Tin học đại cương

Bài 2: Hệ thống máy tính

NGUYỄN Thị Oanh

oanhnt@soict.hut.edu.vn

Bộ môn Hệ thống thông tin - Viện CNTT và Truyền Thông
Đại học Bách Khoa Hà nội

2010 - 2011

Nội dung

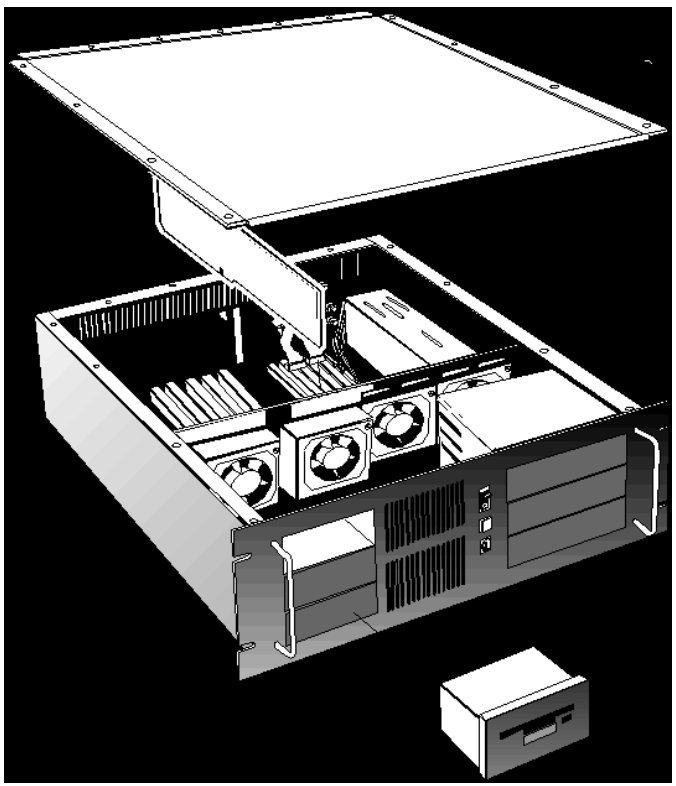
Tổng quan về hệ thống máy tính

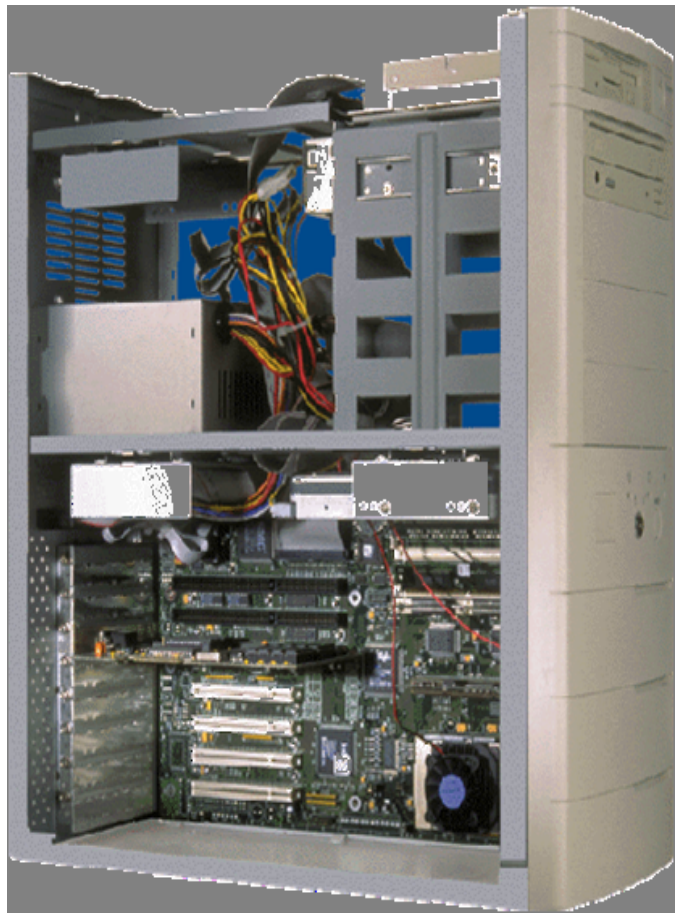
Máy tính gồm 2 thành phần cơ bản:

