm cho vòng *for* không xác định được.

Để sửa lỗi ta bổ sung khai báo biến *I* là biến địa phương của hàm *timthay* tại vị trí (*).

8.4. Truyền tham số cho chương trình con

8.4.1. Vai trò của tham số

- Các tham số hình thức trong phần khai báo tiêu đề chương trình con là công cụ để chương trình con giao tiếp với môi trường bên ngoài. Khi thực hiện chương trình con các tham số được dùng để gửi các giá trị vào để chương trình xử lý.
- Các tên tham số khai báo được dùng trong chương trình con như tên biến địa phương đã được khai báo, vì thế không được khai báo các biến riêng của chương trình con trùng với tên các tham số của nó.
- Điểm khác biệt giữa tham số và biến riêng của chương trình con là khi bắt đầu tham gia câu lệnh, các tham số chỉ có tên chứ chưa có giá trị như các biến. Chỉ khi xuất hiện lời gọi chương trình con thì các tham số mới được truyền giá trị từ bên ngoài vào và chương trình con sẽ được thực hiện với bộ giá trị đó. Cũng vì lý do đó mà chúng được gọi là *tham số hình thức*.
- Khi gọi chương trình con, các tham số hình thức có thể nhận bất cứ một tên biến nào khác với tên hình thức đã khai báo, chỉ cần giá trị của những tên này là tương thích kiểu với kiểu của tham số tương ứng.
- Các tên bên ngoài được truyền vào chương trình con được gọi là *tham số thực* vì chúng thực sự tham gia chương trình con với tư cách là một giá trị.
- Khi thực hiện lời gọi chương trình con, các tham số được truyền vào cần đảm bảo rằng việc truyền là tương ứng một-một về cả kiểu dữ liệu và thứ tự.
- Các tham số thực sự được truyền cho các tham số hình thức khi có lời gọi CTC theo hai phương thức sau:
 - truyền theo tham trị,
 - truyền theo tham biến.

8.4.2. Truyền theo tham trị

Các tham trị hay các tham số thực sự truyền cho các tham số hình thức theo phương hướng truyền tham trị được khai báo:

<tên tham số>:<tên kiểu dữ liệu>

Đối với các tham trị khi có lời gọi chương trình con, hệ thống sẽ cấp phát một vùng nhớ khác và sao giá trị của tham số thực sự vào đó. Chương trình con thao tác với tham số hình thức tương ứng sẽ thực hiện với dữ liệu trên bản sao này và không ảnh hưởng đến tham số thực sự.

Như vậy khi kết thúc chương trình con các bản sao bị xóa đi và giá trị của tham thực sự là giữ nguyên như trước khi truyền vào chương trình.

Việc truyền theo tham trị có những đặc điểm sau:

- Giá trị của biến được truyền theo tham trị ở đầu vào sẽ không bị thay đổi sau lời gọi chương trình con.
- Tồn thêm bộ nhớ khi thực hiện chương trình con do tạo bản sao.
- Khi gọi chương trình con cho phép giá trị ở đầu vào có thể là những giá trị của hằng, biến, hàm hoặc biểu thức.

Ví dụ 8.6:

```
Program      vidu_8_6;
Var  t,k: integer;                                {Khai báo 2 biến t, k là hai biến toàn
                                                    chương trình chính.}

Procedure test(t,k: integer);                    {Khai báo tham số hình thức của chương
                                                    trình con: t, k đều là tham trị.}

Begin
    k:=k+5;
    t:=t+5;
End;

Begin
    k:=5; t:=5;                                    {In ra giá trị k = 5 và t = 5}
    writeln('k=',k,' va t=',t);                  {Lời gọi thủ tục test}
    test(i,k);                                    {In ra màn hình k = 5 và t = 5}
    writeln('k=',k,' va t=',t);
    readln;
End.
```

8.4.3. Truyền theo tham biến

Các tham biến được khai báo trong tiêu đề của chương trình con.

Var <tên tham số>:<tên kiểu>;

Đối với các tham biến, chương trình con thao tác với các tham số thực sự sẽ được thực hiện trực tiếp với dữ liệu trên chính bộ nhớ của tham số thực sự nên khi kết thúc chương trình con thì giá trị của tham số thực sự chính là giá trị đã được chương trình con thay đổi.

