H KHANG (Chủ biên)
NGUYỄ Ô VĂN UY, NGUYỄN HỒNG PHƯƠNG
ĐỖ BÁ LÂM, ĐỖ THỊ NGỌC QUỲNH, ĐỖ TUẤN ANH

GIÁO TRÌNH

TIN HỌC ĐẠI CƯƠNG

NHÀ XUẤT BẢN BÁCH KHOA - HÀ NỘI

TRẦN ĐÌNH KHANG (Chủ biên)
NGUYỄN LINH GIANG, ĐỖ VĂN UY, NGUYỄN HỒNG PHƯƠNG,
ĐỖ BÁ LÂM, ĐỖ THỊ NGỌC QUỲNH, ĐỖ TUẤN ANH

GIÁO TRÌNH
TIN HỌC ĐẠI CƯƠNG

(Tái bản lần thứ ba)

NHÀ XUẤT BẢN BÁCH KHOA – HÀ NỘI

Mã số: 22 – 2014/CXB/244 – 80/BKHN

Biên mục trên xuất bản phẩm của Thư viện Quốc gia Việt Nam

Giáo trình tin học đại cương / Trần Đình Khang, Nguyễn Linh Giang, Đỗ Văn Uy... - Tái bản lần thứ nhất có sửa chữa và bổ sung. - H. : Bách khoa Hà Nội, 2014. - 246tr. : hình vẽ, bảng ; 24cm

Thư mục: tr. 245

ISBN 978-604-911-829-6

1. Tin học đại cương 2. Lập trình 3. Thuật toán 4. Giáo trình

005.1 - dc14

BKB0050p-CIP

LỜI NÓI ĐẦU

Giáo trình Tin học đại cương được biên soạn theo chương trình môn học Tin học đại cương giảng dạy tại Trường Đại học Bách Khoa Hà Nội. Giáo trình cũng có thể làm tài liệu học tập cho sinh viên các trường Đại học, Cao đẳng kỹ thuật và công nghệ trong cả nước.

Với mục tiêu cung cấp tài liệu học tập cho sinh viên, nhóm tác giả đã tập hợp bài giảng và kinh nghiệm của nhiều thầy, cô giáo trong Viện Công nghệ thông tin và Truyền thông, Trường Đại học Bách khoa Hà Nội để biên tập thành giáo trình này.

Bố cục giáo trình gồm ba phần chính như sau:

Phần I: ***Tin học căn bản.*** Phần này trình bày về thông tin, dữ liệu, hệ thống máy tính và các hệ thống ứng dụng. Đây là những kiến thức căn bản về tin học và máy tính, giúp cho người học có nền tảng tốt khi bước tiếp vào thế giới vô cùng rộng lớn thuộc lĩnh vực công nghệ thông tin.

Phần II: ***Giải quyết bài toán.*** Nội dung phần này giới thiệu về giải quyết bài toán, biểu diễn thuật toán và các thuật toán thông dụng. Với kinh nghiệm nhiều năm tham gia giảng dạy, các tác giả nhận thấy, không ít sinh viên thành thạo về tin học và lập trình nhưng vẫn thường tỏ ra lúng túng khi đứng trước một bài toán thực tế cần giải quyết. Do đó, phần này giúp sinh viên có được tư duy ban đầu và biết cách áp dụng các kiến thức tin học để giải quyết một bài toán trong thực tiễn.

Phần III: ***Lập trình.*** Đây là phần trọng tâm của giáo trình, giới thiệu những kiến thức nền tảng của ngôn ngữ lập trình C, một ngôn ngữ mà hầu hết lập trình viên chuyên nghiệp đều sử dụng.

Cuối mỗi phần là một số câu hỏi và bài tập nhằm giúp bạn đọc củng cố những kiến thức đã học.

Các tác giả xin bày tỏ sự biết ơn chân thành đối với các thầy cô giáo, các đồng nghiệp trong Viện Công nghệ thông tin và Truyền thông, Trường Đại học Bách Khoa Hà Nội, đã giúp đỡ và động viên rất nhiều trong quá trình biên soạn giáo trình. Đặc biệt, xin gửi lời cảm ơn sâu sắc tới PGS. Đặng Văn Chuyết và TS. Phạm Đăng Hải đã dành thời gian đọc bản thảo và cho những ý kiến đóng góp quý báu.

Trong quá trình biên soạn, mặc dù đã rất cố gắng, nhưng sai sót là điều khó tránh khỏi, các tác giả rất mong nhận được ý kiến đóng góp của bạn đọc để lần tái bản sau được hoàn thiện hơn.

Mọi ý kiến đóng góp xin gửi về: Viện Công nghệ thông tin và Truyền thông, Trường Đại học Bách Khoa Hà Nội, số 1 Đại Cồ Việt, Hai Bà Trưng, Hà Nội.

CÁC TÁC GIẢ

MỤC LỤC

Danh mục hình vẽ

PHẦN I. TIN HỌC CĂN BẢN

I.1 Thông tin và biểu diễn thông tin

I.1.1. Các khái niệm cơ bản về thông tin và tin học

I.1.1.1. Thông tin và xử lý thông tin

a. Thông tin – Dữ liệu – Tri thức

Thông tin

Khái niệm thông tin (*information*) được sử dụng thường ngày. Thông tin là một khái niệm trừu tượng mô tả những gì 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 thực hiện hợp lý công việc cần làm để đạt tới mục đích một cách tốt nhất.

Người ta quan niệm rằng, thông tin là kết quả xử lý, điều khiển và tổ chức dữ liệu theo cách mà nó sẽ bổ sung thêm tri thức cho người nhận. Nói theo cách khác, thông tin là ngữ cảnh trong đó dữ liệu được xem xét.

Dữ liệu

Dữ liệu (*data*) có thể hiểu là vật liệu thô mang thông tin. Dữ liệu sau khi được tập hợp lại và xử lý sẽ cho ta thông tin.

Dữ liệu trong thực tế có thể là:

- Các số liệu: là dữ liệu bằng số. Ví dụ: các số liệu trong các bảng thống kê.
- Các ký hiệu quy ước như chữ viết, các ký hiệu khắc trên đá, đất, vách núi của người xưa.
- Các tín hiệu vật lý như ánh sáng, âm thanh, nhiệt độ, áp suất,…

Ví dụ: Nhiệt kế đo thân nhiệt của một cháu bé chỉ 39^0C. Con số này cho chúng ta thông tin là cháu bé đang bị sốt. Nếu thân nhiệt giảm xuống 38^0C thì chúng ta biết được thông tin là cháu đã hạ sốt và nếu thân nhiệt là 37^0C thì cháu đã bình thường.

Tri thức

Tri thức (*knowledge*), theo nghĩa thông thường, là thông tin ở mức trừu tượng hơn. Tri thức khá đa dạng, nó có thể là sự kiện, là thông tin và kỹ năng mà một người thu thập được qua kinh nghiệm hoặc qua đào tạo. Nó có thể là sự hiểu biết chung nhiều

nh vực hay về một lĩnh vực cụ thể nào đó. Thuật ngữ tri thức được sử dụng thheo ghĩa "hiểu" về một chủ thể với một tiềm năng cho một mục đích chuyên dụng.

Hệ thống thông tin (*information system*) là một hệ thống ghi nhận dữ liệu, xử lý úng để tạo nên thông tin có ý nghĩa hoặc dữ liệu mới.

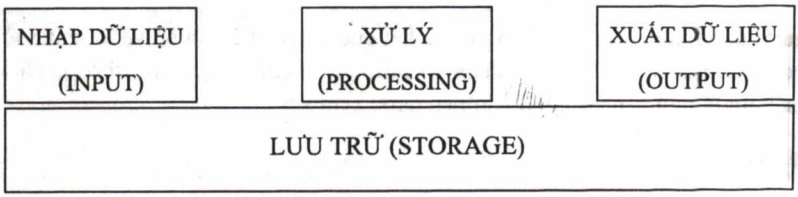
Dữ liệu → Thông tin → Tri thức

xử lý *xử lý*

Quy trình xử lý thông tin

Mọi quá trình xử lý thông tin bằng máy tính hay bởi con người đều được thực iện theo một quy trình như sau:

Dữ liệu (Data) được nhập ở đầu vào (Input), qua quá trình xử lý để nhận được ông tin ở đầu ra (Output). Dữ liệu trong quá trình nhập, xử lý và xuất đều có thể ược lưu trữ.



Hình I.1. Xử lý thông tin bằng máy tính điện tử

Dữ liệu được thu thập và lưu trữ, qua quá trình xử lý có thể trở thành dữ liệu ới để theo một quá trình xử lý dữ liệu khác tạo ra thông tin mới hơn theo ý đồ của on người.

Con người có nhiều cách để có dữ liệu và thông tin. Người ta có thể lưu trữ ông tin qua tranh vẽ, giấy, sách báo, hình ảnh trong phim, băng từ. Trong thời đại iện nay, khi lượng thông tin đến với chúng ta càng lúc càng nhiều thì con người ó thể dùng một công cụ hỗ trợ cho việc lưu trữ, chọn lọc và xử lý thông tin gọi là áy tính điện tử (computer). Máy tính điện tử giúp con người tiết kiệm rất nhiều ời gian, công sức và tăng độ chính xác cao trong việc tự động hoá một phần hay àn phần của quá trình xử lý thông tin.

I.11.2. Máy tính điện tử và phân loại

a. Nguyên lý Von Neumann

Nguyên lý này mang tên người phát minh ra nó, nhà toán học người Mỹ gốc Hungary, John Von Neumann (1903 – 1957) (tên khai sinh là Neumann János Lajos). Đây là một trong những phát minh vĩ đại của nhân loại trong lịch sử phát triển của máy tính.

Năm 1946, Von Neumann đã đề ra một nguyên lý máy tính hoạt động theo một chương trình được lưu trữ và truy nhập theo địa chỉ. Nguyên lý này được trình bày ở một bài báo nổi tiếng nhan đề "Thảo luận sơ bộ về thiết kế logic của máy tính điện tử". Nội dung nguyên lý Von Neumann gồm:

– Máy tính có thể hoạt động theo một chương trình đã được lưu trữ

Theo Von Neumann, chúng ta có thể tập hợp các lệnh cho máy tính thi hành theo một chương trình được thiết kế và coi đó như một tập dữ liệu. Dữ liệu này được cài vào trong máy và được truyền bằng xung điện. Đây là một cuộc cách mạng mới cho máy tính nhằm tăng tốc độ tính toán vào thời đó vì trước kia, máy chỉ có thể nhận được các lệnh từ băng giấy hoặc bìa đục lỗ và nạp vào bằng tay. Nếu gặp bài toán lặp lại nhiều lần thì phải nạp lại một cách thủ công như vậy, gây hạn chế trong tính toán sử dụng.

– Bộ nhớ được địa chỉ hóa

Mỗi dữ liệu đều có một địa chỉ của vùng nhớ chứa dữ liệu đó. Như vậy, để truy nhập dữ liệu, ta chỉ cần xác định địa chỉ của nó trên bộ nhớ.

– Bộ đếm của chương trình

Nếu mỗi câu lệnh phải dùng một vùng nhớ để chứa địa chỉ của câu lệnh tiếp theo thì không gian bộ nhớ sẽ bị thu hẹp. Để khắc phục hạn chế này, máy được gắn một thanh ghi để chỉ ra vị trí của lệnh tiếp theo cần được thực hiện và nội dung của nó tự động được tăng lên mỗi lần lệnh được truy cập. Muốn đổi thứ tự lệnh, ta chỉ cần thay đổi nội dung thanh ghi bằng một địa chỉ của lệnh cần được thực hiện tiếp.

b. Lịch sử phát triển của máy tính điện tử

Donhu cầu cần tăng độ chính xác và giảm thời gian tính toán, con người đã quan tân chế tạo các công cụ tính toán từ xưa: bàn tính của người Trung Quốc, máy cộng cơ học của nhà toán học Pháp Blaise Pascal (1623 – 1662), máy tính cơ học có thể cộng, trừ, nhân, chia của nhà toán học Đức Gottfried Wilhelmvon Leibniz (1646 – 1716), máy sai phân để tính các đa thức toán học,...

Tuy nhiên, máy tính điện tử chỉ thực sự được bắt đầu hình thành vào thập niên 1950, đến nay đã trải qua nhiều thế hệ, dựa vào sự tiến bộ về công nghệ điện tử và vi điện tử cũng như các cải tiến về nguyên lý, tính năng và loại hình của nó.

- Thế hệ 1 (1950 – 1958): Máy tính sử dụng các bóng đèn điện tử chân không, mạch riêng rẽ, vào số liệu bằng phiếu đục lỗ, điều khiển bằng tay. Máy có kích thước rất lớn, tiêu thụ năng lượng nhiều, tốc độ tính chậm khoảng 300 – 3.000 phép tính/giây. Loại máy tính điển hình thế hệ 1 là ENIAC, EDVAC (Mỹ) hay BESEM (Liên Xô cũ),...

- Thế hệ 2 (1958 – 1964): Máy tính dùng bộ xử lý bằng đèn bán dẫn, mạch in. Máy tính đã có chương trình dịch như Cobol, Fortran và hệ điều hành đơn giản. Kích thước máy còn lớn, tốc độ tính khoảng 10.000 – 100.000 phép tính/giây. Điển hình như loại IBM–1070 (Mỹ) hay MINSK (Liên Xô cũ),...

- Thế hệ 3 (1965 – 1974): Máy tính được gắn các bộ vi xử lý bằng vi mạch điện tử cỡ nhỏ, có thể có được tốc độ tính khoảng 100.000 – 1.000.000 phép tính/giây. Máy đã có các hệ điều hành đa chương trình, nhiều người đồng thời hoặc theo kiểu phân chia thời gian. Kết quả từ máy tính có thể in ra trực tiếp ở máy in. Điển hình như loại IBM–360 (Mỹ) hay EC (Liên Xô cũ),...

- Thế hệ 4 (1974 – nay): Máy tính bắt đầu có các vi mạch đa xử lý có tốc độ tính từ hàng chục triệu đến hàng tỷ phép tính/giây. Giai đoạn này hình thành hai loại máy tính chính: máy tính cá nhân để bàn (Personal Computer – PC) và máy tính xách tay (Laptop hoặc Notebook). Ngoài ra còn có các loại máy tính chuyên nghiệp thực hiện đa chương trình, đa xử lý,... hình thành các hệ thống mạng máy tính (Computer Networks) và các ứng dụng phong phú đa phương tiện.

- Thế hệ 5 (1990 – nay): Bắt đầu các nghiên cứu tạo ra các máy tính mô phỏng các hoạt động của não bộ và hành vi của con người, có trí khôn nhân tạo với khả năng tự suy diễn, phát triển các tình huống nhận được và hệ quản lý kiến thức cơ bản để giải quyết các bài toán đa dạng.

- Máy tính lượng tử: Người ta hi vọng và cũng đã bắt tay vào việc nghiên cứu và xây dựng các mẫu máy tính lượng tử (còn gọi là quantum computer) – loại máy tính có khả năng giải quyết cực nhanh những vấn đề

phức tạp mà các siêu máy tính hiện nay dù mất hàng triệu năm cũng chưa tìm ra lời giải. Nguyên lý cơ bản của nó là tận dụng các quy luật cơ học lượng tử cho phép một hạt, một nguyên tử hoặc một phân tử tồn tại ở hai trạng thái cùng một lúc. Trong khi các máy tính hiện nay lưu trữ dữ liệu dưới dạng các bit ở một trong hai trạng thái 0 hoặc 1 thì máy tính lượng tử lưu trữ dữ liệu bằng các bit lượng tử (gọi là qubit) ở đồng thời cả hai trạng thái 0 và 1. Trên lý thuyết, ưu điểm của loại máy tính này là khả năng lưu trữ thông tin của nhiều giải pháp tiềm năng cho một vấn đề trong cùng một bộ nhớ và xử lý tất cả các giải pháp trong cùng một thời điểm bằng các thuật toán phù hợp.

c. Phân loại máy tính điện tử

Tên thực tế, tồn tại nhiều cách phân loại máy tính khác nhau. Dựa trên hiệu năng ính toán, máy tính được phân loại như sau:

- **Máy vi tính** (Micro–computer hay PC): Loại này thường được thiết kế cho một người dùng, do đó giá thành rẻ. Hiện nay, máy vi tính khá phổ dụng và xuất hiện dưới khá nhiều dạng: máy để bàn (Desktop), máy trạm (Workstation), máy xách tay (Notebook, Laptop) và máy tính bỏ túi.

- **Máy tính tầm trung** (Mini Computer): Là loại máy tính có tốc độ và hiệu năng tính toán mạnh hơn các máy vi tính. Chúng thường được thiết kế để sử dụng cho các ứng dụng phức tạp. Giá của các máy này cũng cỡ hàng vài chục nghìn USD.

- **Máy tính lớn** (Mainframe Computer) và **Siêu máy tính** (Super Computer): Là những máy tính có tổ chức bên trong rất phức tạp, có tốc độ siêu nhanh và hiệu năng tính toán cao, cỡ hàng tỷ phép tính/giây. Các máy tính này cho phép nhiều người dùng đồng thời và được sử dụng tại các trung tâm tính toán, viện nghiên cứu để giải quyết các bài toán cực kỳ phức tạp, yêu cầu cao về tốc độ. Chúng có giá thành rất đắt, cỡ hàng trăm ngàn, thậm chí hàng triệu USD.

Theo quan điểm khác, máy tính được phân loại như sau:

- **Máy tính để bàn** (Desktop Computer): Là loại máy tính phổ biến nhất. Có hai loại là máy tính cá nhân (Personal Computer – PC) và máy tính trạm làm việc (Workstation). Giá thành từ khoảng 500 USD đến 10.000 USD.

- **Máy chủ** (Server): Là loại máy tính dùng trong mạng máy tính theo mô hình khách/chủ (Client/Server). Đây thực chất là máy phục vụ, có tốc độ và hiệu năng tính toán cao, dung lượng bộ nhớ lớn, độ tin cậy cao. Giá thành từ khoảng chục nghìn USD đến hàng chục triệu USD.

- **Máy tính nhúng** (Embedded Computer): Là loại máy tính được thiết kế chuyên dụng và được đặt trong các thiết bị để điều khiển hoạt động của thiết bị này. Ví dụ: điện thoại di động, máy ảnh kỹ thuật số, bộ điều khiển trong máy giặt, máy điều hòa, bộ định tuyến trên mạng (router),.... Giá thành dao động từ vài USD đến hàng trăm nghìn USD.

I.1.1.3. Tin học và các ngành công nghệ liên quan

a. Tin học

Thuật ngữ Tin học có nguồn gốc từ tiếng Đức vào năm 1957 do Karl Steinbuch đề xuất trong bài báo "*Informatik: Automatische Informationsverarbeitung*" (tiếng Anh, "Informatics: Automatic information processing"). Sau đó, vào năm 1962, Philippe Dreyfus người Pháp gọi là "informatique", tiếp theo, Walter F.Bauer cũng sử dụng tên này. Phần lớn các nước Tây Âu, trừ Anh, đều chấp nhận. Ở Anh, người ta sử dụng thuật ngữ "computer science" hay "computing science" là thuật ngữ dịch, Nga cũng chấp nhận tên "*informatika*" (1966).

Tin học được xem là ngành khoa học nghiên cứu các phương pháp, công nghệ và kỹ thuật xử lý thông tin một cách tự động. Công cụ chủ yếu sử dụng trong tin học là máy tính điện tử và các thiết bị truyền tin khác. Nội dung nghiên cứu của tin học chủ yếu gồm hai phần:

- **Kỹ thuật phần cứng** (Hardware engineering): Nghiên cứu chế tạo các thiết bị, linh kiện điện tử, công nghệ vật liệu mới... hỗ trợ cho việc thiết kế chế tạo máy tính và mạng máy tính, đẩy mạnh khả năng xử lý và truyền thông.

- **Kỹ thuật phần mềm** (Software engineering): Nghiên cứu phát triển các hệ điều hành, các tiện ích chung cho máy tính và mạng máy tính, các phần mềm ứng dụng phục vụ các mục đích xử lý và khai thác thông tin khác nhau của con người.

b. Công nghệ thông tin

Thuật ngữ Công nghệ thông tin (Information Technology, viết tắt là IT) xuất hiện ở Việt Nam vào những năm 90 của thế kỷ XX. Theo Hiệp hội Công nghệ thông tin Hoa Kỳ (Information Technology Association of America, viết tắt là ITAA):

"Công nghệ thông tin là ngành nghiên cứu các hệ thống thông tin dựa vào máy tính đặc biệt là các phần mềm ứng dụng và phần cứng máy tính. Nói một cách ngắn gọn, IT xử lý với các máy tính điện tử và các phần mềm máy tính nhằm chuyển đổi, lưu trữ, bảo vệ, truyền tin và trích rút thông tin một cách an toàn".

Theo Nghị quyết 49/CP ngày 04/08/1993 của Chính Phủ Việt Nam thì "*Công nghệ thông tin (CNTT) là tập hợp các phương pháp khoa học, các phương tiện và công cụ kỹ thuật hiện đại – chủ yếu là kỹ thuật máy tính và viễn thông – nhằm tổ chức, khai thác và sử dụng có hiệu quả các nguồn tài nguyên thông tin rất phong phú và tiềm năng trong mọi lĩnh vực hoạt động của con người và xã hội*"

Các ứng dụng ngày nay của Công nghệ thông tin:

- *Các bài toán khoa học kỹ thuật*: Tính toán số với các thuật toán phức tạp cần thực hiện hàng trăm triệu đến hàng tỉ phép tính như xử lý các số liệu thực nghiệm, quy hoạch và tối ưu hóa, giải gần đúng các hệ phương trình. Công trình lập bản đồ gen người hoàn thành trong năm 2000 phải tính toán nhiều năm trời trên các siêu máy tính là một ví dụ.

- *Các bài toán quản lý*: Xử lý một khối lượng thông tin lưu trữ lớn (các hồ sơ) với những công việc như tạo lập cơ sở dữ liệu, duy trì cơ sở dữ liệu, khai thác, hỗ trợ cho quá trình ra quyết định. Người ta ước tính, khoảng 85% nguồn lực đầu tư cho tin học là dành cho bài toán quản lý.

- *Tự động hóa*: Có thể tự động hóa những quy trình điều khiển phức tạp, có tính mềm dẻo, có thể thay đổi hành vi tự động hóa bằng cách lập trình lại. Thiết bị được điều khiển và máy tính điều khiển trong đa số trường hợp không tách rời nhau. Các máy móc trở nên "thông minh" nhờ các bộ vi xử lý và bộ nhớ ROM được cấy trực tiếp vào máy.

- *Công tác văn phòng*: Đây là những hoạt động khá phổ biến và được chú ý sớm. Các công việc như tạo văn bản, in ấn, gửi thư,... trở nên nhanh chóng và thuận tiện.

- *Giáo dục và đào tạo*: Hỗ trợ người dạy trong việc trình bày bài giảng, bổ sung kiến thức, giúp kiểm tra, đánh giá trình độ người học. Môi trường internet mở ra những khả năng mới trong giáo dục vào đào tạo từ xa.

- *Thương mại điện tử*: Hỗ trợ các hoạt động thương mại qua mạng một cách nhanh chóng và thuận tiện. Một số hình thức như quảng cáo trên mạng, mua bán hàng hóa, dịch vụ và thanh toán qua mạng. Thách thức lớn nhất

của thương mại điện tử là vấn đề pháp lý và độ an toàn, tin cậy trong c các giao dịch điện tử.

- *Ứng dụng trong cuộc sống thường ngày*: Các máy móc, đồ điện tử, , đồ gia dụng được điều khiển bằng chip với các chương trình điều khhiển thông minh.

:. *Công nghệ thông tin và truyền thông*

Ngày nay, người ta có khuynh hướng sử dụng "information" thay thế cho "daata" và có xu thế mở rộng cho lĩnh vực truyền thông, **IT** trở thành **ICT** (Informatition ınd Communication Technology, dịch là "Công nghệ thông tin và Truyền thôngg"). Thuần tuý theo cách nói thì hai thuật ngữ này là như nhau.

Truyền thông máy tính, nói đơn giản là sự kết nối một số lượng máy tính với nhhau rong một phạm vi địa lý nhỏ. Tuy nhiên, nhiều máy tính có thể kết nối với nhhau heo một phạm vi rộng hơn và việc trao đổi dữ liệu giữa các máy tính được thhực ıiện thông qua một mạng viễn thông nào đó. **Internet** – *Mạng máy tính toàn cầuu* là nột phát minh vĩ đại của nhân loại trong thế kỷ XX, đó cũng chính là sản phẩm :ủa ngành Công nghệ thông tin và Truyền thông.

.1.2. Biểu diễn dữ liệu trong máy tính

.1.2.1. Biểu diễn số trong các hệ đếm

Hệ đếm là tập hợp các ký hiệu và quy tắc dùng để biểu diễn và xác định các giá trị :ủa các số. Mỗi hệ đếm có một số chữ số hữu hạn (chữ số – digits, một số tài liiệu :òn gọi là ký số hay ký tự). Tổng số chữ số của mỗi hệ đếm được gọi là **cơ số** (baase ıay radix), ký hiệu là b.

a. *Hệ đếm cơ số b*

Hệ đếm cơ số b ($b \geq 2$ và nguyên dương) mang tính chất sau :

- Có b chữ số để thể hiện giá trị số. Chữ số nhỏ nhất là **0** và lớn nhất là **b – 1**.
- Giả sử số $N_{(b)}$ trong hệ đếm cơ số b được biểu diễn dưới dạng:

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

trong đó, số $N_{(b)}$ có **n+1** chữ số biểu diễn cho phần nguyên và **m** chữ số biểu diễn cho phần thập phân. Giá trị của số $N_{(b)}$ trong hệ thập phân là:

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

hay:

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

Trong ngành toán – tin học hiện nay, phổ biến bốn hệ đếm là hệ thập phân, hệ nhị phân, hệ bát phân và hệ thập lục phân.

b. Hệ đếm thập phân (Decimal system, b = 10)

Hệ đếm thập phân hay hệ đếm cơ số 10 là một trong các phát minh của người Ả-rập cổ, bao gồm mười chữ số sau: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9.**

Quy tắc tính giá trị của hệ đếm này là mỗi đơn vị ở một hàng bất kỳ có giá trị bằng 10đơn vị của hàng kế cận bên phải. Ở đây b = 10. Bất kỳ số nguyên dương nào trong hệ thập phân cũng có thể biểu diễn như là một tổng các số hạng, mỗi số hạng là ích của một số với 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: Số 5246 có thể được biểu diễn như sau:

$$5246 = 5 \times 10^3 + 2 \times 10^2 + 4 \times 10^1 + 6 \times 10^0$$

$$= 5 \times 1000 + 2 \times 100 + 4 \times 10 + 6 \times 1$$

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

$$5246 = 5000 + 200 + 40 + 6$$

Như vậy, trong số 5246: chữ số 6 trong số nguyên đại diện cho giá trị 6 đơn vị (1s), chữ số 4 đại diện cho giá trị 4 chục (10s), chữ số 2 đại diện cho giá trị 2 trăm (100s) và chữ số 5 đại diện cho giá trị 5 nghìn (1000s). Nghĩa là, số lũy thừa của 10 tăng dần 1 đơn vị từ phải sang trái tương ứng với vị trí ký hiệu số:

$10^0 = 1$; $10^1 = 10$; $10^2 = 100$; $10^3 = 1000$; $10^4 = 10000$;...

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

Phần thập phân thể hiện 10 lũy thừa âm tính từ trái sang phải kể từ dấu chấm phân cách:

Ví dụ: $254.68 = 2 \times 10^{2} + 5 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 8 \times 10^{-2}$

$$= 200 + 50 + 4 + \frac{6}{10} + \frac{8}{100}$$

c. Hệ đếm nhị phân (Binary system, b = 2)

Với cơ số b = 2, chúng ta có hệ đếm nhị phân. Đây là hệ đếm đơn giản nhất với hai chữ số là 0 và 1. Mỗi chữ số nhị phân gọi là BIT (viết tắt từ chữ BInary diggiT). Vì hệ nhị phân chỉ có hai chữ số 0 và 1, nên khi muốn diễn tả một số lớn hơn cần kết hợp nhiều bit với nhau. Ta có thể chuyển đổi số trong hệ nhị phân sang số trong hệ thập phân quen thuộc.

Ví dụ: Số $11101.11_{(2)}$ sẽ tương đương với giá trị thập phân là:

					Vị trí dấu chấm cách ←		
Số nhị phân	1	1	1	0	1	1	1
Số vị trí	4	3	2	1	0	– 1	– 2
Trị vị trí	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}
Hệ 10 là	16	8	4	2	1	0.5	0.25

Như vậy:

$$11101.11_{(2)} = \mathbf{1} \times 16 + \mathbf{1} \times 8 + \mathbf{1} \times 4 + \mathbf{0} \times 2 + \mathbf{1} \times 1 + \mathbf{1} \times 0.5 + \mathbf{1} \times 0.25 = 29.75_{(10)}$$

Số 10101 (hệ 2) sang hệ thập phân sẽ là:

$$10101_{(2)} = \mathbf{1} \times 2^4 + \mathbf{0} \times 2^3 + \mathbf{1} \times 2^2 + \mathbf{0} \times 2^1 + \mathbf{1} \times 2^0 = 16 + 0 + 4 + 0 + 1 = 21_{(100)}$$

d. Hệ đếm bát phân (Octal system, b = 8)

Nếu dùng một tập hợp ba bit thì có thể biểu diễn tám trị khác nhau : 000, 0001, 010, 011, 100, 101, 110, 111. Các trị này tương đương với tám trị trong hệ thập phân là 0, 1, 2, 3, 4, 5, 7. Tập hợp các chữ số này gọi là hệ bát phân, là hệ đếm với $b = 8 = 2^3$. Trong hệ bát phân, trị vị trí là lũy thừa của 8.

Ví dụ:

$$235.64_{(8)} = \mathbf{2} \times 8^{2} + \mathbf{3} \times 8^{1} + \mathbf{5} \times 8^{0} + \mathbf{6} \times 8^{-1} + \mathbf{4} \times 8^{-2} = 157.8125_{(10)}$$

e. Hệ đếm thập lục phân (Hexa–decimal system, b = 16)

Hệ đếm thập lục phân là hệ cơ số $b = 16 = 2^{4}$, tương đương với tập hợp bốn chữ số nhị phân (4 bit). Khi thể hiện ở dạng hexa–decimal, ta có 16 chữ số gồm mười chữ số từ 0 đến 9, và sáu 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:

$$34F5C_{(16)} = \mathbf{3} \times 16^{4} + \mathbf{4} \times 16^{3} + \mathbf{15} \times 16^{2} + \mathbf{5} \times 16^{1} + \mathbf{12} \times 16^{0} = 216294_{(10)}$$

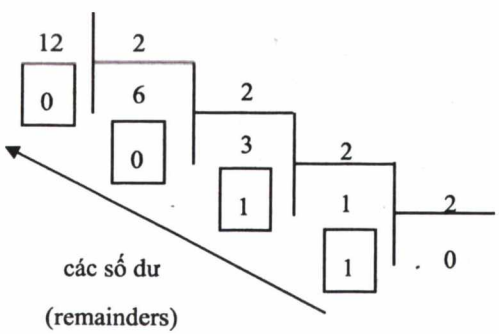
Ghi chú: Một số ngôn ngữ lập trình quy định viết số hexa phải có chữ H ở cuối chữ số. Ví dụ: Số 15 viết là FH.

f. Chuyển đổi một số từ hệ thập phân sang hệ đếm cơ số b

Đổi phần 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: Đổi từ hệ thập phân sang hệ nhị phân số $12_{(10)} = ?_{(2)}$. Dùng phép chia cho 2 liên tiếp, ta có một loạt các số dư như sau:



Kết quả: $\mathbf{12}_{(10)} = \mathbf{1100}_{(2)}$

Đổi phần thập phân từ hệ thập phân sang hệ cơ số b

Tổng quát: Lấy phần thập phân $N_{(10)}$ lần lượt nhân với b cho đến khi phần thập phân của tích số bằng 0. Kết quả số chuyển đổi $N_{(b)}$ là các số phần nguyên trong phép nhân viết ra theo thứ tự tính toán.

Ví dụ: $0.6875_{(10)} = ?_{(2)}$

Phần nguyên của tích

$0.6875 \times 2 = \mathbf{1}.375$ ← Phần thập phân của tích

$0.375 \times 2 = \mathbf{0}.75$

$0.75 \times 2 = \mathbf{1}.5$

$0.5 \times 2 = \mathbf{1}.0$

Kết quả: $\mathbf{0.6875}_{(10)} = \mathbf{0.1011}_{(2)}$

Lưu ý: chuyển đổi giữa hệ nhị phân và hệ thập lục phân. Để chuyển đổi từ hệ nhị phân sang hệ thập lục phân, ta duyệt từ phải sang trái, chia thành các nhóm bốn bit, sau đó thay từng nhóm bốn bit này bằng một chữ số Hexa theo bảng sau:

Hệ thập phân	Hệ nhị phân	Hệ thập lục phân
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Ví dụ: $111110_{(2)}$ chính là $0011\ 1110_{(2)} = 3E_{(16)}$

$$\underbrace{0011\ 1110}_{3\quad E}{}_{(2)} = 3E_{(16)}$$

I.1.2. Biểu diễn dữ liệu trong máy tính và đơn vị thông tin

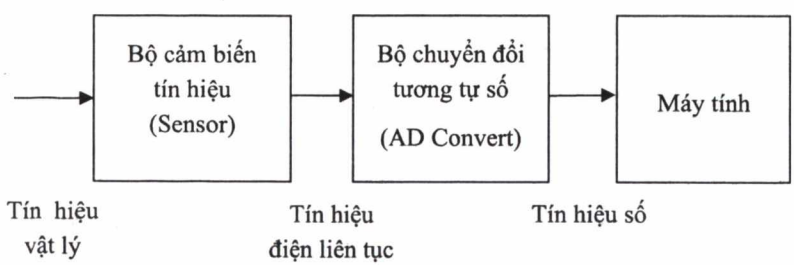
a. Nguyên tắc chung

Thông tin và dữ liệu mà con người hiểu được tồn tại dưới nhiều dạng khác nhau, ví dụ như các số liệu, các ký tự văn bản, âm thanh, hình ảnh,... nhưng trong máy tính mọi thông tin và dữ liệu đều được biểu diễn bằng số nhị phân (chuỗi bit).

Để đưa dữ liệu vào máy tính, cần phải mã hoá nó về dạng nhị phân. Với các kiểu dữ liệu khác nhau cần có cách mã hoá khác nhau. Cụ thể:

- Các dữ liệu dạng số (số nguyên hay số thực) sẽ được chuyển đổi trực tiếp thành các chuỗi số nhị phân theo các chuẩn xác định.
- Các kí tự được mã hoá theo một bộ mã cụ thể, có nghĩa là mỗi ký tự sẽ tương ứng với một chuỗi số nhị phân.
- Các dữ liệu phi số khác như âm thanh, hình ảnh và nhiều đại lượng vật lý khác muốn đưa vào máy phải **số hoá** (*digitalizing*). Có thể hiểu một cách đơn giản khái niệm số hoá như sau: các dữ liệu tự nhiên thường là quá trình biến đổi liên tục, vì vậy để đưa vào máy tính, nó cần được biến đổi sang một dãy hữu hạn các giá trị số (nguyên hay thực) và được biểu diễn dưới dạng nhị phân.

  Với các tín hiệu như âm thanh, video hay các tín hiệu vật lý khác, quy trình mã hoá được biểu diễn như sau:



Hình I.2. Quá trình số hoá tín hiệu vật lý

Tuy rằng mọi dữ liệu trong máy tính đều ở dạng nhị phân, song do bản chất của dữ liệu, người ta thường phân dữ liệu thành hai dạng:

- **Dạng cơ bản**: Gồm dạng số (nguyên hay thực) và dạng ký tự. Số nguyên không dấu được biểu diễn theo dạng nhị phân thông thường, số nguyên có dấu theo mã bù hai, còn số thực theo dạng dấu phảy động. Để biểu diễn một dữ liệu cơ bản, người ta sử dụng một số bit. Các bit này ghép lại với nhau để tạo thành cụm: cụm 8 bit, cụm 16 bit,...
- **Dạng có cấu trúc**: Trên cơ sở dữ liệu cơ bản, trong máy tính, người ta xây dựng nên các dữ liệu có cấu trúc phục vụ cho các mục đích sử dụng khác nhau. Tuỳ theo cách "ghép" mà chúng ta có mảng, tập hợp, xâu, bản ghi,...

b. Đơn vị thông tin

Đơn vị nhỏ nhất để biểu diễn thông tin gọi là **bit**. Một bit tương ứng với một sự kiện có một trong hai trạng thái.

Ví dụ: Một mạch đèn có hai trạng thái là:

- Tắt (Off) khi mạch điện qua công tắc là hở.
- Mở (On) khi mạch điện qua công tắc là đóng.

Số học nhị phân sử dụng hai chữ số 0 và 1 để biểu diễn các số. Vì khả năng sử dụng hai số 0 và 1 là như nhau nên một chỉ thị chỉ gồm một chữ số nhị phân có thể xem như là đơn vị chứa thông tin nhỏ nhất.

Bit là chữ viết tắt của **BI**nary digi**T**. Trong tin học, người ta thường sử dụng các đơn vị đo thông tin lớn hơn như sau:

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

I.1.3. Biểu diễn số nguyên

Số nguyên gồm số nguyên không dấu và số nguyên có dấu. Về nguyên tắc, đều dùng một chuỗi bit để biểu diễn.

a. Số nguyên không dấu

Trong biểu diễn số nguyên không dấu, mọi bit đều được sử dụng để biểu diễn giá trị số. Ví dụ, một dãy 8 bit dùng để biểu diễn số nguyên không dấu, có thể biểu diễn được $2^8 = 256$ số nguyên không âm, có giá trị từ 0 (0000 0000) đến 255 (1111 1111). Với 16 bit thì dải biểu diễn là [0, 65535].

Với n bit, ta có thể biểu diễn một số nguyên có giá trị lớn nhất là $2^n - 1$ và dải giá trị biểu diễn là $[0, 2^n - 1]$.

Ví dụ: Với 8 bit

0000 0000 = 0

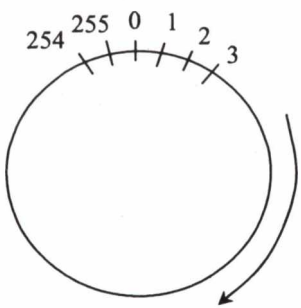
0000 0010 = 2

0000 0100 = 4

.....

1111 1111 = 255

Trục số học của máy tính biểu diễn số nguyên không dấu 8 bit:



b. Số nguyên có dấu

Số nguyên có dấu thể hiện trong máy tính ở dạng nhị phân là số dùng một bit làm bit dấu, người ta quy ước dùng bit ở hàng đầu tiên bên trái làm bit dấu (S): 0 biểu diễn số không âm và 1 biểu diễn số âm. Ví dụ: Một dãy 8 bit dùng để biểu

diễn số nguyên có dấu, có thể biểu diễn được $2^8 = 256$ số nguyên có dấu, có giá trị từ –128 (1000 0000) đến +127 (0111 1111). Với 16 bit, dải biểu diễn là [–322768, +32767]. Với n bit, ta có dải biểu diễn là $[-2^{n-1}, 2^{n-1} - 1]$.

Cách phổ biến biểu diễn số âm có dấu là dùng mã bù hai như sau:

- Biểu diễn số nguyên không dấu.
- Nghịch đảo tất cả các bit (số bù một).
- Cộng thêm một (số bù hai).

Ví dụ: Biểu diễn trên 8 bit:

37 = 0010 0101

Bù một (nghịch đảo) = 1101 1010

Bù hai (cộng thêm 1) 1

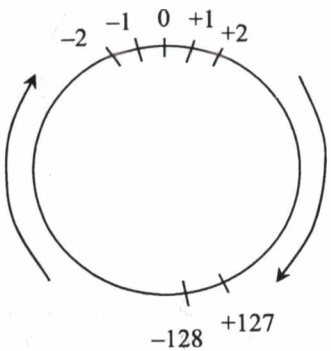
1101 1011 ⇒ số –37

↓

Bit dấu

☞ *Thử biểu diễn mã bù hai của –37 để thu được số +37*

Trục số học máy tính biểu diễn số nguyên có dấu 8 bit:



c. Tính toán số học với số nguyên

Cộng/trừ số nguyên

Cộng trừ số nguyên không dấu

Khi cộng hai số nguyên không dấu n bit ta thu được một số nguyên không dấu cũng bit. Vì vậy:

- Nếu tổng của hai số đó nhỏ hơn hoặc bằng $2^n - 1$ thì kết quả nhận được là đúng.
- Nếu tổng của hai số đó lớn hơn $2^n - 1$ thì khi đó sẽ tràn số và kết quả là sai.

Ví dụ Với trường hợp 8 bit, tổng nhỏ hơn 255 thì ta sẽ có kết quả đúng:

$$\begin{array}{rl} & 57 = 0011\ 1001 \\ + & \\ & 34 = 0010\ 0010 \\ \hline & 91 = 0101\ 1011 \end{array}$$

trong khi đó, phép cộng giữa hai số 209 và 73 có kết quả như sau:

$$\begin{array}{rl} & 209 = \ 1101\ 0001 \\ + & \\ & 73 = \ 0100\ 1001 \\ \hline & 282 = \mathbf{1}0001\ 1010 \end{array}$$

Bit tràn ra ngoài => kết quả = 26 là sai.

☞ Để tránh hiện tượng tràn số này, ta phải sử dụng nhiều bit hơn để biểu diễn.

Cộng/trừ số nguyên có dấu

Số nguyên có dấu được biểu diễn theo mã bù hai, vậy quy tắc chung như sau:

- Cộng hai số nguyên có dấu n – bit sẽ bỏ qua giá trị nhớ ra khỏi bit có ý nghĩa cao nhất, tổng nhận được sẽ có giá trị đúng và cũng được biểu diễn theo mã bù hai, nếu kết quả nhận được nằm trong dải -2^{n-1} đến $2^{n-1} - 1$.
- Để trừ hai số nguyên có dấu X và Y (X – Y), cần lấy bù hai của Y tức là –Y, sau đó cộng X với –Y theo nguyên tắc trên.

Ví dụ: 97 – 52 = 97 + (–52)

```
      +97 =  0110 0001
+
      –52 =  1100 1100
-------------------------
      45 = (1)0010 1101
             ↓
          Bỏ qua
```

Như vậy, khi thực hiện phép tính trên sẽ thừa ra một bit ngoài cùng bên trái, bit này sẽ không được lưu trong kết quả và sẽ được bỏ qua.

Nhân/chia số nguyên

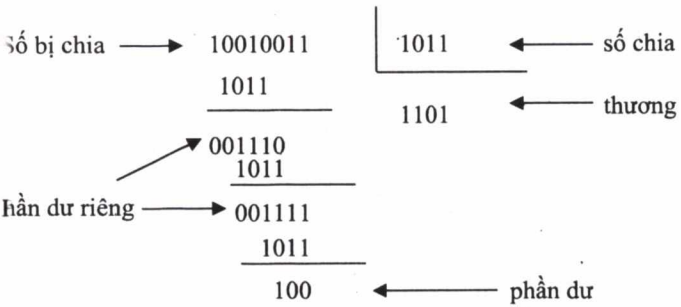
So với phép cộng và phép trừ, phép nhân và phép chia phức tạp hơn nhiều. Dưới đây chỉ giới thiệu phép nhân/phép chia với số nhị phân. Ví dụ sau mô tả phép nhân hai số nhị phân:

```
          1011      (11 cơ số 10)
×
          1101      (13 cơ số 10)
-------------------
          1011
         0000
        1011
       1011
-------------------
      10001111   kết quả 143 trong cơ số 10
```

Chún ta có một số nhận xét sau:

1. Phép nhân tạo ra các tích riêng, mỗi tích thu được là kết quả của việc nhân từng bit.
2. Các tích riêng dễ dàng xác định theo quy tắc:
   - Bit tương ứng số nhân là 1 thì tích riêng bằng số bị nhân.
   - Bit tương ứng số nhân bằng 0 thì tích riêng bằng 0.
3. Tích được tính bằng tổng các tích riêng.

Phép hia phức tạp hơn phép nhân nhưng dựa trên cùng một nguyên tắc. Hãy xem ví dụ au:



Phép hia với số nguyên sẽ cho kết quả là thương và phần dư.

I.1.2. Biểu diễn số thực

a. Nguên tắc chung

Để biu diễn số thực, trong máy tính người ta dùng ký pháp dấu phảy động (Floatng Point Number). Một cách tổng quát, một số thực biểu diễn theo cách này gồm h thành phần:

$$N = M \times R^E$$

Với M là phần định trị (Mantissa), R là cơ số (Radix), còn E là phần số mũ (Expoent)

Cơ sốthường được sử dụng là cơ số 2 hay cơ số 10, còn M và E biểu diễn theo kiểu số nguyên. Thực tế, người ta chỉ cần lưu trữ M và E.

Ví dụ với cơ số R = 10, giả sử hai số thực N1 và N2 được biểu diễn theo phần định trị và số mũ như sau:

M1 = –15 và E1 = +12

M2 = +314 và E2 = –9

Có nghĩa là $N1 = M1 \times 10^{E1} = -15 \times 10^{12} = -15\ 000\ 000\ 000\ 000$

và $N2 = M2 \times 10^{E2} = 314 \times 10^{-9} = 0.000\ 000\ 314$

Như vậy, biểu diễn số bằng phần định trị và phần số mũ sẽ gọn hơn sử dụng giáá trị đúng của số.

Khi thực hiện phép toán với số dấu phảy động sẽ được tiến hành trên cơ sở các ; giá trị của phần định trị và phần mũ. Giả sử có hai số dấu phảy động sau:

$N1 = M1 \times R^{E1}$ và $N2 = M2 \times R^{E2}$

Khi đó, việc thực hiện các phép toán số học sẽ được tiến hành:

$N1 \pm N2 = (M1 \times R^{E1-E2} \pm M2) \times R^{E2}$, giả thiết $E2 \geq E1$

$N1 \times N2 = (M1 \times M2) \times R^{E1+E2}$

$N1 / N2 = (M1 / M2) \times R^{E1-E2}$

Chú ý: Với số thực biểu diễn theo dấu phảy động trên :

- 32 bit: dải giá trị từ 10^{-38} đến 10^{+38}.
- 64 bit: dải giá trị từ 10^{-308} đến 10^{+308}.
- 80 bit: dải giá trị từ 10^{-4932} đến 10^{+4932}.

Từ công thức trên, ta nhận thấy rằng cách biểu diễn này không bao giờ cho giáá trị bằng 0, vì thế, có một số trường hợp phải quy ước:

- Nếu tất cả các bit của E và M đều bằng 0, thì $N = \pm 0$.
- Nếu tất cả các bit của E = 1 và M = 0, thì $N = \pm \infty$.
- Nếu tất cả các bit của E = 1 và có ít nhất 1 bit của M = 1, thì N không phải là số.

b. Chuẩn IEEE 754/85

Việc biểu diễn trong dấu phảy động theo chuẩn IEEE được hình dung như sau:

Sign(1 bit)	*Exponent (8 bit) d*	*Mantissa (23 bit) e*

- Bit dấu là 0 có nghĩa đó là số dương, ngược lại đó là số âm (Matissa sign).
- Phần mũ biểu diễn trong cơ số 2 và giá trị là giá trị gốc cộng thêm 127. Tuy nhiên, nếu giá trị sau khi cộng là 255 thì đó không phải là biểu diễn số.
- Phần định trị biểu diễn dạng số lẻ nhị phân nhỏ hơn 1.

Chú ý: có sự khác nhau giữa biểu diễn dấu phảy động trên main frame :

- Phần mũ là 8 bit và giá trị kết quả được cộng thêm 127 vào phần gốc. Phần thêm này gọi là bias.
- Phần định trị có 23 bit và phần lẻ nhị phân tương đương với phần định trị trừ đi 1 sẽ được lưu. Nói cách khác, số 1 không biểu diễn (bỏ).
- Cơ số phần mũ được hiểu là cơ số 2.

Ví dụ: Số thực +5 sẽ được biểu diễn như sau:

$5_{10} = 101_2 = 101_2 \times 2^0 = (1.01)_2 \times 2^2$ và phần định trị sẽ là $1.01_2 - 1_2 = 0.01_2$.

Nếu 101_2 trượt phải 2 bit sẽ trở thành 1.01_2, 2^{-2} lần từ giá trị ban đầu. Với mục đích chun hóa, 2 được cộng thêm vào phần mũ 0 và phần mũ có giá trị là 2. Do vậy, khi mà phần mũ là 2 cộng thêm phần bias 127 sẽ là 129 và mũ biểu diễn là 10000001_2.

I.1.5. Biểu diễn ký tự

a. Nguyên tắc chung

Trong máy tính, các ký tự cũng cần được chuyển đổi thành chuỗi bit nhị phân gọi là mã của các ký tự đó. Số bit dùng cho mỗi ký tự theo các mã khác nhau là khác nhau. Bộ mã ASCII (American Standard Code for Information Interchange) dùng 8 bit mã hóa cho một ký tự, còn bộ mã Unicode dùng 16 bit. Đây là hai bộ mã thông dụng. Ví dụ, với bộ mã ASCII, chữ 'A' có mã là 65 = 0100 0001.

Ngoài hai bộ mã trên, còn có các bộ mã khác:

- Hệ thập phân mã nhị phân **BCD** (Binary Coded Decima) dùng 6 bit.
- Hệ thập phân mã nhị phân mở rộng **EBCDIC** (Extended Binary Coded Decimal Interchange Code) dùng 8 bit tương đương 1 byte để biểu diễn một ký tự.

b. Bộ mã ASCII

ASCII là bộ mã được dùng để *trao đổi thông tin chuẩn của Mỹ*. Lúc đầu chỉ dùùng 7 bit (128 ký tự), sau đó mở rộng cho 8 bit và có thể biểu diễn 256 ký tự khác nhau trong máy tính.

- *Các ký tự hiển thị thông dụng:* Các mã từ 32 đến 126 dùng để mã hóaa cho các ký tự hiển thị thông dụng, chia thành ba nhóm như sau:
    - Các chữ cái La–tinh, bao gồm: 26 chữ cái thường từ a đến z cóó mã từ 97 đến 122, 26 chữ cái hoa từ A đến Z có mã từ 65 đến 90. (Giữa chữ cái thường và chữ cái hoa tương ứng có mã chênh lệch là 332.
    - Chữ số thập phân: 10 chữ số từ 0 đến 9 được gán mã từ 48 đến 57.
    - Các ký tự khác: các dấu chấm câu, các phép toán, dấu cách,...
- *Các ký tự điều khiển:* 32 ký tự đầu tiên của bảng mã ASCII (có mã từ 0 đến 31) và mã 127 dùng để mã hóa các thông tin điều khiển, dùng cho việc chuyển những thông tin đến màn hình, máy in, máy tính khác,...
- *Các ký tự mở rộng:* có mã từ 128 đến 255, phụ thuộc vào các nhà chếế tạo máy tính và phát triển phần mềm. Ví dụ: bộ mã mở rộng của IBM dùng cho các máy tính dòng IBM, bộ mã mở rộng của Apple dùng cho các máy tính Macintosh, bộ mã TCVN5712 có các ký tự riêng của tiếng Việt.

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 MÃ ASCII với 128 ký tự kế tiếp

Hex	8	9	A	B	C	D	E	F
0	Ç 128	É 144	á 160	▒ 176	└ 192	╨ 208	α 224	≡ 240
1	ü 129	æ 145	í 161	▓ 177	┴ 193	╤ 209	ß 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	¢ 155	½ 171	╗ 187	╦ 203	█ 219	δ 235	√ 251
C	î 140	£ 156	¼ 172	╝ 188	╠ 204	▄ 220	∞ 236	n 252
D	ì 141	¥ 157	¡ 173	╜ 189	═ 205	▌ 221	φ 237	2 253
E	Ä 142	Pts 158	« 174	╛ 190	╬ 206	▐ 222	ε 238	■ 254
F	Å 143	ƒ 159	» 175	┐ 191	╧ 207	▀ 223	∩ 239	255

c. Bộ mã Unicode

Bảng mã ASCII 8 bit với 256 giá trị không thể đủ chỗ để mã hóa các ký tự của các ngôn ngữ dùng chữ tượng hình như tiếng Hán, tiếng Nhật, Hàn Quốc, Thái Lan,... Từ trước đến nay, đã có nhiều giải pháp khác nhau để mã hoá các ký tự của các ngôn ngữ này trên máy tính. Tuy nhiên, những giải pháp này thường dùng kỹ thuật tổ hợp hoặc các chuỗi ký tự điều khiển khá phức tạp và quan trọng hơn cả là các giải pháp này không tương thích với nhau. Do đó, việc sử dụng đồng thời các

ngôn ngữ trong cùng một văn bản và trong cùng một font chữ thường là không thể hoặc rất khó khăn khi thực hiện. **Unicode** ra đời để nhằm khắc phục các nhược điểm nói trên và xây dựng một bộ mã chuẩn vạn năng dùng chung cho tất cả mọi ngôn ngữ trên thế giới. Unicode công–xoóc–xi–om (consortium) được thành lập vào năm 1991 như một tổ chức phi lợi nhuận nhằm phát triển chuẩn Unicode, các thành viên của Unicode công–xoóc–xi–om bao gồm các công ty hàng đầu thế giới trong lĩnh vực phần mềm như Adobe, Aldus, Borland, Digital, GO, IBM, HP, Lotus, Metaphor, Microsoft, NeXT, Novell, Sun, Symantec, Taligent, Unisys, WordPerfect, ... Unicode là bộ mã ký tự 16–bit, tương thích hoàn toàn với chuẩn quốc tế ISO/IEC 10646–1; 1993. **Số ký tự có thể biểu diễn là 2^{16}**. Với 65.536 ký tự, Unicode hầu như có thể mã hoá tất cả các ngôn ngữ trên thế giới. Ngoài ra, với cơ chế mở rộng UTF–16, Unicode và chuẩn ISO 10646 còn cho phép mã hoá hơn một triệu ký tự mà không cần phải dùng đến mã điều khiển Escape.

Một số đặc điểm của Unicode:

Mỗi ký tự trong bảng mã Unicode đều có độ dài cố định là 16 bit, nhờ đó, việc xử lý các xâu ký tự Unicode rất đơn giản, không phức tạp như các giải pháp dùng chuỗi ký tự điều khiển, phải có những thuật toán tương đối phức tạp để nhận diện ký tự trong một chuỗi các Byte. Trong khi đó, với Unicode, mỗi ký tự có độ dài đúng hai Byte nên có thể định vị rất dễ dàng các vị trí của ký tự trong chuỗi Byte cho trước.

Uncode tránh đến mức tối đa việc định nghĩa dư thừa, trùng lặp. Ví dụ, ký tự 'é' chỉ có một mã duy nhất dùng chung cho cả ngôn ngữ tiếng Việt, tiếng Czech, ... Cũng chính vì thế nên hệ thống chữ tiếng Việt có các mã nằm rải rác ở nhiều vị trí không liền nhau. Tiếng Hán, Nhật Bản và Hàn Quốc có khá nhiều ký tự trùng nhau nên chúng được dùng chung cho cả ba ngôn ngữ, tuy nhiên, trong Unicode vẫn có các vùng riêng để định nghĩa những ký tự đặc thù của ba ngôn ngữ này. Unicode, về cơ bản, không quy định việc bố trí các ký tự theo quy định sắp xếp của từng ngôn ngữ, điều này cũng là hệ quả của việc tránh định nghĩa các ký tự dư thừa do phải tận dụng các ký tự dùng chung nên không thể bố trí các ký tự theo từng vùng riêng cho từng ngôn ngữ. Hơn nữa, thực tế với nhiều ngôn ngữ, người ta phải dùng những thuật toán riêng để sắp xếp, chứ không thể sắp xếp theo thứ tự của chúng trong bảng chữ cái (tiếng Việt là một điển hình).

Unicode đã được cài đặt trong các hệ điều hành phổ biến hiện nay. Muốn sử dụng Unicode cần phải có những phần mềm hỗ trợ hiển thị hoặc cho phép gõ ký tự theo chuẩn Unicode, ngoài ra cũng cần phải có font chữ Unicode được cài đặt trong hệ thống.

Tiếng Việt trong bộ mã Unicode:

Tiếng Việt được xếp vào họ La–tinh mở rộng 1 (Latin Extended 1). Tuy các kýý tự tiếng Việt phân bố không tập trung, nhưng có một thuận lợi lớn là tiếng Việt đuược xếp vào họ La–tinh, một thành phần cơ bản của hầu hết tất cả các font cchữ Unicode, có nghĩa là tiếng Việt có mặt ở mọi nơi trong bất kỳ bộ font nào của các ngôn ngữ và như thế có thể đọc được tiếng Việt ở mọi nơi có cài đặt font Unicoode. Trong khi đó các ngôn ngữ không thuộc họ La–tinh như Trung Quốc, Nhật Bản, Lào, Thái Lan,... thì không phải lúc nào cũng có sẵn trong các font Unicode.

Tiếng Việt trong Unicode có thể có hai dạng: ký tự dựng sẵn và ký tự tổ hợp. Unicode có đủ 134 ký tự cho tất cả chữ hoa và chữ thường trong bảng chữ cái tiếng Việt, đồng thời có mã cho 5 dấu thanh (huyền, sắc, hỏi, ngã, nặng) để tạo ra cácc ký tự Việt dạng tổ hợp, ngoài ra Unicode còn có dấu riệng để biểu diễn đơn vị ttiền đồng Việt Nam.

I.2. Hệ thống máy tính

Mỗi loại máy tính có thể có hình dạng hoặc cấu trúc khác nhau. Một cách tổng quát, máy tính điện tử là một hệ xử lý thông tin tự động gồm hai phần chính: pphần cứng và phần mềm.

Phần cứng (*hardware*) có thể được hiểu đơn giản là tất cả các cấu kiện, linh kkiện điện, điện tử trong một hệ máy.

Phần mềm (*software*) có thể xem như một bộ chương trình bao gồm các chỉ thị điện tử ra lệnh cho máy tính thực hiện một điều nào đó theo yêu cầu của ngườii sử dụng. Phần mềm có thể được ví như phần hồn của máy tính mà phần cứng cuaa nó được xem như phần xác.

I.2.1. Hệ thống máy tính

I.2.1.1. Tổ chức bên trong máy tính

a. Mô hình cơ bản của máy tính

Chức năng của hệ thống máy tính

Máy tính thực hiện các chức năng cơ bản sau:

- **Xử lý dữ liệu**: Đây là chức năng quan trọng nhất của máy tính. Dữ liệuu có thể có rất nhiều dạng khác nhau và có yêu cầu xử lý khác nhau.

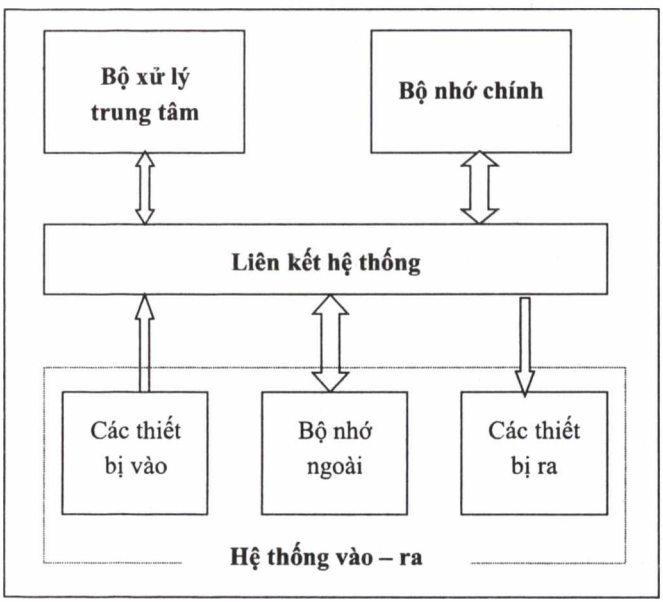
Lưu trữ dữ liệu: Các dữ liệu đưa vào máy tính có thể được lưu trong bộ nhớ để khi cần sẽ được lấy ra xử lý. Cũng có khi dữ liệu đưa vào được xử lý ngay. Các kết quả xử lý được lưu trữ lại trong bộ nhớ và sau đó có thể phục vụ cho các xử lý tiếp theo.

Trao đổi dữ liệu: Máy tính cần phải trao đổi dữ liệu giữa các thành phần bên trong và với thế giới bên ngoài. Các thiết bị vào–ra được coi là nguồn cung cấp dữ liệu hoặc nơi tiếp nhận dữ liệu. Tiến trình trao đổi dữ liệu với các thiết bị gọi là *vào–ra*. Khi dữ liệu được vận chuyển trên khoảng cách xa với các thiết bị hoặc máy tính gọi là *truyền dữ liệu* (data communication).

Điều khiển: Cuối cùng, máy tính phải điều khiển các chức năng trên.

Cấu trúc của hệ thống máy tính.

