

2010

TRƯỜNG ĐẠI HỌC  
BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ  
THÔNG TIN VÀ  
TRUYỀN THÔNG

# [BÀI GIẢNG TIN HỌC ĐẠI CƯƠNG - IT1110]

[Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.]

## **THÔNG TIN MÔN HỌC**

### **Trình độ:**

Sinh viên đại học các ngành

**Khối lượng:** 4(3-1-1-8)

- Lý thuyết: 45 tiết
- Bài tập/BTL: 15 tiết (bài tập, thảo luận)
- Thí nghiệm: 5 bài (x 3 tiết)(thực hành)

### **Mô tả vắn tắt nội dung:**

Khái niệm thông tin, biểu diễn thông tin trong máy tính. Hệ thống máy tính: phần cứng, phần mềm, hệ điều hành, mạng máy tính. Giải quyết bài toán bằng máy tính. Các cấu trúc lập trình cơ bản..

## MỤC LỤC

### PHẦN 1. TIN HỌC CĂN BẢN ..... 8

#### I.1. Thông tin và biểu diễn thông tin (5T lý thuyết)..... 8

I.1.1. Các khái niệm cơ bản về thông tin và tin học (1T lý thuyết) .....	8
I.1.1.1. Thông tin và xử lý thông tin.....	8
a. Thông tin - Dữ liệu – Tri thức.....	8
b. Quy trình xử lý thông tin .....	9
I.1.1.2. Máy tính điện tử và phân loại .....	9
a. Lịch sử phát triển của máy tính điện tử .....	9
b. Phân loại máy tính điện tử .....	10
I.1.1.3. Tin học và các ngành công nghệ liên quan .....	10
a. Tin học .....	10
b. Công nghệ thông tin (Information Technology - IT).....	11
c. Công nghệ thông tin và truyền thông.....	11
I.1.2. Biểu diễn dữ liệu trong máy tính (4T lý thuyết) .....	11
I.1.2.1. Biểu diễn số trong các hệ đếm .....	11
a. Hệ đếm cơ số b.....	11
b. Hệ đếm thập phân (Decimal system, b=10).....	12
c. Hệ đếm nhị phân (Binary system, b=2) .....	13
d. Hệ đếm bát phân (Octal system, b=8) .....	13
e. Hệ đếm thập lục phân (Hexa-decimal system, b=16) .....	13
f. Chuyển đổi một số từ hệ thập phân sang hệ đếm cơ số b .....	14
I.1.2.2. Biểu diễn dữ liệu trong máy tính và đơn vị thông tin .....	14
a. Nguyên tắc chung .....	14
b. Đơn vị thông tin.....	15
I.1.2.3. Biểu diễn số nguyên.....	16
a. Số nguyên không dấu.....	16
b. Số nguyên có dấu.....	16
c. Tính toán số học với số nguyên.....	17
Cộng/ trừ số nguyên.....	17
Nhân/ chia số nguyên.....	17
I.1.2.4. Biểu diễn số thực.....	18
a. Nguyên tắc chung .....	18
b. Chuẩn IEEE754/85 .....	19
I.1.2.5. Biểu diễn ký tự.....	20
a. Nguyên tắc chung .....	20
b. Bộ mã ASCII .....	20
c. Bộ mã Unicode .....	22

#### I.2. HỆ THỐNG MÁY TÍNH (7 tiết)..... 23

I.2.1. Hệ thống máy tính (3T lý thuyết).....	23
--	----

a. Mô hình cơ bản của máy tính.....	23
b. Bộ xử lý trung tâm – CPU .....	25
c. Bộ nhớ.....	26
d. Hệ thống vào-ra .....	27
e. Liên kết hệ thống (buses).....	29
I.2.1.2. Phần mềm máy tính .....	30
a. Dữ liệu và giải thuật.....	30
b. Chương trình và ngôn ngữ lập trình .....	35
c. Phân loại phần mềm máy tính.....	37
I.2.2. Mạng máy tính (2T lý thuyết).....	37
I.2.2.1. Lịch sử phát triển của mạng máy tính.....	37
I.2.2.2. Phân loại mạng máy tính.....	38
I.2.2.3. Các thành phần cơ bản của một mạng máy tính .....	38
I.2.2.4. Mạng Internet.....	40
I.2.3. Giới thiệu hệ điều hành (2T lý thuyết).....	41
I.2.3.1. Các khái niệm cơ bản.....	41
a. Khái niệm hệ điều hành .....	41
b. Tập (File) .....	41
c. Quản lý tập của hệ điều hành.....	42
I.2.3.2. Hệ lệnh của hệ điều hành .....	43
I.2.3.3. Hệ điều hành Windows .....	44
a. Sự ra đời và phát triển.....	44
b. Khởi động và thoát khỏi Windows XP .....	44
c. Một số thuật ngữ và thao tác thường sử dụng.....	45
d. Cấu hình Windows (Control Panel).....	47
e. Windows Explorer .....	52
<b>I.3. Các hệ thống ứng dụng (4T lý thuyết).....</b>	<b>56</b>
I.3.1. Các hệ thống quản lý thông tin.....	56
I.3.1.1. Các khái niệm về thông tin và Hệ thống quản lý thông tin.....	56
I.3.1.2. Phân loại.....	56
a. Phân loại theo cấp bậc quản lý.....	56
b. Phân loại theo chức năng nghiệp vụ .....	59
c. Phân loại theo quy mô tích hợp .....	60
I.3.2. Hệ thông tin bảng tính.....	62
I.3.3. Hệ quản trị cơ sở dữ liệu .....	67
I.3.3.1. Cơ sở dữ liệu .....	67
a. Khái niệm về Cơ Sở Dữ Liệu.....	67
b. Hệ quản trị cơ sở dữ liệu.....	72
I.3.4. Các hệ thống thông minh .....	76
<b>PHẦN II. GIẢI QUYẾT BÀI TOÁN.....</b>	<b>77</b>

<b>II.1. Thuật toán(4 tiết LT)</b>	<b>77</b>
II.1.1. Định nghĩa thuật toán	77
II.1.2. Biểu diễn thuật toán	78
II.1.2.1. Ngôn ngữ lưu đồ	79
II.1.2.2. Mã giả	83
II.1.3. Thuật toán đệ qui	84
II.1.4. Một số thuật toán thông dụng	85
II.1.4.1. Thuật toán số học	85
II.1.4.2. Thuật toán về dãy	86
II.1.5. Thuật giải heuristic	87
II.1.5.1. Thuật giải – Sự mở rộng khái niệm của thuật toán	87
II.1.5.2. Thuật giải heuristic	87
<b>II.2. Giải quyết bài toán (4 tiết LT, 2 tiết BT)</b>	<b>88</b>
II.2.1. Khái niệm về bài toán	88
II.2.2. Các bước giải quyết bài toán bằng máy tính	89
<b>Bài tập về Thuật toán (2 tiết BT)</b>	<b>90</b>
II.2.3. Các phương pháp giải quyết bài toán bằng máy tính	90
II.2.3.1. Giải quyết bài toán theo hướng xác định trực tiếp lời giải	90
II.2.3.2. Giải quyết bài toán theo hướng tìm kiếm lời giải	91
II.2.4. Phân loại bài toán	91
II.2.4.1. Độ phức tạp thuật toán	91
II.2.4.2. Phân loại bài toán	93
<b>PHẦN 3. LẬP TRÌNH</b>	<b>95</b>
<b>III.1. Tổng quan về ngôn ngữ C (3 tiết LT)</b>	<b>95</b>
III.1.1. Lịch sử phát triển	95
III.1.2. Các phần tử cơ bản của ngôn ngữ C	96
1.2.1. Tập kí tự	96
1.2.2. Từ khóa	96
1.2.3. Định danh	97
1.2.4. Các kiểu dữ liệu	98
1.2.5. Hằng	99
1.2.6. Biến	100
1.2.7. Hàm	100
1.2.8. Biểu thức	101
1.2.9. Câu lệnh	101
1.2.10. Chú thích	102
III.1.3. Cấu trúc cơ bản của một chương trình C	102

III.1.4. Biên dịch chương trình C .....	105
III.1.5. Trình biên dịch Turbo C++.....	105
III.1.6. Cài đặt và sử dụng Turbo C++ 3.0 .....	105
<b>III.2. Kiểu dữ liệu và biểu thức trong C (4 tiết LT).....</b>	<b>106</b>
III.2.1. Các kiểu dữ liệu chuẩn trong C (1 tiết LT).....	107
III.2.2. Các biểu thức .....	109
III.2.3. Các phép toán .....	110
III.2.3.1. Phép toán số học.....	110
III.2.3.2. Phép toán quan hệ .....	111
III.2.3.3. Các phép toán logic .....	112
III.2.3.4. Phép toán gán .....	112
III.2.4. Thứ tự ưu tiên các phép toán .....	114
III.2.5. Một số toán tử đặc trưng trong C .....	115
<b>III.3. Cấu trúc lập trình trong C (6 tiết LT).....</b>	<b>118</b>
III.3.1. Vào/ra .....	118
III.3.1.1. Các lệnh vào ra dữ liệu với các biến (printf, scanf) .....	118
III.3.1.2. Các lệnh nhập xuất khác.....	123
III.3.2. Cấu trúc lệnh khối .....	124
III.3.3. Cấu trúc if.....	126
III.3.4. Cấu trúc lựa chọn switch .....	127
III.3.5. Vòng lặp for.....	130
III.3.6. Vòng lặp while và do – while .....	132
III.3.7. Các lệnh thay đổi cấu trúc lập trình.....	135
III.3.7.1. continue.....	136
III.3.7.2. break .....	137
<b>III.4. Mảng và chuỗi ký tự (5 tiết LT) .....</b>	<b>138</b>
III.4.1. Mảng.....	138
III.4.1.1. Khái niệm mảng .....	138
III.4.1.2. Khai báo và sử dụng mảng .....	138
III.4.1.3. Các thao tác cơ bản trên mảng .....	139
a. Nhập dữ liệu cho mảng.....	139
b. Xuất dữ liệu chứa trong mảng.....	140
c. Tìm phần tử có giá trị lớn nhất, phần tử có giá trị nhỏ nhất.....	142
III.4.1.4. Tìm kiếm trên mảng .....	142
III.4.1.5. Sắp xếp mảng .....	144
III.4.2. Chuỗi ký tự (2 tiết LT).....	146
III.4.2.1. Khái niệm chuỗi ký tự .....	146

III.4.2.2. Khai báo và sử dụng xâu .....	147
a. Khai báo xâu kí tự.....	147
b. Truy nhập vào một phần tử của xâu .....	147
III.4.2.3. Các hàm xử lý ký tự .....	147
III.4.2.4. Các hàm xử lý xâu.....	149
a. Vào ra dữ liệu .....	149
b. Một số hàm xử lý xâu kí tự khác.....	149
<b>III.5. Cấu trúc (2 tiết LT) .....</b>	<b>151</b>
III.5.1. Khái niệm cấu trúc.....	151
III.5.2. Khai báo và sử dụng cấu trúc .....	152
III.5.2.1. Khai báo kiểu dữ liệu cấu trúc.....	152
III.5.2.2. Khai báo biến cấu trúc:.....	152
III.5.2.3. Định nghĩa kiểu dữ liệu cấu trúc với typedef.....	153
III.5.3. Xử lý dữ liệu cấu trúc .....	154
III.5.3.1. Truy nhập các trường dữ liệu của cấu trúc.....	154
III.5.3.2. Phép gán giữa các biến cấu trúc .....	156
<b>III.6. Hàm (2 tiết LT) .....</b>	<b>157</b>
III.6.1. Khái niệm hàm.....	157
III.6.1.1. Khái niệm chương trình con.....	157
III.6.1.2. Phân loại chương trình con: .....	158
III.6.2. Khai báo và sử dụng hàm .....	158
III.6.2.1. Khai báo hàm .....	158
III.6.2.2. Sử dụng hàm.....	161
III.6.3. Phạm vi của biến.....	163

## BUỔI 1.

### PHẦN 1. TIN HỌC CĂN BẢN

(16 tiết Lý thuyết, 4 tiết Bài tập, 6 tiết Thực hành)

#### I.1. Thông tin và biểu diễn thông tin (5T lý thuyết)

##### I.1.1. Các khái niệm cơ bản về thông tin và tin học (1T lý thuyết)

###### I.1.1.1. Thông tin và xử lý thông tin

###### a. Thông tin - Dữ liệu – Tri thức

###### Thông tin- Information

Khái niệm thông tin (information) được sử dụng thường ngày. Thông tin mang lại cho con người sự hiểu biết, nhận thức tốt hơn về những đối tượng trong đời sống xã hội, trong thiên nhiên,... giúp cho họ thực hiện hợp lý công việc cần làm để đạt tới mục đích một cách tốt nhất.

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 một cách khác, thông tin là ngữ cảnh trong đó dữ liệu được xem xét

###### Dữ liệu - Data

Dữ liệu (data) là biểu diễn của thông tin được thể hiện bằng các tín hiệu vật lý. Thông tin chứa đựng ý nghĩa còn dữ liệu là các sự kiện không có cấu trúc và không có ý nghĩa nếu chúng không được tổ chức và xử lý.

Dữ liệu trong thực tế có thể là:

- Các số liệu thường được mô tả bằng số như trong các bảng biểu
- Các ký hiệu qui ước, ví dụ chữ viết
- Các tín hiệu vật lý ví dụ như ánh sáng, âm thanh, nhiệt độ, áp suất,...

Theo quan niệm chung của những người làm công nghệ thông tin thì thông tin là những hiểu biết của chúng ta về một lĩnh vực nào đấy, còn dữ liệu là thông tin được biểu diễn và xử lý trong máy tính.

###### Tri thức – Knowledge

Tri thức theo nghĩa thường là thông tin ở mức trừu tượng hơn. Tri thức khá đa dạng, nó có thể là sự kiện, là thông tin và cách mà một người thu thập được qua kinh nghiệm hoặc qua đào tạo. Nó có thể là sự hiểu biết chung hay về một lĩnh vực cụ thể nào đó. Thuật ngữ tri thức được sử dụng theo nghĩa “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 chúng để tạo nên thông tin có ý nghĩa hoặc dữ liệu mới.

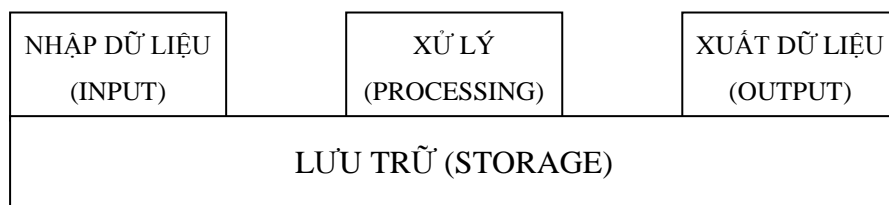




### ***b. Quy trình xử lý thông tin***

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

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



### ***Xử lý thông tin bằng máy tính điện tử***

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

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

#### **1.1.1.2. Máy tính điện tử và phân loại**

##### ***a. Lịch sử phát triển của máy tính điện tử***

Do nhu cầu cần tăng độ chính xác tính toán và giảm thời gian tính toán, con người đã quan tâm chế tạo các công cụ tính toán từ xưa: bàn tính tay 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 Wilhelm von Leibniz (1646 - 1716), máy sai phân để tính các đa thức toán học ...

Tuy nhiên, máy tính điện tử thực sự bắt đầu hình thành vào thập niên 1950 và đến nay đã trải qua 5 thế hệ và 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/s. Loại máy tính điển hình thế hệ 1 như EDVAC (Mỹ) hay BESM (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/s. 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 triệu phép tính/s. 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 hàng chục triệu đến hàng tỷ phép tính/s. Giai đoạn này hình thành 2 loại máy tính chính: máy tính cá nhân để bàn (Personal Computer - PC) hoặc xách tay (Laptop hoặc Notebook computer) và 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 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

### **b. Phân loại máy tính điện tử**

Trên thực tế tồn tại nhiều cách phân loại máy tính khác nhau và chúng ta có thể phân loại máy tính theo hiệu năng tính toán như sau:

- **Máy Vi tính (Microcomputer or 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) 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 (SuperComputer)** 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.

### **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 xướng trong 1 bài báo *Informatik: Automatische Informationsverarbeitung* (i.e. "Informatics: automatic information processing"). Sau đó vào năm 1962, [Philippe Dreyfus](#) người Pháp gọi là "informatique", tiếp theo là [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 2 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 (Information Technology - IT)***

Thuật ngữ Công nghệ thông tin xuất hiện ở Việt nam vào những năm 90 của thế kỷ 20. Theo [Information Technology Association of America](#) (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 NQ49 CP thì “Công nghệ thông tin là...”

Các ứng dụng ngày nay của IT:

- Quản trị dữ liệu
- Thiết kế hệ thống cơ sở dữ liệu
- Quản lý hệ thống thông tin
- Quản lý hệ thống
- ....

### ***c. Công nghệ thông tin và truyền thông***

Ngày nay, khuynh hướng sử dụng "[information](#)" thay thế cho "[data](#)" và có xu thế mở rộng cho lĩnh vực truyền thông và trở thành **ICT** (*Information and Communication Technology*). Thuần túy 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 nhau trong một phạm vi địa lý nhỏ. Tuy nhiên, nhiều máy tính có thể kết nối với nhau theo một phạm vi rộng hơn và việc trao đổi thực hiện qua một mạng viễn thông nào đó. **Internet** - *Mạng máy tính toàn cầu* là một phát minh vĩ đại của nhân loại trong thế kỷ 20, đó cũng chính là sản phẩm của ngành Công nghệ thông tin và Truyền thông.

## **I.1.2. Biểu diễn dữ liệu trong máy tính (4T lý thuyết)**

### **I.1.2.1. Biểu diễn số trong các hệ đếm**

Hệ đếm là tập hợp các ký hiệu và qui tắc sử dụng tập ký hiệu đó để biểu diễn và xác định các giá trị các số. Mỗi hệ đếm có một số ký số (digits) hữu hạn. Tổng số ký số của mỗi hệ đếm được gọi là **cơ số** (base hay 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  $b$  ký số để thể hiện giá trị số. Ký số nhỏ nhất là **0** và lớn nhất là  **$b-1$** .
- Giá trị vị trí thứ  $n$  trong một số của hệ đếm bằng cơ số  $b$  lũy thừa  $n$ :  **$b^n$**
- Số  $N_{(b)}$  trong hệ đếm cơ số  $(b)$  được biểu diễn bởi:

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

trong đó, số  $N_{(b)}$  có  **$n+1$**  ký số biểu diễn cho phần nguyên và  **$m$**  ký số lẻ biểu diễn cho phần  $b$ \_phân, và có giá trị là:

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

hay là:

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

Trong ngành toán - tin học hiện nay phổ biến 4 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 10 ký số theo ký hiệu 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 10 đơn vị của hàng kế cận bên phải. Ở đây  $b=10$ . Bất kỳ số nguyên dương trong hệ thập phân có thể biểu diễn như là một tổng các số hạng, mỗi số hạng là tích 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:

$$\begin{aligned} 5246 &= 5 \times 10^3 + 2 \times 10^2 + 4 \times 10^1 + 6 \times 10^0 \\ &= 5 \times 1000 + 2 \times 100 + 4 \times 10 + 6 \times 1 \end{aligned}$$

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

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

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

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

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

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

$$101101 = 1011002 = 10110003 = \dots$$

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

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

### 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 2 chữ số là 0 và 1. Mỗi chữ số nhị phân gọi là BIT (viết tắt từ chữ BInary digiT). Vì hệ nhị phân chỉ có 2 trị số là 0 và 1, nên khi muốn diễn tả một số lớn hơn, hoặc các ký tự phức tạp hơn thì cần kết hợp nhiều bit với nhau. Ta có thể chuyển đổi 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 :

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

như vậy:

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

số  $10101_{(2)}$  sang hệ thập phân sẽ là:

$$10101_{(2)} = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 16 + 0 + 4 + 0 + 1 = 21_{(10)}$$

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

Nếu dùng 1 tập hợp 3 bit thì có thể biểu diễn 8 trị khác nhau : 000, 001, 010, 011, 100, 101, 110, 111. Các trị này tương đương với 8 trị trong hệ thập phân là 0, 1, 2, 3, 4, 5, 6, 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:

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

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

Ví dụ:

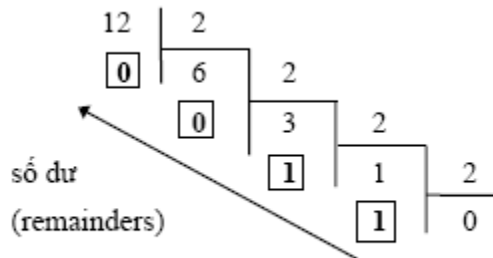
$$34F5C_{(16)} = 3 \times 16^4 + 4 \times 16^3 + 15 \times 16^2 + 5 \times 16^1 + 12 \times 16^0 = 216294_{(10)}$$

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

### 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: Số  $12_{(10)} = ?_{(2)}$ . Dùng phép chia cho 2 liên tiếp, ta có một loạt các số dư như sau:



Kết quả:  $12_{(10)} = 1100_{(2)}$

#### 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 3.11:  $0.6875_{(10)} = ?_{(2)}$

					phần nguyên của tích
0.6875	x 2	=	1	375	phần thập phân của tích
0.3750	x 2	=	0	.75	
0.75	x 2	=	1	.5	
0.5	x 2	=	1	.0	

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

### 1.1.2.2. Biểu diễn dữ liệu trong máy tính và đơn vị thông tin

#### a. Nguyên tắc chung

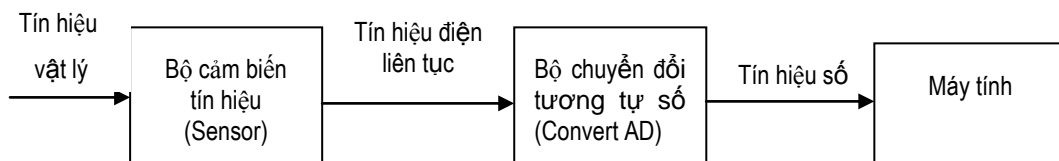
Thông tin và dữ liệu mà con người hiểu được tồn tại dưới nhiều dạng khác nhau, ví dụ như các số liệu, các ký tự văn bản, âm thanh, hình ảnh, ... nhưng trong máy tính mọi thông tin và dữ liệu đều được biểu diễn bằng số nhị phân (chuỗi bit).

Để đưa dữ liệu vào cho 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 ký 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, qui trình mã hoá được biểu diễn như sau:



**Hình I.2.1.1.a. 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 2 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 phẩy động. Để biểu diễn một dữ liệu cơ bản, người ta sử dụng 1 số bit. Các bit này ghép lại với nhau để tạo thành từ: từ 8 bit, từ 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. Tùy theo cách “ghép” 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 1 trong 2 trạng thái.

Ví dụ: Một mạch đèn có 2 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 ký 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.1.2.3. Biểu diễn số nguyên

Số nguyên gồm số nguyên không dấu và số nguyên có dấu. Về nguyên tắc đều dùng 1 chuỗi bit để biểu diễn. Đối với số nguyên có dấu, người ta sử dụng bit đầu tiên để biểu diễn dấu '-' và bit này gọi là bit dấu.

#### 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 1 dãy 8 bit biểu diễn số nguyên không dấu có giá trị:

$2^8 = 256$  số nguyên dương, cho giá trị từ 0 (0000 0000) đến 255 (1111 1111).

Với n bits ta có thể biểu diễn 1 số nguyên có giá trị lớn nhất là  $2^n - 1$  và dải giá trị biểu diễn được từ 0 đến  $2^n - 1$ .

Thí dụ: 00000000 = 0  
 00000010 = 2  
 00000100 = 4  
 .....  
 11111111 = 255

#### b. Số nguyên có dấu

Trong biểu diễn số nguyên có dấu, bit đầu làm bit dấu: 0 là số dương và 1 cho số âm. Số nguyên có dấu thể hiện trong máy tính ở dạng nhị phân là số dùng 1 bit làm bit dấu, người ta qui ước dùng bit ở hàng đầu tiên bên trái làm bit dấu (S): 0 là số dương và 1 cho số âm. Cách phổ biến biểu diễn số âm có dấu là dùng mã bù hai:

Số bù hai được tính như sau:

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

Thí dụ biểu diễn trên 8 bits:

37 = 00100101  
 Bù một (nghịch đảo) = 11011010  
 Bù hai (cộng thêm 1)

1  
 -----  
 11011011 ⇒ số -37

↓  
 Bit dấu

*Chú ý: Thử biểu diễn mã bù hai của -37 để thu được số +37*



### 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 bits ta thu được một số nguyên không dấu cũng n bits. 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 sẽ là sai.

Thí dụ với trường hợp 8 bits, tổng nhỏ hơn 255 thì ta sẽ có kết quả đúng:

$$\begin{array}{r} 57 = 00111001 \\ + 34 = 00100010 \\ \hline 91 = 01011011 \end{array}$$
  
$$\begin{array}{r} 209 = 11010001 \\ + 73 = 01001001 \\ \hline 282 = 100011010 \end{array}$$

↓

Bit tràn ra ngoài  $\Rightarrow$  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 qui 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  $-Y$ , sau đó cộng X với  $-Y$  theo nguyên tắc trên.

Thí dụ:  $97 - 52 = 97 + (-52)$

$$\begin{array}{r} + 97 = 0110\ 0001 \\ - 52 = 1100\ 1100 \\ \hline 45 = \textcircled{1} 0010\ 1101 \\ \text{Bỏ qua} \end{array}$$

Như vậy, khi thực hiện phép tính trên sẽ thừa ra 1 bit bên trái cùng, 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:

$$1011 \quad (11 \text{ cơ số } 10)$$

$$\begin{array}{r}
 1101 \qquad (13 \text{ cơ số } 10) \\
 \hline
 \begin{array}{r}
 1011 \\
 0000 \\
 1011 \\
 1011
 \end{array}
 \end{array}$$

**10001111                      kết quả 143 trong cơ số 10**

Chúng 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 qui 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 chia phức tạp hơn phép nhân nhưng dựa trên cùng 1 nguyên tắc. Hãy xem thí dụ sau:

$$\begin{array}{rcl}
 \text{Số bị chia} & \longrightarrow & 10010 \ 011 \mid 1011 \longleftarrow \text{số chia} \\
 & & \underline{1001} \\
 & & 001110 \qquad 1101 \longleftarrow \text{thương} \\
 & & \underline{1011} \\
 \text{Phần dư riêng} & \longrightarrow & 001111 \\
 & & \underline{1011} \\
 & & 100 \longleftarrow \text{phần dư}
 \end{array}$$

Phép chia với số nguyên sẽ cho 2 kết quả là thương và phần dư.

## BUỔI 2.

### I.1.2.4. Biểu diễn số thực

#### a. Nguyên tắc chung

Để biểu diễn số thực, trong máy tính người ta dùng ký pháp dấu phẩy động (Floating Point Number). Một cách tổng quát, một số thực biểu diễn theo cách này gồm 3 thành phần:

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

Với M: phần định trị (Mantissa), N là cơ số: (Radix), còn E là phần số mũ (Exponent)

Cơ số thường 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 2 số thực  $N_1$  và  $N_2$  được lưu trữ theo phần định trị và số mũ như sau:

$$M_1 = -15 \quad \text{và} \quad E_1 = +12$$

$$M_2 = +314 \quad \text{và} \quad E_2 = -9$$

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

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

Rõ ràng rằng, việc lưu trữ phần định trị và phần số mũ sẽ dễ dàng và đơn giản nhiều so với việc lưu trữ giá trị đúng của nó.

Khi thực hiện phép toán với số dấu chấm động sẽ được tiến hành trên cơ sở các giá trị của phần định trị và phần mũ. Giả sử có 2 số dấu phẩy động sau:

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

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

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

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

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

Chú ý: Với số thực biểu diễn theo dấu phẩy 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 giá trị bằng không, vì thế, có một số trường hợp phải qui ước:

- Nếu tất cả các bit của E và M đều bằng không, 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 IEEE754/85

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

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



phần dấu chấm thập phân

- 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 phẩy 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 một cách khác, số 1 không biểu diễn (bỏ)

- Cơ số phần mũ được hiểu là cơ số 2.  
Thí dụ: số thực +5 sẽ được biểu diễn như sau:

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

Nếu  $101_2$  trượt phải 2 bit sẽ trở thành  $1.01_2$ ,  $2^{-2}$  lần từ giá trị ban đầu. Với mục đích chuẩn 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.1.2.5. 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 Codes for Information Interchangeable) dùng 8 bit cho 1 ký tự, bộ mã Unicode dùng 16 bit. Đây là 2 bộ mã thông dụng. Thí dụ, với bộ mã ASCII, chữ A có mã là  $65 = 01000001$ .

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 1 ký tự.

#### b. Bộ mã ASCII

ASCII là bộ mã được dùng để *trao đổi thông tin chuẩn của Mỹ*. Lúc đầu chỉ dùng 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.

Trong bộ mã hoá 8 bit, các mã từ 32 đến 126 biểu diễn cho các ký tự hiển thị được gồm 52 ký tự la tinh: 26 thường và 26 hoa.. Tiếp theo là 10 mã cho 10 chữ số (mã 30 đến mã 39). Còn lại cho các ký tự phân cách, dấu phép toán.