Truyền theo tham biến có những đặc điểm sau:

- Giá trị của biến được truyền theo tham biến ở đầu vào sẽ bị thay đổi sau lời gọi chương trình con.
- Không tốn thêm bộ nhớ khi thực hiện chương trình con do không tạo bản sao.
- Khi gọi chương trình con giá trị ở đầu vào chỉ có thể là biến.

Ví dụ 8.7:

```
Program    vidu_8_7;
Var  t,k: integer;
{Khai báo biến của chương trình chính: 2 biến t, k là hai biến
toàn chương trình chính.}
Procedure test(k:integer; var t: integer);
{Khai báo biến hình thức của chương trình con:
    k là tham trị, t là tham biến.}
Begin
    k:=k+5;
    t:=t+5;
End;
Begin
    k:=5; t:=5;
    writeln('k = ',k,' va t = ',t);{In ra giá trị k=5 và t=5}
    test(i,k);                      {Lời gọi thủ tục test}
    writeln('k = ',k,' va t = ',t);{In ra màn hình k=5 và t=10}
    readln;
End.
```

Khi nào thì nên dùng tham biến?

- Nếu cần thay đổi giá trị thực sự theo quản lý của chương trình con
- Tránh sao lưu dữ liệu lớn.

Ví dụ 8.8: Viết chương trình hoán vị giá trị của hai biến a, b.

```
Program    vidu_8_8;
Var  a,b: real;
Procedure  hoanvi(var a,b: real); {a, b là tham biến}
Var  tg: real;
Begin
```

```
    tg:=a;
    a:=b;
    b:=tg;
End;
Begin
    Write(' nhập hai số a, b: ');readln(a,b);
    Writeln('Hai số trước hoán vị là: ', a:8:2, b:8:2);
    Hoanvi(a,b);
    Writeln('Hai số sau hoán vị là: ', a:8:2, b:8:2);
    Readln;
End.
```

Trong ví dụ trên nếu khai báo tham số hình thức a, b ở đầu chương trình con là tham trị thì sẽ không thể hoán đổi giá trị của hai biến đó cho nhau.

8.5. Tính đệ qui của chương trình con

8.5.1. Khái niệm về đệ qui

Các chương trình mà chúng ta đã xem xét đều có chung cấu trúc dạng chương trình gọi các chương trình con khác dưới dạng mô hình phân cấp. Tuy nhiên đối với một số bài toán, việc dùng chương trình con gọi ngay chính nó rất hữu dụng. Có thể định nghĩa chương trình con đệ qui là chương trình con sẽ gọi đến chính nó trực tiếp hay gián tiếp thông qua các chương trình con khác.

Cách tiến hành giải một bài toán đệ qui nhìn chung có những điểm chung sau: Trước tiên gọi chương trình con đệ qui để giải bài toán, chương trình con đệ qui thực ra chỉ biết cách giải bài toán trong trường hợp đơn giản nhất (hay còn gọi là trường hợp cơ sở). Nếu chương trình con đệ qui được gọi trong trường hợp cơ sở, chương trình con chỉ cần đơn giản trả lại kết quả. Nếu chương trình con được gọi trong các trường hợp phức tạp hơn, chương trình con đệ qui sẽ chia công việc cần giải quyết thành hai phần. Một phần hàm biết cách giải quyết như thế nào, còn phần kia vẫn không biết cách giải quyết như thế nào tuy nhiên để được gọi là có khả năng đệ qui, phần sau phải giống với bài toán ban đầu nhưng đơn giản hơn hay nhỏ hơn bài toán ban đầu. Bởi vì bài toán mới giống với bài toán ban đầu nên chương trình con sẽ thực hiện gọi chính nó để giải quyết công việc đơn giản hơn này - đây chính là lời gọi đệ qui hay còn gọi là một bước đệ qui. Để đảm bảo việc đệ qui có kết thúc, mỗi một lần gọi đệ qui thì bài toán phải đảm bảo đơn giản hơn và các bước đệ qui này còn thực hiện tiếp cho đến khi nào bài toán đơn giản dần, đơn giản tới mức trở thành trường hợp cơ sở. Có thể nhận thấy hàm đệ qui xử lý trường hợp cơ sở để trả lại kết quả tính được cho các chương trình con mức phức tạp hơn, rồi đến lượt các chương trình con này lại tính trả lại kết quả cho các chương trình con phức tạp hơn nữa ... cứ như vậy cho đến lời gọi chương trình con ban đầu.