Hệ thống máy tính bao gồm các thành phần cơ bản sau: đơn vị xử lý trung tâm (Central Processing Unit – CPU), bộ nhớ chính (Main Memory), hệ thống vào ra (Input–Output System) và liên kết hệ thống (System Interconnection) như chỉ ra trong hình I.3 dưới đây, với các chức năng chính của các thành phần:



Hình I.3. Các thành phần chính của hệ thống máy tính

- **Bộ xử lý trung tâm** (Central Processing Unit – CPU): Điều khiển các hoạt động của máy tính và thực hiện xử lý dữ liệu.
- **Bộ nhớ chính** (Main Memory): Lưu trữ chương trình và dữ liệu.
- **Hệ thống vào ra** (Input–Output System): Trao đổi thông tin giữa thế giới bên ngoài với máy tính.
- **Liên kết hệ thống** (System Interconnection): Kết nối và vận chuyển thôông tin giữa CPU, bộ nhớ chính và hệ thống vào ra của máy tính với nhau.

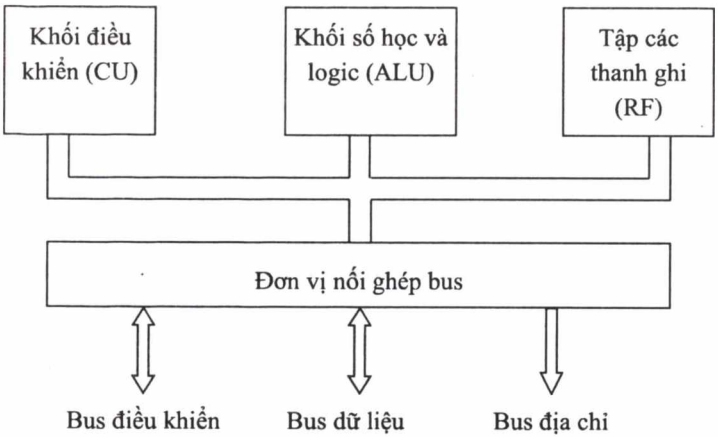
Hoạt động của máy tính

Hoạt động cơ bản của máy tính là thực hiện chương trình. Chương trình gồm một tập các lệnh được lưu trữ trong bộ nhớ. Việc thực hiện chương trình là việc lặp lại *chu trình lệnh* bao gồm các bước sau:

- CPU phát địa chỉ từ con trỏ lệnh đến bộ nhớ nơi chứa lệnh cần nhận.
- CPU nhận lệnh từ bộ nhớ đưa về thanh ghi lệnh.
- Tăng nội dung con trỏ lệnh để trỏ đến nơi lưu trữ lệnh kế tiếp.
- CPU giải mã lệnh để xác định thao tác của lệnh.
- Nếu lệnh sử dụng dữ liệu từ bộ nhớ hay cổng vào ra thì cần phải xác định địa chỉ nơi chứa dữ liệu.
- CPU nạp các dữ liệu cần thiết vào các thanh ghi trong CPU.
- Thực thi lệnh.
- Ghi kết quả vào nơi yêu cầu.
- Quay lại bước đầu tiên để thực hiện lệnh tiếp theo.

b. Bộ xử lý trung tâm – CPU

Bộ xử lý trung tâm (Central Processing Unit– CPU) điều khiển các thành phần của máy tính, xử lý dữ liệu. CPU hoạt động theo chương trình nằm trong bộ nhớ chính, nhận các lệnh từ bộ nhớ chính, giải mã lệnh để phát ra các tín hiệu điều khiển thực hiện lệnh. Trong quá trình thực hiện lệnh, CPU có trao đổi với bộ nhớ chính và hệ thống vào ra. CPU có ba bộ phận chính: khối điều khiển, khối tính toán số học và logic, tập các thanh ghi.



Hình I.4. Mô hình cơ bản của CPU

- **Khối điều khiển** (*Control Unit* – CU)

  Nhận lệnh của chương trình từ bộ nhớ trong đưa vào CPU. Nó có nhiệm vụ giải mã các lệnh, tạo ra các tín hiệu điều khiển công việc của các bộ phận khác của máy tính theo yêu cầu của người sử dụng hoặc theo chương trình đã cài đặt.

- **Khối tính toán số học và logic** (*Arithmetic – Logic Unit* – ALU)

  Bao gồm các thiết bị thực hiện các phép tính số học (cộng, trừ, nhân, chia,...), các phép tính logic (AND, OR, NOT, XOR) và các phép tính quan hệ (so sánh lớn hơn, nhỏ hơn, bằng nhau,...)

  Dữ liệu từ bộ nhớ hay các thiết bị vào–ra sẽ được chuyển vào các thanh ghi của CPU, rồi chuyển đến ALU. Tại đây, dữ liệu được tính toán rồi trả lại các thanh ghi và chuyển về bộ nhớ hay các thiết bị vào–ra.

  Độ dài từ của các toán hạng được đưa vào tính toán trực tiếp ở khối ALU. Độ dài phổ biến với các máy tính hiện nay là 32 hay 64 bit.

  Ban đầu, ALU chỉ gồm khối tính toán số nguyên IU (Integer Unit). Để tăng khả năng tính toán đối với số dấu phảy động, khối ALU hiện nay được

bổ sung thêm khối tính toán dấu phảy động FPU (Floating Point Uniit) – hay còn gọi là bộ đồng xử lý (Co–processing Unit).

- **Tập các thanh ghi** (*Register File*)

  Được gắn chặt vào CPU bằng các mạch điện tử làm nhiệm vụ bộ nhớ trung gian cho CPU. Các thanh ghi mang các chức năng chuyên dụng giúp tăng tốc độ trao đổi thông tin trong máy tính. Trên các CPU hiện nay có từ vài chục đến vài trăm thanh ghi. Độ dài của các thanh ghi cũng khác nhau, từ 8 đến 64 bit.

  Ngoài ra, CPU còn được gắn với một đồng hồ (clock) hay còn gọi là bộ tạo xung nhịp. Tần số đồng hồ càng cao thì tốc độ xử lý thông tin càng nhanh. Thường thì đồng hồ được gắn tương xứng với cấu hình máy và có các tần số dao động (cho các máy Pentium 4 trở lên) là 2.0 GHz, 2.2 GHz,... hoặc cao hơn.

***Bộ vi xử lý** (Microprocessor)*

CPU được chế tạo trên một vi mạch và được gọi là bộ vi xử lý. Vì vậy, chúng ta có thể gọi CPU là bộ vi xử lý. Tuy nhiên, các bộ vi xử lý hiện nay có cấu trúc phức tạp hơn nhiều so với một CPU cơ bản.

c. Bộ nhớ

Bộ nhớ là thiết bị lưu trữ thông tin trong quá trình máy tính xử lý. Bộ nhớ bao gồm bộ nhớ trong và bộ nhớ ngoài.

Bộ nhớ trong

Bộ nhớ trong (*Internal Memory*) là những thành phần nhớ mà CPU có thể trao đổi trực tiếp: các lệnh mà CPU thực thi, các dữ liệu mà CPU sử dụng đều phải nằm ở bộ nhớ trong. Bộ nhớ trong có dung lượng không thật lớn, song có tốc độ trao đổi thông tin cao.

Bộ nhớ chính

Là thành phần quan trọng nhất của bộ nhớ trong, vì vậy nhiều khi người ta đồng nhất bộ nhớ chính với bộ nhớ trong. Bộ nhớ chính tổ chức thành các ngăn theo byte và các ngăn nhớ này được đánh địa chỉ trực tiếp bởi CPU, có nghĩa là mỗi ngăn nhớ của bộ nhớ chính được gán một địa chỉ xác định. CPU muốn đọc/ghi vào ngăn nhớ nào, nó phải biết được địa chỉ của ngăn nhớ đó.

Nội dung của ngăn nhớ là giá trị được ghi trong đó. Số bit được dùng để đánh địa chỉ của ngăn nhớ sẽ quyết định dung lượng tối đa của bộ nhớ chính. Ví dụ:

- Dùng 16 bit địa chỉ thì dung lượng tối đa của bộ nhớ là $2^{16} = 2^6 \times 2^{10} = 64$ KB.
- Bộ xử lý Pentium III có 36 bit địa chỉ, do đó có khả năng quản lý tối đa $2^6 \times 2^{30} = 64$ GB.

Chú ý Kích thước một ngăn nhớ (ô nhớ) thường là 1 byte. Nội dung của ngăn nhớ có thể hay đổi còn địa chỉ ngăn nhớ thì cố định.

Bộ nhớ chính của máy tính được thiết kế bằng bộ nhớ bán dẫn với hai loại ROM và RAM trong đó:

- **ROM (Read Only Memory)** là bộ nhớ chỉ đọc thông tin, dùng để lưu trữ các chương trình hệ thống, chương trình điều khiển việc nhập xuất cơ sở (ROM–BIOS: ROM–Basic Input/Output System). Thông tin trên ROM không thay đổi và không bị mất ngay cả khi không có điện.
- **RAM (Random Access Memory)** là bộ nhớ truy xuất ngẫu nhiên, được dùng để lưu trữ dữ liệu và chương trình trong quá trình thao tác và tính toán. RAM có đặc điểm là nội dung thông tin chứa trong nó sẽ mất đi khi mất điện hoặc tắt máy. Dung lượng bộ nhớ RAM cho các máy tính hiện nay thông thường vào khoảng 2 GB, 4 GB[1] và có thể hơn nữa.

Ngoài ra, trong máy tính cũng còn phần bộ nhớ khác: ***Cache Memory***, cũng thuộc bộ nhớ trong. Bộ nhớ cache được đặt đệm giữa CPU và bộ nhớ trong nhằm làm tăng tốc độ trao đổi thông tin. Bộ nhớ cache có dung lượng nhỏ. Nó chứa một phần chương trình và dữ liệu mà CPU đang xử lý, do vậy thay vì lấy lệnh và dữ liệu từ bộ nhớ chính, CPU sẽ lấy trên cache. Hầu hết các máy tính hiện nay đều có cache tích hợp trên chip vi xử lý.

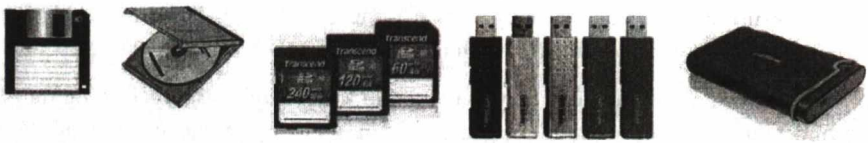
Bộ nhớ ngoài

Bộ nhớ ngoài (*External Memory*): là thiết bị lưu trữ thông tin với dung lượng lớn, thông tin không bị mất khi không có điện. Các thông tin này có thể là phần mềm máy tính hay dữ liệu. Bộ nhớ ngoài được kết nối với hệ thống thông qua mô–đun nối ghép vào–ra. Như vậy, *bộ nhớ ngoài về chức năng thuộc bộ nhớ, song về cấu trúc lại thuộc hệ thống vào–ra*. Có thể cất giữ và di chuyển bộ nhớ ngoài độc lập với máy tính. Hiện nay[2] có các loại bộ nhớ ngoài như:

---

(1), (2) Tính tại thời điểm giáo trình được tái bản (2012).

- Đĩa mềm (Floppy disk): là loại đĩa đường kính 3.5 inch dung lượng 1.44 MB. Đĩa mềm hiện không còn được ưa chuộng vì dung lượng rất thấp và giá thành không rẻ.
- Đĩa cứng (Hard disk): phổ biến là đĩa cứng có dung lượng 160 GB, 250 GB, 320 GB, 500 GB và lớn hơn nữa.
- Đĩa quang (Compact disk): loại 4.72 inch, là thiết bị phổ biến dùng để lưu trữ các phần mềm mang nhiều thông tin, hình ảnh, âm thanh và thường được sử dụng trong các phương tiện đa truyền thông (multimedia). Có hai loại phổ biến là: đĩa CD (dung lượng khoảng 700 MB) và DVD (dung lượng khoảng 4.7 GB).
- Ổ cứng bên ngoài (HDD External Storage) kết nối với máy tính thông qua cổng USB, phổ biến với dung lượng 500 GB, 1 TB,...
- Các loại bộ nhớ ngoài khác như thẻ nhớ (Memory Stick, Compact Flash Card), USB Flash Drive có dung lượng phổ biến là 4 GB, 8 GB, 16 GB, ...



Floppy disk | Compact disk | Compact Flash Card | USB Flash Drive | HDD External Storage

Hình I.5. Một số loại bộ nhớ ngoài

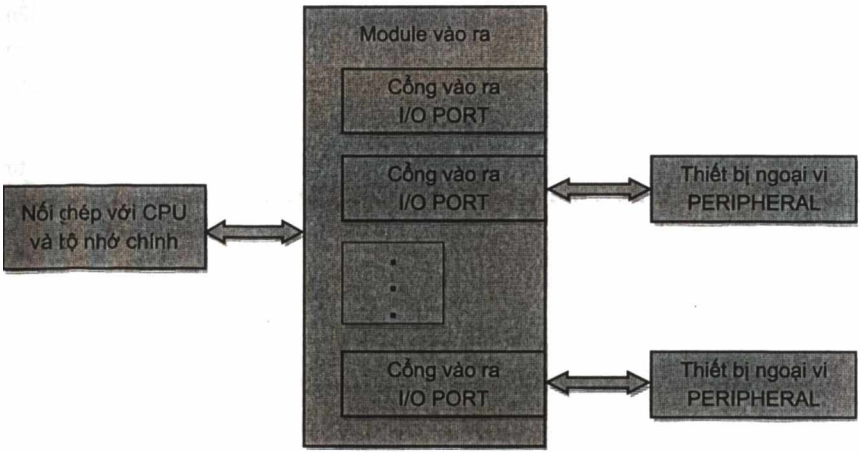
d. Hệ thống vào–ra

Chức năng của hệ thống vào–ra là trao đổi thông tin giữa máy tính với thế giới bên ngoài. Hệ thống vào–ra được xây dựng dựa trên hai thành phần: các ***thiết bị vào–ra*** (IO devices) hay còn gọi là thiết bị ngoại vi (Peripheral devices) và các ***mô–đun ghép nối vào–ra*** (IO Interface modules).

Mô–đun ghép nối vào–ra

Các thiết bị vào ra không kết nối trực tiếp với CPU mà được kết nối thông qua các mô–đun ghép nối vào–ra. Trong các mô đun ghép nối vào–ra có các cổng vào–ra

IO Port), các cổng này cũng được đánh địa chỉ bởi CPU, có nghĩa là mỗi cổng cũng có một địa chỉ xác định. Mỗi thiết bị vào–ra kết nối với CPU thông qua cổng tương ứng với địa chỉ xác định.



Hình I.6. Ghép nối vào–ra

Thiết bị vào–ra

Mỗi thiết bị vào–ra làm nhiệm vụ chuyển đổi thông tin từ một dạng vật lý nào đó về dạng dữ liệu phù hợp với máy tính hoặc ngược lại. Các thiết bị ngoại vi thông dụng là bàn phím, màn hình, máy in hay một máy tính khác. Người ta có thể phân các thiết bị ngoại vi ra nhiều loại:

- *Thiết bị thu nhận dữ liệu*: bàn phím, chuột, máy quét ảnh,...
- *Thiết bị hiển thị dữ liệu*: màn hình, máy in,...
- *Thiết bị nhớ*: các loại ổ đĩa.
- *Thiết bị truyền thông*: modem.
- *Thiết bị hỗ trợ đa phương tiện*: Hệ thống âm thanh, hình ảnh,...

Một số thiết bị vào

- **Bàn phím** (Keyboard, thiết bị nhập chuẩn): là thiết bị nhập dữ liệu và câu lệnh, bàn phím máy vi tính phổ biến hiện nay là một bảng chứa 104 phím có các tác dụng khác nhau.

Có thể chia làm ba nhóm phím chính:

- Nhóm phím đánh máy: gồm các phím chữ, phím số và phím các ký tự đặc biệt (~, !, @, #, $, %, ^,&, ?, ...).
- Nhóm phím chức năng (function keypad): gồm các phím từ F1 đến F12 và các phím như ← ↑ → ↓ (phím di chuyển từng điểm), phím PgUp (lên trang màn hình), PgDn (xuống trang màn hình), Insert (chèn), Delete (xoá), Home (về đầu), End (về cuối).
- Nhóm phím số (numeric keypad) như NumLock (cho các ký tự số), CapsLock (tạo các chữ in), ScrollLock (chế độ cuộn màn hình) thể hiện ở các đèn chỉ thị.

Bàn phím Chuột Máy quét Máy in

Hình I.7. Một số thiết bị vào–ra

- **Chuột** (Mouse): Là thiết bị cần thiết phổ biến hiện nay, nhất là các máy tính chạy trong môi trường Windows. Con chuột có kích thước vừa nắm tay, chuột di chuyển trên một tấm phẳng (mouse pad) theo hướng nào thì dấu nháy hoặc mũi tên trên màn hình sẽ di chuyển theo hướng đó tương ứng với vị trí của của viên bi hoặc tia sáng (chuột quang – optical mouse) nằm dưới bụng của nó. Một số máy tính có con chuột được gắn trên bàn phím.

- **Máy quét** (Scanner): Là thiết bị dùng để quét văn bản, hình vẽ, hình chụp và đưa vào máy tính. Thông tin nguyên thuỷ trên giấy sẽ được quét thành các tín hiệu số để tạo thành các tập tin ảnh (image file).
- **Ổ đĩa:** Khi đọc dữ liệu từ đĩa thì ổ đĩa đóng vai trò thiết bị vào.

Một số thiết bị ra

- **Màn hình** (Display hay Monitor, thiết bị ra chuẩn): Dùng để hiển thị thông tin cho người sử dụng xem. Thông tin được thể hiện ra màn hình bằng phương pháp ánh xạ bộ nhớ (memory mapping), với cách này màn hình chỉ việc đọc liên tục bộ nhớ và hiển thị (display) bất kỳ thông tin nào hiện có trong vùng nhớ ra màn hình.

  Màn hình phổ biến hiện nay trên thị trường là màn hình màu SVGA 15",17", 19" với độ phân giải có thể đạt tới 1280 × 1024 pixel.
- **Ổ đĩa:** Khi ghi dữ liệu lên đĩa thì ổ đĩa đóng vai trò thiết bị ra.
- **Máy in** (Printer): Là thiết bị ra để đưa thông tin ra giấy. Máy in phổ biến hiện nay là loại máy in ma trận điểm (dot matrix) loại 24 kim, máy in phun mực, máy in laser trắng đen hoặc màu.
- **Máy chiếu** (Projector): Chức năng tương tự màn hình, thường được sử dụng thay cho màn hình trong các buổi họp, báo cáo, thuyết trình, …

e. Liên kết hệ thống (buses)

Giữa các thành phần của một hệ thống máy tính hay ngay trong một thành phần phức tạp như CPU cũng cần trao đổi với nhau. Nhiệm vụ này được thực thi bởi hệ thống kết nối mà chúng ta quen gọi là bus. Tuỳ theo nhiệm vụ của chúng mà chúng ta phân làm ba loại chính:

- **Bus điều khiển** (Control bus): Chuyển các thông tin/tín hiệu điều khiển từ thành phần này đến thành phần khác: CPU phát tín hiệu để điều khiển bộ nhớ hay hệ thống vào–ra hoặc từ hệ thống vào–ra gửi tín hiệu yêu cầu đến CPU.
- **Bus dữ liệu** (Data bus): Làm nhiệm vụ chuyển tải dữ liệu (nội dung ngăn nhớ, kết quả xử lý) từ CPU đến bộ nhớ hay ngược lại hoặc từ bộ nhớ/CPU ra các thiết bị ngoại vi. Đây là loại bus hai chiều. Các máy tính hiện nay thường có độ rộng đường bus dữ liệu là 32 bit hay 64 bit.

- **Bus địa chỉ** (Address bus): chuyển tải địa chỉ của các ngăn nhớ khi muốn truy nhập (đọc/ghi) nội dung của ngăn nhớ đó hoặc địa chỉ cổng của các thiết bị mà CPU cần trao đổi. Độ rộng (số bit) của bus địa chỉ cho biết dung lượng cực đại của bộ nhớ mà CPU có thể quản lý được. Với độ rộng là n thì dung lượng bộ nhớ tối đa sẽ là 2^n.

I.2.1.2. Phần mềm máy tính

Việc xử lý thông tin của máy tính theo yêu cầu người dùng được tiến hành theo một quy trình tự động đã định sẵn gọi là chương trình (program). Như vậy, với mỗi yêu cầu của người dùng mà chúng ta tạm gọi là một bài toán (problem) hay một nhiệm vụ (task) cần một quy trình/chương trình. Ngày nay, người ta quen sử dụng một thuật ngữ tương đương song có nghĩa rộng hơn là phần mềm máy tính (Computer Software). Phần dưới đây sẽ đề cập đến những khái niệm rất cơ bản trong việc lập trình máy tính, bước quan trọng trong xây dựng các phần mềm máy tính.

a. Dữ liệu và giải thuật

Mỗi bài toán phải giải quyết thường bao gồm hai phần: phần *dữ liệu* và phần *xử lý*. Phần dữ liệu liên quan đến thông tin của bài toán cụ thể đó như: đầu vào – các dữ liệu được cung cấp để xử lý và đầu ra là kết quả xử lý. Phần xử lý là những thao tác phải được máy tính tiến hành nhằm đáp ứng yêu cầu người dùng.

Ví dụ: Bài toán đơn giản là sắp xếp một dãy số nguyên a_1, a_2,..., a_n theo chiều không giảm dần. Với bài toán này, dữ liệu đầu vào là số lượng phần tử n của dãy và n số nguyên a_i, i = 1, 2,..., n; đầu ra là dãy số đã sắp theo chiều không giảm. Các xử lý là các thao tác để từ một dãy số không có thứ tự chuyển về một dãy số có thứ tự không giảm.

Việc biểu diễn dữ liệu một cách trừu tượng là một vấn đề khá phức tạp, ví dụ như dãy số ở trên sẽ được lưu trữ ra sao trong máy tính (cả dãy vào lẫn dãy kết quả). Tạm thời chúng ta không xét tại mục này.

Việc xác định các thao tác xử lý để đáp ứng yêu cầu đặt ra của người dùng là một nhiệm vụ rất quan trọng. Nó không phải là phương pháp chung mà phải cụ thể về các thao tác và việc liên kết các thao tác đó. Việc xác định các thao tác xử lý được các nhà tin học gọi là xây dựng giải thuật/thuật toán (algorithm).

Thuật toán để giải một bài toán là một dãy hữu hạn các thao tác và trình tự thực hiện các thao tác đó sao cho sau khi thực hiện dãy thao tác này theo trình tự đã chỉ ra, với đầu vào (input) ta thu được kết quả đầu ra (output) mong muốn.

b. Chương trình và ngôn ngữ lập trình

Thuật toán mới chỉ ra cách giải quyết một bài toán theo kiểu tư duy của con người. Để máy có thể hiểu và tiến hành xử lý được, ta phải biến các bước thao tác thành các chỉ thị (statement) và biểu diễn trong dạng mà máy tính hiểu được. Quá trình này gọi là ***lập trình***. Giải thuật được biểu diễn dưới dạng một tập các chỉ thị của một ngôn ngữ nào đó gọi là *chương trình*. Ngôn ngữ dùng để lập trình gọi là *ngôn ngữ lập trình* – ngôn ngữ dùng để trao đổi với máy tính, làm máy tính hiểu và thực thi nhiệm vụ đã chỉ ra.

Tương tự với dữ liệu, máy tính không thể xử lý dữ liệu một cách hình thức như trong giải tích mà nó phải là những con số hay những giá trị cụ thể.

N. Wirth, người sáng lập ra trường phái lập trình cấu trúc, tác giả của ngôn ngữ lập trình Pascal nổi tiếng đã cho rằng:

Chương trình = Cấu trúc dữ liệu + Giải thuật

Có nhiều loại ngôn ngữ lập trình. Sự khác nhau giữa các loại liên quan đến mức độ phụ thuộc của chúng vào kiến trúc và hoạt động máy tính, phụ thuộc vào lớp/lĩnh vực ứng dụng. Có nhiều cách phân loại khác nhau và do đó các ngôn ngữ lập trình được phân thành các nhóm khác nhau. Người ta phân các ngôn ngữ theo một cách chung nhất thành ba nhóm:

1. **Ngôn ngữ máy**

   Mỗi loại máy tính đều có ngôn ngữ máy riêng. Đó là loại ngôn ngữ duy nhất để viết chương trình mà máy tính hiểu trực tiếp và thực hiện được. Các chỉ thị (lệnh) của ngôn ngữ này viết bằng mã nhị phân hay mã hec–xa. Nó gắn chặt với kiến trúc phần cứng của máy và do vậy khai thác được các đặc điểm phần cứng. Tuy nhiên, nó lại không hoàn toàn thuận lợi cho người lập trình do tính khó nhớ của mã, tính thiếu cấu trúc,...Vì thế, để viết một ứng dụng bằng ngôn ngữ máy là việc không dễ, nhất là phải tiến hành các thay đổi, chỉnh sửa hay phát triển thêm về sau.

2. **Hợp ngữ**

   Hợp ngữ cho phép người lập trình sử dụng một số từ tiếng Anh viết tắt để thể hiện các câu lệnh thực hiện. Ví dụ để cộng nội dung của hai thanh ghi AX và BX rồi ghi kết quả vào AX, ta có thể dùng câu lệnh hợp ngữ sau:

   ADD AX, BX

Một chương trình hợp ngữ phải được dịch ra ngôn ngữ máy nhờ chương trình hợp dịch trước khi máy tính thực hiện.

3. Ngôn ngữ bậc cao

Tuy hợp ngữ đã có nhiều ưu việt hơn so với ngôn ngữ máy song nó vẫn tồn tại nhiều hạn chế, nhất là tính không cấu trúc sẽ cản trở việc theo dõi, hiểu và chỉnh sửa chương trình. Từ những năm 50 của thế kỷ trước, khi máy tính xuất hiện, nhiều chuyên gia đã đề xuất sử dụng một ngôn ngữ nào đó ít phụ thuộc vào kiến trúc phần cứng máy tính, gần với tiếng Anh tự nhiên, có tính độc lập cao để khắc phục những hạn chế ở trên. Người ta gọi những ngôn ngữ lập trình này là *ngôn ngữ lập trình bậc cao*. Các chương trình viết trong ngôn ngữ này, trước khi để máy thực thi cần phải chuyển đổi sang ngôn ngữ máy. Quá trình chuyển đổi đó gọi là quá trình dịch.

Có hai phương thức dịch:

- *Thông dịch* (Interpreter): Bộ thông dịch đọc từng lệnh của chương trình nguồn, phân tích cú pháp của câu lệnh đó và nếu đúng thì thực hiện. Quá trình bắt đầu từ lệnh đầu tiên của chương trình đến lệnh cuối cùng nếu không có lỗi. Bộ thông dịch này giống như vai trò của một thông dịch viên (translator).
- *Biên dịch* (Compiler): Khác với thông dịch, trình biên dịch dịch toàn bộ chương trình nguồn sang ngôn ngữ đích. Với chương trình đích này, máy đã có thể hiểu được và biết cách thực thi. Quá trình biên dịch sẽ tạo ra chương trình đích chỉ khi các lệnh trong chương trình nguồn không có lỗi.

Cách ứng xử với lỗi cũng khác nhau. Có chương trình dịch cứ gặp lỗi thì ngưng lại như Turbo Pascal, song có chương trình dịch cứ dịch cho đến hết kể cả khi gặp lỗi như C hay một số ngôn ngữ khác.

Hiện tại có khá nhiều ngôn ngữ lập trình bậc cao. Ngôn ngữ đầu tiên phải kể đến là FORTRAN4 xuất hiện vào năm 1959, sau đó cải tiến thành phiên bản FORTRAN5 năm 1977. Tiếp theo là COBOL 1960, ngôn ngữ sử dụng cho nghiệp vụ quản lý. Vào những năm tiếp theo là ALGOL60, BASIC. Có những ngôn ngữ đã được hoàn thiện hơn như Visual Basic có nguồn gốc từ Basic. Ngoài ra còn nhiều ngôn ngữ khác ra đời như C, C++, Visual C++, Java, C#, Python,...

c. Phân loại phần mềm máy tính

Có nhiều cách phân loại phần mềm máy tính (Software – SW). Nếu phân loại theo quan điểm sử dụng chung thì phần mềm máy tính có hai loại:

- *Phần mềm hệ thống*: Là phần mềm điều khiển hoạt động bên trong của máy tính và cung cấp môi trường giao tiếp giữa người dùng và máy tính nhằm khai thác hiệu quả phần cứng phục vụ cho nhu cầu sử dụng. Loại phần mềm này đòi hỏi tính ổn định và tính an toàn cao. Chẳng hạn các hệ điều hành máy đơn hay hệ điều hành mạng, các tiện ích hệ thống,...
- *Phần mềm ứng dụng*: Là phần mềm dùng để giải quyết các vấn đề phục vụ cho các hoạt động khác nhau của con người như quản lý, kế toán, soạn thảo văn bản, trò chơi,... Nhu cầu về phần mềm ứng dụng ngày càng tăng và đa dạng.

Nếu phân theo đặc thù ứng dụng và môi trường thì phần mềm có thể gồm các loại sau:

- Phần mềm thời gian thực (Real–time SW).
- Phần mềm nghiệp vụ (Business SW).
- Phần mềm tính toán Khoa học và Kỹ thuật (Eng.&Scie. SW).
- Phần mềm nhúng (Embedded SW).
- Phần mềm trên Web (Web–based SW).
- Phần mềm trí tuệ nhân tạo (AI SW – Artificial Intelligence SW).
- ....

I.2.2. Mạng máy tính

I.2.2.1. Khái niệm và lịch sử phát triển của mạng máy tính

Khái niệm mạng máy tính

Mạng máy tính hay ***mạng*** (*computer network, network*) là một tập hợp gồm nhiều máy tính và/hoặc thiết bị xử lý thông tin được kết nối với nhau qua các đường truyền và có sự trao đổi dữ liệu với nhau.

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

Ví dụ về mạng:

- Mạng tại Trung tâm Máy tính, Viện Công nghệ thông tin và truyền thông, trường Đại học Bách Khoa Hà Nội.
- Mạng của Tổng cục thuế.
- Mạng Internet.

Lịch sử phát triển của mạng

- Máy tính ra đời từ những năm 1950. Đến đầu những năm 1960 mạng máy tính bắt đầu xuất hiện. Lúc đầu mạng có dạng là một máy tính lớn nối với nhiều trạm cuối (*terminal*). Đến đầu những năm 1970, mạng máy tính là các máy tính độc lập được nối với nhau. Quy mô và mức độ phức tạp của mạng ngày càng tăng.
- Hiện nay mạng máy tính phát triển rất mạnh trên mọi lĩnh vực ở khắp mọi nơi. Ngày càng hiếm các máy tính đơn lẻ, không nối mạng. Ngay các máy tính cá nhân ở gia đình cũng được kết nối Internet qua đường điện thoại hoặc sóng. Mạng trở thành một yếu tố không thể thiếu của công nghệ thông tin nói riêng, cũng như trong đời sống nói chung.

1.2.2. Phân loại mạng máy tính

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

Cách 1. Theo mối quan hệ giữa các máy trong mạng, có hai loại:

- *Mạng bình đẳng (peer–to–peer)*: Các máy có quan hệ ngang hàng, một máy có thể yêu cầu một máy khác phục vụ.
- *Mạng khách/chủ (client/server)*: Một số máy là server (máy chủ) chuyên phục vụ các máy khác gọi là máy khách (client) hay máy trạm (workstation) khi có yêu cầu. Các dịch vụ có thể là cung cấp thông tin, tính toán hay các dịch vụ Internet.

Cách 2. Theo quy mô địa lý. Tuỳ theo quy mô địa lý, có thể phân ra ba loại chính là:

- *LAN (Local Area Network):* Mạng cục bộ trong phạm vi nhỏ, ví dụ bán kính 500 m, số lượng máy tính không quá nhiều, mạng không quá phức tạp. Với công nghệ mạng hiện nay, tốc độ truyền của mạng cục bộ có thể đạt tới 100 Mb/s. Mạng cục bộ thuộc sở hữu riêng của một tổ chức nào đó nên việc quản lý là tập trung và thống nhất.

- *WAN (Wide Area Network):* Mạng diện rộng, các máy tính có thể ở các thành phố khác nhau. Bán kính có thể 100 – 200 km. Ví dụ mạng của Tổng cục thuế.
- *GAN (Global Area Network):* Mạng toàn cầu, máy tính ở nhiều nước khác nhau. Thường mạng toàn cầu là kết hợp của nhiều mạng con. Ví dụ mạng Internet.

I.2.2.3. Các thành phần cơ bản của một mạng máy tính

Một mạng máy tính có thể có các thành phần sau:

- *Các máy tính*

  Để xử lý, lưu trữ và trao đổi thông tin. Ta cũng thường gọi mỗi máy tính trong mạng máy tính là một *nút* của mạng.

- *Vi mạng*

  Vi mạng (Network Interface Card, NIC) cho mỗi máy tính có chức năng giao tiếp giữa máy tính và đường truyền.

- *Đường truyền*

  Đường truyền, chính xác hơn còn gọi là đường truyền vật lý, là phương tiện (media) truyền tải dữ liệu, là nơi trên đó dữ liệu được truyền đi. Ta có thể chia đường truyền thành hai loại là *hữu tuyến* và *vô tuyến*.

- *Các thiết bị kết nối mạng*

  Để liên kết các máy tính và các mạng với nhau như Hub, Switch, Router, …

- *Các thiết bị đầu cuối (terminal)*

  Như máy tính, máy in, máy phô– tô, máy quét, camera mạng,…

- *Các phụ kiện mạng*

  Các phụ kiện mạng khác gồm giắc cắm, ổ cắm, ….

- *Hệ điều hành mạng*

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

- *Các phần mềm mạng cho máy tính*

  Hiện nay nói chung các hệ điều hành đều có sẵn khả năng kết nối mạng. Trong trường hợp hệ điều hành của máy tính không có sẵn khả năng kết nối mạng thì các phần mềm này là cần thiết.

- *Các ứng dụng trên mạng*

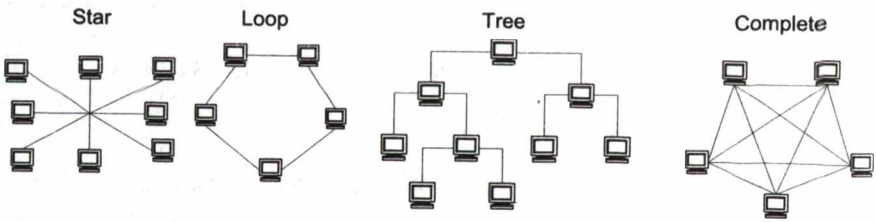
  Ví dụ: Email, Web, Chat, Game online,...

- *Kiến trúc mạng máy tính*

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

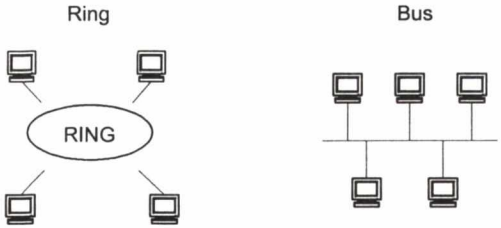
  Có hai kiểu nối mạng chủ yếu là *điểm–điểm* (point to point) và *quảng bá* (broadcast).

  Trong kiểu điểm–điểm, các đường truyền nối các nút thành từng cặp. Như vậy một nút sẽ gửi dữ liệu đến nút lân cận nó (nút được nối trực tiếp với nó). Nút lân cận sẽ tiếp tục chuyển dữ liệu đến nút lân cận trực tiếp khác,... cho đến khi dữ liệu đến đích. Kiểu nối mạng điểm–điểm có các dạng chính là : hình sao (*star*), chu trình (*loop*), cây (*tree*) và đầy đủ (*complete*).



Hình I.8. Một số topo mạng kiểu điểm–điểm

Trong kiểu quảng bá, các nút nối vào đường truyền chung. Như vậy khi một nút gửi dữ liệu, các nút còn lại đều nhận được. Do đó, dữ liệu gửi đi cần có địa chỉ đích. Khi một nút nhận được dữ liệu, nó sẽ kiểm tra địa chỉ đích xem có phải gửi cho mình không.



Hình I.9. Một số topo mạng kiểu quảng bá

Giao thức mạng: Việc trao đổi thông tin phải tuân theo những quy tắc nhất định, ngay cả khi hai người nói chuyện với nhau cũng vậy: có người nói thì phải có người nghe. Việc truyền tín hiệu trên mạng cũng vậy, cần phải có những quy tắc, quy ước về nhiều mặt, từ khuôn dạng của dữ liệu (cú pháp, ngữ nghĩa) đến các thủ tục gửi, nhận dữ liệu, kiểm soát hiệu quả và chất lượng truyền tin, xử lý các lỗi và sự cố. Tập hợp tất cả những quy tắc, quy ước đó gọi là giao thức (protocol) của mạng.

I.2.2.4. Mạng Internet

Khái niệm về Internet

Internet là một mạng máy tính có quy mô toàn cầu (GAN), gồm rất nhiều mạng con và máy tính nối với nhau bằng nhiều loại phương tiện truyền. Internet không thuộc sở hữu của riêng ai mà chỉ có các uỷ ban điều phối và kỹ thuật giúp điều hành Internet.

Internet ban đầu có nguồn gốc là mạng của Bộ Quốc phòng Mỹ (DoD) dùng để đảm bảo liên lạc giữa các đơn vị quân đội. Sau đó, nó phát triển thành mạng cho các trường đại học và viện nghiên cứu. Cuối cùng mạng có quy mô toàn cầu và trở thành mạng Internet.

Các dịch vụ chính của Internet

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

- Truyền tệp tin (FTP, File Transfer Protocol).

- Truy nhập máy tính từ xa (Telnet).
- Web (WWW) để tìm kiếm và khai thác thông tin trên mạng.
- Thư điện tử (E–mail).
- Tán gẫu (Chat) trên mạng.
- ...

Lợi ích của Internet

Trong thời đại của công nghệ thông tin hiện nay, Internet có nhiều lợi ích như truyền tin, phổ biến tin, thu thập tin, trao đổi tin một cách nhanh chóng thuận tiện và rẻ tiền hơn so với các phương tiện khác như điện thoại, fax. Internet ảnh hưởng đến toàn bộ thế giới trong mọi ngành nghề, mọi lĩnh vực xã hội và trở thành một yếu tố quan trọng không thiếu được trong thời đại hiện nay.

Làm sao để có được các dịch vụ Internet?

Để kết nối được Internet ta cần :

- Máy tính có Modem (Dial–up, ADSL) và/hoặc vỉ mạng.
- Có thuê bao kết nối với Internet: qua mạng, qua đường điện thoại, đường thuê riêng. Thông thường hiện nay kết nối qua điện thoại hoặc qua ADSL.
- Có tài khoản Internet ở trên mạng hay của một nhà cung cấp dịch vụ Internet (Internet Service Provider, ISP), ví dụ như VNN, FPT.
- Có các phần mềm Internet thông dụng như trình duyệt web (Web browser) để hiển thị nội dung các trang web (IE, FireFox, Opera,...), phần mềm để nhận, xem và gửi thư điện tử (Outlook), phần mềm tán gẫu trên mạng (Yahoo Messenger, Skype,...).

Một số khái niệm:

Địa chỉ internet:

Địa chỉ IP (Internet Protocol Address): Khi tham gia vào internet, các máy tính còn gọi là host) phải mang một địa chỉ IP dùng để nhận dạng. Địa chỉ IP này là duy nhất và đại diện cho chính máy tính tham gia vào mạng internet. Địa chỉ IPv4 gồm bốn số thập phân nằm trong dải 0–255 (còn gọi là Octet), phân cách nhau bởi

dấu chấm. Mỗi số được lưu bởi 1 byte và IPv4 có kích thước là 4 byte. Ví dụ: 192.168.24.5, 172.16.0.4, ... IPv4 có khả năng cung cấp 2^{32} (khoảng hơn 4 tỉ) địa chỉ. Kho địa chỉ này đang cạn kiệt dần. Do đó, đã xuất hiện IPv6 sử dụng 128 bit, dài gấp 4 lần IPv4 (32 bit), nên có khả năng cung cấp 2^{128} địa chỉ. Địa chỉ IP có thể dễ dàng bị phát hiện ra, người ta có thể lấy được địa chỉ IP máy tính của bạn khi bạn lướt qua một trang web, hay khi bạn ở trên IRC (Internet Relay Chat, một dạng liên lạc cấp tốc qua mạng), trên ICQ (cộng đồng I Seek You), hoặc khi bạn kết nối với một ai đó, nếu ai đó gửi cho bạn một email với một đoạn mã java tóm IP,...

Hệ thống tên miền DNS (Domain Name System): Do người sử dụng khó có thể nhớ được chuỗi số IP khá dài này, vì thế, bên cạnh địa chỉ IP bao giờ cũng có thêm một cái tên mang một ý nghĩa nào đó dễ nhớ gọi là tên miền. Người ta xây dựng hệ thống địa chỉ internet, gọi là hệ thống tên miền DNS, để đặt tên cho các host trên internet. Ví dụ: www.dantri.com.vn, www.soict.hut.edu.vn, www.google.com.vn, www.bbc.co.uk, www.youtube.com, www.facebook.com, ...

Mỗi host trên internet có hai địa chỉ: địa chỉ IP và địa chỉ tên miền được ánh xạ với nhau. Khi người dùng sử dụng tên miền, nó sẽ được ánh xạ sang địa chỉ IP tương ứng.

Tên miền được phân làm ba nhóm: tên miền quốc tế, tên miền quốc gia, tên miền thứ cấp. Tên miền quốc tế và tên miền quốc gia cấp 1 do tổ chức Internet Corporation for Assigned Names and Numbers (viết tắt là ICANN) quản lý. Tên miền quốc gia cấp thấp hơn do cơ quan quản lý tên miền của từng nước quản lý. Ở Việt Nam, cơ quan quản lý tên miền quốc gia là Trung tâm Internet Việt Nam (VNNIC) trực thuộc Bộ Thông tin và Truyền thông. Tên miền là địa chỉ duy nhất trên internet nên không bao giờ có sự trùng nhau. Khách hàng đăng ký là người có toàn quyền sở hữu tên miền. Tên miền khi được đăng ký, thường có giá trị trong một khoảng thời gian nhất định, sau đó, nếu khách hàng tiếp tục có nhu cầu sử dụng thì phải làm các thủ tục gia hạn.

Tên miền quốc tế là tên miền có phần đuôi là com, edu, net, org, gov, int, mil, biz, info, ac, pro.

Tên	Mô tả
.com	Dành cho tổ chức,doanh nghiệp, cá nhân hoạt động thương mại.
.edu	Dành cho các cơ quan, tổ chức, doanh nghiệp có hoạt động liên quan tới giáo dục, đào tạo.
.net	Dành cho các cơ quan, tổ chức, doanh nghiệp thực hiện chức năng về mạng nói chung.
.org	Dành cho các tổ chức chính trị, xã hội và các cơ quan, tổ chức, doanh nghiệp có hoạt động liên quan đến lĩnh vực chính trị, xã hội.
.gov	Dành cho các cơ quan, tổ chức thuộc bộ máy nhà nước ở trung ương và địa phương.
.int	Dành cho các tổ chức quốc tế.
.mil	Dành cho quân sự (Hoa Kỳ).
.biz	Dành cho các tổ chức, doanh nghiệp, cá nhân kinh doanh, tương đương với ".com"
.info	Dành cho các tổ chức cung cấp các nguồn dữ liệu thông tin về các lĩnh vực kinh tế, chính trị, văn hóa, xã hội và các cơ quan, tổ chức, doanh nghiệp liên quan tới lĩnh vực cung cấp các nguồn dữ liệu thông tin và thông tin cá nhân.
.ac	Dành cho các tổ chức nghiên cứu và các cơ quan, tổ chức, doanh nghiệp có hoạt động liên quan tới lĩnh vực nghiên cứu.
.pro	Dành cho các tổ chức, cá nhân hoạt động trong những lĩnh vực có tính chuyên ngành cao.

.health	Dành cho các tổ chức y tế, dược phẩm và các cơ quan, tổ chức, doanh nghiệp có hoạt động liên quan tới lĩnh vực y tế, dược phẩm.
.name	Dành cho tên riêng của cá nhân tham gia hoạt động Internet.

Tên miền quốc gia là tên miền có phần đuôi là ký hiệu của mỗi quốc gia: vn (Việt Nam), cn (Trung Quốc), tw (Đài Loan), kr (Hàn Quốc), jp (Nhật Bản), au (Úc), uk (Anh), ve (Venezuela), fr (Pháp), ca (Canada), us (Hoa Kỳ), de (Đức), ru (Nga),...

Tên miền cấp 2 của các quốc gia do tổ chức quản lý mạng internet của quốc gia đó định nghĩa, có thể định nghĩa khác đi, nhiều lên hay ít đi, nhưng thông thường các quốc gia vẫn định nghĩa các lĩnh vực tương tự như các lĩnh vực dùng chung nêu trên. Ví dụ: .com.vn, .edu.vn, .net.vn, .org.vn, .gov.vn,...

Đường dẫn URL:

URL, viết tắt của *Uniform Resource Locator*, được dùng để tham chiếu tới tài nguyên trên internet. URL mang lại khả năng siêu liên kết cho các trang mạng. Các tài nguyên khác nhau được tham chiếu tới bằng địa chỉ, chính là URL. Một URL có nhiều thành phần, quan sát ví dụ sau:

http://video.google.co.uk:80/videoplay?docid=–724692761280&hl=en#00h02m30

Đây là các thành phần chính của URL này:

- Giao thức là http. Ngoài ra còn có nhiều giao thức khác như https, ftp,...
- Tên host hay hostname là video.google.co.uk.
- Tên miền thứ cấp là video.
- Tên miền chính là google.co.uk.
- Tên miền cấp 1 (top–level domain) là uk.Tên miền uk là tên miền quốc gia.
- Tên miền cấp 2 (second–level domain) là co.uk.
- Cổng giao tiếp là 80, đây là cổng ngầm định của dịch vụ Web. Có thể sử dụng các cổng khác như 8080 hay 8000. Khi cổng là 80, phần lớn người ta không cần thêm vào đường dẫn.

Nhà cung cấp truy cập Internet (Internet Access Provider – IAP): Là nhà cung cấp dịch vụ đường truyền để kết nối với internet, quản lý cổng kết nối với quốc tế. Ví dụ: ở Việt Nam có Công ty Điện toán và Truyền số liệu VDC.

Nhà cung cấp dịch vụ internet (Internet Service Provider – ISP): Là nhà cung cấp các dịch vụ internet cho các tổ chức, cá nhân. ISP phải thuê đường truyền và cổng của IAP. Ví dụ: ở Việt Nam có VDC, FPT, Viettel.

I.2.3. Giới thiệu hệ điều hành

I.2.3.1. Các khái niệm cơ bản

a. Khái niệm hệ điều hành

Hệ điều hành là một trong các phần mềm hệ thống có tính phổ dụng. Có nhiều cách diễn đạt khác nhau về hệ điều hành xuất phát từ góc độ của những người sử dụng khác nhau. Tuy vậy có thể diễn đạt như sau:

Hệ điều hành là hệ thống chương trình đảm bảo quản lý tài nguyên của hệ thống tính toán và cung cấp các dịch vụ cho người sử dụng.

Thông thường trong các máy tính hiện nay, hệ điều hành được cài đặt trên đĩa.

Nhiệm vụ cụ thể hơn của hệ điều hành là:

- Khởi động máy tính, tạo môi trường giao tiếp cho người sử dụng.
- Tự động điều khiển và kiểm soát hoạt động của các thiết bị (ổ đĩa, bàn phím, màn hình, máy in,...).
- Quản lý việc cấp phát tài nguyên của máy tính như bộ xử lý trung tâm, bộ nhớ, các thiết bị vào ra...
- Quản lý các chương trình đang thực hiện trên máy tính.
- Thực hiện giao tiếp với người sử dụng để nhận lệnh và thực hiện lệnh.

Hệ điều hành là phần mềm hệ thống, vì vậy nó phụ thuộc vào cấu trúc của máy tính. Mỗi loại máy tính có hệ điều hành khác nhau. Ví dụ:

+ Máy tính lớn IBM360 có hệ điều hành DOS, TOS.

+ Máy tính lớn EC–1022 có hệ điều hành OC–EC.
+ Máy tính cá nhân PC–IBM có hệ điều hành MS–DOS.
+ Mạng máy tính có các hệ điều hành mạng NETWARE, UNIX WINDOWS–NT...
+ ...

b. Tệp (File)

Tệp (còn gọi là tập tin hay tệp tin) là tập hợp các dữ liệu có liên quan với nhau và được tổ chức theo một cấu trúc nào đó, thường được lưu trữ bên ngoài máy tính.

Nội dung của tệp có thể là chương trình, dữ liệu, văn bản,... Mỗi tập tin được lưu lên đĩa với một tên riêng phân biệt. Mỗi hệ điều hành có quy ước đặt tên khác nhau, tên tập tin thường có hai phần: phần tên (name) và phần mở rộng (extension). Phần tên là phần bắt buộc phải có của một tập tin, còn phần mở rộng thì có thể có hoặc không.

- Phần tên: Bao gồm các ký tự chữ từ A đến Z, các chữ số từ 0 đến 9, các ký tự khác như #, $, %, ~, ^, @, (,), !, _, khoảng trắng (tên tệp trên DOS không có khoảng trắng). Phần tên do người tạo ra tập tin đặt. Với MS–DOS, phần tên có tối đa 8 ký tự, với Windows phần tên có thể đặt tối đa 128 ký tự.
- Phần mở rộng: Thường dùng 3 ký tự trong các ký tự nêu trên. Thông thường phần mở rộng do chương trình ứng dụng tạo ra tập tin tự đặt.
- Giữa phần tên và phần mở rộng có một dấu chấm (.) ngăn cách.

Ta có thể căn cứ vào phần mở rộng để xác định kiểu của file:

COM, EXE : Các file khả thi chạy trực tiếp được trên hệ điều hành.

TXT, DOC, ... : Các file văn bản.

PAS, BAS, ... : Các file chương trình PASCAL, DELPHI, BASIC, ...

WK1, XLS, ... : Các file chương trình bảng tính LOTUS, EXCEL ...

BMP, GIF, JPG, ... : Các file hình ảnh.

MP3, DAT, WMA, ... : Các file âm thanh, video.

Ký hiệu đại diện (Wildcard)

Để chỉ một nhóm các tập tin, ta có thể sử dụng hai ký hiệu đại diện:

+ Dấu ? dùng để đại diện cho một ký tự bất kỳ trong tên tập tin tại vị trí nó xuất hiện.
+ Dấu * dùng để đại diện cho một chuỗi ký tự bất kỳ trong tên tập tin từ vị trí nó xuất hiện.

Ví dụ:

Bai?.doc	Bai1.doc, Bai6.doc, Baiq.doc, ...
Bai*.doc	Bai.doc, Bai6.doc, Bai12.doc, BaiTap.doc, ...
BaiTap.*	BaiTap.doc, BaiTap.xls, BaiTap.ppt, BaiTap.pdf, ...

Lưu ý: nên đặt tên mang tính gợi nhớ.

c. Quản lý tệp của hệ điều hành

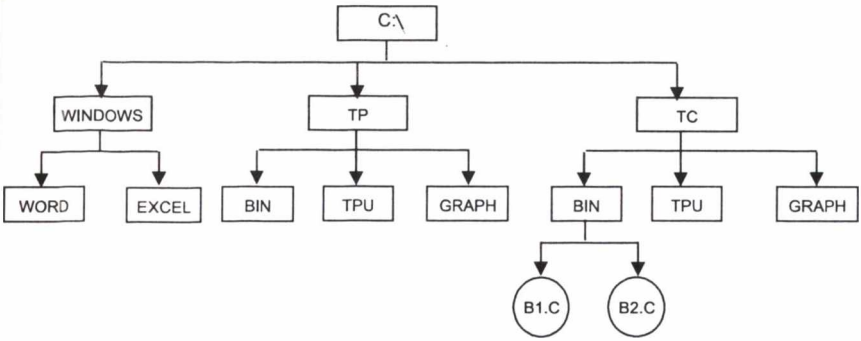
Cấu trúc đĩa từ

Hệ thống đĩa từ gồm nhiều mặt (side) gắn số hiệu là 0, 1,... Về mặt logic, mỗi mặt đĩa có một đầu ghi/đọc (header), đôi khi người ta còn đồng nhất hai khái niệm này. Mỗi mặt chia thành các rãnh (track – các đường tròn đồng tâm). Các rãnh được đánh số từ ngoài vào trong bắt đầu từ 0. Mỗi rãnh chia thành các cung (sector), mỗi cung thông thường có dung lượng 512 byte. Một từ trụ (cylinder) gồm các rãnh có cùng bán kính nằm trên các mặt đĩa khác nhau.

Tổ chức ghi thông tin trên đĩa

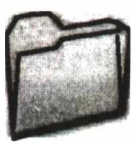
Thông tin lưu trữ trên đĩa dưới dạng các tệp. Mỗi tệp chiếm một hoặc nhiều sectors tuỳ dung lượng tệp.

Để thuận lợi cho việc quản lý tệp, hệ điều hành cho phép chia đĩa thành các vùng, mỗi vùng chia thành các vùng con,.... Mỗi vùng có một vùng con riêng để lưu trữ thông tin về vùng đó, vùng con này được gọi là thư mục (Directory). Tệp được lưu trữ ở các vùng, vì vậy ta có thể thấy tổ chức lưu trữ này có dạng cây (Tree). Ví dụ:



Hình I.10. Cây phân cấp thư mục

Thư mục là nơi lưu giữ các tập tin hoặc thư mục con theo một chủ đề nào đó theo ý người sử dụng. Đây là biện pháp giúp ta quản lý được tập tin, dễ dàng tìm kiếm chúng khi cần. Các tập tin có liên quan với nhau có thể được xếp trong cùng một thư mục. Sau đây là biểu tượng của thư mục hay còn gọi là Folder trong Windows:



Trên mỗi đĩa có một thư mục chung gọi là thư mục gốc. Thư mục gốc không có tên riêng và được ký hiệu là \ (dấu xổ phải: backslash). Dưới mỗi thư mục gốc có các tập tin trực thuộc và các thư mục con. Trong các thư mục con cũng có các tập tin trực thuộc và thư mục con của nó. Thư mục chứa thư mục con gọi là thư mục cha.

Thư mục đang làm việc gọi là thư mục hiện hành.

Tên của thư mục tuân thủ theo cách đặt tên của tập tin.

Cách xác định tên đầy đủ của tệp

Tên tệp đầy đủ gồm nơi lưu trữ tệp – đường dẫn từ gốc đến tệp (Path) và tên tệp Đường dẫn được chỉ ra nhánh cây thư mục chứa tệp, trong đó sử dụng ký hiệu "\' ngăn cách tên các thư mục .

Ví dụ :

C:\TC\BIN\B1.C

...

Hệ điều hành được phân chia thành các phần, phù hợp với các chức năng riêng của công việc. Những phần này được lưu trên đĩa dưới dạng các tệp (File). Ví dụ:

Hệ điều hành MS–DOS gồm tập các tệp, trong đó có ba tệp cơ bản:

+ MSDOS.SYS – tệp hệ thống quan trọng, chứa nhân hệ điều hành.
+ IO.SYS – tệp điều khiển vào ra.
+ COMMAND.COM – tệp lệnh.

I.2.3.2. Hệ lệnh của hệ điều hành

- Thao tác với tệp: sao chép, đi chuyển, xoá, đổi tên, xem nội dung tệp.
- Thao tác với thư mục: tạo, xoá, sao chép.
- Thao tác với đĩa: tạo khuôn (Format), sao chép đĩa.

I.2.3.3. Hệ điều hành Windows

a. Sự ra đời và phát triển

Windows là một bộ chương trình do hãng Microsoft sản xuất. Từ version 3.0 ra đời vào tháng 5 năm 1990 đến nay, Microsoft đã không ngừng cải tiến làm cho môi trường này ngày càng được hoàn thiện.

Windows 95: Vào cuối năm 1995, ở Việt Nam, đã xuất hiện một phiên bản mới của Windows mà chúng ta quen gọi là Windows 95. Những cải tiến mới của Windows 95 được tóm tắt như sau:

– Giao diện với người sử dụng được thiết kế lại hoàn toàn nên việc khởi động các chương trình ứng dụng cùng các công việc như mở và lưu cất các tư liệu, tổ chức các tài nguyên trên đĩa và nối kết với các hệ phục vụ trên mạng – tất cả đều trở nên đơn giản và dễ dàng hơn.

– Cho phép đặt tên cho các tập tin dài đến 255 ký tự. Điều này rất quan trọng vì những tên dài sẽ giúp gợi nhớ đến nội dung của tập tin.

– Hỗ trợ Plug and Play, cho phép tự động nhận diện các thiết bị ngoại vi nên việc cài đặt và quản lý trở nên đơn giản hơn.

– Hỗ trợ tốt hơn cho các ứng dụng Multimedia. Với sự tích hợp Audio và Video của Windows 95, máy tính cá nhân trở thành phương tiện giải trí không thể thiếu.

– Windows 95 là hệ điều hành 32 bit, vì vậy nó tăng cường sức mạnh và khả năng vận hành lên rất nhiều.

– Trong Windows 95 có các công cụ đã được cải tiến nhằm chuẩn hoá, tối ưu hoá và điều chỉnh các sự cố. Điều này giúp bạn yên tâm hơn khi làm việc với máy vi tính trong môi trường của Windows 95.

Tóm lại, với những tính năng mới ưu việt và tích hợp cao, Windows 95 đã trở thành môi trường làm việc được người sử dụng ưa chuộng và tin dùng.

Windows 98, Windows Me: Là những phiên bản tiếp theo của Windows 95. Những phiên bản này tiếp tục phát huy và hoàn thiện những tính năng ưu việt của Windows 95, đồng thời tích hợp thêm những tính năng mới về Internet và Multimedia.

Windows NT 4.0, Windows 2000, Windows XP, Windows 2003: Là những hệ điều hành được phát triển cao hơn, được dùng cho các cơ quan và doanh nghiệp. Giao diện của những hệ điều hành này tương tự như Windows 98/Windows Me. Điểm khác biệt là những hệ điều hành này có tính năng bảo mật cao, vì vậy nó được sử dụng cho môi trường có nhiều người dùng.

Windows VISTA: Ra đời vào năm 2007 nhằm phục vụ các máy tính cá nhân và máy tính xách tay, Windows Vista có giao diện đồ họa đẹp với hiệu ứng Flip 3D và giao diện Aero. Người dùng có thể theo dõi màn hình một ứng dụng bằng cách trỏ chuột vào biểu tượng trên thanh taskbar, người dùng cũng có thể đặt các gadget lên Sidebar như lịch, đồng hồ, ghi chú, khung ảnh. Microsoft đính kèm khá nhiều phần mềm tiện ích trên Windows Vista như phần mềm tạo phim, ghi đĩa, giải trí, sao lưu dữ liệu tự động, nhận dạng tiếng nói,...Windows Vista tiêu tốn nhiều tài nguyên máy và khó sử dụng hơn Windows XP. Hiện có các phiên bản Ultimate, Home Premium, Home Basic, Business và Enterprise.

Windows 7: Đầu tháng 1 năm 2009, bản Windows 7 chính thức được ra mắt. Sau đó là các bản RC (Release Candidate), RTM (Release To Manufacture). Hệ điều hành này có các tính năng độc đáo so với các hệ điều hành trước đây: dễ dàng quản lý các thiết bị, tập tin, media,..., tính năng Jump List trên Taskbar và Start Menu cung cấp sự truy cập nhanh chóng các thông tin vừa được các chương trình sử dụng, người dùng có thể đặt các gadget ở bất kỳ vị trí nào trên desktop,... Hiện có các phiên bản sau: Starter, Home Basic, Home Premium, Professional, Enterprise/Ultimate.

Windows 8: Tháng 9 năm 2011, Microsoft chính thức công bố Windows 8 với giao diện được hỗ trợ tối ưu cho máy tính bảng và hệ điều hành này còn hoạt động được trên mọi thiết bị khác nhau, bất kể kích cỡ. Hệ thống khởi động nhanh với hệ điều hành Windows 8, có giao diện cảm ứng đa chạm Metro, trình duyệt cảm ứng duyệt web nhanh,...

Giáo trình này sẽ trình bày dựa vào hệ điều hành Windows XP.

b. Khởi động và thoát khỏi Windows XP

Khởi động Windows XP

Windows XP được tự động khởi động sau khi bật máy. Sẽ có thông báo yêu cầu nhập vào tài khoản (User name) và mật khẩu (Password) của người dùng. Thao tác này gọi là đăng nhập (Log on).

Mỗi người sử dụng sẽ có một tập hợp thông tin về các lựa chọn tự thiết lập cho mình (như bố trí màn hình, các chương trình tự động chạy khi khởi động máy, tài nguyên/chương trình được phép sử dụng,...) gọi là user profile và được Windows XP lưu giữ lại để sử dụng cho những lần khởi động sau.