Chú ý là 32 mã đầu tiên và (00 đến 1F0 và mã cuối cùng 127 (trong bộ mã chuẩn 128 mã) biểu diễn cho các thông tin điều khiển. Các mã mở rộng từ 128 đến 255 là tập các ký tự có thể thay đổi được bởi các nhà chế tạo máy tính hoặc các nhà phát triển phần mềm. Bộ mã ASCII được minh hoạ qua bảng dưới đây:

**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

<b>5</b>	♣ 5	NAK 21	% 37	5 53	E 69	U 85	e 101	u 117
<b>6</b>	♠ 6	SYN 22	& 38	6 54	F 70	V 86	f 102	v 118
<b>7</b>	BEL 7	ETB 23	‘ 39	7 55	G 71	W 87	g 103	w 119
<b>8</b>	BS 8	CAN 24	( 40	8 56	H 72	X 88	h 104	x 120
<b>9</b>	HT 9	EM 25	) 41	9 57	I 73	Y 89	I 105	y 121
<b>A</b>	LF 10	SUB 26	* 42	: 58	J 74	Z 90	j 106	z 122
<b>B</b>	VT 11	ESC 27	+ 43	; 59	K 75	[ 91	k 107	{ 123
<b>C</b>	FF 12	FS 28	, 44	< 60	L 76	\ 92	l 108	 124
<b>D</b>	CR 13	GS 29	- 45	= 61	M 77	] 93	m 109	} 125
<b>E</b>	SO 14	RS 30	. 46	> 62	N 78	^ 94	n 110	~ 126
<b>F</b>	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	Ł 192	Ⓔ 208	α 224	≡ 240
1	ü 129	æ 145	í 161	 177	⌊ 193	⌋ 209	β 225	± 241
2	é 130	Æ 146	ó 162	 178	⌈ 194	⌋ 210	Γ 226	≥ 242
3	â 131	ô 147	ú 163	 179	⌋ 195	Ⓔ 211	π 227	≤ 243
4	ä 132	ö 148	ñ 164	⌋ 180	— 196	Ⓔ 212	Σ 228	∫ 244
5	à 133	ò 149	Ñ 165	⌋ 181	⌋ 197	Ⓔ 213	σ 229	∫ 245
6	å 134	û 150	ª 166	⌋ 182	⌋ 198	Ⓔ 214	μ 230	÷ 246
7	ç 135	ù 151	º 167	⌋ 183	⌋ 199	Ⓔ 215	τ 231	≈ 247
8	ê 136	ÿ 152	¿ 168	⌋ 184	Ⓔ 200	⌋ 216	Φ 232	° 248
9	ë 137	Ö 153	¬ 169	⌋ 185	Ⓔ 201	⌋ 217	Θ 233	· 249
A	è 138	Û 154	¬ 170	⌋ 186	Ⓔ 202	⌋ 218	Ω 234	· 250
B	ï 139	ç 155	½ 171	⌋ 187	⌋ 203	 219	δ 235	√ 251
C	î 140	£ 156	¼ 172	⌋ 188	⌋ 204	 220	∞ 236	ⁿ 252
D	ì 141	¥ 157	¡ 173	⌋ 189	= 205	 221	φ 237	² 253
E	Ä 142	Þ 158	« 174	⌋ 190	⌋ 206	 222	ε 238	■ 254
F	Å 143	ƒ 159	» 175	⌋ 191	⌋ 207	 223	∩ 239	

#### c. Bộ mã Unicode

Ngày nay, máy tính đã toàn cầu hoá, việc trao đổi thông tin ngày càng mở rộng. Để đáp ứng nhu cầu toàn cầu hoá này, vào những năm 90 của thế kỷ trước, các hãng hàng đầu về máy tính đưa ra bộ mã 16 bit mang tên Unicode. Vậy số ký tự có thể biểu diễn (mã hoá) là  $2^{16}-1$ .

## I.2. HỆ THỐNG MÁY TÍNH (7 tiết)

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 2 phần chính: phần cứng và phần mềm.

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

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

### I.2.1. Hệ thống máy tính (3T lý thuyết)

#### I.2.1.1. Tổ chức bên trong máy tính

##### a. Mô hình cơ bản của máy tính

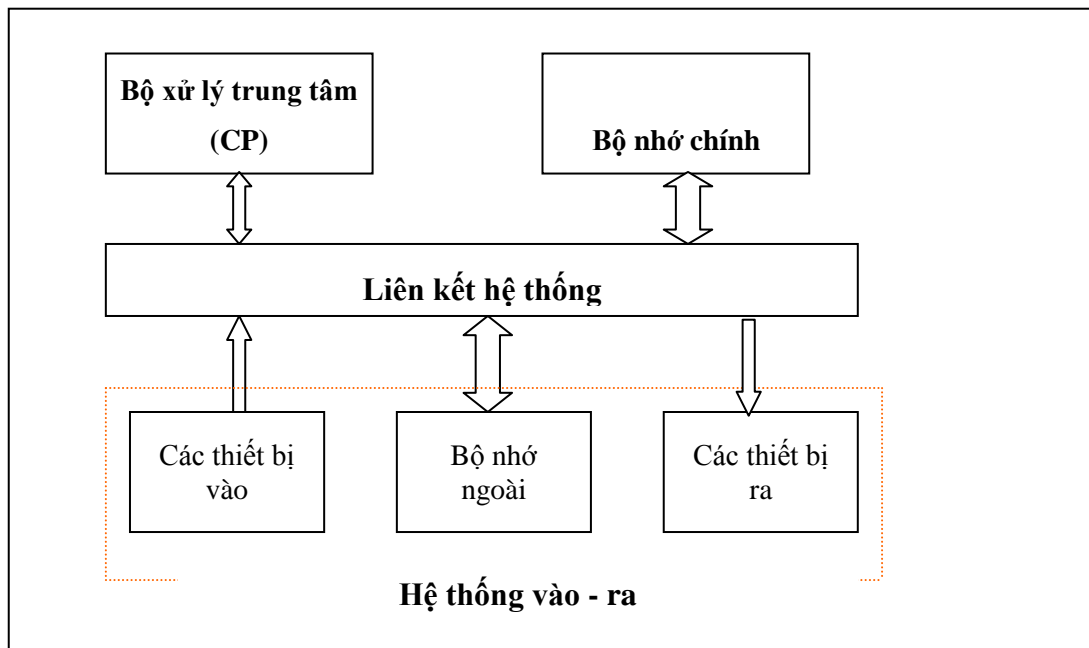
*Chức năng của hệ thống máy tính*

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 liệu 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 chúng 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.
- **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 Processor Unit – CPU), bộ nhớ chính (Main Memory), hệ thống vào ra (Input-Output System) và liên kết hệ thống (Buses) như chỉ ra trong hình 3.1 dưới đây, với các chức năng chính của các thành phần:



**Hình I.2.1.1.a. Các thành phần chính của hệ thống máy tính**

- **Bộ xử lý trung tâm – 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 thông tin giữa CPU, bộ nhớ chính và hệ thống vào ra của máy tính với nhau.

#### *Hoạt động của máy tính.*

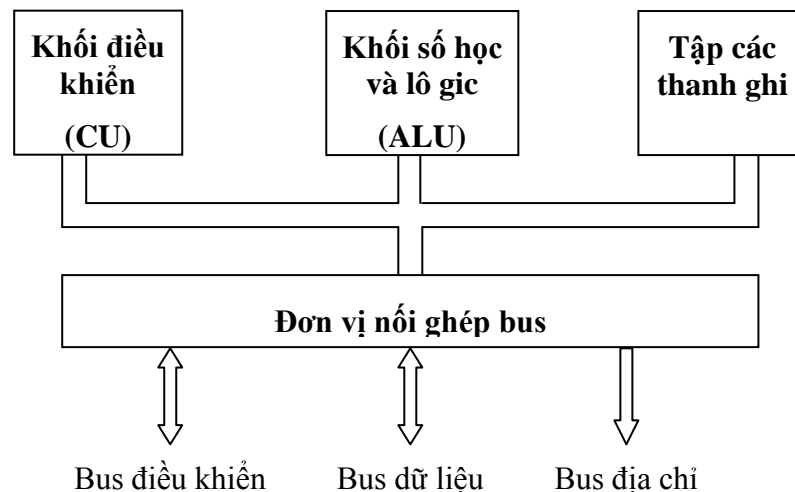
Hoạt động cơ bản của máy tính là thực hiện chương trình. Chương trình gồm một tập các lệnh được lưu trữ trong bộ nhớ. Việc thực hiện chương trình là việc lặp lại *chu trình lệnh* bao gồm các bước sau:

- 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 công 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 Processor 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 thi lệnh. Trong quá trình thực hiện lệnh, CPU có trao đổi với bộ nhớ chính và hệ thống vào ra. CPU có 3 bộ phận chính: khối điều khiển, khối tính toán số học và logic, và tập các thanh ghi (hình 3.2).



**Hình I.2.1.1.b. 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 nhất là trong dấu phẩy động. Khối tính toán hiện nay được bổ sung thêm khối tính toán dấu phẩy động FPU (Floating Point Unit)- hay còn gọi là bộ đồng xử lý (Co-processor Unit) .

- **Tập các thanh ghi** (*Registers*)

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 trong 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. Thí dụ:

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

*Chú ý:* Nội dung của ngăn nhớ có thể thay đổi còn địa chỉ ngăn nhớ thì cố định.

Bộ nhớ chính của máy tính được thiết kế bằng bộ nhớ bán dẫn với 2 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 thể 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 128 MB, 256 MB, 512 MB 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 thuộc bộ nhớ RAM, 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 nó lại thuộc hệ thống vào ra*. Có thể cất giữ và di chuyển bộ nhớ ngoài độc lập với máy tính. Hiện nay có các loại bộ nhớ ngoài phổ biến như:

- Đĩa mềm (Floppy disk) : là loại đĩa đường kính 3.5 inch dung lượng 1.44 MB.
- Đĩa cứng (Hard disk) : phổ biến là đĩa cứng có dung lượng 20 GB, 30 GB, 40 GB, 60 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).
- Các loại bộ nhớ ngoài khác như thẻ nhớ (Memory Stick, Compact Flash Card), USB Flash Drive có dung lượng phổ biến là 32 MB, 64 MB, 128 MB, ...



Floppy disk



Compact disk



Compact Flash Card



USB Flash Drive

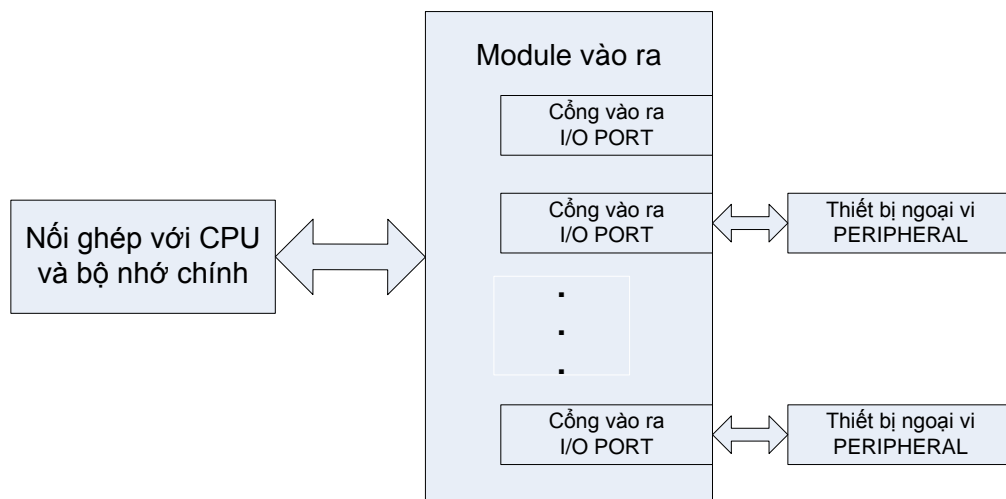
Hình 1.2.1.1.c. Một số loại bộ nhớ

### *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.



### 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 như bàn phím, màn hình, máy in hay một máy tính khác. Người ta có thể phân các thiết bị ngoại vi ra nhiều loại:

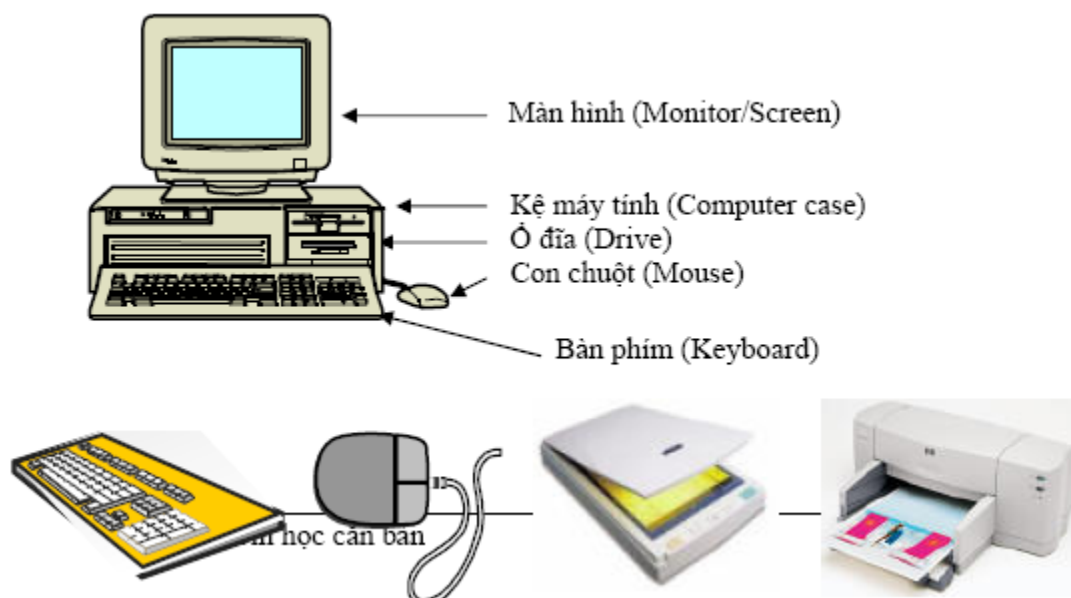
- *Thiết bị thu nhận dữ liệu:* Bàn phím, chuột, máy quét ảnh,...
- *Thiết bị hiển thị dữ liệu:* màn hình, máy in,...
- *Thiết bị nhớ:* các loại ổ đĩa
- *Thiết bị truyền thông:* modem
- *Thiết bị hỗ trợ đa phương tiện:* hệ thống âm thanh, hình ảnh,...

Các 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 3 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 (xóa), 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.



**Hình I.2.1.1.d. 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 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 viên bi hoặc tia sáng (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 nhập văn bản hay hình vẽ, hình chụp vào máy tính. Thông tin nguyên thủy trên giấy sẽ được quét thành các tín hiệu số tạo thành các tập tin ảnh (image file).

Các thiết bị ra:

- **Màn hình** (Screen 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 1280 X 1024 pixel.
- **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 Seminar, 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. Tùy theo nhiệm vụ của chúng mà chúng ta phân làm 3 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 2 chiều. Các máy tính hiện nay thường có đường bit dữ liệu 32 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 là 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  $2^n$ .

### 1.2.1.2. Phần mềm máy tính

Việc xử lý thông tin của máy tính theo yêu cầu người dùng được tiến hành theo một qui 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 qui trình/chương trình. Ngày nay, người ta quen sử dụng một thuật ngữ tương đương song theo nghĩa rộng hơn là phần mềm máy tính (Software Computer). Phần dưới đây chúng ta 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 2 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.

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

Việc biểu diễn dữ liệu một cách trừu tượng là một vấn đề khá phức tạp, thí 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 ở đây mà sẽ trình bày trong một giáo trình khác.

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). Có nhiều cách diễn đạt khác nhau về thuật ngữ này, song chúng ta có thể diễn đạt như sau:

*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.*

Để hiểu rõ về thuật toán chúng ta xét một số thí dụ sau:

**Thí dụ 1:** Tìm giá trị lớn nhất của một dãy số nguyên có  $N$  số

- Đầu vào: số số nguyên dương  $N$  và  $N$  số nguyên  $a_1, a_2, \dots, a_N$

- Đầu ra: số nguyên lớn nhất của dãy  $a_k$ ,  $k$  trong khoảng  $[1 \dots N]$

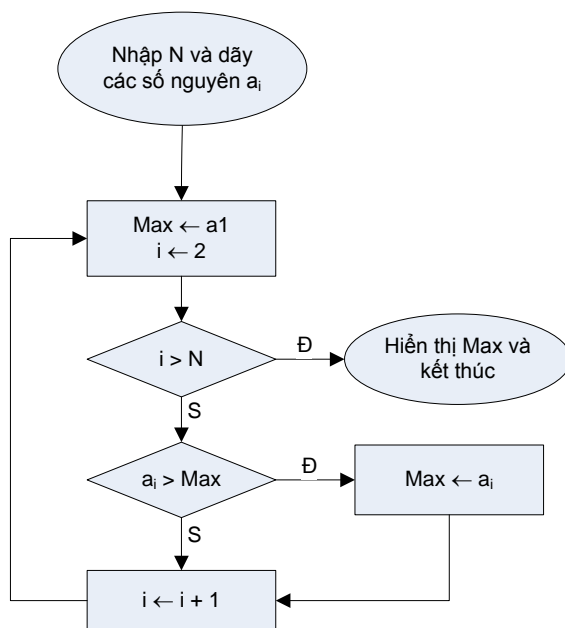
#### Ý tưởng

- Khởi tạo giá trị  $\text{Max} = a_1$
- Lần lượt so sánh  $\text{Max}$  với  $i=2, 3, \dots, N$ , nếu  $a_i > \text{Max}$  ta gán giá trị mới cho  $\text{Max}$ .

#### Thuật toán

- B1: Nhập dãy số  $a_i$ .
- B2:  $\text{Max} \leftarrow a_1$ .
- B3: Nếu  $i > N$ , thuật toán kết thúc và  $\text{Max}$  là giá trị lớn nhất của dãy cần tìm
- B4: Nếu  $a_i > \text{Max}$ , gán  $a_i$  cho  $\text{Max}$ .
- B5: Tăng  $i$  lên 1 đơn vị.
- B6: Quay lên b4.
- B7: Kết thúc.

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



#### Qui ước:

- Hình thoi biểu diễn thao tác so sánh
- Hình chữ nhật thể hiện 1 thao tác tính toán đơn giản hay phức tạp
- Hình ô van thể hiện thao tác bắt đầu hay kết thúc.

#### Thí dụ 2: Sắp xếp bằng phương pháp trao đổi (Exchange Sort)

- Đầu vào: Dãy A gồm N số nguyên  $a_1, a_2, \dots, a_N$
- Đầu ra: Dãy A được sắp lại theo thứ tự không giảm

### *Ý tưởng*

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

### *Thuật toán*

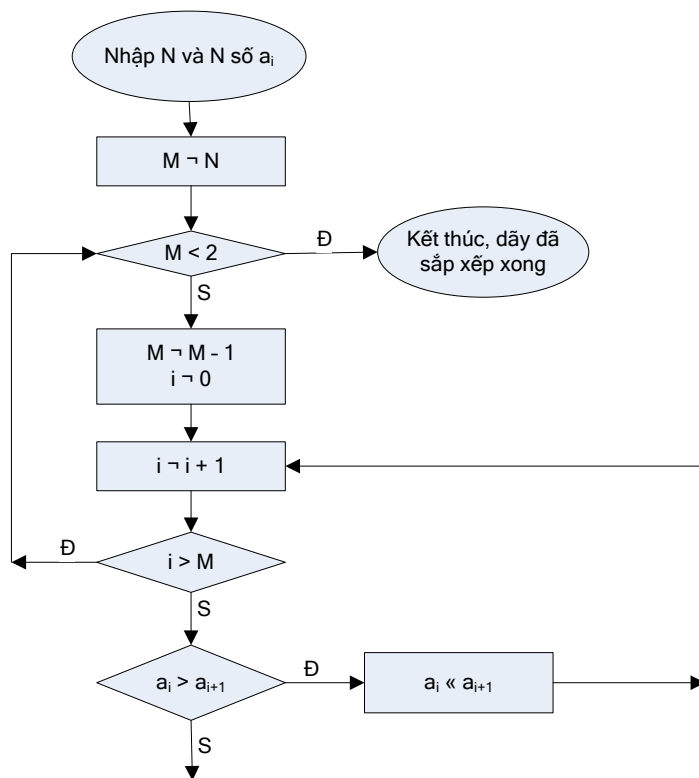
- B1: Nhập số  $N$  và dãy số  $a_1, a_2, \dots, a_N$
- B2:  $M \leftarrow N$ .
- B3: Nếu  $M < 2$  thì thuật toán kết thúc và hiển thị dãy đó.
- B4:  $M \leftarrow M - 1, i \leftarrow 0$ .
- B5: Tăng  $i$  lên 1 đơn vị.
- B6: Nếu  $i > M$  thì quay lại bước 3.
- B7: Nếu  $a_i > a_{i+1}$  thì trao đổi hai số đó cho nhau .
- B8: Quay lên bước 5.

### *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 trao 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  $M$  được gán với giá trị  $N$  và dừng khi  $M < 2$ .
- Trong thuật toán trên,  $i$  được khởi tạo giá trị 0 và tiến tới  $M + 1$

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





Ngoài cách biểu diễn theo kiểu liệt kê hay sơ đồ khối, còn nhiều cách biểu diễn khác. Xu hướng chung hiện nay là sử dụng các ngôn ngữ lập trình như Pascal, C/C++ hay Java. Tuy nhiên, không nhất thiết phải sử dụng đúng ký pháp của các ngôn ngữ đó mà có thể được bỏ một số ràng buộc.

Cũng có khi người ta dùng một giả ngữ gọi là ngôn ngữ mô phỏng chương trình PDL (Programming Description Language).

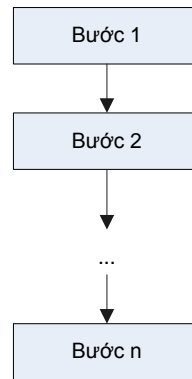
Điều quan trọng của giải thuật mà chúng ta công nhận ở đây là giải thuật phải thoả mãn 3 tiêu chí sau:

1. Tính hữu hạn: giải thuật phải dừng sau một thời gian hữu hạn.
2. Khi kết thúc, giải thuật phải cung cấp kết quả đúng đắn.
3. Tính hiệu quả: thời gian tính toán nhanh, sử dụng ít tài nguyên không gian như bộ nhớ, thiết bị,... Hơn nữa, giải thuật cần mang tính phổ dụng, dễ hiểu, dễ cài đặt và mở rộng cho các lớp bài toán khác.

Theo phong cách lập trình cấu trúc, người ta khuyến cáo, giải thuật được xây dựng với 3 cấu trúc cơ bản sau:

1. Cấu trúc tuần tự:

Các bước được thực hiện theo 1 trình tự tuyến tính, hết bước này đến bước khác



## 2. Cấu trúc rẽ nhánh

Việc thực hiện bước nào phụ thuộc vào điều kiện xác định. Thí dụ tìm max của 2 số  $a, b$ . Nếu  $a > b$  thì max là  $a$ , ngược lại max sẽ là  $b$ . Theo cách biểu diễn giải thuật ta có thể mô tả như sau:

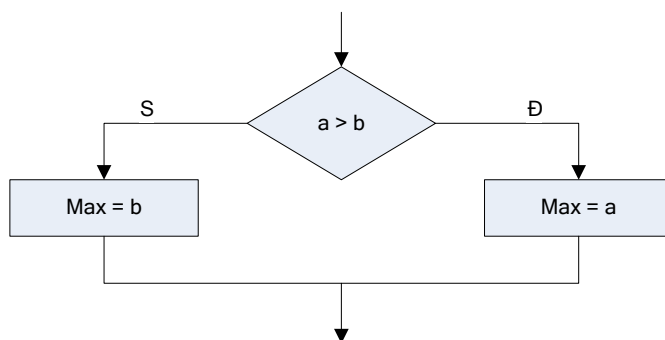
B1: Nhập 2 số  $a, b$ .

B2: Nếu  $a > b$  thì  $\text{max} = a$  và đi đến bước kết thúc (B4).

B3:  $\text{max} = b$  ( $a < b$ ).

B4: Kết thúc.

Cấu trúc rẽ nhánh này thường được biểu diễn dưới dạng sơ đồ như sau:

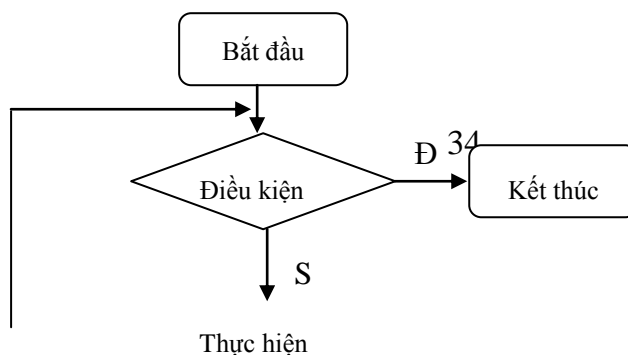


Cấu trúc rẽ nhánh (2 nhánh) có thể mở rộng cho nhiều nhánh

## 3. Cấu trúc lặp

Một tác động/ nhiệm vụ có thể được thực hiện lặp nhiều lần. Tuy nhiên số lần lặp phải hữu hạn, và số lần lặp có thể biết trước hoặc không biết trước. Thí dụ tìm số lớn nhất của 1 dãy có  $n$  số. Để thực hiện yêu cầu này, ta lần lượt phải so sánh số max tạm thời (lúc đầu max được gán bằng phần tử thứ nhất,  $a_1$ ) với  $a_i$ , với  $i$  từ 2, 3, ...,  $n$ . Việc so sánh này được thực hiện lặp nhiều lần giữa max và  $a_i$ . Khi kết thúc quá trình lặp, ta sẽ thu được số max, số lớn nhất của dãy  $n$  số.

Sơ đồ chung của cấu trúc lặp:



## 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, 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:

$$\text{Chương trình} = \text{Giải thuật} + \text{Cấu trúc dữ liệu}$$

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 3 nhóm:

### 1. Ngôn ngữ máy

Mỗi loại máy tính đều có ngôn ngữ máy riêng. Đó chính 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 nó 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 thì quả 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. Thí dụ để cộng nội dung của 2 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 có thể 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 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 có thể 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

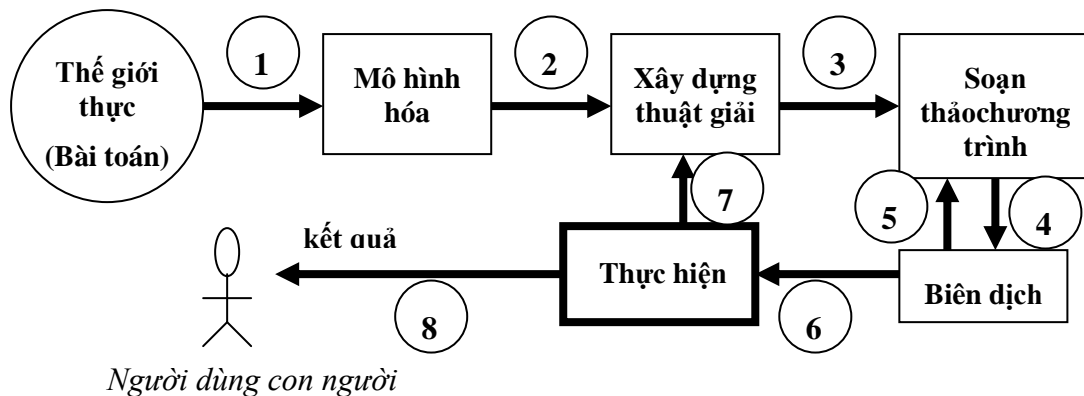
của chương trình đến lệnh cuối cùng nếu không có lỗi. Bộ thông dịch này giống như vai trò của 1 thông dịch viên (translator).

- **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 1 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 đến nay đã không còn được sử dụng hay đã được hoàn thiện hơn như Visual Basic có nguồn gốc từ Basic. Nhiều ngôn ngữ mới ra đời như C, C++, Visual C, Java, C#, Python,...

Quá trình giải quyết một bài toán trên máy tính có thể biểu diễn theo một sơ đồ chung sau:



**Hình 3.5 Quy trình giải quyết một bài toán trên máy tính**

*Chú ý:*

- Bước 5: quay lại soạn thảo khi quá trình dịch phát hiện lỗi cú pháp trong chương trình nguồn
- Bước 7: xem lại giải thuật khi kết quả thực hiện không đúng (lỗi lô gíc).

Theo cách tiếp cận về phần mềm, các nhà khoa học và các nhà phát triển phần mềm đã định ra qui trình phát triển phần mềm máy tính bao gồm các bước sau:

- B1 Xác định bài toán: Thuật ngữ mới cho bước này là xác định yêu cầu người dùng, người mong muốn có phần mềm để sử dụng.
- B2 Phân tích bài toán: Tìm hiểu nhiệm vụ (chức năng) mà phần mềm cần xây dựng phải có và các dữ liệu cần thiết. Qua đó xây dựng các giải pháp khả thi. Nói một cách ngắn gọn, bước này tìm hiểu hệ thống là gì? Và làm gì?
- B3 Thiết kế hệ thống: thực hiện thiết kế kiến trúc hệ thống, thiết kế các mô đun chương trình, thiết kế giao tiếp, thiết kế an toàn,... Như vậy, nhiệm vụ thiết kế mô đun chính là xây dựng giải thuật cho mô đun đó và cách diễn tả giải thuật.
- B4 Xây dựng chương trình : Viết code cho các mô đun theo ngôn ngữ lập trình đã xác định.

- B6 Kiểm thử chương trình: nhằm kiểm tra tính đúng đắn của từng mô đun và cả hệ thống trước khi bàn giao cho khách hàng.
- B7 Triển khai: bước này gồm cả nhiệm vụ viết tài liệu phần mềm, hướng dẫn sử dụng và bảo trì phần mềm. Đây cũng là mục đích của phần mềm được yêu cầu và nhằm kéo dài vòng đời phần mềm (Software Life Cycle).

### c. Phân loại phần mềm máy tính

Có nhiều cách phân loại phần mềm máy tính. Nếu phân theo quan điểm sử dụng chung thì phần mềm máy tính có 2 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, 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 KH&KT (Eng.&Scie. SW)
- Phần mềm nhúng (Embedded SW)
- Phần mềm trên Web (Web-based SW)
- Phần mềm trí tuệ nhân tạo (IA SW)
- ....