8.5.2. Cách dùng đệ qui

8.5.2.1. Các bài toán có thể dùng Đệ Qui

Thường áp dụng cho các bài toán phụ thuộc tham số có hai đặc điểm sau:

Bài toán dễ dàng giải quyết trong một số trường hợp riêng ứng với các giá trị đặc biệt của tham số. Ta gọi đây là trường hợp suy biến.

Trong trường hợp tổng quát, bài toán có thể qui về một bài toán cùng dạng nhưng giá trị tham số thì bị thay đổi. Và sau một số hữu hạn bước biến đổi Đệ Qui sẽ dẫn đến trường hợp suy biến.

8.5.2.2. Cách xây dựng hàm Đệ Qui

Hàm Đệ Qui thường được viết theo thuật toán sau:

```
if ( trường hợp suy biến) then
begin
    trình bày cách giải bài toán
    (giả định đã có cách giải)
end
else { trường hợp tổng quát }
begin
    gọi đệ qui tới hàm (đang lập) với
    giá trị khác của tham số
end;
```

8.5.2.3. Một số ví dụ

Ví dụ 8.9: Tính $S = k!$ bằng đệ qui. Ta viết :

$$k! = \begin{cases} 1 & \text{ khi } k = 0 \\ (k-1)! & \text{ khi } k > 0 \end{cases}$$

Muốn tính $k!$ ta phải tính được $(k-1)!$, muốn tính $(k-1)!$ lại phải tính $(k-2)!$, ..., suy ra cuối cùng phải tính được $0!$, nhưng vì $0! = 1$ nên quá trình kết thúc.

Chương trình sau nhập N , tính và in giá trị $N!$. Trong chương trình có xây dựng và sử dụng một hàm đệ qui tính $k!$:

```
PROGRAM VIDU_8_9;
    { Tính N! bằng đệ qui }
Var N : Byte;
Function Gt( k : Byte) : Real;      { Hàm tính k! bằng đệ qui }
Begin
```

```
    If k=0 then Gt:= 1
    else
        Gt:= k* Gt (k-1) ;
End;

BEGIN
    Repeat
        Write(' Nhập N: ');    Readln(N);
    Until N>0;
    Writeln( N, ' != ', Gt (N) :8:2 );
    Readln;
END.
```

Khi nhập N=4, quá trình gọi các hàm được diễn giải như sau:

$$\begin{aligned} Gt(4) &= 4 * Gt(3) \\ &= 4 * 3 * Gt(2) \\ &= 4 * 3 * 2 * Gt(1) \\ &= 4 * 3 * 2 * 1 * Gt(0) \quad \{ \text{vì } k=0 \text{ nên } Gt=1 \} \\ &= 4 * 3 * 2 * 1 * 1 \\ &= 24. \end{aligned}$$

Ví dụ 8.10: Tính số hạng $U(k)$ của dãy Fibonacci bằng đệ quy:

$U(0)=1, U(1)=1, U(k)=U(k-1) + U(k-2)$ với $k>1$.

Ta viết:

$$\begin{aligned} U(k) &= 1 \text{ nếu } k=0 \text{ hoặc } k=1 \\ &= U(k-1) + U(k-2) \text{ nếu } k>1. \end{aligned}$$

Công thức truy hồi trên là cơ sở để xây dựng hàm đệ quy tính $U(k)$: để tính được một số hạng ta phải tính được hai số hạng đứng trước nó.

Chương trình sau in ra số Fibonacci thứ N bằng cách gọi hàm đệ quy Fibo.

```
PROGRAM VIDU_8_10;    { Tính số Fibonacci thứ N }
Var
    N : Integer;

Function Fibo( k : Integer) : Integer;
{ Hàm tính số Fibonacci thứ k bằng đệ quy}
Begin
    If k=0 then Fibo:= 1
```

```
        else
            if k=1 then Fibo:=1
            else
                Fibo:=Fibo(k-1) + Fibo( k-2);
End;

BEGIN
    Write(' Nhập N: ');    Readln(N);
    Writeln(' Số Fibo thứ ', N, ' = ', Fibo(N) );
    Readln;
END.
```

Ghi chú: Việc sử dụng đệ qui đòi hỏi nhiều về bộ nhớ. Do đó tránh dùng đệ qui nếu có thể được.