Thoát khỏi Windows XP

Khi muốn thoát khỏi Windows XP, bạn phải đóng tất cả các cửa sổ đang mở. Tiếp theo bạn nhấn tổ hợp phím Alt + F4 hoặc chọn menu Start (nếu không nhìn thấy nút Start ở phía dưới bên góc trái màn hình thì bạn nhấn tổ hợp phím Ctrl + Esc) và chọn Turn Off Computer. Sau thao tác này một hộp thoại sẽ xuất hiện.

Nếu bạn chọn Turn Off, ứng dụng đang làm việc sẽ được đóng lại và máy sẽ tự động tắt. Nếu vì một lý do nào đó mà máy tính không sẵn sàng để đóng (chưa lưu dữ liệu cho một ứng dụng hoặc sự trao đổi thông tin giữa hai máy nối mạng đang tiếp diễn...) thì sẽ có thông báo để xử lý.

Chú ý: Nếu không làm những thao tác đóng Windows như trên mà tắt máy ngay thì có thể sẽ xảy ra việc thất lạc một phần nội dung các tập tin, dẫn đến trục trặc khi khởi động lại ở lần sử dụng tiếp theo.

c. Một số thuật ngữ và thao tác thường sử dụng

Biểu tượng (icon)

Biểu tượng là các hình vẽ nhỏ đặc trưng cho một đối tượng nào đó của Windows hoặc của các ứng dụng chạy trong môi trường Windows. Phía dưới biểu tượng là

tên biểu tượng. Tên này mang một ý nghĩa nhất định, thông thường nó diễn giải cho chức năng được gán cho biểu tượng (ví dụ nó mang tên của một trình ứng dụng).

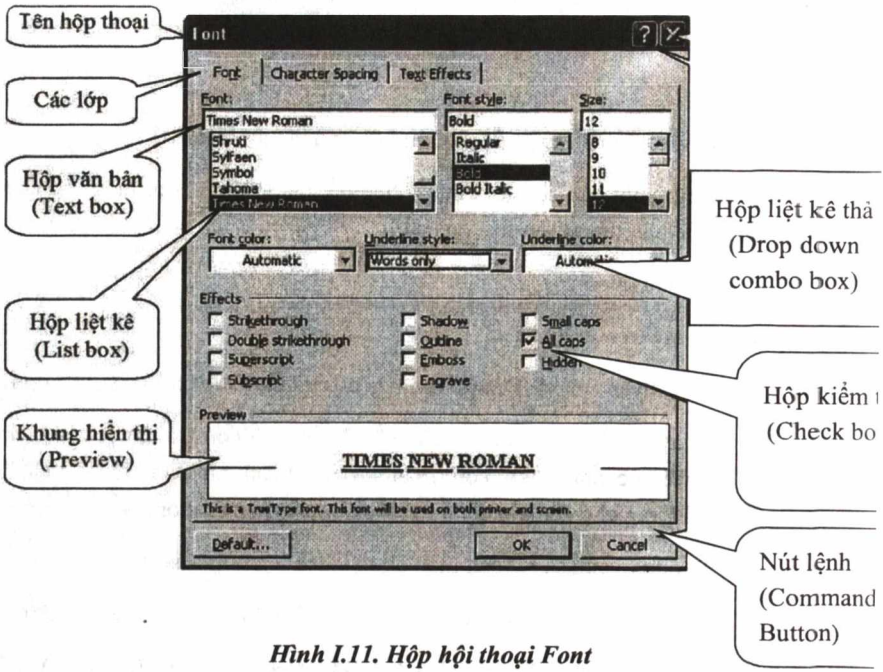
Cửa sổ (Windows)

Cửa sổ là khung giao tiếp đồ họa của một ứng dụng hoặc một lệnh.

- Bố cục của một cửa sổ: gồm thanh tiêu đề, thanh thực đơn, một số thành phần khác phụ thuộc vào loại cửa sổ,...
- Các hộp giao tiếp.
- Các thao tác trên một cửa sổ:
  - + Di chuyển cửa sổ: Drag thanh tiêu đề cửa sổ (Title bar) đến vị trí mới.
  - + Thay đổi kích thước của cửa sổ: Di chuyển con trỏ chuột đến cạnh hoặc góc cửa sổ, khi con trỏ chuột biến thành hình mũi tên hai chiều thì Drag cho đến khi đạt được kích thước mong muốn.
  - + Phóng to cửa sổ ra toàn màn hình: Click lên nút Maximize .
  - + Phục hồi kích thước trước đó của cửa sổ: Click lên nút Restore .
  - + Thu nhỏ cửa sổ thành biểu tượng trên Taskbar: Click lên nút Minimize  .
  - + Chuyển đổi giữa các cửa sổ của các ứng dụng đang mở: Để chuyển đổi giữa các ứng dụng nhấn tổ hợp phím Alt + Tab hoặc chọn ứng dụng tương ứng trên thanh Taskbar.
  - + Đóng cửa sổ: Click lên nút Close của cửa sổ hoặc nhấn tổ hợp phím Alt + F4.

Hộp hội thoại (Dialogue box)

Trong khi làm việc với Windows và các chương trình ứng dụng chạy dưới môi trường Windows, bạn thường gặp những hộp hội thoại. Các hộp thoại này xuất hiện khi nó cần thêm những thông số để thực hiện lệnh theo yêu cầu của bạn. Hình dưới đây giới thiệu các thành phần của hộp hội thoại.



Hình I.11. Hộp hội thoại Font

Thông thường, trên một hộp hội thoại sẽ có các thành phần sau:

- Hộp văn bản (Text box): Dùng để nhập thông tin.
- Hộp liệt kê (List box): Liệt kê sẵn một danh sách có các mục có thể chọn lựa, nếu số mục trong danh sách nhiều không thể liệt kê hết thì sẽ xuất hiện thanh trượt để cuộn danh sách.
- Hộp liệt kê thả (Drop down list box/ Combo box): Khi nhấp chuột vào nút thả thì sẽ buông xuống một danh sách.
- Hộp kiểm tra (Check box): Có hai dạng, dạng hình vuông thể hiện việc cho phép không chọn, chọn một hoặc nhiều mục không loại trừ lẫn nhau. Dạng ô tròn (Option button): bắt buộc phải chọn một trong số các mục. Đây là những lựa chọn loại trừ lẫn nhau.
- Nút lệnh (Command Button): Lệnh cần thực thi. Các loại nút lệnh thường gặp có:
  + **OK(hoặc bấm phím Enter)**: Thực hiện lệnh (chấp nhận).
  + **Close**: Giữ lại các thông số đã chọn và đóng cửa sổ.

+ **Cancel (hay nhấn phím Esc)**: Không thực hiện lệnh (từ chối thực hiện).
+ **Apply**: Áp dụng các thông số đã chọn.
+ **Default**: Đặt mặc định theo các thông số.

Sử dụng chuột trong Windows

Chuột là thiết bị không thể thiếu khi làm việc trong môi trường Windows XP. Con trỏ chuột (mouse pointer) cho biết vị trí tác động của chuột trên màn hình. Hình dáng của con trỏ chuột trên màn hình thay đổi theo chức năng và chế độ làm việc của ứng dụng. Khi làm việc với thiết bị chuột bạn thường sử dụng các thao tác cơ bản sau:

+ **Point:** Trỏ chuột trên mặt phẳng mà không nhấn nút nào cả.
+ **Click:** Nhấn nhanh và thả nút chuột trái, dùng để lựa chọn thông số, đối tượng hoặc câu lệnh.
+ **Double Click (D_Click):** Nhấn nhanh nút chuột trái hai lần liên tiếp, dùng để khởi động một chương trình ứng dụng hoặc mở thư mục/tập tin.
+ **Drag** (kéo thả): Nhấn và giữ nút chuột trái khi di chuyển đến nơi khác và buông ra, dùng để chọn một khối văn bản, để di chuyển một đối tượng trên màn hình hoặc mở rộng kích thước của cửa sổ...
+ **Right Click (R_Click):** Nhấn nhanh và thả nút chuột phải, dùng để mở menu tương ứng với đối tượng để chọn các lệnh thao tác trên đối tượng đó.

Chú ý:

- Đa số chuột hiện nay có bánh xe trượt hoặc nút đẩy ở giữa dùng để cuộn màn hình giúp làm việc nhanh hơn và thuận tiện hơn.
- Trong Windows các thao tác được thực hiện mặc nhiên với nút chuột trái, vì vậy để tránh lặp lại, khi nói **Click** (nhấn chuột) hoặc **D_Click** (nhấn đúp chuột) thì được ngầm hiểu đó là nút chuột trái. Khi cần thao tác với nút chuột phải thì sẽ mô tả rõ ràng.

d. Cấu hình Windows (Control Panel)

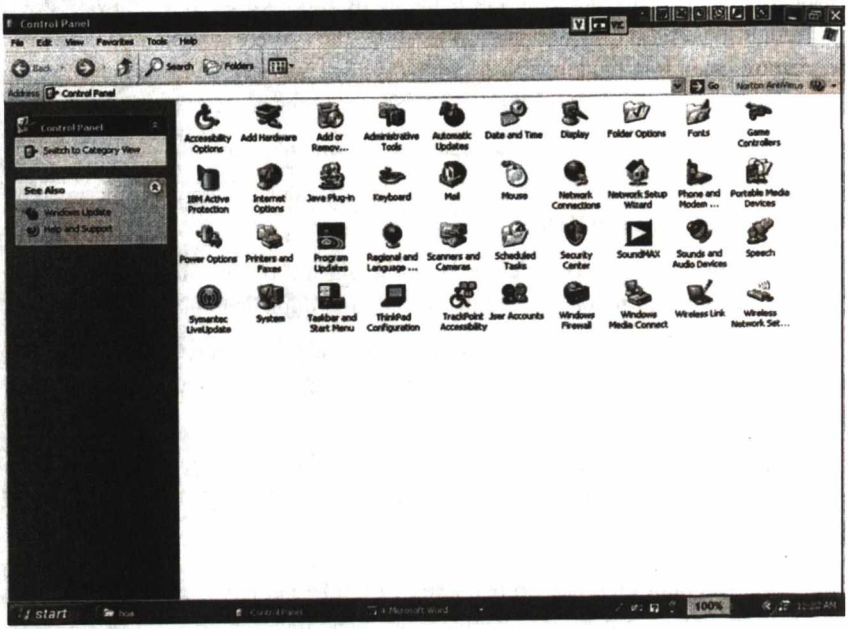
Giới thiệu về Control Panel

Control Panel là một chương trình cho phép người sử dụng xem và chỉnh sửa các tham số của hệ thống máy tính như dạng hiện của dữ liệu ngày tháng, dữ liệu số, thiết lập hoặc thay đổi cấu hình cho phù hợp với công việc hoặc sở thích của người dùng, cài đặt phần cứng, phần mềm.

i. Khởi động chương trình Control Panel

Chọn lệnh Start / Settings / Control Panel.

ii. Cửa sổ làm việc của Control Panel



Hình I.12. Cửa sổ Control Panel

iii. Cài đặt và loại bỏ Font chữ

Để cài đặt thêm những Font chữ khác hoặc loại bỏ các Font chữ, ta chọn chương trình **Fonts.**

- **Loại bỏ font chữ:** Từ cửa sổ Fonts:
  - Chọn những Font cần loại bỏ
  - Chọn File/Delete (hoặc nhấn phím Delete).
- **Thêm font chữ mới:** Từ cửa sổ Fonts, chọn lệnh File/Install New Font, xuất hiện hộp thoại Add Fonts. Trong hộp thoại này **chỉ ra nơi chứa các**

Font nguồn muốn thêm bằng cách chọn tên ổ đĩa chứa các tập tin Font chữ, sau đó chọn các tên Font và Click OK.

iv. Thay đổi dạng hiện màn hình desktop

Chọn lệnh **Start/Settings/Control Panel/Display** hoặc **R_Click trên màn hình** nền (Desktop), chọn Properties. Xuất hiện cửa sổ Display Properties với các thành phần như sau:

Desktop: Chọn ảnh nền cho Desktop bằng cách Click chọn các ảnh nền có sẵn hoặc Click vào nút Browse để chọn tập tin ảnh không có trong danh sách những ảnh có sẵn.

Screen Saver: Xác lập màn hình nghỉ.

Settings: Thay đổi chế độ màu và độ phân giải của màn hình. Chế độ màu càng cao thì hình ảnh càng đẹp và rõ nét. Các chế độ màu: 64.000 màu (16 bit) , 16 triệu màu (24 bit). Chế độ màu trên mỗi máy tính có thể khác nhau tùy thuộc vào dung lượng bộ nhớ của card màn hình. Độ phân giải càng lớn thì màn hình càng hiển thị được nhiều thông tin.

v. Cài đặt và loại bỏ chương trình

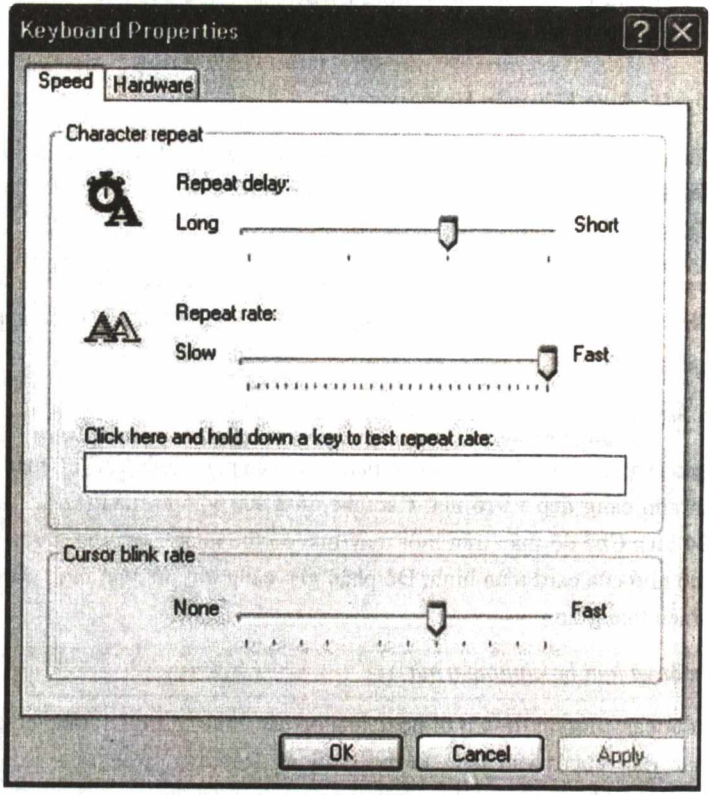
Để cài đặt các chương trình mới hoặc loại bỏ các chương trình không còn sử dụng bạn nhấn đúp chuột vào biểu tượng **Add or Remove Programs** trong cửa sổ Control Panel, khi đó sẽ xuất hiện hộp thoại và thao tác theo chỉ dẫn.

vi. Cấu hình ngày, giờ cho hệ thống

Bạn có thể thay đổi ngày giờ của hệ thống bằng cách D_Click lên biểu tượng đồng hồ trên thanh Taskbar hoặc vào Control Panel, chọn nhóm **Date/Time – Date & Time**: thay đổi ngày, tháng, năm, giờ, phút, giây.

vii. Thay đổi thuộc tính của bàn phím và chuột

Lệnh Start/Settings/Control Panel, chọn biểu tượng Mouse.



Hình I.13. Hộp thoại Keyboard Properties

Thay đổi thuộc tính của bàn phím:

Repeat delay: Thay đổi thời gian trễ cho phím.

Repeat rate: Thay đổi tốc độ lặp lại khi nhấn phím.

Thay đổi thuộc tính của thiết bị chuột

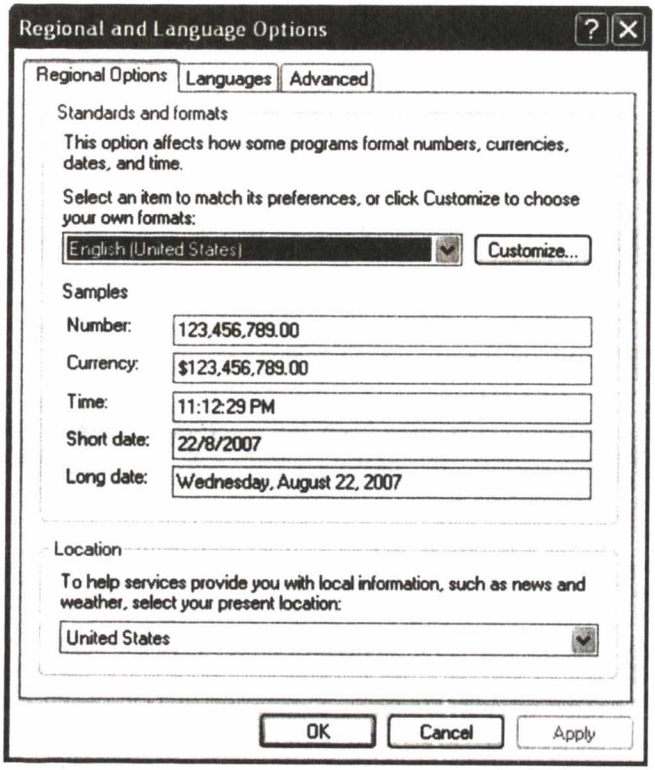
Lệnh Start/Settings/Control Panel, chọn biểu tượng Keyboard.

Lớp Buttons: Thay đổi phím trái và phím chuột phải (thuận tay trái hay phải) và tốc độ nhấp đúp chuột.

Lớp Pointers: Cho phép chọn hình dạng trỏ chuột trong các trạng thái làm việc.

viii. Thay đổi thuộc tính vùng (Regional Settings)

Bạn có thể thay đổi các thuộc tính như định dạng tiền tệ, đơn vị.



Hình I.14. Hộp thoại Regional and Language Options

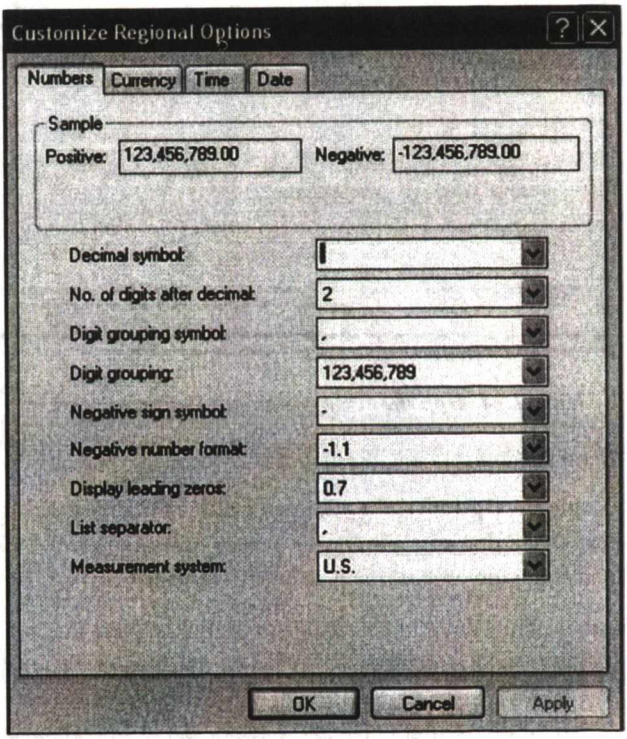
Lệnh Start/Settings/Control Panel/Regional and Language Options.

Lớp Regional Options: Thay đổi thuộc tính vùng địa lý, sau đó sẽ kéo theo sự thay đổi các thuộc tính của Windows. Click chọn **Customize**, cửa sổ Customize xuất hiện để thay đổi quy ước về dạng số, tiền tệ, thời gian, ngày tháng.

- **Number:** Thay đổi định dạng số, cho phép định dạng việc hiển thị.
  - + Decimal symbol: Thay đổi ký hiệu phân cách hàng thập phân.
  - + No. of digits after decimal: Thay đổi số các số lẻ ở phần thập phân.
  - + Digit grouping symbol: Thay đổi ký hiệu phân nhóm hàng ngàn.

+ Digit grouping: Thay đổi số ký số trong một nhóm (3 số / 4 số/ ...).
+ Negative sign symbol: Thay đổi ký hiệu của số âm.
+ Negative number format: Thay đổi dạng thể hiện của số âm.
+ Display leading zeros: **0.7** hay **.7**.
+ Measurement system: Chọn hệ thống đo lường như cm, inch, ...
+ List separator: Chọn dấu phân cách giữa các mục trong danh sách.

- **Currency:** Thay đổi định dạng tiền tệ ($,VND,...).
- **Time:** Thay đổi định dạng giờ theo chế độ 12 giờ hay 24 giờ.
- **Date:** Thay đổi định dạng ngày tháng (Date), cho phép chọn cách thể hiện ngày.



Hình I.15. Hộp thoại Customize Regional Options

ix. Cài đặt / loại bỏ máy in

Cài đặt thêm máy in

Với một số máy in thông dụng, Windows đã tích hợp sẵn chương trình điều khiển (driver) của các máy in, tuy nhiên cũng có những máy in mà trong Windows chưa có chương trình điều khiển. Muốn sử dụng những máy in này, ta cần phải thực hiện chương trình **Printers and Faxes** trong Control Panel.

Các bước cài đặt máy in:

- Chọn lệnh Start/Settings/Printers and Faxes.
- Click chọn Add a Printer, xuất hiện hộp thoại Add.
- Làm theo các bước hướng dẫn của hệ thống.

Loại bỏ máy in đã cài đặt

- Chọn lệnh Start/Settings/Printers and Faxes.
- Click chuột chọn máy in muốn loại bỏ.
- Nhấn phím Delete, sau đó chọn Yes.

e. Windows Explorer

Giới thiệu Windows Explorer

Windows Explorer là một chương trình được hỗ trợ từ phiên bản Windows 95 cho phép người sử dụng thao tác với các tài nguyên có trong máy tính như tập tin, thư mục, ổ đĩa và những tài nguyên khác có trong máy của bạn cũng như các máy tính trong hệ thống mạng (nếu máy tính của bạn có nối mạng).

Với Windows Explorer, các thao tác như sao chép, xóa, đổi tên thư mục và tập tin,... được thực hiện một cách thuận tiện và dễ dàng.

♦ **Khởi động chương trình Windows Explorer**: Bạn có thể thực hiện một trong những cách sau:

- Chọn lệnh Start/Programs/Accessories/Windows Explorer.
- R_Click lên Start, sau đó chọn Explorer.
- R_Click lên biểu tượng My Computer, sau đó chọn Explorer …

♦ **Cửa sổ làm việc** của Windows Explorer

– **Cửa sổ trái** (Folder) là cấu trúc cây thư mục. Nó trình bày cấu trúc thư mục của các đĩa cứng và các tài nguyên kèm theo máy tính, bao gồm ổ đĩa mềm, ổ đĩa cứng, ổ đĩa CD...

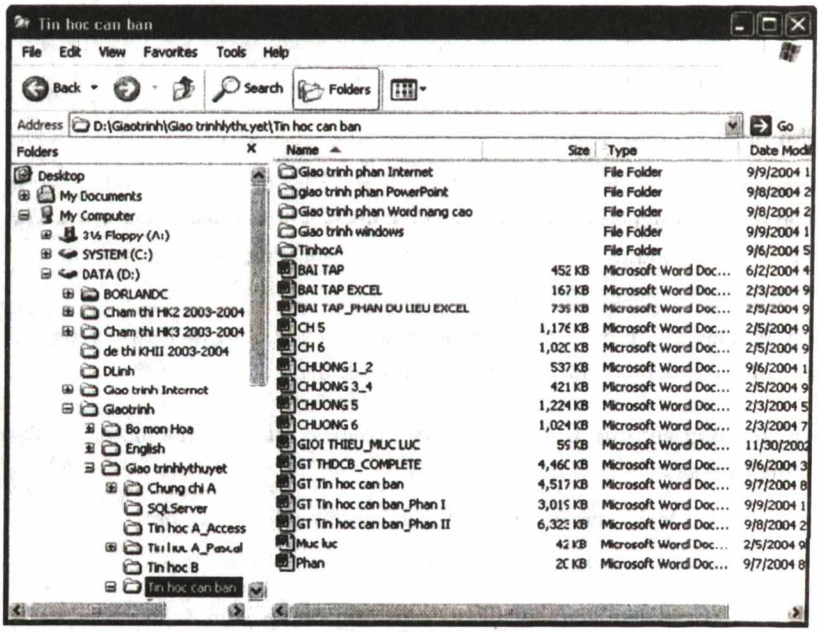
Những đối tượng có dấu cộng (+) ở phía trước cho biết đối tượng đó còn chứa những đối tượng khác trong nó nhưng không được hiển thị. Nếu Click vào dấu + thì Windows Explorer sẽ hiển thị các đối tượng chứa trong đối tượng đó. Khi đó, dấu + sẽ đổi thành dấu – và nếu Click vào dấu – thì đối tượng sẽ được thu gọn trở lại.

– **Cửa sổ phải** liệt kê nội dung của đối tượng được chọn tương ứng bên cửa sổ trái.

♦ **Thanh địa chỉ (Address bar)**

Cho phép nhập đường dẫn thư mục/tập tin cần tới hoặc để xác định đường dẫn hiện hành.

♦ **Các nút công cụ trên thanh Toolbar**



Hình I.16. Cửa sổ Windows Explorer

Thư mục (Folder)

	– Back: Chuyển về thư mục trước đó.
	– Up: Chuyển lên thư mục cha.
	– Forward: Chuyển tới thư mục vừa quay về (Back).
	– Search: Tìm kiếm tập tin/ thư mục.
	– Folder: Cho phép ẩn/ hiện cửa sổ Folder bên trái.
	– Views: Các chế độ hiển thị các đối tượng (tập tin/thư mục/ổ đĩa)

Nội dung trong cửa sổ có thể được sắp xếp thể hiện theo thứ tự. Đối với kiểu th hiện Details, bạn có thể thực hiện bằng cách luân phiên nhấn chuột lên cột tươn ứng (Name, Size, Type, Date Modified).

Trong các kiểu thể hiện khác bạn có thể thực hiện lệnh **View/Arrange Icons By** v lựa chọn tiếp một trong các khóa sắp xếp (theo tên, kích cỡ tập tin, kiểu tập tir hoặc ngày tháng cập nhật).

Trong kiểu thể hiện bằng các biểu tượng lớn và biểu tượng nhỏ, bạn có thể đ Windows sắp xếp tự động bằng lệnh **View/Arrange Icons By/Auto Arrange**. Tu chọn Auto Arrange chỉ áp dụng cho cửa sổ của thư mục hiện hành.

Thao tác với thư mục và tệp

i. Mở tập tin/thư mục

Có ba cách thực hiện:

- Cách 1: D_Click lên biểu tượng của tập tin/thư mục.
- Cách 2: R_Click lên biểu tượng của tập tin/thư mục và chọn mục Open.
- Cách 3: Chọn tập tin/thư mục và nhấn phím Enter.

Nếu tập tin thuộc loại tập tin văn bản thì chương trình ứng dụng kết hợp sẽ được khởi động và tài liệu sẽ được nạp vào.

Trong trường hợp chương trình ứng dụng không được cài đặt trong máy tính thì Windows sẽ mở hộp thoại Open With và cho chọn chương trình kết hợp. Nếu tập tin thuộc dạng chương trình ứng dụng thì chương trình tương ứng sẽ được khởi động.

ii. Chọn tập tin/thư mục

- Chọn một tập tin/ thư mục: Click lên biểu tượng tập tin/thư mục.
- Chọn một nhóm tập tin/ thư mục: Có thể thực hiện theo hai cách:
  - Các đối tượng cần chọn là một danh sách gồm các đối tượng liên tục: Click lên đối tượng đầu danh sách để chọn, sau đó nhấn giữ phím Shift và Click lên đối tượng ở cuối danh sách.
  - Các đối tượng cần chọn nằm rời rạc nhau: Nhấn giữ phím Ctrl và Click chọn các đối tượng tương ứng.

iii. Tạo thư mục

- Chọn nơi chứa thư mục cần tạo (thư mục/ổ đĩa ở cửa sổ bên trái).
- Chọn menu **File/New/Folder** hoặc **R_Click/New/ Folder**.
- Nhập tên thư mục mới, sau đó gõ Enter để kết thúc.

iv. Sao chép thư mục và tập tin

Chọn các thư mục và tập tin cần sao chép. Sau đó có thể thực hiện theo một trong hai cách sau:

- Cách 1: Nhấn giữ phím Ctrl và Drag đối tượng đã chọn đến nơi cần chép.
- Cách 2: Nhấn tổ hợp phím Ctrl + C (hoặc Edit/Copy hoặc R_Click và chọn Copy) để chép vào Clipboard, sau đó chọn nơi cần chép đến và nhấn tổ hợp phím Ctrl + V (hoặc Edit/Paste hoặc R_Click và chọn Paste).

v. Di chuyển thư mục và tập tin

Chọn các thư mục và tập tin cần di chuyển. Sau đó có thể thực hiện theo một trong hai cách sau:

- Cách 1: Drag đối tượng đã chọn đến nơi cần di chuyển.
- Cách 2: Nhấn tổ hợp phím Ctrl + X (hoặc Edit/Cut hoặc R_Click và chọn Cut) để chép vào Clipboard, sau đó chọn nơi cần di chuyển đến và nhấn tổ hợp phím Ctrl + V (hoặc Edit/Paste hoặc R_Click và chọn Paste).

vi. Xoá thư mục và tập tin

- Chọn các thư mục và tập tin cần xóa.
- Chọn File/Delete hoặc nhấn phím Delete hoặc R_Click và chọn mục Delete.
- Xác nhận có thực sự muốn xoá hay không (Yes/No).

vii. Phục hồi thư mục và tập tin

Các đối tượng bị xóa sẽ được đưa vào Recycle Bin. Nếu muốn phục hồi các đối tượng đã xóa, bạn thực hiện các thao tác sau đây:

- D_Click lên biểu tượng Recycle Bin.
- Chọn tên đối tượng cần phục hồi.
- Thực hiện lệnh **File/Restore** hoặc R_Click và chọn mục **Restore**.

Ghi chú: Nếu muốn xóa hẳn các đối tượng, ta thực hiện thao tác xóa một lần nữa đối với các đối tượng ở trong Recycle Bin. Nếu muốn xoá hẳn tất cả các đối tượng trong Recycle Bin, R_Click lên mục Recycle Bin và chọn mục Empty Recycle Bin.

viii. Đổi tên thư mục và tập tin

- Chọn đối tượng muốn đổi tên
- Thực hiện lệnh **File/Rename** hoặc nhấn phím F2 hoặc R_Click trên đối tượng và chọn mục Rename.
- Nhập tên mới, sau đó gõ Enter để kết thúc.

Ghi chú: Với tập tin đang sử dụng thì các thao tác di chuyển, xoá, đổi tên không thể thực hiện được.

ix. Thay đổi thuộc tính tập tin và thư mục

- Nhấn chuột phải lên đối tượng muốn thay đổi thuộc tính và chọn mục **Properties**.
- Thay đổi các thuộc tính.
- Chọn **Apply** để xác nhận thay đổi, ngược lại thì nhấn **Cancel**.

I.3. Các hệ thống ứng dụng

I.3.1. Hệ thống thông tin quản lý

I.3.1.1. Khái niệm

Nhu cầu quản lý thông tin trong các tổ chức đã có từ rất lâu. Thông tin của tổ chức có thể là thông tin về nhân sự, người lao động, kế hoạch sản xuất, kế hoạch tài chính, bảng lương,... Khi máy tính chưa xuất hiện hoặc chưa phổ biến thì người ta quản lý thông tin của tổ chức một cách thủ công dựa trên sổ sách, giấy tờ và con người trực tiếp thao tác với hệ thống sổ sách, giấy tờ này. Cách quản lý thủ công có thể gây ra những sai sót, nhầm lẫn không mong muốn. Hơn nữa, việc lưu trữ hệ thống sổ sách này đòi hỏi khá nhiều diện tích, không gian cũng như quy trình sắp xếp và bảo quản.

Kể từ khi máy tính xuất hiện và trở nên phổ biến, các tổ chức đã tận dụng và phát huy được khả năng của công cụ mới này. *Hệ thống thông tin quản lý* là hệ thống bao gồm phần cứng, phần mềm, con người, quy trình thu thập, phân tích, xử lý, đánh giá và phân phối, chia sẻ những thông tin cần thiết một cách kịp thời và chính xác dựa trên nhu cầu của tổ chức.

Lưu ý, một hệ thống thông tin không nhất thiết phải cần đến máy tính, đó là những hệ thống thông tin thủ công sử dụng giấy, bút. Ở đây, giáo trình sử dụng cụm từ hệ thống thông tin để chỉ hệ thống thông tin có sự trợ giúp của máy tính (công nghệ phần cứng và phần mềm).

Một hệ thống thông tin quản lý được hình thành với *năm thành phần cơ bản*: (1) cơ sở hạ tầng (phần cứng và hệ thống truyền thông), (2) phần mềm, (3) cơ sở dữ liệu, (4) quy trình và (5) nhân sự. Năm thành phần cơ bản này xuất hiện với tất cả các dạng hệ thống thông tin bao gồm cả hệ thống thông tin đơn giản nhất tới hệ thống thông tin phức tạp nhất.

I.3.1.2. Các chức năng của hệ thống thông tin quản lý

Về chức năng, hệ thống thông tin quản lý thường có những chức năng chủ yếu sau:

Nhập dữ liệu: Hoạt động thu thập và nhận dữ liệu từ bên trong hoặc bên ngoài tổ chức để xử lý.

Xử lý thông tin: Quá trình chuyển đổi từ những dữ liệu hỗn hợp thành dạng có ý nghĩa đối với người sử dụng.

Xuất dữ liệu: Sự phân phối các thông tin đã được xử lý tới những người hoặc những hoạt động cần thông tin đó.

Lưu trữ thông tin: Các thông tin không chỉ được xử lý để sử dụng ngay tại thời điểm tổ chức thu nhận nó mà còn có thể được xử lý và phân tích trong tương lai. Vì vậy, việc lưu trữ thông tin cũng là một trong những hoạt động quan trọng của hệ thống thông tin. Thông tin của các tổ chức, doanh nghiệp thường được lưu trữ dưới dạng các trường, các tệp, các báo cáo và trong các hệ quản trị cơ sở dữ liệu.

Thông tin phản hồi: Hệ thống thông tin thường được điều khiển thông qua các thông tin phản hồi, giúp cho những người điều hành mạng lưới thông tin có thể đánh giá lại và hoàn thiện quá trình thu thập và xử lý dữ liệu mà họ đang thực hiện.

I.3.1.3. Các dạng và các đặc tính của thông tin trong tổ chức

Thông tin trong các tổ chức, doanh nghiệp được sử dụng với nhiều mục đích khác nhau và được thu thập từ hai nguồn chủ yếu: nguồn thông tin bên ngoài và nguồn thông tin bên trong. *Các dạng thông tin chủ yếu gồm có:*

Thông tin theo quan điểm cá nhân: Bất cứ một nhân viên nào của tổ chức, doanh nghiệp cũng đều làm việc với các thông tin và tạo ra những thông tin mới cho tổ chức đó. Khi thực hiện công việc này, người nhân viên xem xét những thông tin đó theo ba khía cạnh: thời gian, địa điểm và dạng thức của thông tin. Nói đến khía cạnh thời gian, chúng ta đang đề cập đến mức độ cập nhật của thông tin và thời điểm xem xét thông tin. Khía cạnh địa điểm của thông tin liên quan tới việc cho phép truy cập thông tin ở các địa điểm khác nhau như ở nhà, ở cơ quan hay trên máy bay. Khía cạnh dạng thức của thông tin liên quan tới (1) việc sử dụng thông tin ở những dạng đơn giản, dễ hiểu và (2) tính chính xác của thông tin.

Thông tin theo quan điểm tổ chức: Khi xem xét các dạng thông tin theo quan điểm tổ chức, cần chú ý tới (1) dòng thông tin và (2) mức độ chi tiết của thông tin. Dòng thông tin có thể được truyền đi theo bốn hướng: từ trên xuống, từ dưới lên, thông tin ngang cấp và thông tin ra bên ngoài. Thông tin được sử dụng cho việc ra

quyết định ở các cấp lãnh đạo là những thông tin tổng hợp, còn thông tin phục vụ cho các hoạt động tác nghiệp thường là những thông tin rất chi tiết và cụ thể.

Chất lượng của thông tin được xác định thông qua *các đặc tính sau*:

Chính xác: Thông tin có độ chính xác thấp sẽ gây cho doanh nghiệp những hậu quả tồi tệ.

Đầy đủ: Thể hiện sự bao quát các vấn đề đáp ứng yêu cầu của nhà quản lý. Nếu nhà quản lý sử dụng thông tin không đầy đủ có thể dẫn đến những quyết định sai lầm.

Thống nhất: Thông tin tổng hợp và thông tin chi tiết cần có sự thống nhất.

Thích hợp và dễ hiểu: Một số nhà quản lý có thể không sử dụng một số báo cáo do chúng chưa thích hợp và khó hiểu (có quá nhiều thông tin không thích ứng cho người nhận, thiếu rõ ràng, sử dụng nhiều từ viết tắt, đa nghĩa, bố trí chưa hợp lý).

Kịp thời: Thông tin cần được gửi tới cho người sử dụng vào đúng lúc cần thiết.

I.3.1.4. Phương pháp xây dựng và phát triển hệ thống thông tin

Có một số phương pháp xây dựng và phát triển một hệ thống thông tin quản lý. Trong giáo trình sẽ giới thiệu phương pháp chu kỳ hệ thống SDLC (Systems Development Life Cycle). Phần lớn các hệ thống thông tin sử dụng sự trợ giúp của máy tính đều được hình dung, thiết kế và thực hiện nhờ một số quá trình phát triển có tính tuần tự. Các giai đoạn chính của SDLC như sau:

Bước 1 – **Lập kế hoạch**: Thiết lập một kế hoạch đầy đủ cho việc phát triển hệ thống thông tin. Mục tiêu của bước này là xác định (1) dạng hệ thống thông tin sẽ được phát triển, (2) phạm vi làm việc của hệ thống, (3) các nhiệm vụ cụ thể, các nguồn lực và khung thời gian dành cho việc phát triển hệ thống.

Bước 2 – **Phân tích**: Các chuyên gia và người sử dụng hệ thống tham gia vào bước này, cùng nhau thu thập thông tin, tìm hiểu về hệ thống, nhiệm vụ của hệ thống và lập tài liệu xác định các yêu cầu.

Bước 3 – **Thiết kế**: Thiết kế kỹ thuật, xây dựng mô hình hệ thống. Về thiết kế, có thiết kế tổng thể cho hệ thống và thiết kế chi tiết cho từng thành phần của hệ thống. Kết thúc bước này đánh dấu bằng việc có tài liệu thiết kế tổng thể và chi tiết.

Bước 4 – **Cài đặt**: Thiết lập cơ sở dữ liệu, cơ sở hạ tầng công nghệ và xây dựng theo những gì đã được thiết kế tại bước trên.

Bước 5 – **Kiểm định**: Kiểm định hệ thống một cách đầy đủ để đảm bảo rằng h thống được phát triển đã đáp ứng các mục tiêu và yêu cầu đề ra.

Bước 6 – **Vận hành**: Đưa hệ thống vào vận hành trong thực tế theo các chiến lược (1) song song (vừa sử dụng hệ thống cũ, vừa sử dụng hệ thống mới trong một thời gian cho đến khi hệ thống mới làm việc tốt thì mới chấm dứt sử dụng hệ thống cũ), (2) thí điểm (đưa hệ thống mới vào sử dụng tại một bộ phận của tổ chức để kiểm định các vấn đề nảy sinh trước khi triển khai trong toàn bộ tổ chức), (3) giai đoạn (từng chức năng của hệ thống sẽ được triển khai sử dụng từng bước một, chiến lược này phù hợp với hệ thống thông tin lớn) và (4) thay thế (loại bỏ hoàn toàn hệ thống cũ khi đưa hệ thống mới vào sử dụng, chiến lược này có độ rủi ro khá cao).

Bước 7 – **Bảo trì**: Thường xuyên kiểm tra, giám sát, hỗ trợ những thay đổi cần thiết trong thời gian vận hành hệ thống.

Các tổ chức, doanh nghiệp đã từng bước tiến hành quá trình tin học hóa. Thông tin dữ liệu dần được số hóa và được xử lý bởi các phần mềm hệ thống thông tin quản lý. Máy tính và phần mềm đã giải phóng con người, làm tăng năng suất lao động, giảm các chi phí cho việc lưu trữ và nhân công, tạo hiệu quả kinh tế cho tổ chức.

I.3.2. Hệ soạn thảo văn bản

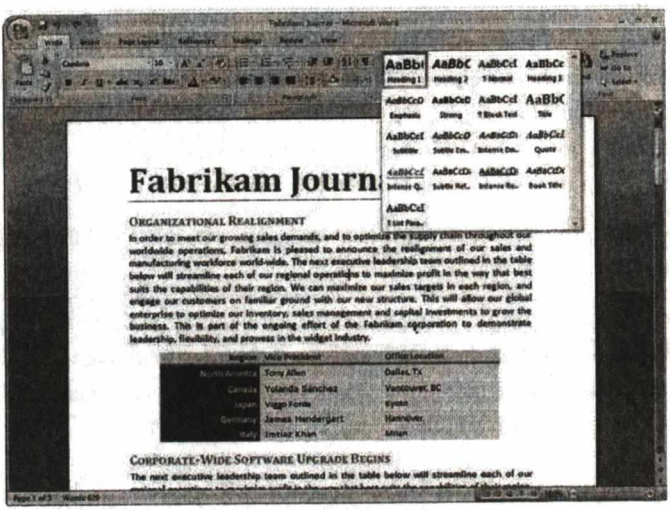
I.3.2.1. Khái niệm

Soạn thảo văn bản là một trong những công việc mà người làm công tác văn phòng, nhân viên các bộ phận tham mưu giúp việc trong các cơ quan, đơn vị thường phải thực hiện. Học sinh, sinh viên ngày nay cũng phải soạn thảo các báo cáo, tiểu luận, khóa luận,... nhằm phục vụ việc học tập của mình.

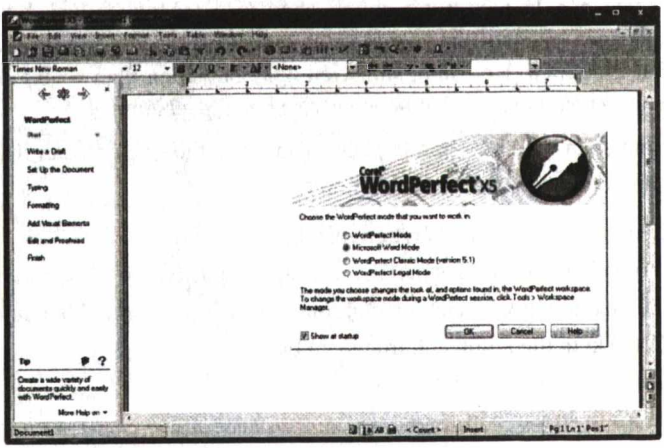
Chương trình soạn thảo văn bản là phần mềm được thiết kế để soạn thảo các văn bản điện tử với một số chức năng chính như:

- Hiển thị nội dung văn bản trên màn hình.
- Cho phép người dùng sửa đổi, bổ sung tại vị trí bất kỳ trong văn bản.
- Thể hiện nhiều kiểu chữ, cỡ chữ, màu sắc, các công thức khoa học.
- Có thể kèm theo hình ảnh trong văn bản.
- Lưu giữ văn bản dưới dạng file.
- Hỗ trợ in ấn văn bản.
- Có chức năng ghép ảnh.
- Có thể làm mục lục tự động, trộn thư,...

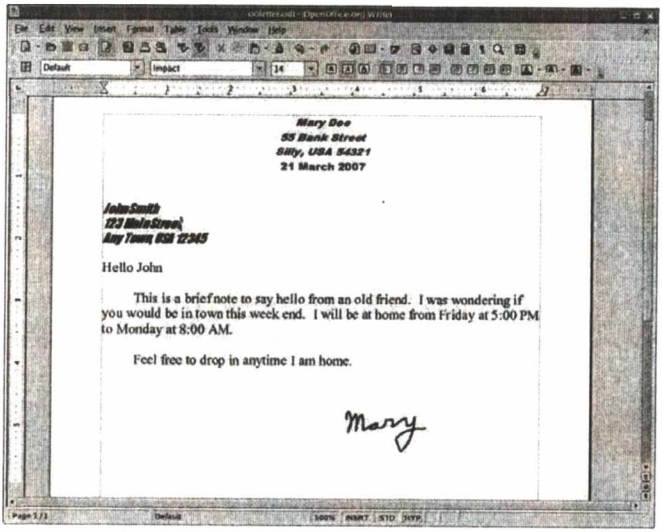
Có rất nhiều chương trình soạn thảo văn bản khác nhau. Các phần mềm thương mại nổi tiếng nhất có Microsoft Word của Microsoft, Word Perfect của Corel. Các chương trình soạn thảo văn bản thuộc loại mã nguồn mở thường gặp bao gồm: Writer trong bộ Open Office, KWord trong môi trường KDE, AbiWord trong môi trường GNOME.



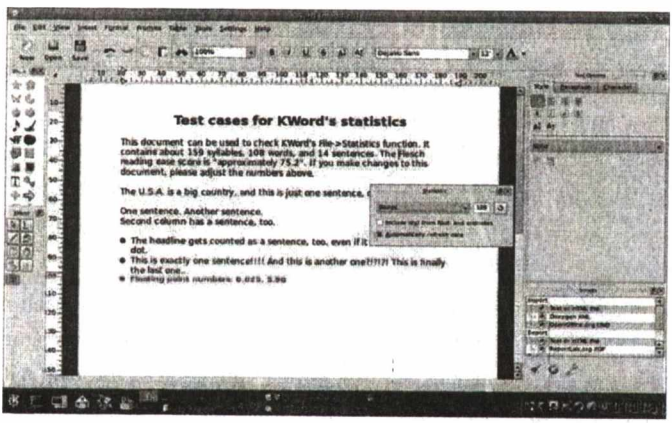
Hình I.17. Hệ soạn thảo văn bản MS Word của Microsoft



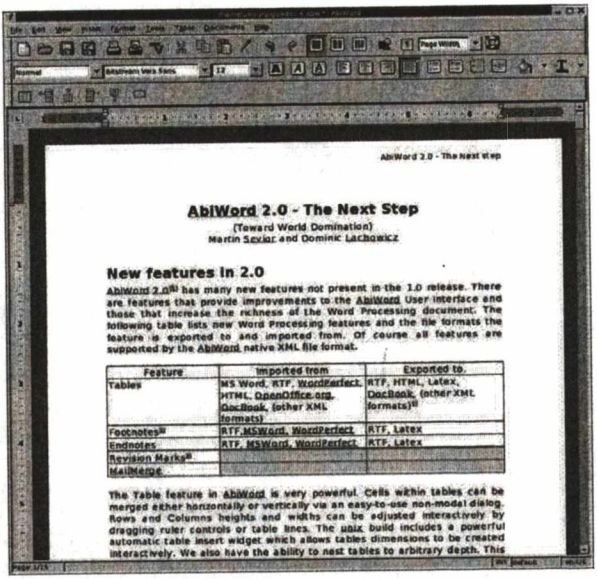
Hình I.18. Hệ soạn thảo văn bản WordPerfect của Corel



Hình I.19. Hệ soạn thảo văn bản Writer trong bộ OpenOffice



Hình I.20. Hệ soạn thảo KWord trong KDE



Hình I.21. Hệ soạn thảo AbiWord trong Gnome

Ký tự, từ, câu, dòng, đoạn:

Khi làm việc với văn bản, đối tượng chủ yếu ta thường xuyên phải tiếp xúc là các **ký tự** (Character). Các ký tự phần lớn được gõ vào trực tiếp từ bàn phím. Nhiều ký tự khác **ký tự trắng** (Space) ghép lại với nhau thành một **từ** (Word). Tập hợp các từ kết thúc bằng dấu ngắt câu, ví dụ dấu chấm ".", gọi là **câu** (Sentence). Nhiều câu có liên quan với nhau về mặt ngữ nghĩa nào đó tạo thành một **đoạn văn bản** (Paragraph).

Trong các phần mềm soạn thảo, đoạn văn bản được kết thúc bằng cách nhấn phím Enter. Như vậy, phím Enter được dùng khi cần tạo ra một đoạn văn bản mới. Đoạn là thành phần rất quan trọng của văn bản. Nhiều định dạng sẽ được áp đặt cho đoạn như căn lề, kiểu dáng,... Nếu trong một đoạn văn bản, ta cần ngắt xuống dòng, lúc đó dùng tổ hợp Shift+Enter. Thông thường, giãn cách giữa các đoạn văn bản sẽ lớn hơn giữa các dòng trong một đoạn.

Đoạn văn bản hiển thị trên màn hình sẽ được chia thành nhiều dòng tùy thuộc vào kích thước trang giấy in, kích thước chữ,... Có thể tạm định nghĩa dòng là một tập hợp các ký tự nằm trên cùng một đường cơ sở (Baseline) từ bên trái sang bên phải màn hình soạn thảo.

Nguyên tắc tự xuống dòng của từ:

Trong quá trình soạn thảo văn bản, khi gõ đến cuối dòng, phần mềm sẽ thực hiện động tác tự xuống dòng. Nguyên tắc của việc tự động xuống dòng là không được làm ngắt đôi một từ. Do vậy, nếu không đủ chỗ để hiển thị cả từ trên hàng, máy tính sẽ ngắt cả từ đó xuống hàng tiếp theo. Vị trí của từ bị ngắt dòng vì thế phụ thuộc vào rất nhiều yếu tố khác nhau như độ rộng trang giấy in, độ rộng cửa sổ màn hình, kích thước chữ. Do đó, nếu không có lý do để ngắt dòng, ta cứ tiếp tục gõ dù con trỏ đã nằm cuối dòng. Việc quyết định ngắt dòng tại đâu sẽ do máy tính lựa chọn.

Cách ngắt dòng tự động của phần mềm hoàn toàn khác với việc sử dụng các phím tạo ra các ngắt dòng "nhân tạo" như các phím Enter, Shift+Enter hoặc Ctrl+Enter. Nếu sử dụng các phím này, máy tính sẽ luôn ngắt dòng tại vị trí đó. Nguyên tắc tự xuống dòng của từ là một trong những nguyên tắc quan trọng nhất của soạn thảo văn bản trên máy tính. Đây là đặc thù chỉ có đối với công việc soạn thảo trên máy tính và không có đối với việc gõ máy chữ hay viết tay. Chính vì điều này mà đã nảy sinh một số quy tắc mới đặc thù cho công việc soạn thảo trên máy tính.

I.3.2.2. Một số quy tắc gõ văn bản cơ bản

Các nguyên tắc này được áp dụng cho mọi phần mềm soạn thảo và trên mọi hệ điều hành khác nhau.

a. Khi gõ văn bản không dùng phím Enter để điều khiển xuống dòng

Trong soạn thảo văn bản trên máy tính, hãy để cho phần mềm tự động thực hiện việc xuống dòng. Phím Enter chỉ dùng để kết thúc một đoạn văn bản hoàn chỉnh. Chú ý rằng điều này hoàn toàn ngược lại so với thói quen của máy chữ. Với máy chữ, chúng ta luôn phải chủ động trong việc xuống dòng của văn bản.

b. Giữa các từ chỉ dùng một dấu trắng để phân cách, không sử dụng dấu trắng đầu dòng cho việc căn chỉnh lề

Một dấu trắng là đủ để phần mềm phân biệt được các từ. Khoảng cách thể hiện giữa các từ cũng do phần mềm tự động tính toán và thể hiện. Nếu ta dùng nhiều hơn một dấu cách giữa các từ, phần mềm sẽ không tính toán được chính xác khoảng cách giữa các từ và vì vậy văn bản sẽ được thể hiện rất xấu.

c. Các dấu ngắt câu như chấm ".", phẩy ",", hai chấm ":", chấm phảy ";", chấm than "!", hỏi chấm "?" phải được gõ sát vào từ đứng trước nó, tiếp theo là một dấu trắng nếu sau đó vẫn còn nội dung

Lý do đơn giản của quy tắc này là nếu như các dấu ngắt câu trên không được gõ sát vào ký tự của từ cuối cùng, phần mềm sẽ hiểu rằng các dấu này thuộc vào một từ khác, do đó có thể bị ngắt xuống dòng tiếp theo so với câu hiện thời và điều này không đúng với ý nghĩa của các dấu này.

d. Các dấu mở ngoặc và mở nháy đều phải được hiểu là ký tự đầu từ, do đó ký tự tiếp theo phải viết sát vào bên phải của các dấu này. Tương tự, các dấu đóng ngoặc và đóng nháy phải hiểu là ký tự cuối từ và được viết sát vào bên phải của ký tự cuối cùng của từ bên trái.

Chú ý:

- Các quy tắc gõ văn bản trên chỉ áp dụng đối với các văn bản hành chính bình thường. Chúng được áp dụng cho hầu hết các loại công việc hàng ngày như công văn, thư từ, hợp đồng kinh tế, báo chí, văn học,... Tuy nhiên, có một số lĩnh vực chuyên môn hẹp, ví dụ soạn thảo các công thức toán học, lập trình máy tính thì không nhất thiết áp dụng các quy tắc trên.

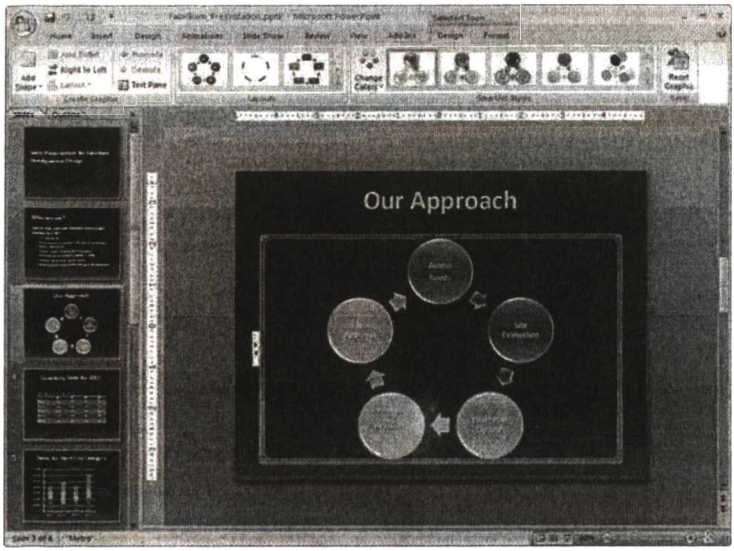
- Các quy tắc vừa nêu trên có thể không bao quát hết các trường hợp cần chú ý khi soạn thảo văn bản trên thực tế. Nếu gặp các trường hợp đặc biệt khác, hãy vận dụng các suy luận có lý của nguyên tắc tự xuống dòng của máy tính để suy luận cho trường hợp riêng của mình.

I.3.3. Hệ trình diễn văn bản

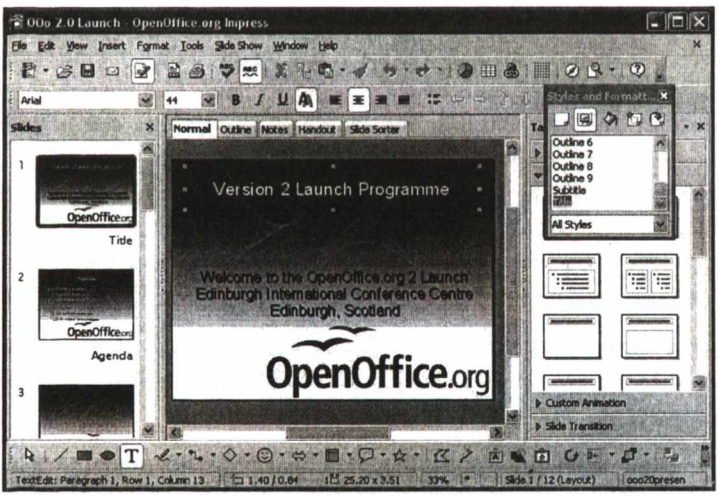
I.3.3.1. Khái niệm

Phần mềm trình diễn là một gói phần mềm máy tính, được sử dụng để trình bày thông tin bao gồm văn bản, âm thanh, hình ảnh, thường là dưới dạng thức chiếu slide. Một phần mềm trình diễn thường có ba chức năng cơ bản: chức năng biên tập để gõ và tạo định dạng chữ, chức năng chèn và điều chỉnh khuôn hình đồ họa, chức năng chiếu slide để thể hiện nội dung.

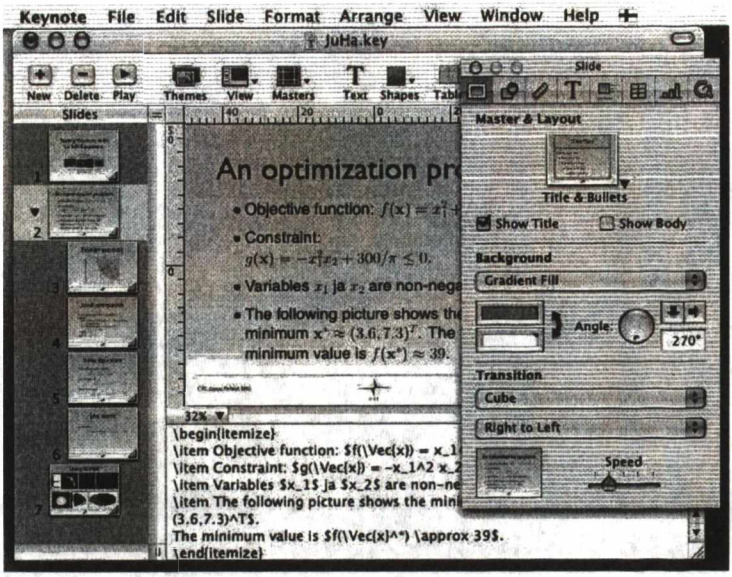
Phần mềm trình diễn nổi tiếng nhất là MS PowerPoint của Microsoft. Các phần mềm trình diễn hay được sử dụng khác là Impress trong bộ OpenOffice và Keynote của Apple.



Hình I.22. Phần mềm MS PowerPoint



Hình I.23. Phần mềm OpenOffice Impress



Hình I.24. Phần mềm Keynote

Các phần mềm trình diễn văn bản cho phép tạo các file trình diễn mới theo chủ quan của người tạo hoặc dựa trên một thư viện các mẫu trình diễn (template) đã có sẵn. Ngoài việc biên tập nội dung văn bản (text), người sử dụng có thể chèn các tập tin âm thanh, hình ảnh để làm cho slide thêm sinh động. Các phần mềm hiện nay đều hỗ trợ các hiệu ứng như hiệu ứng chuyển trang, hiệu ứng chữ. Việc tạo hiệu ứng cho slide–show sẽ làm tăng tính hấp dẫn lôi cuốn người xem. Tuy nhiên cần chú ý tới tính hợp lý cho từng mục đích của buổi thuyết trình. Đối với các buổi trình chiếu quảng cáo sản phẩm hay tiếp thị, nên sử dụng nhiều hiệu ứng mạnh gây ấn tượng. Còn với buổi trình chiếu luận văn tốt nghiệp, thời gian thường không dài, vì vậy, nếu tạo nhiều hiệu ứng thì sẽ làm mất thời gian vô ích.

Người diễn thuyết có thể nhìn những gợi ý trên màn hình máy tính của mình trong quá trình diễn thuyết trong khi những gợi ý này không xuất hiện đối với khán giả và người nghe.

Khi trình diễn, các đối tượng của slide–show hoặc từng slide–show sẽ hiện ra theo từng cái nhấp chuột hoặc biểu tượng nào đó của bàn phím. Người sử dụng nên cho các tiểu mục hiện ra lần lượt để dễ theo dõi.

I.3.3.2. Một số lưu ý khi tạo các file trình diễn

a. Chọn màu sắc

Chọn màu cho chữ là một nghệ thuật. Màu xanh lá cây, xanh nước biển và nâu là những màu "dịu", nhưng khó gây chú ý. Màu đỏ và xanh lá cây có thể khó thấy đối với một số người bị hội chứng mù màu.

Cách chọn màu còn tùy vào bối cảnh và môi trường. Cũng cần phân biệt màu chữ (text color) và màu nền (background color). Kinh nghiệm cho thấy:

- Nếu hội trường nhỏ hay nhằm mục tiêu giảng dạy: chọn chữ màu tối trên nền sáng. Ví dụ như chữ màu đen hay màu xanh đậm và nền trắng.
- Nếu hội trường rộng lớn: chọn chữ sáng trên nền tối, như chữ màu trắng/ vàng trên nền xanh đậm.

Tránh slide với chữ màu xanh lá cây và màu nền đỏ (hay chữ màu đỏ trên nền màu xanh lá cây), vì rất nhiều người bị mù màu với sự kết hợp này. Nói chung, tránh chọn màu nền đỏ hoặc màu cam, dễ làm cho mắt bị mệt và khó theo dõi.

b. Font và cỡ chữ

Có hai nhóm font chữ chính: nhóm chữ không có chân (sans serif) và nhóm có chân. Nhóm sans serif bao gồm Arial, Comic Sans, Papyrus,... Nhóm font chữ có chân bao gồm Times New Roman, Courier, Script,... Nhiều nghiên cứu tâm lí chỉ ra rằng font chữ sans serif thường dễ đọc. Người đọc sử dụng ít thời gian để đọc các font chữ như Arial hơn là Times hay Times New Roman.