## BUỔI 3.

### I.2.2. Mạng máy tính (2T lý thuyết)

#### I.2.2.1. 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 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 xử lý dữ liệu, hệ thống đường truyền như là hệ thống đường sá giao thông.

*Ví dụ về mạng:*

- Mạng tại Trung tâm Máy tính, Khoa CNTT, Trường ĐHBK 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. Qui mô và mức độ phức tạp của mạng ngày càng tăng.
- Hiện nay mạng máy tính phát triển rất mạnh ở mọi lĩnh vực 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. Mạng trở thành một yếu tố không thể thiếu của công nghệ thông tin nói riêng, cũng như đời sống nói chung.

### **I.2.2.2. 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

- *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 qui mô địa lý. Tùy theo qui mô địa lý, có thể phân ra ba loại chính là:

- *LAN (Local Area Network)* mạng cục bộ ở trong phạm vi nhỏ, ví dụ bán kính 500m, số lượng máy tính không quá nhiều, mạng không quá phức tạp.
- *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**

Hình vẽ minh họa mạng để trình bày...???

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.
- *Vì mạng*  
Vì 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 thông tin dữ liệu, là nơi trên đó thông tin 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). Ví dụ: ...?
- 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 sẵn có 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.

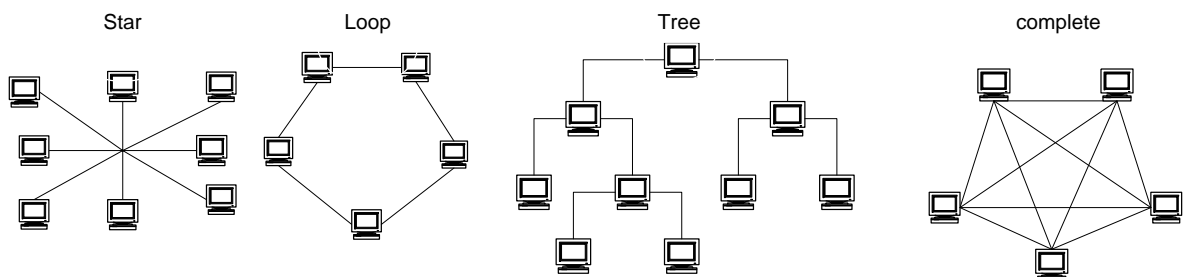
Ví dụ như Email, hệ quản trị cơ sở dữ liệu.

- Kiến trúc mạng máy tính

Kiến trúc mạng máy tính (network architecture) thể hiện cách kết nối máy tính với nhau và qui 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 qui ước truyền thông gọi là giao thức (protocol).

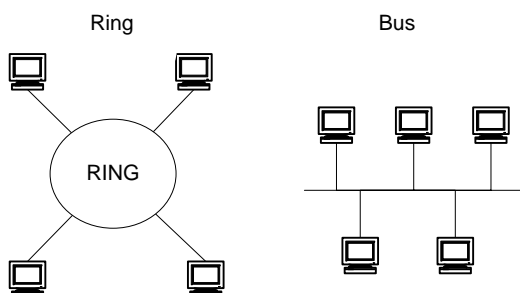
- Có hai kiểu nối mạng chủ yếu là điểm-điểm (point to point) và quảng bá (broadcast). Trong kiểu điểm-điểm các đường truyền nối các nút thành từng cặp. Như vậy một nút sẽ gửi dữ liệu đến nút lân cận nó (nút được nối trực tiếp với nó). Nút lân cận sẽ chuyển tiếp dữ liệu như vậy cho đến khi dữ liệu đến đích.

Kiểu nối mạng điểm- điểm có ba dạng chính là : hình sao (star), chu trình (loop), cây (tree) và đầy đủ (complete).



**Hình 4.5 : 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 1.2.2.3: Một số topo mạng kiểu quảng bá***

#### **I.2.2.4. Mạng Internet**

##### *Khái niệm về Internet*

*Internet* là một mạng máy tính có qui 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 ai cả. Chỉ có các uỷ ban điều phối và kỹ thuật giúp điều hành Internet.

Ban đầu là mạng của Bộ Quốc phòng Mỹ (DoD) dùng để đảm bảo liên lạc giữa các đơn vị quân đội. Sau đó phát triển thành mạng cho các trường đại học và viện nghiên cứu. Cuối cùng mạng có qui 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 thông 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 rẻ tiền hơn so với các phương tiện khác như điện thoại, fax. Internet ảnh hưởng đến toàn bộ thế giới đến mọi ngành, mọi lĩnh vực xã hội. Hiện nay Internet thành một yếu tố quan trọng không thiếu được trong thời đại hiện nay, có mặt ở mọi nơi, mọi lĩnh vực, mọi ngành.

##### *Làm sao để có được các dịch vụ Internet*

Để kết nối được Internet ta cần :

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



## BÀI TẬP TIN HỌC CƠ BẢN (2T BÀI TẬP)

....

### BUỔI 4.

#### I.2.3. Giới thiệu hệ điều hành (2T lý thuyết)

##### 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 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 hệ 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 ương, 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, nên 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 là DOS, TOS.
- + Máy tính lớn EC-1022 có hệ điều hành là 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. Tập (File)

Tập là tập hợp các dữ liệu có liên quan với nhau và được tổ chức theo 1 cấu trúc nào đó, thường được lưu trữ bên ngoài máy tính.

Nội dung của tập 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 qui ước đặt tên khác nhau, tên tập tin thường có 2 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. Phần tên do người tạo ra tập tin đặt. Với MS-DOS phần tên có tối đa là 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, Bai Tap.doc, ...

BaiTap.\*    BaiTap.doc, BaiTap.xls, BaiTap.ppt, BaiTap.dbf, ...

*Lưu ý: Nên đặt tên mang tính gợi nhớ.*

### **c. Quản lý tệp của hệ điều hành**

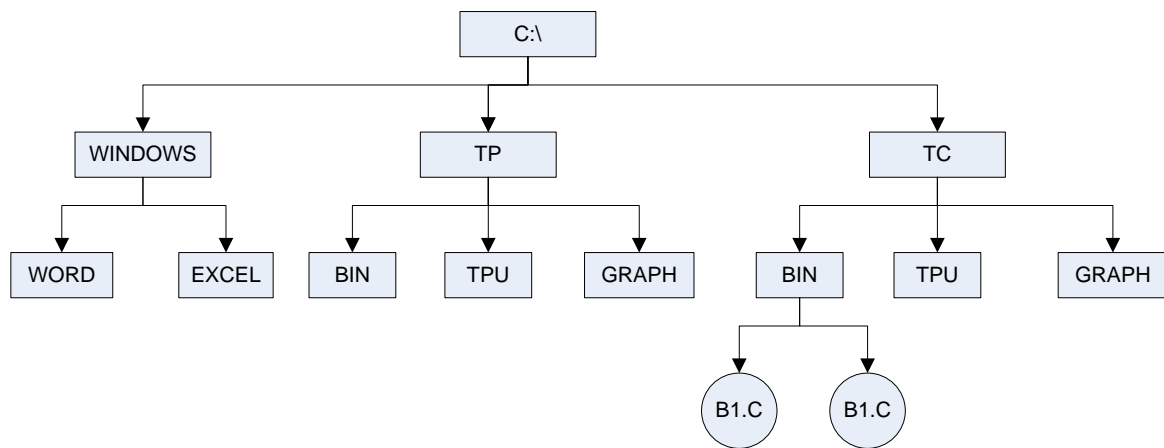
*Cấu trúc đĩa từ*

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 2 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 sector 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 1 hoặc nhiều sectors tùy 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 1 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:



Thư mục là nơi lưu giữ các tập tin 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 truy xuất. 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 xỏ 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 3 tệp cơ bản:

- + MSDOS.SYS - tệp.
- + 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, di 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

### 1.2.3.3. 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 liệt kê 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 ta gọi 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 chúng 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 được.

- 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 và 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,** đây là sản phẩm mới của MicroSoft hỗ trợ tốt cho các dịch vụ mạng, trò chơi, văn phòng...

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 (logging 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, v.v...) 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 như bên dưới.

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 v.v..) thì sẽ có thông báo để xử lý.

**Chú ý:** nếu không làm những thao tác đóng Windows như vừa nói ở trên mà tắt máy ngay thì có thể sẽ xảy ra việc thất lạc một phần của 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 1 trình ứng dụng).

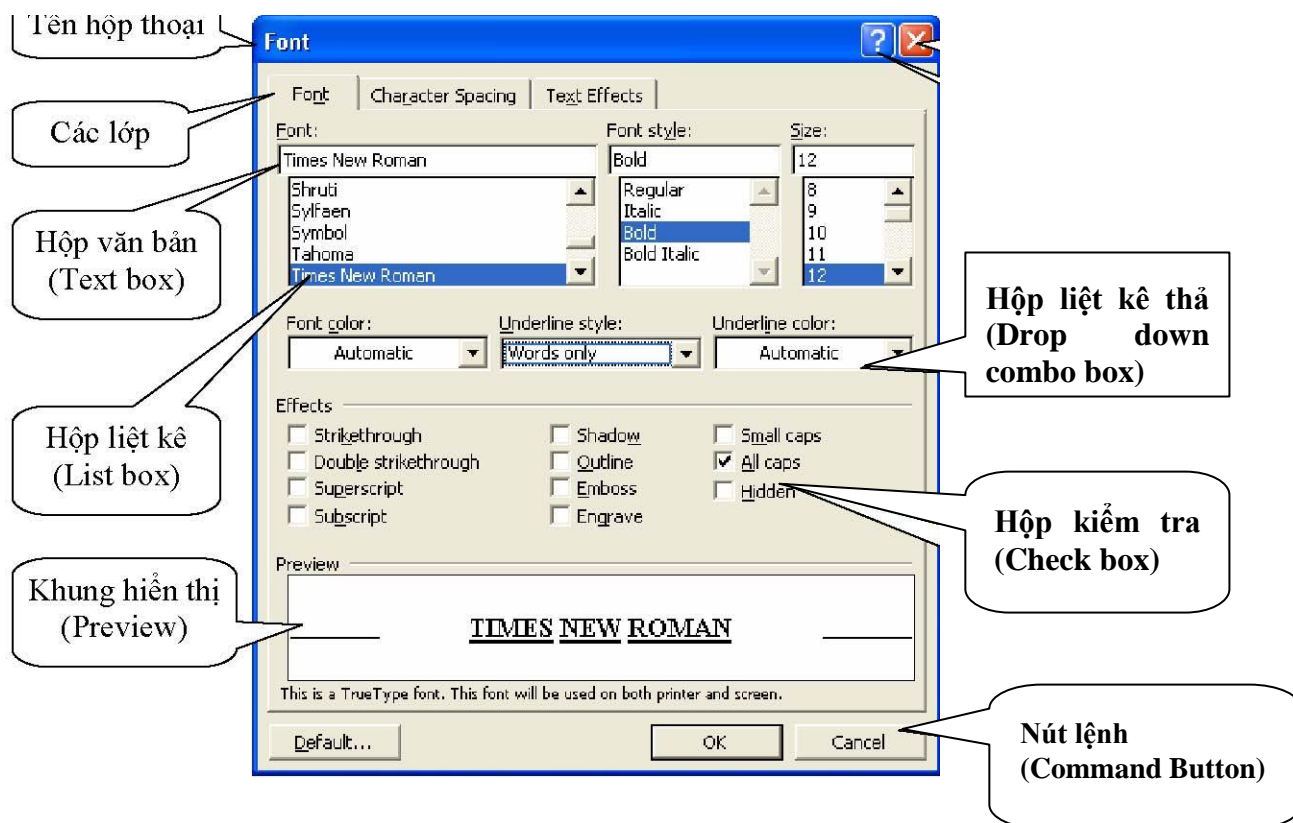
#### *Cửa sổ (Windows):*

Cửa sổ là khung giao tiếp đồ họa của 1 ứng dụng hoặc 1 lệnh.

- Bố cục của 1 cửa sổ : gồm thanh tiêu đề, thanh thực đơn, 1 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



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 bung xuống một danh sách.
- Hộp kiểm tra (Check box): có 2 dạng, dạng hình vuông thể hiện việc cho phép không chọn, chọn 1 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 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 làm việc được 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 nào cần thao tác với nút chuột phải sẽ mô tả rõ ràng.

### *d. Cấu hình Windows (Control Panel)*

Giới thiệu về Control Panel

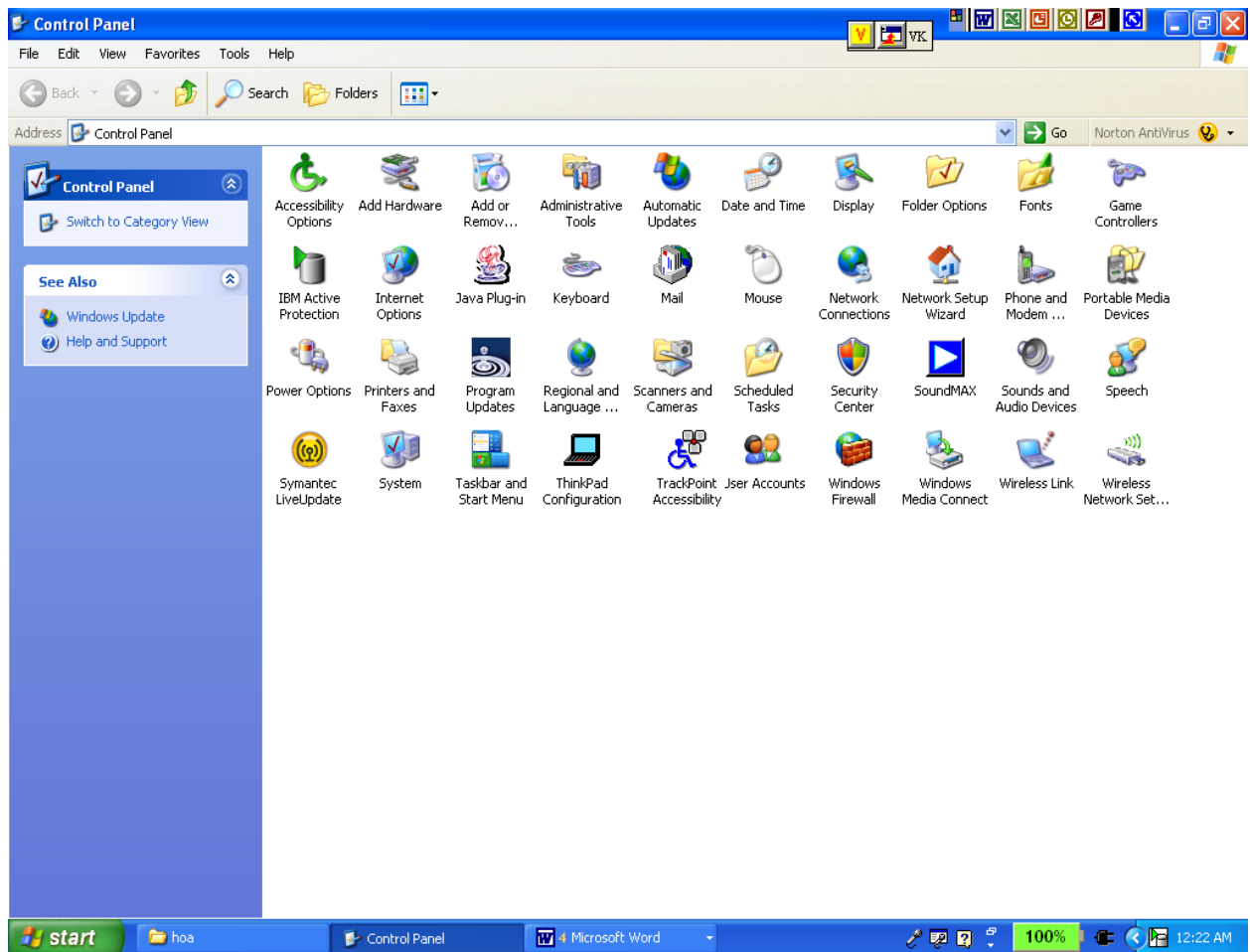
Control Panel là một chương trình cho phép người sử dụng xem và chỉnh sửa các tham số của hệ thống máy tính như dạng hiện của dữ liệu ngày tháng, dữ liệu số, thiết lập hoặc thay đổi cấu hình cho phù hợp với công việc hoặc sở thích của người dùng, cài đặt phần cứng, phần mềm.

#### *i. Khởi động chương trình Control Panel:*

Chọn lệnh Start / Settings / Control Panel



## ii. Cửa sổ làm việc của Control Panel:



## 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 bits) , 16 triệu màu (24 bits). 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:*

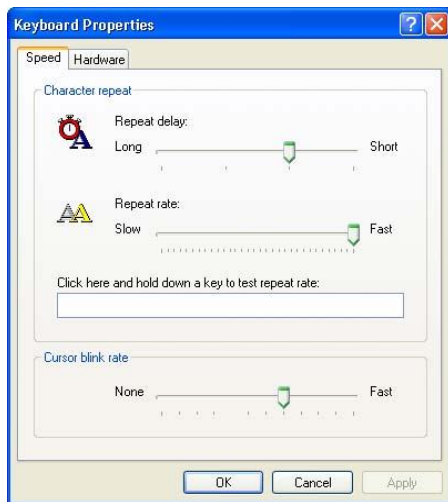
Để cài đặt các chương trình mới hoặc loại bỏ các chương trình không còn sử dụng bạn nhấn đúp chuột vào biểu tượng **Add or Remove Programs** trong cửa sổ Control Pane xuất hiện hộp thoại và thao tác theo chỉ dẫn.

*vi. Cấu hình ngày, giờ cho hệ thống*

Bạn có thể thay đổi ngày giờ của hệ thống bằng cách 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, rồi chọn biểu tượng Mouse



**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:**

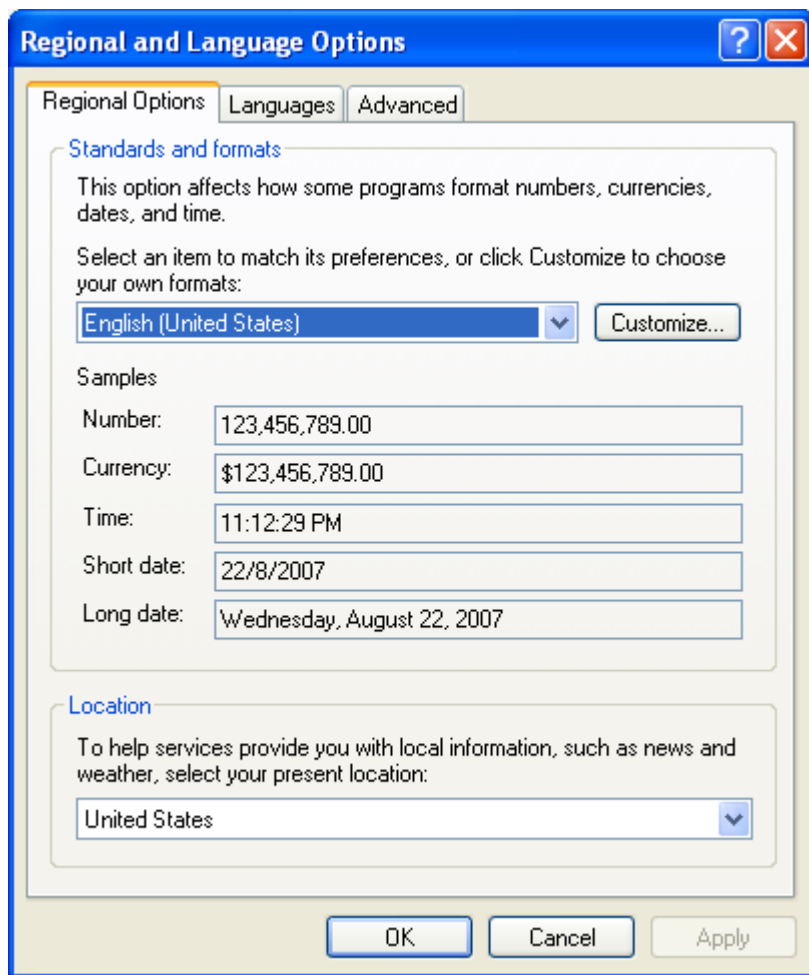
Lệnh Start/ Settings/ Control Panel, rồi chọn biểu tượng Keyboard

**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

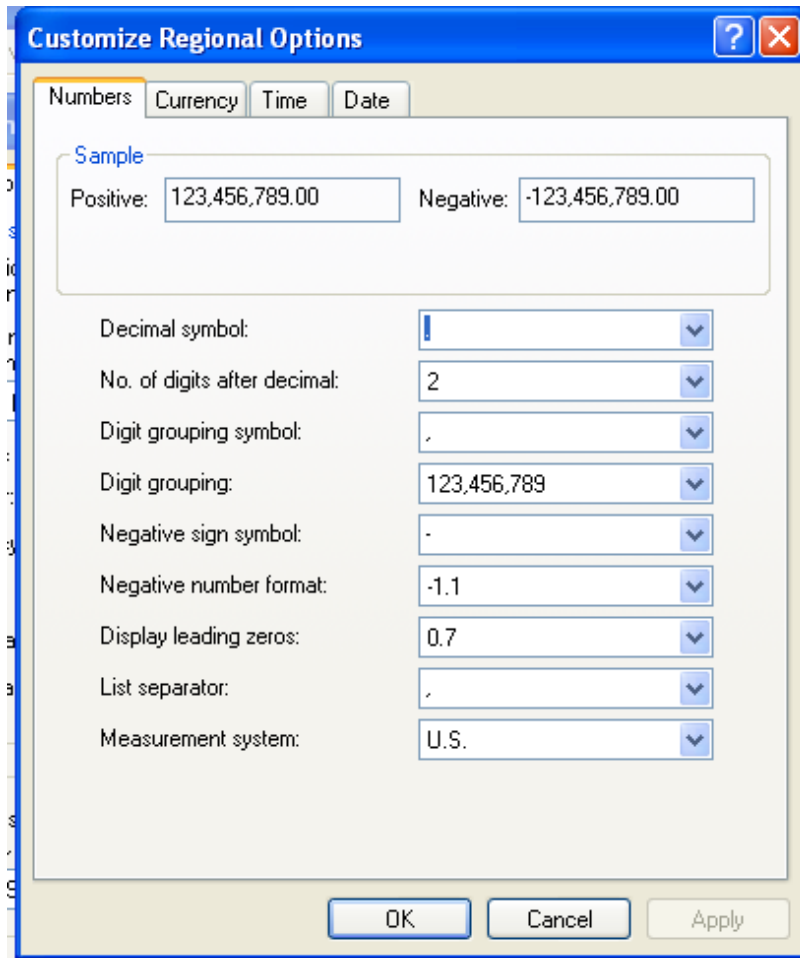


Lệnh Start/ Settings/ Control Panel/ Regional and Language Options

**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 để thay đổi qui 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.



#### 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 gọi 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 Explore
- R\_Click lên biểu tượng My Computer, sau đó chọn Explore ...

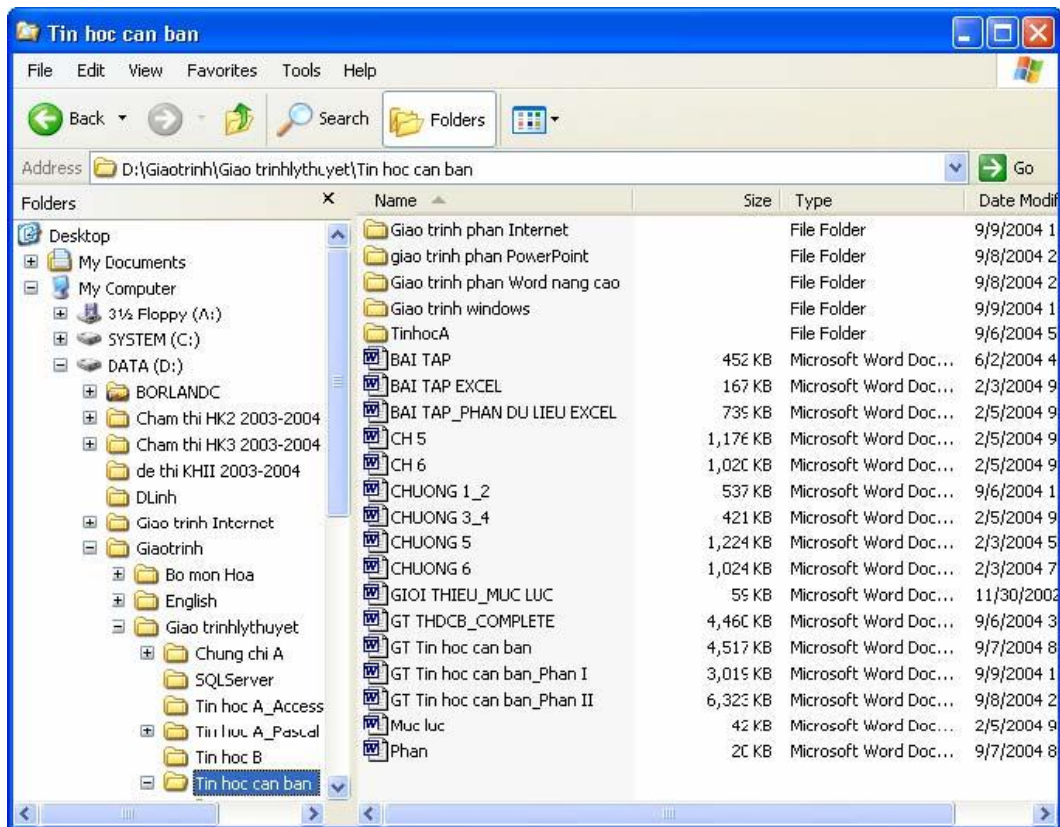
♦ **Cửa sổ làm việc của Windows Explorer:**

- **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 4.1: 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.
 Thumbnails Tiles Icons List ● Details	- 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 thể hiện Details, bạn có thể thực hiện bằng cách luân phiên nhấn chuột lên cột tương ứng (Name, Size, Type, Date Modified).

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

Trong kiểu thể hiện bằng các biểu tượng lớn và biểu tượng nhỏ bạn có thể để Windows sắp xếp tự động bằng lệnh **View/ Arrange Icons By / Auto Arrange**. Tùy 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 2 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. Xóa 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 xóa 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 xóa 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, xóa, 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 (4T lý thuyết)

#### I.3.1. Các hệ thống quản lý thông tin

##### I.3.1.1. Các khái niệm về thông tin và Hệ thống quản lý thông tin

Hai ý nghĩa của khái niệm “Hệ thống thông tin quản lý” (HTTTQL):

Thứ nhất, HTTTQL là một lĩnh vực khoa học quản lý nhằm nghiên cứu việc phát triển, ứng dụng, duy trì các Hệ thống thông tin vì tính trong các lĩnh vực kinh doanh và quản lý khác. Là sự kết hợp giữa nghiên cứu công nghệ và nghiên cứu quản lý.

Thứ hai, HTTTQL là một loại hệ thống thông tin trong phân loại tổng thể:

- Do các nhà quản lý bậc trung sử dụng
- Nhằm hỗ trợ việc giám sát, lập kế hoạch trong toàn doanh nghiệp

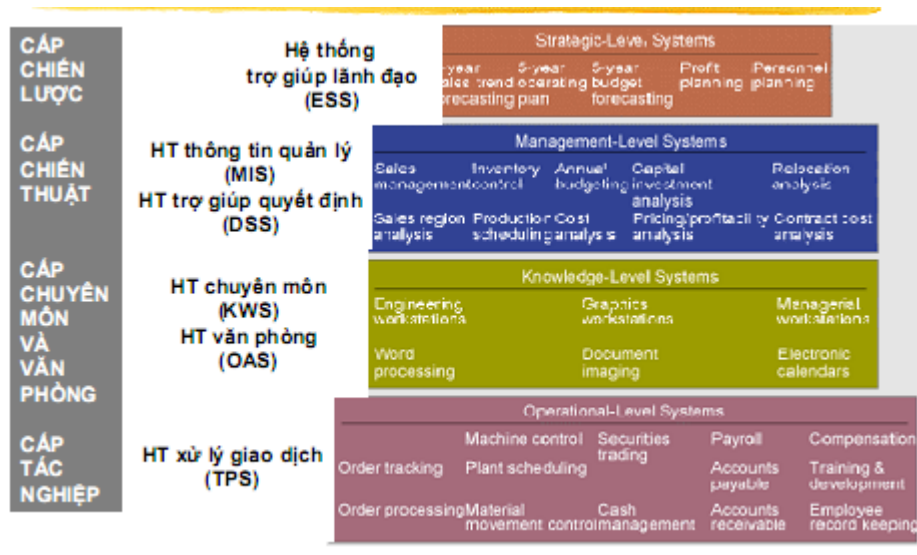
##### I.3.1.2. Phân loại

###### a. Phân loại theo cấp bậc quản lý

Các HTTTQL được phân loại theo 4 cấp bậc quản lý từ thấp đến cao: Cấp tác nghiệp, Cấp chuyên gia và văn phòng, Cấp chiến thuật, Cấp chiến lược. Mỗi cấp là các loại HTTTQL riêng biệt.







### Hệ thống xử lý giao dịch (TPS)

Hệ thống xử lý giao dịch là một hệ thống thông tin giúp thi hành và lưu lại các giao dịch thường ngày cần thiết cho hoạt động sản xuất kinh doanh.

Ví dụ: nhập đơn đặt hàng, đặt phòng khách sạn, bảng lương, lưu hồ sơ nhân viên và vận chuyển vật tư.

Đây là HTTTQL ở Cấp tác nghiệp. Trong đó:

- Thu thập: các giao dịch, sự kiện
- Xử lý: cập nhật, sắp xếp, tổng hợp
- Phân phối: các báo cáo chi tiết, danh sách, tóm tắt
- Người dùng: nhân viên tác nghiệp, quản đốc, trưởng nhóm

### Hệ thống thông tin văn phòng (OAS)

Hệ thống thông tin văn phòng là hệ thống hỗ trợ các nhân viên văn phòng trong các chức năng phối hợp và liên lạc trong văn phòng.