Ví dụ 8.11:

Tam giác Pascal là một bảng các số có $n+1$ hàng (từ hàng 0 đến hàng n), mỗi một số hạng trong hàng i là tổ hợp chập k của i (với định nghĩa tổ hợp chập k của n là

$C_n^k = n!/(k!(n-k)!)$. Bằng cách tạo tính tổ hợp (trong đó có sử dụng hàm tính giai thừa). Hãy in ra màn hình tam giác Pascal theo dạng:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
.....
```

```
Program Tam_giac_Pascal;
VAR n, i, k: Integer;

Function Giai_thua(n: integer): longint;
VAR gt : LongInt;
BEGIN
    gt := 1;
    FOR i :=1 TO n DO
        gt := gt*i;
```

```
        Giai_thua := gt;
END;

Function To_hop(n,k: integer):LongInt;
BEGIN { Than chương trình con To_hop }
    To_hop := Giai_thua(n) div ( Giai_thua(k)*Giai_thua(n-k) );
END;

BEGIN { Chương trình chính }
    WRITE('Nhap n: '); READLN(n);
    FOR i:=0 TO n DO
        BEGIN
            FOR k:= 0 TO i DO
                WRITE(To_hop(i,k):5); {In 1 hàng của tam giác Pascal}
                WRITELN; { xuong dòng }
            END;
        READLN;
    END.
```

Ví dụ 8.12: Bài toán Tháp Hà Nội: Có một cái tháp bao gồm n tầng, tầng trên nhỏ hơn tầng dưới. Hãy tìm cách chuyển cái tháp này từ vị trí A sang vị trí B nhờ vị trí trung gian C. Yêu cầu mỗi lần chỉ được chuyển một tầng và không được để tầng lớn trên tầng nhỏ.

```
Program Thap_Ha_Noi;
USES Crt;
VAR n : Integer;
Procedure Chuyen(n:Integer; VT1, VT2, VT3:Char);
BEGIN
    IF n = 1 THEN WRITELN('Chuyen tu ', VT1, ' sang ', VT2)
    ELSE
        BEGIN
            Chuyen(n-1, VT1, VT3, VT2);
            Chuyen(1, VT1, VT2, VT3);
            Chuyen(n-1, VT3, VT2, VT1);
        END;
    END;
END;
BEGIN
```



```
ClrScr;  
WRITE('Thap co bao nhieu tang? '); READLN(n);  
WRITELN('Qua trinh dich chuyen nhu sau: ');  
Chuyen(n, 'A', 'B', 'C');  
READLN;  
END.
```

BÀI TẬP CHƯƠNG 8

Bài 8.1: Phân biệt biến địa phương và biến toàn cục? Có thể đặt tên biến địa phương và biến toàn cục cùng tên không? Điều gì sẽ xảy ra? Giải thích?

Bài 8.2: Tên các tham số hình thức có thể trùng tên với biến toàn cục hay biến địa phương của chương trình con đó hay không? Điều gì sẽ xảy ra? Giải thích vì sao?

Bài 8.3: Có bao nhiêu phương thức truyền tham số cho chương trình con? Hãy phân biệt chúng? Lấy một ví dụ ?

Bài 8.4: Cho các khai báo sau:

```
Const   max = 1000;
Var     x,y,z: real;
        m,n: integer;
Procedure CTC_1(var x, y: real; z: integer);
Function  CTC_2(i,n: integer):real;
```

Trong các lời gọi sau, lời gọi nào sai? Vì sao? Nếu có thể hãy sửa lại cho đúng.

CTC_1(x,y,m);	Writeln(CTC_1(x,y,m));
CTC_1(x,y,z);	m = CTC_1(x,y,5);
CTC_1(x-y,x+y,5);	m = CTC_2(m,n);
CTC_1(x,y,max);	x = CTC_2(m,5);
CTC_1(2.5,y,3);	CTC_2(m,3);
CTC_1(x,y,5,10);	x = y + CTC_2(x,m);
CTC_1(x,y,m+2)	writeln(CTC_2(m,n));

Bài 8.5: Viết chương trình tạo một bảng chọn đơn giản gồm các mục sau:

1. Giải phương trình bậc nhất
2. Giải phương trình bậc hai
3. Giải hệ phương trình bậc nhất
0. Kết thúc chương trình.

Gõ các số 0, 1, 2, 3 để lựa chọn công việc tương ứng. Mỗi công việc sẽ được xây dựng bằng chương trình con và chỉ cần gọi ra trong chương trình chính.

Bài 8.6: Viết chương trình nhập vào một số n. In lên màn hình các số nguyên tố nhỏ hơn n.

Bài 8.7: Viết thủ tục nhập vào và in ra màn hình theo hàng dãy số thực có n phần tử. In lên màn hình các số chính phương có mặt trong dãy số.

Bài 8.8:Viết một thủ tục nhập hai ma trận vuông A, B cấp N có các phần tử là các số nguyên.

Viết một thủ tục tính ma trận $C = A + 2B$

Viết một thủ tục in các ma trận A, B và C lên màn hình

Viết một hàm kiểm tra A, B, C có phải là ma trận đối xứng không

Bài 8.9: Viết một hàm kiểm tra hàng thứ k của một ma trận A cấp $M \times N$ có lập thành một dãy tăng không?.

Nhập ma trận A và cho biết những hàng nào của A lập thành một dãy tăng ?

Bài 8.10: Viết chương trình giải phương trình bậc hai $ax^2 + bx + c = 0$ yêu cầu tạo hai chương trình con một chương trình con giải phương trình bậc nhất trong trường hợp hệ số $a = 0$ và một chương trình con giải phương trình bậc hai thực sự trong trường hợp hệ số a khác 0.

Bài 8.11: Viết một hàm để chuẩn hóa một chuỗi: xóa bỏ mọi ký tự trắng thừa ở đầu và cuối chuỗi, và giữa hai từ chỉ giữ lại đúng một ký tự trắng.

Bài 8.12: Viết một hàm để đổi một ký tự từ chữ hoa ra chữ thường. Dùng hàm đó đổi tất cả các ký tự của một chuỗi St nhập từ bàn phím ra chữ thường hết.

Bài 8.13: Nhập vào một chuỗi số nhị phân, đổi ra số hệ thập phân tương ứng.

Ví dụ : nhập chuỗi '1111' , đổi ra số 15.

Bài 8.14: Viết chương trình sử dụng chương trình con tính các tổng sau với n là số nguyên dương, x là số thực bất kỳ nhập từ bàn phím khi thực hiện chương trình:

a) $S = 1 + x + x^2 + x^3 + \dots + x^n$

b) $S = 1 + x + x^2 + x^3 + \dots + (-1)^n x^n$

c) $S = 1 + x/1! + x^2/2! + x^3/3! + \dots + x^n/n!$

Bài 8.15: Viết một hàm đệ qui tính $S = x^n$ (x thực, n nguyên dương).

Bài 8.16:Viết một hàm đệ qui tính S_n :

$$S_n = \sqrt{2 + \sqrt{2 + \dots + \sqrt{2 + \sqrt{2}}}} \quad (n \text{ dấu căn})$$

PHỤ LỤC

Bảng 1: Bảng mã ASCII với 128 ký tự đầu tiên

Hex	0	1	2	3	4	5	6	7
0	NUL 0	DLE 16	SP 32	0 48	@ 64	P 80	` 96	p 112
1	SOH 1	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113
2	STX 2	DC2 18	“ 34	2 50	B 66	R 82	b 98	r 114
3	♥ 3	DC3 19	# 35	3 51	C 67	S 83	c 99	s 115
4	♦ 4	DC4 20	\$ 36	4 52	D 68	T 84	d 100	t 116