Về cỡ chữ (size), phần lớn các chuyên gia khuyến cáo nên dùng cỡ (size) từ 18 trở lên. Nếu dùng font chữ với cỡ nhỏ hơn 18 khán giả sẽ khó đọc, nhất là trong các hội trường rộng. Riêng phần tựa đề, cỡ font chữ phải 40 đến 50. Tuy nhiên, trong trường hợp phải trình bày tài liệu tham khảo thì font size khoảng 12 – 14 có thể chấp nhận được.

Không bao giờ dùng chữ viết hoa như "THIS IS A TEST". Chữ viết hoa được hiểu là không lịch sự. Ngoài ra, chữ viết hoa cũng khó đọc và khó theo dõi. Có thể viết nghiêng hay tô đậm, tuy nhiên đừng nên lạm dụng những cách viết này. Chỉ dùng gạch chân khi cần nhấn mạnh một điều gì quan trọng, nếu không thì nên tránh cách viết này.

c. Số lượng dòng chữ trên một trang

Mỗi trang không nên có quá sáu dòng chữ, mỗi dòng chữ không nên có quá nhiều chữ.

d. Dùng biểu đồ và hình ảnh

Chúng ta thường nhớ hình ảnh hơn các dòng chữ, nhớ biểu đồ hơn các con số. Do đó, cần phải đầu tư thời gian để suy nghĩ về cách trình bày biểu đồ một cách có ý nghĩa.

Có nhiều dạng biểu đồ và mỗi dạng chỉ có thể áp dụng cho một tình huống riêng: biểu đồ hình tròn (pie chart) dùng để thể hiện phần trăm, cơ cấu; biểu đồ cột (bar chart) dùng để so sánh, tương quan hay xếp hạng; biểu đồ tán xạ (scatter plot) dùng để mô tả biến đổi theo thời gian, mối tương quan; bảng số liệu (table) dùng để so sánh số liệu;...

Khi trình bày biểu đồ, cần chú ý định danh và đơn vị trên các trục tung, trục hoành và cần nhắm vào điểm trình bày. Tránh dùng hình hoạt họa vì nó có thể làm giảm sự trang trọng của bài trình bày.

e. Dùng các dấu chấm đầu câu (bullet)

Không nên dùng quá nhiều bullet trong một bài trình bày, không nên lặp lại các từ ngữ trong các câu trên cùng một trang.

f. Mỗi slide chỉ nên trình bày một ý tưởng

Đây là điều quan trọng: một slide chỉ nên trình bày một ý tưởng, không nên nhồi nhét hơn một ý tưởng vào một slide. Do đó, tất cả những bullet, dữ liệu hoặc biểu đồ trong slide chỉ nên dùng để yểm trợ cho ý tưởng chính.

Ý tưởng của slide có thể thể hiện qua tựa đề của slide. Nếu tựa đề slide không chuyển tải được ý tưởng một cách nhanh chóng thì diễn giả sẽ phải tốn thời gian giải thích và có thể làm loãng hay làm cho khán giả sao nhãng vấn đề. Tựa đề trên mỗi slide cũng giống như bảng chỉ đường. Bảng chỉ đường dẫn dắt câu chuyện một cách logic và lí thú. Do đó, tác giả cần phải suy nghĩ cách đặt tựa đề cho mỗi slide sao cho đơn giản nhưng đủ để khán giả biết mình đang ở đâu trong câu chuyện.

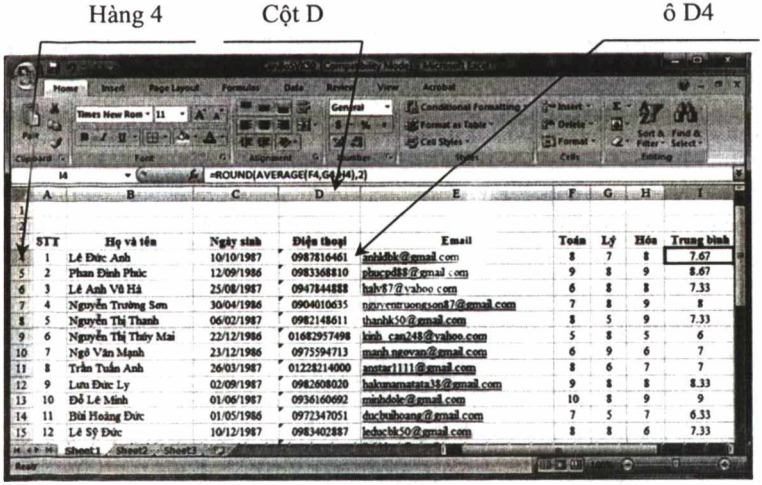
I.3.4. Hệ thông tin bảng tính

Ban đầu, máy tính chỉ là một công cụ trợ giúp việc tính toán cho con người, nhưng trong quá trình phát triển của mình, máy tính đã trở thành một công cụ phục vụ đắc lực cho nhiều lĩnh vực khác. Dù vậy, vai trò chính của máy tính vẫn không hề thay đổi. Trong nhiều lĩnh vực, công việc tính toán vẫn chiếm những vị trí hết sức quan trọng, nhất là lĩnh vực kế toán và phân tích thống kê. Loại phần mềm trợ giúp tính toán thông dụng nhất hiện nay là các phần mềm bảng tính (spreadsheet software – viết tắt là PMBT), bắt nguồn từ ý tưởng của Bricklin và sau này được công ty

Lotus phát triển thành phần mềm thương mại Lotus 1–2–3. Các nhà doanh nghiệp, kỹ sư, nhà khoa học và nhiều người khác dùng PMBT chỉ vì một lý do: PMBT giúp họ tính oán các số liệu, từ đó cho phép họ xây dựng và làm việc với những tình huống nô phỏng thế giới thực.

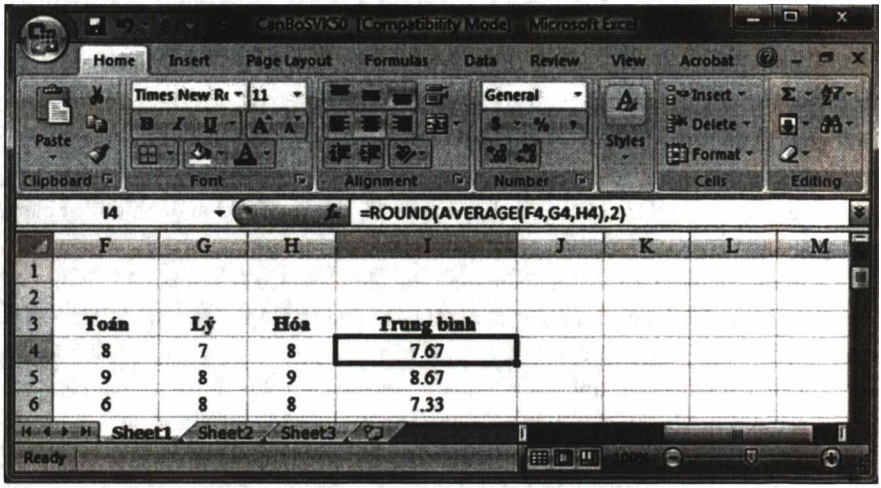
Phần mơn bảng tính đã tạo nên những thay đổi lớn trong hoạt động kinh doanh của con ngươi. Nó cho phép người dùng thao tác và kiểm soát trên các con số và các ký tự với nột lượng lớn dữ liệu. PMBT giúp chúng ta rút ngắn thời gian thực hiện các tác vụ, giúp phát hiện những mối liên hệ ẩn trong dữ liệu, tạo cơ sở cho những dự đoán tương lai.

Các phần mềm bảng tính làm việc dựa trên một khái niệm cơ bản là bảng tính (worksheet). Một bảng tính được thể hiện trên màn hình dưới dạng một ô lưới với các hàng được đánh thứ tự bằng số (bắt đầu từ 1) và các cột được đánh thứ tự bằng chữ (bắtđầu bằng chữ A). Phần giao của một hàng hoặc một cột được gọi là một ô. Mỗi ô trong bảng được xác định bằng một địa chỉ duy nhất tạo bởi chữ cái của cột và số của hàng. Ví dụ, ô nằm ở góc bên trái của bảng là ô A1 (cột A hàng 1), ô nằm ngay bên phải là ô B1, kế đến là C1,..., ô nằm ngay dưới ô A1 là A2, kế đến là A3,... Đối với một bảng tính rỗng (chưa có dữ liệu) thì tất cả các ô đều rỗng; dĩ nhiên, sau này người dùng sẽ điền nội dung vào các ô. Mỗi ô trong bảng tính có thể chứa một giá trị số, một chuỗi kí tự hoặc một công thức hiển thị mối liên hệ giữa các con số trong các ô. Các giá trị số được xem là vật liệu thô dùng để tính toán.



Hình I.25. Phần mềm Excel

Ví dụ: Để thực hiện một thao tác tính toán nào đó, như tính điểm trung bình, chúng ta phải đưa vào một công thức tính toán để hướng dẫn máy tính cách tính ra kết quả cuối cùng. Giả sử ô F4 giá trị là 8 (ô ở hàng thứ 4 cột F), ô G4 có giá trị 7, ô H4 có giá trị 8. Ô I4 sẽ chứa giá trị trung bình của ba ô này và làm tròn, lấy hai chữ số phần thập phân. Tại ô I4, gõ công thức "=ROUND(AVERAGE(F4,G4,H4),2)" và nhấn Enter. Chúng ta sẽ có kết quả mong muốn.



Hình I.26. Tính điểm trung bình và làm tròn với phần mềm Excel

Các phần mềm bảng tính khác nhau có thể khác nhau ở một số chức năng hoặc giao diện nhưng đa số đều có những tính năng gần giống nhau. Một số chức năng thông dụng của PMBT:

Tự động lặp các giá trị, tiêu đề và công thức: Dữ liệu trong các bảng tính có nhiều chỗ bị lặp lại như công thức tính điểm trung bình là giống nhau đối với mọi sinh viên trong cùng một lớp. Các phần mềm bảng tính thường cung cấp nhiều chức năng giúp đơn giản hóa việc nhập các dữ liệu lặp. Các chức năng này khác nhau tùy theo từng phần mềm bảng tính nhưng tất cả đều có điểm chung là dựa trên sự mở rộng của thao tác cắt – dán (cut – paste) cơ bản. Các phần mềm bảng tính còn có khái niệm tham chiếu tương đối đến những ô khác, khi công thức được chép đến một vị trí khác, thì các tham chiếu cũng sẽ chỉ đến những ô khác.

Tự động tính lại: Tự động tính lại là một trong những chức năng quan trọng của các phần mềm bảng tính. Khi có sự thay đổi xảy ra tại một ô tính, phần mềm bảng tính sẽ tự động tính lại toàn bộ bảng tính và còn cho phép người dùng dò tìm đáp

số của bài toán với những dữ liệu vào khác nhau. Đối với các bảng tính lớn, phức tạp, thao tác tự động tính lại có thể rất chậm, vì vậy các phần mềm bảng tính cho phép bạn bật/tắt chức năng tự động tính lại khi cần thiết.

Các hàm thư viện: Các phần mềm bảng tính có sẵn các hàm thư viện. Các phần mềm bảng tính hiện đại đều có một thư viện khổng lồ. Các hàm như SUM AVERAGE, MIN, MAX là những hàm tính toán đơn giản được sử dụng thường xuyên trong mọi loại bảng tính. Ngoài ra, còn có những hàm rất phức tạp liên quan đến lĩnh vực tài chính, toán học, thống kê... Hàm IF cho phép bảng tính quyết định phải làm gì dựa trên nội dung của một ô nào đó, từ đó cung cấp khả năng suy luận logic.

Liên kết các bảng tính: Khi làm việc, người ta sẽ làm việc với nhiều bảng tính liên quan đến nhau cùng lúc nên một thay đổi trong một bảng tính sẽ ảnh hưởng đến các bảng tính khác. Hầu hết các PMBT cho phép tạo ra những liên kết động giữa các bảng tính, vì vậy khi giá trị của một bảng tính bị thay đổi, tất cả các bảng tính liên kết với nó sẽ tự động cập nhật.

Đồ thị trong bảng tính: Đa số phần mềm bảng tính đều có chức năng liên quan đến vẽ đồ thị – một chức năng dùng để chuyển những con số thành hình vẽ, đồ thị một cách tự động. Quá trình tạo một biểu đồ cũng đơn giản như việc điền một thông tin vào một hộp thoại. Như vậy việc tạo lập các biểu đồ từ con số sẽ trở nên dễ dàng, trực quan hơn.

I.3.5. Hệ quản trị cơ sở dữ liệu

I.3.5.1. Khái niệm

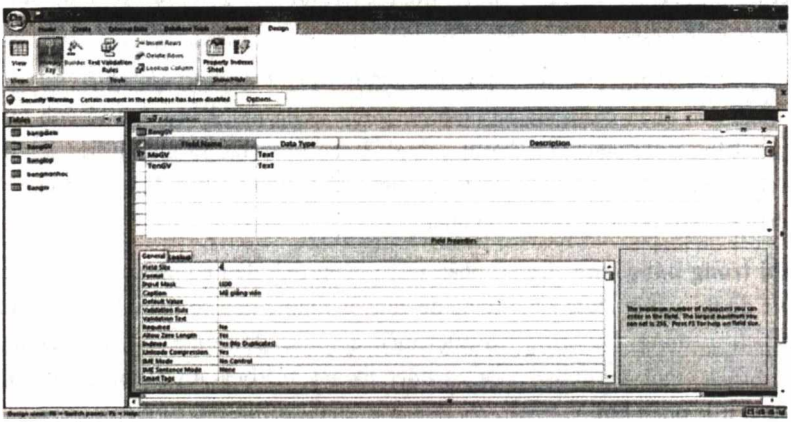
Cơ sở dữ liệu (database) là một bộ sưu tập các dữ liệu tác nghiệp được lưu trữ lại và được các hệ ứng dụng của một tổ chức cụ thể nào đó sử dụng.

Trước đây, khi chưa có các phần mềm quản trị cơ sở dữ liệu, người ta lưu trữ và xử lý dữ liệu dựa trên các hệ thống xử lý tệp truyền thống. Đây là bước khởi đầu của quá trình tin học hóa doanh nghiệp. Cách tiếp cận này tập trung vào nhu cầu xử lý dữ liệu của các bộ phận riêng lẻ trong tổ chức mà không xem xét tổng thể tổ chức này. Do đó, người ta viết một chương trình mới đối với các ứng dụng đơn lẻ. Mỗi chương trình ứng dụng này định nghĩa và quản lý các tệp dữ liệu của riêng nó. Cách tiếp cận này gây ra một số hạn chế: dư thừa và không nhất quán dữ liệu, cô lập và hạn chế chia sẻ dữ liệu, các vấn đề về an toàn và toàn vẹn dữ liệu,..

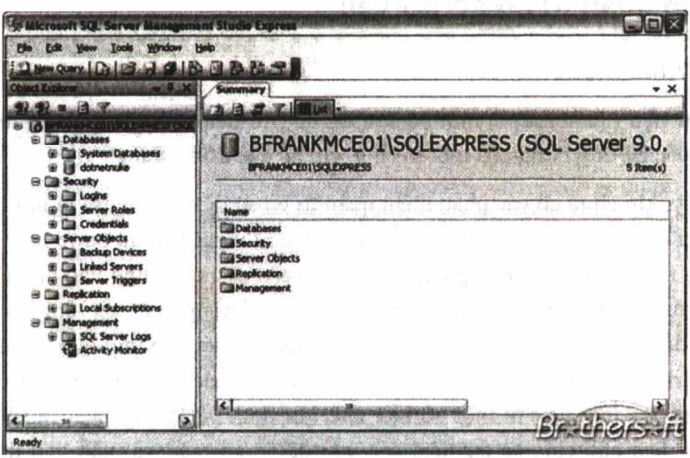
Phần mềm hệ quản trị cơ sở dữ liệu ra đời đã khắc phục được những hạn chế của các hệ thống xử lý dựa trên tệp truyền thống.

Hệ quản trị cơ sở dữ liệu (Database Management System – DBMS) là hệ thống phần mềm cho phép *định nghĩa, tạo lập cơ sở dữ liệu* như xác định kiểu, cấu trúc, ràng buộc dữ liệu, lưu trữ dữ liệu trên các thiết bị nhớ và *thực hiện các thao tác* như truy vấn, cập nhật, kết xuất,... các cơ sở dữ liệu cho các ứng dụng khác nhau.

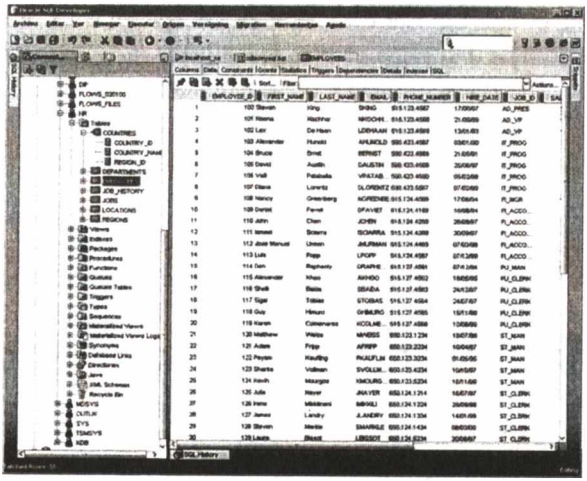
Một số hệ quản trị cơ sở dữ liệu phổ biến hiện nay như: MS SQL Server, MS Access, DB2, PostGreSQL, MySQL, Oracle, FoxPro,...



Hình I.27. Hệ quản trị cơ sở dữ liệu Access



Hình I.28. Hệ quản trị cơ sở dữ liệu SQL Server



Hình I.29. Hệ quản trị cơ sở dữ liệu Oracle

I.3.5.2. Các tính năng của một hệ quản trị cơ sở dữ liệu

Các hệ quản trị cơ sở dữ liệu của các hãng khác nhau có những đặc điểm và mục tiêu khác nhau, nhưng nhìn chung, một hệ quản trị cơ sở dữ liệu thường có các đặc điểm sau:

Quản lý dữ liệu tồn tại lâu dài: Các tổ chức, xí nghiệp có nhu cầu tổ chức, lưu trữ dữ liệu không những trong một vài năm mà thậm chí trong hàng chục năm. Do đó, các hệ quản trị cơ sở dữ liệu phải đảm nhiệm được chức năng này.

Truy xuất dữ liệu một cách hiệu quả: Dữ liệu được lưu trữ bên trong hệ quản trị cơ sở dữ liệu có thể rất lớn và các tổ chức, xí nghiệp thường có nhu cầu thao tác với dữ liệu đó trong một khoảng thời gian nhất định. Các nhà thiết kế và sản xuất phần mềm hệ quản trị cơ sở dữ liệu phải tính toán sao cho dữ liệu được truy xuất một cách hiệu quả.

Hỗ trợ ít nhất một mô hình dữ liệu: Để tổ chức và biểu diễn dữ liệu cần có một mô hình. Các hệ quản trị cơ sở dữ liệu ngày nay thường hỗ trợ mô hình quan hệ trong biểu diễn dữ liệu.

Đảm bảo tính độc lập dữ liệu: Dữ liệu bên dưới và các chương trình thao tác với dữ liệu cần có sự độc lập tương đối, do đó, các chương trình bên trên có thể được phát triển theo nhu cầu của tổ chức mà không ảnh hưởng tới kho dữ liệu đã tồn tại từ trước.

Hỗ trợ các ngôn ngữ cấp cao nhất định cho phép người sử dụng định nghĩa cấu trúc của dữ liệu, truy nhập và thao tác dữ liệu: Ngày nay, các hệ quản trị cơ sở dữ liệu đều hỗ trợ các ngôn ngữ bậc cao như SQL hay QBE khá thân thiện với người dùng. Các ngôn ngữ này giúp họ tạo lập cơ sở dữ liệu, thực hiện các thao tác như thêm mới, tìm kiếm, cập nhật, xóa các khoản mục dữ liệu trong cơ sở dữ liệu.

Quản trị giao dịch: Các hệ quản trị cơ sở dữ liệu có khả năng cung cấp các truy nhập đồng thời đối với cơ sở dữ liệu từ nhiều người sử dụng tại cùng một thời điểm. Việc xử lý các giao dịch tương tranh nhằm đảm bảo tính đúng đắn, nhất quán và chính xác của dữ liệu.

Điều khiển truy nhập: Trong cùng một cơ sở dữ liệu, có thể có nhiều người thực hiện các thao tác. Việc tổ chức phân cấp, phân quyền người sử dụng nhằm đảm bảo dữ liệu được truy cập bởi người dùng hợp pháp.

Sao lưu và phục hồi dữ liệu: Để tránh mất mát dữ liệu khi xảy ra các sự cố về phần cứng và phần mềm cũng như nguyên nhân do con người gây nên, dữ liệu cần được sao lưu với chính sách phù hợp. Khi xảy ra sự cố thì hệ thống có khả năng phục hồi lại trạng thái dữ liệu như trước khi xảy ra sự cố.

I.3.6. Các hệ thống thương mại điện tử

I.3.6.1. Khái niệm thương mại điện tử

Có nhiều khái niệm về thương mại điện tử (TMĐT), nhưng hiểu một cách tổng quát, TMĐT là việc tiến hành một phần hay toàn bộ hoạt động thương mại bằng những phương tiện điện tử. TMĐT vẫn mang bản chất như các hoạt động thương mại truyền thống. Tuy nhiên, thông qua các phương tiện điện tử mới, các hoạt động thương mại được thực hiện nhanh hơn, hiệu quả hơn, giúp tiết kiệm chi phí và mở rộng không gian kinh doanh.

TMĐT càng được biết tới như một phương thức kinh doanh hiệu quả từ khi Internet hình thành và phát triển. Chính vì vậy, nhiều người hiểu TMĐT theo nghĩa cụ thể hơn là giao dịch thương mại, mua sắm qua Internet.

I.3.6.2. Lợi ích của thương mại điện tử

Một trong những lợi ích to lớn của việc áp dụng thương mại điện tử là sự tiết kiệm chi phí và tạo điều kiện thuận lợi cho các bên tham gia vào giao dịch. Giao dịch bằng phương tiện điện tử diễn ra nhanh hơn so với giao dịch thông thường.

Các bên có thể tiến hành giao dịch khi ở cách xa nhau, giữa thành phố với nông thôn, từ nước này sang nước khác mà không bị giới hạn về không gian địa lý. Điều này cho phép các doanh nghiệp tiết kiệm chi phí đi lại, thời gian gặp mặt trong khi mua bán. Với người tiêu dùng, họ có thể ngồi tại nhà để đặt hàng, mua sắm nhiều loại hàng hóa, dịch vụ thật nhanh chóng.

Những lợi ích như trên chỉ có được với những doanh nghiệp thực sự nhận thức được giá trị của TMĐT. Vì vậy, TMĐT góp phần thúc đẩy sự cạnh tranh giữa các doanh nghiệp để thu được nhiều lợi ích nhất. Điều này đặc biệt quan trọng trong bối cảnh hội nhập kinh tế quốc tế, khi các doanh nghiệp trong nước phải cạnh tranh một cách bình đẳng với các doanh nghiệp nước ngoài.

I.3.6.3. Các loại hình ứng dụng thương mại điện tử

Dựa vào chủ thể của TMĐT, có thể phân chia TMĐT ra các loại hình phổ biến như sau:

– Giao dịch giữa doanh nghiệp với doanh nghiệp – **B2B** (business to business): Loại hình này chiếm tỉ trọng lớn trong TMĐT. Các doanh nghiệp có thể tìm kiếm bạn hàng, đặt hàng, ký kết hợp đồng, thanh toán qua các hệ thống này. Ở một mức độ cao, các giao dịch này có thể diễn ra một cách tự động. TMĐT B2B đem lại nhiều lợi ích thực tế cho doanh nghiệp, đặc biệt giúp giảm các chi phí về thu thập thông tin tìm hiểu thị trường, quảng cáo, tiếp thị, đàm phán, tăng các cơ hội kinh doanh,...

– Giao dịch giữa doanh nghiệp với khách hàng – **B2C** (business to consumer): Doanh nghiệp sử dụng các phương tiện điện tử để bán hàng hóa, dịch vụ tới người tiêu dùng. Người tiêu dùng thông qua các phương tiện điện tử để lựa chọn, mặc cả, đặt hàng, thanh toán, nhận hàng. Giao dịch B2C tuy chiếm tỷ trọng nhỏ trong TMĐT nhưng có sự phạm vi ảnh hưởng rộng. Để tham gia hình thức kinh doanh này, thông thường doanh nghiệp sẽ thiết lập website, hình thành cơ sở dữ liệu về hàng hoá, dịch vụ; tiến hành các quy trình tiếp thị, quảng cáo, phân phối trực tiếp tới người tiêu dùng. TMĐT B2C đem lại lợi ích cho cả doanh nghiệp lẫn người tiêu dùng. Doanh nghiệp tiết kiệm nhiều chi phí bán hàng do không cần phòng trưng bày hay thuê người giới thiệu bán hàng, chi phí quản lý cũng giảm hơn. Người tiêu dùng sẽ cảm thấy thuận tiện vì không phải tới tận cửa hàng, có khả năng lựa chọn và so sánh nhiều mặt hàng cùng một lúc.

– Giao dịch giữa doanh nghiệp với cơ quan nhà nước – **B2G** (business to government): **B2G** là loại hình giao dịch giữa doanh nghiệp với cơ quan nhà nước,

trong đó cơ quan nhà nước đóng vai trò khách hàng. Quá trình trao đổi thông tin giữa doanh nghiệp với cơ quan nhà nước được tiến hành thông qua các phương tiện điện tử. Cơ quan nhà nước cũng có thể thiết lập những website để tại đó đăng tải thông tin về nhu cầu mua hàng của các cơ quan nhà nước, tiến hành việc đấu thầu hàng hoá, dịch vụ và lựa chọn nhà cung cấp trên website. Điều này một mặt giúp tiết kiệm các chi phí tìm nhà cung cấp, đồng thời giúp tăng cường tính minh bạch trong hoạt động mua sắm công.

– Giao dịch trực tiếp giữa các cá nhân với nhau – **C2C** (consumer to consumer): **C2C** là loại hình giao dịch giữa các cá nhân với nhau. Sự phát triển của các phương tiện điện tử làm cho nhiều cá nhân có thể tham gia hoạt động thương mại với tư cách là người bán, người cung cấp dịch vụ. Một cá nhân có thể tự thiết lập website để kinh doanh những mặt hàng do mình làm ra hoặc sử dụng một website có sẵn để đấu giá một số món hàng mình có. C2C góp phần tạo nên sự đa dạng của thị trường.

– Giao dịch giữa cơ quan nhà nước với cá nhân – **G2C** (government to consumer): **G2C** là loại hình giao dịch giữa cơ quan nhà nước với cá nhân. Đây chủ yếu là các giao dịch mang tính hành chính, nhưng có thể mang những yếu tố của TMĐT. Ví dụ khi người dân đóng tiền thuế qua mạng, trả phí khi đăng ký hồ sơ trực tuyến,...

I.3.6.4. Thanh toán điện tử

Thanh toán điện tử là hình thức thanh toán tiến hành trên môi trường internet. Thông qua hệ thống thanh toán điện tử, người sử dụng có thể tiến hành các hoạt động thanh toán, chi trả, chuyển tiền, ...

Thanh toán điện tử được sử dụng khi chủ thể tiến hành mua hàng hóa và dịch vụ trên các siêu thị ảo và thanh toán qua mạng. Để thực hiện việc thanh toán, hệ thống máy chủ của siêu thị phải có được chức năng thanh toán trong website của mình.

I.3.6.5. Quảng cáo trên internet

Cũng như các hình thức quảng cáo khác, quảng cáo trên mạng nhằm cung cấp thông tin đẩy nhanh tiến độ giao dịch giữa người bán và người mua. Tuy nhiên, quảng cao trên mạng khác hẳn với quảng cáo trên các phương tiện thông tin đại chúng khác vì nó giúp người tiêu dùng có thể tương tác với quảng cáo. Trên mạng mọi thứ đều có thể đưa vào quảng cáo, từ bố trí sản phẩm tới thiết kế các ảnh nền phía sau nội dung quảng cáo, làm cho logo hoặc bất cứ nhãn hiệu sản phẩm nào

cũng trở nên nổi bật. Quảng cáo trên Internet cũng tạo cơ hội cho các nhà quảng cáo nhắm chính xác vào đối tượng khách hàng của mình và giúp họ quảng cáo với đúng sở thích và thị hiếu người tiêu dùng. Ngoài ra, quảng cáo trên mạng còn là sự kết hợp giữa quảng cáo truyền thống và tiếp thị trực tiếp. Đó là sự kết hợp giữa cung cấp nhãn hiệu, cung cấp thông tin và trao đổi buôn bán ở cùng một nơi.

Các hình thức quảng cáo trên Internet

– Quảng cáo bằng các banner, đường link qua các website khác.

– Quảng cáo qua E–mail.

– Quảng cáo trên Website.

I.3.7. Các hệ thống thông minh

Sự tiến bộ của khoa học, đặc biệt là khoa học trong lĩnh vực máy tính, đã mang lại cho con người những hệ thống xử lý mang đặc tính thông minh, gần với trí tuệ con người. Các hệ thống này được xây dựng mô phỏng theo cơ chế hoạt động và suy nghĩ của con người.

Trí tuệ nhân tạo hay trí khôn nhân tạo (tiếng Anh là Artificial Intelligence, viết tắt là AI) là trí tuệ được biểu diễn bởi bất cứ một hệ thống nhân tạo nào. Trí tuệ nhân tạo là một trong những ngành trọng yếu của tin học. Trí tuệ nhân tạo liên quan đến cách cư xử, sự học hỏi và khả năng thích ứng thông minh của máy móc. Một số ứng dụng như lập kế hoạch, lập lịch, chẩn đoán bệnh, nhận dạng chữ viết, nhận dạng khuôn mặt người, nhận dạng tiếng nói,... Ngày nay các hệ thống nhân tạo được dùng thường xuyên trong kinh tế, y dược, các ngành kỹ thuật quân sự, các phần mềm máy tính thông dụng, trò chơi điện tử. Trí tuệ nhân tạo chia thành hai trường phái tư duy: trí tuệ nhân tạo truyền thống và trí tuệ tính toán.

Hệ chuyên gia, một đại diện của trường phái truyền thống, là một hệ thống áp dụng các khả năng suy luận để đạt tới một kết luận. Một hệ chuyên gia có thể xử lý một lượng lớn thông tin đã biết và đưa ra các kết luận dựa trên các thông tin đó. Hệ chuyên gia làm việc như một chuyên gia thực thụ và cung cấp các ý kiến dựa trên kinh nghiệm của chuyên gia con người đã được đưa vào hệ chuyên gia.

Trí tuệ tính toán nghiên cứu việc học và phát triển lặp. Hệ mờ, một đại diện của trường phái trí tuệ tính toán, áp dụng các kỹ thuật suy luận không chắc chắn, đã được sử dụng rộng rãi trong các hệ thống công nghiệp hiện đại và trong các hệ thống quản lý sản phẩm tiêu dùng.

Chắc hẳn, chúng ta đã từng nghe nói đến hệ cơ sở tri thức. Hệ cơ sở tri thức (CSTT) là hệ thống dựa trên tri thức, cho phép mô hình hóa các tri thức của chuyên gia, dùng tri thức này để giải quyết các vấn đề phức tạp thuộc cùng lĩnh vực. Hai yếu tố quan trọng của hệ CSTT là tri thức chuyên gia và lập luận, tương ứng với hệ thống có hai khối chính là CSTT và mô–tơ suy diễn.

Những hệ thống này, căn cứ trên cơ sở tri thức được trang bị hoặc do học được, dựa trên thông tin đầu vào để trả về kết quả đầu ra theo yêu cầu của con người. Ngoài ra, người ta cũng đã xây dựng những hệ đa tác tử thông minh. Các tác tử này có khả năng cảm nhận, thích nghi với môi trường, khả năng học hỏi từ môi trường, khả năng liên kết và phối hợp với các tác tử khác trong các nhiệm vụ liên quan.

Nhờ có sự phát triển vượt bậc của công nghệ như công nghệ cảm biến, công nghệ số mà chúng ta đã thấy những ngôi nhà thông minh với các đồ đạc và trang thiết bị gần gũi, thân thiện và hiểu được mong muốn của con người. Cảm biến đóng vai trò quan trọng trong hệ thống đo lường và điều khiển hiện đại, nó quyết định việc có thể tự động hóa các quá trình hay không, quyết định độ chính xác và chất lượng của hệ thống. Gần như chúng ta có hết các cảm biến cho các đại lượng vật lý. Sensor là một bộ chuyển đổi đo lường với đầu vào là các đại lượng không điện, đầu ra là các đại lượng mang tính chất điện. Các đại lượng mang tính chất điện này sẽ được số hóa, đưa vào các hệ thống máy tính để xử lý, sau đó máy tính sẽ trả lại kết quả là thông tin hoặc điều khiển một quá trình nào đó.

Trong một tương lai không xa, những hệ thống thông minh sẽ được phát triển và hoàn thiện nhằm phục vụ tốt hơn cuộc sống của con người.

I.4. Câu hỏi và bài tập

1. Trình tự xử lý thông tin trong hệ thống thông tin ?

a. Dữ liệu → Tri thức → Thông tin

b. Thông tin → Tri thức → Dữ liệu

c. Dữ liệu → Thông tin → Tri thức

d. Thông tin → Dữ liệu → Tri thức

2. Thế nào là hệ đếm? Có những hệ đếm cơ bản nào trong hệ thống máy tính?

3. Biểu diễn các số thập phân sau ở hệ nhị phân: 12_{10}, 12.6875_{10}, 35.375_{10}.

4. Biểu diễn các số nhị phân sau ở hệ thập phân: 10101_2, 11101.11_2.

5. Biểu diễn các số bát phân sau ở hệ thập phân: 235.64_8.

6. Biểu diễn các số thập lục phân sau ở hệ thập phân: $34F5C_{16}$.

7. Biểu diễn các số thập phân sau ở dạng hệ thập lục phân: 14988.

8. Biểu diễn số nguyên không dấu sau đây bằng 8 bit: 45, 156.

9. Cho các số nguyên không dấu biểu diễn bằng 8 bit như sau, hãy xác định giá trị của chúng: 0010 1011, 1001 0110.

10. Xác định giá trị của các số nguyên có dấu 8 bit sau đây: 0101 0110, 1101 0010.

11. Biểu diễn các số nguyên sau với 8 bit: +58, –80.

12. Đoạn mã ASCII sau biểu diễn câu gì?

0100 0100 0100 0001 0100 1001 0100 1000 0100 1111 0100 0011 0100 0010
0100 0001 0100 0011 0100 1000 0100 1011 0100 1000 0100 1111 0100 0001

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

    a. A < B

    b. A = B

    c. A > B

14. Số nguyên X ở hệ 10 có giá trị 146, Y ở hệ 16 có giá trị 98 và Z ở hệ 2 có giá trị 10010000. Thứ tự nào dưới dây là đúng ?

    a. Y < X < Z

    b. Y < Z < X

    c. Z < X < Y

    d. X < Y < Z

15. Cho $A = 011010_2$ và $B = 0A_{16}$. C = A*B. Ở hệ 10, giá trị của C là :

    a. 164

    b. 282

    c. 260

    d. 320

16. Hãy nêu chức năng cơ bản của hệ thống máy tính?

17. Cấu trúc cơ bản của CPU gồm có những đơn vị nào?

18. Hãy nêu chức năng của bộ nhớ máy tính?

19. Hãy kể tên một số thiết bị ngoại vi, sau đó phân loại chúng vào các nhóm thiết bị vào, thiết bị ra, thiết bị nhớ, thiết bị truyền thông.

20. Có mấy loại bus? Độ rộng của bus là gì?

21. Một bộ xử lý có đường bus địa chỉ là 32 bit thì dung lượng bộ nhớ tối đa của bộ nhớ chính là bao nhiêu, biết rằng mỗi ngăn nhớ có kích thước là 1 byte?

a. 16 GB

b. 32 GB

c. 8 GB

d. 4 GB

22. Chọn câu trả lời đúng nhất: Các thiết bị sau thuộc về bộ nhớ ngoài?

a. ROM, RAM

b. Đĩa cứng, đĩa mềm

c. Đĩa cứng, đĩa CD ROM, ROM

d. Đĩa cứng, đĩa mềm, bộ nhớ cache

23. Chức năng nào trong số các chức năng sau đây không phải của CPU?

a. Thực hiện các phép toán số học và logic

b. Nhận lệnh

c. Lưu trữ dữ liệu lâu dài

d. Điều khiển hoạt động của các thiết bị

24. Đơn vị đo thông tin nhỏ nhất BIT là viết tắt của:

a. Binary Information Transmission

b. Binary Information Technology

c. Binary Information uniT

d. Binary digiT

25. Nêu khái niệm mạng máy tính.

26. Theo quy mô địa lý, người ta phân loại mạng máy tính ra làm mấy loại? Đó là những loại nào?

27. Khái niệm mạng Internet?

28. Hãy kể tên một số dịch vụ chính của Internet?

29. Hãy nêu khái niệm hệ điều hành.

30. Kể tên những hệ điều hành mà bạn biết.

31. Trong môi trường hệ điều hành Windows, phím tắt dùng để copy là:

a. Ctrl + C
b. Ctrl + V
c. Ctrl + X
d. Ctrl + A

32. Chức năng nào không phải của hệ điều hành?

a. Đọc, ghi ổ đĩa
b. Đọc ghi bộ nhớ trong
c. Soạn thảo văn bản
d. Quản lý các thiết bị

33. FTP là:

a. Dịch vụ truyền tệp tin
b. Dịch vụ tán gẫu trên mạng
c. Dịch vụ truy nhập máy tính từ xa
d. Dịch vụ thư điện tử

34. Chọn câu đúng nhất về vai trò của Modem. Modem cho phép :

a. Sử dụng máy tính như máy điện thoại
b. Tăng tốc độ của máy tính
c. Kết nối máy tính vào mạng internet
d. Tất cả các chức năng trên

35. Thế nào là hệ thống thông tin quản lý? Các thành phần của hệ thống thông tin quản lý gồm có những gì? Các chức năng của hệ thống thông tin quản lý? Trình bày phương pháp SDLC?

36. Hãy kể một số chức năng chính của chương trình soạn thảo văn bản. Thế nào là nguyên tắc tự xuống dòng của từ? Các quy tắc gõ văn bản cơ bản?

37. Ba chức năng cơ bản của phần mềm trình diễn? Các lưu ý khi tạo file trình diễn?

38. Trình bày một số chức năng thông dụng của phần mềm bảng tính.

39. Hãy kể tên một số hệ quản trị cơ sở dữ liệu mà bạn biết. Hãy trình bày các tính năng cơ bản của một hệ quản trị cơ sở dữ liệu?

40. Trình bày khái niệm và lợi ích của thương mại điện tử? Các loại hình ứng dụng thương mại điện tử? Các hình thức quảng cáo trên internet?

41. Trình bày khái niệm chung về các hệ thống thông minh?

PHẦN II. GIẢI QUYẾT BÀI TOÁN

II.1. Giải quyết bài toán

II.1.1. Khái niệm về bài toán

Các hoạt động thực tiễn của cá nhân cũng như cộng đồng xã hội luôn đặt ra các vấn đề hay bài toán cần được giải quyết, từ những hoạt động hàng ngày như giải đáp câu đố, chơi cờ, cân nhắc lựa chọn các phương án, các hành động cần thực hiện, ... đến những vấn đề lớn trong quản lý kinh tế – xã hội hay khoa học – kỹ thuật.

Nhà toán học Pitago đã phân chia mọi vấn đề mà con người phải giải quyết thành hai loại:

- *Theorema*: Là vấn đề cần được khẳng định tính đúng sai, như thường gặp khi chứng minh các định lý trong toán học.
- *Problema*: Là vấn đề cần tìm được giải pháp để đạt được một mục tiêu xác định từ những điều kiện ban đầu nào đó, ví dụ như bài toán dựng hình, tổng hợp chất hóa học, lập thời khóa biểu học tập, tìm đường đi ngắn nhất từ điểm đầu đến điểm đích, ...

Nếu hình thức hóa, thì cả hai loại vấn đề mà Pitago nêu ra đều có thể diễn đạt theo một sơ đồ chung:

$$A \rightarrow B$$

Ở đây: A có thể là giả thiết, điều kiện ban đầu, B có thể là kết luận, mục tiêu cần đạt được, còn $\rightarrow$ là suy luận, giải pháp cần xác định.

Theo sơ đồ này, việc cho một vấn đề – bài toán có nghĩa là cho A và cho B.

Việc giải quyết vấn đề – bài toán có nghĩa là, từ A dùng một số hữu hạn các bước suy luận hợp lý hoặc hành động thích hợp để đạt được B.

Để giải quyết được vấn đề – bài toán thì cần phải chỉ ra tập các thao tác cơ bản được dùng trong suy luận và hành động, nghĩa là những điều kiện ràng buộc đối với yếu tố $\rightarrow$ trong sơ đồ đã nêu. Đối với tin học, sơ đồ này được hiểu theo nghĩa A là input (thông tin vào), B là output (thông tin ra) và $\rightarrow$ cần một giải thuật để thực thi. Hầu hết các giải thuật đều có thể viết thành các chương trình máy tính tuy có thể còn hạn chế bởi khả năng của máy tính và của người lập trình.

Như vậy, chương trình chỉ là một cách mã hóa lại giải pháp để giải quyết vấn đề – bài toán đã cho.

Một bài toán được phát biểu hoàn chỉnh, nếu biết được dạng đầu A, dạng đích B và có thể quyết định khi nào thì bài toán được coi là giải quyết xong. Tuy nhiên, trong rất nhiều bài toán còn tồn tại các yếu tố không xác định. Tính không xác định này thể hiện qua:

- Thông báo về A và B không đầy đủ, rõ ràng
- Thông báo về các điều kiện đặt ra cho giải pháp (→) trong bài toán không đầy đủ, rõ ràng.

Sự xuất hiện của máy tính với tư cách một công cụ trợ giúp quá trình xử lý thông tin đã đem lại diện mạo mới mẻ cho những hoạt động sáng tạo của con người. Càng ngày, con người càng nhận thức sâu sắc vai trò quan trọng của việc tổ chức dữ liệu và giải thuật. Chương trình máy tính là sự kết hợp của hai yếu tố trên, như công thức của N. Wirth: "Chương trình = Cấu trúc dữ liệu + Giải thuật". Có thể nói, giải thuật cung cấp đủ lượng thông tin để giải quyết bài toán.

Hiện nay, để giải quyết một vấn đề trên máy tính, việc thiết kế giải thuật vẫn chủ yếu được thực hiện bởi con người. Từ những thông tin được phản ánh rõ ràng hoặc tiềm ẩn trong A, B hoặc →, cùng với các tri thức liên quan có ở người giải quyết, bài toán sẽ được kết hợp tạo nên giải thuật cần thiết.

Từ khi có máy tính điện tử, các nhà khoa học đã tìm cách chuyển giao dần dần các bước giải quyết vấn đề cho máy tính. Tuy nhiên muốn tự động hóa xây dựng giải thuật chúng ta cần phải làm sao biểu diễn được nội dung bài toán cũng như mọi tri thức có liên quan dưới dạng tường minh và hết sức đầy đủ. Đây chính là lĩnh vực mới mẻ, đang được nghiên cứu mạnh mẽ và có ý nghĩa thực tiễn lớn lao là xây dựng trí tuệ nhân tạo cho máy.

II.1.2. Quá trình giải quyết bài toán bằng máy tính

Việc sử dụng máy tính để giải quyết một bài toán nào đó không đơn giản chỉ là việc lập trình thuần túy, mà cần hiểu lập trình chỉ là một bước trong một quá trình phức tạp bao gồm nhiều bước:

Bước 1 – **Xác định bài toán:** Làm rõ yêu cầu của người sử dụng, đánh giá, nhận định tính khả thi của bài toán.

Bước 2 – **Lựa chọn phương pháp giải:** Có thể có nhiều phương pháp giải khác nhau về thời gian thực hiện, chi phí lưu trữ dữ liệu, độ chính xác... Nhìn chung không có phương pháp tối ưu, cần tùy theo nhu cầu cụ thể và khả năng xử lý tự động mà ta sẽ sử dụng để lựa chọn phương pháp thích hợp.

Bước 3 – **Xây dựng giải thuật:** Xây dựng mô hình chặt chẽ, chính xác và chi tiết hóa của phương pháp đã lựa chọn, xác định rõ ràng dữ liệu vào, dữ liệu ra cho các bước thực hiện và trật tự thực hiện. Nên áp dụng phương pháp thiết kế có cấu trúc, từ thiết kế tổng thể tiến hành làm mịn dần từng bước.

Bước 4 – **Cài đặt chương trình:** Mô tả giải thuật bằng chương trình.

Bước 5 – **Hiệu chỉnh chương trình:** Chạy thử để phát hiện và điều chỉnh các sai sót có thể có ở bước 4. Có hai loại lỗi: lỗi cú pháp và lỗi ngữ nghĩa.

Bước 6 – **Thực hiện chương trình:** Cho máy tính thực hiện chương trình. Tiến hành phân tích kết quả thu được nhằm khẳng định xem kết quả đó có phù hợp hay không. Nếu không, cần kiểm tra lại toàn bộ các bước một lần nữa.

Về tổng thể, giải quyết một bài toán cụ thể được thực hiện qua hai giai đoạn:

- Giai đoạn quan niệm: gồm các bước xác định bài toán, lựa chọn phương pháp giải, xây dựng giải thuật, cài đặt chương trình.
- Giai đoạn khai thác và bảo trì: gồm các bước hiệu chỉnh chương trình và thực hiện chương trình.

Trong quá trình sử dụng, nói chung thường có nhu cầu về cải tiến, mở rộng chương trình do các yếu tố của bài toán ban đầu có thể thay đổi.

II.1.3. Các phương pháp giải quyết bài toán bằng máy tính

II.1.3.1. Giải quyết bài toán theo hướng xác định trực tiếp lời giải

Phương pháp này thường sử dụng trong quá trình học tập. Ví dụ: Tìm nghiệm phương trình bậc hai theo định lý Viet.

Đặc điểm: Xác định trực tiếp được lời giải qua các thủ tục tính toán như công thức, hệ thức, định luật... hoặc thủ tục bao gồm một số hữu hạn các thao tác sơ cấp như dựng hình, phản ứng hóa học... Những thủ tục này có thể chuyển thành các giải thuật và chương trình chạy trên máy.

Một cách tiếp cận hiệu quả là tính toán qua các công thức lặp để xác định gần đúng lời giải của bài toán. Về nguyên tắc, lời giải xác định bởi các công thức lặp có thể xấp xỉ lời giải thật sự của bài toán với độ chính xác tăng theo quá trình lặp. Đây cũng được xem là cách xác định trực tiếp lời giải, khó thực hiện khi tính toán thủ công nhưng lại là thế mạnh của máy tính.

II.1.3.2. Giải quyết bài toán theo hướng tìm kiếm lời giải

Đây là cách tiếp cận chủ yếu của loài người từ xưa đến nay dựa theo nguyên lý thường được gọi là "thử và sai". Một loạt các phương pháp tìm kiếm lời giải theo nguyên lý "thử–sai" được ứng dụng rộng rãi để giải quyết có hiệu quả rất nhiều vấn đề và bài toán phức tạp.

Điển hình như các phương pháp sau:

- *Phương pháp liệt kê hay vét cạn:* Yêu cầu phải xác định tập các khả năng chứa các lời giải và cách thức liệt kê của từng khả năng để thử, không bỏ sót một khả năng nào.
- *Phương pháp thử ngẫu nhiên:* Dựa trên việc thử một số khả năng được chọn ngẫu nhiên trong tập các khả năng (thường là rất lớn). Khả năng thành công tùy theo chiến lược chọn ngẫu nhiên và một số điều kiện cụ thể của bài toán.
- *Phân chia bài toán thành các bài toán con*, cho đến khi bài toán ban đầu được quy thành các bài toán con có lời giải. Kết hợp lời giải của các bài toán con cho lời giải của bài toán ban đầu.
- *Phương pháp quay lui:* Nhiều khi không thể xác định hay liệt kê từ trước mọi khả năng chứa lời giải của bài toán. Chẳng hạn tìm đường đi trong mê cung thì không thể biết hết được mọi ngả đường có thể dẫn đến lối ra. Khi đó cần phải có một cách đánh dấu các thử nghiệm thất bại (đường đi vào ngõ cụt) và thử khả năng mới (quay lui tìm đường khác).

Ví dụ: Bài toán phân tích số tự nhiên N ra thừa số nguyên tố.

Ở đây, thông tin ban đầu là số tự nhiên N, thông tin đích là các thừa số nguyên tố của N, có thể tính toán trực tiếp ra lời giải bằng cách chia số N cho các số nguyên tố lần lượt từ 2 cho đến số nguyên tố lớn nhất nhỏ hơn căn bậc hai của N. Nếu chia hết, hay là phần dư bằng 0, thì số nguyên tố đó là một thừa số nguyên tố của N và tiếp tục quá trình tính với thương số của N và thừa số nguyên tố vừa tìm ra.

Kết thúc quá trình tính, sau khi đã kiểm tra hết các số nguyên tố cần xét mà số còn lại vẫn bằng N thì N chính là một số nguyên tố.

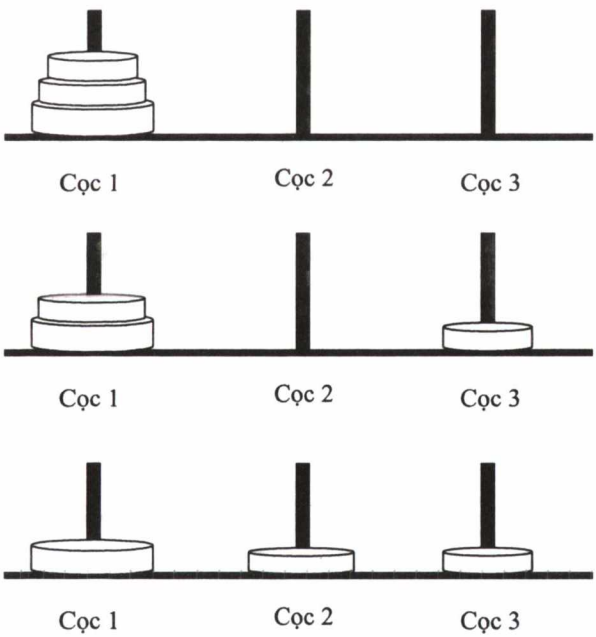
Ví dụ: Bài toán tháp Hà Nội.

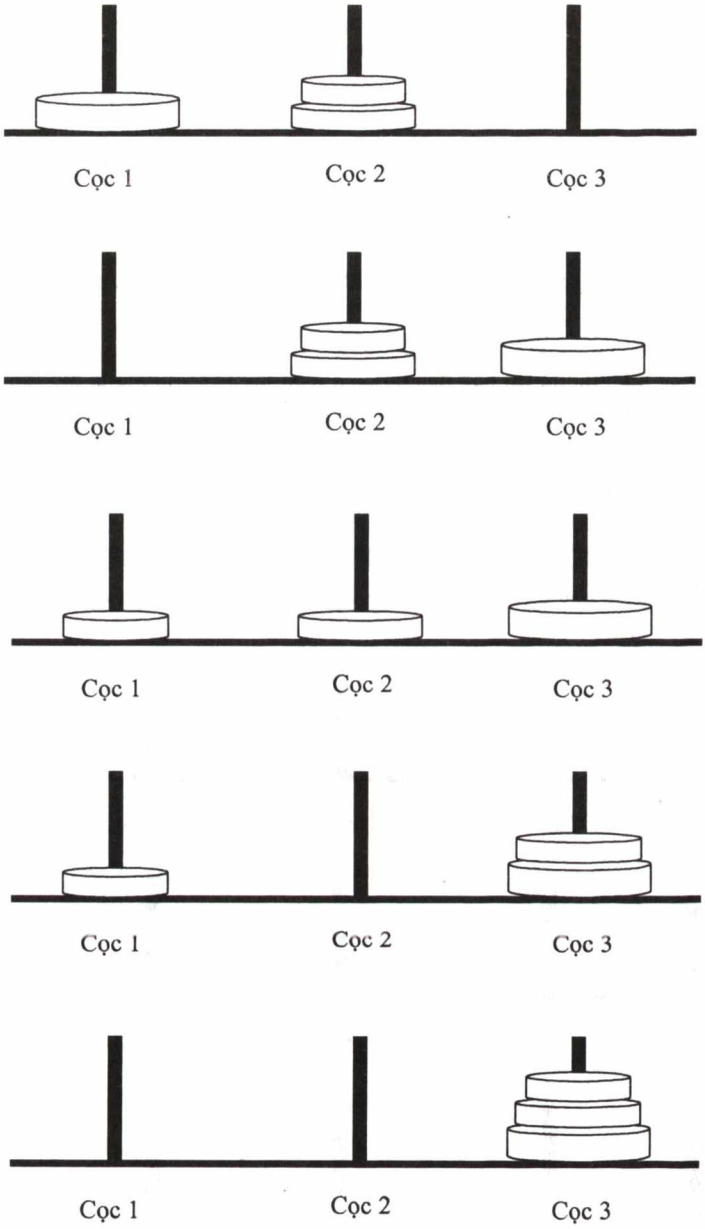
Cho ba cọc 1, 2, 3. Ở cọc 1 ban đầu có N đĩa sắp xếp theo thứ to dần từ trên xuống dưới. Cần dịch chuyển N đĩa đó sang cọc 3 sao cho:

- Mỗi lần chuyển chỉ làm với 1 đĩa.
- Trong mỗi cọc không cho phép đĩa to nằm trên đĩa nhỏ hơn.

Giả sử N = 3, có ba đĩa, gọi tên: đĩa lớn, đĩa vừa, đĩa bé.

Có thể biểu diễn thông tin ban đầu của bài toán là "111", nghĩa là đĩa lớn ở cọc 1 đĩa vừa ở cọc 1 và đĩa bé cũng ở cọc 1. Dạng đích của bài toán sẽ là "333". Cứ mỗi lần chuyển một đĩa từ cọc này sang cọc khác thì sẽ làm biến đổi trạng thái của bài toán. Ví dụ, từ trạng thái ban đầu, nếu chuyển đĩa bé từ cọc 1 sang cọc 3, thì trạng thái của bài toán chuyển từ "111" thành "113". Ở đây, có thể dùng phương pháp tìm kiếm lời giải. Một ví dụ về đường đi tìm kiếm ra lời giải: 111→ 113 → 123 → 122 → 322 → 321 → 331 → 333





Hình II.1. Bài toán tháp Hà Nội

II.2. Thuật toán

II.2.1. Định nghĩa thuật toán

Thuật toán là một khái niệm cơ sở của toán học và tin học. Thuật toán là một tập các quy tắc hay quy trình nhằm hướng dẫn việc thực hiện một công việc nào đó. Chính xác hơn, thuật toán bao gồm một dãy hữu hạn các chỉ thị rõ ràng cho việc hoàn tất một công việc từ một trạng thái ban đầu cho trước, nhằm đạt được mục tiêu đề ra. Khi các chỉ thị này được áp dụng thì sẽ dẫn đến kết quả. Việc học hay nghiên cứu thuật toán giữ vai trò rất quan trọng trong khoa học máy tính vì máy tính có khả năng thực hiện công việc theo một thuật toán xác định rõ phải từng bước làm gì. Về mặt phương pháp, ta có thể xem thuật toán như là sự thể hiện của một phương pháp để giải quyết một vấn đề.

Trong khoa học máy tính, thuật toán được định nghĩa gồm một dãy các bước không mập mờ và có thể thực thi được. Nghĩa là, tại mỗi bước, thì hành động kế tiếp phải được xác định một cách duy nhất theo chỉ thị và theo dữ liệu thích hợp ở thời điểm đó.

Ngoài ra, quá trình thực hiện theo thuật toán phải dừng và cho ra kết quả như mong muốn.

Ví dụ: Thuật toán tìm phần tử lớn nhất trong một dãy hữu hạn các số nguyên.

Có nhiều bài toán trong thực tế đòi hỏi phải tìm ra số nguyên lớn nhất trong một dãy số. Chẳng hạn như việc tìm ra một học sinh có điểm số cao nhất trong một kỳ thi, hay tìm một nhân viên có năng suất cao nhất trong một xí nghiệp...

Một trong các phương pháp để tìm phần tử lớn nhất trong một dãy số nguyên là thực hiện một thủ tục theo các bước sau đây:

1. Trước hết đặt cho giá trị lớn nhất tạm thời bằng số nguyên đầu tiên (Giá trị lớn nhất tạm thời này chính là giá trị lớn nhất ở mỗi giai đoạn của thủ tục).
2. So sánh số nguyên kế tiếp trong dãy với giá trị lớn nhất tạm thời và nếu nó lớn hơn giá trị lớn nhất tạm thời thì đặt cho giá trị lớn nhất tạm thời bằng số nguyên này.
3. Lặp lại bước 2 nếu còn số nguyên trong dãy chưa được xét tới.
4. Dừng nếu không còn số nguyên nào trong dãy chưa được xét tới. Giá trị lớn nhất tạm thời lúc này chính là giá trị lớn nhất trong dãy số.

Bài toán trên là một ví dụ khá thích hợp để minh họa thuật toán, có nhiều phương pháp khác nhau để giải. Tuy nhiên, để phát biểu một thuật toán thì phải rõ ràng và thực thi được. Ví dụ, nếu xét dãy các bước sau:

(i) Tạo danh sách tất cả các số nguyên,

(ii) Sắp xếp danh sách theo thứ tự giảm dần,

(iii) Lấy ra số nguyên đầu tiên từ danh sách đã được sắp xếp,

(iv) Dừng,

thì không phải là một thuật toán vì các bước (i) và bước (ii) không thể thực hiện được.

Các đặc trưng của thuật toán

Khi mô tả một thuật toán chúng ta cần chú ý đến một số tính chất đặc trưng sau đây:

- ***Tính đúng đắn***: Để đảm bảo kết quả tính toán hay các thao tác mà máy tính thực hiện được là chính xác.
- ***Tính xác định***: Thuật toán phải được thể hiện bằng các câu lệnh minh bạch; các câu lệnh được sắp xếp theo thứ tự nhất định.
- ***Tính khách quan***: Một thuật toán dù được viết bởi nhiều người trên nhiều máy tính vẫn phải cho kết quả như nhau.
- ***Tính tổng quát***: Thuật toán có thể áp dụng được cho tất cả các bài toán có dạng như mong muốn chứ không phải chỉ áp dụng cho một số trường hợp riêng lẻ nào đó.
- ***Tính dừng***: Thuật toán phải cho ra lời giải (hay kết quả) sau một số hữu hạn các bước.
- ***Hiệu quả của thuật toán***: được đánh giá dựa trên một số tiêu chuẩn như khối lượng tính toán, không gian và thời gian được sử dụng khi thực hiện thuật toán trên máy tính.

II.2.2. Biểu diễn thuật toán

Để có thể trình bày hay truyền đạt một thuật toán cho người khác hoặc chuyển thuật toán thành chương trình máy tính, thì cần phải tìm cách biểu diễn thuật toán. Có thể sử dụng các ngôn ngữ hình thức để biểu diễn thuật toán.

Các ngôn ngữ thường được sử dụng để biểu diễn thuật toán là:

- Ngôn ngữ tự nhiên
- Ngôn ngữ lưu đồ (sơ đồ khối)
- Ngôn ngữ tựa ngôn ngữ lập trình (mã giả)
- Ngôn ngữ lập trình

Trong cách biểu diễn thuật toán theo ngôn ngữ tự nhiên, người ta liệt kê các bước của thuật toán bằng ngôn ngữ. Cách này không yêu cầu người viết thuật toán cũng như người đọc thuật toán phải chuẩn bị một số kiến thức đặc biệt như đối với cách biểu diễn bằng lưu đồ hay mã giả. Tuy nhiên cách biểu diễn theo ngôn ngữ tự nhiên thường dài dòng, không làm nổi bật được cấu trúc của thuật toán. Trong một số trường hợp, việc viết một số bước thực hiện trong thuật toán theo ngôn ngữ tự nhiên tỏ ra không hề dễ dàng và rất khó hiểu.

Ví dụ: Tìm giá trị lớn nhất của một dãy số nguyên có N số.

- Đầu vào: Số nguyên dương N và N số nguyên a[1], a[2],..., a[N]
- Đầu ra: Số nguyên a[k] lớn nhất của dãy, k nguyên dương và nằm trong đoạn [1, N].

Ý tưởng

- Khởi tạo giá trị max = a[1].
- Lần lượt so sánh max với a[i], i = 2, 3,..., N, nếu a[i] > max ta gán giá trị mới cho max.

Thuật toán

B1 Nhập dãy số a[1], a[2], ..., a[N].

B2 max ← a[1], i ← 2.

B3 Nếu i > N, thuật toán kết thúc và max là giá trị lớn nhất của dãy cần tìm.

B4 Nếu a[i] > max, gán a[i] cho max.

B5: Tăng i lên 1 đơn vị.

B6: Quay lên B4.

B7: Kết thúc.