Đây là HTTTQL ở Cấp chuyên môn và văn phòng. Trong đó:

- Thu thập: văn bản, tài liệu, lịch trình
- Xử lý: quản lý văn bản, lập lịch trình, thông tin liên lạc
- Phân phối: văn bản, lịch biểu, thư điện tử
- Người dùng: nhân viên văn thư, tất cả nhân viên

### Hệ thống chuyên môn (KWS)

Hệ thống chuyên môn là hệ thống hỗ trợ lao động có trình độ cao trong công việc chuyên môn hàng ngày của họ.

Đây là HTTTQL ở Cấp chuyên môn và văn phòng. Trong đó:

- Thu thập: các ý tưởng thiết kế, thông số kỹ thuật
- Xử lý: xây dựng mô hình chuyên môn
- Phân phối: bản thiết kế, đồ họa, kế hoạch
- Người dùng: chuyên gia, kỹ thuật viên

### **Hệ thống trợ giúp ra quyết định (DSS)**

Hệ thống trợ giúp ra quyết định hỗ trợ các nhà quản lý ra các quyết định đặc thù, nhanh thay đổi và không có quy trình định trước.

Đây là HTTTQL ở Cấp chiến thuật. Trong đó:

- Thu thập: dữ liệu khối lượng nhỏ
- Xử lý: tương tác
- Phân phối: các báo cáo phân tích, trợ giúp quyết định
- Người dùng: Nhà quản lý bậc trung, chuyên gia

### **Hệ thống thông tin quản lý (MIS)**

Hệ thống thông tin quản lý là hệ thống phục vụ các chức năng lập kế hoạch, giám sát và ra quyết định ở cấp quản lý.

Đây là HTTTQL ở cấp chiến thuật. Trong đó:

- Thu thập: dữ liệu khối lượng lớn, từ Hệ thống xử lý giao dịch
- Xử lý: các quy trình đơn giản
- Phân phối: các báo cáo tổng hợp, tóm tắt
- Người dùng: nhà quản lý bậc trung

### **Hệ thống trợ giúp lãnh đạo (ESS)**

Hệ thống trợ giúp lãnh đạo là môi trường khai thác thông tin tổng thể từ trong và ngoài doanh nghiệp phục vụ việc ra các quyết định đòi sự đánh giá, suy xét và không có quy trình thống nhất.

Đây là HTTTQL ở Cấp chiến lược. Trong đó:

- Thu thập: dữ liệu đã tổng hợp
- Xử lý: tương tác
- Phân phối: các dự báo, phân tích, báo cáo tổng hợp
- Người dùng: lãnh đạo cao cấp

## b. Phân loại theo chức năng nghiệp vụ



Ta có thể chia các hệ thống thông tin quản lý theo các nhóm chức năng nghiệp vụ: Marketing, Sản xuất, tài chính, kế toán, nhân sự.

### Hệ thống quản lý marketing

Là hệ thống trợ giúp các hoạt động của chức năng marketing.

#### VÍ DỤ VỀ HỆ THỐNG QUẢN LÝ MARKETING

Hệ thống	Mô tả	Cấp tổ chức
Xử lý đặt hàng	Nhập liệu, xử lý, và theo dõi đặt hàng	Tác nghiệp
Phân tích thị trường	Phân tích khách hàng và thị trường sử dụng dữ liệu về nhân khẩu, thị trường, thái độ của người tiêu dùng, và các xu hướng	Chuyên môn và văn phòng
Phân tích giá cả	Định giá cho sản phẩm hoặc dịch vụ	Chiến thuật
Dự báo chi tiêu hướng doanh số	Chuẩn bị kế hoạch 5 năm dự báo doanh số	Chiến lược

### Hệ thống quản lý sản xuất

Là hệ thống trợ giúp các hoạt động của chức năng sản xuất.

#### VÍ DỤ VỀ HỆ THỐNG THÔNG TIN QUẢN LÝ SẢN XUẤT

Hệ thống	Mô tả	Cấp tổ chức
Điều khiển máy móc	Điều khiển hoạt động của máy móc và thiết bị	Tác nghiệp
Thiết kế bằng máy tính (CAD)	Thiết kế sản phẩm mới sử dụng máy tính	Chuyên môn và văn phòng
Hoạch định sản xuất	Quyết định số lượng sản phẩm nên sản xuất	Chiến thuật
Định vị khu sản xuất	Quyết định đặt các khu vực sản xuất mới ở đâu	Chiến lược

### Hệ thống quản lý tài chính kế toán

Là hệ thống trợ giúp các hoạt động của chức năng tài chính, kế toán.

### VÍ DỤ VỀ HỆ THỐNG THÔNG TIN QUẢN LÝ TÀI CHÍNH - KẾ TOÁN

Hệ thống	Mô tả	Cấp tổ chức
Quản lý công nợ	Giám sát các khoản công ty cho vay	Tác nghiệp
Phân tích danh mục vốn đầu tư	Thiết kế danh mục vốn đầu tư của công ty	Chuyên môn và văn phòng
Ngân quỹ	Chuẩn bị ngân sách ngắn hạn	Chiến thuật
Hoạch định lợi nhuận	Hoạch định lợi nhuận dài hạn	Chiến lược

### Hệ thống quản lý nhân sự

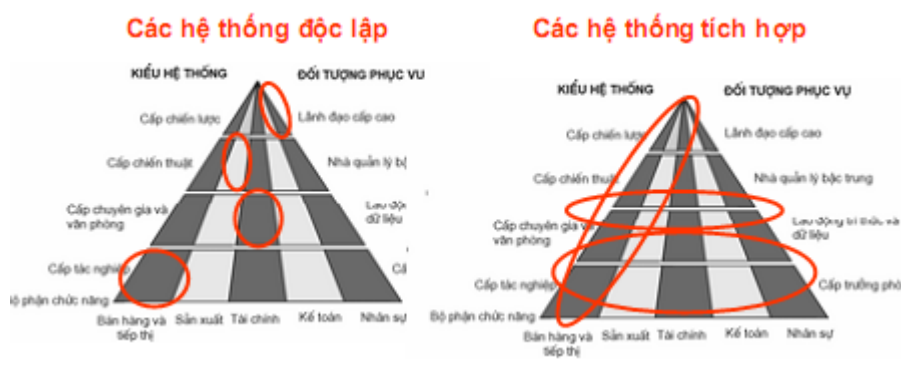
Là hệ thống trợ giúp các hoạt động của chức năng tổ chức, nhân sự.

### VÍ DỤ VỀ HỆ THỐNG THÔNG TIN QUẢN LÝ NHÂN SỰ

Hệ thống	Mô tả	Cấp tổ chức
Đào tạo và phát triển	Giám sát đào tạo, kỹ năng, và đánh giá thành tích	Tác nghiệp
Định hướng sự nghiệp	Thiết kế con đường sự nghiệp cho nhân viên	Chuyên môn và văn phòng
Phân tích chế độ đãi ngộ	Điều khiển phạm vi và phân bổ khoản lương, thưởng và phúc lợi	Chiến thuật
Hoạch định nhân sự	Hoạch định nhu cầu về nhân lực lâu dài của doanh nghiệp	Chiến lược

### c. Phân loại theo quy mô tích hợp

Khái niệm Hệ thống doanh nghiệp tích hợp: Là những hệ thống liên kết xuyên suốt nhiều bộ phận chức năng, cấp bậc tổ chức và đơn vị kinh doanh.



### Hệ thống quản lý nguồn lực (ERP)

Là hệ thống tích hợp và phối hợp hầu hết các quy trình tác nghiệp chủ yếu của doanh nghiệp.



## Hệ thống quản lý chuỗi cung ứng (SCM)

Là hệ thống tích hợp giúp quản lý và liên kết các bộ phận sản xuất, khách hàng và nhà cung cấp.



## Hệ thống quản lý quan hệ khách hàng (CRM)

Là hệ thống tích hợp giúp quản lý và liên kết toàn diện các quan hệ với khách hàng qua nhiều kênh và bộ phận chức năng khác nhau.



## Hệ thống quản lý tri thức (KM)

Là hệ thống tích hợp giúp thu thập, hệ thống hoá, phổ biến, phát triển tri thức trong và ngoài doanh nghiệp.



### I.3.2. Hệ thống tin bảng tính

Ban đầu, máy tính chỉ là một công cụ trợ giúp tính cho con người, nhưng trong quá trình phát triển của mình, máy tính đã trở thành một công cụ đắc lực trong vô số các lĩnh vực khác. Dù vậy, vai trò chính của máy tính vẫn không hề thay đổi. Trong nhiều lĩnh vực, công tác tính toán vẫn chiếm những vị trí hết sức quan trọng, nhất là lĩnh vực kế toán và phân tích thống kê, và đây chính là nơi cần đến sức mạnh tính toán của máy tính. Loại phần mềm trợ giúp tính toán thông dụng nhất hiện nay là các phần mềm bảng tính (viết tắt là PMBT) - spreadsheet software, 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, khoa học gia và nhiều người khác dùng PMBT chỉ vì một lý do: PMBT giúp họ tính toán các số liệu, từ đó cho phép họ xây dựng và làm việc với những tình huống mô phỏng thế giới thực.

Một phần mềm mô phỏng tốt, dù được xây dựng bằng một PMBT hoặc những phần mềm ứng dụng khác đều giúp con người hiểu biết tốt hơn thế giới bên ngoài chiếc máy tính. Mô phỏng bằng máy tính cũng có những giới hạn và hiểm họa của nó. Trong chương này, chúng ta sẽ tìm hiểu cách làm việc với các con số và các chương trình mô phỏng. Đầu tiên hãy bắt đầu với các PMBT.

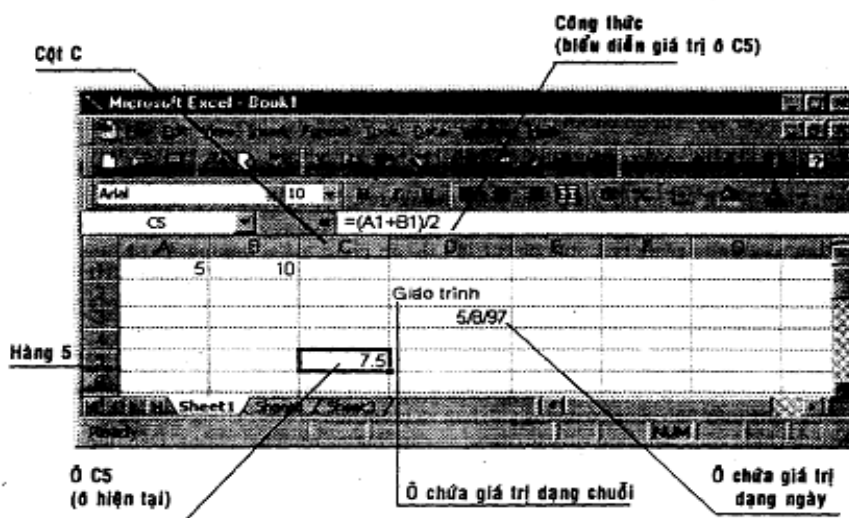
#### **Bảng tính - phần mềm của dự toán**

Phần mềm bảng tính là một trong những phần mềm đã tạo nên những thay đổi lớn lao trong hoạt động kinh doanh của con người. Phần mềm bảng tính cho phép người dùng kiểm soát các con số, thao tác trên chúng theo những phương thức mà nếu làm bằng tay sẽ rất khó hoặc không làm được. Phần mềm bảng tính sẽ giúp chúng ta rút ngắn thời gian thực hiện những công việc tính toán với những phép tính, công thức bị lặp đi lặp lại một cách buồn tẻ như tính điểm trung bình, tính toán ngân quỹ, đầu tư, quản lý, các đề án kinh doanh, mô phỏng khoa học,... Ngoài ra, phần mềm bảng tính còn giúp chúng ta khám phá các mối liên hệ ẩn giữa các con số, tạo cơ sở cho những dự đoán tương lai.

#### ***Bảng tính: những ô lưới linh động***

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 (bắt đầu bằng chữ A). Phần nằm ở

phần giao của một hàng hoặc một cột được gọi là một ô. Mỗi ô trong bản được xác định bằng một địa chỉ duy nhất tạo bởi số của hàng và chữ cái của cột. 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.



Các con số này có thể biểu diễn cho tiền lương, điểm thi, dữ liệu thời tiết, kết quả bầu cử hoặc bất cứ thứ gì có thể định lượng được. Để giúp người đọc dễ dàng hiểu được ý nghĩa các con số, người ta thường xuyên dùng thêm các chuỗi tiêu đề (label) ở đầu các cột và ở bên trái các hàng, như “Lương tháng”, “Kết quả học kì I”, “Tốc độ gió trung bình”, và “Tỷ lệ bầu cử cuối cùng”. Đối với máy tính, các tiêu đề trên chỉ là những chuỗi kí tự vô nghĩa. Một tiêu đề tên là “Điểm tổng kết” hoàn toàn không yêu cầu máy tính tính toán điểm tổng kết mà hiển thị kết quả tại một ô nào đó; nó chỉ là một dạng hướng dẫn người đọc bảng tính – chứ không phải hướng dẫn máy tính!

Để thực hiện một thao tác tính toán gì đó, như tính điểm trung bình chẳng hạn, chúng ta phải đưa vào một công thức tính toán để hướng dẫn máy tính cách tính ra kết quả cuối cùng. Giả sử, bạn có điểm Toán và điểm Lý, bạn đã đưa điểm Toán (giả sử là 9) vào ô B2 và điểm Lý (giả sử là 7) vào ô B3, và bạn muốn biết kết quả điểm trung bình đó sẽ được hiển thị trong ô B5. bạn sẽ đưa vào ô B5 công thức  $= (B2+B3)/2$ . Công thức này yêu cầu máy tính cộng con số ở ô B2 và B3, được bao nhiêu chia cho 2 và hiển thị kết quả cuối cùng trong ô B5. Bạn sẽ không thấy công thức nằm ở ô B5 mà chỉ thấy kết quả của nó (là 8). Với máy tính, ý nghĩa của các con số ở ô B2 và B3 không quan trọng, nó chỉ biết tính toán con số trung bình theo công thức đã cho và hiển thị kết quả. Nếu con số ở ô B2 hoặc B3 bị thay đổi thì con số trung bình ở ô B5 cũng sẽ được tính toán lại (chẳng hạn bạn sửa giá trị ô B2 từ 9 thành 5 thì giá trị ở ô B5 sẽ là 6).

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. Hiếm có phần mềm bảng tính nào thiếu một trong những chức năng sau đây:

- **Tự động lấp các giá trị, tiêu đề và công thức:** Dữ liệu trong các bảng tính có nhiều điểm bị lặp lại như công thức tính điểm trung bình là giống nhau đối với mọi sinh viên trong cùng một lớp, công thức tính lương nhân viên là không đổi qua nhiều tháng,.. 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 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 đến chỉ đến những ô khác. Chẳng hạn để tính điểm trung bình của nhiều sinh viên, giả sử mỗi sinh viên chỉ có 2 điểm Toán và Lý.

Tất cả điểm của mỗi sinh viên được đặt trên một hàng của bảng tính, trong đó điểm Toán được đặt ở cột A, điểm Lý được đặt ở cột B và điểm trung bình cần tính ở cột C. Giả sử ta có 10 sinh viên, nếu không sử dụng tham chiếu tương đối, bạn sẽ phải nhập 10 lần công thức tính điểm trung bình sẽ là  $= (A1 + B1) / 2$ . Điểm trung bình điểm trung bình của sinh viên thứ 2 là  $= (A2 + B2) / 2$ , và cứ như thế. Nếu sử dụng tham chiếu tương đối, bạn chỉ cần nhập công thức này cho một lần cho sinh viên 1 (ở ô C1)  $= (A1 + B1) / 2$ . Trong đó dấu \$ ngụ ý rằng các con số 1 là tham chiếu tương đối theo hàng. Như vậy, khi bạn chép nội dung ô C1 xuống các ô C2, C3, ... C10 thì các công thức sẽ tự động được cập nhật cho phù hợp; cụ thể ở ô C2 công thức sẽ là  $= (A2 + B2) / 2$ , và cứ thế... Tuy nhiên, do tham chiếu tương đối được dùng quá rộng rãi còn tham chiếu tuyệt đối hầu như không dùng nên các phần mềm bảng tính thường mặc định sử dụng tham chiếu tương đối. Như ở ví dụ tính điểm trung bình trên, ta chỉ cần ghi  $= (A1 + B1) / 2$  là đủ.

- **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 tính sẽ tự động tính lại toàn bộ bảng tính (chẳng hạn nhập sai dữ liệu) mà 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:** với những chiếc máy tính nguyên sơ, người ta phải tính toán căn bậc hai của một con số bằng một loạt các bước buồn chán và dễ nhầm lẫn. Với các máy tính tay hiện đại ngày nay, chỉ cần bấm nút “căn số” là chiếc máy đã tự động làm mọi chuyện để có được căn bậc hai của con số ta đưa vào. Các phần mềm bảng tính có sẵn các hàm thư viện – một khái niệm giống như nút “căn số” trên máy tính tay. Một hàm trong một công thức toán học yêu cầu máy tính thực hiện một loạt thao tác tính toán đã định sẵn. Chẳng hạn công thức  $= \text{SQRT}(C5)$  sẽ yêu cầu máy tính tính căn bậc hai của con số nằm ở ô C5. Các phần mềm bảng tính hiện đại đều có một thư viện khổng lồ. Các hàm như Sum, AVERAGE( hoặc AVG), 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. (Ví dụ: IF số giờ làm việc của công nhân  $\geq 40$ , tăng lương thêm 50 cho công nhân đó). Cũng giống như nút căn bậc hai trên các máy tính tay, các hàm thư viện giúp tiết kiệm thời gian giảm bớt nguy cơ sinh lỗi.



Microsoft Excel - Sosanh.xls

VNI-Times 10

L41 5

BẢNG ĐIỂM CÁC MÔN HK 1 - HK 2																	
LỚP 25 K19																	
Hệ số	3	5	5	4	4	4	3	4	3	4	2	2	3	5			
STT	Tên	LSD	A.VĂN	TRIẾT	TCCẤP	ĐC	CS	TKT	KTCT	LKT	TB MÔN	XẾP					
1	BÌNH	5	5	5	5	5	6	5	3	6	2	5	5	5	5	4.745	YẾ
2	CHƯƠNG	4	5	7	5	4	7	8	7	6	4	5	5	6	5	5.588	TE
3	CƯỜNG	5	5	7	4	5	6	7	5	7	7	7	5	5	5	5.667	TE
4	CƯỜNG	3	5	6	4	5	7	4	6	6	4	5	4	6	4	4.98	YẾ
5	ĐÀO	6	7	7	6	7	6	7	8	7	5	5	5	6	6	6.392	TE
6	DŨNG	6	5	6	6	6	8	8	8	6	5	7	7	6	5	6.235	TE
7	DUYÊN	5	5	6	6	6	6	6	5	6	5	3	6	6	5	5.471	TE
8	HẢI	5	5	6	5	5	2	5	5	6	5	5	6	6	5	5.02	TE
9	HIẾU	5	6	6	6	6	7	8	5	6	5	7	6	5	4	5.765	TE

Sheet1

Macro:  
Menu các chức năng của phần mềm bảng tính, cũng giống như trong menu u trong một nhà

hàng, thường chỉ liệt kê những chức năng thường dùng nhất. Trong trường hợp, người dùng thường lặp đi lặp lại một thao tác nhất định nào đó, các phần mềm bảng tính sẽ cho phép người dùng “thu” lại các thao tác đó và định nghĩa nó là một macro. Sau này, khi cần thực hiện lại các thao tác đã ghi nhận trong macro, người dùng chỉ cần gọi lại macro đó thay vì phải lặp lại một loạt các thao tác buồn chán. Ta có thể định nghĩa macro bằng ngôn ngữ macro hoặc dùng bộ thu macro dùng để thu lại mọi thao tác đã tiến hành trên bàn phím và chuột.

Bảng tính mẫu: Ngay sau khi đã dùng macro và hàm việc tạo lập một bảng tính phức tạp từ những đồng dữ liệu hỗn tạp vẫn còn đây gian nan. Nhiều người dùng đã biết tận dụng lợi điểm của mẫu một bảng tính chỉ gồm các tiêu đề và công thức nhưng không chứa dữ liệu. Khi người dùng sẽ tự điền các dữ liệu cần thiết. Nhờ tính năng tự động tính lại, ngay khi có dữ liệu phần mềm bảng tính sẽ tính toán và cho chúng ta kết quả. Thông thường các phần mềm bảng tính sẽ gắn kèm theo một số bảng tính mẫu; và bán riêng một số bản mẫu khác. Khi không có các bảng tính mẫu người dùng sẽ tự tạo ra nó hoặc nhờ một lập trình viên viết. Dù xuất phát từ đâu thì, một bảng tính mẫu được thiết kế tốt sẽ giúp tiết kiệm được rất nhiều thời gian và công sức.

Liên kết: 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 một bảng tính khác. Hầu hết các PMBT cho phép bạn tạo ra những liên kết động giữa các bản 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. Một số chương trình như Lotus 1-2-3 có thể tạo được bảng tính 3 chiều bằng cách xếp lớp(stackinh) và liên kết với nhiều bảng tính 2 chiều.

Chức năng cơ sở dữ liệu: Các phần mềm cơ sở dữ liệu sẽ được đề cập chi tiết ở chương 8. ở đây chúng ta chỉ cần biết là nhiều phần mềm bảng tính có thể thực hiện được những chức năng cơ sở dữ liệu cơ bản: lưu trữ và truy cập thông tin, tìm kiếm sắp xếp, phát sinh báo cáo, trộn thư(mail merge). Với những chức năng này phần mềm bảng tính có thể đáp ứng những yêu cầu cơ sở dữ liệu phức tạp hơn, phần mềm bảng tính vẫn có thể hữu dụng; nhiều chương trình bảng tính hỗ trợ liên kết tự động hai chiều với các phần mềm cơ sở dữ liệu.

**Câu hỏi điều gì sẽ xảy ra nếu...**

Một phần mềm bảng tính là một công cụ linh hoạt, nhưng nó đặc biệt hữu dụng cho câu hỏi “điều gì sẽ xảy ra nếu...”; “Điểm trung bình của tôi sẽ ra sao nếu điểm môn A của tôi là”; “điều gì sẽ xảy ra nếu tôi bán giá sản phẩm 100 thay vì 200”. Các phần mềm mô phỏng cho phép bạn thay đổi các con số và ngay lập tức thấy thay đổi của những con số này, và điều này sẽ làm đơn giản hóa quá trình tìm kiếm lời giải đối với những câu hỏi dạng này.

Một vài phần mềm mô phỏng còn có cả một chương trình chuyên giải phương trình. Thay vì bạn phải thay đổi số liệu nhập cho đến khi các công thức cho ra con số mà bạn mong muốn, chương trình cho phép bạn định nghĩa một chương trình, đưa vào kết quả mong muốn và nó sẽ tính toán số liệu cần thiết để tạo ra kết quả mong muốn này. Bạn thử hình dung việc trả lời câu hỏi “Điểm trung bình của tôi sẽ ra sao nếu điểm môn A, môn B là...” sẽ dễ dàng đến mức nào khi có công thức này. Công cụ này còn rất hữu ích để giải những bài toán tối ưu.

### **Các phần mềm bảng tính tương lai**

Với tất cả những gì phần mềm bảng tính đã mang lại cho người dùng, phải chăng nó đã đi đến điểm kết thúc của cuộc cách mạng. Bước tiếp theo là gì ?

Các ấn bản gần đây của Lotus 1-2-3 còn cho phép hỗ trợ người dùng trong công nghệ multimedia. Microsoft Excel là phần mềm bảng tính đã bắt đầu sử dụng công nghệ trí tuệ nhân tạo để hướng dẫn người dùng trong những tác vụ phức tạp. Để giúp người dùng kiểm tra tính toàn vẹn của dữ liệu và công thức của các bảng tính phức tạp, các phần mềm bảng tính trong tương lai có lẽ sẽ cung cấp các chức năng đại loại bộ kiểm hợp lệ validator hoạt động như các bộ kiểm lỗi chính tả và lỗi ngữ pháp trong các trình xử lý từ.

Nhìn xa hơn nữa có lẽ các phần mềm bảng tính sẽ biến mất cùng với rất nhiều các ứng dụng khác. Các phần mềm bảng tính khổng lồ, đầy các chức năng có thể bằng các phần mềm nhỏ hơn để phụ trợ những dạng ứng dụng không lồ, bao quát. Người dùng sẽ làm việc với văn bản, các con số và những kiểu dữ liệu khác mà không cần phải nghĩ về những trình xử lý, phần mềm bảng tính riêng biệt. Theo quan điểm của người dùng, dữ liệu chiếm vị trí quan trọng hơn công cụ. Nhưng cho đến lúc điều đó xảy ra, các phần mềm bảng tính vẫn tiếp tục là các công cụ đặc lực cho hàng triệu người dùng trên khắp thế giới.

### **Đồ thị trong bảng tính: Từ số thành hình vẽ**

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. Các chương trình lưu đồ riêng biệt cho phép tạo các biểu đồ từ bất kỳ tập hợp các con số nào, dù được chứa trong các bảng tính làm việc hay không. Quá trình tạo một biểu đồ cũng đơn giản như việc điền một thông tin vào một hộp thoại. Và như vậy thì việc tạo lập các biểu đồ từ con số sẽ trở nên dễ dàng, trực quan hơn.

Sự gia tăng dân số nhanh của một quốc gia sẽ dễ hình dung hơn khi ta trình bày nó dưới dạng một đồ thị ngày càng lên cao dần thay vì chỉ quan sát các con số. Hầu hết các PMBT và chương trình vẽ biểu đồ đều cung cấp nhiều kiểu biểu đồ cơ sở và nhiều chọn lựa để xây dựng đồ thị. Các loại đồ thị khác nhau được dùng không chỉ vì lý do mỹ thuật; mỗi loại đồ thị phù hợp với mỗi kiểu diễn đạt thông tin nhất định nào đó. Đồ thị tròn (pie chart) cho thấy sự tương quan của từng bộ phận với toàn thể. Đồ thị đường (line chart) thường dùng để thể hiện các biến động thông qua thời gian. Đồ thị cột (Bar chart) tương tự đồ thị đường, nhưng phù hợp khi dữ liệu được phân ra thành nhiều loại, nhiều kiểu... Một đồ thị được thiết kế tốt có thể chuyển tải những thông tin rất giá trị, và ngược lại một đồ thị đuwocj thiết kế tồi có thể dẫn đến hiểu sai hoặc thiếu sót.

### **Trao đổi kinh nghiệm**

Khi làm việc với bảng tính, chúng ta thường phạm nhiều sai sót nhưng lại ít khi phát hiện ra các sai sót này. Khi tạo một bảng tính, bạn có thể giảm thiểu các sai sót có thể xảy ra bằng những gợi ý sau:

- *Hãy hình dung bảng tính trước khi bạn đưa ra các giá trị và công thức vào.* Hãy nghĩ đến mục tiêu của bạn và thiết kế bảng tính theo mục tiêu này.
- *Kiểm tra nhiều lần mỗi công thức và giá trị.* Các giá trị và công thức là dữ liệu vào của bảng tính, và dữ liệu vào quyết định dữ liệu ra. Các chuyên gia máy tính thường nói ngắn gọn điều này bằng cụm từ GIGO( rác vào thì rác ra – Garbage In Garbage Out).
- *Làm bảng tính trở nên dễ đọc.* Nếu phần mềm bảng tính của bạn cho phép bạn đặt tên cho các ô, hãy cố gắng dùng những tên có nghĩa. Sẽ dễ dàng hơn cho bạn trong công việc tìm và sửa sai khi bạn đọc những công thức như:  $\text{payrate} * 40 + 1.5 * \text{payrate} * (\text{houseworked} - 40)$  thay vì một công thức chứa đầy những ẩn số khó hiểu  $C2 * 40 + 1.5 * C2 * (D2 - 40)$
- *Kiểm tra kết quả bằng những cách khác:* Dùng một chương trình khác, một máy tính tay, viết bút để kiểm tra độ chính xác của kết quả tạo ra bởi bảng tính.
- *Xây dựng các hàm kiểm tra chéo.* So sánh tổng của từng hàng với tổng của cột, có phù hợp không ?
- *Đổi giá trị đầu vào và quan sát kết quả.* Nếu chỉ thay đổi một tí giá trị đầu vào mà kết quả thay đổi quá lớn hoặc ngược lại thì có lẽ đã có gì sai.
- *Hãy tận dụng những hàm có sẵn.* Tại sao bạn phải tự chế tạo ra một chiếc xe khi bạn đang có trong tay cả một chiếc xe hiện đại ?
- *Dùng phần mềm bảng tính như một công cụ hỗ trợ quyết định chứ không phải là công cụ quyết định thay bạn.* Có thể có nhiều sai sót tiềm ẩn trong câu trả lời từ bảng tính và có thể bạn sẽ không phát hiện ra. Đừng đặt trọn niềm tin vào những kết quả đó.

## **BUỔI 5.**

### **I.3.3. Hệ quản trị cơ sở dữ liệu**

#### **I.3.3.1. Cơ sở dữ liệu**

##### **a. Khái niệm về Cơ Sở Dữ Liệu**

Chúng ta sẽ bắt đầu bằng cách nhìn vào điểm căn bản của cơ sở dữ liệu. Giống các trình xử lý từ, phần mềm bảng tính và chương trình đồ họa, các chương trình cơ sở dữ liệu cũng là những công cụ nhằm trợ giúp con người. Nếu một trình xử lý là một máy đánh chữ siêu hiện đại và một phần mềm bảng tính là một sổ cái điện tử thì chúng ta có thể nghĩ về các chương trình cơ sở dữ liệu như là một tủ đựng tài liệu điện tử.