- Phần cứng: toàn bộ máy móc, thiết bị vật lý cấu tạo máy tính
- Phần mềm: là chương trình chạy trên máy tính

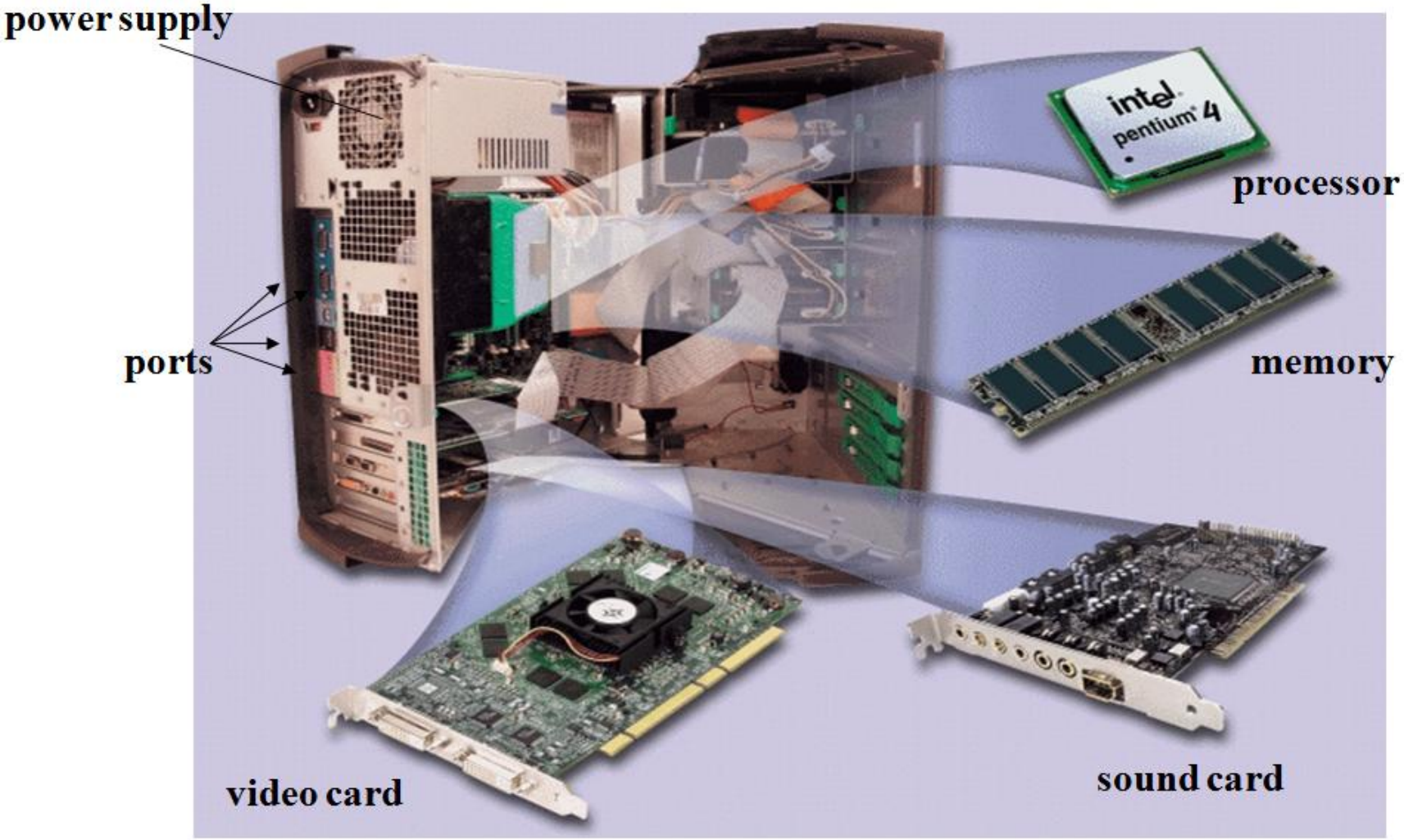
Phần cứng

Tháo ốc vít $\rightarrow$ mở nắp hộp



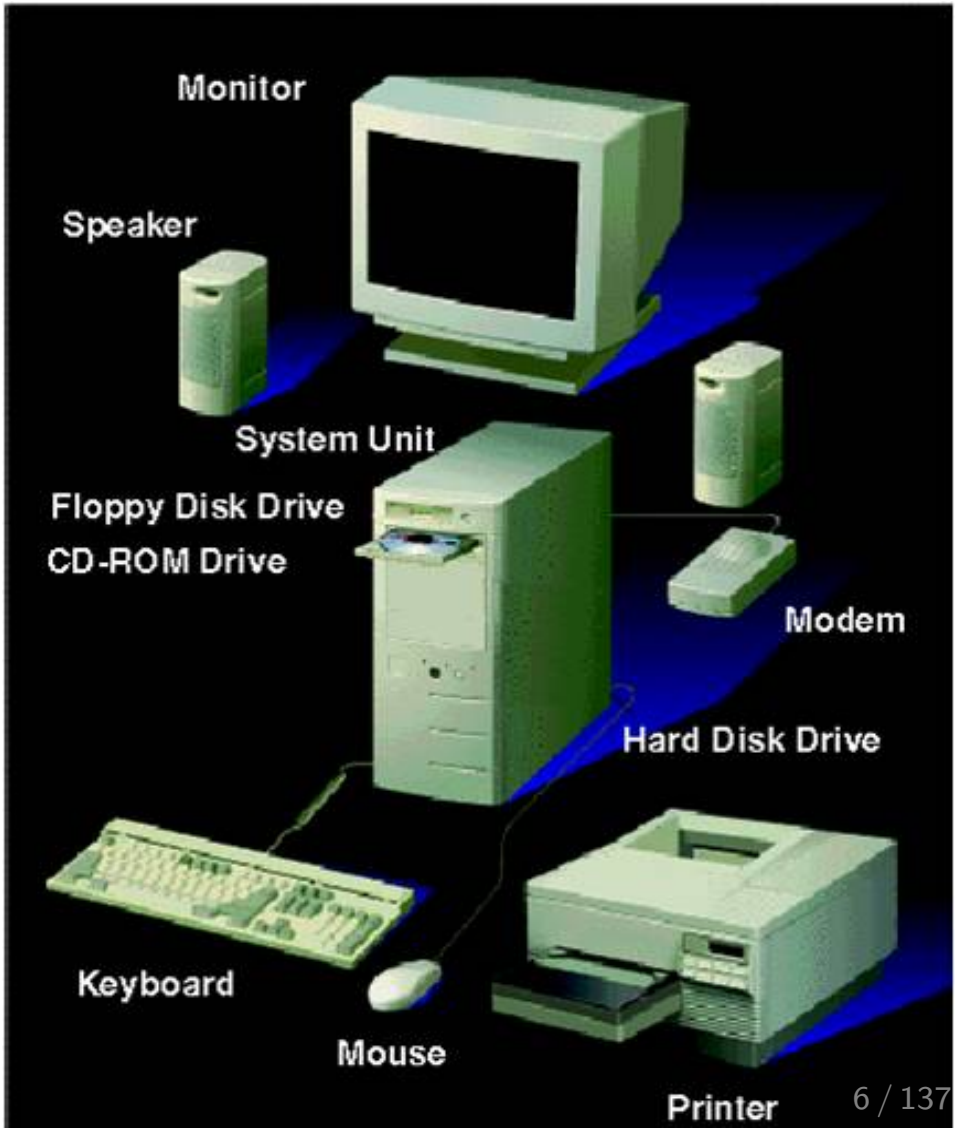


Phần cứng


power supply
ports
intel
pentium 4
processor
memory
video card
sound card

Phần cứng

- Phần cứng:
  - Màn hình
  - Loa
  - Bàn phím
  - Chuột
  - CPU
  - ...



Phần mềm

- Hệ điều hành:

- Ứng dụng:

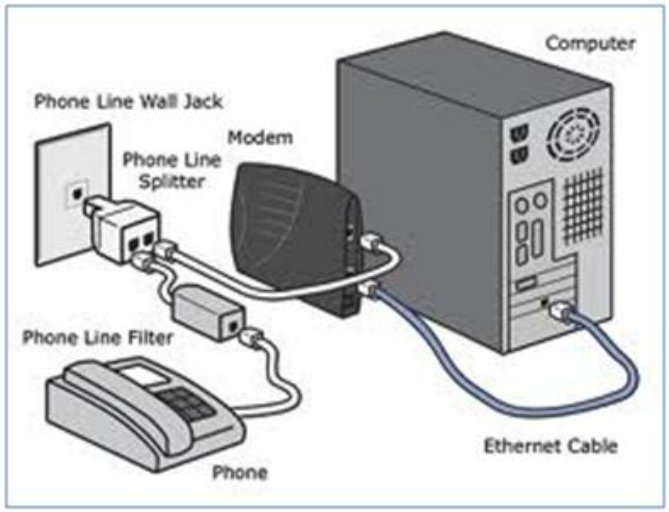
Hệ điều hành

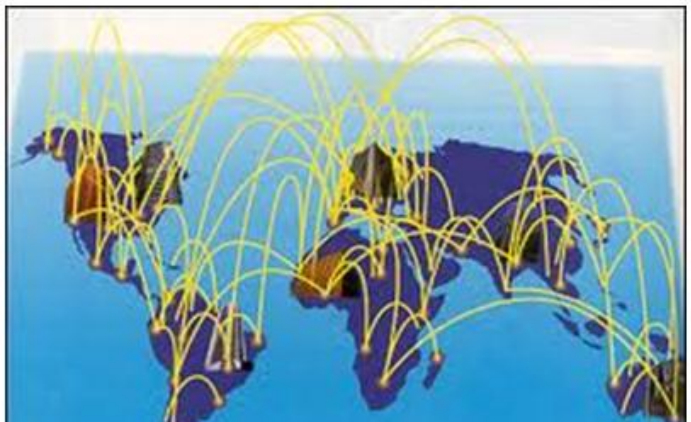
Commercial

Free

Mạng máy tính

Các máy tính được kết nối với nhau để tạo thành mạng máy tính





Các vấn đề sẽ đề cập

- Chức năng cơ bản của máy tính
- Cấu trúc máy tính
- Hoạt động của máy tính

Chức năng cơ bản của máy tính

- Ví dụ:
  - Cuối mỗi học kỳ, phòng Đào tạo *nhập* điểm thi của SV vào máy tính, *tính toán* kết quả học tập của SV, *in ra* danh sách SV đạt học bổng
  - Máy tính thực hiện *play* một bài hát, bộ phim được *lưu trữ* trong máy tính. Bài hát hay bộ phim này có thể được *sao chép* cho người khác
  - ...
- Chức năng:
  - Xử lý dữ liệu (*tính toán*, *play*, ...)
  - Lưu trữ dữ liệu (*lưu trữ*, ...)
  - Trao đổi dữ liệu (*nhập điểm*, *in ra*, *sao chép*, ...)
  - Điều khiển

Chức năng cơ bản của máy tính(...)

- Xử lý dữ liệu : chức năng quan trọng nhất
  - dữ liệu có thể có rất nhiều dạng khác nhau và có yêu cầu xử lý khác nhau
- Lưu trữ dữ liệu
  - dữ liệu đưa vào máy tính có thể được xử lý ngay hoặc có thể được lưu trong bộ nhớ
  - khi cần chúng sẽ được lấy ra xử lý
- Trao đổi dữ liệu
  - trao đổi dữ liệu giữa các thành phần bên trong và bên ngoài máy tính -> Quá trình vào ra (input-output)
  - các thiết bị vào-ra: nguồn cung cấp hoặc nơi tiếp nhận dữ liệu
  - dữ liệu được vận chuyển trên khoảng cách xa gọi là truyền dữ liệu (data communication)
- Điều khiển: máy tính cần phải điều khiển được 3 chức năng trên

Chức năng cơ bản của máy tính(...)

- Xử lý dữ liệu : chức năng quan trọng nhất
  - dữ liệu có thể có rất nhiều dạng khác nhau và có yêu cầu xử lý khác nhau
- Lưu trữ dữ liệu
  - dữ liệu đưa vào máy tính có thể được xử lý ngay hoặc có thể được lưu trong bộ nhớ
  - khi cần chúng sẽ được lấy ra xử lý
- Trao đổi dữ liệu
  - trao đổi dữ liệu giữa các thành phần bên trong và bên ngoài máy tính -> Quá trình vào ra (input-output)
  - các thiết bị vào-ra: nguồn cung cấp hoặc nơi tiếp nhận dữ liệu
  - dữ liệu được vận chuyển trên khoảng cách xa gọi là truyền dữ liệu (data communication)
- Điều khiển: máy tính cần phải điều khiển được 3 chức năng trên