Cách sử dụng lưu đồ cũng như sử dụng mã giả tỏ ra khá thuận lợi trong việc viết cũng như đọc thuật toán. Thông thường người ta sử dụng ba hình thức biểu diễn thuật toán: ngôn ngữ tự nhiên, lưu đồ, mã giả. Trong phần này sẽ giới thiệu cách biểu diễn thuật toán bằng lưu đồ và mã giả.

II.2.2.1. Ngôn ngữ lưu đồ

Ngôn ngữ lưu đồ hay sơ đồ khối là một công cụ rất trực quan để diễn đạt các thuật toán. Lưu đồ là một hệ thống các nút có hình dạng khác nhau, thể hiện các chức năng khác nhau và được nối với nhau bởi các cung.

Biểu diễn bằng lưu đồ sẽ giúp ta có được một cái nhìn tổng quan về toàn cảnh của quá trình xử lý theo thuật toán.

Lưu đồ được tạo thành từ bốn thành phần chủ yếu sau đây:

a. Nút giới hạn

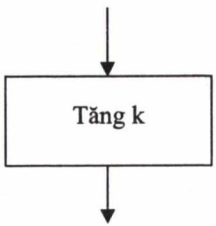
Được biểu diễn bởi hình ô–van có ghi chữ bên trong như:

Các nút trên còn được gọi là nút đầu và nút cuối của lưu đồ.

b. Nút thao tác

Là một hình chữ nhật có ghi các lệnh cần thực hiện.

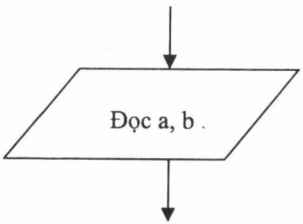
Ví dụ:



c. Nút vào/ra dữ liệu

Là một hình bình hành có ghi các thao tác nhập và xuất dữ liệu.

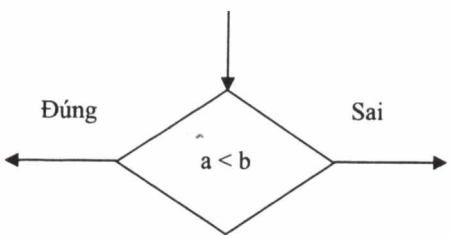
Ví dụ:



d. Nút điều kiện

Thường là một hình thoi có ghi điều kiện cần kiểm tra. Trong các cung nối với nút này, có hai cung ra chỉ hướng đi theo hai trường hợp: điều kiện đúng (True) và điều kiện sai (False).

Ví dụ:



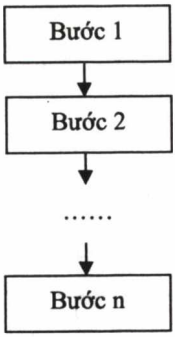
e. Cung

Là các đường nối từ nút này đến nút khác của lưu đồ.

Hoạt động của thuật toán theo lưu đồ được bắt đầu từ nút đầu tiên. Sau khi thực hiện các thao tác hoặc kiểm tra điều kiện ở mỗi nút thì bộ xử lý sẽ theo một cung đến một nút khác. Quá trình thực hiện thuật toán sẽ dừng khi gặp nút kết thúc hay nút cuối.

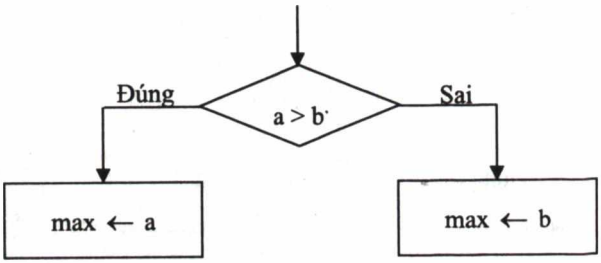
Trong kỹ thuật lập trình cấu trúc, thuật toán được biểu diễn sử dụng các cấu trúc cơ bản sau:

Cấu trúc tuần tự: Các bước được thực hiện theo một trình tự tuyến tính, hết bước này sang bước khác.



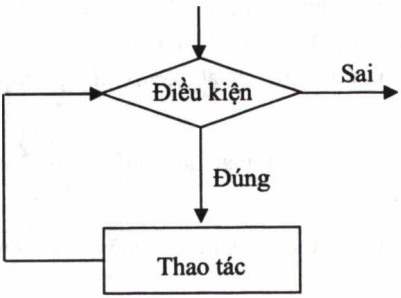
Hình II.2. Cấu trúc tuần tự

Cấu trúc rẽ nhánh: Việc thực hiện bước tiếp theo phụ thuộc vào điều kiện trước đó. Ví dụ, tìm max của hai số a và b. Nếu a > b thì max là a, ngược lại max là b.



Hình II.3. Cấu trúc rẽ nhánh

Cấu trúc lặp: Một thao tác/hành động/nhiệm vụ có thể được thực hiện nhiều lần. Số lần lặp có thể được biết trước hoặc không. Tuy nhiên, số lần lặp phải hữu hạn.

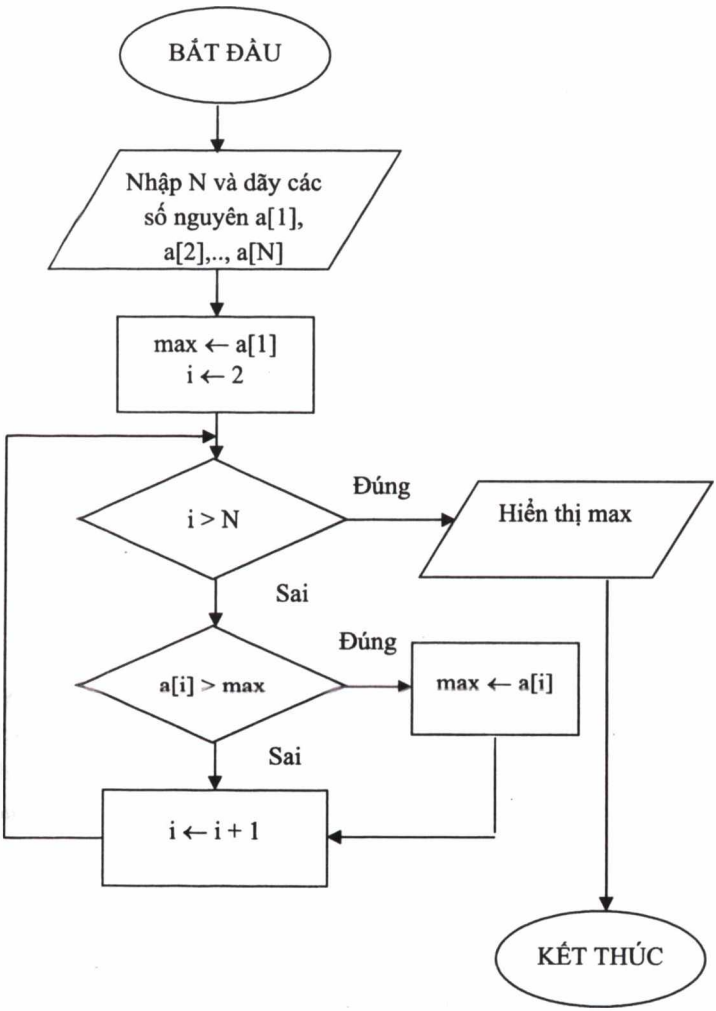


Hình II.4. Cấu trúc lặp

Ví dụ:

1: Tìm giá trị lớn nhất của một dãy số nguyên có N số.

Vì thuật toán là chỉ ra dãy thao tác và trình tự thao tác để đạt mục đích và dùng cho con người, nên ngoài cách liệt kê trên, người ta có thể dùng sơ đồ khối để minh hoạ (biểu diễn). Với thuật toán trên, cách biểu diễn theo sơ đồ khối như sau:



Hình II.5. Thuật toán tìm giá trị lớn nhất của dãy số nguyên

2: Sắp xếp bằng phương pháp tráo đổi (Exchange Sort)

- Đầu vào: Dãy gồm N số nguyên a[1], a[2], …, a[N].
- Đầu ra: Dãy được sắp xếp theo thứ tự không giảm.

Ý tưởng

- Với mỗi cặp số liên tiếp trong dãy, nếu số trước lớn hơn số sau ta đổi chỗ chúng cho nhau.
- Việc đó được lặp cho đến khi không còn sự đổi chỗ nào nữa.

Thuật toán

B1: Nhập số N và dãy số a[1], a[2], …, a[N].

B2: j ← N.

B3: Nếu j < 2 thì thuật toán kết thúc và hiển thị dãy đó.

B4: j ← j – 1, i ← 0.

B5: Tăng i lên 1 đơn vị.

B6: Nếu i > j thì quay lại bước B3.

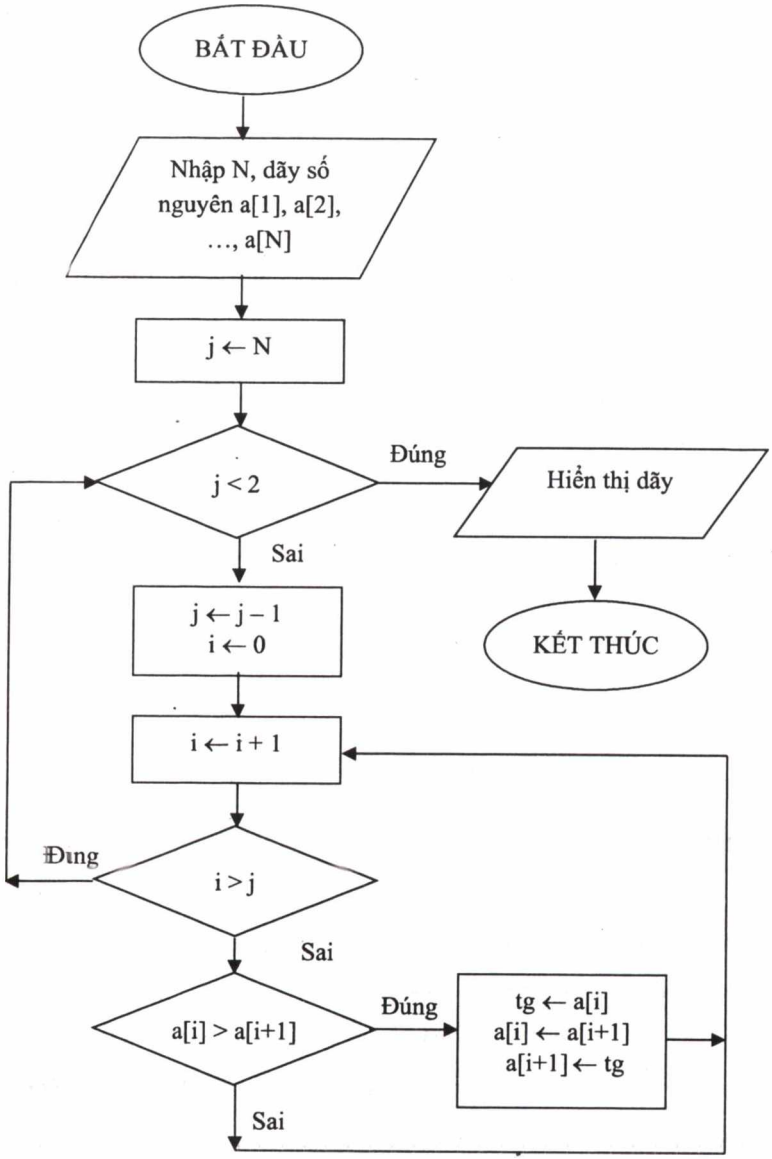
B7: Nếu a[i] > a[i+1] thì tráo đổi hai số đó cho nhau.

B8: Quay lên bước B5.

Chú ý:

- Thuật toán này tạo ra sau mỗi lượt sắp một phần tử đúng vị trí và phần tử này không còn tham gia vào quá trình sắp nữa. Nó giống như bọt nước nổi lên mặt nước: bóng nhẹ sẽ được đẩy dần lên trên. Cũng chính vì thế mà sắp xếp tráo đổi còn có tên "*nổi bọt*" (Bubble Sort).
- Quá trình được lặp lại với dãy sau khi đã bỏ phần tử cuối dãy, do vậy lúc đầu j được gán với giá trị N và dừng khi j < 2.
- Trong thuật toán trên, i được khởi tạo giá trị 0 và tiến tới j + 1.

Với thuật toán trên, cách biểu diễn theo sơ đồ khối như sau:



Hình II.6. Thuật toán sắp xếp bằng phương pháp tráo đổi

II.2.2.2. Mã giả

Với các thuật toán phức tạp thì việc sử dụng lưu đồ sẽ rất cồng kềnh. Do vậy, người ta còn biểu diễn các thuật toán theo một ngôn ngữ tựa ngôn ngữ lập trình được gọi là mã giả. Trong mã giả ta sử dụng các mệnh đề có cấu trúc chuẩn hóa và vẫn dùng ngôn ngữ tự nhiên. Cách viết thuật toán bằng mã giả tỏ ra tiện lợi, đơn giản và dễ hiểu.

Trong mã giả cũng sử dụng các ký hiệu toán học, các biến và đôi khi cả cấu trúc kiểu thủ tục. Cấu trúc thuật toán kiểu thủ tục thường được sử dụng để trình bày các thuật toán đệ quy hay các thuật toán phức tạp cần phải được trình bày thành nhiều cấp độ.

Cùng với việc sử dụng các biến, trong thuật toán rất thường gặp một phát biểu hành động đặt (hay gán) một giá trị cho một biến. Ví dụ: Hành động tăng biến i lên 1 có thể được viết như sau:

$i \leftarrow i + 1$

Các cấu trúc thường gặp trong mã giả gồm:

Cấu trúc điều kiện

if (điều kiện) **then** (hành động) **end if**

và

if (điều kiện) **then** (hành động)

else (hành động)

end if

Cấu trúc lặp

while (điều kiện) **do** (hành động) **end while**

repeat (hành động) **until** (điều kiện)

for (biến) = (giá trị đầu) **to** (giá trị cuối) **do** (hành động) **end for**

for (biến) = (giá trị đầu) **downto** (giá trị cuối) **do** (hành động) **end for**

Cấu trúc nhảy goto

goto (nhãn)

II.2.3. Một số thuật toán thông dụng

II.2.3.1. Thuật toán hoán vị giá trị hai biến

Bài toán: Nhập giá trị cho hai biến x, y. Hãy hoán đổi giá trị của hai biến này.

Tiến hành như sau:

1. Nhận các giá trị cho x và y.
2. Dùng một biến trung gian tg và lưu giá trị chứa trong x vào biến tg này (gán tg = x).
3. Lưu giá trị chứa trong y vào x (gán x = y).
4. Lưu giá trị trong biến tg vào y (gán y = tg).
5. Kết thúc.

Thuật toán được biểu diễn bằng mã giả như sau

Nhập: x, y

Xuất: x, y đã hoán đổi giá trị

Thuật toán:

1. tg ← x
2. x ← y
3. y ← tg
4. Xuất x và y

II.2.3.2. Thuật toán kiểm tra số nguyên tố

Bài toán đặt ra là cho một số nguyên dương p, hỏi p có phải là số nguyên tố hay không?

Cách đơn giản nhất để biết p có phải số nguyên tố hay không là dựa vào định nghĩa số nguyên tố. Tiến hành theo các bước sau đây:

1. Nhận giá trị p.
2. Nếu p = 1 thì xuất kết quả: p không phải số nguyên tố.
3. Nếu p ≠ 1 thì kiểm tra xem trong các số nguyên dương từ 2 đến p – 1 có tồn tại ước số của p hay không? Nếu có thì p không là số nguyên tố; ngược lại p là số nguyên tố.
4. Kết thúc.

Thuật toán được mô tả như sau

Nhập: p nguyên dương.

Xuất: Kết luận về tính nguyên tố của p.

Thuật toán:

1. if p = 1 then
   Xuất: p không nguyên tố, Dừng
   end if
2. flag ← TRUE (gán cho cờ hiệu "flag" giá trị TRUE)
3. for k = 2 to p–1 do
   if (k là ước số của p) then
   flag ← FALSE
   break (ngắt vòng lặp FOR)
   end if
   end for
4. if flag = TRUE then
   Xuất: p là số nguyên tố
   else
   Xuất: p không là số nguyên tố
   end if

II.2.3.3. Thuật toán tìm phần tử lớn nhất trong một dãy hữu hạn số

Bài toán đặt ra là cho một dãy hữu hạn các số nguyên (hoặc số thực), hãy tìm và trả về phần tử lớn nhất trong dãy. Có nhiều cách để giải bài toán này. Một trong các phương pháp để tìm phần tử lớn nhất trong một dãy số nguyên là thực hiện một thủ tục theo các bước sau đây:

1. Nhận các giá trị của dãy.
2. Đặt giá trị lớn nhất tạm thời bằng số nguyên đầu tiên (giá trị lớn nhất tạm thời này chính là giá trị lớn nhất ở mỗi giai đoạn của thủ tục).

3. So sánh số nguyên kế tiếp trong dãy với giá trị lớn nhất tạm thời, nếu nó lớn hơn giá trị lớn nhất tạm thời thì đặt lại giá trị lớn nhất tạm thời bằng số nguyên này.
4. Lặp lại bước 2 nếu còn số nguyên trong dãy chưa được xét.
5. Dừng nếu không còn số nguyên nào trong dãy chưa được xét. Giá trị lớn nhất tạm thời lúc này chính là giá trị lớn nhất trong dãy số.
6. Kết thúc.

Thuật toán được mô tả như sau

Nhập: Dãy số a[1], a[2], a[3],... a[n].

Xuất: Max là giá trị lớn nhất trong dãy số đã cho.

Thuật toán:

1. max ← a[1]
2. for i = 2 to n do

```
        if max < a[i] then
            max ← a[i]
        end if
    end for
```

3. Xuất: max là giá trị lớn nhất trong dãy số

II.2.3.4. Thuật toán giải phương trình bậc hai

Bài toán đặt ra: Tìm và trả về nghiệm thực (nếu có) của phương trình bậc hai $ax^2 + bx + c = 0$ với a, b, c là các tham số nhập từ bàn phím.

Bài toán được giải quyết theo các bước sau:

1. Nhận các giá trị a, b, c.
2. Kiểm tra hệ số a = 0 hay không? Nếu a bằng 0 thì kết luận đây không phải phương trình bậc hai và dừng.
3. Nếu $a \neq 0$, tính giá trị biểu thức $\Delta = b^2 - 4*a*c$.
4. Xét dấu biểu thức Δ. Nếu $\Delta > 0$ thì tính các nghiệm thực theo công thức:

x_1 = (–b–sqrt(Δ))/(2*a), x_2 = (–b+sqrt(Δ))/(2*a) rồi xuất kết quả ra màn hình.

5. Nếu Δ = 0 thì xuất kết quả phương trình có nghiệm kép là –b/(2*a)
6. Nếu Δ < 0 thì xuất kết quả: Phương trình không có nghiệm thực.
7. Kết thúc.

Thuật toán được mô tả như sau

Nhập: Các hệ số a, b, c.

Xuất: Kết luận về nghiệm của phương trình bậc hai.

Thuật toán:

1. if a = 0 then

   Xuất: Không phải phương trình bậc hai, Dừng

   end if
2. delta ← b*b–4*a*c
3. if delta > 0 then

   x1 ← (–b–sqrt(Δ))/(2*a)

   x2 ← (–b+sqrt(Δ))/(2*a)
4. else if delta = 0 then x12 ← –b/(2*a), Xuất: nghiệm kép x12
5. else Xuất: phương trình không có nghiệm thực

   end if

II.2.3.5. Thuật toán sắp xếp dãy

Bài toán: Cho một dãy hữu hạn các số (nguyên, thực), hãy sắp xếp dãy này theo chiều tăng dần (hoặc giảm dần) giá trị của các phần tử của dãy.

Có khá nhiều thuật toán sắp xếp một dãy tăng (hoặc giảm) dần. Trong giáo trình sẽ giới thiệu một thuật toán khá phổ biến. Bài toán sắp xếp theo chiều tăng dần thực hiện theo các bước sau:

1. Nhận giá trị dãy.
2. Đầu tiên, so sánh phần tử x[1] với n–1 phần tử từ danh sách x[2], x[3],…, x[n], nếu x[1] lớn hơn phần tử nào thì hoán vị nó với phần tử đó.

3. Tiếp theo, so sánh phần tử x[2] với n–2 phần tử từ danh sách x[3], x[4],..., x[n], nếu x[2] lớn hơn phần tử nào thì hoán vị nó với phần tử đó.
4. Tổng quát ở bước thứ i, so sánh phần tử x[i] với n–i phần tử từ danh sách x[i+1], x[i+2],..., x[n], nếu x[i] lớn hơn phần tử nào thì hoán vị nó với phần tử đó.
5. Kết thúc.

Thuật toán được mô tả như sau

Nhập: Dãy số x[1], x[2], ..., x[n].

Xuất: Dãy đã sắp xếp theo chiều giá trị các phần tử tăng dần.

Thuật toán:

```
1. for i = 1 to n–1 do
        for j = i+1 to n do
            if x[i] > x[j] then
                tg ← x[i], x[i] ← x[j], x[j] ← tg
            end if
        end for
   end for
```

II.2.4. Thuật toán đệ quy

Thuật toán đệ quy là một trong những sự mở rộng của khái niệm thuật toán. Như đã biết, một thuật toán được đòi hỏi phải thỏa mãn các tính chất như: tính xác định, tính dừng, tính đúng đắn,...

Tuy nhiên có những trường hợp việc tìm ra một thuật toán có những tính chất đòi hỏi như trên rất khó khăn, nhưng có cách giải có thể vi phạm các tính chất thuật toán mà lại khá đơn giản và chấp nhận được. Ví dụ, trong những trường hợp bài toán có thể được phân tích và đưa tới việc giải một bài toán cùng loại nhưng cấp độ thấp hơn, chẳng hạn độ lớn dữ liệu nhập nhỏ hơn, giá trị cần tính toán nhỏ hơn,... Ta cũng thường thấy những định nghĩa về những đối tượng, những khái niệm dựa trên chính những đối tượng, những khái niệm đó như ví dụ dưới đây:

Ví dụ: Định nghĩa giai thừa

Giai thừa của một số tự nhiên n, ký hiệu là n!, được định nghĩa bằng cách quy nạp như sau:

0! = 1,

n! = (n–1)! * n, với mọi n > 0

Thuật toán để giải bài toán trong những trường hợp như trên thường được dựa trên chính nó, tức là trong các bước của thuật toán có thể có trường hợp thực hiện lại thuật toán đó (nhưng thường với dữ liệu nhập có độ lớn thấp hơn). Những thuật toán loại này gọi là "thuật toán đệ quy". Tư tưởng giải bài toán bằng thuật toán đệ quy là đưa bài toán hiện tại về một bài toán cùng loại, cùng tính chất (đồng dạng) nhưng ở cấp độ thấp hơn (chẳng hạn, độ lớn dữ liệu vào nhỏ hơn, giá trị cần tính toán nhỏ hơn,...) và quá trình này tiếp tục cho đến khi bài toán được đưa về một cấp độ mà tại đó có thể giải được.

Thuật toán đệ quy tính giai thừa của một số tự nhiên

Nhập: Số tự nhiên n

Xuất: F(n) = n!

Thuật toán:

1. F ← 1
2. if n > 0 then

   F ← F(n–1)*n

   end if
3. Xuất: F

Đối với thuật toán đệ quy chúng ta cần lưu ý một số điểm sau đây:

1. Trong thuật toán đệ quy thường gồm hai phần: phần cơ sở và phần đệ quy:
   - Phần cơ sở nằm trong các trường hợp không cần thực hiện lại thuật toán (hay không có yêu cầu gọi đệ quy). Nếu thuật toán đệ quy không có phần này thì sẽ dẫn đến bị lặp vô hạn và sinh lỗi khi thi hành. Vì lý do này mà người ta còn gọi phần cơ sở là trường hợp dừng.
   - Phần đệ quy là phần trong thuật toán có yêu cầu gọi đệ quy, tức là yêu cầu thực hiện lại thuật toán. Trong phần đệ quy, yêu cầu gọi đệ quy thường được đặt trong một điều kiện kiểm tra việc gọi đệ quy.

2. Về mặt cài đặt, nếu như có sử dụng biến cục bộ trong thủ tục hay hàm đệ quy thì các biến này được tạo ra và được đặt trong vùng nhớ "STACK". Do đó, quá trình gọi đệ quy dễ gây ra tình trạng tràn STACK. Trong nhiều trường hợp có thể viết lại thuật toán đệ quy dưới dạng lặp.

II.2.5. Thuật giải heuristic

II.2.5.1. Thuật giải – Sự mở rộng khái niệm của thuật toán

Như đã trình bày ở phần trên, cách giải đệ quy không đáp ứng đầy đủ các yêu cầu đối với một thuật toán. Đó là một sự mở rộng khái niệm thuật toán. Trong quá trình nghiên cứu giải quyết bài toán, với các vấn đề đặt ra, người ta nhận thấy có những tình huống như sau:

- Có những bài toán đến nay vẫn chưa có một cách giải theo kiểu thuật toán được tìm ra và cũng không biết có tồn tại thuật toán hay không.
- Có những bài toán đã có thuật toán để giải nhưng không chấp nhận được vì thời gian giải theo thuật toán đó quá lâu hoặc các điều kiện cho thuật toán khó đáp ứng.
- Có những bài toán được giải theo cách giải vi phạm thuật toán nhưng vẫn được chấp nhận.

Từ những nhận xét trên, người ta thấy rằng cần phải có những mở rộng cho khái niệm thuật toán. Các tiêu chuẩn của thuật toán có thể mở rộng như: tính xác định, tính đúng đắn. Việc mở rộng tính xác định đối với thuật toán được thể hiện qua các thuật giải đệ quy và các giải thuật ngẫu nhiên. Tính đúng đắn bây giờ không còn bắt buộc với một số cách giải cho các bài toán, nhất là các cách giải gần đúng. Trong thực tiễn có nhiều trường hợp người ta chấp nhận các cách giải chỉ cho kết quả gần đúng nhưng ít phức tạp và hiệu quả.

Một trong số các thuật toán thường được đề cập đến và sử dụng trong khoa học trí tuệ nhân tạo là các cách giải theo kiểu heuristic. Chúng thường đơn giản, tự nhiên nhưng cho kết quả đúng hoặc gần đúng trong phạm vi cho phép.

Các cách giải chấp nhận được nhưng không hoàn toàn đáp ứng đầy đủ các tiêu chuẩn của thuật toán thường gọi là các thuật giải. Sự mở rộng này cho phép đưa ra các tiếp cận mới để giải quyết bài toán.

II.2.5.2. Thuật giải heuristic

Thuật giải heuristic là sự mở rộng khái niệm thuật toán. Nó thể hiện cách giải bài toán với các đặc tính sau:

- Thuật giải thường tìm được lời giải tốt (nhưng không chắc là tốt nhất).
- Thực hiện cách giải theo thuật giải heuristic thường dễ dàng và nhanh chóng hơn so với giải thuật tối ưu.
- Thuật giải heuristic thường thể hiện một cách hành động khá tự nhiên, gần gũi với cách suy nghĩ và hành động của con người.

Có nhiều cách tiếp cận cho việc thiết kế một thuật giải heuristic, trong đó có thể dựa vào một số nguyên lý cơ sở sau:

Nguyên lý vét cạn thông minh: Trong một bài toán tìm kiếm nào đó, khi không gian tìm kiếm lớn, ta thường tìm cách để giới hạn lại không gian hoặc thực hiện một kiểu dò tìm đặc biệt dựa vào đặc thù của bài toán để nhanh chóng tìm ra mục tiêu.

Nguyên lý tham lam: Lấy tiêu chuẩn tối ưu (trên phạm vi toàn bộ) của bài toán để làm tiêu chuẩn chọn lựa hành động cho phạm vi cục bộ của từng bước (hay từng giai đoạn) trong quá trình tìm kiếm lời giải.

Nguyên lý thứ tự: Thực hiện hành động dựa trên một cấu trúc thứ tự hợp lý của không gian khảo sát nhằm nhanh chóng đạt được một lời giải tốt.

Ngoài ra, trong việc thiết kế thuật giải heuristic người ta cũng thường đưa ra các *hàm heuristic*. Đó là các hàm mà giá trị của nó phụ thuộc vào các trạng thái hay tình huống trong quá trình thực hiện thuật giải. Nhờ việc so sánh giá trị hàm heuristic ta có thể chọn được cách hành động tương đối hợp lý trong từng bước của thuật giải.

Ví dụ: Xét bài toán tháp Hà Nội đã trình bày trong phần trước. Từ trạng thái 111, có thể chuyển đến hai trạng thái tiếp theo là 112 hoặc 113. Trong thuật giải heuristic "hướng đích", đĩa lớn nhất sẽ được ưu tiên chuyển về cọc 3. Chỉ làm được điều này khi cả hai đĩa nhỏ hơn đều ở cọc 2, nghĩa là xuất hiện bước chuyển 122 sang 322. Do đó sẽ chọn 111 →113 để có thể chuyển đĩa vừa sang cọc 2 (113 → 123) và trả lại đĩa bé về cọc 2 (123 → 122).

II.3. Câu hỏi và bài tập

1. Hãy mô tả quá trình giải quyết bài toán bằng máy tính.
2. Có những phương pháp nào để giải quyết bài toán bằng máy tính?
3. Hãy nêu các đặc trưng của thuật toán?
4. Sử dụng sơ đồ khối và mã giả mô tả thuật toán tìm số lớn nhất trong hai số A, B nhập vào từ bàn phím.
5. Vẽ sơ đồ khối và mã giả mô tả thuật toán cho phép tính tổng của một dãy số a[1], a[2],..., a[n] nhập vào từ bàn phím.
6. Vẽ sơ đồ khối mô tả thuật toán tìm giá trị nhỏ nhất trong ba số A, B, C nhập vào từ bàn phím.
7. Sử dụng mã giả để thể hiện thuật toán đếm số lần xuất hiện của một số x trong dãy số a[1], a[2],..., a[n].
8. Cho n số thực. Trình bày các thuật toán:

   a) Tìm giá trị nhỏ nhất trong n số đã cho.

   b) Tìm giá trị lớn nhất trong n số đã cho.
9. Cho k số nguyên. Trình bày các thuật toán:

   a) Tìm giá trị âm lớn nhất trong k số đã cho.

   b) Tìm giá trị dương nhỏ nhất trong k số đã cho.
10. Viết thuật toán đệ quy tính giá trị của dãy Fibonaci:

    $f_0 = f_1 = 1$; $f_n = f_{n-1} + f_{n-2}$ với mọi $n > 1$

PHẦN III. LẬP TRÌNH

III.1. Tổng quan về ngôn ngữ C

III.1.1. Lịch sử phát triển

Đầu những năm 1970, việc lập trình hệ thống vẫn dựa trên hợp ngữ (*Assembly*) nên công việc nặng nề, phức tạp và khó chuyển đổi chương trình giữa các hệ thống máy tính khác nhau. Điều này dẫn đến nhu cầu cần có một ngôn ngữ lập trình hệ thống bậc cao đồng thời có khả năng chuyển đổi dễ dàng từ hệ thống máy tính này sang hệ thống máy tính khác (còn gọi là tính *khả chuyển – portability*) để thay thế hợp ngữ.

Cũng vào thời gian đó, người ta muốn viết lại hệ điều hành Unix để cài đặt trên các hệ máy tính khác nhau, vì vậy cần có một ngôn ngữ lập trình hệ thống có tính khả chuyển cao để viết lại hệ điều hành Unix.

Ngôn ngữ C ra đời tại phòng thí nghiệm BELL của tập đoàn AT&T (Hoa Kỳ) do Brian W. Kernighan và Dennis Ritchie phát triển vào đầu những năm 1970 và hoàn thành vào năm 1972.

C được phát triển dựa trên nền các ngôn ngữ BCPL (*Basic Combined Programming Language*) và ngôn ngữ B. Cũng vì được phát triển dựa trên nền ngôn ngữ B nên ngôn ngữ mới được Brian W. Kernighan và Dennis Ritchie đặt tên là ngôn ngữ C như là sự tiếp nối ngôn ngữ B.

C có các đặc điểm là một ngôn ngữ lập trình hệ thống mạnh, khả chuyển, có tính linh hoạt cao và có thế mạnh trong xử lí các dạng dữ liệu số, văn bản, cơ sở dữ liệu. Vì vậy C thường được dùng để viết các chương trình hệ thống như hệ điều hành (ví dụ hệ điều hành Unix có 90% mã nguồn được viết bằng C, 10% còn lại viết bằng hợp ngữ) và các chương trình ứng dụng chuyên nghiệp có can thiệp tới dữ liệu ở mức thấp như xử lí văn bản, cơ sở dữ liệu, xử lí ảnh…

W. Kernighan và Dennis Ritchie công bố ngôn ngữ C trong lần xuất bản đầu của cuốn sách "*The C programming language*" (1978). Sau đó người ta đã bổ sung thêm những yếu tố và khả năng mới vào trong ngôn ngữ C (ví dụ như đưa thêm kiểu liệt kê **enum**, cho phép kiểu dữ liệu trả về bởi hàm là kiểu **void**, **struct** hoặc **union**… và đặc biệt là sự bổ sung các thư viện cho ngôn ngữ. Lúc đó đồng thời tồn tại nhiều

phiên bản khác nhau của ngôn ngữ C nhưng không tương thích với nhau. Điều này gây khó khăn cho việc trao đổi mã nguồn chương trình C viết trên các phiên bản ngôn ngữ C khác nhau (bạn sẽ rất khó đọc và hiểu chương trình của người khác và khi bạn muốn sửa nó thành chương trình của mình và dịch trên bộ dịch của mình thì sẽ tốn rất nhiều công sức), từ đó dẫn đến nhu cầu chuẩn hóa ngôn ngữ C.

Năm 1989, Viện Tiêu chuẩn Quốc gia của Hoa Kỳ (*American National Standards Institute* - ANSI) đã công bố phiên bản chuẩn hóa của ngôn ngữ C trong lần tái bản thứ hai cuốn sách "*The C programming language*" của các tác giả W. Kernighan và Dennis Ritchie. Từ đó đến nay phiên bản này vẫn thường được nhắc đến với tên gọi là *ANSI C, C chuẩn* hay C89 (vì được công bố năm 1989). ANSI C là sự kế thừa phiên bản đầu tiên của ngôn ngữ C (do K. Kernighan & D. Ritchie công bố năm 1978) có đưa thêm vào nhiều yếu tố mới và ANSI C đã quy định các thư viện chuẩn dùng cho ngôn ngữ C. Tất cả các phiên bản của ngôn ngữ C hiện đang sử dụng đều tuân theo các mô tả đã được nêu ra trong ANSI C, sự khác biệt nếu có thì chủ yếu nằm ở việc đưa thêm vào các thư viện bổ sung cho thư viện chuẩn của ngôn ngữ C.

Hiện nay cũng có nhiều phiên bản của ngôn ngữ C khác nhau và mỗi phiên bản này gắn liền với một bộ chương trình dịch cụ thể của ngôn ngữ C. Các bộ chương trình dịch phổ biến của ngôn ngữ C có thể kể tên như:

- Turbo C++ và Borland C++ của Borland Inc.
- MSC và VC của Microsoft Corp.
- GCC của GNU project.

...

Trong các trình biên dịch trên thì Turbo C++ là trình biên dịch rất quen thuộc và sẽ được chọn làm trình biên dịch cho các ví dụ sử dụng trong giáo trình này.

III.1.2. Các phần tử cơ bản của ngôn ngữ C

III.1.2.1. Tập kí tự

Chương trình nguồn của mọi ngôn ngữ lập trình đều được tạo nên từ các phần tử cơ bản là tập kí tự của ngôn ngữ đó. Các kí tự tổ hợp với nhau tạo thành các từ, các từ liên kết với nhau theo một quy tắc xác định (quy tắc đó gọi là cú pháp của ngôn ngữ) để tạo thành các câu lệnh. Từ các câu lệnh người ta sẽ tổ chức nên chương trình

Tập kí tự sử dụng trong ngôn ngữ lập trình C gồm có:

26 chữ cái hoa:	A B C ... X Y Z
26 chữ cái thường:	a b c ... x y z
10 chữ số:	0 1 2 3 4 5 6 7 8 9
Các kí hiệu toán học:	+ - * / = < >
Các dấu ngăn cách:	. ; , : space tab
Các dấu ngoặc:	( ) [ ] { }
Các kí hiệu đặc biệt:	_ ? $ & # ^ \ ! ' " ~ .v.v.

III.1.2.2. Từ khóa

Từ khóa (*Keyword*) là những từ có sẵn của ngôn ngữ và được sử dụng dành riêng cho những mục đích xác định.

Một số từ khóa hay dùng trong Turbo C++:

break	**case**	**char**	**const**	**continue**	**default**
do	**double**	**else**	**enum**	**float**	**for**
goto	**if**	**int**	**interrupt**	**long**	**return**
short	**signed**	**sizeof**	**static**	**struct**	**switch**
typedef	**union**	**unsigned**	**void**	**while**	

Chú ý: Tất cả các từ khóa trong C đều viết bằng chữ thường.

Các từ khóa trong C được sử dụng để:

- Đặt tên cho các kiểu dữ liệu: **int, float, double, char, struct, union**...
- Mô tả các lệnh, các cấu trúc điều khiển: **for, do, while, switch, case, if, else, break, continue**...

III.1.2.3. Định danh

Định danh (*Identifier* – hoặc còn gọi là *Tên*) là một dãy các kí tự dùng để gọi tên các đối tượng trong chương trình. Các đối tượng trong chương trình gồm có biến, hằng, hàm, kiểu dữ liệu..., ta sẽ làm quen ở những mục tiếp theo.

Định danh có thể được đặt tên sẵn bởi ngôn ngữ lập trình (đó chính là các từ khóa) hoặc do người lập trình đặt. Khi đặt tên cho định danh trong C, người lập trình cần tuân thủ các quy tắc sau:

1. Các kí tự được sử dụng trong các định danh của ngôn ngữ C chỉ được gồm có: chữ cái, chữ số và dấu gạch dưới "_" (*underscore*).
2. Bắt đầu của định danh phải là chữ cái hoặc dấu gạch dưới, không được bắt đầu định danh bằng chữ số.
3. Định danh do người lập trình đặt không được trùng với từ khóa.
4. Turbo C++ không giới hạn độ dài của định danh, nhưng chỉ 32 kí tự đầu của định danh được chương trình biên dịch sử dụng (khi định danh có độ dài lớn hơn 32 kí tự thì Turbo C++ sẽ tự động cắt bỏ, không xem xét các kí tự cuối bắt đầu từ kí tự thứ 33).

Ngoài các quy tắc bắt buộc trên, lập trình viên có quyền tùy ý đặt tên định danh trong chương trình của mình.

Một số ví dụ về định danh:

i, x, y, a, b, _function, _MY_CONSTANT, PI, gia_tri_1...

Ví dụ về định danh không hợp lệ:

1_a, 3d, 55x	bắt đầu bằng chữ số
so luong, ti le	có kí tự không hợp lệ (dấu cách – *space*) trong tên
int, char	trùng với từ khóa của ngôn ngữ C

Lưu ý:

- Đôi khi định danh do ta đặt gồm nhiều từ, khi đó để dễ đọc, ta nên tách các từ bằng cách sử dụng dấu gạch dưới. Ví dụ, định danh danh_sach_sinh_vien dễ đọc và dễ hiểu hơn so với định danh danhsachsinhvien.
- Định danh nên có tính chất gợi nhớ, ví dụ nếu ta muốn lưu trữ các thông tin về các sinh viên vào một biến nào đó thì biến đó nên được đặt tên là danh_sach_sinh_vien hay ds_sv... Đồng thời định danh danh_sach_sinh_vien chỉ nên dùng để đặt tên cho những đối tượng liên quan đến sinh viên chứ không nên đặt tên cho các đối tượng chứa thông tin về cán bộ viên chức. Việc đặt tên có tính gợi nhớ giúp cho người lập trình và những người khác đọc chương trình viết ra được dễ dàng hơn.

- Ngôn ngữ C phân biệt chữ cái thường và chữ cái hoa trong các định danh, tức là dinh_danh khác với Dinh_danh.
- Một thói quen của những người lập trình là các hằng thường được đặt tên bằng chữ hoa, còn các biến, hàm hay cấu trúc thì đặt tên bằng chữ thường. Nếu tên gồm nhiều từ thì ta nên phân cách các từ bằng dấu gạch dưới.

  Ví dụ:

Định danh	**Loại đối tượng**
HANG_SO_1, _CONSTANT_2	hằng
a, b, i, j, count	biến
nhap_du_lieu, tim_kiem, xu_li	hàm
sinh_vien, mat_hang	cấu trúc

III.1.2.4. Các kiểu dữ liệu

Kiểu dữ liệu là gì?

Dữ liệu là tài nguyên quan trọng nhất trong máy tính, song dữ liệu trong máy tính lại không phải tất cả đều giống nhau. Có dữ liệu là chữ viết, có dữ liệu là con số, lại có dữ liệu khác là hình ảnh, âm thanh... Ta nói rằng các dữ liệu đó thuộc các kiểu dữ liệu khác nhau.

Một cách hình thức, kiểu dữ liệu có thể được định nghĩa gồm hai điểm như sau:

- Một kiểu dữ liệu là một tập hợp các giá trị mà một dữ liệu thuộc kiểu dữ liệu đó có thể nhận được.
- Trên một kiểu dữ liệu ta xác định một số phép toán đối với các dữ liệu thuộc kiểu dữ liệu đó.

Ví dụ:

Trong ngôn ngữ C có kiểu dữ liệu **int**. Một dữ liệu thuộc kiểu dữ liệu **int** sẽ là một số nguyên (*integer*) và nó có thể nhận giá trị từ - 32,768 (-2^{15}) đến 32,767 ($2^{15} - 1$). Trên kiểu dữ liệu **int,** ngôn ngữ C định nghĩa các phép toán số học đối với số nguyên như:

Tên phép toán	Kí hiệu
Đảo dấu	-
Cộng	+
Trừ	-
Nhân	*
Chia lấy phần nguyên	/
Chia lấy phần dư	%
So sánh bằng	= =
So sánh lớn hơn	>
So sánh nhỏ hơn	<
...	

Trong máy tính, việc phân biệt kiểu dữ liệu là cần thiết vì qua kiểu dữ liệu, máy tính biết được đối tượng mà nó đang xử lí thuộc dạng nào, có cấu trúc ra sao, có thể thực hiện các phép xử lí nào đối với đối tượng đó, hay cần phải lưu trữ đối tượng đó như thế nào...

III.1.2.5. Hằng

Định nghĩa: Hằng (*constant*) là đại lượng có giá trị không đổi trong chương trình. Để giúp chương trình dịch nhận biết hằng ta cần nắm được cách biểu diễn hằng trong một chương trình C.

Biểu diễn hằng số nguyên

Trong ngôn ngữ C, một hằng số nguyên có thể được biểu diễn dưới những dạng sau:

- Dạng thập phân: Là cách viết giá trị số đó dưới hệ đếm cơ số 10 thông thường.
- Dạng thập lục phân: Ta viết giá trị số đó dưới dạng hệ đếm cơ số 16 và thêm tiền tố **0x** ở đầu.
- Dạng bát phân: Ta viết giá trị số đó dưới dạng hệ đếm cơ số 8 và thêm tiền tố **0** ở đầu.

Ví dụ:

Giá trị thập phân	Giá trị hệ bát phân	Giá trị hệ thập lục phân
2007	03727	0×7D7
396	0614	0×18C

Biểu diễn hằng số thực

Có hai cách biểu diễn hằng số thực:

– Dưới dạng số thực dấu phảy tĩnh.

– Dưới dạng số thực dấu phảy động.

Ví dụ:

Số thực dấu phảy tĩnh	Số thực dấu phảy động
3.14159	31.4159 E–1
123.456	12.3456 E+1 hoặc 1.23456 E+2

Biểu diễn hằng kí tự

Có hai cách biểu diễn hằng kí tự:

– Bằng kí hiệu của kí tự đó đặt giữa hai dấu nháy đơn.

– Bằng số thứ tự của kí tự đó trong bảng mã ASCII (lưu ý số thứ tự của một kí tự trong bảng mã ASCII là một số nguyên nên có một số cách biểu diễn).

Ví dụ:

Kí tự cần biểu diễn	Cách 1	Cách 2		
Chữ cái A	'A'	65	hoặc 0101	hoặc 0×41
Dấu nháy đơn '	'\''	39	hoặc 047	hoặc 0×27
Dấu nháy kép "	'\"'	34	hoặc 042	hoặc 0×22
Dấu gạch chéo ngược \	'\\'	92	hoặc 0134	hoặc 0×5c
Kí tự xuống dòng	'\n'			
Kí tự NUL	'\0'	0	hoặc 00	hoặc 0×0
Kí tự Tab	'\t'	9	hoặc 09	hoặc 0×9

Biểu diễn hằng xâu kí tự

Hằng xâu kí tự được biểu diễn bởi dãy các kí tự thành phần có trong xâu đó và được đặt trong cặp dấu nháy kép.

Ví dụ:

"ngon ngu lap trinh C", "tin hoc dai cuong"...

III.1.2.6. Biến

Định nghĩa: Biến (*variable*) là đại lượng mà giá trị có thể thay đổi trong chương trình.

Trong chương trình, hằng và biến được sử dụng để lưu trữ dữ liệu. Dữ liệu lưu trữ trong biến, hằng phải thuộc một kiểu dữ liệu nào đó.

Biến và hằng đều phải được đặt tên để khi cần thì có thể gọi đến. Tên biến và hằng được đặt theo quy tắc đặt tên cho định danh.

III.1.2.7. Hàm

Trong lập trình chúng ta rất hay phải tính toán giá trị của một số đại lượng thường gặp như sin(x), cos(x), căn bậc hai, lũy thừa, logarithm...

Ngôn ngữ C cung cấp cho người lập trình một công cụ dùng để tính toán giá trị các đại lượng đó mỗi khi cần trong chương trình, đó là các hàm.

Ví dụ: Để tính giá trị của đại lượng có tên là y và có giá trị $y = \sin(x)$, với x là một đại lượng nào đó, trong chương trình ta viết câu lệnh y = sin(x). Trong câu lệnh này sin() là một hàm, x là đối số của nó (hay còn gọi là tham số). Khi gặp hàm trong chương trình, ngôn ngữ C sẽ tự động gọi một đoạn chương trình tương ứng với hàm để tính toán giá trị của hàm đó rồi trả lại kết quả tính được cho chương trình.

Ngoài hàm sin(x), ngôn ngữ C còn có nhiều hàm khác hỗ trợ người lập trình tính toán giá trị của các đại lượng thường gặp như cos(x), sqrt(x) (để tính căn bậc hai của x), exp(x) (để tính lũy thừa e^x), log(x) (để tính logarithm cơ số e của x)...

Một số hàm toán học hay được sử dụng trong C

Hàm	Ý nghĩa	Kí hiệu toán học	Ví dụ
sqrt(x)	Căn bậc 2 của *x*	$\sqrt{x}$	sqrt(16.0) bằng 4.0
pow(x,y)	*x* mũ *y*	x^y	pow(2,3) bằng 8
exp(x)	*e* mũ *x*	e^x	exp(1.0) bằng 2.718
log(x)	logarithm tự nhiên (cơ số e) của x	$ln\ x$	log(2.718282) bằng
log10(x)	logarithm cơ số 10 của x	$log\ x$	log10(100) bằng 2
sin(x)	sin của x	$sin\ x$	sin(0.0) bằng 0.0
cos(x)	cosin của x	$cos\ x$	cos(0.0) bằng 1.0
tan(x)	tang của x	$tg\ x$	tan(0.0) bằng 0.0
ceil(x)	phần nguyên già của x, tức là số nguyên nhỏ nhất không nhỏ hơn x	$\lceil x \rceil$	ceil(2.5) bằng 3 ceil(-2.5) bằng –2
floor(x)	phần nguyên non của x, tức là số nguyên lớn nhất không lớn hơn x	$\lfloor x \rfloor$	floor(2.5) bằng 2 floor(-2.5) bằng –3

Khái niệm hàm sẽ được đề cập kĩ hơn trong mục III.6.

III.1.2.8. Biểu thức

Biểu thức là sự ghép nối các toán tử (*operator*) và các toán hạng (*operand*) t một quy tắc xác định.

Các toán hạng trong biểu thức có thể là biến, hằng, hàm hoặc một biểu thức k Bản thân một biến, hằng, hàm đứng độc lập cũng được coi là một biểu thức.

Các toán tử trong biểu thức rất đa dạng như cộng, trừ, nhân, chia, so sánh…

Biểu thức thường là sự thể hiện công thức tính toán giá trị một đại lượng nào đó

Ví dụ về biểu thức:

```
chieu_dai * chieu_rong * chieu_cao
```

Trong biểu thức trên, chieu_dai, chieu_rong, chieu_cao là các biến hoặc hằng, * l hiệu của toán tử nhân. Nếu chieu_dai, chieu_rong, chieu_cao là các biến (hoặc hằ

lưu trữ giá trị chiều dài, chiều rộng và chiều cao của một khối hộp chữ nhật thì biểu thức trên sẽ tính giá trị thể tích của khối hộp chữ nhật đó.

Vấn đề biểu thức sẽ tiếp tục được đề cập trong mục III.2.

III.1.2.9. Câu lệnh

Câu lệnh (*statement*) diễn tả một hoặc một nhóm các thao tác trong giải thuật. Chương trình được tạo thành từ dãy các câu lệnh.

Cuối mỗi câu lệnh đều có dấu chấm phẩy ';' để đánh dấu kết thúc câu lệnh cũng như để phân tách các câu lệnh với nhau.

Câu lệnh được chia thành hai nhóm chính:

- Nhóm các câu lệnh đơn: là những câu lệnh không chứa câu lệnh khác. Ví dụ: phép gán, phép cộng, phép trừ...
- Nhóm các câu lệnh phức: là những câu lệnh chứa câu lệnh khác trong nó. Ví dụ: khối lệnh, các cấu trúc lệnh rẽ nhánh, cấu trúc lệnh lặp...

Khối lệnh là một số các lệnh đơn được nhóm lại với nhau và đặt trong cặp dấu ngoặc nhọn { } để phân tách với các lệnh khác trong chương trình.

III.1.2.10. Chú thích

Để giúp việc đọc và hiểu chương trình viết ra được dễ dàng hơn, cần đưa vào các lời chú thích (*comment*). Lời chú thích là lời mô tả, giải thích vắn tắt cho một câu lệnh, một đoạn chương trình hoặc cả chương trình, nhờ đó người đọc có thể hiểu được ý đồ của người lập trình và công việc mà chương trình đang thực hiện.

Lời chú thích chỉ có tác dụng duy nhất là giúp chương trình viết ra dễ đọc và dễ hiểu hơn, nó không phải là câu lệnh và nó hoàn toàn không ảnh hưởng gì đến hoạt động của chương trình.

Khi gặp kí hiệu lời chú thích trong chương trình, trình biên dịch sẽ tự động bỏ qua không dịch phần nội dung nằm trong phạm vi của vùng chú thích đó.

Trong C và C++, có hai cách để viết lời chú thích:

- Dùng hai dấu sổ chéo liên tiếp // để kí hiệu toàn bộ vùng bắt đầu từ hai dấu sổ chéo liên tiếp đó đến cuối dòng là vùng chú thích. Ví dụ:

```
// khai bao 2 bien nguyen
int a, b;
a = 5; b = 3;    // khoi tao gia tri cho cac bien nay
```

Cách này thường dùng nếu đoạn chú thích ngắn, có thể viết đủ trên một dòng.

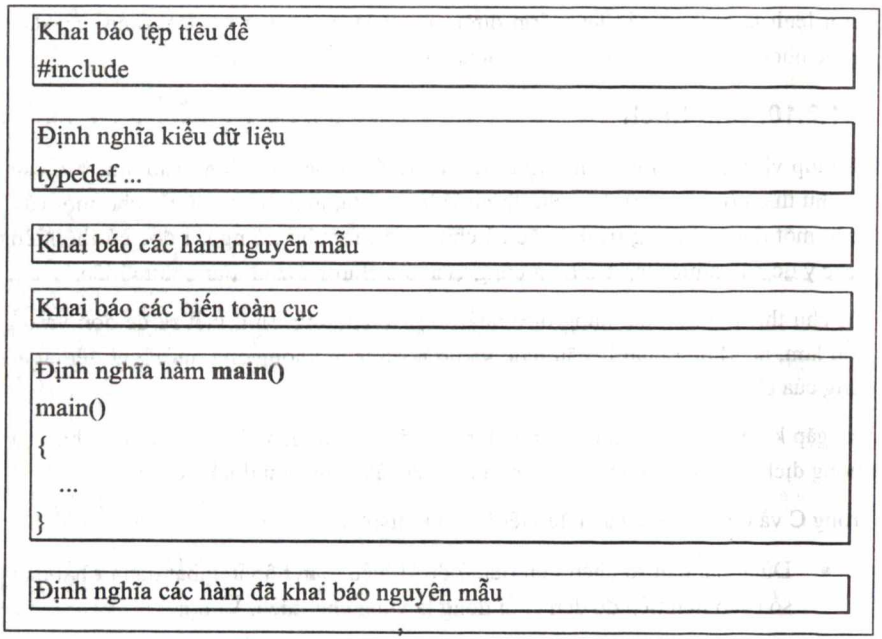
- Dùng hai cặp kí hiệu /* và */ để kí hiệu rằng toàn bộ vùng bắt đầu từ cặp kí hiệu /* kéo dài đến cặp kí hiệu */ là vùng chú thích. Ví dụ:

```
/* doan chuong trinh sau khai bao 2 bien nguyen va khoi tao gia tri cho 2
bien nguyen nay */
int a, b;
a = 5; b = 3;
```

Cách này thường dùng khi đoạn chú thích dài, phải viết trên nhiều dòng.

III.1.3. Cấu trúc cơ bản của một chương trình C

Về cơ bản, một chương trình viết bằng ngôn ngữ C có cấu trúc gồm 6 phần thứ tự như sau:



Hình III.1. Cấu trúc cơ bản của một chương trình C

- Phần 1: Phần khai báo các tệp tiêu đề. Phần này có chức năng thông báo cho chương trình dịch biết là chương trình có sử dụng những thư viện nào (mỗi tệp tiêu đề tương ứng với một thư viện).
- Phần 2: Định nghĩa các kiểu dữ liệu mới dùng cho cả chương trình.
- Phần 3: Phần khai báo các hàm nguyên mẫu. Phần này giúp cho chương trình dịch biết được những thông tin cơ bản (gồm tên hàm, danh sách các tham số và kiểu dữ liệu trả về) của các hàm sử dụng trong chương trình.
- Phần 4: Phần khai báo các biến toàn cục.
- Phần 5: Phần định nghĩa hàm **main( )**. Hàm **main( )** là một hàm đặc biệt trong C. Khi thực hiện, chương trình sẽ gọi hàm **main( )**, hay nói cách khác chương trình sẽ bắt đầu bằng việc thực hiện các lệnh trong hàm **main( )**. Trong hàm **main( )** ta mới gọi tới các hàm khác.
- Phần 6: Phần định nghĩa các hàm đã khai báo nguyên mẫu. Ở phần 3 ta đã khai báo nguyên mẫu (*prototype*) của các hàm, trong đó chỉ giới thiệu các thông tin cơ bản về hàm như tên hàm, danh sách các tham số và kiểu dữ liệu trả về. Nguyên mẫu hàm không cho ta biết cách thức cài đặt và hoạt động của các hàm. Ta sẽ làm việc đó ở phần định nghĩa các hàm.

Trong 6 phần trên, thì phần 5 định nghĩa hàm **main( )** bắt buộc phải có trong mọi chương trình C. Các phần khác có thể có hoặc không.

Dưới đây là ví dụ một chương trình viết trên ngôn ngữ C:

```
1.  // Chuong trinh sau se nhap vao tu ban phim 2 so nguyen
2.  // va hien thi ra man hinh tong, hieu tich cua 2 so nguyen vua nhap vao
3.  #include <stdio.h>
4.  #include <conio.h>
5.  void main()
6.  {
7.      // khai bao cac bien trong chuong trinh
8.      int a, b
9.      int tong, hieu, tich;
10.     // Nhap vao tu ban phim 2 so nguyen
11.     printf("\n Nhap vao so nguyen thu nhat: ");
12.     scanf("%d",&a);
```

```
13.     printf("\n Nhap vao so nguyen thu hai: ");
14.     scanf("%d",&b);
15.     // Tinh tong, hieu, tich cua 2 so vua nhap
16.     tong = a+b;
17.     hieu = a – b;
18.     tich = a*b;
19.     // Hien thi cac gia tri ra man hinh
20.     printf("\n Tong cua 2 so vua nhap la %d", tong);
21.     printf("\n Hieu cua 2 so vua nhap la %d", hieu);
22.     printf("\n Tich cua 2 so vua nhap la %d", tich);
23.     // Cho nguoi su dung an phim bat ki de ket thuc
24.     getch();
25. }
```

Trong chương trình trên chỉ có hai phần là khai báo các thư viện và định nghĩa hàm **main( )**. Các phần khai báo hàm nguyên mẫu, khai báo biến toàn cục và định nghĩa hàm nguyên mẫu không có trong chương trình này.

Các dòng 1, 2 là các dòng chú thích mô tả khái quát công việc chương trình sẽ thực hiện.

Dòng thứ 3 và thứ 4 là khai báo các tệp tiêu đề. Bởi vì trong chương trình ta sử dụng các hàm printf() (nằm trong thư viện **stdio** – *standard input/output*, thư viện chứa các hàm thực hiện các thao tác vào ra chuẩn) và getch() (nằm trong thư viện **conio** – *console input/output*, thư viện chứa các hàm thực hiện các thao tác vào ra qua bàn phím, màn hình...) nên ta phải khai báo với chương trình dịch gộp các thư viện đó vào chương trình. Nếu ta không gộp thư viện vào trong chương trình thì ta sẽ không thể sử dụng các hàm có trong thư viện đó.

Để gộp một thư viện vào trong chương trình (nhờ đó ta có thể sử dụng các hàm của thư viện đó), ta khai báo tệp tiêu đề tương ứng với thư viện đó ở đầu chương trình bằng chỉ thị có mẫu sau:

```
#include <tên_tệp_tiêu_đề>
```

Ví dụ: Để gộp thư viện conio vào chương trình ta dùng chỉ thị:

```
#include <conio.h>
```

Lưu ý: Các tệp tiêu đề có tên là tên của thư viện, có phần mở rộng là .h (viết tắt của từ *header*).

Các dòng tiếp theo (từ dòng thứ 5 đến dòng thứ 25) là phần định nghĩa hàm **main()**, trong đó các dòng 7, 10, 15, 19, 23 là các dòng chú thích mô tả công việc mà các câu lệnh sau đó sẽ thực hiện.

III.1.4. Biên dịch chương trình C

III.1.4.1. Trình biên dịch Turbo C++

Một chương trình sau khi viết ra phải được biên dịch thành mã máy (tức là chuyển các câu lệnh viết bằng ngôn ngữ lập trình thành các câu lệnh tương ứng của ngôn ngữ máy) thì mới có thể thực thi được. Công việc biên dịch được thực hiện bởi trình biên dịch (*compiler*).

Hiện có nhiều trình biên dịch ngôn ngữ C khác nhau như Turbo C++ của hãng Borland Inc, MSC của Microsoft Corp, GCC do GNU project phát triển, hay Lattice C của Lattice, Dev-C++ của Bloodshed Software... Tuy vậy, đối với đa phần người bắt đầu học C ở Việt Nam thì Turbo C++ là trình biên dịch ngôn ngữ C quen thuộc.

Lưu ý: Turbo C++ có khả năng biên dịch chương trình viết bằng cả ngôn ngữ C và C++.

Turbo C++ cũng có nhiều phiên bản khác nhau, nhưng để bắt đầu học và thực hành C thì Turbo C++ 3.0 là thích hợp vì nó có đặc điểm là gọn nhẹ, đủ tính năng và dễ sử dụng.

III.1.4.2. Cài đặt và sử dụng Turbo C++ 3.0

Để sử dụng Turbo C++ 3.0 ta cần phải cài đặt nó lên máy. Quá trình cài đặt thực hiện theo các bước sau:

B1: Chuẩn bị đĩa chứa bộ cài của Turbo C++ 3.0, kích thước của bộ cài khoảng 4 MB. Hãy copy bộ cài này vào máy của bạn, giả sử vào thư mục *C:\TC_Setup*.

B2: Tìm đến thư mục chứa bộ cài Turbo C++ 3.0 (như giả sử ở trên là *C:\TC_Setup*) và kích hoạt file *INSTALL.EXE* để chạy chương trình cài đặt Turbo C++ 3.0. Chương trình cài đặt Turbo C++ 3.0 ban đầu sẽ yêu cầu bạn chỉ ra ổ đĩa trên đó chứa bộ cài Turbo C++ 3.0.

Enter the SOURCE drive to use:

Hãy nhập vào tên ổ đĩa, chẳng hạn C (ta để bộ cài Turbo C++ 3.0 ở thư mục *C:\TC_Setup*).

B3: Sau đó chương trình yêu cầu nhập vào đường dẫn tới thư mục chứa các file của Turbo C++ 3.0.