Trong khi các trình xử lý và phần mềm bảng tính thường được dùng để tạo ra các bảng in, các chương trình cơ sở dữ liệu được thiết kế quản lý cơ sở dữ liệu – tập hợp các thông tin được chứa trên các thiết bị lưu trữ của máy tính. Một cơ sở dữ liệu có thể là một bản điện tử của các sổ điện thoại, danh sách các đơn từ, sổ lưu thẻ đọc của thư viện, một danh sách lưu các thẻ kho, một tập hồ sơ điểm của sinh viên, một sổ lưu tên và số điện thoại của những khách hàng, hoặc là một sổ lưu tên các đĩa CD mà bạn có được. Bất kỳ tập hợp thông tin nào cũng có thể được chuyển thành cơ sở dữ liệu.

**Cơ sở dữ liệu có gì tốt ?**

Tại sao người ta dùng máy tính để thực hiện những thao tác quản lý thông tin trong khi những thao tác này có thể thực hiện được với các sổ sách thông thường? Cơ sở dữ liệu trên máy tính có những ưu điểm vượt trội so với người bạn giấy và bút:

+ Cơ sở dữ liệu làm cho việc lưu trữ một lượng thông tin khổng lồ trở nên dễ dàng. Nếu bạn chỉ có 20 đến 30 đĩa CD, bạn vẫn có thể theo dõi chúng bằng một cuốn sổ. Khi bạn có 2000 đến 3000 đĩa CD (bạn là chủ một cửa hàng bán đĩa CD chẳng hạn!) thì cuốn sổ của bạn sẽ dày cộm và rối tung y như đồng đĩa CD của bạn! Nhưng với một cơ sở dữ liệu trên máy tính, toàn bộ thông tin về đĩa CD của bạn chỉ nằm gọn trên một cái đĩa. Thông tin càng khổng lồ, việc sử dụng một cơ sở dữ liệu càng mang lại nhiều lợi ích.

+ Cơ sở dữ liệu giúp nhanh chóng và mềm dẻo trong việc tra cứu thông tin. Bạn thường phải mất khoảng một phút hoặc hơn để tra một số điện thoại trong một sổ ghi điện thoại, cũng việc đó nhưng với cơ sở dữ liệu chỉ tốn chưa đầy 1 giây. Nếu bạn có đến 200 con số để ghi nhớ mỗi tuần, lợi điểm khi sử dụng Cơ sở dữ liệu đã rõ ràng. Lợi điểm đó còn lớn hơn nữa khi việc tìm kiếm của bạn không hợp với cách tổ chức lưu trữ thông tin của cơ quan bạn. Ví dụ, giả sử bạn thấy một số điện thoại trên mảnh giấy nhỏ, bạn muốn tìm tên và địa chỉ của người có số điện thoại đó. Việc tìm kiếm này sẽ làm bạn mất hàng giờ nếu thông tin trên sổ địa chỉ được sắp xếp theo thứ tự alphabet theo tên. Nếu dùng cơ sở dữ liệu trên máy tính hầu như ngay lập tức, bạn sẽ có kết quả. Và như vậy bạn sẽ không lầy gì làm ngạc nhiên khi tổng đài 116 có thể nhanh chóng cho chúng ta biết tên họ và địa chỉ người có số điện thoại cho trước trong khi họ có đến hàng trăm số điện thoại khác nhau.

+ Cơ sở dữ liệu giúp dễ dàng sắp xếp và tổ chức thông tin, khi có quá nhiều tài liệu hoặc sách vở, bạn thường phải sắp xếp chúng lại theo những trật tự nhất định nào đó. Bạn nên sắp xếp chúng theo tên tác giả, tựa đề, ngày xuất bản hay theo chủ đề? Có rất nhiều thứ sẽ làm bạn phải rất phân vân vì nếu sau này phải sắp xếp lại thì sẽ mất rất nhiều thời gian. Với một cơ sở dữ liệu bạn có thể ngay lập tức chuyển từ kiểu sắp xếp này với kiểu sắp xếp khác khi bạn thích; và bạn sẽ không phải chịu bất cứ sự bực mình nào.

+ Cơ sở dữ liệu giúp in và phân phối thông tin theo nhiều cách, giả sử bạn muốn gửi các lá thư đến hàng trăm người bạn để mời họ đến ăn mừng ngày bạn ra trường. Bạn sẽ phải chỉ đường đi đến nơi của bạn cho từng người bạn ở thành phố khác. Quá mất công? Một cơ sở dữ liệu, khi được dùng với một Trình xử lý từ có thể in những dạng thư từ cá nhân và đưa vào đó những thông tin chỉ dẫn cho những người cần nó, và in địa chỉ của từng người lên bì thư chỉ tốn chút thời gian và hầu như không bao giờ sai.

### **Bên trong cơ sở dữ liệu**

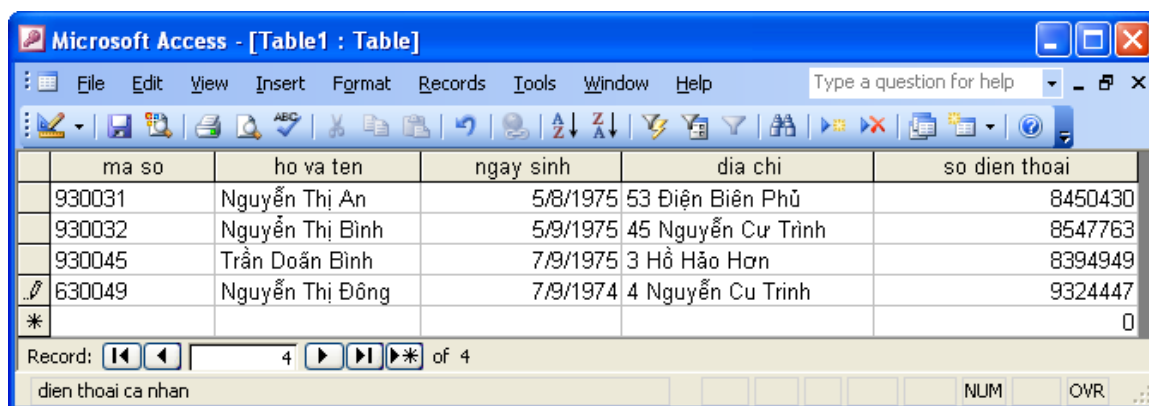
Hiểu một cách nôm na, một cơ sở dữ liệu là một tập hợp thông tin được lưu trữ trên máy tính theo một dạng tổ chức nào đó, và một chương trình cơ sở dữ liệu là một công cụ phần mềm để tổ chức, lưu trữ và truy cập các thông tin đó. Có nhiều loại chương trình thỏa mãn định nghĩa chung này, từ các chương trình tra cứu sổ địa chỉ đến các hệ thống theo dõi hàng tồn kho khổng lồ. Chúng ta sẽ bàn về sự khác nhau giữa các kiểu chương trình cơ sở dữ liệu sau, nhưng bây giờ chúng ta sẽ xem chúng như là một.

Một cơ sở dữ liệu được hình thành từ các file. Một File chứa một tập các thông tin có liên quan; giống như một ngăn kéo đựng hồ sơ thông thường. Nếu cơ sở dữ liệu được dùng để ghi nhận thông tin về các hàng bán cho một công ty, mỗi file có thể sẽ chứa dữ liệu hàng bán có liên quan cho mỗi năm. Đối với các cơ sở dữ liệu địa chỉ, các file riêng có thể sẽ lưu trữ các thông tin cá



nhân và đối tác. Tùy theo người thiết kế cơ sở dữ liệu để quyết định có nên để các thông tin thuộc các dạng khác nhau vào các file trên máy tính khác nhau hay không.

Từ file thường gây nhầm lẫn do tính nhiều nghĩa của nó. Một đĩa cứng có thể chứa chương trình ứng dụng, chương trình hệ thống, chương trình tiện ích và các văn bản, tất cả mọi thứ, nếu đứng từ góc nhìn máy tính đều là các file. Nhưng đối với những người sử dụng cơ sở dữ liệu, từ file thường mang nghĩa là một phần của cơ sở dữ liệu. Trong chương này, từ file mang ý nghĩa là các file dữ liệu tạo bởi một phần mềm cơ sở dữ liệu.



ma so	ho va ten	ngay sinh	dia chi	so dien thoai
930031	Nguyễn Thị An	5/8/1975	53 Điện Biên Phủ	8450430
930032	Nguyễn Thị Bình	5/9/1975	45 Nguyễn Cư Trinh	8547763
930045	Trần Doãn Bình	7/9/1975	3 Hồ Hảo Hớn	8394949
630049	Nguyễn Thị Đông	7/9/1974	4 Nguyễn Cu Trinh	9324447
*				0

Record: 4 of 4

dien thoai ca nhan NUM OVR

Một file cơ sở dữ liệu bao gồm nhiều bản ghi (record). Một bản ghi là thông tin liên quan đến một người, một sản phẩm hoặc một sự kiện nào đó. Trong cơ sở dữ liệu lưu các thẻ thư viện, một record tương đương với một thẻ thư viện. Trong một cơ sở dữ liệu chứa địa chỉ, một record chứa thông tin về một người. Một cơ sở dữ liệu liên quan đến đĩa CD sẽ có một record cho mỗi một đĩa CD.

Mỗi một đoạn thông tin riêng rẽ trong một record được gọi là một trường (field). Một record trong cơ sở dữ liệu thư viện sẽ có những field cho tác giả, tựa đề sách, nhà xuất bản, địa chỉ, ngày tháng và mã số sách. Cơ sở dữ liệu về CD có thể chia record thành các field như tựa đề CD, tác giả....

Kiểu thông tin một field được xác định bằng kiểu field(field type). Ví dụ: field tên tác giả trong cơ sở dữ liệu thư viện có kiểu chữ (text field) để có thể chứa được các chữ cái tên của tác giả. Field cho biết số bản sách hiện có thể sẽ được định nghĩa là field kiểu số để nó chỉ có thể chứa các con số. Field cho biết ngày nhập sách vào thư viện sẽ là field kiểu ngày. Bên cạnh những kiểu field cơ bản này, nhiều chương trình cơ sở dữ liệu cho phép các field chứa các hình ảnh, âm thanh, và thậm chí cả những đoạn video. Các field tính toán (computed) chứa các công thức tương tự như công thức trong các phần mềm bảng tính; chúng hiển thị giá trị được tính toán từ các giá trị của những field khác. Ví dụ, một field tính toán có tên là DTB sẽ chứa công thức tính điểm trung bình với các điểm nằm các field khác.

Hầu hết các phần mềm cơ sở dữ liệu cho phép bạn xem dữ liệu theo nhiều cách khác nhau, thường là kiểu xem đọc(form view) chỉ cho thấy mỗi record một lần và kiểu xem ngang (list view) hiển thị nhiều record trong một danh sách trông giống như một bảng tính. Trong bất cứ kiểu xem nào, các field cũng có thể được sắp xếp lại và không làm thay đổi dữ liệu bên dưới.

Một khi cấu trúc của cơ sở dữ liệu đã được định nghĩa, việc đưa thông tin vào trở nên dễ dàng; chỉ là vấn đề gõ phím mà thôi. Và thậm chí việc gõ phím cũng sẽ không cần thiết nếu dữ liệu đã được lưu đâu đó trên máy tính. Hầu hết các chương trình cơ sở dữ liệu có thể dễ dàng hiểu và

chấp nhận dữ liệu dưới dạng các file văn bản tạo bởi các trình xử lý từ, phần mềm bảng tính hoặc những phần mềm khác. Khi thông tin bị thay đổi hoặc khi phát hiện lỗi, các record có thể được điều chỉnh, thêm hoặc xóa.

Phần thử thách của việc sử dụng một Cơ sở dữ liệu là truy cập thông tin theo cách nào sao cho ít tốn thời gian và phù hợp. Các thông tin sẽ mất nhiều giá trị nếu ta không truy cập đến nó được. Một cách để tìm thông tin là dò (browse) từng record của file Cơ sở dữ liệu giống như dò trên giấy tờ. Hầu hết các chương trình Cơ sở dữ liệu cung cấp được các lệnh bàn phím, các nút trên màn hình và những công cụ khác để qua lại nhanh chóng giữa các record. Những kiểu làm việc này chẳng mang lại lợi lộc gì so với dung giấy; thật là kinh khủng khi gặp phải những file quá lớn.

Nhưng may thay, đa số các chương trình Cơ sở dữ liệu thường có nhiều lệnh và chức năng làm cho việc tìm kiếm thông tin của bạn muốn trở nên dễ dàng hơn.

### **Truy vấn cơ sở dữ liệu (query)**

Một phương pháp thay vì dò từng record là hỏi Cơ sở dữ liệu về những thông tin nào đó. Trong các thuật ngữ liên quan đến Cơ sở dữ liệu, một yêu cầu về thông tin được gọi là query. Một query có thể là một yêu cầu tìm kiếm đơn giản một record nào đó (chẳng hạn như tìm một sinh viên có mã số ABCD01) hoặc là tìm tất cả record thỏa mãn một tập điều kiện nào đó (chẳng hạn, tìm tất cả sinh viên có điểm trung bình dưới 5 và còn nợ trên 2 môn học). Một khi đã chọn xong một nhóm các record, bạn có thể dò tìm trên nhóm record này, tạo ra 1 bảng báo cáo; nói chung là bất cứ thao tác nào bạn có thể thực hiện trên một file thì bạn cũng có thể thực hiện trên 1 nhóm các record có được từ 1 query.

### **Sắp xếp dữ liệu**

Thỉnh thoảng cũng phải sắp xếp lại các record để sử dụng dữ liệu hữu hiệu hơn. Giả sử danh sách sinh viên nên sắp xếp theo alphabet để dễ tra cứu theo tên, nhưng để phân loại kết quả học tập lại cần phải sắp xếp danh sách này theo điểm trung bình. Lệnh sort cho phép bạn sắp xếp các record theo trình tự alphabet hoặc theo giá trị số tùy theo kiểu dữ liệu của file được sắp xếp.

### **In các báo cáo, tiêu đề và các dạng thư từ**

Để hiển thị thông tin lên màn hình, các chương trình Cơ sở dữ liệu có thể tạo ra rất nhiều báo cáo – một danh sách các record được chọn lựa và sắp xếp theo một trình tự dễ đọc. Đa số các bảng báo cáo thương mại đều sắp xếp theo dữ liệu thành các bác với càng hàng là các record và các cột cho các file được chọn; ngoài ra chúng thường có các hàng tổng cộng chứa tổng và trung bình giá trị của một nhóm record.

Các chương trình dữ liệu còn có thể được dùng để tạo ra các nhãn thư từ và các dạng thư từ theo ý muốn. Nhiều chương trình Cơ sở dữ liệu không thực sự in ra các lá thư; chúng chỉ đơn giản cung cấp các dữ liệu cần thiết cho các Trình xử lý từ có khả năng trộn thư (mail merge) để chúng tiến hành công việc in thư.

### **Các query phức tạp**

Các query có thể đơn giản hay phức tạp, nhưng theo cách nào đi nữa kết quả vẫn phải chính xác và không được mờ nghĩa. Với các Cơ sở dữ liệu phù hợp, các query có được xây dựng để:

- Trong một Cơ sở dữ liệu về bệnh nhân trong một bệnh viện, xác định tên và vị trí của tất cả bệnh nhân trên tầng 5 và 6 của bệnh viện.

- Trong Cơ sở dữ liệu của lịch trình chuyến bay, xác định lịch trình rẻ tiền nhất để bay từ Boston đến San Francisco vào trưa thứ 3.

- Trong một Cơ sở dữ liệu về chính trị, xác định tất cả cử tri đã đóng góp tài chính cho cuộc vận động bầu cử và tất cả những cử tri đã bày tỏ ý kiến về bộ luật lao động mới của chính phủ.

Những điều này đều là những mục tiêu đúng đắn của query, nhưng chưa được diễn tả theo một ngôn ngữ chung mà tất cả chương trình Cơ sở dữ liệu có thể hiểu được. Phương pháp chính xác để thực hiện một query phụ thuộc vào giao diện người dùng của Cơ sở dữ liệu. Hầu hết chương trình cho phép người dùng chỉ ra các luật tìm kiếm bằng cách điền thông tin vào các hộp thoại. Một vài phần mềm yêu cầu người dùng gõ các yêu cầu bằng cách dùng một ngôn ngữ đặc biệt, ngôn ngữ của query, chính xác hơn tiếng Anh. Ví dụ để xác định tất cả record của những người nam giới trong độ tuổi từ 18 đến 35, bạn sẽ gõ

***Select \* from Population Where Sex = M and Age > 18 and Age < 35***

(Chọn tất cả công dân trong file <Danh sách công dân> sao cho giới tính là nam và tuổi từ 18 đến 35)

Nhiều chương trình Cơ sở dữ liệu có sử dụng ngôn ngữ lập trình để các query đưa vào chương trình và được thực hiện tự động khi chương trình được thi hành. Dù chi tiết của quá trình này là khác nhau, những điểm cơ bản là thống nhất với mọi chương trình.

Hầu hết các chương trình quản trị Cơ sở dữ liệu đều hỗ trợ SQL (một ngôn ngữ chuẩn để diễn đạt các câu query phức tạp, ví dụ ở trên dùng SQL). Vì SQL làm việc trên nhiều hệ quản trị Cơ sở dữ liệu khác nhau, các lập trình viên và những người dùng tinh tế không cần phải học lại khi họ làm việc với các hệ phân cứng và phần mềm khác nhau. Người dùng thường được tách biệt khỏi sự phức tạp của ngôn ngữ query phức tạp bằng các giao diện người dùng bằng đồ họa cho phép thực hiện các query bằng trỏ và click.

### **Các chương trình Cơ sở dữ liệu chuyên biệt**

Các phần mềm Cơ sở dữ liệu chuyên biệt được lập trình sẵn cho một kiểu dữ liệu xác định và một cách truy cập xác định. Các Cơ sở dữ liệu về CD-ROM được sử dụng trong các thư viện là một ví dụ của các chương trình Cơ sở dữ liệu chuyên biệt. Người dùng của các Cơ sở dữ liệu chuyên biệt không cần phải định nghĩa các cấu trúc file hoặc định nghĩa các màn hình giao diện vì những chi tiết này đã được người thiết kế phần mềm làm sẵn. Trên thực tế, nhiều chương trình Cơ sở dữ liệu chuyên biệt không được bán dưới hình thức một Cơ sở dữ liệu; chúng được đặt tên chính xác hơn, phản ánh mục đích sử dụng của chúng .

Ví dụ:

- a. Danh bạ điện thoại điện tử có thể chứa hàng triệu tên và số điện thoại trên một đĩa CD-ROM. Dùng một danh bạ điện thoại điện tử của Mỹ quốc ngay cả khi bạn không biết họ ở đâu. Bạn có thể tra tên riêng của một người nếu bạn có số điện thoại hoặc địa chỉ. Bạn có thể in một bảng báo cáo cho biết tất cả các nhà sĩ trong bất kỳ thành phố nào. Sau đó dùng thêm một Cơ sở dữ liệu nữa, một bảng đồ điện tử, bạn sẽ xác định được chính xác các vị trí các vị trí bạn quan tâm.

- b. **Hệ thống tin địa lý ( *Geographical Information System\_GIS* )** có nhiều khả năng hơn hẳn một chương trình bản đồ thông thường. Một hệ GIS cho phép một doanh nghiệp phối hợp bản đồ địa lý với các bảng dữ kiện ( chẳng hạn danh sách khách hàng ), với các thông tin nhân khẩu từ cục dân số và những nguồn khác với nhau. Sự phối hợp đúng cách có thể đóng góp đáng kể cho chiến lược kinh doanh. Ví dụ những người môi giới chứng khoán có thể xác định chính xác nơi đặt các văn phòng chi nhánh dựa trên thu nhập trung bình và những dữ liệu xung quanh khác; một công ty truyền hình cáp có thể xác định những khách hàng quan trọng sống gần những đường dây có sẵn. Vì chúng có thể hiển thị dữ liệu địa lý và nhân khẩu trên bản đồ, chúng cho phép người dùng thấy được những mối liên hệ không dùng được ở dạng dùng bảng.

Nhiều dạng chương trình Cơ sở dữ liệu được bán dưới dạng hệ quản trị thông tin cá nhân (Personal Information Manager viết tắt là PIM ) hoặc hệ tổ chức điện tử ( Electronic Organizer). Một hệ quản lý thông tin cá nhân có thể tự động hóa một số chức năng sau :

+ *Sổ địa chỉ / Danh bạ điện thoại*. Các sổ địa chỉ cung cấp các chọn lựa để nhanh chóng hiển thị những record xác định và in tất cả các địa chỉ gửi thư và các báo cáo. Một vài hệ có cả chức năng quay số điện thoại tự động và các trường để ghi nhận những ghi chú về số điện thoại.

+ *Lịch hẹn*. Một PIM tiêu biểu thường cho phép bạn lên danh sách các cuộc hẹn, các sự kiện và hiển thị hoặc in chúng theo nhiều kiểu khác nhau, từ một ngày một lần hoặc tổng quát cả tháng. Nhiều hệ còn có chức năng báo động để nhắc nhở những sự kiện sắp đến.

+ *Danh sách công việc phải làm*. Hầu hết PIM cho phép người dùng lên danh sách và sắp xếp danh sách các việc cần làm và kết quả thực hiện của những công việc đã tiến hành.

+ *Chú thích chung*. Vài hệ PIM còn cho phép người dùng ghi nhật ký, ghi chú cá nhân và những thông tin phân loại khác .

PIM được sử dụng rộng rãi ở những người có lịch làm việc dày đặc và có nhiều mối liên hệ. PIM dễ hiểu và dễ sử dụng hơn các chương trình Cơ sở dữ liệu công dụng chung, và chúng có tốc độ nhanh và sự mềm dẻo hơn hẳn cây bút và giấy tờ. Với những người đang trên đường công tác, PIM làm việc rất hữu hiệu trên các máy tính xách tay. Nhưng PIM không phải dành cho mọi người; nhiều người thấy rằng các chức năng chuẩn được xây dựng trong các phần mềm quản trị thông tin cá nhân không thỏa hoặc không hợp với nhu cầu của họ. Dù sao đi nữa, quản trị thông tin cá nhân cuối cùng cũng là riêng tư .

## ***b. Hệ quản trị cơ sở dữ liệu***

### **Từ quản trị file sang hệ quản trị Cơ sở dữ liệu**

Về mặt kỹ thuật, các chương trình quản lý khách hàng và thông tin cá nhân không thật sự là một trình quản trị cơ sở dữ liệu: chúng chỉ là quản lý file. Một trình quản lý file cho phép người dùng làm việc chỉ trên một file một lúc. Một hệ quản trị cơ sở dữ liệu thực sự là một chương trình hoặc hệ các chương trình có thể thao tác trên dữ liệu trong một tập nhiều file, có những tham chiếu giữa các file nếu cần thiết. Hệ quản trị cơ sở dữ liệu có thể được dùng theo kiểu tương tác hoặc có thể được điều khiển trực tiếp bằng chương trình khác. Một trình quản lý file là đủ để theo dõi danh sách khách hàng và những ứng dụng quản lý dữ liệu thông thường khác. Nhưng đối với các công việc phức tạp, khổng lồ, không gì có thể thay thế được một hệ quản trị cơ sở dữ liệu thực sự.



Ví dụ, xét bài toán quản lý thông tin sinh viên. Bạn cũng sẽ dễ thấy cách dùng cơ sở dữ liệu để chứa các thông tin: một file chứa một record cho một sinh viên, với các field cho tên, mã số sinh viên, địa chỉ, số điện thoại và nhiều thứ khác. Thông thường, một file sẽ không thể chứa hết mọi thông tin của sinh viên.

Hầu hết các trường chọn cách dùng vài file để chứa thông tin sinh viên: một cho các dữ liệu về tài chính, một theo dõi việc đăng ký môn học, điểm và nhiều thứ khác. Mỗi file chứa một record cho một sinh viên. Bên cạnh đó, nhà trường còn phải theo dõi danh sách các sinh viên đăng ký vào các lớp với một record cho một lớp và các field cho các thông tin trên mỗi sinh viên đăng ký vào lớp. 3 file này được sắp xếp như sau:

**Thông tin sinh viên**

Mã số sinh viên

Tên

Địa chỉ

Chuyên ngành

...

**Học kỳ 1**

Trung bình

Số tín chỉ đã đạt

...

**Học kỳ 2**

Trung bình

Số tín chỉ đã đạt

**Thông tin tài chính**

Mã số sinh viên

Tên

Địa chỉ

Chuyên ngành

...

Học phí

Phí ký túc xá

Học bổng

**Danh sách lớp**

Mã môn học

Phòng học

GV hướng dẫn

Tổng số sinh viên

...

**Sinh viên 1**

MSSV

Họ tên

Ngành

...

**Sinh viên 2**

MSSV

Họ tên

Ngành

Trong cơ sở dữ liệu ở trên, mỗi file trong 3 file đều chứa những thông tin cơ bản về mỗi sinh viên (họ tên, địa chỉ). Kiểu dư thừa dữ liệu này không chỉ làm tốn không gian lưu trữ mà còn làm khó khăn trong việc đảm bảo các thông tin sinh viên là chính xác và cập nhật. Nếu một sinh viên được chuyển sang lớp khác, 3 file phải được cập nhật lại để phản ánh sự thay đổi này. Càng thay đổi nhiều, khả năng sai sót càng lớn.

Với một hệ quản trị cơ sở dữ liệu, không cần phải chứa tất cả thông tin này trong mỗi file. Cơ sở dữ liệu có thể dùng một file chứa các thông tin cơ sở về sinh viên – những thông tin duy nhất cho mỗi sinh viên. Vì các thông tin này được chứa trên một file riêng biệt nên nó không cần đưa vào các file theo dõi tài chính, file theo dõi điểm, và lớp hay bất kỳ một file nào khác. Các mã số sinh viên được đưa vào trong các file hoạt động như là một key file; dùng để lấy ra các thông tin sinh viên liên quan trong file sinh viên khi cần thiết ở một nơi nào khác. Field mã số sinh viên được dùng chung trong tất cả các file dùng dữ liệu từ file này. Nếu sinh viên đổi nhà, việc thay đổi địa chỉ chỉ cần ghi nhận ở một chỗ. Các cơ sở dữ liệu được tổ chức theo kiểu này được gọi là Cơ sở dữ liệu quan hệ.

#### **Theo dõi học tập**

Mã số sinh viên

...

#### **Học kỳ 1**

Điểm TB

...

#### **Học kỳ 2**

#### **Thông tin sinh viên**

Mã số sinh viên

Tên

Địa chỉ

Chuyên ngành

...

#### **Học kỳ 1**

Trung bình

Số tín chỉ đã đạt

...

#### **Học kỳ 2**

Trung bình

Số tín chỉ đã đạt

#### **Danh sách lớp**

Mã môn học

Phòng học

GV hướng dẫn

Tổng số sinh viên

...

#### **Sinh viên 1**

MSSV

Họ tên

Ngành

...

#### **Sinh viên 2**

MSSV

Họ tên

Ngành

## **Điều gì làm cho một cơ sở dữ liệu có tính quan hệ?**

Đối với hầu hết người dùng, một chương trình cơ sở dữ liệu quan hệ là chương trình cho phép các file có liên hệ với nhau để các thay đổi trên một file sẽ được tự động phản ánh trên những file khác. Đối với các lập trình viên, từ cơ sở dữ liệu quan hệ có một định nghĩa kỹ thuật liên quan đến cấu trúc bên dưới của dữ liệu và các luật xác định cách thao tác dữ liệu.

Cấu trúc của một cơ sở dữ liệu quan hệ dựa trên mô hình quan hệ (relational model).

Một mô hình toán học phối hợp dữ liệu giữa các mảng. Những hệ quản trị cơ sở dữ liệu loại khác sẽ dựa trên những mô hình lý thuyết khác với những lợi điểm và yếu điểm kỹ thuật riêng. Nhưng những hệ quản trị cơ sở dữ liệu chính yếu được dùng ngày nay, kể cả những hệ quản trị cơ sở dữ liệu dựa trên máy PC cũng dùng mô hình quan hệ. Vì vậy, từ góc nhìn của một người dùng trung bình, sự phân biệt giữa định nghĩa kỹ thuật và phổ thông của từ quan hệ có vẻ hoàn toàn lý thuyết.

## **Tính nhiều mặt của cơ sở dữ liệu**

Các cơ sở dữ liệu không lồ có thể chứa hàng trăm file với vô số mối quan hệ bên trong. Kiểu mô cung thông tin này có thể rất khó hiểu đối với người dùng nếu họ bị buộc phải làm việc trực tiếp với nó. May thay, một hệ quản trị cơ sở dữ liệu có thể “che” người dùng khỏi những công việc phức tạp bên trong của hệ thống, chỉ cung cấp thông tin và các lệnh cần thiết để hoàn thành công việc. Trên thực tế, một cơ sở dữ liệu tốt sẽ có nhiều “bộ mặt” khác nhau đối với những người dùng khác nhau.

Nhân viên bán lẻ không cần (và nhiều lúc không được quyền) truy cập đến mọi thông tin trong cơ sở dữ liệu được lưu trữ của công ty; họ chỉ cần ghi nhận các giao dịch mua bán từ các máy tính trạm đặt ở điểm bán lẻ. Cơ sở dữ liệu được thiết kế cho những người bán lẻ thông thường chỉ cần một giao diện đơn giản, cung cấp cho người bán các thông tin cần thiết giúp họ thực hiện việc bán hàng. Nhà quản lý kế toán, những chuyên gia truy cập dữ liệu và các khách hàng sẽ nhìn cơ sở dữ liệu dưới những góc nhìn khác nhau vì họ cần làm việc với những thông tin khác nhau.