5	♣ 5	NAK 21	% 37	5 53	E 69	U 85	e 101	u 117
6	♠ 6	SYN 22	& 38	6 54	F 70	V 86	f 102	v 118
7	BEL 7	ETB 23	‘ 39	7 55	G 71	W 87	g 103	w 119
8	BS 8	CAN 24	(40	8 56	H 72	X 88	h 104	x 120
9	HT 9	EM 25) 41	9 57	I 73	Y 89	I 105	y 121
A	LF 10	SUB 26	* 42	: 58	J 74	Z 90	j 106	z 122
B	VT 11	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123
C	FF 12	FS 28	, 44	< 60	L 76	\ 92	l 108	 124
D	CR 13	GS 29	- 45	= 61	M 77] 93	m 109	} 125

E	SO 14	RS 30	. 46	> 62	N 78	^ 94	n 110	~ 126
F	SI 15	US 31	/ 47	? 63	O 79	_ 95	o 111	DEL 127

Bảng 2: Bảng mã ASCII với ký tự số 128 - số 255

Hex	8	9	A	B	C	D	E	F
0	Ç 128	É 144	á 160	⌘ 176	Ł 192	Ⓔ 208	α 224	≡ 240
1	ü 129	æ 145	í 161	⌘ 177	⌞ 193	Ⓙ 209	β 225	± 241
2	é 130	Æ 146	ó 162	⌘ 178	Ⓢ 194	Ⓣ 210	Γ 226	≥ 242
3	â 131	ô 147	ú 163	 179	Ⓣ 195	Ⓕ 211	π 227	≤ 243
4	ä 132	ö 148	ñ 164	Ⓣ 180	— 196	Ⓛ 212	Σ 228	∫ 244
5	à 133	ò 149	Ñ 165	Ⓣ 181	Ⓡ 197	Ⓢ 213	σ 229	∫ 245
6	å 134	û 150	ª 166	Ⓣ 182	Ⓡ 198	Ⓢ 214	μ 230	÷ 246
7	ç 135	ù 151	º 167	Ⓣ 183	Ⓣ 199	Ⓣ 215	τ 231	≈ 247
8	ê 136	ÿ 152	¿ 168	Ⓣ 184	Ⓖ 200	Ⓡ 216	Φ 232	° 248
9	ë 137	Ö 153	⌞ 169	Ⓣ 185	Ⓢ 201	Ⓡ 217	Θ 233	• 249
A	è 138	Ü 154	⌞ 170	Ⓣ 186	Ⓔ 202	Ⓡ 218	Ω 234	• 250
B	ï 139	ç 155	½ 171	Ⓣ 187	Ⓙ 203	■ 219	δ 235	√ 251

C	î 140	£ 156	¼ 172	Ɔ 188	⌋ 204	■ 220	∞ 236	ⁿ 252
D	ì 141	¥ 157	ı 173	Ɔ 189	= 205	▮ 221	φ 237	² 253
E	Ä 142	ℙ 158	« 174	Ɔ 190	⌋ 206	▮ 222	ε 238	■ 254
F	Å 143	f 159	» 175	ƭ 191	± 207	■ 223	∩ 239	255

TÀI LIỆU THAM KHẢO

- [1] Turbo Pascal 5 & 6 - Giáo trình cơ sở và nâng cao kỹ thuật lập trình hướng đối tượng.
Phạm Văn Ất - NXB Giáo dục, Hà Nội - 1993.
- [2] Giáo trình Tin học cơ sở
Đào Kiến Quốc, Bùi Thế Duy - NXB ĐH Quốc Gia Hà Nội - 2005.
- [3] Theory and Problems of Programming with Pascal,
Byron S. Gottfried, 2/ed, Schaum's Outline Series, McGraw-Hill Int. Ed., New York, USA
- 1994.
- [4] Lập trình căn bản ngôn ngữ Pascal
Đoàn Nguyên Hải, Nguyễn Trung Trực, Ng. Anh Dũng - NXB Khoa Tin học, Đại học Bách khoa TP. HCM - 1993.
- [5] Giáo trình Tin học căn bản
Quách Tuấn Ngọc - NXB. Giáo dục, Hà Nội - 1995.
- [6] Ngôn ngữ lập trình Pascal
Quách Tuấn Ngọc - NXB. Giáo dục, Hà Nội - 1995.
- [7] Lập trình bằng ngôn ngữ Pascal
Nguyễn Đình Hóa - NXB Đại học Quốc gia Hà Nội - 2005
- [8] Cấu trúc dữ liệu và giải thuật
Nguyễn Đình Hóa - NXB Đại học Quốc gia Hà Nội - 2005
- [9] Bài tập Tin học I
Hồ Sĩ Đàm, Nguyễn Tô Thành, Dương Việt Thắng, Nguyễn Thanh Tùng - NXB Giáo dục,
Hà Nội - 1995.
- [10] Lập trình Pascal nâng cao
Nguyễn Tô Thành - NXB ĐH Quốc Gia Hà Nội - 2001.
- [11] Giáo trình lý thuyết và bài tập Pascal, tập 1, tập 2
Hoàng Đức Hải, Nguyễn Đình Tê - NXB Giáo dục - 1999.
- [12] Lập trình bằng Pascal với các cấu trúc dữ liệu
Larry Nyhoff, Sanford Leedstma, (người dịch: Lê Minh Trung) - NXB Đà Nẵng - 1998.
- [13] Bài giảng Mạng và truyền thông dữ liệu
Nguyễn Đình Việt - Đại học Công nghệ, ĐH Quốc gia Hà Nội.
- [14] Mạng máy tính và các hệ thống mở
Nguyễn Thúc Hải - NXB Giáo dục - 1999
- [15] Giáo trình Hệ thống mạng máy tính CCAN (Semester 1)
Nguyễn Hồng Sơn (chủ biên), Hoàng Đức Hải - NXB Lao động - Xã hội - 2003.