Enter the SOURCE Path:

Thông thường chương trình sẽ tự tìm, chỉ cần ấn Enter để chuyển sang bước tiếp theo.

B4: Ở bước 4, cần xác định thư mục cài đặt. Thư mục này sẽ chứa các file của Turbo C++ 3.0 để sử dụng sau này.

Directories... [C:\TC]
Option... [IDE CMD LIB CLASS BGI HELP EXMPL]
Start Installation

Thư mục cài đặt mặc định sẽ là *\TC* nằm trên thư mục gốc của ổ đĩa chứa bộ cài. Nếu muốn thay đổi thư mục cài đặt thì hãy dùng các phím ↑ và ↓ để di chuyển hộp sáng đến phần Directories, gõ Enter và nhập vào đường dẫn mới, sau đó ấn phím Esc để trở về.

Dùng các phím ↑ và ↓ để di chuyển hộp sáng đến phần Start Installation và ấn Enter. Chương trình sẽ tự động thực hiện và hoàn tất quá trình cài đặt.

Lưu ý: Có thể copy toàn bộ thư mục đã cài đặt của Turbo C++ 3.0 về máy và sử dụng, nhưng phải cho Turbo C++ biết đường dẫn tới các tệp tiêu đề và các tệp thư viện bằng cách vào menu *Option*, chọn *Directories*.

III.1.4.3. Sử dụng môi trường Turbo C++ 3.0

Sau khi cài đặt xong, tìm đến thư mục *BIN* trong thư mục cài đặt và chạy file *TC.EXE* để khởi động Turbo C++ 3.0.

Sau khi khởi động Turbo C++ 3.0, sẽ xuất hiện màn hình làm việc của Turbo C++ 3.0.

Bạn dùng chuột di chuyển đến menu File (hoặc ấn Alt-F), sau đó chọn mục New để mở cửa sổ soạn thảo mới.

Giờ hãy gõ vào toàn bộ chương trình viết bằng ngôn ngữ C của bạn lên cửa sổ soạn thảo này. Ấn phím F2 để lưu trữ tệp chương trình nguồn trên máy, một cửa sổ cất giữa tệp sẽ hiện ra yêu cầu bạn nhập vào tên mới cho tệp chương trình nguồn (tên mặc định sẽ là *NONAME.CPP*). Hãy đặt một tên cho tệp rồi chọn OK để lưu tệp chương trình nguồn lại.

Cuối cùng là ấn phím F9 để biên dịch chương trình viết ra. Nếu chương trình có lỗi thì Turbo C++ 3.0 sẽ báo lỗi và bạn phải sửa lại đến khi không còn lỗi. Nếu chương trình không có lỗi thì Turbo C++ 3.0 sẽ thông báo biên dịch thành công và bạn có thể ấn Ctrl-F9 để chạy chương trình đã biên dịch.

Với chương trình ví dụ ở phần trước, sau khi biên dịch và thực hiện bạn sẽ thấy trên màn hình kết quả như sau:

```
Nhap vao so nguyen thu nhat: 523
Nhap vao so nguyen thu hai: 257
Tong cua 2 so vua nhap la 780
Hieu cua 2 so vua nhap la 266
Tich cua 2 so vua nhap la 134411
```

III.2. Kiểu dữ liệu và biểu thức trong C

III.2.1. Các kiểu dữ liệu chuẩn trong C

Kiểu dữ liệu	Ý nghĩa	Kích thước	Miền biểu diễn
unsigned char	Kí tự	1 byte	0 ÷ 255
char	Kí tự	1 byte	-128 ÷ 127
unsigned int	Số nguyên không dấu	2 byte	0 ÷ 65,535
short int	Số nguyên có dấu	2 byte	-32,768 ÷ 32,767
int	Số nguyên có dấu	2 byte	- 32,768 ÷ 32,767
unsigned long	Số nguyên không dấu	4 byte	0 ÷ 4,294,967,295
long	Số nguyên có dấu	4 byte	-2,147,483,648 ÷ 2,147,483,647
float	Số thực dấu phảy động, độ chính xác đơn	4 byte	± 3.4E-38 ÷ ± 3.4E+38
double	Số thực dấu phảy động, độ chính xác kép	8 byte	± 1.7E-308 ÷ ± 1.7E+308
long double	Số thực dấu phảy động	10 byte	± 3.4E-4932 ÷ ± 1.1E+4932

Khai báo và sử dụng biến, hằng

Khai báo và sử dụng biến

Một biến trước khi sử dụng phải được khai báo. Cú pháp khai báo:

kiểu_dữ_liệu tên_biến;

Ví dụ:

```
float x;      // biến kiểu thực
float y;      // biến kiểu thực
double z;     // biến kiểu thực
int i;        // biến kiểu nguyên
int j;        // biến kiểu nguyên
```

Nếu các biến thuộc cùng kiểu dữ liệu thì C cho phép khai báo chúng trên cùng một dòng, ngăn cách với nhau bởi dấu phảy:

kiểu_dữ_liệu danh_sách_tên_biến;

Ví dụ:

```
float x, y;
int i, j;
```

Sau khi khai báo, biến có thể nhận giá trị thuộc kiểu dữ liệu đã khai báo. Chúng ta có thể khởi tạo giá trị đầu cho biến nếu muốn với cú pháp:

kiểu_dữ_liệu tên_biến = giá_trị_đầu;

Ví dụ:

```
int a = 3;    // sau lenh nay bien a se co gia tri bang 3
float x = 5.0, y = 2.6;// sau lenh nay x co gia tri 5.0, y co gia tri 2.6
```

Biến dùng để lưu giữ giá trị, dùng làm toán hạng trong biểu thức, làm tham số cho hàm, làm biến chỉ số cho các cấu trúc lặp (**for, while, do while**), làm biểu thức điều kiện trong các cấu trúc rẽ nhánh (**if, switch**)...

Khai báo hằng

Có hai cách để khai báo hằng trong C là dùng chỉ thị **#define** hoặc khai báo với từ khóa **const**.

Dùng chỉ thị **#define**

Cú pháp khai báo:

#define tên_hằng giá_trị

Lưu ý: Không có dấu chấm phảy ở cuối dòng chỉ thị.

Với cách khai báo này, mỗi khi chương trình dịch gặp tên_hằng trong chương trình, nó sẽ tự động thay thế bằng giá_trị. Ở đây kiểu dữ liệu của hằng tự động được chương trình dịch xác định dựa theo nội dung của giá_trị.

Ví dụ:

```
#define MAX_SINH_VIEN  50                  // hằng kiểu số nguyên
#define CNTT "Cong nghe thong tin"         // hằng kiểu xâu kí tự (string)
#define DIEM_CHUAN  23.5                   // hằng kiểu số thực
```

Dùng từ khóa **const** để khai báo với cú pháp:

const kiểu_dữ_liệu tên_hằng = giá_trị;

Khai báo này giống với khai báo biến có khởi tạo giá trị đầu, tuy nhiên cần lưu ý:

- Do có từ khóa **const** ở đầu nên giá trị của đối tượng tên_hằng sẽ không được phép thay đổi trong chương trình. Những lệnh nhằm làm thay đổi giá trị của tên_hằng trong chương trình sẽ dẫn tới lỗi biên dịch.
- Trong khai báo biến thông thường, người lập trình có thể khởi tạo giá trị cho biến ngay từ khi khai báo hoặc không khởi tạo cũng được. Nhưng trong khai báo hằng, giá trị của tất cả các hằng cần được xác định ngay trong lệnh khai báo.

Các khai báo hằng ở ví dụ trước giờ có thể viết lại theo cách khác như sau:

```
const int  MAX_SINH_VIEN  = 50;
const char  CNTT[20] = "Cong nghe thong tin";
const float  DIEM_CHUAN = 23.5;
```

III.2.2. Các biểu thức

Biểu thức số học

Biểu thức số học là biểu thức mà giá trị của nó là các đại lượng số học (số nguyên, số thực).

Trong biểu thức số học, các toán tử là các phép toán số học (cộng, trừ, nhân, chia...), các toán hạng là các đại lượng số học. Ví dụ (giả sử a, b, c là các số thực):

3 * 3.7, 8 + 6/3, a + b – c...

Biểu thức logic

Biểu thức logic là biểu thức mà giá trị của nó là các giá trị logic, tức là một trong hai giá trị: Đúng (*TRUE*) hoặc Sai (*FALSE*).

Ngôn ngữ C coi các giá trị nguyên khác 0 (ví dụ 1, –2, –5) là giá trị logic Đúng (TRUE), giá trị 0 là giá trị logic Sai (FALSE).

Các phép toán logic gồm có

– AND (VÀ logic, trong ngôn ngữ C được kí hiệu là &&)

– OR (HOẶC logic, trong ngôn ngữ C được kí hiệu là ||)

– NOT (PHỦ ĐỊNH, trong ngôn ngữ C kí hiệu là !)

Biểu thức quan hệ

Biểu thức quan hệ là những biểu thức trong đó có sử dụng các toán tử quan hệ so sánh như lớn hơn, nhỏ hơn, bằng nhau, khác nhau... Biểu thức quan hệ cũng chỉ có thể nhận giá trị là một trong hai giá trị Đúng (TRUE) hoặc Sai (FALSE). Ví dụ về biểu thức quan hệ:

```
5 > 7                   // có giá trị logic là sai, FALSE
9 != 10                 // có giá trị logic là đúng, TRUE
2 >= 2                  // có giá trị logic là đúng, TRUE
a > b                   // giả sử a, b là 2 biến kiểu int
a+1 > a                 // có giá trị đúng, TRUE
```

Ví dụ về biểu thức logic:

```
(5 > 7)&&(9!=10)        // có giá trị logic là sai, FALSE
0 || 1                  // có giá trị logic là đúng, TRUE
(5 > 7)||(9!=10)        // có giá trị logic là đúng, TRUE
0                       // có giá trị logic là sai, FALSE
!0                      // phủ định của 0, có giá trị logic là đúng, TRUE
3                       // có giá trị logic là đúng, TRUE
!3                      // phủ định của 3, có giá trị logic là sai, FALSE
(a > b)&&( a < b)       // Có giá trị sai, FALSE. Giả sử a, b là 2 biến kiểu int
```

Sử dụng biểu thức

Trong chương trình, biểu thức được sử dụng cho các mục đích sau:

- Làm vế phải của lệnh gán (sẽ đề cập ở mục sau).
- Làm toán hạng trong các biểu thức khác.
- Làm tham số thực trong lời gọi hàm.
- Làm chỉ số trong các cấu trúc lặp **for**, **while**, **do while**.
- Làm biểu thức kiểm tra trong các cấu trúc rẽ nhánh **if**, **switch**.

III.2.3. Các phép toán

Các phép toán trong C được chia thành các nhóm phép toán cơ bản sau: nhóm các phép toán số học, nhóm các phép toán thao tác trên bit, nhóm các phép toán quan hệ, nhóm các phép toán logic. Ngoài ra C còn cung cấp một số phép toán khác nữa như phép gán, phép lấy địa chỉ...

III.2.3.1. Phép toán số học

Các phép toán số học (*Arithmetic operators*) gồm có:

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ
–	Phép đổi dấu	Số thực hoặc số nguyên	int a, b; –12; –a; –25.6;
+	Phép toán cộng	Số thực hoặc số nguyên	float x, y; 5 + 8; a + x; 3.6 + 2.9;
–	Phép toán trừ	Số thực hoặc số nguyên	3 – 1.6; a – 5;
*	Phép toán nhân	Số thực hoặc số nguyên	a * b; b * y; 2.6 * 1.7;
/	Phép toán chia	Số thực hoặc số nguyên	10.0/3.0; (bằng 3.33...) 10/3.0; (bằng 3.33...) 10.0/3; (bằng 3.33...)

/	Phép chia lấy phần nguyên	Giữa hai số nguyên	10/3;	(bằng 3)
%	Phép chia lấy phần dư	Giữa hai số nguyên	10%3;	(bằng 1)

Các phép toán trên bit

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ
&	Phép VÀ nhị phân	Hai số nhị phân	0 & 0 (có giá trị 0) 0 & 1 (có giá trị 0) 1 & 0 (có giá trị 0) 1 & 1 (có giá trị 1) 101 & 110 (có giá trị 100)
|	Phép HOẶC nhị phân	Hai số nhị phân	0 | 0 (có giá trị 0) 0 | 1 (có giá trị 1) 1 | 0 (có giá trị 1) 1 | 1 (có giá trị 1) 101 | 110 (có giá trị 111)
^	Phép HOẶC LOẠI TRỪ nhị phân	Hai số nhị phân	0 ^ 0 (có giá trị 0) 0 ^1 (có giá trị 1) 1 ^ 0 (có giá trị 1) 1 ^ 1 (có giá trị 0) 101 ^ 110 (có giá trị 011)
<<	Phép DỊCH TRÁI nhị phân	Số nhị phân	a << n (có giá trị $a*2^n$) 101 << 2 (có giá trị 10100)
>>	Phép DỊCH PHẢI nhị phân	Số nhị phân	a >> n (có giá trị $a/2^n$) 101 >> 2 (có giá trị 1)
~	Phép ĐẢO BIT nhị phân (lấy Bù 1)	Số nhị phân	~ 0 (có giá trị 1) ~ 1 (có giá trị 0) ~ 110 (có giá trị 001)

III.2.3.2. Phép toán quan hệ

Các phép toán quan hệ (*Comparison operators*) gồm có:

Toán tử	Ý nghĩa	Ví dụ
>	So sánh lớn hơn giữa hai số nguyên hoặc thực.	2 > 3 (có giá trị 0) 6 > 4 (có giá trị 1) a > b
>=	So sánh lớn hơn hoặc bằng giữa hai số nguyên hoặc thực.	6 >= 4 (có giá trị 1) x >= a
<	So sánh nhỏ hơn giữa hai số nguyên hoặc thực.	5 < 3 (có giá trị 0)
<=	So sánh nhỏ hơn hoặc bằng giữa hai số nguyên hoặc thực.	5 <= 5 (có giá trị 1) 2 <= 9 (có giá trị 1)
==	So sánh bằng nhau giữa hai số nguyên hoặc thực.	3 == 4 (có giá trị 0) a == b
!=	So sánh không bằng (so sánh khác) giữa hai số nguyên hoặc thực.	5 != 6 (có giá trị 1) 6 != 6 (có giá trị 0)

III.2.3.3. Các phép toán logic

Các phép toán logic (*Logical operators*) gồm có:

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ
&&	Phép VÀ LOGIC. Biểu thức VÀ LOGIC bằng 1 khi và chỉ khi cả hai toán hạng đều bằng 1	Hai biểu thức logic	3<5 && 4<6 (có giá trị 1) 2<1 && 2<3 (có giá trị 0) a > b && c < d
||	Phép HOẶC LOGIC. Biểu thức HOẶC LOGIC bằng 0 khi và chỉ khi cả hai toán hạng bằng 0.	Hai biểu thức logic	6 || 0 (có giá trị 1) 3<2 || 3<3 (có giá trị 1) x >= a || x == 0
!	Phép PHỦ ĐỊNH LOGIC một ngôi. Biểu thức PHỦ ĐỊNH LOGIC có giá trị bằng 1 nếu toán hạng bằng 0 và có giá trị bằng 0 nếu toán hạng bằng 1	Biểu thức logic	!3 (có giá trị 0) !(2>5) (có giá trị 1)

III.2.3.4. Phép toán gán

Phép toán gán có dạng:

tên_biến = biểu_thức;

Phép toán gán có chức năng lấy giá trị của biểu_thức gán cho tên_biến. Dấu = là kí hiệu cho toán tử gán.

Ví dụ:

```
int a, b, c;
a = 3;
b = a + 5;
c = a * b;
```

Sau đoạn lệnh trên, biến a có giá trị là 3, b có giá trị là 8 và c có giá trị là 24.

Trong phép toán gán nếu ta bỏ dấu ; ở cuối đi thì sẽ thu được biểu thức gán. Biểu thức gán là biểu thức có dạng:

tên_biến = biểu_thức

Biểu thức gán là biểu thức nên cũng có giá trị. Giá trị của biểu thức gán bằng giá trị của biểu_thức, do đó ta có thể gán giá trị của biểu thức gán cho một biến khác hoặc sử dụng như một biểu thức bình thường. Ví dụ:

```
int a, b, c;
a = b = 2007;
c = (a = 20) * (b = 30);
```

Trong câu lệnh thứ hai, ta đã gán giá trị 2007 cho biến *b*, sau đó gán giá trị của biểu thức *b* = 2007 cho biến a. Giá trị của biểu thức *b* = 2007 là 2007, do đó kết thúc câu lệnh này ta có *a* bằng 2007, *b* bằng 2007.

Trong câu lệnh thứ ba, ta gán giá trị 20 cho a, gán giá trị 30 cho b. Sau đó ta tính giá trị của biểu thức tích (a = 20) * (b = 30) từ giá trị các biểu thức con *a* = 20 (có giá trị là 20) và *b* = 30 (có giá trị là 30). Cuối cùng ta gán giá trị của biểu thức tích thu được (600) cho biến *c*.

Phép toán gán thu gọn

Xét lệnh gán sau:

```
x = x + y;
```

Lệnh gán này sẽ tăng giá trị của biến x thêm một lượng có giá trị bằng giá trị của y. Trong C ta có thể viết lại lệnh này một cách gọn hơn mà thu được kết quả tương đương:

```
x + = y;
```

Dạng lệnh gán thu gọn này còn áp dụng được với các phép toán khác nữa.

Lệnh gán thông thường	Lệnh gán thu gọn
x = x + y	x + = y
x = x – y	x – = y
x = x * y	x * = y
x = x / y	x / = y
x = x % y	x % = y
x = x >> y	x >> = y
x = x << y	x << = y
x = x & y	x & = y
x = x | y	x | = y
x = x ^ y	x^ = y

III.2.4. Thứ tự ưu tiên các phép toán

Khái niệm thứ tự ưu tiên (operator precedence) của phép toán

Thứ tự ưu tiên của các phép toán dùng để xác định trật tự kết hợp các toán hạng với các toán tử khi tính toán giá trị của biểu thức.

Bảng thứ tự ưu tiên của các phép toán trong C

Mức	Các toán tử	Trật tự kết hợp
1	() [] . -> ++ (hậu tố) – (hậu tố)	——>
2	! ~ ++ (tiền tố) -- (tiền tố) - * & sizeof	<——
3	* / %	——>
4	+ –	——>

5	<< >>	——>
6	< <= > >=	——>
7	== !=	——>
8	&	——>
9	^	——>
10	|	——>
11	&&	——>
12	||	——>
13	?:	<——
14	= += –=	<——

Ghi chú: ——> Trật tự kết hợp từ trái qua phải.

<—— Trật tự kết hợp từ phải qua trái.

Nguyên tắc xác định trật tự thực hiện các phép toán

i. Biểu thức con trong ngoặc được tính toán trước các phép toán khác.

ii. Phép toán một ngôi đứng bên trái toán hạng được kết hợp với toán hạng đi liền nó.

iii. Nếu toán hạng đứng cạnh hai toán tử thì có hai khả năng là:

a. Nếu hai toán tử có độ ưu tiên khác nhau thì toán tử nào có độ ưu tiên cao hơn sẽ kết hợp với toán hạng.

b. Nếu hai toán tử cùng độ ưu tiên thì dựa vào trật tự kết hợp của các toán tử để xác định toán tử được kết hợp với toán hạng.

III.2.5. Một số toán tử đặc trưng trong C

Các phép toán tăng giảm một đơn vị

Trong lập trình chúng ta thường xuyên gặp những câu lệnh tăng (hoặc giảm) giá trị của một biến thêm (đi) một đơn vị. Để làm điều đó chúng ta dùng lệnh sau:

<tên biến> = <tên biến> + 1;

<tên biến> = <tên biến> – 1;

Ta cũng có thể tăng (hoặc giảm) giá trị của một biến bằng cách sử dụng hai phép toán đặc biệt của C là phép toán ++ và phép toán --. Phép toán ++ sẽ tăng giá trị của biến thêm 1 đơn vị, phép toán -- sẽ giảm giá trị của biến đi 1 đơn vị. Ví dụ:

```
int a = 5;
float x = 10;
a ++;  // lệnh này tương đương với a = a + 1;
x --;  // tương đương với x = x – 1;
```

Phép toán tăng, giảm một đơn vị ở ví dụ trên là dạng hậu tố (vì phép toán đứng sau toán hạng). Ngoài ra còn có dạng tiền tố của phép toán tăng, giảm một đơn vị. Trong dạng tiền tố, ta thay đổi giá trị của biến trước khi sử dụng biến đó để tính toán giá trị của biểu thức. Trong dạng hậu tố, ta tính toán giá trị của biểu thức bằng giá trị ban đầu của biến, sau đó mới thay đổi giá trị của biến.

Ví dụ:

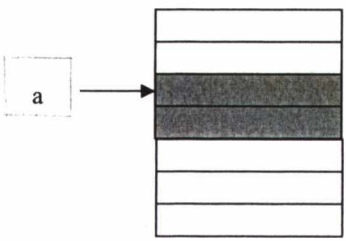
```
int a, b, c;
a = 3;          // a bằng 3;
b = a++;        // dạng hậu tố. b bằng 3; a bằng 4;
c = ++b;        // dạng tiền tố. b bằng 4, c bằng 4;
```

Sau khi thực hiện đoạn chương trình trên, ta có *a*, *b* và *c* đều có giá trị bằng 4.

Phép toán lấy địa chỉ biến (&)

Một biến thực chất là một vùng nhớ được đặt tên (là tên của biến) trên bộ nhớ của máy tính. Mọi ô nhớ trên bộ nhớ máy tính đều được đánh địa chỉ. Do đó mọi biến đều có địa chỉ.

Địa chỉ của một biến được định nghĩa là địa chỉ của ô nhớ đầu tiên trong vùng nhớ dành cho biến đó. Hình III.2 minh họa một biến tên là *a*, kiểu dữ liệu **int** được lưu trữ trong bộ nhớ tại hai ô nhớ. Khi đó địa chỉ của biến *a* sẽ là địa chỉ ô nhớ thứ nhất trong hai ô nhớ này.



Hình III.2. Địa chỉ của biến

Trong C để xác định địa chỉ của một biến ta sử dụng toán tử một ngôi & đặt trước tên biến, cú pháp là:

& <tên biến>;

Ví dụ: &a;

Phép toán chuyển đổi kiểu bắt buộc

Chuyển đổi kiểu là chuyển kiểu dữ liệu của một biến từ kiểu dữ liệu này sang kiểu dữ liệu khác. Cú pháp của lệnh chuyển kiểu dữ liệu là như sau:

(<kiểu dữ liệu mới>) <biểu thức>;

Có những sự chuyển đổi được thực hiện hết sức tự nhiên, không có khó khăn gì, thậm chí đôi khi chương trình dịch sẽ tự động chuyển đổi kiểu hộ cho ta, ví dụ chuyển một dữ liệu kiểu số nguyên **int** sang một số nguyên kiểu **long int**, hay từ một số **long int** sang một số thực **float**... Đó là vì một số nguyên kiểu **int** thực ra cũng là một số nguyên kiểu **long int**, một số nguyên kiểu **long int** cũng chính là một số thực kiểu **float**, một số thực kiểu **float** cũng là một số thực kiểu **double**.

Tuy nhiên điều ngược lại thì chưa chắc, ví dụ số nguyên **long int** 50,000 không phải là một số nguyên kiểu **int** vì phạm vi biểu diễn của kiểu **int** là từ (–32,768 đến 32,767). Khi đó nếu phải chuyển kiểu dữ liệu thì phải cẩn thận nếu không sẽ bị mất dữ liệu. Ví dụ, một số thực **float** khi chuyển sang kiểu số nguyên **int** sẽ bị loại bỏ phần thập phân, còn một số nguyên kiểu **long int** khi chuyển sang kiểu **int** sẽ nhiều khả năng thu được một giá trị xa lạ. Ví dụ: 1,193,046 (0x123456) là một số kiểu **long int**, khi chuyển sang kiểu **int** sẽ thu được 13,398 (0x3456). Đoạn chương trình sau minh họa điều đó:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    long int li;
    int i;
    float f;
    clrscr();
    li = 0x123456;
    f = 123.456;
    i = (int) li;
    printf("\n li = %ld",li);
```

```
    printf("\n i  = %d",i);
    i = (int) f;
    printf("\n f  = %f",f);
    printf("\n i  = %d",i);
    getch();
}
```

Kết quả:

```
li = 1193046
i = 13398
f = 123.456001
i = 123
```

C hỗ trợ chuyển kiểu tự động trong những trường hợp sau:

char → int → long int → float → double → long double

Khả năng chuyển kiểu dữ liệu là một điểm mạnh của C, tạo sự linh hoạt cho biến trong chương trình. Tuy nhiên với những người mới bắt đầu lập trình thì không nên chuyển kiểu dữ liệu trong chương trình vì có thể sinh ra những kết quả không như ý nếu bạn chưa có kinh nghiệm. Ngoài ra hình dung trước kiểu dữ liệu cho một biến và duy trì kiểu dữ liệu đó trong suốt chương trình cũng là một thói quen lập trình tốt.

Biểu thức điều kiện

Là biểu thức có dạng:

biểu_thức_1 ? biểu_thức_2 : biểu_thức_3

Giá trị của biểu thức điều kiện sẽ là giá trị của biểu_thức_2 nếu biểu_thức_1 có giá trị khác 0 (tương ứng với giá trị logic TRUE), trái lại giá trị của biểu thức điều kiện sẽ là giá trị của biểu_thức_3 nếu biểu_thức_1 có giá trị bằng 0 (tương ứng với giá trị logic FALSE).

Ví dụ sau sẽ cho ta xác định được giá trị nhỏ nhất của 2 số nhờ sử dụng biểu thức điều kiện:

```
float x, y, z;          // khai báo biến
x = 3.8; y = 2.6;       // gán giá trị cho các biến x, y
```

```
z = (x<y) ? x : y;          // z sẽ có giá trị bằng giá trị
                            // nhỏ nhất trong 2 số x và y
```

Lệnh dãy

Lệnh dãy là lệnh gồm một dãy các biểu thức phân cách nhau bằng dấu phẩy và kết thúc lệnh là dấu chấm phẩy. Nó có dạng:

biểu_thức_1, biểu_thức_2, ..., biểu_thức_n;

Trong lệnh dãy các biểu thức được tính toán độc lập với nhau.

III.3. Cấu trúc lập trình trong C

III.3.1. Vào/ra

III.3.1.1. Các lệnh vào ra dữ liệu với các biến (printf, scanf)

Để vào ra dữ liệu, C cung cấp 2 hàm vào ra cơ bản là printf() và scanf(). Muốn sử dụng 2 hàm printf() và scanf() ta cần khai báo tệp tiêu đề stdio.h.

Hàm printf()

Cú pháp sử dụng hàm printf ( ):

printf(xâu_định_dạng [,danh_sách_tham_số]);

Hàm printf() được dùng để hiển thị ra màn hình các loại dữ liệu cơ bản như số, kí tự và xâu kí tự cùng một số hiệu ứng hiển thị đặc biệt.

xâu_định_dạng là xâu điều khiển cách thức hiển thị dữ liệu trên thiết bị ra chuẩn (màn hình máy tính). Trong xâu_định_dạng có chứa:

- Các kí tự thông thường, chúng sẽ được hiển thị ra màn hình bình thường.
- Các nhóm kí tự định dạng dùng để xác định quy cách hiển thị các tham số trong phần danh_sách_tham_số.
- Các kí tự điều khiển dùng để tạo các hiệu ứng hiển thị đặc biệt như xuống dòng ('\n') hay sang trang ('\f')...

Phần danh_sách_tham_số là các giá trị biến, hằng, biểu thức mà ta muốn hiển thị ra màn hình và được ngăn cách với nhau bởi dấu phảy. Nhóm kí tự định dạng thứ *k* trong xâu_định_dạng dùng để xác định quy cách hiển thị tham số

thứ *k* trong danh_sách_tham_số. Do đó danh_sách_tham_số phải phù hợp về số lượng, thứ tự và kiểu với các nhóm kí tự định dạng trong xâu_định_dạng.

Ví dụ:

```
#include <conio.h>
#include <stdio.h>
void main()
{
    int a = 5;
    float x = 1.234;
    printf("Hien thi mot so nguyen %d và mot so thuc %f",a,x);
    getch();
}
```

Kết quả:

```
Hien thi mot so nguyen 5 va mot so thuc 1.234000
```

Trong ví dụ trên "Hien thi mot so nguyen %d và mot so thuc %f" là xâu định dạng, còn a và x là các tham số của hàm printf(). Trong xâu định dạng trên có hai nhóm kí tự định dạng là %d và %f, với %d dùng để báo cho máy biết rằng cần phải hiển thị tham số tương ứng (biến a) theo định dạng số nguyên và %f dùng để báo cho máy cần hiển thị tham số tương ứng (biến x) theo định dạng số thực.

Lưu ý là mỗi nhóm kí tự định dạng chỉ dùng cho một kiểu dữ liệu, còn một kiểu dữ liệu có thể hiển thị theo nhiều cách khác nhau nên có nhiều nhóm kí tự định dạng khác nhau. Nếu giữa nhóm kí tự định dạng và tham số tương ứng không phù hợp với nhau thì sẽ hiển thị ra kết quả không như ý. Phần sau đây giới thiệu một số nhóm kí tự định dạng hay dùng trong C và ý nghĩa của chúng.

Nhóm kí tự định dạng	Áp dụng cho kiểu dữ liệu	Ghi chú
%d	**int, long, char**	Hiển thị tham số tương ứng dưới dạng số nguyên có dấu hệ đếm thập phân.
%i	**int, long, char**	Hiển thị tham số tương ứng dưới dạng số nguyên có dấu hệ đếm thập phân.
%o	**int, long, char**	Hiển thị tham số tương ứng dưới dạng số nguyên không dấu trong hệ đếm cơ số 8.

%u	**int, long, char**	Hiển thị tham số tương ứng dưới dạng số nguyên không dấu.
%x	**int, long, char**	Hiển thị tham số tương ứng dưới dạng số nguyên hệ đếm 16 (không có 0x đứng trước), sử dụng các chữ cái a b c d e f.
%X	**int, long, char**	Hiển thị tham số tương ứng dưới dạng số nguyên hệ đếm 16 (không có 0x đứng trước), sử dụng các chữ cái A B C D E F.
%e	**float, double**	Hiển thị tham số tương ứng dưới dạng số thực dấu phẩy động.
%f	**float, double**	Hiển thị tham số tương ứng dưới dạng số thực dấu phẩy tĩnh.
%g	**float, double**	Hiển thị tham số tương ứng số thực dưới dạng ngắn gọn hơn trong hai dạng dấu phảy tĩnh và dấu phảy động.
%c	**int, long, char**	Hiển thị tham số tương ứng dưới dạng kí tự.
%s	**char *** (xâu kí tự)	Hiển thị tham số tương ứng dưới dạng xâu kí tự.

Để trình bày dữ liệu được đẹp hơn, C cho phép đưa thêm một số thuộc tính định dạng dữ liệu khác vào trong xâu định dạng như độ rộng tối thiểu, căn lề trái, căn lề phải.

Độ rộng tối thiểu

Thông thường khi hiển thị dữ liệu, C tự động xác định số chỗ cần thiết sao cho hiển thị vừa đủ nội dung dữ liệu.

Nếu ta muốn C hiển thị dữ liệu trên một số lượng vị trí xác định bất kể nội dung dữ liệu đó có điền đầy số chỗ được cung cấp hay không, ta có thể chèn một số nguyên vào trong nhóm kí tự định dạng, ngay sau dấu %.

Ví dụ khi hiển thị số nguyên:

```
a = 1234;
printf("\n%5d",a); // dành 5 chỗ để hiển thị số nguyên a
printf("\n%5d",34);// dành 5 chỗ để hiển thị số nguyên 34
```

Kết quả:

Ở đây □ kí hiệu thay cho dấu cách (space).

Như vậy với nhóm kí tự định dạng **%md**, m dùng để báo số chỗ cần dành để hiển thị dữ liệu, còn d báo rằng hãy hiển thị dữ liệu đó dưới dạng một số nguyên. Tương tự với các nhóm kí tự định dạng **%mc** khi hiển thị kí tự và **%ms** khi hiển thị xâu kí tự.

Ví dụ:

```
printf("\n%3d %15s %3c", 1, "nguyen van a", 'g');
printf("\n%3d %15s %3c", 2, "tran van b", 'k');
```

Lưu ý: trong lệnh printf viết ở trên, giữa các tham số định dạng có ngăn cách nhau bởi một dấu cách (space).

Kết quả:

Nếu nội dung dữ liệu không điền đầy số chỗ được cấp thì những chỗ không dùng đến sẽ được điền bởi dấu cách.

Khi số chỗ cần thiết để hiển thị nội dung dữ liệu lớn hơn m thì C tự động cung cấp thêm chỗ mới để hiển thị chứ không cắt bớt nội dung của dữ liệu để cho vừa m vị trí.

Với dữ liệu là số thực ta sử dụng mẫu nhóm kí tự định dạng **%m.nf** để báo rằng cần dành m vị trí để hiển thị số thực và trong m vị trí đó dành n vị trí để hiển thị phần thập phân.

Ví dụ:

```
printf("\n%f",12.345);
printf("\n%.2f",12.345);
printf("\n%8.2f",12.345);
```

Kết quả:

```
12.345000
12.35
□□□12.35
```

Căn lề trái

Khi hiển thị dữ liệu, mặc định C căn lề phải. Nếu muốn căn lề trái khi hiển thị dữ liệu ta chỉ cần thêm dấu trừ - vào ngay sau dấu %.

Ví dụ:

```
printf("\n%-3d %-15s %-4.2f %-3c", 1, "nguyen van a", 8.5, 'g');
printf("\n%-3d %-15s %-4.2f %-3c", 2, "tran van b", 6.75, 'k');
```

Kết quả:

```
1   nguyen van a    8.50 g
2   tran van b      6.75 k
```

Dễ thấy các thuộc tính định dạng độ rộng tối thiểu, căn lề... giúp cho việc hiển thị dữ liệu được thẳng, đều và đẹp hơn.

Thuộc tính	**Quy cách kí tự định dạng (m, n là các số nguyên)**	**Ví dụ**

Độ rộng tối thiểu	**%md, %ms, %mc**	printf("%3d",10);
		printf("%4s","CNTT");
		printf("%2c",'A');

Độ rộng dành cho phần thập phân	**%m.nf**	printf("%5.1f",1234.5);

Căn lề trái	**%-md, %-ms, %-mc**	printf("%-3d",10);
		printf("%-4s","CNTT");
		printf("%-2c",'A');

Hàm scanf()

Cú pháp:

scanf(xâu_định_dạng [, danh_sách_địa_chỉ]);

Hàm scanf() dùng để nhập dữ liệu từ bàn phím. Cụ thể nó sẽ đọc các kí tự được nhập từ bàn phím, sau đó căn cứ theo xâu_định_dạng sẽ chuyển những thông tin đã nhập được sang kiểu dữ liệu phù hợp. Cuối cùng sẽ gán những giá trị vừa nhập được vào các biến tương ứng trong danh_sách_địa_chỉ.

xâu_định_dạng trong hàm scanf() xác định khuôn dạng của các dữ liệu được nhập vào. Trong xâu_định_dạng có chứa các nhóm kí tự định dạng xác định khuôn dạng dữ liệu nhập vào.

Địa chỉ của một biến được viết bằng cách đặt dấu & trước tên biến. Ví dụ, ta có các biến có tên là a, x, ten_bien thì địa chỉ của chúng lần lượt sẽ là &a, &x, &ten_bien.

danh_sách_địa_chỉ phải phù hợp với các nhóm kí tự định dạng trong xâu_định_dạng về số lượng, kiểu dữ liệu và thứ tự. Các địa chỉ cũng được ngăn cách với nhau bởi dấu phẩy. Dưới đây là một số nhóm kí tự định dạng hay dùng và ý nghĩa.

Nhóm kí tự định dạng	**Ghi chú**
%d	Định khuôn dạng dữ liệu nhập vào dưới dạng số nguyên kiểu **int**
%o	Định khuôn dạng dữ liệu nhập vào dưới dạng số nguyên kiểu **int** hệ cơ số 8
%x	Định khuôn dạng dữ liệu nhập vào dưới dạng số nguyên kiểu **int** hệ cơ số 16
%c	Định khuôn dạng dữ liệu nhập vào dưới dạng kí tự kiểu **char**
%s	Định khuôn dạng dữ liệu nhập vào dưới dạng xâu kí tự
%f	Định khuôn dạng dữ liệu nhập vào dưới dạng số thực kiểu **float**
%ld	Định khuôn dạng dữ liệu nhập vào dưới dạng số nguyên kiểu **long**
%lf	Định khuôn dạng dữ liệu nhập vào dưới dạng số thực kiểu **double**

Ví dụ:

```
#include <conio.h>
#include <stdio.h>
void main()
{
    // khai bao bien
    int a;
    float x;
    char ch;
    char  str[30];
    // Nhap du lieu
    printf("Nhap vao mot so nguyen");
    scanf("%d",&a);
    printf("\n  Nhap vao mot so thuc");
    scanf("%f",&x);
    printf("\n Nhap vao mot ki tu");
    fflush(stdin); scanf("%c",&ch);
    printf("\n Nhap vao mot xau ki tu");
    fflush(stdin); scanf("%s",str);
    // Hien thi du lieu vua nhap vao
    printf("\n Nhung du lieu vua nhap vao");
    printf("\n So nguyen: %d",a);
    printf("\n So thuc : %.2f",x);
    printf("\n Ki tu: %c",ch);
    printf("\n Xau ki tu: %s",str);
}
```

Kết quả:

```
Nhap vao mot so nguyen: 2007
Nhap vao mot so thuc: 18.1625
Nhap vao mot ki tu: b
Nhap vao mot xau ki tu: ngon ngu lap trinh C
Nhung du lieu vua nhap vao
So nguyen: 2007
```

```
So thuc: 18.16
Ki tu: b
Xau ki tu: ngon
```

Một số quy tắc cần lưu ý khi sử dụng hàm scanf():

- Quy tắc 1: Khi đọc số, hàm scanf() quan niệm rằng mọi kí tự số, dấu chấm ('.') đều là kí tự hợp lệ. Khi gặp các dấu phân cách như tab, xuống dòng hay dấu cách (space bar) thì scanf() sẽ hiểu là kết thúc nhập dữ liệu cho một số.
- Quy tắc 2: Khi đọc kí tự, hàm scanf() cho rằng mọi kí tự có trong bộ đệm của thiết bị vào chuẩn đều là hợp lệ, kể cả các kí tự tab, xuống dòng hay dấu cách.
- Quy tắc 3: Khi đọc xâu kí tự, hàm scanf() nếu gặp các kí tự dấu cách, dấu tab hay dấu xuống dòng thì nó sẽ hiểu là kết thúc nhập dữ liệu cho một xâu kí tự. Vì vậy trước khi nhập dữ liệu kí tự hay xâu kí tự ta nên dùng lệnh fflush(stdin).

III.3.1.2. Các lệnh nhập xuất khác

Hàm gets(), có cú pháp:

gets(xâu_kí_tự);

Hàm gets() dùng để nhập vào từ bàn phím một xâu kí tự bao gồm cả dấu cách, điều mà hàm scanf() không làm được.

Hàm puts(), có cú pháp:

puts(xâu_kí_tự);

Hàm puts() sẽ hiển thị ra màn hình nội dung xâu_kí_tự và sau đó đưa con trỏ xuống dòng mới. Vì vậy nó tương đương với lệnh printf("%s\n",xâu_kí_tự).

Hàm getch(), có cú pháp:

getch();

Hàm getch() là hàm không có tham số. Nó đọc một kí tự bất kì nhập vào từ bàn phím nhưng không hiển thị kí tự đó lên màn hình. Lệnh getch() thường dùng để chờ người sử dụng ấn một phím bất kì rồi sẽ kết thúc chương trình.

Để sử dụng các hàm gets(), puts(), getch() ta cần khai báo tệp tiêu đề conio.h.

Ví dụ:

```
#include <conio.h>
#include <stdio.h>
void main()
{
    // khai bao bien
    char str[30];
    // Nhap du lieu
    puts("Nhap vao mot xau ki tu:");
    fflush(stdin); gets(str);
    // Hien thi du lieu vua nhap vao
    puts("Xau vua nhap vao: ");
    puts(str);
    puts("An phim bat ki de ket thuc ...");
    getch();
}
```

Kết quả:

```
Nhap vao mot xau ki tu:
ngon ngu lap trinh C
Xau vua nhap vao:
ngon ngu lap trinh C
An phim bat ki de ket thuc ...
```

III.3.2. Cấu trúc khối lệnh

Một cách hình thức ta có thể định nghĩa một khối lệnh là dãy các câu lệnh được đặt trong cặp dấu ngoặc nhọn { }.

```
{
    lệnh_1;
    lệnh_2;
    ....
    lệnh_n;
}
```

Trong khối lệnh có thể chứa khối lệnh khác, ta gọi đó là các khối lệnh lồng nhau. Sự lồng nhau của các khối lệnh là không hạn chế. Các lệnh trong khối lệnh được thực hiện tuần tự theo trật tự xuất hiện.

```
{
lệnh;
    {
        lệnh;
        ...
    }
    ...
}
```

C cho phép khai báo biến trong khối lệnh. Ràng buộc duy nhất là phần khai báo phải nằm trước phần câu lệnh.

Ví dụ:

```
#include <conio.h>
#include <stdio.h>
void main()
// Noi dung cua ham main() cung la mot khoi lenh
{
        // khai bao bien
        int c;
        c = 10;
        printf(" Gia tri cua c = %d day la c ngoai",c);
        // bat dau mot khoi lenh khac
        {
            int c;
            c = 10;
            printf("\n Gia tri cua c = %d day la c trong",c);
            printf("\n Tang gia tri cua c them 10 don vi");
            c = c + 10;
            printf("\n Gia tri cua c = %d day la c trong",c);
        }
        printf("\n Gia tri cua c = %d day la c ngoai",c);
        getch();
}
```

Kết quả:

```
Gia tri cua c = 10 day la c ngoai
Gia tri cua c = 10 day la c trong
Tang gia tri cua c them 10 don vi
Gia tri cua c = 20 day la c trong
Gia tri cua c = 10 day la c ngoai
```

III.3.3. Cấu trúc if

Cú pháp cấu trúc **if**

if (biểu_thức_điều_kiện)

lệnh;

Cú pháp cấu trúc **if ... else**

if (biểu_thức_điều_kiện)

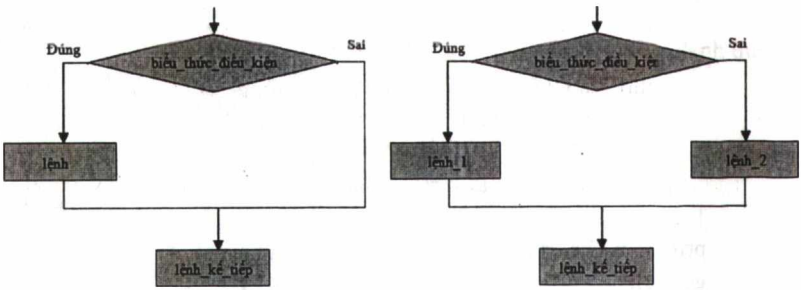
lệnh_1;

else

lệnh_2;

Lưu ý: lệnh, lệnh_1 và lệnh_2 có thể là khối lệnh.

Dưới đây là sơ đồ khối của hai cú pháp lệnh trên.



Hình III.3. Sơ đồ khối cấu trúc if

Ví dụ: Bài toán tìm số lớn nhất trong hai số thực *a* và *b*. Cách làm là so sánh *a* với *b*, nếu *a* nhỏ hơn *b* thì *b* là số lớn nhất, còn nếu không, tức là $a \geq b$, thì *a* là số lớn nhất.

```
#include <conio.h>
#include <stdio.h>
void main()
{
        // khai bao bien
        float a, b;
        float max;
        printf(" Nhap gia tri a va b: ");
        scanf("%f %f",&a,&b);
        if(a<b)
            max = b;
        else
            max = a;
        printf("\n So lon nhat trong 2 so %.0f va %.0f la %.0f ",a,b,max);
        getch();
}
```

Kết quả:

```
Nhap vao 2 gia tri a va b: 23 247
So lon nhat trong hai so 23 va 247 la 247
```

III.3.4. Cấu trúc lựa chọn switch

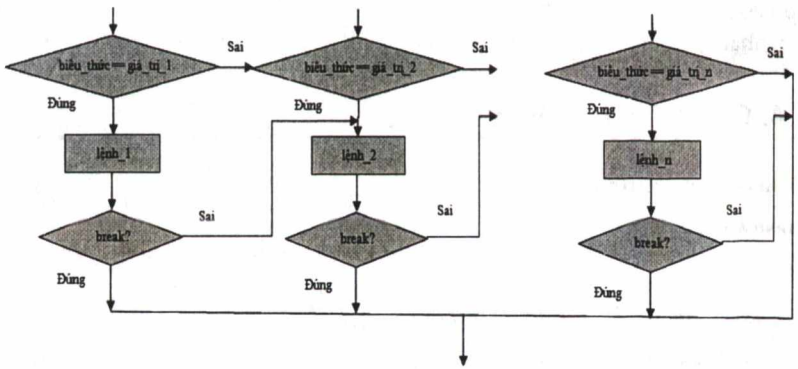
Cú pháp cấu trúc **switch**

```
switch (biểu_thức)
{
    case gia_tri_1:     lenh_1; [break;]
    case gia_tri_2:     lenh_2; [break;]
      ...
    case gia_tri_n:     lenh_n; [break;]
    [default:       lenh_n+1; [break;]]
}
```

Cơ chế hoạt động: Câu lệnh **switch** ban đầu sẽ tính giá trị của biểu_thức, sau đó so sánh với các giá_trị_k với k = 1, 2, …, n đứng sau **case**. Xảy ra hai trường hợp:

- Nếu trong dãy các giá trị giá_trị_1, giá_trị_2, ... tồn tại giá trị bằng biểu_thức: Gọi i là chỉ số của giá trị đầu tiên trong dãy thỏa mãn giá_trị_i bằng biểu_thức, khi đó lệnh_i sẽ được thực hiện. Sau khi thực hiện xong lệnh_i, nếu có lệnh **break** thì chương trình sẽ chuyển sang thực hiện lệnh tiếp sau cấu trúc **switch**. Nếu không có lệnh **break** thì chương trình sẽ chuyển sang thực hiện các lệnh sau lệnh_i nằm trong **switch** (tức là lệnh_i+1, lệnh_i+2…) cho đến khi gặp lệnh **break** đầu tiên hoặc sau khi thực hiện xong lệnh n. Sau đó chương trình sẽ chuyển sang thực hiện lệnh tiếp theo sau cấu trúc **switch**.
- Nếu không tồn tại giá_trị_k (với k = 1, 2, …n) nào bằng giá trị của biểu_thức thì sẽ có hai khả năng:
  - Nếu có nhãn **default**: Chương trình sẽ thực hiện lệnh_n+1 rồi chuyển sang thực hiện lệnh tiếp theo sau cấu trúc **switch**.
  - Nếu không có nhãn **default**: Chương trình chuyển sang thực hiện lệnh tiếp theo sau cấu trúc **switch**.

Sơ đồ:



Hình III.4. Sơ đồ khối cấu trúc switch

Ví dụ:

/* Ví dụ sau yêu cầu người dùng nhập vào một số nguyên không âm và đưa ra ngày trong tuần tương ứng với số nguyên đó. Ở đây ta quy ước những số chia

hết cho 7 ứng với Chủ nhật, chia 7 dư 1 ứng với Thứ Hai, ..., chia 7 dư 6 ứng với Thứ Bảy.*/

```
#include <conio.h>
#include <stdio.h>
void main()
{
        // khai bao bien
        int a;
        do
        {
            printf("\n Nhap mot gia tri so nguyen khong am: ");
            scanf("%d",&a);
            if(a<0)
               printf("\n so vua nhap la so am");
        }while(a<0);
        printf("\n Thu trong tuan tuong ung voi so do la: ");
        switch(a % 7)
        {
           case 0: printf(" Chu nhat"); break;
           case 1: printf(" Thu Hai"); break;
           case 2: printf(" Thu Ba"); break;
           case 3: printf(" Thu Tu"); break;
           case 4: printf(" Thu Nam"); break;
           case 5: printf(" Thu Sau"); break;
           case 6: printf(" Thu Bay"); break;
        }
        getch();
}
```

Kết quả:

```
Nhap vao mot gia tri so nguyen: 2356
Thu tuong ung voi so do la Thu Nam
```

Người ta thường dựa trên tính chất tự động chuyển xuống các câu lệnh sau khi không có lệnh break để viết chung mã lệnh cho các trường hợp khác nhau nhưng cùng được xử lý giống nhau. Ví dụ, khi viết chương trình hỗ trợ menu dòng lệnh không phân biệt chữ hoa chữ thường hay bài toán in ra số ngày trong các tháng

trong năm dưới đây. Trong một năm các tháng có 30 ngày là 4, 6, 9, 11 còn các tháng có 31 ngày là 1, 3, 5, 7, 8, 10, 12. Riêng tháng hai có thể có 28 hoặc 29 ngày.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
        int thang;
         clrscr();
         printf("\n Nhap vao thang trong nam ");
         scanf("%d",&thang);
         switch(thang)
        {
          case 1:
          case 3:
          case 5:
          case 7:
          case 8:
          case 10:
          case 12:
             printf("\n Thang %d co 31 ngay ",thang);
             break;
          case 4:
          case 6:
          case 9:
          case 11:
             printf("\n Thang %d co 30 ngay ",thang);
             break;
          case 2:
             printf ("\ Thang 2 co 28 hoac 29 ngay");
             break;
          default :
             printf("\n Khong co thang %d", thang);
             break;
        }
        getch();
```

```
    retum 0;
}
```

Lưu ý: Giá trị của biểu thức kiểm tra phải là số nguyên tức là phải có kiểu dữ liệu là **char**, **int**, **long**. Một cách tương ứng các giá trị sau **case** cũng phải nguyên. Đây là một trong những điểm phân biệt giữa cấu trúc **switch** và **if...else**.

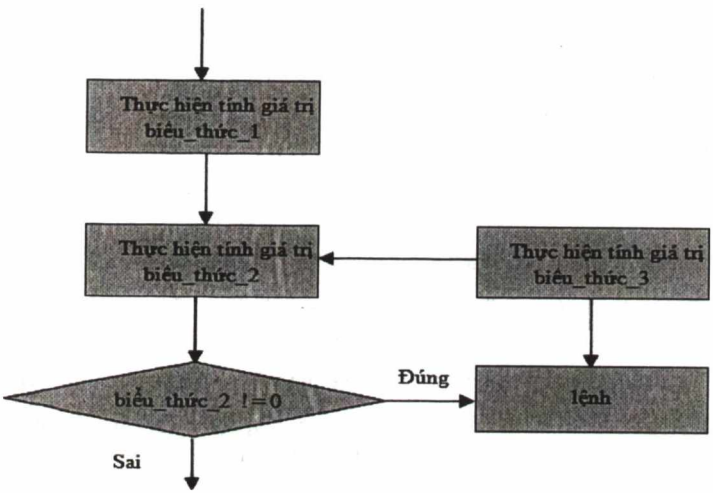
III.3.5. Vòng lặp for

Dạng thường gặp của vòng lặp **for** là:

for([biểu_thức_1];[biểu_thức_2];[biểu_thức_3])

lệnh;

Sơ đồ:



Hình III.5. Sơ đồ khối cấu trúc for

Câu lệnh **for** thường dùng để thực hiện lặp đi lặp lại một công việc nào đó với số lần lặp xác định.

Ví dụ 1: Hãy đưa ra màn hình các số nguyên dương nhỏ hơn 10.

Cách làm: Có 9 số nguyên dương nhỏ hơn 10 là 1, 2, 3, 4, 5, 6, 7, 8 và 9. Để in 9 số nguyên dương này ta cần sử dụng một biến nguyên đặt tên là *i*.

Bước 1: Gán cho i giá trị bằng 1;

Bước 2: Đưa ra màn hình giá trị của i;

Bước 3: Tăng giá trị của i thêm 1 đơn vị;

Bước 4: Kiểm tra nếu giá trị của i ≤ 9 thì quay về bước 2, nếu giá trị của i > 9 thì chuyển sang bước 5;

Bước 5: Kết thúc.

Chương trình in ra màn hình các số nguyên dương nhỏ hơn 10 sử dụng vòng lặp for:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i;
    for(i = 1; i <= 9; i++)
        printf("%5d",i);// ta danh 5 vi tri de in moi so
    getch();
}
```

Kết quả thực hiện:

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

Ta có thể so sánh cách làm này với việc phải viết một lệnh printf() trong đó liệt kê toàn bộ các số nguyên dương lẻ nhỏ hơn 10.

Ví dụ 2: Tính và hiển thị ra màn hình tổng của 100 số tự nhiên lẻ đầu tiên.

Cách làm: 100 số lẻ đầu tiên là 1, 3, 5, ..., 199. Ta cần sử dụng một biến nguyên S để chứa giá trị của tổng và một biến nguyên i.

Bước 1: Ban đầu gán cho S giá trị bằng 0, gán cho i giá trị bằng 1;

Bước 2: S = S + i;

Bước 3: Tăng giá trị của i thêm 2 đơn vị;

Bước 4: Kiểm tra nếu giá trị của i ≤ 199 thì quay về bước 2, ngược lại nếu i > 199 thì chuyển sang bước 5;

Bước 5: Kết thúc.

Chương trình:

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int i;
        int S;
        S = 0;
        for(i = 1; i <= 199; i = i+2)
            S = S+i;
        printf("Tong cua 100 so nguyen duong le dau tien la %d",S);
        getch();
}
```

Kết quả thực hiện:

```
Tong cua 100 so nguyen duong le dau tien la 10000
```

III.3.6. Vòng lặp while và do – while

Cú pháp vòng lặp **while**

while (biểu_thức)

lệnh;

Cú pháp vòng lặp **do {...} while**

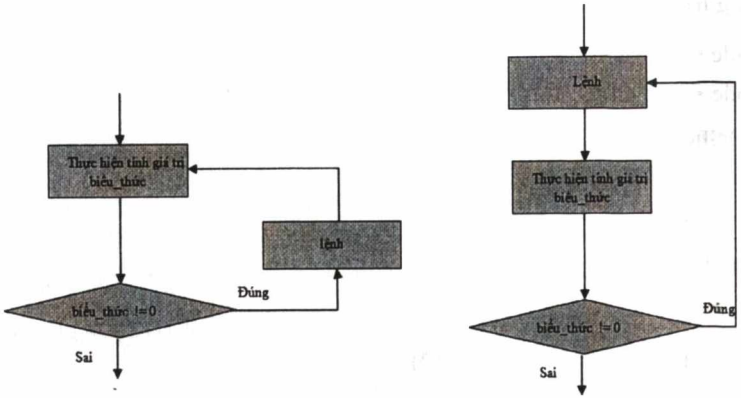
do

{

lệnh;

}while (biểu_thức);

Sơ đồ vòng lặp **while** và **do{...}while:**



Hình III.6. Sơ đồ khối cấu trúc while và do while

Sự khác nhau giữa **while** và **do{...}while:**

- Lệnh **while** kiểm tra điều kiện vòng lặp (tức là giá trị của biểu thức) trước rồi mới thực hiện lệnh, do vậy sẽ có khả năng lệnh không được thực hiện lần nào.
- Lệnh **do{...}while** thực hiện lệnh trước rồi mới kiểm tra điều kiện của vòng lặp, do vậy lệnh sẽ luôn được thực hiện ít nhất một lần.

Cấu trúc **while** và **do{...}while** được dùng để thực hiện lặp đi lặp lại một công việc nào đó với số lần lặp không xác định.

Ví dụ: Sau đây là hai đoạn chương trình có chức năng tương đương nhau nhưng một đoạn sử dụng cấu trúc **while**, một đoạn sử dụng cấu trúc **do{...}while**.

Chức năng của chương trình là yêu cầu người dùng nhập vào giá trị một số nguyên, đưa ra thông báo số đó có phải là số hoàn thiện hay không, sau đó hỏi người dùng có muốn nhập lại số nguyên khác và kiểm tra có phải số hoàn thiện hay không.

*Đoạn chương trình sử dụng cấu trúc **do{...}while***

```
#include <stdio.h>
#include <conio.h>
void main()
{
    long int n;
```

```
    long int tong;
    long int i;
    char ch;
    clrscr();
    do
    {
        tong = 0;
        printf("\n Nhap vao mot so nguyen: ");
        scanf("%ld",&n);
        printf("\n Cac uoc so cua %ld la: ",n);
        for(i = 1;i<n;i++)
            if(n % i == 0)
                        {
                                printf("%5d",i);
                                tong = tong + i;
            }
        printf("\n Tong cac uoc so cua %ld bang %ld",n,tong);
        if(tong == n)
            printf("\n %5ld LA so hoan thien", n);
        else
            printf("\n %5ld KHONG LA so hoan thien", n);
        printf("\n Ban co muon thuc hien lai(c/k)? ");
        fflush(stdin);
        scanf("%c",&ch);
    }while((ch!='k')&&(ch!='K'));
    printf("\n An phim bat ki de ket thuc ...");
    getch();
}
```

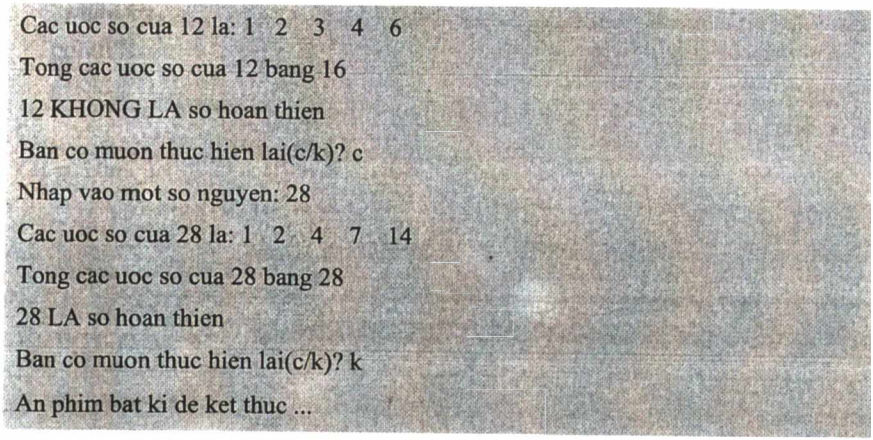
*Đoạn chương trình viết bằng cấu trúc **while***

```
#include <stdio.h>
#include <conio.h>
void main()
{
    long int n;
    long int tong;
    long int i;
```

```
    char ch;
    clrscr();
    ch = 'c';
    while((ch != 'k')&&(ch != 'K'))
    {
        tong = 0;
        printf("\n Nhap vao mot so nguyen: ");
        scanf("%ld",&n);
        printf("\n Cac uoc so cua %ld la: ", n);
        for(i = 1; i<n; i++)
            if(n % i == 0)
                        {
                                printf("%5d", i);
                                tong = tong + i;
            }
        printf("\n Tong cac uoc so cua %ld bang %ld", n, tong);
        if(tong == n)
            printf("\n %5ld LA so hoan thien", n);
        else
            printf("\n %5ld KHONG LA so hoan thien", n);
        printf("\n Ban co muon thuc hien lai(c/k)? ");
        fflush(stdin);
        scanf("%c",&ch);
    }
    printf("\n An phim bat ki de ket thuc ...");
    getch();
}
```

Kết quả thực hiện chương trình:

```
Nhap vao mot so nguyen: 10
Cac uoc so cua 10 la: 1   2   5
Tong cac uoc so cua 10 bang 8
10 KHONG LA so hoan thien
Ban co muon thuc hien lai(c/k)? c
Nhap vao mot so nguyen: 12
```



III.3.7. Các lệnh thay đổi cấu trúc lập trình

Các vòng lặp **while**, **do{...}while**, hay **for** sẽ kết thúc quá trình lặp khi biểu thức điều kiện của vòng lặp không còn được thỏa mãn. Tuy nhiên trong lập trình đôi khi ta cũng cần thoát khỏi vòng lặp ngay cả khi biểu thức điều kiện của vòng lặp vẫn còn được thỏa mãn. Để hỗ trợ người lập trình làm việc đó, ngôn ngữ C cung cấp hai câu lệnh là **continue** và **break**.

III.3.7.1. Continue

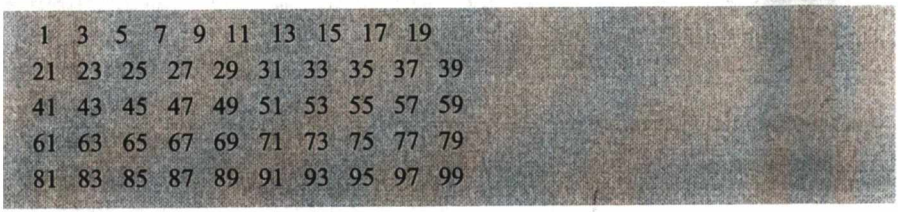
Khi gặp lệnh **continue** trong thân vòng lặp, chương trình sẽ chuyển sang thực hiện một vòng lặp mới và bỏ qua việc thực hiện các câu lệnh nằm sau lệnh **continue** trong thân vòng lặp.