## **Xu hướng client/server**

Các tiến bộ trong hai thập kỷ vừa qua đã thay đổi cách hầu hết các tổ chức làm việc với dữ liệu. Các chương trình quản lý file đầu tiên chỉ có thể thực hiện xử lý theo lô (batch processing), yêu cầu người dùng tích trữ các giao dịch và đưa chúng vào máy tính một lần. Những hệ thống xử lý theo lô này không thể cũng cấp các câu trả lời ngay lập tức như chúng ta vẫn thấy ngày nay. Các câu hỏi đại loại như “số dư của tài khoản của tôi là bao nhiêu?” hoặc là “có chuyến bay nào đến Denver thứ 3 tới không?” sẽ được trả lời bằng cách “chờ đã, những thông tin này sẽ được cập nhật vào tối nay, ngày mai chúng tôi sẽ cho bạn biết!”

Các ổ đĩa, bộ nhớ và những phần mềm tinh tế và rẻ tiền ngày nay đã cho xử lý tương tác (interactive processing) thay thế xử lý theo lô đối với hầu hết ứng dụng. Người dùng bây giờ có thể tương tác với dữ liệu thông qua các trạm làm việc, xem và thay đổi giá trị ngay lập tức theo thời gian thực (nghĩa là sự thay đổi sẽ được cập nhật và những người khác sẽ lập tức thấy sự thay đổi này). Xử lý theo lô vẫn được dùng để in các hóa đơn và báo cáo định kỳ và để tạo các

văn bản sao phòng hồ các file dữ liệu – các công việc làm cho việc phải làm nhiều giao dịch cùng lúc trở nên có nghĩa. Nhưng đối với các ứng dụng yêu cầu phản ứng nhanh chẳng hạn như đặt chỗ máy bay, giao dịch ngân hàng và những kiểu hệ thống tương tác nhiều người dùng thì hệ Cơ sở quản lý dữ liệu lại chiếm ưu thế.

Trong những ngày trước khi PC xuất hiện, hầu hết các Cơ sở dữ liệu đều cư trú trên các máy tính mainframe. Nhưng kiểu tập trung dữ liệu khó truy cập (vì chỉ những chuyên gia mới được phép thao tác) trên các hệ thống mainframe trước kia đã không còn được sử dụng. Ngày nay các doanh nghiệp đã nhanh chóng làm quen với các giải pháp client/server: chương trình client trong máy tính để bàn gửi các yêu cầu thông qua một mạng máy tính đến dữ liệu trên các máy tính server, có thể là các máy tính mainframe, máy mini hoặc một máy tính để bàn khác; máy server sẽ tiến hành tìm kiếm kết quả và sẽ gửi kết quả về lại máy client. Hệ thống client/server cho phép người dùng tận dụng được lợi điểm giao diện đơn giản trên các máy PC và sức mạnh của các máy tính cỡ lớn để làm việc với những Cơ sở dữ liệu khổng lồ. Nhiều công ty đã lưu trữ tất cả bảng sao dữ liệu của công ty trong một kho dữ liệu (data warehouse); các công ty khác dùng Cơ sở dữ liệu phân tán, dùng để chia nhỏ dữ liệu ra và để chứa nó rải rác ở nhiều máy tính trong mạng. Từ góc nhìn của người dùng, sự khác biệt giữa các giải pháp này không rõ ràng. Cho dù dữ liệu được lưu trữ ở đâu và như thế nào, mục tiêu cuối cùng vẫn là cho phép truy cập nhanh những thông tin quan trọng.

### **Cơ sở dữ liệu của ngày mai?**

Thay đổi lớn nhất trong công nghệ Cơ sở dữ liệu trong vài năm sắp tới có lẽ là sẽ xảy ra bên dưới con mắt của người dùng. Ví dụ nhiều chuyên gia máy tính tin rằng mô hình dữ liệu quan hệ sẽ bị đánh bật ra khỏi cuộc chiến bởi mô hình dữ liệu hướng đối tượng, và hầu hết các Cơ sở dữ liệu tương lai sẽ là Cơ sở dữ liệu hướng đối tượng thay vì quan hệ. Thay vì phải chứa các thông tin trong các bảng và record có liên quan với nhau, Cơ sở dữ liệu hướng đối tượng sẽ chứa các thông tin trong các đối tượng có chứa cả mã lệnh cùng với dữ liệu. Cơ sở dữ liệu quan hệ thường được dùng với ngôn ngữ lập trình hướng đối tượng, chúng ta sẽ đề cập ở chương kế. Các chuyên gia cho rằng công nghệ hướng đối tượng sẽ làm cho việc xây dựng và thao tác trên các Cơ sở dữ liệu trở nên đơn giản và tốn ít thời gian hơn. Người dùng sẽ cảm thấy Cơ sở dữ liệu uyển chuyển hơn và phản ứng tốt hơn khi công nghệ đối tượng được sử dụng rộng rãi, ngay khi họ không để ý đến những lý do kỹ thuật bên dưới của những cải tiến này.

### **I.3.4. Các hệ thống thông minh**

## Bài tập về Tin học căn bản (2 tiết BT)

...

## BUỔI 6.

### PHẦN II. GIẢI QUYẾT BÀI TOÁN

(8 tiết Lý thuyết, 2 tiết Bài tập)

#### II.1. Thuật toán(4 tiết LT)

##### II.1.1. Định nghĩa thuật toán

Thuật toán là một khái niệm cơ sở của Toán học và Tin học. Nói một cách nôm na, thuật toán là một tập các lệnh hay chỉ thị 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 và có thể thi hành được để hướng dẫn thực hiện hành động nhằm đạt được mục tiêu đề ra. 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 chỉ đạo nó 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 là một dãy gồm các bước không mập mờ và có thể thực thi được. Hơn nữa quá trình hành động theo thuật toán phải dừng và cho kết quả như mong muốn.

Cụm từ “không mập mờ” trong định nghĩa thuật toán ở trên có nghĩa là: tại mỗi bước, hành động kế tiếp phải được xác định một cách duy nhất theo chỉ thị hành động và theo dữ liệu thích hợp ở thời điểm đó. Điều này không có nghĩa là mỗi chỉ thị hành động trong thuật toán phải xác định được hành động trong mỗi bước khi xét riêng. Ví dụ như chỉ thị hành động theo kiểu” Nếu... thì...” không đủ thông tin để xác định hành động cần phải thực hiện.

Theo định nghĩa trên đây thì dãy các bước sau đây:

1. Tạo danh sách tất cả các số nguyên dương
2. Sắp xếp danh sách theo thứ tự giảm dần
3. Lấy ra số nguyên đầu tiên từ danh sách đã được sắp xếp
4. Dừng

không phải là một thuật toán vì các bước 1 và 2 không thể thực hiện được.

Ví dụ về thuật toán:

Thuật toán tìm phân tử lớn nhất trong một dãy hữu hạn các số nguyên.

Bài toán tìm phân tử lớn nhất trong một dãy hữu hạn tương đối tầm thường. Tuy nhiên đây là một trong các ví dụ khá tốt để minh họa cho khái niệm về thuật toán. Có nhiều vấn đề mà trong đó 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...

Chúng ta có nhiều cách để giải bài toán này. Một trong các phương pháp để tìm phân 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.

Các đặc trưng của thuật toán:

Theo định nghĩa của thuật toán ra có thể thấy rằng: khi mô tả một thuật toán chúng ta cần chú ý đến các tính chất đặc trưng sau đây của thuật toán:

- Nhập (input): các thuật toán thường có các giá trị nhập (input values) từ một tập hợp nhất định nào đó.
- Xuất (output): Từ mỗi tập hợp các giá trị được nhập một thuật toán thường tạo ra những giá trị xuất (output values) thuộc một tập hợp nhất định nào đó thể hiện lời giải cho bài toán (hay vấn đề)
- Tính xác định (definiteness): Các bước trong thuật toán phải chính xác rõ ràng.
- Tính hữu hạn (finiteness): thuật giải phải cho ra lời giải (hay kết quả) sau một số hữu hạn các bước.
- Tính hiệu quả: Tính hiệu quả 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).
- Tính tổng quát: Thuật toán phải áp dụng được cho tất cả các bài toán có dạng như mong muốn chứ không phải chỉ áp dụng được cho một số trường hợp riêng lẻ nào đó.

### II.1.2. Biểu diễn thuật toán

Để có thể truyền đạt một thuật toán cho người khác hay chuyển thuật toán cho máy tính (thành chương trình điều khiển máy tính), ta phải tìm cách biểu diễn thuật toán. Một cách tiếp cận tự nhiên là sử dụng các ngôn ngữ để biểu diễn thuật toán.

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

- 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 sử dụng một loại ngôn ngữ tự nhiên để liệt kê các bước của thuật toán. 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 khó hiểu.

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

- Đầu vào: số số nguyên dương  $N$  và  $N$  số nguyên  $a_1, a_2, \dots, a_N$
- Đầu ra: số nguyên lớn nhất của dãy  $a_k$ ,  $k$  trong khoảng  $[1 \dots N]$

*Ý tưởng*

- Khởi tạo giá trị  $Max = a_1$
- Lần lượt so sánh  $Max$  với  $i=2, 3, \dots, N$ , nếu  $a_i > Max$  ta gán giá trị mới cho  $Max$ .

*Thuật toán*

- B1: Nhập dãy số  $a_i$ .
- B2:  $Max \leftarrow a_1$ .
- 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 giả mã tỏ ra khá thuận lợi trong việc viết cũng như đọc thuật toán. Thông thường người ta sử dụng 3 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.

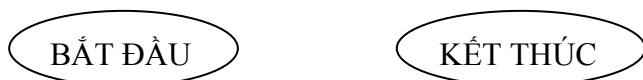
### II.1.2.1. Ngôn ngữ lưu đồ

Ngôn ngữ lưu đồ hay sơ đồ khối là một công cụ rất trực quan để diễn đạt các thuật toán. 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 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. Lưu đồ được tạo thành từ 4 thành phần chủ yếu sau đây:

- a. Nút giới hạn:

Được biểu diễn bởi hình ovan có ghi chữ bên trong như:

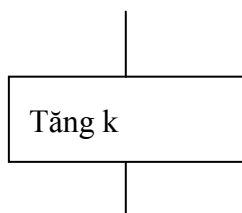


Các nút trên còn được gọi là nút đầu và nút cuối của lưu đồ.

b. Nút thao tác:

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

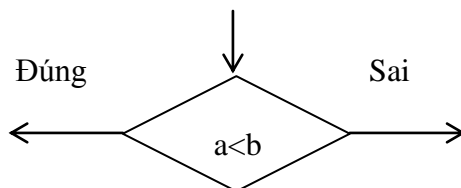
Ví dụ:



c. Nút điều kiện:

Thường là một hình thoi có ghi điều kiện cần kiểm tra. Trong các cung nối với nút này có 2 cung ra chỉ hướng đi theo 2 trường hợp: điều kiện đúng và điều kiện sai.

Ví dụ:



d. 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.

Ví dụ:

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

- Đầu vào: số số nguyên dương N và N số nguyên  $a_1, a_2, \dots, a_N$
- Đầu ra: số nguyên lớn nhất của dãy  $a_k$ , k trong khoảng  $[1 \dots N]$

*Ý tưởng*

- Khởi tạo giá trị  $\text{Max} = a_1$

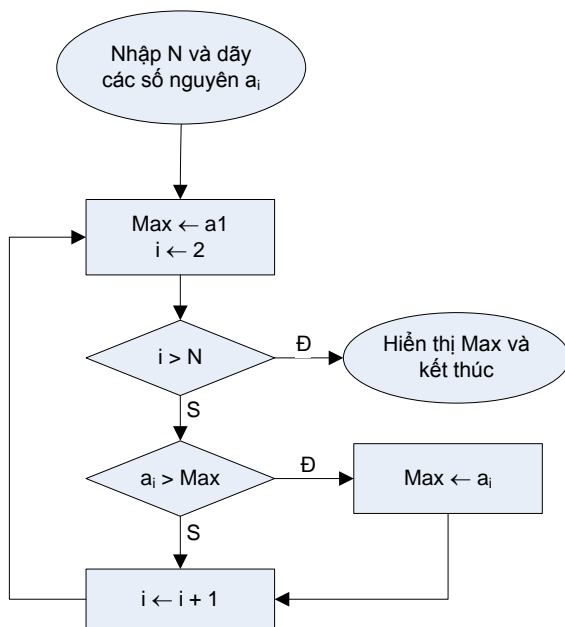


- Lần lượt so sánh Max với  $i=2, 3, \dots, N$ , nếu  $a_i > \text{Max}$  ta gán giá trị mới cho Max.

*Thuật toán*

- B1: Nhập dãy số  $a_i$ .
- B2:  $\text{Max} \leftarrow a_1$ .
- B3: Nếu  $i > N$ , thuật toán kết thúc và Max là giá trị lớn nhất của dãy cần tìm
- B4: Nếu  $a_i > \text{Max}$ , gán  $a_i$  cho Max.
- B5: Tăng  $i$  lên 1 đơn vị.
- B6: Quay lên b4.
- B7: Kết thúc.

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



*Qui ước:*

- Hình thoi biểu diễn thao tác so sánh
- Hình chữ nhật thể hiện 1 thao tác tính toán đơn giản hay phức tạp
- Hình ô van thể hiện thao tác bắt đầu hay kết thúc.

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

- Đầu vào: Dãy A gồm N số nguyên  $a_1, a_2, \dots, a_N$
- Đầu ra: Dãy A được sắp lại theo thứ tự không giảm

*Ý tưởng*

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

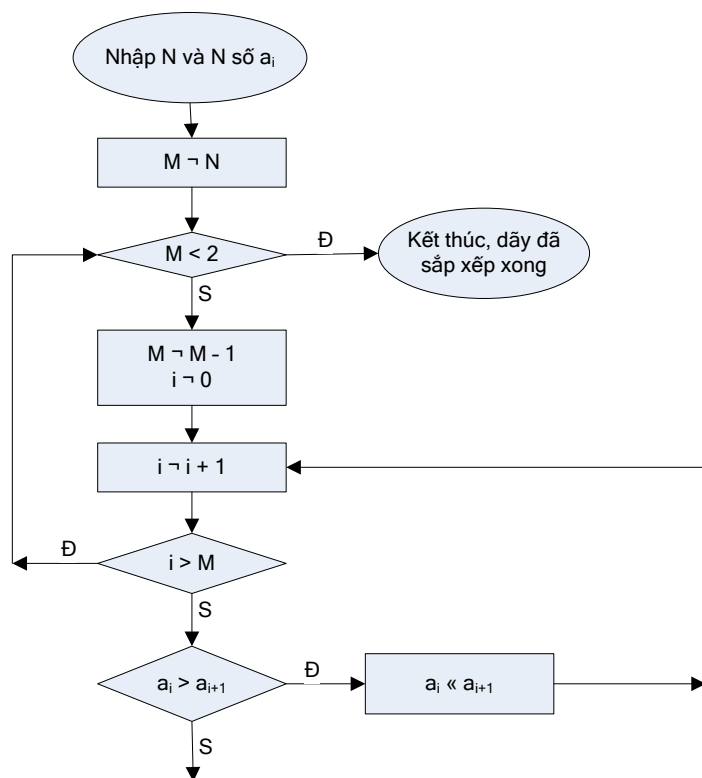
*Thuật toán*

- B1: Nhập số  $N$  và dãy số  $a_1, a_2, \dots, a_N$
- B2:  $M \leftarrow N$ .
- B3: Nếu  $M < 2$  thì thuật toán kết thúc và hiển thị dãy đó.
- B4:  $M \leftarrow M - 1, i \leftarrow 0$ .
- B5: Tăng  $i$  lên 1 đơn vị.
- B6: Nếu  $i > M$  thì quay lại bước 3.
- B7: Nếu  $a_i > a_{i+1}$  thì trao đổi hai số đó cho nhau .
- B8: Quay lên bước 5.

*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 trao 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  $M$  được gán với giá trị  $N$  và dừng khi  $M < 2$ .
- Trong thuật toán trên,  $i$  được khởi tạo giá trị 0 và tiến tới  $M + 1$

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



### II.1.2.2. Mã giả

Việc sử dụng lưu đồ sẽ rất cồng kềnh đối với các thuật toán phức tạp. Vì vậy người ta còn viết 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 ta còn sử dụng cả 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 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:

$i := i + 1$

Hay

$i \leftarrow i + 1$

Các cấu trúc thường gặp trong mã giả gồm:

- Cấu trúc chọn:

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

và

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

else (hành động)

- Cấu trúc lặp:

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

hoặc

**repeat**

(hành động)

**until** (điều kiện)

hoặc

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

hoặc

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

- Cấu trúc nhảy goto:

**goto** nhãn x;

**Ví dụ:**

### II.1.3. Thuật toán đệ qui

Thuật giải đệ qui 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:

- Tính xác định
- Tính dừng
- Tính đúng

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 nhưng lại khá đơn giản và được chấp nhận. 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 nêu 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 qui”.

### **Thuật toán đệ qui tính giai thừa của một số tự nhiên:**

Input: số tự nhiên  $n$

Output:  $F(n)=n!$

Thuật giải:

1.  $F:=1;$
2. If  $n>0$  then  
 $F:=F(n-1)*n;$
3. Output  $F$ .

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

1. Trong thuật toán đệ qui thường gồm 2 phần: phần cơ sở và phần đệ qui:
  - Phần cơ sở nằm trong các trường hợp không cần thực hiện lại thuật toán (hay không có yêu cầu gọi đệ qui).
  - Phần đệ qui là phần trong thuật toán có yêu cầu gọi đệ qui, tức là yêu cầu thực hiện lại thuật toán. Trong phần đệ qui, yêu cầu gọi đệ qui thường được đặt trong một điều kiện kiểm tra việc gọi đệ qui.
2. Về mặt cài đặt, nếu như có sử dụng biến cục bộ trong thủ tục hay hàm đệ qui thì các biến này được tạo ra và được đặt trong vùng nhớ “STACK”. Do đó quá trình gọi đệ qui dễ gây ra tình trạng tràn STACK. Trong nhiều trường hợp có thể viết lại thuật toán đệ qui dưới dạng lặp.

## **II.1.4. Một số thuật toán thông dụng**

### **II.1.4.1. Thuật toán số học**

#### **Thuật toán kiểm tra số nguyên tố**

Vấn đề: Cho một số nguyên dương  $p$ . Làm thế nào để biết được  $p$  có phải 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. Trước hết ta xem xét điều kiện  $p \neq 1$ . Nếu  $p=1$  thì  $p$  không nguyên tố. Nếu đúng là  $p \neq 1$  thì tiếp tục kiểm tra xem trong các số từ 2 đến  $p-1$  có ước số của  $p$  hay không? Nếu có  $p$  không nguyên tố; ngược lại  $p$  nguyên tố.

Thuật toán:

Nhập:  $p$  nguyên dương

Xuất: kết luận về tính nguyên tố của p

Thuật toán:

1.       if p=1 then  
          begin  
              Xuất: p không nguyên tố;  
              Dừng thuật toán;  
          end
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  
              begin  
                  flag:=FALSE;  
                  break; (ngắt vòng lặp FOR)  
              end
4.       if flag=TRUE then  
          Xuất: p là số nguyên tố  
      else  
          Xuất: p không là số nguyên tố

**Thuật toán ...**

#### II.1.4.2. Thuật toán về dãy

##### **Tìm phần tử lớn nhất trong một dãy hữu hạn**

Chúng ta 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. 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.

Nhập: dãy số  $a_1, a_2, a_3, \dots, a_n$

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

---

Thuật toán:

1.  $\text{max} := a_1;$
2. for  $i := 2$  to  $n$  do  
if  $\text{max} < a_i$  then  $\text{max} := a_i$
3.  $\text{max}$  là giá trị lớn nhất trong dãy số

## II.1.5. Thuật giải heuristic

### II.1.5.1. Thuật giải – Sự mở rộng khái niệm của thuật toán

Như chúng ta đã biết, cách giải đệ qui không đáp ứng đầy đủ các yêu cầu đối với một thuật toán. Đó là sự mở rộng khái niệm thuật toán. Trong quá trình nghiên cứu giải quyết bài toán, 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 dài 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 định trên, người ta thấy rằng cần phải có những đổi mới cho khái niệm thuật toán. Một số tiêu chuẩn của thuật toán được mở rộng: tính xác định, tính đúng đắn. Việc mở rộng tính xác định đối với thuật toán được thể hiện qua các thuật giải đệ qui và các giải thuật ngẫu nhiên. Tính đúng 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. Khái niệm mở rộng này của thuật toán đã mở rộng cửa cho chúng ta trong việc tìm kiếm các phương pháp giải quyết bài toán.

### II.1.5.2. 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 ta 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 tư: 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: ...**

## **BUỔI 7.**

### **II.2. Giải quyết bài toán (4 tiết LT, 2 tiết BT)**

#### **II.2.1. Khái niệm về bài toán**

Mọi cá nhân cũng như mọi cộng đồng xã hội phải liên tục giải quyết những vấn đề và bài toán đặt ra trong quá trình tồn tại và phát triển.

Nhà triết học cổ Hy Lạp Xôcrat thường nói: “thông thường, nhiều người quan niệm vấn đề có nghĩa rộng hơn bài toán và bài toán là một loại vấn đề mà để giải quyết phải liên quan ít nhiều đến tính toán: bài toán trong vật lý, hóa học, xây dựng, kinh tế... cho nên từ vấn đề vẫn thường được dùng với nghĩa rộng hơn trong toán học”.

Nhà toán học Pitago đã phân chia mọi vấn đề mà con người phải giải quyết thành 2 loại:

- Theorema: là vấn đề cần được khẳng định tính đúng sai. Chúng ta thường quen với loại vấn đề này qua việc 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. Các ví dụ minh họa điển hình cho loại bài toán này là bài toán dựng hình, tổng hợp chất hóa học, tìm đường đi ngắn nhất...

Như nhiều nhà nghiên cứu, việc giải quyết vấn đề-bài toán đã chỉ ra sau này, cả 2 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. hiện

→ 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 có lý hoặc hành động thích hợp để đạt được B.

Để cho việc giải quyết vấn đề - bài toán được xác định cũng 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 → trong sơ đồ đã nêu. Đối với tin học, sơ đồ này được hiểu với nghĩa A là input, B là output và → là chương trình tạo thành từ các lệnh cơ bản của máy tính cho phép biến đổi A thành B.

Như ta đã thấy, chương trình chỉ là một cách mã hóa lại thuật toán hoặc thuật giải để giải quyết vấn đề - bài toán đã cho.

Điều khó khăn nhất hiện nay đối với việc xây dựng thuật toán hoặc thuật giải là tính không xác định của hầu hết mọi vấn đề và bài toán. 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 thuật giải ( → ) thường được nêu ra trong bài toán hoặc vấn đề.

Có thể nói: thuật toán hay thuật giải cung cấp đủ lượng thông tin để giải quyết vấn đề - bài toán. Chính vì thế, hiện nay để giải quyết một vấn đề trên máy tính, việc thiết kế thuật giải 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 trong đầu óc con người giải quyết vấn đề - bài toán sẽ được kiến tạo nên thuật toán hay thuật giải 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 thuật toán hay thuật giải chúng ta cần phải làm sao biểu diễn được nội dung bài toán – vấn đề cũng như mọi tri thức có liên quan dưới dạng tường minh và hết sức đầy đủ. Đây chính là lĩnh vực mới mẻ và có ý nghĩa thực tiễn lớn lao: xây dựng một trí tuệ nhân tạo cho máy.

### II.2.2. Các bước giải quyết bài toán bằng máy tính

Việc sử dụng máy tính để giải quyết một vấn đề nào đó thường được quan niệm một cách không chuẩn xác rằng đơn giản đó chỉ là việc lập trình thuần túy. Thực ra, đó là một quá trình phức tạp bao gồm nhiều giai đoạn phát triển. Các giai đoạn quan trọng gồm có :

#### B1. Xác định bài toán

Làm rõ yêu cầu của người sử dụng, đánh giá, nhận định tính khả thi của bài toán.

#### B2. 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... Nói 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.

#### B3. Xây dựng thuật toán hoặc thuật giải

Xây dựng mô hình chặt chẽ, chính xác hơn và chi tiết hóa hơn phương pháp đã lựa chọn, xác định rõ ràng dữ liệu vào, 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 từng bước.

#### B4. Cài đặt chương trình

Mô tả thuật giải bằng chương trình.

#### B5. Hiệu chỉnh chương trình

Chạy thử để phát hiện và điều chỉnh các sai sót có thể có ở bước 4. Có 2 loại lỗi : lỗi cú pháp và lỗi ngữ nghĩa.

#### B6. 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 1 lần nữa.

Việc giải quyết một bài toán cụ thể được thực hiện qua 2 giai đoạn :

Giai đoạn quan niệm : gồm các bước phân tích, lựa chọn mô hình, xây dựng thuật giải, cài đặt chương trình

Giai đoạn khai thác và bảo trì

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.

### **Bài tập về Thuật toán (2 tiết BT)**

...

## **BUỔI 8.**

### **II.2.3. Các phương pháp giải quyết bài toán bằng máy tính**

#### **II.2.3.1. Giải quyết bài toán theo hướng xác định trực tiếp lời giải:**

Thường sử dụng trong quá trình học tập. Ví dụ: Tìm nghiệm phương trình bậc 2 theo định lý Viet.

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

Các công thức lập để tính toán gần đúng nghiệm của bài toán cũng được xem là cách xác định trực tiếp lời giải. Về nguyên tắc, lời giải xác định bởi các công thức lập có thể xấp xỉ lời giải thật

sự của bài toán với độ chính xác tăng theo quá trình lặp. Điều này hạn chế khi tính toán thủ công nhưng lại là thế mạnh của máy tính.

### II.2.3.2. 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.

Diễn hình như các phương pháp sau:

- Phương pháp liệt kê hay vét cạn: chỉ 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.
- Phương pháp quay lui: nhiều khi không thể xác định hay liệt kê từ trước khi thử tập mọi khả năng chứa lời giải của bài toán. Chẳng hạn tìm đường đi trong mê cung thì làm sao biết hết mọi ngã 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 thừa số nguyên tố, bài toán 8 quân hậu...

### II.2.4. Phân loại bài toán

#### II.2.4.1. Độ phức tạp thuật toán

Thuật ngữ “Độ phức tạp thuật toán” thể hiện khối lượng thời gian và không gian cần thiết để thực hiện thuật toán. Việc xem xét độ phức tạp về không gian của thuật toán phụ thuộc phần lớn vào cấu trúc dữ liệu được sử dụng trong cài đặt thuật toán. Trong phần này chúng ta chỉ đề cập đến độ phức tạp về thời gian của thuật toán.

Đánh giá một cách chính xác thời gian thực hiện một chương trình phụ thuộc vào nhiều yếu tố là một công việc khó khăn. Tuy nhiên chúng ta cũng có thể đạt được những thông tin rất hữu ích khi phân tích độ phức tạp thời gian của thuật toán cơ sở. Độ phức tạp thời gian của thuật toán thường được đánh giá dựa vào số lượng thao tác được sử dụng trong thuật toán và số lượng thao tác này phụ thuộc vào độ lớn của dữ liệu nhập. Các thao tác sử dụng để đo độ phức tạp của thuật toán có thể là: phép so sánh 2 số nguyên, cộng, nhân, chia hai số nguyên hay bất kỳ một thao tác cơ bản nào khác. Như thế ta có thể xem thời gian thực hiện thuật toán là một hàm phụ thuộc vào dữ liệu nhập (thường là độ lớn của dữ liệu nhập).

Chúng ta có thể đặt ra câu hỏi về thời gian thực hiện thuật toán nhỏ nhất đối với dữ liệu nhập có độ lớn  $n$ . Ta có thể nêu lên một số thuật toán có dữ liệu nhập có độ lớn  $n$  như: sắp xếp dãy  $n$  số nguyên, tìm số nhỏ nhất trong dãy  $n$  số nguyên... Thời gian nhỏ nhất này được gọi là thời gian

thực hiện thuật toán trong trường hợp tốt nhất đối với dữ liệu có dữ liệu nhập có độ lớn  $n$ . Tương tự ta cũng thường đề cập đến thời gian thực hiện thuật toán lớn nhất đối với các dữ liệu nhập có độ lớn  $n$  và gọi là thời gian thực hiện thuật toán trong trường hợp xấu nhất đối với dữ liệu có độ lớn  $n$ . Ngoài ra, đối với thuật toán có dữ liệu nhập có độ lớn  $n$  trong một tập hữu hạn nào đó, ta còn muốn tính ra thời gian trung bình để thực hiện thuật toán.

Ví dụ: Thuật toán tìm giá trị lớn nhất trong dãy gồm  $n$  số nguyên. (đã nêu trong các phần trước)

Trong thuật toán này nếu ta xem thời gian thực hiện thuật toán là số lần thực hiện phép so sánh hay phép gán thì thời gian thực hiện thuật toán trong trường hợp xấu nhất là:

$$t(n) = 1 + 2(n-1) = 2n + 1$$

Và thời gian thực hiện thuật toán trong trường hợp lớn nhất là:

$$T(n) = 1 + (n-1) = n$$

Thông thường ta ít chú ý tới con số chính xác về thời gian thực hiện thuật toán trong trường hợp xấu nhất và trong trường hợp tốt nhất. Điều mà chúng ta thường quan tâm hơn khi đánh giá độ phức tạp thời gian của thuật toán là mức độ tăng lên của thời gian thực hiện thuật toán khi độ lớn của dữ liệu nhập tăng lên. Chẳng hạn, một thuật toán đang được xem xét nào đó có thời gian thực hiện trong trường hợp xấu nhất và tốt nhất lần lượt là:

$$t(n) = 20n^2 + 5n + 1$$

$$T(n) = n^2 + 10n + 1$$

Như thế nếu như  $n$  rất lớn thì ta có thể xấp xỉ  $t(n)$  và  $T(n)$  với  $20n^2$  và  $n^2$ . Có thể nói rằng  $t(n)$  tăng giống như  $n^2$  khi  $n$  tăng, hay nói  $t(n)$  có “cấp cao nhất” là  $n^2$  và ta viết  $t(n) = O(n^2)$ .

Tương tự  $T(n) = O(n^2)$ . Cuối cùng ta nói độ phức tạp của thuật toán là  $O(n^2)$  (độ phức tạp của thuật toán có cấp cao nhất là  $n^2$ ).

Định nghĩa: Cho 2 hàm  $f$  và  $g$  có miền xác định trong tập số tự nhiên  $N$ . Ta viết:

$$f(n) = O(g(n))$$

Và nói  $f(n)$  có cấp cao nhất là  $g(n)$  khi có một hằng số dương  $C$  sao cho:

$$|f(n)| \leq Cg(n)$$

Với hầu hết  $n$  thuộc miền xác định của các hàm  $f$  và  $g$ . Từ “hầu hết” ở đây ý nói là “với mọi chỉ trừ một số hữu hạn”.

Định lý. Nếu  $f(n)$  là một đa thức bậc  $k$  theo  $n$  thì ta có  $f(n) = O(n^k)$ .

Liên quan đến độ phức tạp của thuật toán ta có một số thuật ngữ thông dụng thường dùng trong một số trường hợp sau đây:

- Độ phức tạp hằng:  $O(1)$
- Độ phức tạp logarith:  $O(\log n)$
- Độ phức tạp tuyến tính:  $O(n)$

- Độ phức tạp  $n \log n$ :  $O(n \log n)$
- Độ phức tạp đa thức:  $O(n^b)$
- Độ phức tạp lũy thừa:  $O(b^n)$ , trong đó  $b > 1$
- Độ phức tạp giai thừa:  $O(n!)$

#### II.2.4.2. Phân loại bài toán

##### Bài toán đa thức:

Lớp giải thuật đa thức chứa những giải thuật có độ phức tạp bị chặn bởi một đa thức.

Vì độ phức tạp của ta thật sự là độ đo thời gian cần thi hành một thuật giải, do vậy thường nói những giải thuật ở lớp này chạy trong thời gian đa thức.

Ví dụ: thuật toán sắp chèn và Quicksort

Một giải thuật không ở lớp này thì được tính cách hóa bởi thời gian cực lâu khi giá trị đưa vào càng tăng. Ví dụ một giải thuật có độ phức tạp  $2^n$  thì sẽ gần như không chạy được vì thời gian quá lâu khi dữ liệu vào lớn.

Sự quyết định của một bài toán có thể giải trên lý thuyết được hay không phụ thuộc vào việc nó có thuộc lớp đa thức hay không.

##### Bài toán không đa thức

Nhiều bài toán có thể giải được lại không thuộc lớp đa thức. Ví dụ bài toán lập danh sách tất cả nhóm người có số lượng là một hay nhiều hơn có thể hình thành từ nhóm  $n$  người bởi vì có  $2^n - 1$  nhóm như vậy bất cứ giải thuật nào để giải bài toán này phải có ít nhất  $2^n - 1$  bước và độ phức tạp ít nhất phải cỡ đó. Do đó bất cứ lời giải nào cũng trở nên tốn nhiều thời gian khi kích thước nhóm ban đầu tăng lên.

##### NP problems (những bài toán NP)

Tới thời điểm này, ta biết rằng giải thuật tuân theo thuyết tự quyết nghĩa là tại mỗi bước trong việc thực thi một giải thuật, chỉ thị kế tiếp để thực thi phải được nhận ra là duy nhất và được mô tả rõ ràng. Bây giờ ta xem cái gì có thể được thực hiện nếu ta cho phép những giải thuật được đưa ra những lựa chọn từ đó ta được phép chọn một cách không tự quyết.

Ví dụ: Khi bạn đến đèn giao thông thứ 3, bạn hãy theo một trong những con đường tại “giao lộ” mà không đưa ra bất kỳ một dấu hiệu nào để mà ta biết được sự lựa chọn nào tốt hơn. (ngược lại, một giải thuật đúng phải giải thích sự quyết định phải được làm thế nào và chỉ có một lựa chọn là có thể). Ta đặt những giải thuật như vậy là những giải thuật không tự quyết.

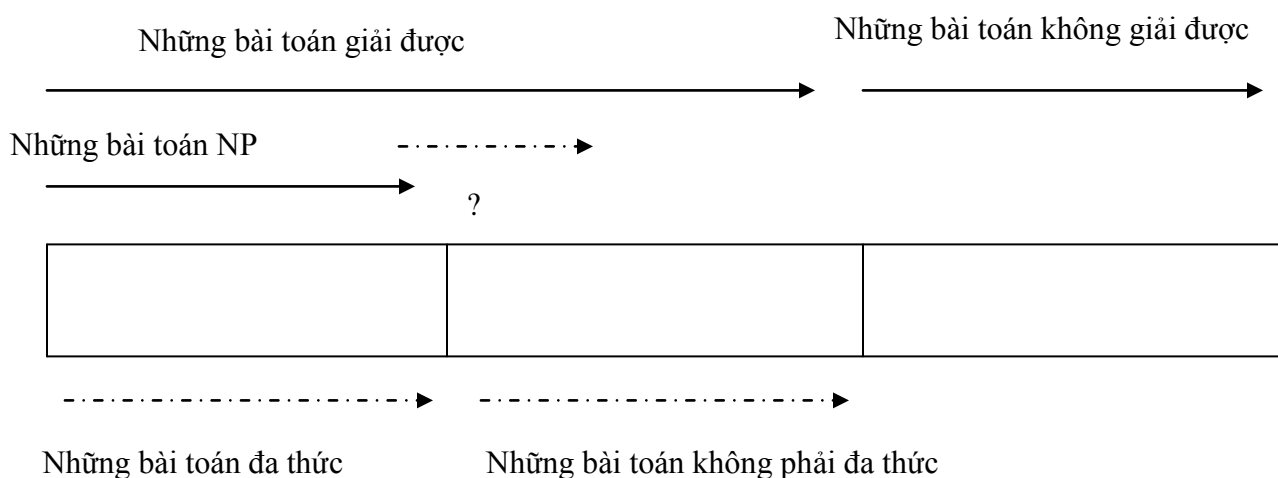
Tại sao các nhà khoa học máy tính lại quan tâm đến những giải thuật không tự quyết sau hết, những giải thuật không tự quyết dường như là khá hữu dụng trong thực tế. Một câu trả lời là bằng cách nghiên cứu khả năng của những giải thích này, ta có thể hiểu sâu hơn về khả năng và hạn chế của giải thuật tự quyết.

Ta định nghĩa độ phức tạp về thời gian để giải một bài toán không tự quyết là thời gian cần thiết để thực hiện giải bài toán đó nếu ta chọn đúng tại mỗi điểm của sự lựa chọn. “chọn lựa đúng đắn” ta muốn nói đến sự lựa chọn mà dẫn đến lời giải cho bài toán trong khoảng thời gian tốt nhất. Ta có thể xem định nghĩa này như là một cố gắng để đánh giá một giải thuật tốt hơn là sự lựa chọn của ta là tốt; quan điểm của ta là nếu ta chọn sai thì đó là lỗi của ta chứ không phải của giải thuật.

Ví dụ: bài toán người bán hàng rong?

Ta gọi một bài toán mà có thể giải được trong thời gian đa thức bằng một giải thuật không tự quyết là bài toán đa thức không tự quyết hay bài toán NP. Dĩ nhiên lớp bài toán NP chứa những bài toán đa thức thực sự, bởi vì nếu một bài toán có thể giải bằng giải thuật tự quyết trong thời gian đa thức, lúc đó chắc chắn nó được giải trong thời gian đa thức bằng một giải thuật không tự quyết. Hơn nữa có những gợi ý rằng những bài toán NP mà không phải bài toán đa thức. (ví dụ bài toán người bán hàng rong). Phân loại bài toán NP hiện vẫn còn là một điều bí mật trong ngành máy tính ngày nay.

Tóm lại, ta thấy rằng những bài toán được phân loại có thể giải quyết (có một lời giải giải thuật) hay không thể giải (không có lời giải giải thuật) như hình sau:



Hơn nữa, nội tại trong lớp của bài toán có thể giải có 2 lớp con: tập hợp các bài toán đa thức được xem là có lời giải thực tế; tập hợp bài toán không đa thức mà lời giải của nó được xem là thực tế chỉ cho những số liệu đưa vào có chọn lựa cẩn thận và tương đối nhỏ. Cuối cùng, có những bài toán NP được xếp vào lớp bài toán có thể giải và chứa những bài toán đa thức. Có hay không những lời giải đa thức đúng đắn cho tất cả các bài toán NP vẫn còn là câu hỏi mở.

## PHẦN 3. LẬP TRÌNH

(21 tiết LT, 9 tiết BT)

### III.1. Tổng quan về ngôn ngữ C (3 tiết LT)

#### III.1.1. Lịch sử phát triển

Điều kiện ra đời:

- 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 thể mạnh trong xử lý 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 lý văn bản, cơ sở dữ liệu, xử lý 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 dịch trên bộ dịch của mình thì sẽ tốn rất nhiều công sức) và 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 2 của cuốn sách "*The C programming language*" của các tác giả W. Kernighan và Dennis Ritchie. Và từ đó đến nay phiên bản này vẫn thường được nhắc đến với tên gọi là *ANSI C* hay *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 tài liệu này.

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

#### 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.

#### 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++

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



*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...**

### 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 – <i>space</i> ) 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 `danhsachsinhvien` để 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 `danhsachsinhvien` hay `ds_sv...` Và ngược lại định danh `danhsachsinhvien` 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 `ding_danh` khác với `Dinh_danh`.
- Một thói quen của những người lập trình là các hằng thường được đặt tên bằng chữ hoa, 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
<code>HANG_SO_1, _CONSTANT_2</code>	hằng
<code>a, b, i, j, count</code>	biến
<code>nhap_du_lieu, tim_kiem, xu_li</code>	hàm
<code>sinh_vien, mat_hang</code>	cấu trúc

#### 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 2 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** thì nó 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 lý thuộc dạng nào, có cấu trúc ra sao, có thể thực hiện các phép xử lý nào đối với đối tượng đó, hay là cần phải lưu trữ đối tượng đó như thế nào...

### 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: đó chính 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	0x7D7
396	0614	0x18C

#### Biểu diễn hằng số thực

Có 2 cách biểu diễn hằng số thực:

- Dưới dạng số thực dấu phẩy tĩnh.
- Dưới dạng số thực dấu phẩy động.

Ví dụ:

Số thực dấu phẩy tĩnh	Số thực dấu phẩy động
3.14159	31.4159 E-1
123.456	12.3456 E+1 hoặc 1.23456 E+2

#### Biểu diễn hằng kí tự

Có 2 cách biểu diễn hằng kí tự:

- Bằng kí hiệu của kí tự đó đặt giữa 2 dấu nháy đơn.
- Bằng số thứ tự của kí tự đó trong bảng mã ASCII (và 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 0x41
Dấu nháy đơn '	'\''	39	hoặc 047	hoặc 0x27
Dấu nháy kép "	'\"'	34	hoặc 042	hoặc 0x22
Dấu gạch chéo ngược \	'\\'	92	hoặc 0134	hoặc 0x5c
Kí tự xuống dòng	'\n'			
Kí tự NUL	'\0'	0	hoặc 00	hoặc 0x0
Kí tự Tab	'\t'	9	hoặc 09	hoặc 0x9

*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”...

### 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, và 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.

### 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 trong ta cần 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 2.718282
log(x)	logarithm tự nhiên (cơ số e) của x	$\ln x$	log(2.718282) bằng 1.0
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	$\tan x$	tan(0.0) bằng 0.0
ceil(x)	phần nguyên giả 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 ở bài 6.

### 1.2.8. Biểu thức

Biểu thức là sự ghép nối các toán tử (*operator*) và các toán hạng (*operand*) theo một quy tắc xác định.

Các toán hạng trong biểu thức có thể là biến, hằng, hàm hoặc một biểu thức khác. 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, \* là kí hiệu của toán tử nhân. Nếu chieu\_dai, chieu\_rong, chieu\_cao là các biến (hoặc hằng) 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 ở bài 2.

### 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 2 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: lệnh khối, các cấu trúc lệnh rẽ nhánh, cấu trúc lệnh lặp...

Lệnh khối 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.

#### 1.2.10. Chú thích

Để giúp việc đọc và hiểu chương trình viết ra được dễ dàng hơn, chúng ta 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, có 2 cách để viết lời chú thích

- Dùng 2 dấu sỏ chéo liên tiếp // để kí hiệu toàn bộ vùng bắt đầu từ 2 dấu sỏ 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 2 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, mọi chương trình viết bằng ngôn ngữ C sẽ có cấu trúc gồm 6 phần có thứ tự như sau:

Khai báo tệp tiêu đề

```
#include
```

Định nghĩa kiểu dữ liệu

```
typedef ...
```

Khai báo các hàm nguyên mẫu

Khai báo các biến toàn cục

Định nghĩa hàm **main()**

```
main()
```

```
{
```

```
    ...
```

```
}
```

Định nghĩa các hàm đã khai báo nguyên mẫu

- 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. // Chương trình sau sẽ nhập vào từ bàn phím 2 số nguyên
2. // và hiển thị ra màn hình tổng, hiệu tích của 2 số nguyên vừa nhập vào
3. #include <stdio.h>
4. #include <conio.h>
5. void main()
6. {
7.     // khai báo các biến trong chương trình
8.     int a, b
9.     int tong, hieu, tich;
10.    // Nhập vào từ bàn phím 2 số nguyên
11.    printf("\n Nhập vào số nguyên thứ nhất: ");
12.    scanf("%d",&a);
13.    printf("\n Nhập vào số nguyên thứ hai: ");
14.    scanf("%d",&b);
15.    // Tính tổng, hiệu, tích của 2 số vừa nhập
16.    tong = a+b;
17.    hieu = a - b;
18.    tich = a*b;
19.    // Hiển thị các giá trị ra màn hình
20.    printf("\n Tổng của 2 số vừa nhập là %d", tong);
21.    printf("\n Hiệu của 2 số vừa nhập là %d", hieu);
22.    printf("\n Tích của 2 số vừa nhập là %d", tich);
23.    // Cho người sử dụng ấn phím bất kỳ để kết thúc
24.    getch();
25. }
```

Trong chương trình trên chỉ có 2 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.



## BUỔI 9.

### III.1.4. Biên dịch chương trình C

### III.1.5. 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 chương trình biên dịch (*compiler*)

Hiện có nhiều chương 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.1.6. 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: Bạn cần 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 bạn 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 cho bạn, và bạn chỉ cần ấn Enter để chuyển sang bước tiếp theo.

B4: Ở bước 4, bạn cần xác định thư mục cài đặt. Thư mục này sẽ chứa các file của Turbo C++ 3.0 để bạn 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 bạn 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 cho bạn.

*Lưu ý:* Bạn có thể copy toàn bộ thư mục đã cài đặt của Turbo C++ 3.0 về máy và sử dụng, nhưng bạn phải chỉ cho Turbo C++ biết đường dẫn tới các tệp tiêu đề và các tệp thư viện bằng cách vào menu *Option*, chọn *Directories*.

### Sử dụng môi trường Turbo C++ 3.0

Sau khi cài đặt xong, bạn có thể 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 giữ 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 của bạn có lỗi thì Turbo C++ 3.0 sẽ báo lỗi và bạn phải sửa lại đến khi không còn lỗi. Nếu chương trình bạn 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 nhât: 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 (4 tiết LT)

### III.2.1. Các kiểu dữ liệu chuẩn trong C (1 tiết LT)

Kiểu dữ liệu	Ý nghĩa	Kích thước	Miền biểu diễn
<b>unsigned char</b>	Kí tự	1 byte	$0 \div 255$
<b>char</b>	Kí tự	1 byte	$-128 \div 127$
<b>unsigned int</b>	Số nguyên không dấu	2 bytes	$0 \div 65,535$
<b>short int</b>	Số nguyên có dấu	2 bytes	$-32,768 \div 32,767$
<b>int</b>	Số nguyên có dấu	2 bytes	$-32,768 \div 32,767$
<b>unsigned long</b>	Số nguyên không dấu	4 bytes	$0 \div 4,294,967,295$
<b>long</b>	Số nguyên có dấu	4 bytes	$-2,147,483,648 \div 2,147,483,647$
<b>float</b>	Số thực dấu phẩy động, độ chính xác đơn	4 bytes	$\pm 3.4E-38 \div \pm 3.4E+38$
<b>double</b>	Số thực dấu phẩy động độ chính xác kép	8 bytes	$\pm 1.7E-308 \div \pm 1.7E+308$
<b>long double</b>	Số thực dấu phẩy động	10 bytes	$\pm 3.4E-4932 \div \pm 1.1E+4932$

#### 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:

***kiểu\_dữ\_liệu danh\_sách\_tên\_biến;***

Ví dụ:

```
float x, y;
double z;
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 lệnh nay bien a se co gia tri bang 3
float x = 5.0, y = 2.6; // sau lệnh 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 2 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 phẩy 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 cho 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;
```

## Bài tập về Giải quyết bài toán (2 tiết BT)

### BUỔI 10.

#### III.2.2. Các biểu thức

##### *Biểu thức số học*

Biểu thức số học là biểu thức mà giá trị của nó là cái 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)

$$3 * 3.7, 8 + 6/3, a + b - c \dots$$

##### *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 2 giá trị Đúng (*TRUE*) hoặc Sai (*FALSE*), vì vậy biểu thức quan hệ là một trường hợp riêng của biểu thức logic. 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 2 số nguyên	10/3; (bằng 3)

%	Phép chia lấy phần dư	Giữa 2 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	2 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	2 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 111)
^	Phép HOẶC CÓ LOẠI TRỪ nhị phân	2 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	$a \ll n$ (có giá trị $a \cdot 2^n$ ) 101 << 2 (có giá trị 10100)
>>	Phép DỊCH PHẢI nhị phân	Số nhị phân	$a \gg n$ (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 2 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 2 số nguyên hoặc thực.	6 >= 4 (có giá trị 1) $x \geq a$
<	So sánh nhỏ hơn giữa 2 số nguyên hoặc thực.	5 < 3 (có giá trị 0),

<=	So sánh nhỏ hơn hoặc bằng giữa 2 số nguyên hoặc thực.	5 <= 5 (có giá trị 1) 2 <= 9 (có giá trị 1)
==	So sánh bằng nhau giữa 2 số nguyên hoặc thực.	3 == 4 (có giá trị 0) a == b
!=	So sánh không bằng (so sánh khác) giữa 2 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 2 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 2 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 ta sẽ thu được biểu thức gán. Biểu thức gán là biểu thức có dạng

$$\text{tên\_biến} = \text{biểu\_thức}$$

Biểu thức gán là biểu thức nên nó 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 2, ta đã gán giá trị 2007 cho biến *b*, sau đó ta 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 3, 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**

```
x = x + y
```

```
x = x - y
```

```
x = x * y
```

```
x = x / y
```

```
x = x % y
```

```
x = x >> y
```

```
x = x << y
```

```
x = x & y
```

### **Lệnh gán thu gọn**

```
x += y
```

```
x -= y
```

```
x *= y
```

```
x /= y
```

```
x %= y
```

```
x >>= y
```

```
x <<= y
```

```
x &= y
```

$$x = x \mid y$$

$$x \mid = y$$

$$x = x \wedge y$$

$$x^+ = y$$

### 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*

- Biểu thức con trong ngoặc được tính toán trước các phép toán khác
- Phép toán một ngôi đứng bên trái toán hạng được kết hợp với toán hạng đi liền nó.
- Nếu toán hạng đứng cạnh hai toán tử thì có 2 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

**<ten biến> = <ten biến> + 1;**

**<ten biến> = <ten 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*

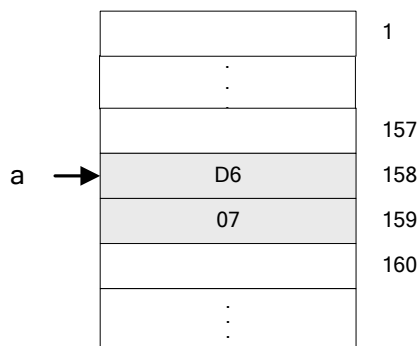
```
int a, b, c;
a = 3;           // a bằng 3
b = a++;        // dạng hậu tố. b bằng 3; a bằng 4
c = ++b;        // dạng tiền tố. b bằng 4, c bằng 4;
```

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 dưới đây minh họa một biến tên là *a*, kiểu dữ liệu **int** được lưu trữ trong bộ nhớ tại 2 ô nhớ có địa chỉ lần lượt là 158 và 159. Giá trị của biến *a* là 2006 = 0x07D6. Khi đó địa chỉ của biến *a* sẽ là 158 hay 0000:9E (vì địa chỉ được mã hóa bởi 2 byte).



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;      // có giá trị là 158 hay 9E

### *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 ta 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 nếu các bạn mới bắt đầu lập trình thì các bạn 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 ĐÚNG), và 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 SAI).

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 (6 tiết LT)

#### 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. 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. Số lượng tham số trong danh\_sách\_tham\_số bằng số lượng nhóm các 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 và 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 2 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	<b>int, long, char</b>	Hiển thị tham số tương ứng dưới dạng số nguyên có dấu hệ đếm thập phân
%i	<b>int, long, char</b>	Hiển thị tham số tương ứng dưới dạng số nguyên có dấu hệ đếm thập phân
%o	<b>int, long, char</b>	Hiển thị tham số tương ứng dưới dạng số nguyên không dấu trong hệ đếm cơ số 8.
%u	<b>int, long, char</b>	Hiển thị tham số tương ứng dưới dạng số nguyên không dấu.
%x	<b>int, long, char</b>	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	<b>int, long, char</b>	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	<b>float, double</b>	Hiển thị tham số tương ứng dưới dạng số thực dấu phẩy động
%f	<b>float, double</b>	Hiển thị tham số tương ứng dưới dạng số thực dấu phẩy tĩnh
%g	<b>float, double</b>	Hiển thị tham số tương ứng số thực dưới dạng ngắn gọn hơn trong 2 dạng dấu phẩy tĩnh và dấu phẩy động
%c	<b>int, long, char</b>	Hiển thị tham số tương ứng dưới dạng kí tự
%s	<b>char *</b> (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 của ta 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

```
□1234
□□□34
```

Ở đây □ kí hiệu thay cho dấu trắng (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');
```

Kết quả

```
□ □ 1□ □ □ □ nguyen van a□g
□ □ 2□ □ □ □ □ tran van b□ □ k
```

Nếu nội dung dữ liệu không điền đầy số chỗ được cấp thì những chỗ không dùng đến sẽ được điền bởi dấu trắng.

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
```

**Cần lè 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	<b>%md, %ms, %mc</b>	printf("%3d",10); printf("%4s","CNTT"); printf("%2c",'A');
Độ rộng dành cho phần thập phân	<b>%m.nf</b>	printf("%5.1f",1234.5);
Căn lề trái	<b>%-md, %-ms, %-mc</b>	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 giả sử 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. Số nhóm kí tự định dạng bằng số địa chỉ của các biến trong *danh\_sách\_địa\_chỉ*. 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 <b>int</b>

%o	Định khuôn dạng dữ liệu nhập vào dưới dạng số nguyên kiểu <b>int</b> hệ cơ số 8
%x	Định khuôn dạng dữ liệu nhập vào dưới dạng số nguyên kiểu <b>int</b> hệ cơ số 16
%c	Định khuôn dạng dữ liệu nhập vào dưới dạng kí tự kiểu <b>char</b>
%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 <b>float</b>
%ld	Định khuôn dạng dữ liệu nhập vào dưới dạng số nguyên kiểu <b>long</b>
%lf	Định khuôn dạng dữ liệu nhập vào dưới dạng số thực kiểu <b>double</b>

*Ví dụ:*

```
#include <conio.h>
#include <stdio.h>
void main()
{
    // khai bao bien
    int a;
    float x;
    char ch;
    char* str;
    // 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
```

Kí tự: b

Xấu kí tự: 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 chuỗi kí tự, hàm scanf() nếu gặp các kí tự dấu trắng, dấu tab hay dấu xuống dòng thì nó sẽ hiểu là kết thúc nhập dữ liệu cho một chuỗi kí tự. Vì vậy trước khi nhập dữ liệu kí tự hay chuỗi kí tự ta nên dùng lệnh fflush(stdin).

### III.3.1.2. Các lệnh nhập xuất khác

Hàm gets(), cú pháp

***gets(xâu\_kí\_tự);***

Hàm gets() dùng để nhập vào từ bàn phím một chuỗi kí tự bao gồm cả dấu cách, điều mà hàm scanf() không làm được.

Hàm puts(), cú pháp

***puts(xâu\_kí\_tự);***

Hàm puts() sẽ hiển thị ra màn hình nội dung chuỗi\_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(), 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 báo biến
    char* str;
    // Nhập dữ liệu
    puts(“Nhập vào một xấu kí tự:”);
    fflush(stdin); gets(str);
    // Hiển thị dữ liệu vừa nhập vào
```

```
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 lệnh khối

Một cách hình thức ta có thể định nghĩa một lệnh khối là dãy các câu lệnh được đặt trong cặp dấu ngoặc nhọn { }.

```
{
    lệnh_1;
    lệnh_2;
    ....
    lệnh_n;
}
```

Trong lệnh khối có thể chứa lệnh khối khác, ta gọi đó là các lệnh khối lồng nhau. Sự lồng nhau của các lệnh khối là không hạn chế. Các lệnh trong lệnh khối được thực hiện tuần tự theo trật tự xuất hiện.

```
{
    lệnh;

    {
        lệnh;
        ...
    }
    ...
}
```

C cho phép khai báo biến trong lệnh khối. Ràng buộc duy nhất là phần khai báo phải nằm trước phần câu lệnh.

*Ví dụ:*

```
#include <conio.h>
#include <stdio.h>
void main()
// Nội dung của hàm main() cũng là một khối lệnh
{
    // khai báo biến
    int c;
    c = 10;
    printf(" Gia trị của c = %d đây là c ngoài",c);
    // bắt đầu một khối lệnh khác
    {
        int c;
        c = 10;
        printf("\n Gia trị của c = %d đây là c trong",c);
        printf("\n Tăng giá trị của c thêm 10 đơn vị");
        c = c + 10;
        printf("\n Gia trị của c = %d đây là c trong",c);
    }
    printf("\n Gia trị của c = %d đây là c ngoài",c);
    getch();
}
```

Kết quả:

```
Gia trị của c = 10 đây là c ngoài
Gia trị của c = 10 đây là c trong
Tăng giá trị của c thêm 10 đơn vị
Gia trị của c = 20 đây là c trong
Gia trị của c = 10 đây là c ngoài
```

### III.3.3. Cấu trúc if

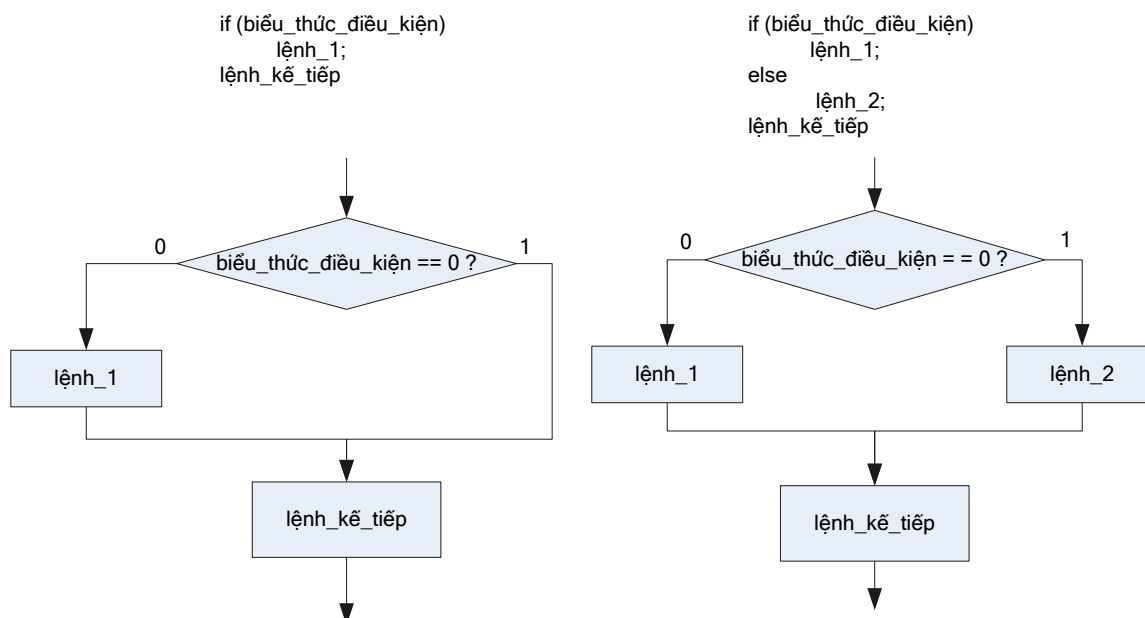
Cú pháp cấu trúc **if**

```
if (biểu_thức_điều_kiện)  
    lệnh;
```

Cú pháp cấu trúc **if ... else**

```
if (biểu_thức_điều_kiện)  
    lệnh_1;  
  
else  
    lệnh_2;
```

Lưu ý là lệnh, lệnh\_1 và lệnh\_2 có thể là lệnh khối.



*Ví dụ:* Bài toán tìm số lớn nhất trong 2 số thực  $a$  và  $b$ . Cách làm là ta 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 báo biến
    float a, b;
    float max;
    printf("Nhập giá trị a và 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
```