Ví dụ sau đây sẽ in ra màn hình các số tự nhiên lẻ và nhỏ hơn 100:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i;
    for(i = 1;i<100;i++)
    {
        if(i%2 == 0) continue;
```

```
            printf("%5d", i);
        if((i+1)%20 ==0) printf("\n");
    }
    getch();
}
```

Kết quả thực hiện:



III.3.7.2. Break

Khi gặp lệnh **break**, chương trình sẽ thoát khỏi vòng lặp (đối với trường hợp lệnh **break** nằm trong các cấu trúc lặp **while**, **do{...}while**, **for**) hoặc thoát khỏi cấu trúc **switch** (với trường hợp lệnh **break** nằm trong cấu trúc **switch**).

Ví dụ sau sẽ thực hiện việc yêu cầu người dùng nhập vào một kí tự và màn hình thông báo về kí tự vừa nhập. Việc này được lặp đi lặp lại cho đến khi kí tự nhập vào là kí tự 't' hoặc 'T' (viết tắt của từ thoát).

```
#include <stdio.h>
#include <conio.h>
void main()
{
    char ch;
    clrscr();
    do
    {
        printf("\n Nhap vao mot ki tu: ");
        fflush(stdin);
        scanf("%c",&ch);
        printf("\n Ki tu vua nhap vao la %c",ch);
        if((ch =='T')||(ch =='t')) break;
```

```
    }while(1);
    printf("\n An phim bat ki de ket thuc chuong trinh...");
    getch();
}
```

Kết quả thực hiện chương trình:

III.4. Mảng, con trỏ và xâu ký tự

III.4.1. Mảng

III.4.1.1. Khái niệm mảng

Khái niệm mảng

Mảng là một tập hợp hữu hạn các phần tử có cùng kiểu dữ liệu được lưu trữ kế tiếp nhau trong bộ nhớ. Các phần tử trong mảng có cùng tên (và cũng là tên mảng) nhưng phân biệt với nhau ở chỉ số cho biết vị trí của chúng trong mảng.

III.4.1.2. Khai báo và sử dụng mảng

Cú pháp khai báo

Trong C để khai báo một mảng ta sử dụng cú pháp khai báo sau:

kiểu_dữ_liệu tên_mảng [kích_thước_mảng];

Trong đó kiểu_dữ_liệu là kiểu dữ liệu của các phần tử trong mảng, tên_mảng là tên của mảng, kích_thước_mảng cho biết số phần tử trong mảng.

Ví dụ:

```
int mang_nguyen[10]; // khai bao mang 10 phan tu co kieu du lieu int
float mang_thuc[4];  // khai bao mang 4 phan tu co kieu du lieu float
char mang_ki_tu[6];  // khai bao mang 6 phan tu co kieu du lieu char
```

Trong ví dụ trên, mảng mang_nguyen được lưu trữ trên 20 ô nhớ (mỗi ô nhớ có kích thước 1 byte, hai ô nhớ kích thước là 2 byte lưu trữ được một số nguyên kiểu **int**) liên tiếp nhau. Do C đánh số các phần tử của mảng bắt đầu từ 0 nên phần tử thứ i của mảng sẽ có chỉ số là i-1 và do vậy sẽ có tên là mang_nguyen[i-1]. Ví dụ, phần tử thứ nhất của mảng là mang_nguyen[0], phần tử thứ hai là mang_nguyen[1], phần tử thứ năm là mang_nguyen[4]...

Kích thước của mảng bằng kích thước một phần tử nhân với số phần tử.

Mảng một chiều và nhiều chiều

Các mảng trong ví dụ trên là các mảng một chiều. Mảng là tập hợp các phần tử cùng kiểu dữ liệu, nếu mỗi phần tử của mảng cũng là một mảng khác thì khi đó ta có mảng nhiều chiều. Khái niệm mảng rất giống với khái niệm vector trong toán học.

Ví dụ sau khai báo một mảng gồm sáu phần tử, trong đó mỗi phần tử lại là một mảng gồm năm số nguyên kiểu **int**. Mảng này là mảng hai chiều:

```
int a[6][5];
```

Còn khai báo:

```
int b[3][4][5];
```

thì lại khai báo một mảng gồm ba phần tử, mỗi phần tử lại là một mảng hai chiều gồm bốn phần tử. Mỗi phần tử của mảng hai chiều lại là một mảng (một chiều) gồm năm số nguyên kiểu **int**. Mảng b ở trên được gọi là một mảng ba chiều.

Sử dụng mảng

Để truy nhập vào một phần tử của mảng ta phải sử dụng tên của nó. Tên một phần tử của mảng được tạo thành từ tên mảng và theo sau là chỉ số của phần tử đó trong mảng được đặt trong cặp dấu ngoặc vuông.

tên_mảng[chỉ_số_của_phần_tử]

Ví dụ: Với khai báo

int mang_nguyen[3];

Thì mang_nguyen[0] sẽ là phần tử thứ nhất của mảng.

mang_nguyen[1] sẽ là phần tử thứ hai của mảng.

mang_nguyen[2] sẽ là phần tử thứ ba của mảng.

Với mảng nhiều chiều như:

int a[6][5];

thì a[0] là phần tử đầu tiên của mảng, phần tử này bản thân nó lại là một mảng một chiều. Phần tử đầu tiên của mảng một chiều a[0] sẽ là a[0][0]. Phần tử tiếp theo của a[0] sẽ là a[0][1]... và dễ dàng tính được a[2][3] sẽ là phần tử thứ tư của phần tử thứ ba của a.

Một cách tổng quát a[i][j] sẽ là phần tử thứ j+1 của a[i], mà phần tử a[i] lại là phần tử thứ i+1 của a.

III.4.1.3. Các thao tác cơ bản trên mảng

a. Nhập dữ liệu cho mảng

Sau khi khai báo mảng ta phải nhập dữ liệu cho mảng. Nhập dữ liệu cho mảng là nhập dữ liệu cho từng phần tử của mảng. Mỗi một phần tử của mảng thực chất là một biến có kiểu dữ liệu là kiểu dữ liệu chung của mảng. Để nhập dữ liệu cho các phần tử của mảng ta có thể dùng hàm scanf() hoặc lệnh gán tương tự như biến thông thường.

Ví dụ:

```
float a[10];    // khai bao mot mang so thuc co 10 phan tu.
int i;
// Nhap tu ban phim mot so thuc va gan gia tri so thuc do
// cho phan tu thu 2 cua mang, tuc la a[1].
scanf("%f",&a[1]);
// Gan gia tri cho phan tu a[2].
a[2] = a[1] + 5;
```

Nếu muốn gán giá trị cho các phần tử của mảng một cách hàng loạt, ta có thể dùng lệnh **for**. Ví dụ:

```
int b[10];
int i;
// Nhap gia tri tu ban phim cho tat ca cac phan tu cua mang b.
for(i = 0; i < 10; i++)
{
   printf("\n Nhap gia tri cho b[%d]", i);
   scanf("%d",&b[i]);
}
```

Trường hợp ta không biết trước mảng sẽ có bao nhiêu phần tử mà chỉ biết số phần tử tối đa có thể có của mảng, còn số phần tử thực sự của mảng thì chỉ biết khi chạy chương trình; khi đó cần khai báo mảng với số phần tử bằng số phần tử tối đa, ngoài ra cần một biến để lưu giữ số phần tử thực sự của mảng.

```
int a[100]; // khai bao mang, gia su so phan tu toi da cua a la 100.
int n;      // bien luu giu so phan tu thuc su cua mang.
int i;
printf("\n Cho biet so phan tu cua mang: ");
scanf("%d",&n);
for(i = 0; i < n; i++)
{
   printf("\n a[%d] = ", i);
   scanf("%d",&a[i]);
}
```

Mảng có thể được khởi tạo giá trị ngay khi khai báo, ví dụ:

```
int a[4] = {4, 9, 22, 16};
float b[3] = {40.5, 20.1, 100};
char c[5] = {'h', 'e', 'l', 'l', 'o'};
```

Câu lệnh thứ nhất có tác dụng tương đương với bốn lệnh gán:

```
a[0] = 4; a[1] = 9; a[2] = 22; a[3] = 16;
```

b. Xuất dữ liệu chứa trong mảng

Để hiển thị giá trị của các phần tử trong mảng ta dùng hàm printf(). Ví dụ sau minh họa việc nhập giá trị cho các phần tử của mảng, sau đó hiển thị giá trị của các phần tử đó theo các cách khác nhau.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a[5];
    int i, k;
    // Nhap gia tri cho cac phan tu cua mang a tu ban phim
    for(i = 0; i < 5; i++)
    {
        printf("\n a[%d] = ", i);
        scanf("%d", &a[i]);
    }
    // Hien thi gia tri cua phan tu bat ki, gia su a[3] len man hinh
    printf("\n a[3] = %d", a[3]);
    // Hien thi gia tri cua tat ca cac phan tu, moi phan tu tren mot dong
    for(i = 0; i < 5; i++)
        printf("\n%d", a[i]);
    // Hien thi gia tri cua tat ca cac phan tu tren cung mot dong, cac gia tri cach
    // nhau 2 vi tri.
    printf("\n");     // Xuong dong moi
    for(i = 0; i < 5; i++)
        printf("%d  ", a[i]);
    // Hien thi gia tri cua tat ca cac phan tu, trong do k phan tu tren mot dong.
    // Cac phan tu tren cung dong cach nhau 2 vi tri
    printf("\n Cho biet gia tri cua k = ");
    scanf("%d",&k);
    for(i = 0; i < 5; i++)
    {
        printf("%d  ",a[i]);
        if((i+1)%k == 0)// da hien thi du k phan tu tren mot dong thi phai xuong dong.
            printf("\n");
    }
    getch();
```

Kết quả

```
a[0] = 6
a[1] = 14
a[2] = 23
a[3] = 37
a[4] = 9
a[3] = 37
6
14
23
37
9
6  14  23  37  9
 Cho biet gia tri cua k = 2
6  14
23  37
9
```

c. Tìm phần tử có giá trị lớn nhất, phần tử có giá trị nhỏ nhất

Để tìm phần tử có giá trị lớn nhất trong mảng, ban đầu, giả sử phần tử đó là phần tử đầu tiên của mảng. Sau đó, lần lượt so sánh với các phần tử còn lại trong mảng. Nếu gặp phần tử nhỏ hơn thì chuyển sang so sánh với phần tử tiếp theo. Nếu gặp phần tử lớn hơn thì coi phần tử này là phần tử lớn nhất rồi chuyển sang so sánh với phần tử tiếp theo. Quá trình so sánh lặp lại cho đến khi gặp phần tử cuối cùng của dãy, khi đó tìm được phần tử lớn nhất của mảng. Đoạn chương trình sau minh họa giải thuật tìm phần tử lớn nhất:

```
int a[100];
int i, n;
int max;
printf("\n Cho biet so phan tu cua mang: ");
scanf("%d",&n);
// Nhap du lieu cho mang
for(i = 0; i < n; i++)
{
   printf("\n a[%d] = ",i);
   scanf("%d",&a[i]);
}
```

```
// Tim phan tu lon nhat
max = a[0];   // Ban dau gia su phan tu lon nhat la a[0]

// Lan luot so sanh voi cac phan tu con lai trong mang

for(i = 1; i < n; i++)
   if(max < a[i])       // Gap phan tu co gia tri lon hon
        max = a[i]; // coi phan tu nay la phan tu lon nhat
printf("\n Phan tu lon nhat trong mang la: %d", max);
```

Ta cũng làm tương tự với trường hợp tìm phần tử nhỏ nhất trong mảng.

III.4.1.4. Tìm kiếm trên mảng

Yêu cầu của thao tác tìm kiếm trên mảng: Cho một mảng dữ liệu và một dữ liệu bên ngoài mảng. Hãy tìm (các) phần tử của mảng có giá trị bằng giá trị của dữ liệu bên ngoài trên. Nếu có (các) phần tử như vậy thì hãy chỉ ra vị trí của chúng trong mảng. Nếu không tồn tại (các) phần tử như vậy thì đưa ra thông báo không tìm thấy.

Cách làm là lần lượt duyệt qua từng phần tử của mảng, so sánh giá trị của phần tử đang được duyệt với giá trị của dữ liệu bên ngoài. Nếu phần tử đang xét bằng dữ liệu bên ngoài thì ta ghi nhận vị trí của phần tử đó. Sau đó chuyển qua duyệt phần tử kế tiếp trong mảng. Quá trình được lặp đi lặp lại cho đến khi duyệt xong phần tử cuối cùng của mảng. Để có thể trả lời cho cả tình huống không tồn tại phần tử như vậy trong mảng, ta nên sử dụng một biến kiểm tra và khi gặp phần tử bằng giá trị dữ liệu bên ngoài thì ta bật biến đó lên, nếu biến đó không được bật lần nào thì trả lời là không có phần tử như vậy trong mảng. Phương pháp trên được gọi là phương pháp tìm kiếm tuần tự (*sequential search*).

Dưới đây là cài đặt của thuật toán tìm kiếm tuần tự cho trường hợp mảng dữ liệu là mảng các số nguyên kiểu **int**.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int m[100], chi_so[100];
    int n;   // n la so phan tu trong mang
    int i, k, kiem_tra;
    clrscr();    // xoa man hinh de tien theo doi
    // Nhap gia tri du lieu cho mang m
```

```
    // Truoc tien phai biet so phan tu cua mang
    printf(" Cho biet so phan tu co trong mang: ");
    scanf("%d",&n);
    // Roi lan luot nhap gia tri cho cac phan tu trong mang
        for(i  = 0;i<n;i++)
    {
        int temp;
        printf("\n Cho biet gia tri cua m[%d] = ",i);
        scanf("%d",&temp);
        m[i] = temp;
    }
    // Yeu cau nguoi su dung nhap vao gia tri cho du lieu k
    printf("\n Cho biet gia tri cua du lieu k: ");
    scanf("%d",&k);
    // Bat đau qua trinh tim kiem
    kiem_tra = 0;
    // Duyet qua tat ca cac phan tu
    for(i = 0;i<n;i++)
        if(m[i] = = k)//So sanh phan tu dang xet voi du lieu k
        {
            // Ghi nhan chi so cua phan tu dang xet
            chi_so[kiem_tra] = i;
                kiem_tra ++; //Tang bien kiem_tra them 1 don vi
        }
            // Ket luan
    if(kiem_tra > 0)
    {
        printf("\n Trong mang co %d phan tu co gia tri bang %d",kiem_tra,k);
        printf("\n Chi so cua cac phan tu la: ");
        for(i = 0;i < kiem_tra;i++)
            printf("%3d",chi_so[i]);
    }
    else
        printf("\n Trong mang khong co phan tu nao co gia tri bang %d",k);
    getch();     // Cho nguoi su dung an phim bat ky de ket thuc.
}
```

III.4.1.5. Sắp xếp mảng

Yêu cầu của bài toán: Cho một mảng dữ liệu m[n] với n là số phần tử trong mảng. Hãy sắp xếp các phần tử trong mảng theo một trật tự nào đó, giả sử là theo chiều tăng dần (với chiều giảm dần ta hoàn toàn có thể suy luận từ cách làm với chiều tăng dần).

Sắp xếp kiểu lựa chọn (*Selection sort*): Ý tưởng của phương pháp là ta cần thực hiện n – 1 lượt sắp xếp, trong đó:

- Ở lượt sắp xếp đầu tiên ta so sánh phần tử đầu tiên của mảng m[0] với tất cả các phần tử đứng sau nó trong mảng (tức là các phần tử m[1], m[2]...m[n-1]). Nếu có giá trị m[i] nào đó (i = 1, 2,...n–1) nhỏ hơn m[0] thì ta hoán đổi giá trị giữa m[0] và m[i] cho nhau. Rõ ràng sau lượt sắp xếp thứ nhất m[0] sẽ là phần tử có giá trị nhỏ nhất trong mảng.

- Ở lượt sắp xếp thứ hai ta so sánh phần tử thứ hai của mảng m[1] với tất cả các phần tử đứng sau nó trong mảng (tức là các phần tử m[2], m[3]...m[n-1]). Nếu có giá trị m[i] nào đó (i = 2, 3,...n–1) nhỏ hơn m[1] thì ta hoán đổi giá trị giữa m[1] và m[i] cho nhau. Sau lượt sắp xếp thứ hai thì m[1] sẽ là phần tử có giá trị nhỏ thứ hai trong mảng.

  ...

- Ở lượt sắp xếp thứ k ta so sánh phần tử thứ k của mảng là m[k–1] với tất cả các phần tử đứng sau nó trong mảng (tức là các phần tử m[k], m[k+1]...m[n–1]). Nếu có giá trị m[i] nào đó (i = k, k+1,...n–1) nhỏ hơn m[k] thì ta hoán đổi giá trị giữa m[k] và m[i] cho nhau. Sau lượt sắp xếp thứ k thì m[k–1] sẽ là phần tử có giá trị nhỏ thứ k trong mảng.

  ...

- Ở lượt sắp xếp thứ n–1 ta so sánh phần tử thứ n-1 của mảng m[n–2] với tất cả các phần tử đứng sau nó trong mảng (tức là phần tử m[n–1]). Nếu m[n–1] nhỏ hơn m[n–2] thì ta hoán đổi giá trị giữa m[n–2] và m[n-1] cho nhau. Sau lượt sắp xếp thứ n–1 thì m[n–2] sẽ là phần tử có giá trị nhỏ thứ n–2 trong mảng. Và dĩ nhiên phần tử còn lại là m[n–1] sẽ là phần tử nhỏ thứ n trong mảng (tức là phần tử lớn nhất trong mảng). Kết thúc n–1 lượt sắp xếp ta có các phần tử trong mảng đã được sắp xếp theo thứ tự tăng dần.

Cài đặt giải thuật:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int m[100];
    int n;    // n la so phan tu trong mang
    int i, j, k;
    clrscr();      // Xoa man hinh de tien theo doi
    // Nhap gia tri du lieu cho mang m
    // Truoc tien phai biet so phan tu cua mang
    printf(" Cho biet so phan tu co trong mang: ");
    scanf("%d",&n);
    // Roi lan luot nhap gia tri cho cac phan tu trong mang
        for(i  = 0;i<n;i++)
    {
        int temp;
            printf("\n Cho biet gia tri cua m[%d] = ",i);
        scanf("%d",&temp);
        m[i] = temp;
    }
    // Hien thi ra man hinh mang vua nhap vao
    printf("\n Mang truoc khi sap xep:    ");
    for(i=0;i<n;i++)
        printf("%3d",m[i]);
    // Bat dau sap xep
    for(i = 0; i<n-1;i++)
    {
        // O luot sap xep thu i+1
            for(j = i+1;j<n;j++)
        {
            // So sanh m[i] voi cac phan tu con lai
           // va doi cho khi tim thay phan tu < m[i].
            if(m[j]<m[i])
            {
                int temp;
```

```
                temp = m[j]; m[j] = m[i]; m[i] = temp;
            }
        }
        // Hien thi mang sau luot sap xep thu i+1
                printf("\n Mang o luot sap xep thu %d",i+1);
        for(k = 0;k < n ;k++)
                        printf("%3d",m[k]);
    }
    getch();     // Cho nguoi su dung an phim bat ki de ket thuc.
}
```

Kết quả thực hiện:

```
Cho biet so phan tu co trong mang: 5
Cho biet gia tri cua m[0]: 34
Cho biet gia tri cua m[1]: 20
Cho biet gia tri cua m[2]: 17
Cho biet gia tri cua m[3]: 65
Cho biet gia tri cua m[4]: 21
Mang truoc khi sap xep:   34 20 17 65 21
Mang o luot sap xep thu 1: 17 34 20 65 21
Mang o luot sap xep thu 2: 17 20 34 65 21
Mang o luot sap xep thu 3: 17 20 21 65 34
Mang o luot sap xep thu 4: 17 20 21 34 65
```

III.4.2. Con trỏ

Từ khi bắt đầu môn học cho đến giờ, chúng ta chỉ truy nhập, tác động hay thay đổi nội dung của các biến một cách trực tiếp – qua tên của chúng. Tuy nhiên, ngôn ngữ C cung cấp cho người lập trình một phương tiện gián tiếp khác đôi khi rất tiện dụng để truy xuất đến biến, đó là các con trỏ.

III.4.2.1. Khái niệm và cách khai báo con trỏ

Địa chỉ và giá trị của một biến

Bộ nhớ có thể hiểu như một dãy các byte nhớ được xác định một cách duy nhất qua một *địa chỉ*. Tất cả các biến trong chương trình được lưu ở một vùng nào đó trong bộ nhớ.

Khi chúng ta khai báo một biến, chương trình dịch sẽ cấp phát cho biến đó một số ô nhớ liên tiếp đủ để chứa nội dung của biến, ví dụ một biến ký tự được cấp phát 1 byte, một biến nguyên được cấp phát 2 byte, một biến thực được cấp phát 4 byte... Địa chỉ của một biến chính là địa chỉ của byte đầu tiên trong số đó.

Một biến luôn có hai đặc tính:

– Địa chỉ của biến.

– Giá trị của biến.

Xét ví dụ sau:

```
int i, j;
i = 3;
j = i;
```

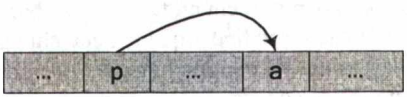
Giả sử chương trình dịch đặt biến i tại địa chỉ FFEC trong bộ nhớ, và biến j ở địa chỉ FFEE, chúng ta có

biến	địa chỉ	giá trị
i	FFEC	3
j	FFEE	3

Hai biến khác nhau có địa chỉ khác nhau. Phép gán i = j; chỉ có tác dụng trên giá trị của các biến, nó chỉ có ý nghĩa nội dung của vùng nhớ lưu trữ tương ứng biến j được sao chép sang nội dung vùng nhớ dành cho i. Hai biến i, j có kiểu nguyên, chúng được lưu trong hai byte.

Khái niệm con trỏ

Con trỏ là một biến mà giá trị của nó là địa chỉ của một vùng nhớ.Vùng nhớ này có thể chứa các biến thuộc các kiểu dữ liệu cơ sở như int, char, hay double hoặc dữ liệu có cấu trúc như mảng.



Cú pháp khai báo một con trỏ

*Kiểu_dữ_liệu *tên_contrỏ;*

Câu lệnh này khai báo một định danh tên_contrỏ gắn với một biến con trỏ có giá trị là địa chỉ của một biến có kiểu chỉ định. Ký tự * xác định rằng một biến con trỏ đang được khai báo. Giá trị của biến con trỏ hoàn toàn có thể thay đổi được.

Kiểu của một con trỏ phụ thuộc vào kiểu biến mà nó trỏ đến. Trong ví dụ sau, ta định nghĩa con trỏ p trỏ đến biến nguyên i:

```
int i = 3;
int *p;
p = &i;
```

Khi gán địa chỉ của i cho p, ta nói rằng p trỏ đến i. Giá trị của các biến này trong bộ nhớ sẽ là:

biến	địa chỉ	giá trị
i	FFEC	3
p	FFEE	FFEC

Chú ý: : Một con trỏ chỉ có thể trỏ tới một đối tượng cùng kiểu, không thể dùng con trỏ floaat để trỏ vào một biến kiểu int hay ngược lại. Tuy vậy, trong một số trường hợp ép p kiểu đặc biệt, quy tắc này có thể bị phá vỡ.

II.4.2.2. Toán tử & và *

Có hai i toán tử đặc biệt được dùng với con trỏ: & và *. Toán tử & là một toán tử một ngôi và nó trả về địa chỉ của biến. Toán tử thứ hai, toán tử * là một toán tử một ngôi và trả về giá trị chứa trong vùng nhớ được trỏ bởi giá trị của biến con trỏ. Trong v ví dụ sau, chương trình:

```
#inoclude<stdio.h>
voioid main()
{
        int i = 3;
        int *p;
        p = &i;
       printf("*p = %d \n",*p);
}
```

iển thịị ra màn hình *p = 3. Trong chương trình này i và *p là tương đương. Mọi hay đổổi trên *p sẽ thay đổi luôn i. Ví dụ, nếu ta gán *p = 0 thì i sẽ có giá trị là 0.

ả hai t toán tử * và & có độ ưu tiên cao hơn tất cả các toán tử số học ngoại trừ toán t đảo c dấu. Chúng có cùng độ ưu tiên với toán tử lấy giá trị âm.

Gán giá trị cho con trỏ

Một cách tổng quát, một biến con trỏ có thể được gán bởi địa chỉ của một biến khác:

```
pointer_variable = &variable;
```

Hay bởi giá trị của một con trỏ khác (tốt nhất là cùng kiểu):

```
pointer_variable2 = pointer_variable;
```

Giá trị NULL cũng có thể được gán đến một biến con trỏ bằng số 0 như sau:

```
pointer_variable = 0;
```

Và cuối cùng, các biến cũng có thể được gán giá trị thông qua con trỏ của chúng.

```
*pointer_variable = 10;
```

Nói chung, các biểu thức có chứa con trỏ cũng theo cùng quy luật như các biểu thức khác trong C. Điều quan trọng cần chú ý là phải gán giá trị cho biến con trỏ trước khi sử dụng chúng, nếu không chúng có thể trỏ đến một giá trị không xác định nào đó. Trong chương trình chúng ta có thể thao tác đồng thời trên con trỏ p và *p, nhưng ý nghĩa của chúng là rất khác nhau. Quan sát hai chương trình sau:

```
void main()
{
        int i = 3, j = 6;
        int *p1, *p2;
        p1 = &i;
        p2 = &j;
        *p1 = *p2;
}
```

và:

```
void main()
{
        int i = 3, j = 6;
        int *p1, *p2;
        p1 = &i;
        p2 = &j;
        p1 = p2;
}
```

Trước lệnh gán cuối cùng, giả sử giá trị các biến trong bộ nhớ như sau:

biến	địa chỉ	giá trị
i	FFEC	3
j	FFEE	6
p1	FFDA	FFEC
p2	FFDC	FFEE

Sau lệnh gán *p1 = *p2; ở chương trình thứ nhất:

biến	địa chỉ	giá trị
i	FFEC	6
j	FFEE	6
p1	FFDA	FFEC
p2	FFDC	FFEE

Trong khi đó lệnh gán p1 = p2 trong chương trình thứ hai sẽ dẫn tới kết quả sau:

biến	địa chỉ	giá trị
i	FFEC	3
j	FFEE	6
p1	FFDA	FFEE
p2	FFDC	FFEE

III.4.2.3. Các phép toán trên con trỏ

Một điểm mạnh của ngôn ngữ C là khả năng thực hiện tính toán trên các con trỏ. Các phép toán số học có thể thực hiện trên con trỏ là:

- Cộng con trỏ với một số nguyên (int, long) và kết quả là một con trỏ cùng kiểu.
- Trừ con trỏ với một số nguyên và kết quả là một con trỏ cùng kiểu.
- Trừ hai con trỏ cùng kiểu cho nhau, kết quả là một số nguyên. Kết quả này nói lên khoảng cách (số phần tử thuộc kiểu dữ liệu của con trỏ) ở giữa hai con trỏ.

Chú ý là phép toán cộng hai con trỏ và nhân chia, lấy phần dư trên con trỏ là không hợp lệ. Mỗi khi con trỏ được tăng giá trị, nó sẽ trỏ đến ô nhớ của phần tử kế tiếp. Mỗi khi được giảm giá trị, nó sẽ trỏ đến vị trí của phần tử đứng trước nó.

Xét ví dụ sau:

```
int x, *p1, *p2;
p1= &x;
p2= p1+1;
```

Khi đó p2 sẽ trỏ tới số nguyên nằm ngay kề sau x trong bộ nhớ. Cũng phải chú ý rằng mặc dù 1 chỉ là khoảng cách tương đối tính theo số phần tử kiểu int giữa p1 và p2, còn thực tế giá trị địa chỉ theo byte của p2 hơn p1 là 2. Nếu p2, p1 là con trỏ kiểu float thì khoảng cách sẽ là 4, chính là kích thước của kiểu dữ liệu trỏ bởi con trỏ.

III.4.2.4. Con trỏ void

Được khai báo như sau:

```
void  *con_trỏ;
```

Đây là con trỏ đặc biệt, con trỏ không có kiểu, nó có thể nhận giá trị là địa chỉ của một biến thuộc bất kỳ kiểu dữ liệu nào. Con trỏ void được dùng làm đối số để nhận bất kỳ địa chỉ nào từ tham số của các lời gọi hàm. Các lệnh sau đây là hợp lệ:

```
void *p, *q;
int x = 21;
float y = 34.34;
p = &x; q = &y;
```

III.4.2.5. Mối quan hệ giữa con trỏ và mảng một chiều

Với một mảng một chiều có tên là a thì a là một địa chỉ và a có giá trị bằng &a[0].

Như vậy nếu ta khởi tạo một con trỏ p với giá trị bằng địa chỉ của một mảng, ta có thể dùng con trỏ này để duyệt qua các phần từ trong mảng.

Xét khai báo mảng tab gồm 10 số nguyên sau:

```
int a[10], *p;
```

nếu p = a; khi đó:

p+1 sẽ trỏ tới phần tử cùng kiểu ngay sau phần tử đầu tiên của mảng, chính là a[1], nghĩa là *(p+1) chính là a[1].

p+2 sẽ trỏ tới a[2], hay *(p+2) là a[2]

....

p+i sẽ trỏ tới a[i]

Ví dụ sau đây minh họa việc sử dụng con trỏ để duyệt mảng một chiều:

```
#include<stdio.h>
#define N 5
int tab[5] = {1, 2, 6, 0, 7};
void main()
{
    int i;
    int *p;
    p = tab;
    for (i = 0; i < N; i++)
     {
            printf(" %d \n",*p);
            p++;
     }
}
```

Ngoài ký pháp *(p+i) để chỉ nội dung phần tử thứ i tính từ đầu mảng (trỏ bởi p), ngôn ngữ C còn cho phép chúng ta sử dụng chỉ số i trên con trỏ. Cụ thể là p[i] có vai trò như *(p+i) và cũng là phần tử thứ i của mảng được trỏ bởi p. Ví dụ trước có thể được viết lại như sau:

```
#include<stdio.h>
#define N 5
int tab[5] = {1, 2, 6, 0, 7};
main()
{
    int i;
    int *p;
    p = tab;
    for (i = 0; i < N; i++)
       printf(" %d \n", p[i]);
}
```

Lưu ý:

- Con trỏ luôn cần phải được khởi tạo, hoặc bằng cách gán cho nó một địa chỉ nào đó, hoặc qua thao tác cấp phát động bộ nhớ.
- Một (tên) mảng là một hằng địa chỉ, nó không bao giờ có thể nằm ở vế trái của một phép gán như con trỏ, cũng như chấp nhận các phép toán số học trên nó, ví dụ như tab++;

III.4.3. Xâu ký tự

III.4.3.1. Khái niệm xâu ký tự

Xâu kí tự (*string*): là một dãy các kí tự viết liên tiếp nhau.

Xâu rỗng: là xâu không gồm kí tự nào cả.

Độ dài xâu: là số kí tự có trong xâu.

Biểu diễn xâu kí tự: Xâu kí tự được biểu diễn bởi dãy các kí tự đặt trong cặp dấu ngoặc kép. Các kí tự nằm trong cặp dấu ngoặc kép là nội dung của xâu.

Ví dụ:

- "String" là một xâu kí tự gồm 6 kí tự: 'S', 't', 'r', 'i', 'n', 'g' được viết liên tiếp nhau.
- "Tin hoc" là một xâu kí tự gồm 7 kí tự: 'T', 'i', 'n', dấu cách (' '), 'h', 'o', và 'c'.

Lưu trữ dữ liệu kiểu xâu kí tự: Các kí tự của xâu được lưu trữ kế tiếp nhau và kết thúc bằng kí tự kết thúc xâu (kí tự '\0' hay kí tự NUL, có số thứ tự 0 trong bảng mã ASCII). Nhờ có kí tự NUL mà người ta xác định được độ dài của xâu kí tự bằng cách đếm các kí tự có trong xâu đến khi gặp kí tự NUL (kí tự NUL không được tính vào độ dài xâu).

Ví dụ: Xâu kí tự "Tin hoc" sẽ được lưu trữ như sau:

'T'	'i'	'n'	' '	'h'	'o'	'c'	'\0'

Lưu ý:

- Xâu kí tự khác mảng kí tự ở chỗ xâu kí tự có kí tự kết thúc xâu (kí tự NUL hay '\0') trong khi mảng kí tự không có kí tự kết thúc.
- Phân biệt giữa một kí tự và xâu kí tự có một kí tự: Ví dụ 'A' là một kí tự, nó được lưu trữ trong 1 byte, còn "A" là xâu kí tự, nó được lưu trữ trong 2 byte, trong đó byte đầu tiên lưu trữ kí tự 'A', byte thứ hai lưu trữ kí tự kết thúc xâu (NUL).

III.4.3.2. Khai báo và sử dụng xâu

a. Khai báo xâu kí tự

Trong C, một xâu kí tự được khai báo với cú pháp như sau:

char tên_xâu [số_kí_tự_tối_đa];

Trong đó số_kí_tự_tối_đa cho biết số lượng kí tự nhiều nhất có thể có trong xâu.

Sau khi khai báo, biến xâu kí tự tên_xâu có thể được dùng để lưu trữ một xâu kí tự bất kì, miễn là độ dài xâu kí tự (số kí tự có trong xâu) đó không vượt quá giá trị số_kí_tự_tối_đa.

Ví dụ:

char ho_va_ten[20];

Đây là khai báo của một biến xâu kí tự tên là ho_va_ten, biến này có thể có tối đa 20 kí tự.

Lưu ý: Đôi khi ta vẫn có thể nhập một xâu có nhiều hơn 20 kí tự cho xâu ho_va_ten mà trình biên dịch C vẫn không báo lỗi, tuy nhiên cần tránh điều này vì khi chạy chương trình thì chương trình quản lí bộ nhớ của hệ điều hành sẽ bắt lỗi và buộc chương trình kết thúc.

b. Truy nhập vào một phần tử của xâu

Có thể truy nhập đến từng phần tử của xâu tương tự như truy nhập đến từng phần tử của mảng. Cú pháp sử dụng để truy nhập là:

tên_xâu[chỉ_số_của_kí_tự_cần_truy_nhập]

Ví dụ: Ta đã có khai báo char que_quan[10], và giả sử xâu que_quan có nội dung là "Ha Noi". Khi đó ta có thể hình dung xâu kí tự que_quan như sau :

Phần tử thứ	Chỉ số của phần tử	Tên của phần tử	Nội dung lưu trữ
1	0	que_quan[0]	'H'
2	1	que_quan[1]	'a'
3	2	que_quan[2]	' '
4	3	que_quan[3]	'N'
5	4	que_quan[4]	'o'
6	5	que_quan[5]	'i'

7	6	que_quan[6]	'\0'
8	7	que_quan[7]	
9	8	que_quan[8]	
10	9	que_quan[9]	

III.4.3.3. Các hàm xử lý ký tự

Lưu ý: Để sử dụng các hàm này ta khai báo tệp tiêu đề ctype.h.

Hàm toupper():

int toupper(int ch)

Hàm toupper() dùng để chuyển một kí tự chữ cái thường (các kí tự 'a', 'b', …, 'z') thành kí tự chữ cái hoa tương ứng ('A', 'B', …, 'Z').

Hàm tolower():

int tolower(int ch)

Hàm tolower() dùng để chuyển một kí tự chữ cái hoa ('A', 'B', …, 'Z') thành kí tự chữ cái thường tương ứng ('a', 'b', …'z').

Hàm isalpha():

int isalpha(int ch)

Hàm isalpha() dùng để kiểm tra một kí tự có phải là chữ cái hay không ('a', 'b', …, 'z', 'A', 'B', …, 'Z'). Hàm trả về giá trị khác không nếu đúng là chữ cái, trả về giá trị 0 nếu ngược lại.

Hàm isdigit():

int isdigit(int ch)

Hàm isdigit() dùng để kiểm tra một kí tự có phải là chữ số hay không ('0', '1', …'9'). Hàm trả về giá trị khác không nếu đúng, trả về giá trị 0 nếu ngược lại.

Hàm islower():

int islower(int ch)

Hàm islower() dùng để kiểm tra một kí tự có phải là chữ cái thường hay không ('a', 'b', …'z'). Hàm trả về giá trị khác không nếu đúng, trả về giá trị 0 nếu ngược lại.

Hàm isupper():

int isupper(int ch)

Hàm isupper() dùng để kiểm tra một kí tự có phải là chữ cái hoa hay không ('A', 'B', ...'Z'). Hàm trả về giá trị khác không nếu đúng, trả về giá trị 0 nếu ngược lại.

Hàm iscntrl():

int iscntrl(int ch)

Hàm iscntrl() dùng để kiểm tra một kí tự có phải là kí tự điều khiển hay không (là các kí tự không hiển thị được và có mã ASCII từ 0 đến 31). Hàm trả về giá trị khác không nếu đúng, trả về giá trị 0 nếu ngược lại.

Hàm isspace():

int isspace(int ch)

Hàm isspace() dùng để kiểm tra một kí tự có phải là dấu cách (space, mã ASCII là 32), kí tự xuống dòng ('\n', mã ASCII là 10), kí tự về đầu dòng ('\r', mã ASCII là 13), dấu tab ngang ('\t', mã ASCII là 9) hay dấu tab dọc ('\v', mã ASCII là 11) hay không. Hàm trả về giá trị khác không nếu đúng, trả về giá trị 0 nếu ngược lại.

III.4.3.4. Các hàm xử lý xâu

Vào ra dữ liệu

Vào ra dữ liệu trên xâu kí tự tức là nhập dữ liệu cho xâu và hiển thị dữ liệu chứa trong xâu.

Để nhập dữ liệu cho xâu ta có thể sử dụng hai hàm scanf() hoặc gets()

scanf("%s",tên_xâu);

gets(tên_xâu);

Để hiển thị nội dung xâu ta có thể dùng hai hàm printf() hoặc puts()

printf("%s",tên_xâu);

puts(tên_xâu);

Các hàm scanf(), gets, printf(), puts() được khai báo trong tệp tiêu đề stdio.h.

Ví dụ: Nhập vào một chuỗi và hiển thị trên màn hình chuỗi vừa nhập.

#include<conio.h>

```
#include<stdio.h>
#include<string.h>
int main(){
char Ten[12];
printf("Nhap chuoi: ");gets(Ten);
printf("Chuoi vua nhap: ");puts(Ten);
getch();
return 0;
}
```

Một số hàm xử lí xâu kí tự khác

Hàm strlen():

size_t strlen(char* tên_xâu);

Hàm trả về độ dài (số kí tự có trong xâu) của xâu kí tự tên_xâu.

Hàm strcpy():

char* strcpy(char* xâu_đích, char* xâu_nguồn);

Hàm này sẽ sao chép nội dung xâu_nguồn và ghi lên xâu_đích.

Hàm strcmp():

int strcmp(char* xâu_thứ_nhất, char* xâu_thứ_hai);

Hàm strcmp trả về:

– giá trị < 0 nếu xâu_thứ_nhất nhỏ hơn xâu_thứ_hai

– giá trị 0 nếu xâu_thứ_nhất bằng xâu_thứ_hai

– giá trị > 0 nếu xâu_thứ _nhất lớn hơn xâu_thứ_hai

Hàm strcat():

char* strcat(char* xâu_đích, char* xâu_nguồn);

Hàm strcat sẽ ghép nối xâu_nguồn vào ngay sau xâu_đích. Kết quả trả về của hàm strcat() là xâu mới ghép nối từ hai xâu: xâu_nguồn và xâu_đích.

Hàm strchr():

char* strchr(char* str, int ch);

Hàm strchr() dùng để tìm kiếm vị trí của kí tự ch trong xâu str. Nếu có kí tự ch trong str thì hàm strchr() trả về con trỏ trỏ tới kí tự ch đầu tiên trong str, ngược lại nó sẽ trả về con trỏ NULL.

Hàm strstr():

char* strstr(char* str1, char* str2);

Hàm strstr() dùng để tìm kiếm vị trí của xâu con str2 trong xâu str1. Nếu str2 là xâu con của str1 thì hàm strstr() trả về con trỏ trỏ tới kí tự đầu tiên của xâu con str2 đầu tiên trong str1, ngược lại nó sẽ trả về con trỏ NULL.

Hàm atoi():

int atoi(char* str);

Hàm atoi() dùng để chuyển một xâu kí tự là biểu diễn của một số nguyên thành số nguyên tương ứng. Nếu chuyển đổi thành công, hàm atoi() trả về giá trị số nguyên chuyển đổi được, ngược lại trả về giá trị 0.

Hàm atol():

long int atol(char* str);

Hàm atol() dùng để chuyển một xâu kí tự là biểu diễn của một số nguyên dài (tương ứng với kiểu dữ liệu **long int**) thành số nguyên dài tương ứng. Nếu chuyển đổi thành công, hàm atol() trả về giá trị số nguyên chuyển đổi được, ngược lại trả về giá trị 0.

Hàm atof():

double atof(char* str);

Hàm atof() dùng để chuyển một xâu kí tự là biểu diễn của một số thực (ở cả dạng số dấu phảy tĩnh và động đều được) thành số thực tương ứng. Nếu chuyển đổi thành công, hàm atof() trả về giá trị số thực chuyển đổi được, ngược lại trả về giá trị 0.

Các hàm strcpy(), strlen(),strcmp(), strcat(), strchr(), strstr() khai báo trong tệp tiêu đề string.h.

Các hàm atoi(), atol(), atof() khai báo trong tệp tiêu đề stdlib.h.

Ví dụ minh họa:

```
#include <stdio.h>
#include <conio.h>
```

```
#include <string.h>     // Phai co thu vien string.h thi moi
                        // su dung duoc cac ham strcpy, strcmp...
void main(){
    char str1[10] = "abc";
    char str2[10] = "def";
    clrscr();
    printf(" str1: %s",str1);
    printf("\n str2: %s",str2);
    printf("\n strcmp(str1,str2) = %d",strcmp(str1,str2));
    printf("\n strcat(str1,str2) = %s",strcat(str1,str2));
    printf("\n str1: %s",str1);
    printf("\n str2: %s",str2);
    printf("\n strcpy(str1,str2) = %s",strcpy(str1,str2));
    printf("\n str1: %s",str1);
    printf("\n str2: %s",str2);
    strcpy(str1,"ab");
    strcpy(str2,"abc");
    printf("\n strcmp(str1,str2) = %d",strcmp(str1,str2));
    getch();
}
```

Kết quả:

```
str1: abc
str2: def
strcmp(str1,str2) = -3
strcat(str1,str2) = abcdef
str1: abcdef
str2: def
strcpy(str1,str2) = def
str1: def
str2: def
strcmp(str1,str2) = -3
```

Cần lưu ý khi sử dụng các hàm strcat(), strcpy(),.. là kích thước bộ nhớ lưu trữ dành cho xâu đích phải đủ để chứa kết quả thu được sau lời gọi các hàm trên.

III.5. Cấu trúc

III.5.1. Khái niệm cấu trúc

Kiểu dữ liệu cấu trúc (**struct**) là kiểu dữ liệu phức hợp bao gồm nhiều thành phần, mỗi thành phần có thể thuộc những kiểu dữ liệu khác nhau.

Các thành phần dữ liệu trong cấu trúc được gọi là các trường dữ liệu (*field*).

Ví dụ: Khi cần lưu giữ thông tin về một dạng đối tượng nào đó như đối tượng sinh viên chẳng hạn, ta lưu giữ các thông tin liên quan đến sinh viên như họ tên, tuổi, kết quả học tập... Mỗi thông tin thành phần lại có kiểu dữ liệu khác nhau như họ tên có kiểu dữ liệu là xâu kí tự, tuổi có kiểu dữ liệu là số nguyên, kết quả học tập có kiểu dữ liệu là số thực.

III.5.2. Khai báo và sử dụng cấu trúc

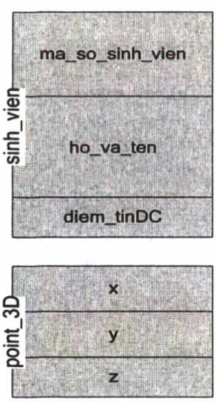
III.5.2.1. Khai báo kiểu dữ liệu cấu trúc

Để khai báo một kiểu dữ liệu cấu trúc ta dùng cú pháp khai báo sau:

```
struct tên_cấu_trúc
{
        <khai báo các trường dữ liệu>;
};
```

Ví dụ:

```
struct sinh_vien
{
   char ma_so_sinh_vien[10];
   char ho_va_ten[30];
   float diem_TinDC;
}
struct point_3D
{
   float x;
   float y;
   float z;
}
```



Khai báo thứ nhất là khai báo của một kiểu dữ liệu cấu trúc có tên là sinh_vien gồm có ba trường dữ liệu là ma_so_sinh_vien kiểu xâu kí tự, ho_va_ten kiểu xâu kí tự và diem_TinDC kiểu số thực **float**.

Ở khai báo thứ hai, ta đã khai báo một kiểu dữ liệu cấu trúc có tên là point_3D gồm có ba trường và cả ba trường này đều có cùng kiểu dữ liệu số thực **float**. Vì ba trường này cùng kiểu dữ liệu nên ta có thể sử dụng mảng để thay thế cấu trúc, tuy nhiên việc sử dụng cấu trúc để mô tả dữ liệu điểm ba chiều sẽ giúp sự mô tả được tự nhiên hơn, "thật" hơn. Như vậy, trong cấu trúc các tọa độ vẫn độc lập với nhau, nhưng nhìn từ bên ngoài thì các tọa độ này lại là một thể thống nhất cung cấp thông tin về vị trí của một điểm. Nếu sử dụng mảng thì các tọa độ của một điểm độc lập và rời rạc với nhau, ta không thấy mối liên hệ giữa chúng. Khi đó, ta sẽ phải tự mình ngầm quy ước rằng các phần tử của mảng có chỉ số là $3*k$, $3*k+1$ và $3*k+2$ với $k = 0, 1, 2, \ldots$ là tọa độ của một điểm.

III.5.2.2. Khai báo biến cấu trúc

Để khai báo biến cấu trúc ta dùng cú pháp khai báo sau:

struct tên_cấu_trúc tên_biến_cấu_trúc;

Ví dụ:

```
struct sinh_vien a, b, c;
```

Câu lệnh trên khai báo ba biến lần lượt tên là a, b, c có kiểu dữ liệu là cấu trúc sinh_vien.

Tuy nhiên ta cũng có thể kết hợp đồng thời vừa khai báo kiểu dữ liệu cấu trúc vừa khai báo biến cấu trúc bằng cách sử dụng cú pháp khai báo sau:

```
struct [tên_cấu_trúc]
{
    <khai báo các trường>;
} tên_biến_cấu_trúc;
```

Theo cú pháp khai báo trên thì ta thấy phần tên_cấu_trúc có thể có hoặc không. Nếu có tên_cấu_trúc thì sau này ta có thể khai báo bổ sung biến có kiểu dữ liệu là tên_cấu_trúc đó, còn nếu không có tên_cấu_trúc thì cấu trúc khai báo tương ứng không được sử dụng về sau.

Ví dụ:

```
struct diem_thi
{
    float diem_Toan;
    float diem_Ly;
    float diem_Hoa;
}
struct thi_sinh
{
    char SBD[10];      // so bao danh
    char ho_va_ten[30];
    struct diem_thi ket_qua;
} thi_sinh_1, thi_sinh_2;
```

Qua ví dụ trên ta cũng thấy rằng các cấu trúc có thể lồng nhau, nghĩa là cấu trúc này có thể là trường dữ liệu của cấu trúc khác và mức độ lồng là không hạn chế. Để tăng thêm sự tiện lợi, ngôn ngữ C còn cho phép khai báo trực tiếp trường dữ liệu là cấu trúc bên trong cấu trúc chứa nó, vì thế ta có thể viết lại ví dụ ở trên như sau:

```
struct thi_sinh
{
    char SBD[10];
    char ho_va_ten[30];
    struct diem_thi
      {
          float diem_Toan;
          float diem_Ly;
          float diem_Hoa;
    }ket_qua;
} thi_sinh_1, thi_sinh_2;
```

II.5.2.3. Định nghĩa kiểu dữ liệu cấu trúc với typedef

Trong các ví dụ trước ta thấy một khai báo biến cấu trúc bắt đầu bằng từ khóa **truct**, sau đó đến tên cấu trúc rồi mới đến tên biến. Cách khai báo này có phần rắc rối" hơn so với khai báo biến thông thường và có không ít trường hợp người ập trình quên đặt từ khóa **struct** ở đầu. Để tránh điều đó, ngôn ngữ C cho phép ta ặt tên mới cho kiểu dữ liệu cấu trúc bằng câu lệnh **typedef** có cú pháp khai báo hư sau:

typedef struct tên_cũ tên_mới;

Hoặc ta có thể đặt lại tên cho cấu trúc ngay khi khai báo bằng cú pháp:

typedef struct [tên_cũ]

{

<khai báo các trường>;

}danh_sách_các_tên_mới;

Sau câu lệnh này ta có thể sử dụng tên_mới thay cho tổ hợp struct tên_cũ khi khai báo biến.

Lưu ý: Được phép đặt tên_mới trùng với tên_cũ.

Ví dụ:

```
    struct point_3D
{
    float x, y, z;
} P;
struct point_3D M;
typedef struct point_3D point_3D;
point_3D N;
```

Trong ví dụ trên ta đã đặt lại tên cho cấu trúc struct point_3D thành point_3D và dùng tên mới này làm kiểu dữ liệu cho khai báo của biến N. Các biến P, M được khai báo theo cách chúng ta đã biết.

Ví dụ:

```
typedef struct point_2D
{
    float x, y;
}point_2D, diem_2_chieu, ten_bat_ki;
point_2D X;
diem_2_chieu Y;
ten_bat_ki Z;
```

Với ví dụ này ta cần chú ý là point_2D, diem_2_chieu và ten_bat_ki không phải là tên biến mà là tên mới của cấu trúc struct point_2D. Các biến X, Y, Z được khai báo với kiểu dữ liệu là các tên mới này.

III.5.3. Xử lý dữ liệu cấu trúc

III.5.3.1. Truy nhập các trường dữ liệu của cấu trúc

Dữ liệu của một biến cấu trúc bao gồm nhiều trường dữ liệu, và các trường dữ liệu này độc lập với nhau. Muốn thay đổi nội dung dữ liệu bên trong một biến cấu trúc, ta cần truy nhập tới từng trường và thực hiện thao tác cần thiết trên từng trường đó. Để truy nhập tới một trường trong cấu trúc ta dùng cú pháp sau:

tên_biến_cấu_trúc.tên_trường

Dấu chấm *"."* sử dụng trong cú pháp trên là toán tử truy nhập thành phần cấu trúc, Nếu trường được truy nhập lại là một cấu trúc, thì ta có thể tiếp tục áp dụng toán tử này để truy nhập tới các trường thành phần nằm ở mức sâu hơn.

Giờ đây ta có thể "đối xử" tên_biến_cấu_trúc.tên_trường giống như một biến thông thường có kiểu dữ liệu là kiểu dữ liệu của tên_trường, tức là ta có thể nhập giá trị, hiển thị giá trị của biến cấu trúc, sử dụng giá trị đó trong các biểu thức...

Ví dụ:

```
// duoi day la mot cau truc mo ta mot diem trong khong gian 2 chieu.
// cac truong du lieu gom: ten cua diem va toa do cua diem do.
// toa do la mot cau truc gom 2 truong: hoanh do va tung do
#include <stdio.h>
#include <conio.h>
void main()
{
    struct point_2D
   {
       char ten_diem;
       struct
       {
           float x, y;
       } toa_do;
   } p;
   float temp_float;
   char temp_char;
   printf("\n Hay nhap thong tin ve mot diem");
   printf("\n Ten cua diem: ");
       fflush(stdin);
   scanf("%c",&temp_char);
```

```
        p.ten_diem = temp_char;
    printf("\n nhap vao hoanh do cua diem: ");
        scanf("%f",&temp_float);
    p.toa_do.x = temp_float;
        // gia su diem dang xet nam tren duong thang y = 3x + 2;
    p.toa_do.y = 3*p.toa_do.x + 2;
    printf("\n %c = (%5.2f,%5.2f)",p.ten_diem, p.toa_do.x, p.toa_do.y);
    getch();
}
```

Kết quả khi chạy chương trình:

```
Hay nhap thong tin ve mot diem
Ten cua diem: A
nhap vao hoanh do cua diem: 5
A = ( 5.00,17.00)
```

Lưu ý: Cũng như việc nhập giá trị cho các phần tử của mảng, việc nhập giá trị cho các trường của cấu trúc (đặc biệt là các trường có kiểu dữ liệu **float**) nên thực hiện qua biến trung gian để tránh những tình huống có thể làm treo máy hoặc kết quả nhập được không như ý.

III.5.3.2. Phép gán giữa các biến cấu trúc

Giả sử ta có hai biến cấu trúc *a* và *b* có cùng kiểu dữ liệu là một cấu trúc nào đó và giả sử các trường dữ liệu của *a* đều đã được khởi tạo (tức là giá trị của các trường dữ liệu đó đều đã được xác định). Giờ đây ta cũng muốn giá trị các trường dữ liệu của *b* có giá trị giống với các trường dữ liệu tương ứng của *a*. Ta có thể thực hiện điều đó bằng cách gán giá trị từng trường của *a* cho các trường tương ứng của *b*. Cách này có vẻ rất "thủ công" và rất bất tiện nếu như trong cấu trúc có nhiều trường dữ liệu. Do vậy C cung cấp cho ta một phương tiện để thực hiện việc này theo cách thứ hai, đó là sử dụng phép gán các biến cấu trúc. Phép gán cấu trúc có cú pháp tương tự như phép gán thông thường.

biến_cấu_trúc_1 = biến_cấu_trúc_2;

Câu lệnh trên sẽ gán giá trị của các trường trong biến_cấu_trúc_2 cho các trường tương ứng trong biến_cấu_trúc_1.

Ví dụ:

```
#include <stdio.h>
```

```
#include <conio.h>
#include <string.h>
void main()
{
    struct s
    {
        char ho_ten[20];
        float diem;
    }a, b, c;
    float temp_f;
    printf("\na.ho_ten: ");fflush(stdin);
    gets(a.ho_ten);
    printf("\na.diem = ");scanf("%f",&temp_f);
    a.diem = temp_f;
    strcpy(c.ho_ten, a.ho_ten);
    c.diem = a.diem;
    b = a;
    printf("\na: %20s %5.2f", a.ho_ten, a.diem);
    printf("\nb: %20s %5.2f", b.ho_ten, b.diem);
    printf("\nc: %20s %5.2f\n", c.ho_ten, c.diem);
}
```

Kết quả thực hiện

```
a.ho_ten: nguyen van minh
a.diem = 7.5
a:    nguyen van minh  7.50
b:    nguyen van minh  7.50
c:    nguyen van minh  7.50
```

Trong chương trình trên ta đã nhập giá trị cho các trường của biến cấu trúc a từ bàn phím, sau đó chép dữ liệu từ biến a sang biến c bằng cách sao chép từng trường và chép dữ liệu từ biến a sang biến b bằng cách dùng lệnh gán. Kết quả là như nhau và rõ ràng cách thứ hai ngắn gọn hơn.

Lưu ý : Để chép dữ liệu là xâu kí tự ta phải dùng lệnh strcpy(), không được dùng lệnh gán thông thường để chép nội dung xâu kí tự.

III.6. Hàm

III.6.1. Khái niệm hàm

III.6.1.1. Khái niệm chương trình con

Trong khi lập trình chúng ta thường gặp những đoạn chương trình lặp đi lặp lại nhiều lần ở những chỗ khác nhau. Để tránh rườm rà và tiết kiệm thời gian, chương trình đó được thay thế bằng các chương trình con tương ứng và khi cần ta chỉ việc gọi những chương trình con đó ra mà không phải viết lại cả đoạn chương trình đó.

Lấy ví dụ khi giải các bài toán lượng giác ta thường xuyên cần phải tính giá trị sin của biến lượng giác x nào đó. Như vậy ta nên lập một chương trình con tên là sin và tham số là x để tính giá trị sin(x). Mỗi khi cần tính toán giá trị sin của một biến y nào đó, ta chỉ cần gọi chương trình con sin đã lập sẵn và truyền giá trị biến y làm tham số. Khi đó ta vẫn thu được kết quả mong muốn mà không phải viết lại cả đoạn chương trình tính giá trị sin(y).

Bên cạnh chương trình con sin còn có rất nhiều chương trình con khác được tạo sẵn như cos, exp (dùng để tính lũy thừa cơ số e), pow (tính lũy thừa), sqrt (tính căn bậc 2), ... giúp người lập trình tính toán giá trị của các đại lượng thông dụng. Những chương trình con này nằm trong *thư viện các chương trình con mẫu* và được trình biên dịch C quản lý, vì vậy chúng còn được gọi là các chương trình con chuẩn. Trình biên dịch Turbo C++ phân loại và đặt khai báo các chương trình con chuẩn này trong các đơn vị chương trình khác nhau dưới dạng các tệp tiêu đề như stdio.h, conio.h, math.h, string.h...

Ngoài ra còn có một lý do khác cần đến chương trình con. Khi ta giải quyết một bài toán lớn thì chương trình của ta có thể rất lớn và dài, điều này làm cho việc sửa chữa, gỡ rối, hiệu chỉnh chương trình gặp nhiều khó khăn. Nhưng nếu ta chia bài toán lớn, phức tạp ban đầu thành các bài toán con nhỏ hơn và tương đối độc lập với nhau, rồi lập các chương trình con giải quyết từng bài toán con, cuối cùng ghép các chương trình con đó lại thành một chương trình giải quyết bài toán ban đầu thì sẽ rất tiện lợi cho việc phát triển, kiểm tra và sửa chữa cả chương trình.

Việc này tương tự như trong dây chuyền sản xuất công nghiệp, khi ta lắp ráp sản phẩm hoàn thiện từ các bán thành phẩm, các mô đun được chế tạo ở những nơi khác nhau. Vì các bán thành phẩm này được chế tạo độc lập nên khi phát hiện lỗi ở mô đun nào ta chỉ việc tìm đến nơi sản xuất ra nó để sửa chữa.

Việc chia nhỏ một chương trình thành các chương trình con đảm nhận những công việc nhỏ khác nhau chính là tư tưởng chính cho phương pháp lập trình có cấu trúc (*structured programming*).

Cần lưu ý là có khi một chương trình con chỉ sử dụng đúng một lần nhưng nó vẫn làm cho chương trình trở nên sáng sủa và dễ đọc, dễ hiểu hơn.

III.6.1.2. Phân loại chương trình con

Có hai loại chương trình con là hàm (*function*) và thủ tục (*procedure*). Sự khác nhau giữa hàm và thủ tục là ở chỗ hàm sau khi thực hiện xong thì sẽ trả về giá trị, còn thủ tục không trả về giá trị.

Mặc dù vậy hàm và thủ tục là tương đương nhau, tức là có thể xây dựng được thủ tục có chức năng tương đương với một hàm bất kì và có thể xây dựng được hàm có chức năng tương đương với một thủ tục bất kì. Vì thế có những ngôn ngữ lập trình cho phép chương trình con có thể là hàm và thủ tục (Pascal) và có những ngôn ngữ chỉ cho phép chương trình con là hàm mà thôi (như C, Java).

Lưu ý là nếu chương trình con là hàm thì nó luôn có giá trị trả về. Nếu thực sự không có giá trị gì để trả về (nghĩa là nó hoạt động giống thủ tục) thì ta phải khai báo hàm đó có kiểu giá trị trả về là "*không là kiểu giá trị nào cả*" (kiểu **void** trong C).

III.6.2. Khai báo và sử dụng hàm

II.6.2.1. Khai báo hàm

Cú pháp khai báo một hàm trong C là như sau:

[<kiểu giá trị trả về>] <tên hàm>([<danh sách tham số>,...])
Thân hàm

Khai báo của một hàm được chia làm hai phần:

– Dòng đầu hàm:

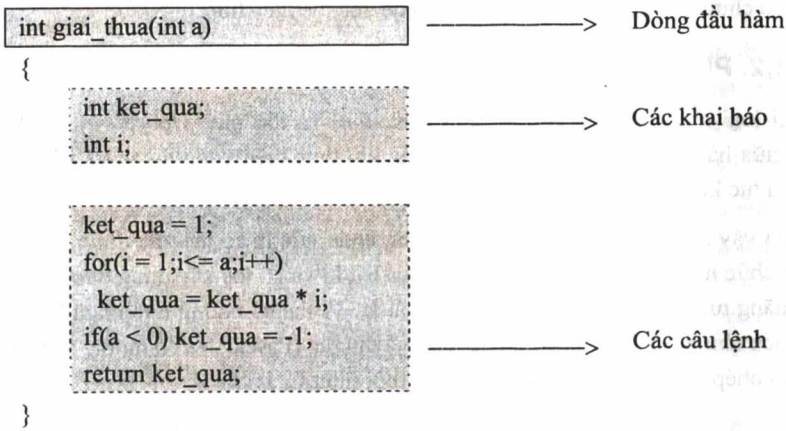
[<kiểu giá trị trả về>] <tên hàm>([<danh sách tham số>,...])

– Thân hàm: là tập hợp các khai báo và câu lệnh đặt trong cặp dấu ngoặc nhọn.

```
{
        <Các khai báo>
        ...
        <Các câu lệnh>
}
```

Trong thân hàm có ít nhất một lệnh return.

Ví dụ sau khai báo và định nghĩa hàm tính giai thừa của một số nguyên dương. Ta quy ước rằng giai thừa của một số âm thì bằng –1, của 0 bằng 1, của một số nguyên dương a là $a! = a \times (a-1) \times \ldots \times 1$.



Các thành phần của dòng đầu hàm

Dòng đầu hàm là các thông tin được trao đổi giữa bên trong và bên ngoài hàm. Khi nói tới dòng đầu hàm là nói tới tên của hàm, hàm đó cần những thông tin gì từ môi trường để hoạt động (các tham số đầu vào), hàm đó cung cấp những thông tin gì cho môi trường (những tham số đầu ra và giá trị trả về).

Dòng đầu hàm phân biệt các hàm với nhau, hay nói cách khác không được có hai hàm có dòng đầu hàm giống nhau.

Kiểu dữ liệu trả về của hàm

Thông thường hàm sau khi được thực hiện sẽ trả về một giá trị kết quả tính toán nào đó. Để sử dụng được giá trị đó ta cần phải biết nó thuộc kiểu dữ liệu gì. Kiểu dữ liệu của đối tượng tính toán được hàm trả về được gọi là kiểu dữ liệu trả về của hàm.

Trong C, kiểu dữ liệu trả về của hàm có thể là kiểu dữ liệu bất kỳ (kiểu dữ liệu có sẵn hoặc kiểu dữ liệu do người dùng tự định nghĩa) nhưng không được là kiểu dữ liệu mảng.

Nếu kiểu dữ liệu trả về là kiểu **void** thì hàm không trả về giá trị nào cả.

Trường hợp ta không khai báo kiểu dữ liệu trả về thì chương trình dịch của C sẽ ngầm hiểu rằng kiểu dữ liệu trả về của hàm là kiểu **int**.

Tên hàm

Tên hàm là có thể là bất kì một định danh hợp lệ nào, tuy nhiên tên hàm nên mang nghĩa gợi ý chức năng công việc mà hàm thực hiện. Ví dụ, một hàm có chức năng tính và trả về bình phương của một số thực *x* thì nên có tên là *binh_phuong*. Trong C, các hàm không được đặt tên trùng nhau.

Tham số của hàm

Tham số của hàm là các thông tin cần cho hoạt động của hàm và các thông tin, kết quả tính toán được hàm trả lại, tức là có những tham số chứa dữ liệu vào cung cấp cho hàm, có những tham số chứa dữ liệu ra mà hàm tính toán được.

Các tham số sử dụng trong lời khai báo hàm được gọi là tham số hình thức, nó là tham số giả định của hàm. Khi khai báo tham số hình thức, phải chỉ ra tên của tham số và kiểu dữ liệu của tham số.

Các tham số được cung cấp trong quá trình thực hiện hàm được gọi là tham số thực. Kiểu dữ liệu của tham số thực cung cấp cho hàm trong chương trình phải giống kiểu dữ liệu của tham số hình thức tương ứng với tham số thực đó và có cùng số lượng, nếu không sẽ có báo lỗi biên dịch.

Một hàm có thể có một, nhiều hoặc không có tham số nào cả, nếu có nhiều tham số thì chúng phải được phân cách với nhau bằng dấu phẩy. Lưu ý là nếu hàm không có tham số nào cả thì vẫn phải có cặp dấu ngoặc đơn sau tên hàm, ví dụ **main()**.

*Lệnh **return***

Một hàm được thực hiện khi ta gặp lời gọi hàm của hàm đó trong chương trình. Một lời gọi hàm là tên hàm theo sau bởi các tham số thực trong chương trình. Sau khi hàm thực hiện xong, nó sẽ trở về chương trình đã gọi. Có hai cách để từ hàm trở về chương trình đã gọi hàm:

- Sau khi thực hiện tất cả các câu lệnh có trong thân hàm.
- Khi gặp lệnh **return**.

Cú pháp chung của lệnh **return** là:

*return **biểu_thức**;*

Khi gặp lệnh này, chương trình sẽ tính toán giá trị của ***biểu_thức***, lấy kết quả tính toán được làm giá trị trả về cho lời gọi hàm rồi kết thúc việc thực hiện hàm, trở về chương trình đã gọi nó. Lưu ý: Kiểu của biểu_thức phải trùng với kiểu giá trị trả về trong phần khai báo hàm.

Trong lệnh **return** cũng có thể không có phần ***biểu_thức***, khi đó ta sẽ kết thúc thực hiện hàm mà không trả về giá trị nào cả.

Ví dụ:

```
#include <stdio.h>
#include <conio.h>
int max(int x, int y, int z)
{
    int max;
    max = x>y?x:y;
    max = max>z?max:z;
    return max;
}
void main()
{
    int a,b,c;
    clrscr();
    printf("\n Nhap gia tri cho 3 so nguyen a, b, c: ");
    scanf("%d %d %d",&a,&b,&c);
    printf("\n Gia tri cac so vua nhap: ");
    printf(" a = %-5d b = %-5d c = %-5d",a,b,c);
    printf("\n Gia tri lon nhat trong 3 so la %d",max(a,b,c));
    getch();
}
```

Kết quả khi thực hiện:

```
Nhap gia tri cho 3 so nguyen a, b, c: 3 12 8
Gia tri cac so vua nhap: a =  3    b = 12    c = 8
Gia tri lon nhat trong 3 so la 12
```

III.6.2.2. Sử dụng hàm

Có thể sử dụng hàm sau khi đã khai báo. Để sử dụng một hàm (hay còn nói là *gọi hàm*) ta sử dụng cú pháp sau:

<tên hàm> ([danh sách các tham số])

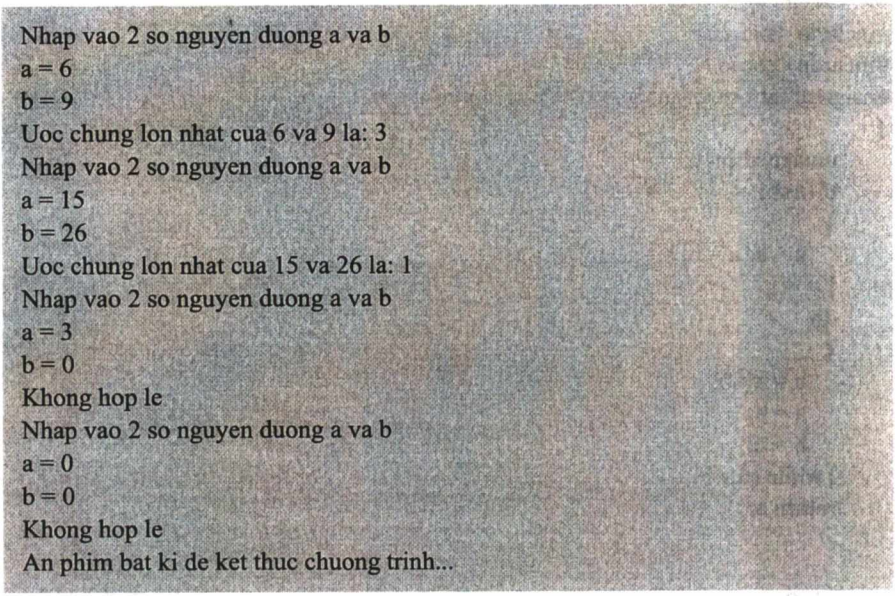
Ví dụ: Chương trình dưới đây sẽ khai báo và định nghĩa một hàm có tên là Uscln với 2 tham số đều có kiểu **unsigned int**. Hàm Uscln tìm ước số chung lớn nhất của 2 tham số này theo thuật toán *Euclid* và trả về ước số chung tìm được. Sau đó ta sẽ

gọi hàm Uscln trong hàm **main** để tìm ước số chung lớn nhất của hai số nguyên được nhập từ bàn phím.

```
#include <stdio.h>
#include <conio.h>
unsigned int Uscln(unsigned int a, unsigned int b)
{
    unsigned int u;
    if (a<b)
    {
       u = a; a = b; b = u;
    }
    do
    {
       u = a%b;
       a = b;
       b = u;
    }while (u!=0);
    return a;
}

int main()
{
    unsigned int a, b;
    do
    {
        printf("\n Nhap vao 2 so nguyen duong a va b ");
        printf("\n a = "); scanf("%d",&a);
        printf("\n b = "); scanf("%d",&b);
        if(a*b == 0)
        {
          printf("\n Khong hop le");
          continue;
        }
        printf("\n Uoc chung lon nhat cua %d va %d la: %d", a, b, Uscln(a, b));
      }while ((a != 0)||(b != 0));
    printf("\n An phim bat ki de ket thuc chuong trinh...");
    getch();
    return 0;
}
```

Kết quả khi thực hiện:



Trong chương trình, khi gặp một lời gọi hàm thì hàm bắt đầu thực hiện bằng cách chuyển các lệnh thi hành đến hàm được gọi, quá trình diễn ra như sau:

- Nếu hàm có tham số, trước tiên các tham số sẽ được *gán giá trị thực tương ứng*.
- Chương trình sẽ thực hiện tiếp các câu lệnh trong thân hàm bắt đầu từ lệnh đầu tiên đến câu lệnh cuối cùng.
- Khi gặp lệnh **return** hoặc dấu } cuối cùng trong thân hàm, chương trình sẽ thoát khỏi hàm để trở về chương trình gọi nó và thực hiện tiếp tục những câu lệnh của chương trình này.

III.6.3. Truyền tham số trong lời gọi hàm

Thông thường trong các ngôn ngữ lập trình có hai cách để truyền tham số trong lời gọi hàm, đó là truyền theo tham trị và truyền theo tham biến.

Truyền theo tham trị

Khi tham số được truyền theo tham trị, một bản sao giá trị của tham số thực được tạo ra và gán cho tham số hình thức của hàm. Vì vậy, mọi thay đổi trong hàm sẽ không ảnh hưởng đến giá trị ban đầu của biến (nếu có) nằm trong lời gọi hàm.

Truyền theo tham biến

Khi tham số được truyền theo tham biến, hàm sẽ truyền trực tiếp tham số đó (bắt buộc phải là biến) cho hàm được gọi. Trong trường hợp này, tham số hình thức và tham số thực là một. Vì vậy, mọi thay đổi trong hàm sẽ ảnh hưởng đến giá trị ban đầu của biến được truyền trong lời gọi hàm.

Truyền theo tham trị được dùng khi không cần thay đổi giá trị của biến được truyền vào trong lời gọi hàm. Trong khi đó, truyền theo tham biến chỉ nên dùng khi thật sự cần thiết thay đổi giá trị của biến được truyền vào.

Trong ngôn ngữ C, cách truyền tham số vào các hàm là truyền theo tham trị.

Dưới đây là một ví dụ về truyền tham số theo tham trị:

```
#include <stdio.h>
#include<conio.h>
void swap (int a, int b){
    int temp = a;
    a = b;
    b = temp;
}
void main(){
    int x = 3, y = 4;
    clrscr();
    printf("Truoc khi goi ham swap\n");
    printf("%3d %3d\n",x,y);
    swap(x,y);
    printf("Sau khi goi ham swap\n");
    printf("%3d %3d\n",x,y);
    getch();
}
```

Kết quả khi thực hiện:

```
Truoc khi goi ham swap
  3  4
Sau khi goi ham swap
  3  4
```

ể có thể thay đổi giá trị của tham số truyền vào, chúng ta sẽ khai báo tham số nh thức của hàm là con trỏ, khi đó tham số thực tương ứng sẽ là một địa chỉ. Khi ıyền địa chỉ của biến cho hàm thì người lập trình có thể thay đổi giá trị của biến ợc truyền vào. Khi truyền tên của mảng cho hàm, chúng ta còn tiết kiệm được ời gian sao chép dữ liệu.

ı chỉnh sửa lại chương trình trên:

```
#include <stdio.h>
#include<conio.h>
void swap (int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
void main(){
    int x = 3, y = 4;
    clrscr();
    printf("Truoc khi goi ham swap\n");
    printf("%3d %3d\n",x,y);
    swap(&x,&y);
    printf("Sau khi goi ham swap\n");
    printf("%3d %3d\n",x,y);
    getch();
}
```

ết quả khi thực hiện:

```
Truoc khi goi ham swap
   3  4
Sau khi goi ham swap
   4  3
```

III.6.4. Phạm vi của biến

Phạm vi của các biến

Một biến sau khi khai báo thì có thể được sử dụng trong chương trình. Tuy nhiên tù vào vị trí khai báo biến mà phạm vi sử dụng các biến sẽ khác nhau. Nguyên tắc s dụng biến là biến khai báo trong phạm vi nào thì được sử dụng trong phạm vi đó.

Một biến có thể được khai báo trong chương trình chính hay trong các chươn trình con hoặc thậm chí trong một khối lệnh. Nếu biến được khai báo trong mộ khối lệnh thì nó chỉ có thể được gọi trong khối lệnh đó, không thể gọi từ bên ngoà khối lệnh được. Một biến được khai báo trong một chương trình con chỉ có th được sử dụng trong phạm vi chương trình con đó. Một biến được khai báo tron chương trình chính thì có thể được sử dụng trong toàn bộ chương trình, trong tất c các chương trình con cũng như trong các khối lệnh của chương trình.

Lưu ý:

Một số ngôn ngữ lập trình như Pascal cho phép khai báo một chương trình con nằn trong một chương trình con khác, nhưng ngôn ngữ C không cho phép điều này.

Bên trong một khối lệnh thì có thể có chứa khối lệnh khác, khi đó biến được kha báo ở khối lệnh bên ngoài có thể được sử dụng ở khối lệnh bên trong.

Việc trùng tên của các biến: Trong cùng một phạm vi ta không được phép khai báo ha biến có cùng tên nhưng ta có thể khai báo hai biến trùng tên thuộc hai phạm v khác nhau. Nếu có hai biến trùng tên khai báo ở hai phạm vi khác nhau thì xảy r hai trường hợp:

– Hai phạm vi này tách rời nhau: Khi đó các biến sẽ có tác dụng ở phạm v riêng của nó, không ảnh hưởng đến nhau.

– Phạm vi này nằm trong phạm vi kia: Khi đó nếu chương trình đang ở phạm v ngoài (tức là đang thực hiện câu lệnh nằm ở phạm vi ngoài) thì biến khai báo ở phạm vi ngoài có tác dụng, còn nếu chương trình đang ở phạm vi trong (đang thực hiện câu lệnh nằm ở phạm vi trong) thì biến khai báo ở phạm vi trong sẽ có tác dụng và nó che lấp biến trùng tên ở bên ngoài.

Ví dụ:

```
#include <stdio.h>
void main()
{
        {
            int a = 1;
```

```
        printf("\n a = %d",a);
        {
            int a = 2;
            printf("\n a = %d",a);
        }
        printf("\n a = %d",a);
    }
    {
        int a = 3;
        printf("\n a = %d",a);
    }
}
```

Kết quả thực hiện chương trình:

a = 1

a = 2

a = 1

a = 3

Phân loại biến:

Theo phạm vi sử dụng, biến chia làm hai loại: biến cục bộ (biến địa phương – local variable) và biến toàn cục (global variable).

Biến địa phương

Là các biến được khai báo trong khối lệnh hoặc trong thân chương trình con. Việc khai báo các biến cục bộ phải được đặt trước phần câu lệnh trong khối lệnh hoặc trong chương trình con.

Biến toàn cục

Là biến được khai báo trong chương trình chính. Vị trí khai báo của biến toàn cục là sau phần khai báo tệp tiêu đề và khai báo hàm nguyên mẫu.

Lưu ý:

– Hàm **main()** cũng chỉ là một chương trình con, nhưng nó là chương trình con đặc biệt ở chỗ chương trình được bắt đầu thực hiện từ hàm **main()**.

– Biến khai báo trong hàm **main()** không phải là biến toàn cục mà là biến cục bộ của hàm **main()**.

Một số lệnh đặc trưng của C: **register, static**

Chúng ta biết rằng các thanh ghi có tốc độ truy nhập nhanh hơn so với các loại bộ nhớ khác (RAM, bộ nhớ ngoài), do vậy nếu một biến thường xuyên sử dụng trong chương trình được lưu vào trong thanh ghi thì tốc độ thực hiện của chương trình sẽ được tăng lên. Để làm điều này ta đặt từ khóa **register** trước khai báo của biến đó.

Ví dụ:

```
register int a;
```

Tuy nhiên có một lưu ý khi khai báo biến **register** là vì số lượng các thanh ghi có hạn và kích thước của các thanh ghi cũng rất hạn chế (ví dụ trên dòng máy 80x86, các thanh ghi có kích thước 16 bit = 2 byte) cho nên số lượng biến khai báo **register** sẽ không nhiều và thường chỉ áp dụng với những biến có kích thước nhỏ như kiểu **char**, **int**.

Như ta đã biết, một biến cục bộ khi ra khỏi phạm vi của biến đó thì bộ nhớ dành để lưu trữ biến đó sẽ được giải phóng. Tuy nhiên trong một số trường hợp ta cần lưu giá trị của các biến cục bộ này để phục vụ cho những tính toán sau này, khi đó ta hãy khai báo biến với từ khóa **static** ở đầu.

Ví dụ:

```
static int a;
```

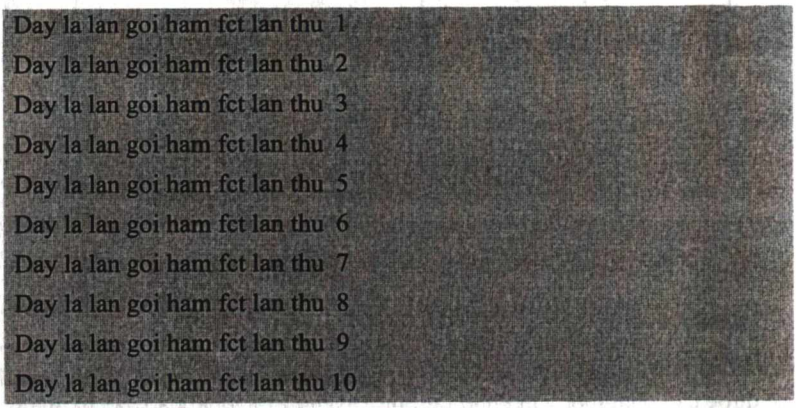
Từ khóa **static** giúp chương trình dịch biết được đây là một biến tĩnh, nghĩa là nó sẽ được cấp phát một vùng nhớ thường xuyên từ lúc khai báo và chỉ giải phóng khi chương trình chính kết thúc. Như vậy về thời gian tồn tại biến **static** rất giống với biến toàn cục, chỉ có một sự khác biệt nhỏ là biến toàn cục thì có thể truy cập ở mọi nơi trong chương trình (miễn là ở đó không có biến địa phương nào cùng tên che lấp nó), còn biến **static** thì chỉ có thể truy nhập trong phạm vi mà nó được khai báo mà thôi.

Hãy xét ví dụ sau:

```
# include <stdio.h>
# include <conio.h>
void fct()
{
    static int count = 1;
    printf("\n Day la lan goi ham fct lan thu %2d", count++);
```

```
}
void main()
{
    int i;
    for(i = 0;  i < 10; i++)
        fct();
    getch();
}
```

Kết quả khi thực hiện:



III.7. Tệp dữ liệu

III.7.1. Khái niệm và phân loại tệp

Khái niệm tệp

Tệp dữ liệu (*File*) là một tập hợp các dữ liệu có liên quan với nhau và có cùng kiểu dữ liệu.

Tệp được lưu trữ trên các thiết bị nhớ ngoài (đĩa mềm, đĩa cứng, CD-ROM...) với một tên nào đó để phân biệt với nhau.

Phân loại tệp

Dựa theo bản chất dữ liệu của tệp, người ta chia tệp thành hai loại là tệp văn bản và tệp nhị phân.

Tệp văn bản (*text file*) là tệp mà các phần tử của nó là các kí tự như chữ cái, chữ số, các dấu câu, các dấu cách và một số kí tự điều khiển (như CR – *Carriage Return* – có mã ASCII là 13, để về đầu dòng, LF – *Line Feed* – có mã ASCII là 10, để xuống dòng mới).

Tệp nhị phân (*binary file*) là tệp mà các phần tử của nó là các số nhị phân 0 và 1 mã hóa thông tin. Thông tin được mã hóa bởi các bit nhị phân có thể là số nguyên, số thực, các cấu trúc dữ liệu... Nếu thông tin được mã hóa là kí tự thì khi đó tệp nhị phân trở thành tệp văn bản. Vì vậy tệp văn bản là một trường hợp riêng của tệp nhị phân.

Vai trò của tệp

Dữ liệu phục vụ cho chương trình được lưu trữ trong các biến, tức là nằm ở bộ nhớ trong. Do đó khi chương trình kết thúc hoặc khi tắt máy thì các thông tin dữ liệu đó cũng không còn. Muốn lưu trữ dữ liệu lâu dài để sử dụng cho những lần sau ta phải lưu dữ liệu lên tệp, tức là để dữ liệu nằm ở bộ nhớ ngoài. Dữ liệu trên bộ nhớ ngoài không bị mất đi khi chương trình kết thúc hay khi ta tắt máy. Vì vậy tệp là phương tiện dùng để cất giữ dữ liệu lâu dài.

Phân biệt tệp và mảng

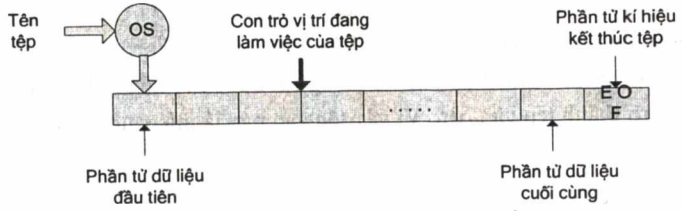
Về mặt tổ chức tệp và mảng hai rất giống nhau, đó là tập hợp các phần tử cùng kiểu, nhưng tệp khác với mảng ở hai điểm quan trọng sau đây:

- Tệp được lưu trữ trên bộ nhớ ngoài, do vậy dữ liệu trên tệp tồn tại lâu dài. Còn mảng thì được lưu trữ ở bộ nhớ trong, nên dữ liệu của mảng sẽ không còn khi chương trình kết thúc hoặc khi tắt máy.
- Kích thước của một tệp thường lớn hơn kích thước của một mảng rất nhiều vì tệp được lưu trữ trên bộ nhớ ngoài có dung lượng lớn hơn nhiều so với dung lượng của bộ nhớ trong là nơi lưu trữ mảng.

Tổ chức của tệp

Như đã nói, kiểu dữ liệu tệp khá giống với kiểu dữ liệu mảng ở chỗ là một tập hợp các phần tử cùng kiểu dữ liệu. Điểm khác nhau là số lượng phần tử trong mảng bị giới hạn trong khi số lượng phần tử của tệp không bị giới hạn. Vì số lượng phần tử trong tệp không bị giới hạn nên để biết được khi nào tới phần tử cuối cùng của tệp,

tất cả các tệp đều sử dụng một phần tử đặc biệt gọi là phần tử kí hiệu kết thúc tệp (*End Of File indicator* – **EOF**). Còn làm thế nào để biết được phần tử nằm ở vị trí đầu của tệp? Từ đường dẫn và tên tệp hợp lệ của tệp, hệ điều hành sẽ tự động giúp ta tìm ra được vị trí lưu trữ phần tử đầu tiên của tệp trên đĩa.



Hình III.7. Cấu trúc của tệp

Con trỏ tệp

Các phần tử của một tệp tạo thành một dãy và tại một thời điểm ta chỉ có thể truy cập được vào một phần tử của tệp mà thôi. Việc truy cập vào một phần tử được thực hiện thông qua một biến đệm, biến đệm này dùng để đánh dấu vị trí truy cập vào tệp tại thời điểm xác định. Biến đệm này được gọi là con trỏ tệp (*File positon locator* – tức là con trỏ chỉ vị trí đang làm việc của tệp, gọi tắt là con trỏ tệp).

Khi mở tệp, con trỏ tệp sẽ luôn trỏ vào vị trí đầu tiên của tệp. Sau mỗi thao tác đọc ghi trên tệp, con trỏ tệp sẽ tự động dịch chuyển về phía cuối tệp. Khoảng cách dịch chuyển (tính theo byte) sẽ bằng số byte đã được đọc từ tệp hoặc ghi lên tệp.

Quy trình thao tác với tệp

Các thao tác với tệp phải tuân thủ theo trình tự sau:

– Khai báo tệp

– Mở tệp để làm việc

– Truy nhập tệp

– Đóng tệp

III.7.2. Các thao tác với tệp

III.7.2.1. Khai báo

Trong C truy nhập tệp phải thông qua con trỏ tệp. Một con trỏ tệp (*file pointer*) được khai báo như sau:

FILE *tên_con_trỏ_tệp;

Ví dụ:

FILE *f1, *f2;

III.7.2.2. Mở tệp

Muốn làm việc với tệp trước hết ta phải mở tệp. Để mở một tệp ta dùng lệnh sau:

tên_con_trỏ_tệp = fopen(tên_tệp,chế_độ_mở_tệp);

Trong đó hàm fopen()là hàm có chức năng mở tệp với hai tham số là tên_tệp và chế_độ_mở_tệp. Nếu mở thành công thì nó sẽ trả về một con trỏ tệp tương ứng với tệp đã được mở, nếu không thì sẽ trả về con trỏ NULL.

Lưu ý: Để sử dụng hàm fopen() cũng như các hàm truy nhập và đóng tệp trình bày ở phần dưới ta phải khai báo tệp tiêu đề stdio.h.

Chế độ mở tệp

Phụ thuộc vào mục đích sử dụng tệp và bản chất dữ liệu trên tệp, người ta có các chế độ mở tệp sau:

"rb", **"wb"**, **"ab"**, **"r+b"**, **"w+b"**, **"a+b"** và **"rt"**, **"wt"**, **"at"**, **"r+t"**, **"w+t"**, **"a+t"**. Trong xâu kí hiệu cho chế độ mở tệp, kí tự cuối cùng dùng để kí hiệu cho bản chất dữ liệu của tệp, các kí tự còn lại ở đầu dùng để kí hiệu cho mục đích sử dụng tệp.

Kí hiệu	Mục đích sử dụng tệp
"r"	Mở tệp đã có để đọc, không được ghi. Nếu tệp không tồn tại, hàm fopen() sẽ trả lại trạng thái lỗi.
"w"	Mở tệp mới để ghi. Nếu tệp đã tồn tại nội dung của nó sẽ bị xóa hết.
"a"	Mở tệp để ghi thêm dữ liệu vào cuối tệp. Nếu tệp chưa tồn tại, nó sẽ được tạo mới.
"r+"	Mở tệp để vừa đọc vừa ghi. Nếu tệp chưa tồn tại thì sẽ báo lỗi.
"w+"	Mở tệp để vừa đọc vừa ghi. Nếu tệp đã tồn tại, nội dung của nó sẽ bị xóa hết.
"a+"	Mở tệp để ghi thêm dữ liệu vào cuối tệp. Tệp mới sẽ được tạo nếu nó chưa tồn tại.

Kí hiệu	Bản chất dữ liệu của tệp
"b"	Tệp nhị phân
"t"	Tệp văn bản

Ví dụ với khai báo:

```
FILE *f1, *f2, *f3;
```

Để mở tệp *c:\abc.txt* để đọc ta dùng lệnh

```
f1 = fopen("c:\\abc.txt", "r");
```

Để mở tệp *c:\ho_so.dat* để ghi ta dùng lệnh

```
f2 = fopen("c:\\ho_so.dat", "w");
```

Để mở tệp *c:\abc.txt* để vừa đọc và ghi ta dùng lệnh

```
f3 = fopen("c:\\abc.txt", "r+");
```

Lưu ý: Khi mở tệp, nếu không chỉ rõ bản chất dữ liệu của tệp thì C sẽ ngầm hiểu đó là tệp văn bản. Nếu muốn chỉ rõ bản chất dữ liệu của tệp là tệp nhị phân thì ta sử dụng thêm kí hiệu b đi kèm với kí hiệu xác định mục đích mở tệp.

Ví dụ: Để mở tệp *c:\ho_so.dat* theo chế độ nhị phân và để ghi, ta dùng lệnh:

```
f2 = fopen("c:\\ho_so.dat", "wb");
```

Đôi khi do những lý do khách quan hoặc chủ quan mà việc mở tệp không thành công và hàm fopen() sẽ trả về giá trị NULL để báo rằng việc mở tệp không thành công. Khi đó ta nên "bắt" lấy kết quả trả về này để có những xử lí thích hợp, nếu không chương trình sẽ báo lỗi và tự động thoát ra ngoài.

Để bắt lỗi phát sinh khi mở tệp ta có thể sử dụng mẫu sau:

```
// Trường hợp mở tệp có lỗi
if(con_trỏ_tệp = fopen(tên_tệp, chế_độ_mở_tệp)) == NULL)
{
    <Xử lí cho trường hợp mở tệp không thành công>
}else // Trường hợp mở tệp thành công
{
    <Xử lí khi mở tệp thành công>
}
```

III.7.2.3. Truy nhập tệp văn bản

Đọc dữ liệu từ tệp

Để đọc tệp ta dùng các hàm sau: fscanf(), fgets(), getc().

Hàm fscanf() có cú pháp khai báo là:

int fscanf(FILE* con trỏ tệp, xâu định dạng, [danh_sách_địa_chỉ]);

Lệnh fscanf() có chức năng đọc từ tệp văn bản tương ứng với con_trỏ_tệp dãy các dữ liệu, định dạng các dữ liệu đó theo khuôn dạng có trong xâu_định_dạng, sau đó lưu các giá trị đã được định dạng vào các vùng nhớ xác định trong danh-_sách_địa_chỉ.

Kết quả trả về: Nếu thực hiện thành công, hàm fscanf() trả về một giá trị nguyên là số byte đọc được từ tệp. Nếu thực hiện không thành công thì hàm trả về giá trị **EOF**.

Ví dụ:

fscanf(fptr, "%d %c",&a, &c);

Lệnh trên nhập giá trị cho hai biến a, c từ tệp.

Hàm fflush(): tương tự như hàm scanf(), trước khi dùng hàm fscanf() để nhập dữ liệu là kí tự, xâu kí tự từ tệp ta nên dùng lệnh fflush(). Hàm fflush() có cú pháp khai báo như sau:

int fflush(FILE* con trỏ tệp);

Hàm fflush() ghi toàn bộ dữ liệu chứa trong vùng đệm của tệp (nằm ở bộ nhớ trong) tương ứng với con_trỏ_tệp ra vùng nhớ của tệp trên bộ nhớ ngoài.

Giá trị trả về: Nếu hàm fflush() thực hiện thành công thì sẽ trả về giá trị 0, ngược lại giá trị trả về là **EOF**.

Hàm fgets() có cú pháp khai báo là:

char* fgets(char* xâu_kí_tự, int n, FILE* con_trỏ_tệp);

Hàm fgets() đọc từ tệp một xâu kí tự (cho phép chứa dấu cách) và gán xâu đọc được cho biến xâu_kí_tự. Việc đọc từ tệp sẽ dừng lại khi fgets() đọc đủ n-1 kí tự hoặc khi nó gặp kí tự xuống dòng, tùy gặp cái nào trước. Hàm fgets() sẽ tự động thêm kí tự xuống dòng ("\n") và kí tự kết thúc xâu ("\0", kí tự NUL) vào cuối xâu_kí_tự.

Giá trị trả về: Nếu đọc thành công, hàm fgets() sẽ trả về xâu kí tự trỏ bởi xâu_kí_tự, nếu có lỗi nó sẽ trả về con trỏ NULL.

Hàm getc() có cú pháp khai báo là:

int getc(FILE* con trỏ tệp);

Hàm getc() đọc từ tệp một kí tự (tức là một byte dữ liệu), sau đó chuyển đổi kí tự đó sang dạng số nguyên **int** (bằng cách thêm byte cao 0x00) rồi lấy giá trị số nguyên thu được làm giá trị trả về của hàm.

Giá trị trả về: Nếu thành công hàm getc() trả về kí tự đọc được sau khi đã chuyển sang dạng **int**. Nếu không thành công hàm trả về giá trị **EOF**.

Ghi dữ liệu lên tệp

Để ghi dữ liệu lên tệp ta dùng các hàm sau: fprintf(), fputs(), putc().

Hàm fprintf() có cú pháp khai báo là:

int fprintf(FILE* con_trỏ_tệp, xâu_định_dạng, [danh_sách_tham_số]);

Hàm fprintf() có chức năng hoàn toàn tương tự như hàm printf(), chỉ có một chỗ khác là hàm printf() ghi dữ liệu lên thiết bị ra chuẩn (stdin, thông thường là màn hình) còn fprintf() ghi dữ liệu lên một tệp được chỉ định trong tham số con_trỏ_tệp.

Kết quả trả về: Nếu thực hiện thành công, hàm fprintf() trả về một giá trị nguyên là số byte dữ liệu đã ghi lên tệp. Nếu thực hiện không thành công thì hàm fprintf() trả về giá trị **EOF**.

Hàm fputs() có cú pháp khai báo là:

int fputs(char* xâu_kí_tự, FILE* con_trỏ_tệp);

Hàm fputs() sẽ ghi nội dung của xâu_kí_tự lên tệp tương ứng với con_trỏ_tệp, tuy nhiên nó khác với hàm puts() ở chỗ không tự động ghi thêm kí tự xuống dòng lên tệp.

Giá trị trả về: Nếu thực hiện thành công, hàm fputs() trả về kí tự cuối cùng mà nó ghi được lên tệp, còn nếu không thành công nó trả về giá trị **EOF**.

Hàm putc() có cú pháp khai báo là:

int putc(int ch, FILE* con_trỏ_tệp);

Hàm putc() ghi nội dung của kí tự chứa trong biến **int** ch (kí tự được chứa trong byte thấp của biến ch) lên tệp tương ứng với con_trỏ_tệp.

Giá trị trả về: Nếu thành công, hàm putc() sẽ trả về số nguyên (kiểu **int**) là số thứ tự trong bảng mã ASCII của kí tự đã ghi lên tệp. Nếu không thành công nó trả về giá trị **EOF**.

Ví dụ:

```
int m;
m = putc(0x2345,stdout); // m = 0x0045 = 69, la so thu tu cua ki tu E
```

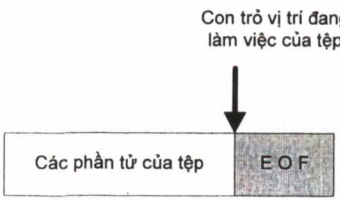
Một số thao tác khác

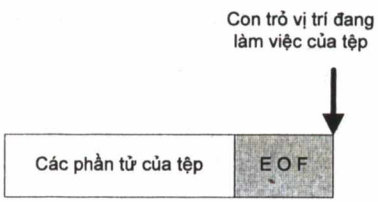
Hàm feof()

int feof(FILE* con_trỏ_tệp);

Hàm feof() dùng để kiểm tra xem đã duyệt đến vị trí cuối tệp hay chưa. Nó thực hiện việc này bằng cách kiểm tra xem trong khối dữ liệu được đọc vào ở lần thực hiện gần nhất có phần tử **EOF** hay không (tức là phần tử **EOF** có được đọc trong lần đọc gần đây nhất hay không), nếu có thì hàm feof() trả về một giá trị khác 0, còn nếu chưa thì trả về giá trị 0.

Cần lưu ý điều này vì ta thường lầm tưởng rằng hàm feof() kiểm tra việc đã duyệt đến phần tử cuối cùng của tệp hay chưa bằng cách kiểm tra xem con trỏ tệp đã trỏ đến phần tử cuối cùng hay chưa. Tuy nhiên, khi con trỏ vừa đọc xong phần tử cuối cùng, thì phần tử **EOF** vẫn chưa được đọc.





Hàm fseek()

int fseek(FILE* con_trỏ_tệp, long int n, int vị_trí_ban_đầu);

Hàm fseek() dùng để dịch chuyển con trỏ tệp từ vị_trí_ban_đầu đi một khoảng cách có độ dài n byte.

Giá trị trả về: Hàm fseek() sẽ trả về giá trị 0 nếu như việc dịch chuyển thành công và trả về giá trị khác 0 nếu việc dịch chuyển không thành công.

Lưu ý:

– Giá trị của biến n có thể lớn hơn, nhỏ hơn hoặc bằng 0. Nếu lớn hơn 0 thì hướng của sự dịch chuyển là về phía cuối tệp, nhỏ hơn 0 nghĩa là dịch chuyển về phía đầu tệp, bằng 0 nghĩa là thực tế không dịch chuyển gì cả.

– Cần thận trọng khi dùng hàm fseek() với tệp mở theo kiểu văn bản vì khi mở tệp theo kiểu văn bản có thể có sự tự động chuyển đổi kí tự (ví dụ chuyển đổi cặp kí tự '\r' và '\n' thành kí tự '\n'), điều đó khiến cho việc tính toán số bước dịch chuyển của con trỏ tệp rất dễ bị sai.

– Tham số vị_trí_ban_đầu có thể nhận 1 trong 3 giá trị hằng số sau:

Tên hằng	Giá trị	Ý nghĩa
SEEK_SET	0	Vị trí ban đầu là đầu tệp
SEEK_CUR	1	Vị trí ban đầu là vị trí hiện thời của con trỏ vị trí làm việc của tệp
SEEK_END	2	Vị trí ban đầu là cuối tệp

Ví dụ sử dụng:

```
fseek(file_ptr, 50, SEEK_SET);// con trỏ tệp cách vị trí đầu tệp 50 byte
fseek(file_ptr, - 40, 2);          // con trỏ tệp cách vị trí cuối tệp 40 byte
```

Hàm rewind():

void rewind(FILE* con_trỏ_tệp);

Hàm rewind()sẽ đưa con trỏ tệp về đầu tệp. Với file_ptr là một biến con trỏ tệp, hàm rewind(file_ptr) tương đương với fseek(file_ptr,0,SEEK_SET);

Hàm rewind() không có giá trị trả về.

Lưu ý: Để sử dụng các hàm fscanf(), fgets(), getc(), fflush(), fprintf(), fputs(), putc(), feof(), fseek() và rewind() ta cần khai báo tệp tiêu đề stdio.h.

III.7.2.4. Truy nhập tệp nhị phân

Đọc dữ liệu trên tệp

Để đọc dữ liệu từ tệp nhị phân ta dùng hàm fread() có cú pháp khai báo như sau:

int fread (void *địa_chỉ_biến, int kích_thước_phần_tử, int số_phần_tử, FILE* con_trỏ_tệp);

Hàm fread() đọc từ tệp một khối dữ liệu kích thước số_phần_tử × kích_thước_phần_tử byte, sau đó ghi khối dữ liệu đó lên vùng nhớ có địa chỉ là địa_chỉ_biến.

Kết quả trả về: Nếu việc đọc dữ liệu từ tệp thực hiện thành công, hàm fread() trả về một giá trị nguyên là số mục (không phải số byte) đọc được từ tệp. Nếu thực hiện không thành công thì hàm fread() trả về giá trị 0.

Ghi dữ liệu trên tệp

Để ghi dữ liệu lên tệp nhị phân ta dùng hàm fwrite() có cú pháp khai báo là:

int fwrite (void *địa_chỉ_biến, int kích_thước_phần_tử, int số_phần_tử, FILE* con trỏ tệp);

Hàm fwrite() sẽ đọc từ bộ nhớ một khối dữ liệu có địa chỉ bắt đầu là địa_chỉ_biến và có kích thước là số_phần_tử × kích_thước_phần_tử byte, sau đó nó ghi khối dữ liệu này lên tệp.

Kết quả trả về: Nếu việc ghi dữ liệu lên tệp thực hiện thành công, hàm fwrite() sẽ trả về một giá trị nguyên là số mục (không phải số byte) đã ghi lên tệp. Nếu thực hiện không thành công thì hàm fwrite()trả về giá trị 0.

Dịch chuyển con trỏ tệp

Tương tự như tệp văn bản, ta có thể dùng các hàm fseek()và rewind()để dịch chuyển con trỏ tệp trên tệp nhị phân. Hàm fseek()khi dùng với tệp nhị phân thì không phải lưu ý như khi dùng với tệp văn bản.

Nhận xét: Các hàm trong các cặp hàm fread() – fwrite(), fscanf() – fprintf(), fputs() – fgets(), và getc() – putc() có chức năng đối ngẫu nhau.

III.7.2.5. Đóng tệp

Đóng tệp là đảm bảo những thay đổi dữ liệu được lưu lại trên tệp.

Để đóng tệp, ta dùng hàm fclose() có cú pháp khai báo:

int fclose(FILE* <tên con trỏ tệp>);

Hàm fclose() trả lại giá trị 0 nếu đóng thành công, trả về giá trị **EOF** nếu không đóng tệp thành công.

Ví dụ tổng hợp:

Bài toán: Tạo tệp nhị phân float.dat đặt tại ổ đĩa C để ghi 100 số thực. Sau đó mở lại tệp này và ghi nội dung sang một tệp khác, có tên được nhập từ bàn phím. Tiếp theo đó, hiển thị giá trị số thực đầu tiên trong tệp float.dat; cho phép người dùng nhập một số thứ tự từ bàn phím và hiển thị số thực ở vị trí này trong tệp float.dat.

```
#include <stdio.h>
#include <conio.h>
#include <process.h>
void main(){
    FILE* fptr1, *fptr2;// Khai bao bien con tro tep
    int i;// Bien dem
    float f, a[100]; // Mot bien so thuc va mot mang so thuc
    char file_name_2[20];// xau ki tu chua ten tep thu 2
    clrscr();
    // Mở tệp c:\float.dat de ghi len do 100 so thuc
    if((fptr1 = fopen("c:\\float.dat","wb"))==NULL)
    {
        printf("\n Khong mo duoc file c:\\float.dat");
        printf("\n An phim bat ki de ket thuc chuong trinh");
        exit(1);
    }
    // Tao 100 so thuc theo mot cong thuc nao do va ghi vao mang a[100]
```

```
for(i=0;i<100;i++)
    a[i] = (i*i+1.0)/(i+1);
// Ghi cac so thuc len tep c:\float.dat
for(i=0;i<100;i++)
    fwrite(&a[i],sizeof(float),1,fptr1);
// Đóng tệp c:\float.dat lai de luu du lieu vua ghi
fclose(fptr1);
// Mở tệp c:\float.dat vua tao de doc du lieu
if((fptr1 = fopen("c:\\float.dat","rb"))==NULL)
{
    printf("\n Khong mo duoc file c:\\float.dat");
    printf("\n An phim bat ki de ket thuc chuong trinh");
    exit(1);
}
// Mo tep thu 2 co ten nhap tu ban phim
// ta se copy du lieu tu tep c:\float.dat sang tep nay
printf("\n Nhap vao ten tep thu 2: ");
fflush(stdin);
gets(file_name_2);
if((fptr2 = fopen(file_name_2,"wb"))==NULL)
{
    printf("\n Khong mo duoc file %s",file_name_2);
    printf("\n An phim bat ki de ket thuc chuong trinh");
    exit(1);
}
// Sao chep du lieu tu tep c:\float.dat sang tep thu 2
fread(&f,4,1,fptr1);
```

```
    while(!feof(fptr1))
    {
        fwrite(&f,4,1,fptr2);
        fread(&f,4,1,fptr1);
    }
    // Dich chuyen con tro vi tri lam viec hien tai cua tep
    // c:\float.dat ve dau tep
    // Doc va hien thi so thuc dau tien trong tep c:\float.dat
   rewind(fptr1);
    fread(&f,sizeof(float),1,fptr1);
    printf("\n So thuc dau tien tren c:\\float.dat la %f",f);
    // Doc va hien thi so thuc trong tep c:\float.dat
   // co so thu tu nhap tu ban phim
    printf("\n Cho biet thu tu cua so thuc trong c:\\float.dat: ");
    scanf("%d",&i);
    // Dich chuyen con tro cua tep c:\float.dat
    // den vi tri tuong ung voi so thuc muon hien thi
    fseek(fptr1,(i-1)*sizeof(float),SEEK_SET);
    fread(&f,sizeof(float),1,fptr1);
    printf(" So thuc thu %d trong c:\\float.dat la %f",i,f);
    fclose(fptr1);
    fclose(fptr2);
    // Cho an phim bat ki de ket thuc chuong trinh
    getch();
}
```

Kết quả thực hiện:

Nhap vao ten tep thu 2: c:\f_copy.dat

So thuc dau tien tren c:\float.dat la 1.000000

Cho biet thu tu cua so thuc trong tep c:\float.dat: 10

So thuc thu 10 trong c:\float.dat la 8.200000

Và trên thư mục gốc của ổ đĩa C ta sẽ thấy có hai tệp là *float.dat* và *f_copy.dat* cùng có kích thước 400 byte.

III.8. Câu hỏi và bài tập

1. Trong các định danh sau, đâu là định danh hợp lệ, đâu là định danh không hợp lệ?

1_a, 3d, so luong, int, char, string, _constant_2, a, sinh_vien

2. Hãy cho biết các hàm toán học sau thực hiện chức năng gì?

sqrt(x), pow(x,y), exp(x), log(x), cos(x), ceil(x), floor(x)

3. Để viết chú thích trong ngôn ngữ C, có mấy cách? Đó là gì?

4. Viết các khai báo cho các yêu cầu sau:

a. x là biến nguyên, y là biến thực

b. biến k sẽ nhận các giá trị trong đoạn [0, 255]

c. m, n là các biến nguyên trong đoạn [-32768, +32767]

d. i, j là các biến có thể nhận giá trị trong đoạn [0, 65535]

e. ch là biến ký tự

5. Chọn cách khai báo đúng tệp tiêu đề trong ngôn ngữ C:

a. #include<tên_tệp_ tiêu_ đề>

b. #include tên_tệp_ tiêu_ đề

a. include < tên_tệp_ tiêu_ đề >

b. #include< tên_tệp_ tiêu_ đề >;

6. Tìm các ký hiệu ghi chú thích trong ngôn ngữ C:

a. (*dòng chú thích*)

b. {dòng chú thích}

c. /* dòng chú thích */

d. // dòng chú thích

e. <-- dòng chú thích-->

7. Để sử dụng hàm clrscr(), phải thêm tệp tiêu đề nào vào chương trình?

a. string.h

b. conio.h

c. stdio.h

d. math.h

8. Xét khai báo:

int a, b;

Chỉ ra biểu thức không hợp lệ:

a. a-=b

b. a==b

c. a=b

d. a-b=0

9. Chọn khai báo hợp lệ:

a. float: a-9;

b. a=9.0: float;

c. a:float=9;

d. float a=9.0;

10. Nhập vào một số nguyên từ bàn phím và hiển thị số đó ra màn hình.

11. Nhập vào một số thực từ bàn phím và hiển thị số đó ra màn hình theo quy cá có ba vị trí sau dấu chấm.

12. Nhập vào hai số nguyên từ bàn phím, sau đó tính và hiển thị ra màn hình giá trị tổng, hiệu, tích, thương của hai số vừa nhập.

13. Nhập vào một xâu ký tự và hiển thị xâu vừa nhập ra màn hình.

14. Viết chương trình tính và hiển thị ra màn hình chu vi và diện tích hình tròn theo giá trị bán kính r nhập vào từ bàn phím.

15. Nhập vào ba số nguyên a, b, c từ bàn phím. Hiển thị giá trị của các biểu thức sau ra màn hình:

a. a++ + ++a

b. --a - b-- * ++c

16. Dùng biểu thức điều kiện ?: để tìm và hiển thị ra số lớn nhất trong ba số thực nhập vào từ bàn phím

17. Nhập vào hai số thực từ bàn phím. Viết chương trình dùng lệnh if else tìm và hiển thị giá trị nhỏ nhất ra màn hình.

18. Viết chương trình nhập vào số nguyên từ bàn phím. Nếu đó là số chẵn thì hiển thị "Ban vua nhap so chan", nếu là số lẻ thì hiển thị "Ban vua nhap so le".

19. Nhập vào ba số thực từ bàn phím, kiểm tra xem nó có tạo thành ba cạnh của một tam giác không. Nếu có thì hiện thông báo ra màn hình "La 3 canh cua tam giac", ngược lại hiển thị "Không phải la 3 canh cua tam giac".

20. Trong một năm các tháng có 30 ngày là 4, 6, 9, 11 còn các tháng có 31 ngày là 1, 3, 5, 7, 8, 10, 12. Riêng tháng hai có thể có 28 hoặc 29 ngày. Hãy viết chương trình nhập vào một số thể hiện tháng, sau đó đưa ra kết luận tháng đó có bao nhiêu ngày (dùng cấu trúc switch).

21. Đưa ra màn hình các số nguyên lẻ nhỏ trong khoảng từ 1 đến 100 (dùng vòng lặp for).

22. Nhập vào một số N nguyên, N>1 từ bàn phím. Tính tổng các số nguyên từ 1 đến N (dùng vòng lặp for, while, do while).

23. Viết chương trình nhập vào số nguyên dương N từ bàn phím, sau đó tính giai thừa và hiển thị ra màn hình (dùng for, while và do while).

24. Hãy tìm giá trị của biến sum sau đoạn chương trình sau:

```
for(int i =0;i+1<=10;i+=2)
if(i==0) sum = i;
else sum+=i;
```

a. 12

b. 20

c. 30

d. Các phương án trên đều sai

25. Lập chương trình giải phương trình bậc hai: $ax^2 + bx + c = 0$ với a, b, c nhập vào từ bàn phím.

26. Viết chương trình kiểm tra một số nguyên nhập từ bàn phím có phải là số nguyên tố hay không?

27. Lập trình tính và hiển thị tổng S = 1 + 1/2 + 1/3 + ... + 1/n với n nhập vào từ bàn phím.

28. Viết chương trình nhập vào hai số nguyên dương từ bàn phím và đưa ra ước số chung lớn nhất của chúng.

29. Viết chương trình nhập vào hai số nguyên từ bàn phím và đưa ra bội số chung nhỏ nhất của chúng.

30. Nhập vào một mảng số thực gồm 10 phần tử. Tìm và hiển thị giá trị và chỉ số của phần tử lớn nhất.

31. Nhập vào một mảng n số nguyên (tối đa 20 phần tử) với n nhập vào từ bàn phím. Sắp xếp lại mảng theo sự tăng dần của các giá trị phần tử.

32. Chỉ ra kết quả in trên màn hình:

```
int i = 0;
while(i++<=2) printf("Hello! ");
```

a. Hello!

b. Hello! Hello! Hello! Hello!

c. Hello! Hello!

d. Hello! Hello! Hello!

33. Nhập một xâu kí tự từ bàn phím gồm các từ, ví dụ "Thu do Ha Noi". Lập chương trình để bỏ bớt các dấu trống giữa các từ sao cho các từ chỉ cách nhau ít nhất một dấu trống.

34. Viết chương trình nhập vào từ bàn phím họ và tên của một người, sau đó in phần tên ra màn hình. Ví dụ: "Tran Hung Dao" thì in ra "Dao".

35. Nhập vào một câu, kết thúc bằng dấu chấm. In ra câu đó có bao nhiêu từ.

36. Cho khai báo:

```
struct quoc_gia
{  char ten[31];
   char thu_do[31];
   int dan_so;
}dsqg[10];
```

Mảng dsqg chiếm số byte trong bộ nhớ là:

a. 440

b. 460

c. 640

d. 660

37. Viết một chương trình thực hiện những công việc sau:

a. Yêu cầu người dùng nhập vào một số nguyên dương n với $5 \leq n \leq 20$ (có kiểm tra tính hợp lệ của giá trị được nhập vào, nếu giá trị n nhập vào không thỏa mãn điều kiện thì yêu cầu nhập lại).

b. Yêu cầu người dùng nhập vào thông tin của n sinh viên gồm những mục sau:

- Họ và tên: Có kiểu dữ liệu là xâu kí tự gồm không quá 30 kí tự
- Lớp: Xâu kí tự có độ dài không quá 5 kí tự
- Điểm thi Tin đại cương: Là một số nguyên có giá trị từ 0 đến 10

i. Đưa ra màn hình danh sách các sinh viên cùng thông tin của họ mà người dùng vừa nhập vào.

ii. Yêu cầu người dùng nhập vào từ bàn phím một số thực. Đưa ra màn hình danh sách các sinh viên có điểm thi Tin đại cương nhỏ hơn giá trị số thực vừa nhập vào.

iii. Đưa ra màn hình danh sách sinh viên được sắp xếp theo chiều giảm dần của điểm thi Tin đại cương.

iv. Đưa ra màn hình danh sách sinh viên với họ và tên được sắp xếp theo chiều của bảng chữ cái.

38. Thông tin về nguồn vốn viện trợ ODA mà các quốc gia viện trợ cho Việt Nam **từ năm 2000 đến 2007** được lưu trong **cấu trúc kiểu ODA** gồm có:

tên quốc gia *(xâu không quá 30 kí tự thường)*; **số tiền** *(mà quốc gia đó tài trợ, đơn vị: triệu USD)*, **năm** *(năm tài trợ)*.

Giả sử thông tin trên đã được lưu trữ trong tệp **TAITRO.DAT** tại thư mục gốc ổ **D**. Lập chương trình thực hiện những công việc sau:

a. Nhập thêm vào cuối file trên thông tin về sự tài trợ của các quốc gia trong năm 2008. Điều kiện kết thúc là khi nhập tên quốc gia là xâu "$$$".

b. Nhập vào một số N ($2000 \le N \le 2008$). Đọc file trên, đưa ra tên, số tiền tài trợ của các quốc gia cho Việt Nam trong năm N theo quy cách: 30 vị trí cho phần tên quốc gia, 6 vị trí cho phần số tiền tài trợ, mỗi quốc gia trên một dòng.

c. Nhập vào một xâu kí tự S thể hiện tên một quốc gia. Đọc lại file trên và đưa ra tổng số tiền từ năm 2003 đến nay mà quốc gia đó đã tài trợ cho Việt Nam. Nếu không tìm thấy thì thông báo "KHÔNG TÌM THẤY".

39. Viết hàm tìm ước số chung lớn nhất uscln(int x, int y) và bội số chung nhỏ nhất bscnn(int x, int y) của hai số.

40. Lập hàm giải phương trình bậc hai ptb2(float a, float b, float c, float *x1, float *x2), trong đó a,b,c là các hệ số của phương trình. Hàm nhận giá trị 0 nếu phương trình vô nghiệm, giá trị 1 nếu phương trình có nghiệm kép (chứa trong *x1), giá trị 2 nếu phương trình có hai nghiệm (chứa trong *x1 và *x2), giá trị 3 nếu phương trình có vô số nghiệm.

41. Lập hàm tính giai thừa và dùng hàm này tính tổng: 1! + 2! +...+N! (N nhập từ bàn phím và <=10).

42. Đa thức cấp n được tính theo công thức.

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x^1 + a_0 x^0$$

Lập hàm đa thức f(float a[], int n, float x) để tính giá trị của đa thức.

43. Viết chương trình nhập từ bàn phím N số thực lưu vào một mảng (N < 100 và N được nhập từ bàn phím). Sau đó ghi ra một file văn bản có tên là "float.dat" theo quy cách: dòng đầu tiên lưu số lượng các số thực, các dòng tiếp theo lưu các số thực, mỗi số lưu trên một dòng. Đọc lại tệp văn bản đó và lưu các số thực đọc được vào một mảng. Sắp xếp các số thực trong mảng theo thứ tự tăng dần và ghi ra một tệp văn bản khác có tên là "float_sx.dat" theo quy cách giống như tệp "float.dat".

44. Viết chương trình sao chép nội dung tệp mã nguồn chương trình C có tên là file_1.C sang tệp có tên là file_2.C.

45. Viết chương trình copy file:

a. Nhập vào từ bàn phím hai xâu kí tự là đường dẫn của file nguồn và file đích.

b. Copy nội dung của file nguồn sang file đích.

46. Viết chương trình ghép nối nội dung hai file:

a. Nhập vào từ bàn phím hai xâu kí tự là đường dẫn của file nguồn và file đích.

b. Ghép nội dung của file nguồn vào cuối file đích.

47. Viết chương trình so sánh nội dung hai file:

a. Nhập từ bàn phím hai xâu kí tự là đường dẫn tới hai file cần so sánh.

b. Hiển thị ra màn hình dòng thông báo:

Hai file ten_1 và ten_2 giống nhau

nếu hai file có nội dung giống nhau (hai file có cùng kích thước và các bít cùng vị trí thì có giá trị giống nhau).

Hai file ten_1 và ten_2 không giống nhau

nếu hai file có nội dung khác nhau. Ở đây ten_1 và ten_2 là đường dẫn của hai file cần so sánh.

48. Mở một tệp văn bản, đếm xem trong văn bản đó có bao nhiêu kí tự, bao nhiêu từ, bao nhiêu câu.

49. Một tệp văn bản tên là "thisinh.dat" lưu dữ liệu về các thí sinh và có tổ chức như sau:

– Dòng đầu tiên lưu số lượng thí sinh.

– Các dòng tiếp theo mỗi dòng lưu thông tin về một thí sinh gồm có: số báo danh (10 kí tự), họ và tên (30 kí tự), điểm thi (bốn kí tự với một kí tự dành cho phần thập phân, một kí tự cho dấu "." dùng để ngăn cách và hai kí tự cho phần nguyên).

Hãy viết chương trình:

a. Đọc dữ liệu từ tệp "thisinh.dat" và hiển thị ra màn hình danh sách các thí sinh theo quy cách:

Số thứ tự Số báo danh Họ tên Điểm thi

Trong đó số thứ tự chiếm ba vị trí, số báo danh chiếm 10 vị trí, họ và tên chiếm 30 vị trí, điểm thi chiếm năm vị trí với hai vị trí dành cho phần thập phân.

b. Sắp xếp các thí sinh theo kết quả điểm thi tăng dần và lưu vào tệp "thisinh2.dat" với quy cách giống như quy cách của tệp "thisinh.dat".

TÀI LIỆU THAM KHẢO

[1] Brian W. Kernighan and Dennis M. Ritchie, ***The C programming language***, 2nd, K&R Publishion 1995.

[2] Steve Oualline, ***Practical C programming***, 3rd Edition, O'Reilly publishion.

[3] Quách Tuấn Ngọc, ***Giáo trình Tin học căn bản***, Nhà xuất bản Thống kê, 2001.

[4] Hoàng Kiếm, ***Giáo trình Tin học đại cương***, Nhà xuất bản Giáo dục, 1997.

[5] Hoàng Kiếm, ***Giáo trình Tin học đại cương nâng cao***, Nhà xuất bản Giáo dục, 1998.

[6] Tô Văn Nam, ***Giáo trình Tin học đại cương***, Nhà xuất bản Giáo dục Việt Nam, 2009.

[7] Nguyễn Thúc Hải, ***Mạng máy tính và các hệ thống mở***, Nhà xuất bản Giáo dục, 1999.

[8] Phạm Thị Thanh Hồng, ***Giáo trình Hệ thống thông tin quản lý***, Nhà xuất bản Bách Khoa – Hà Nội, 2010.

[9] Quách Tuấn Ngọc, ***Ngôn ngữ lập trình C***, Nhà xuất bản Thống kê, 2003.

[10] Phạm Văn Ất, ***Kỹ thuật lập trình C cơ sở và nâng cao***, Nhà xuất bản Khoa học kỹ thuật, 1999.

[11] Nguyễn Thanh Thủy và các cộng sự, ***Nhập môn Lập trình ngôn ngữ C***, Nhà xuất bản Khoa học kỹ thuật, 2003.

[12] Tô Văn Nam, ***Bài tập Tin học đại cương***, Nhà xuất bản Giáo dục Việt Nam, 2009.

[13] Nguyễn Thanh Thủy, Nguyễn Quang Huy, ***Bài tập Lập trình ngôn ngữ C***, Nhà xuất bản Khoa học kỹ thuật, 2001.

[14] Bùi Thế Tâm, ***Ngôn ngữ C và lập trình hướng đối tượng***, Nhà xuất bản Giao thông vận tải, 2006.

GIÁO TRÌNH TIN HỌC ĐẠI CƯƠNG

NHÀ XUẤT BẢN BÁCH KHOA – HÀ NỘI
Ngõ 17 – Tạ Quang Bửu – Hai Bà Trưng – Hà Nội
Điện thoại: 04. 38684569; Fax:04. 38684570
www.nxbbk.hust.edu.vn

Chịu trách nhiệm xuất bản:
TS. PHÙNG LAN HƯƠNG

Phản biện: PGS. ĐẶNG VĂN CHUYẾT
TS. PHẠM ĐĂNG HẢI
Biên tập: ĐỖ THANH THÙY
Sửa bản in: TRẦN THỊ PHƯƠNG
VŨ THỊ HẰNG
Trình bày: TRẦN THỊ PHƯƠNG

In 1200 cuốn khổ (16 × 24) cm tại Công ty TNHH in và thương mại Sông Lam, số 1 Phùng Chí Kiên, P. Nghĩa Đô, Cầu Giấy, Hà Nội.
Số đăng ký KHXB: 22–2014/CXB/244–80/BKHN; ISBN: 978-604-911-829-6.
Số QĐXB: 102/QĐ – ĐHBK – BKHN ngày 08/7/2014.
In xong và nộp lưu chiểu quý III năm 2014.