## BUỔI 11.

### III.3.4. Cấu trúc lựa chọn switch

Cú pháp cấu trúc **switch**

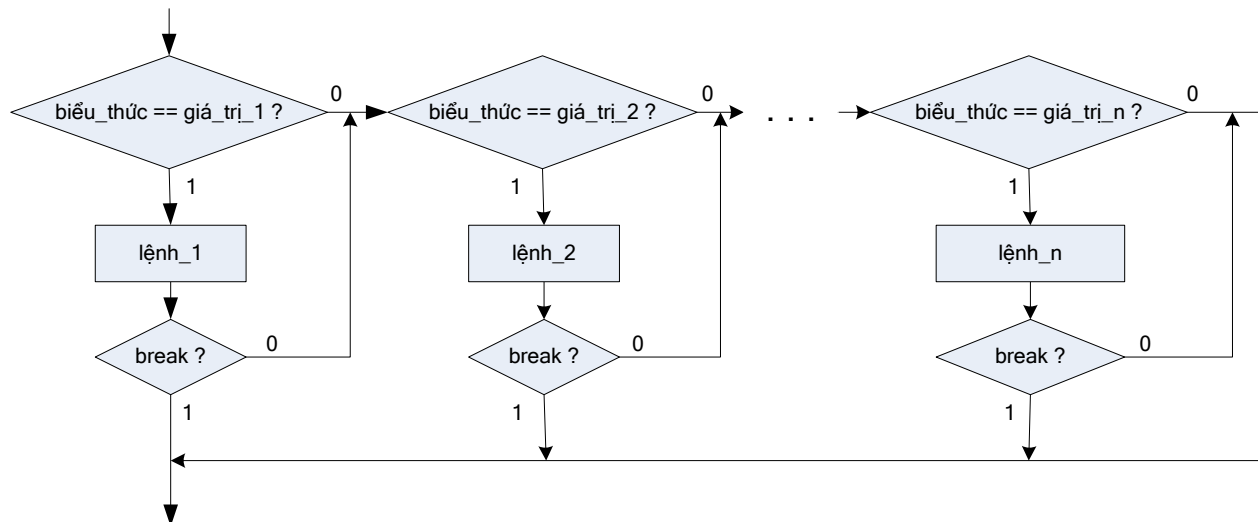
```
switch (biểu_thức)
{
    case giá_trị_1:      lệnh_1; [break;]
    case giá_trị_2:      lệnh_2; [break;]
    ...
    case giá_trị_n:      lệnh_n; [break;]
    [default:           lệnh_n+1; [break;]]
}
```

*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 2 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 2 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:*



*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. Và 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 thangs 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();
return 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

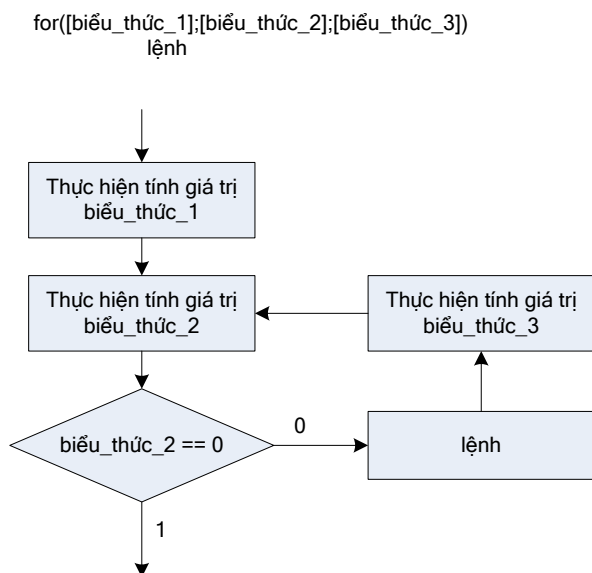
Dạng thường gặp của vòng lặp **for** là

```

for([biểu_thức_1];[biểu_thức_2];[biểu_thức_3])
    lệnh;

```

Sơ đồ



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  $i \leq 9$  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 dành 5 vị trí để in mỗi số
    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 phải 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  $i \leq 199$  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("Tổng của 100 số nguyên dương lẻ đầu tiên là %d", S);
}
```

```
    getch();  
}
```

Kết quả thực hiện:

Tổng của 100 số nguyên dương le đầu tiên là 10000

## Bài tập về Lập trình (2 tiết BT)

...

## BUỔI 12.

### III.3.6. Vòng lặp while và do – while

*Cú pháp vòng lặp while*

**while** (*biểu\_thức*)

*lệnh*;

*Cú pháp vòng lặp do {...} while*

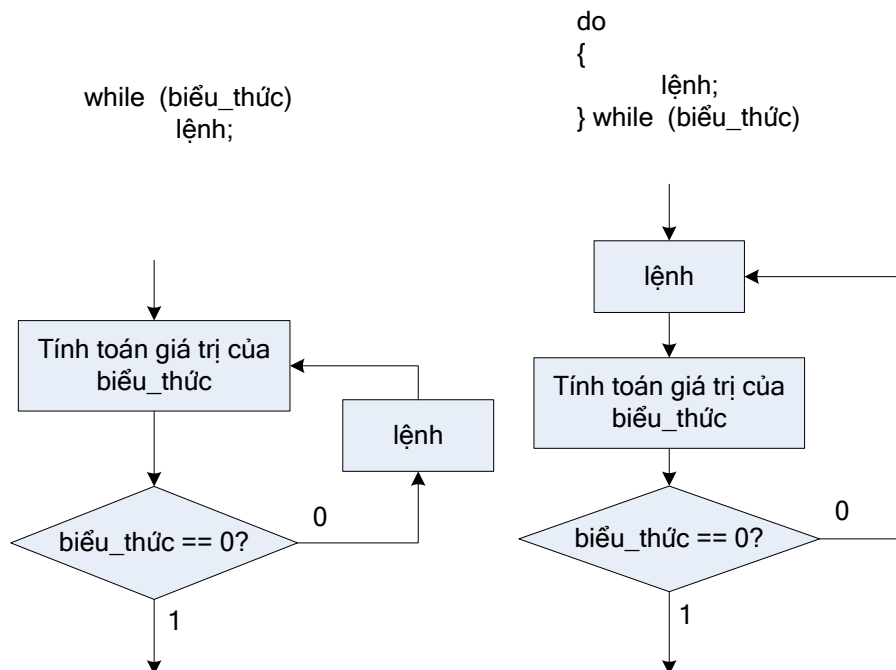
*do*

{

*lệnh*;

*}while* (*biểu\_thức*);

Sơ đồ vòng lặp **while** và **do{...}while**



#### 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, và 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, và 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 2 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 1 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");
    else
        printf("\n %5ld KHONG LA so hoan thien");
    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);

```

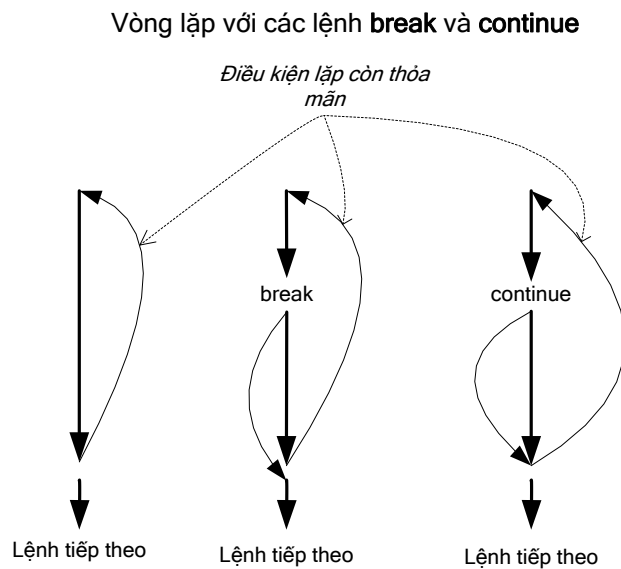
```
tong = tong + i;
    }
    printf("\n Tong cac uoc so cua %ld bang %ld",n,tong);
    if(tong == n)
        printf("\n %5ld LA so hoan thien");
    else
        printf("\n %5ld KHONG LA so hoan thien");
    printf("\n Ban co muon thuc hien lai(c/k)? ");
    fflush(stdin);
    scanf("%c",&ch);
}
printf("\n 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
Cac uoc so cua 12 la: 1  2  3  4  6
Tong cac uoc so cua 12 bang 16
12 KHONG LA so hoan thien
Ban co muon thuc hien lai(c/k)? c
Nhap vao mot so nguyen: 28
Cac uoc so cua %ld la: 1  2  4  7  14
Tong cac uoc so cua 28 bang 28
28 LA so hoan thien
Ban co muon thuc hien lai(c/k)? k
An phim bat ki de ket thuc ...
```

### 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 2 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

```
1  3  5  7  9 11 13 15 17 19
21 23 25 27 29 31 33 35 37 39
41 43 45 47 49 51 53 55 57 59
61 63 65 67 69 71 73 75 77 79
```



81 83 85 87 89 91 93 95 97 99

### 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 Nhập vào 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

```
Nhap vao mot ki tu: a
Ki tu vua nhap vao la a
Nhap vao mot ki tu: 5
Ki tu vua nhap vao la 5
Nhap vao mot ki tu: t
Ki tu vua nhap vao la t
An phim bat ki de ket thuc chuong trinh...
```

### III.4. Mảng và xâu ký tự (5 tiết LT)

#### 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 báo mảng 10 phần tử có kiểu dữ liệu int
float mang_thuc[4];  // khai báo mảng 4 phần tử có kiểu dữ liệu float
char mang_ki_tu[6];  // khai báo mảng 6 phần tử có kiểu dữ liệu char
```

Trong ví dụ trên, mảng *mang\_nguyen* được lưu trữ trên 20 ô nhớ (mỗi ô nhớ có kích thước 1 byte, 2 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 2 là *mang\_nguyen[1]*, phần tử thứ 5 là *mang\_nguyen[4]*...

mang_nguyen[0]	mang_nguyen[1]	...	...	mang_nguyen[9]
----------------	----------------	-----	-----	----------------

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 6 phần tử, trong đó mỗi phần tử lại là một mảng gồm 5 số nguyên kiểu **int**. Mảng này là mảng 2 chiều

```
int a [6][5];
```

Còn khai báo

```
int b [3][4][5];
```

thì lại khai báo một mảng gồm 3 phần tử, mỗi phần tử lại một mảng 2 chiều gồm 4 phần tử. Mỗi phần tử của mảng 2 chiều lại là một mảng (1 chiều) gồm 5 số nguyên kiểu **int**. Mảng b ở trên được gọi là một mảng 3 chiều.

### *Sử dụng mảng*

Để truy nhập vào một phần tử của mảng thông qua 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 2 của mảng

`mang_nguyen[2]` sẽ là phần tử thứ 3 của mảng

Với mảng nhiều chiều như

```
int a[6][5];
```

thì `a[0]` là phần tử đầu tiên của một 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 4 của phần tử thứ 3 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]);  
// Gán giá trị cho phần tử a[2]  
a[2] = a[1] + 5;
```

Nếu ta 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;  
// Nhập giá trị từ bàn phím cho tất cả các phần tử của mảng b  
for(i = 0; i < 10; i++)  
{  
    printf("\n Nhập giá trị 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 báo mảng, giả sử số phần tử tối đa của a là 100  
int n;        // biến lưu giữ số phần tử thực sự của mảng  
int i;  
printf("\n Cho biết số phần tử của mảng: ");  
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 4 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;
// Nhập giá trị cho các phần tử của mảng a từ bàn phím
for(i = 0; i < 5; i++)
{
    printf("\n a[%d] = ", i);
    scanf("%d", &a[i]);
}
// Hiện thị giá trị của phần tử bất kỳ, giả sử a[3] lên màn hình
printf("\n a[3] = %d", a[3]);
// Hiện thị giá trị của tất cả các phần tử, mỗi phần tử trên một dòng
for(i = 0; i < 5; i++)
    printf("\n%d", a[i]);
// Hiện thị giá trị của tất cả các phần tử trên cùng một dòng, các giá trị cách nhau 2 vị trí
printf("\n"); // Xuống dòng mới
for(i = 0; i < 5; i++)
    printf("%d ", a[i]);
// Hiện thị giá trị của tất cả các phần tử, trong đó k phần tử trên một dòng.
// Các phần tử trên cùng dòng cách nhau 2 vị trí
printf("\n Cho biết giá trị của k = ");
scanf("%d",&k);
for(i = 0; i < 5; i++)
{
    printf("%d ",a[i]);
    if((i+1)%k == 0) // đã hiện thị đủ k phần tử trên một dòng thì phải xuống
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 biết giá trị của k = 2
6 14

```

### *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 ta 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 ta sẽ coi phần tử này là phần tử lớn nhất rồi chuyển sang so sánh với phần tử tiếp theo. Sau khi so sánh với phần tử cuối cùng thì ta sẽ tìm được phần tử lớn nhất trong 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.

## **BUỔI 13.**

### **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 ta 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 là số phần tử trong mảng
    int i, k, kiem_tra;
    clrscr(); // xóa màn hình để tiện theo dõi
    // Nhập giá trị dữ liệu cho mảng m
    // Trước tiên phải biết số phần tử của mảng
    printf(" Cho biet so phan tu co trong mang: ");
    scanf("%d",&n);
    // Rồi lần lượt nhập giá trị cho các phần tử trong mảng
    for(i = 0;i<n;i++)
    {
        int temp;
        printf("\n Cho biet gia tri cua m[%d] = ",i);
        scanf("%d",&temp);
        m[i] = temp;
    }
    // Yêu cầu người sử dụng nhập vào giá trị cho dữ liệu k
    printf("\n Cho biết giá trị của dữ liệu k: ");
    scanf("%d",&k);
    // Bắt đầu quá trình tìm kiếm
    kiem_tra = 0;
    // Duyệt qua tất cả các phần tử
    for(i = 0;i<n;i++)
        if(m[i] == k)//So sánh phần tử đang xét với dữ liệu k
        {
            // Ghi nhận chỉ số của phần tử đang xét
            chi_so[kiem_tra] = i;
            kiem_tra++; //Tăng biến kiem_tra thêm 1 đơn vị
        }
    // Kết luận
    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();      // Chờ người sử dụng ấn phím bất kì để kết thúc.
}
```

### 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 như sau 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  $m[1], m[2] \dots m[n-1]$ ). Nếu có giá trị  $m[i]$  nào đó ( $i = 1, 2, \dots, n-1$ ) 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 2 ta so sánh phần tử thứ 2 của mảng  $m[1]$  với tất cả các phần tử đứng sau nó trong mảng (tức là các phần tử  $m[2], m[3] \dots m[n-1]$ ). Nếu có giá trị  $m[i]$  nào đó ( $i = 2, 3, \dots, n-1$ ) nhỏ hơn  $m[1]$  thì ta hoán đổi giá trị giữa  $m[1]$  và  $m[i]$  cho nhau. Sau lượt sắp xếp thứ 2 thì  $m[1]$  sẽ là phần tử có giá trị nhỏ thứ 2 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  $m[k], m[k+1] \dots m[n-1]$ ). Nếu có giá trị  $m[i]$  nào đó ( $i = k, k+1, \dots, n-1$ ) 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 là số phần tử trong mảng
    int i, j, k;
    clrscr(); // xóa màn hình để tiện theo dõi
    // Nhập giá trị dữ liệu cho mảng m
    // Trước tiên phải biết số phần tử của mảng
    printf(" Cho biet so phan tu co trong mang: ");
```



```

scanf("%d",&n);
// Rồi lần lượt nhập giá trị cho các phần tử trong mảng
for(i = 0;i<n;i++)
{
    int temp;
    printf("\n Cho biet gia tri cua m[%d] = ",i);
    scanf("%d",&temp);
    m[i] = temp;
}
// Hiển thị ra màn hình mảng vừa nhập vào
printf("\n Mang truoc khi sap xep: ");
for(i=0;i<n;i++)
    printf("%3d",m[i]);
// Bắt đầu sắp xếp
for(i = 0; i<n-1;i++)
{
    // Ở lượt sắp xếp thứ i+1
    for(j = i+1;j<n;j++)
    {
        // So sánh m[i] với các phần tử còn lại
        // và đổi chỗ khi tìm thấy phần tử < m[i].
        if(m[j]<m[i])
        {
            int temp;
            temp = m[j]; m[j] = m[i]; m[i] = temp;
        }
    }
    // Hiển thị mảng sau lượt sắp xếp thứ i+1
    printf("\n Mang o luot sap xep thu %d",i+1);
    for(k = 0;k < n ;k++)
        printf("%3d",m[k]);
}
getch();    // Chờ người sử dụng ấn phím bất kì để kết thúc.
}

```

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

## Bài tập về Lập trình (3 tiết BT)

...

## BUỔI 14.

### III.4.2. Xâu ký tự (2 tiết LT)

#### III.4.2.1. Khái niệm xâu ký tự

Xâu ký tự (*string*) là một dãy các ký tự viết liên tiếp nhau.

Xâu rỗng: là xâu không gồm ký tự nào cả.

Độ dài xâu là số ký tự có trong xâu.

Biểu diễn xâu ký tự: xâu ký tự được biểu diễn bởi dãy các ký tự đặt trong cặp dấu ngoặc kép. Các ký tự nằm trong cặp dấu ngoặc kép là nội dung của xâu.

Ví dụ:

- “String” là một xâu ký tự gồm 6 ký tự: ‘S’, ‘t’, ‘r’, ‘i’, ‘n’, ‘g’ được viết liên tiếp nhau.
- “Tin hoc” là một xâu ký tự gồm 7 ký tự: ‘T’, ‘i’, ‘n’, dấu cách (‘ ’), ‘h’, ‘o’, và ‘c’.

Lưu trữ dữ liệu kiểu xâu ký tự: các ký tự của xâu được lưu trữ kế tiếp nhau và kết thúc bằng ký tự kết thúc xâu (ký tự ‘\0’ hay ký tự NUL, có số thứ tự 0 trong bảng mã ASCII). Nhờ có ký tự NUL mà người ta xác định được độ dài của xâu ký tự bằng cách đếm các ký tự có trong xâu đến khi gặp ký tự NUL (ký tự NUL không được tính vào độ dài xâu).

Ví dụ xâu ký tự “Tin hoc” sẽ được lưu trữ như sau

‘T’	‘i’	‘n’	‘ ’	‘h’	‘o’	‘c’	‘\0’
-----	-----	-----	-----	-----	-----	-----	------

Lưu ý:

- Xâu ký tự khác mảng ký tự ở chỗ xâu ký tự có ký tự kết thúc xâu (ký tự NUL hay ‘\0’) trong khi mảng ký tự không có ký tự kết thúc.
- Phân biệt giữa một ký tự và xâu ký tự có một ký tự: ví dụ ‘A’ là một ký tự, nó được lưu trữ trong 1 byte, còn “A” là xâu ký tự, nó được lưu trữ trong 2 bytes, trong đó byte đầu tiên lưu trữ ký tự ‘A’, byte thứ 2 lưu trữ ký tự kết thúc xâu (NUL).

### III.4.2.2. Khai báo và sử dụng chuỗi

#### a. Khai báo chuỗi ký tự

Trong C, một chuỗi ký 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 ký tự nhiều nhất có thể có trong chuỗi.

Sau khi khai báo, biến chuỗi ký tự tên\_xâu có thể được dùng để lưu trữ một chuỗi ký tự bất kỳ, miễn là độ dài chuỗi ký tự (số ký tự có trong chuỗi) 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 chuỗi ký tự tên là ho\_va\_ten, biến này có thể có tối đa 20 ký tự.

Lưu ý: Đôi khi ta vẫn có thể nhập một chuỗi có nhiều hơn 20 ký tự cho chuỗi 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 lý bộ nhớ của hệ điều hành sẽ bắt lỗi và buộc chương trình kết thúc.

#### b. Truy nhập vào một phần tử của chuỗi

Có thể truy nhập đến từng phần tử của chuỗi 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 chuỗi que\_quan có nội dung là "Ha Noi". Khi đó ta có thể hình dung chuỗi ký 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.4.2.3. 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.4.2.4. Các hàm xử lý chuỗi

#### a. Vào ra dữ liệu

Vào ra dữ liệu trên chuỗi ký tự tức là nhập dữ liệu cho chuỗi và hiển thị dữ liệu chứa trong chuỗi.

Để nhập dữ liệu cho chuỗi ta có thể sử dụng 2 hàm `scanf()` hoặc `gets()`

**`scanf("%s",tên_xâu);`**

**`gets(tên_xâu);`**

Để hiển thị nội dung chuỗi ta có thể dùng 2 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;
}
```

#### b. Một số hàm xử lý chuỗi ký tự khác

Hàm `strlen()`

**`size_t strlen(char* tên_xâu);`**

Hàm trả về độ dài (số ký tự có trong chuỗi) của chuỗi ký tự `tên_xâu`.

Hàm `strcpy()`

**`char* strcpy(char* chuỗi_đích, char* chuỗi_nguồn)`**

Hàm này sẽ sao chép nội dung chuỗi\_nguồn và ghi lên chuỗi\_đích.

Hàm `strcmp()`

**`int strcmp(char* chuỗi_thứ_nhất, char* chuỗi_thứ_hai);`**

Hàm `strcmp` trả về

- giá trị < 0 nếu chuỗi\_thứ\_nhất nhỏ hơn chuỗi\_thứ\_hai
- giá trị 0 nếu chuỗi\_thứ\_nhất bằng chuỗi\_thứ\_hai
- giá trị > 0 nếu chuỗi\_thứ\_nhất lớn hơn chuỗi\_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 2 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 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 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 phẩy thập 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>    // Phải có thư viện string.h thì mới
                      // sử dụng được các hàm 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 (2 tiết LT)

#### 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

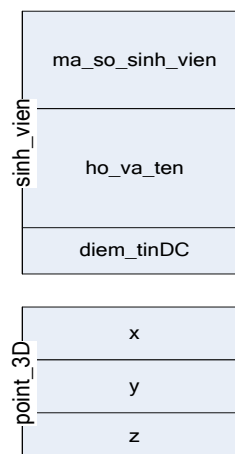
#### III.5.2.1. Khai báo kiểu dữ liệu cấu trúc

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 3 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 2 ta đã khai báo một kiểu dữ liệu cấu trúc có tên là `point_3D` gồm có 3 trường và cả 3 trường này đều có cùng kiểu dữ liệu số thực **float**. Vì 3 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 3 chiều sẽ giúp sự mô tả được tự nhiên hơn, “thật” hơn (nghĩa là 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. Còn nếu sử dụng mảng thì các tọa độ của một điểm độc lập và rời rạc với nhau, không thấy mối liên hệ. 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  $k = 0, 1, 2, \dots$  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 3 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];           // số báo 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;
```

### III.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 **struct**, 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 lập trình quên đặt từ khóa **struct** ở đầu. Để tránh

điều đó, ngôn ngữ C cho phép ta đặt tên mới cho kiểu dữ liệu cấu trúc bằng câu lệnh **typedef** có cú pháp khai báo như 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. Do đó 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, và nếu như 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 “đổi 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:*

// dưới đây là một cấu trúc mô tả một điểm trong không gian 2 chiều.

// các trường dữ liệu gồm: tên của điểm và tọa độ của điểm đó.

// tọa độ là một cấu trúc gồm 2 trường: hoành độ và tung độ

```
#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;
```

```
    // giả sử điểm đang xét nằm trên đường thẳng  $y = 3x + 2$ ;
```

```
    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

nhập vào hoành độ của điểm: 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 2 biến cấu trúc là *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 như cách thứ 2, 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 copy dữ liệu từ biến a sang biến c bằng cách sao chép từng trường, và copy dữ liệu từ biến a sang biến b bằng cách dùng lệnh gán. Kết quả là như nhau và rõ ràng cách thứ 2 ngắn gọn hơn.

*Lưu ý* : Để copy dữ liệu là chuỗi ký tự ta phải dùng lệnh strcpy(), không được dùng lệnh gán thông thường để copy nội dung chuỗi ký tự.

## BUỔI 15.

### III.6. Hàm (2 tiết LT)

#### 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 công sức, những đoạn chương trình đó được thay thế bởi 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 đại lượng 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 đại lượng  $y$  nào đó thì ta chỉ cần gọi chương trình con sin đã lập sẵn và truyền đại lượng  $y$  làm tham số cho chương trình con sin đó thì 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 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 module 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 module 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 2 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 gì cả.

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

### III.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 2 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>**

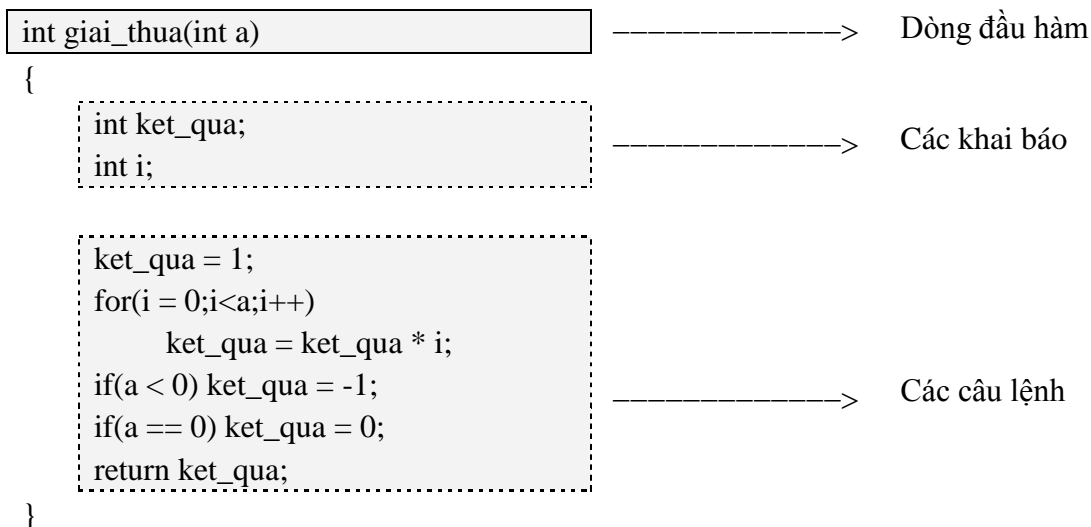
**...**

### <Các câu lệnh>

}

Trong thân hàm có ít nhất một lệnh return.

Ví dụ sau là 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 0, của một số nguyên dương  $a$  là  $a! = a \times (a-1) \times \dots \times 1$ .



### 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 ta 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 2 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 kì (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 *bình\_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 của hàm phải chỉ ra tên của tham số và kiểu dữ liệu của tham số.

Các tham số được cung cấp cho hàm trong quá trình thực hiện của 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, 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*

Trong chương trình, 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 nó. Có 2 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.

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 và phân tích.

```
#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 Nhập 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");
        printf("\n Gia tri lon nhat trong 3 so la %d",max(a,b,c));
        getch();
    }

```

### III.6.2.2. Sử dụng hàm

Một hàm sau khi khai báo thì có thể sử dụng. 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 2 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 Nhập vào 2 số nguyên dương a và b ");
printf("\n a = "); scanf("%d",&a);
printf("\n b = "); scanf("%d",&b);
if(a*b == 0)
{
    printf("\n Không hợp lệ");
    continue;
}
printf("\n Ước chung lớn nhất của %d và %d là: %d", a, b, Uscln(a, b));
} while ((a != 0) || (b != 0));
printf("\n Ấn phím bất kỳ để kết thúc chương trình...");
getch();
return 0;
}
```

Kết quả khi thực hiện:

```
Nhập vào 2 số nguyên dương a và b
a = 6
b = 9
Ước chung lớn nhất của 6 và 9 là: 3
Nhập vào 2 số nguyên dương a và b
a = 15
b = 26
Ước chung lớn nhất của 15 và 26 là: 1
Nhập vào 2 số nguyên dương a và b
a = 3
b = 0
Không hợp lệ
Nhập vào 2 số nguyên dương a và b
a = 0
b = 0
Không hợp lệ
Ấn phím bất kỳ để kết thúc chương trình...
```

*Lưu ý:*

- Nếu có nhiều tham số trong danh sách tham số thì các tham số được phân cách với nhau bằng dấu phẩy
- Cho dù hàm có một, nhiều hay không có tham số thì vẫn luôn luôn cần cặp dấu ngoặc đơn đứng sau tên hàm

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. 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 tùy vào vị trí khai báo biến mà phạm vi sử dụng các biến sẽ khác nhau. Nguyên tắc sử 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 hoặc trong các chương trình con hoặc thậm chí trong một lệnh khối. Nếu biến được khai báo trong một lệnh khối thì nó chỉ có thể được gọi trong lệnh khối đó thôi, không thể gọi từ bên ngoài lệnh khối được. Một biến được khai báo trong một chương trình con chỉ có thể được sử dụng trong phạm vi chương trình con đó. Một biến được khai báo trong chương trình chính thì có thể được sử dụng trong toàn bộ chương trình, trong tất cả các chương trình con cũng như trong các lệnh khối của chương trình.

#### *Lưu ý*

- Một số ngôn ngữ lập trình như Pascal cho phép khai báo một chương trình con nằm trong một chương trình con khác, nhưng ngôn ngữ C không cho phép khai báo một chương trình con nằm trong một chương trình con khác.
- Bên trong một lệnh khối thì có thể có chứa lệnh khối khác. Khi đó biến được khai báo ở lệnh khối bên ngoài có thể được sử dụng ở lệnh khối bên trong.
- Việc trùng tên của các biến: Trong cùng một phạm vi ta không được phép khai báo 2 biến có cùng tên nhưng ta có thể khai báo 2 biến trùng tên thuộc 2 phạm vi khác nhau. Nếu có 2 biến trùng tên khai báo ở 2 phạm vi khác nhau thì xảy ra 2 trường hợp:
  - Hai phạm vi này tách rời nhau: khi đó các biến sẽ có tác dụng ở phạm vi riêng của nó, không ảnh hưởng đến nhau.
  - Phạm vi này nằm trong phạm vi kia: khi đó nếu chương trình đang ở phạm vi 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 2 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 lệnh khởi hoặc trong thân chương trình con. Việc khai báo các biến cục bộ phải được đặt trước phần câu lệnh trong lệnh khởi 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

```
Day la lan goi ham fct lan thu 1
Day la lan goi ham fct lan thu 2
Day la lan goi ham fct lan thu 3
Day la lan goi ham fct lan thu 4
Day la lan goi ham fct lan thu 5
Day la lan goi ham fct lan thu 6
Day la lan goi ham fct lan thu 7
Day la lan goi ham fct lan thu 8
Day la lan goi ham fct lan thu 9
Day la lan goi ham fct lan thu 10
```

**Bài tập về Lập trình (2 tiết BT)**

...