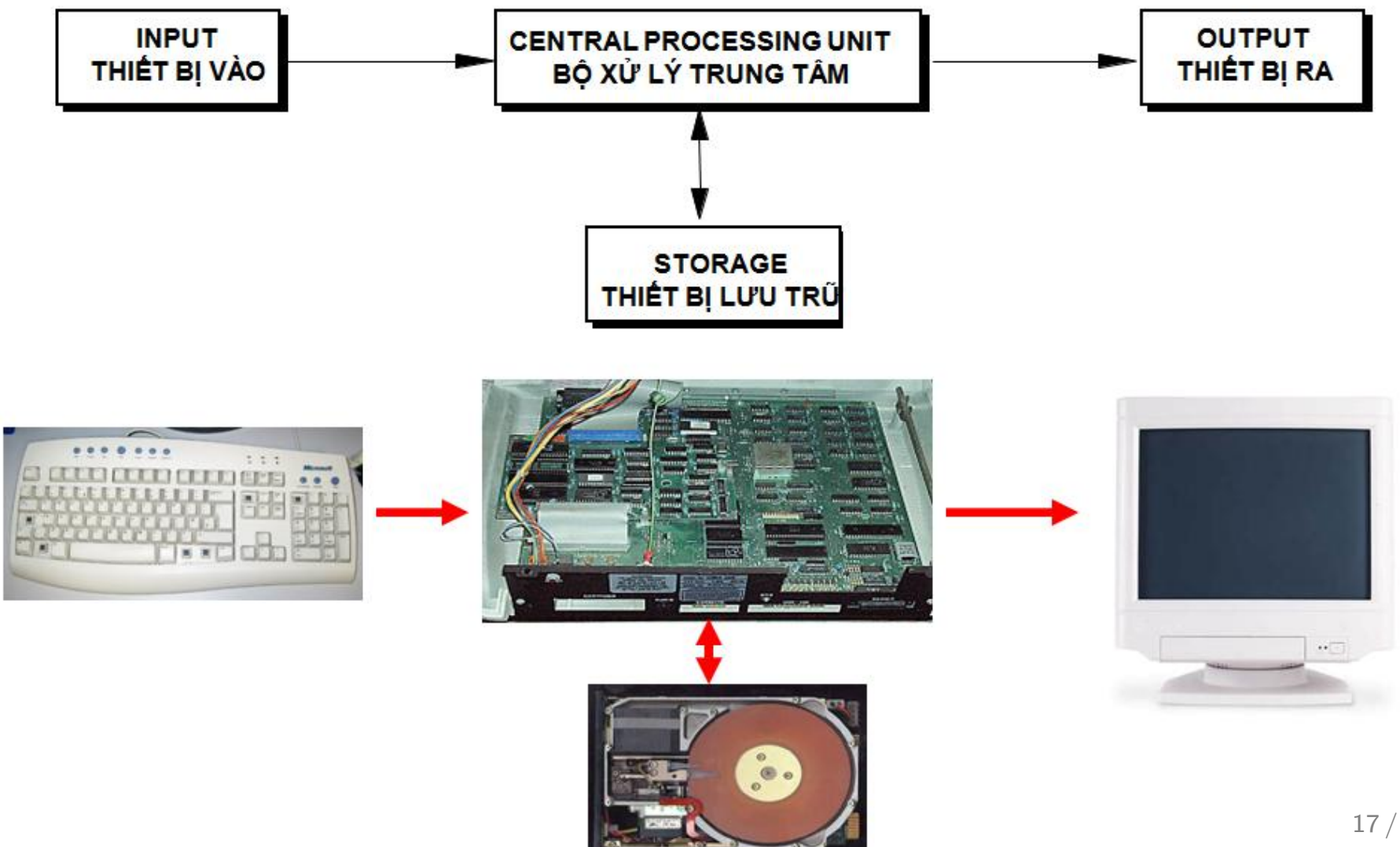
Chức năng cơ bản của máy tính(...)

- Xử lý dữ liệu : chức năng quan trọng nhất
  - dữ liệu có thể có rất nhiều dạng khác nhau và có yêu cầu xử lý khác nhau
- Lưu trữ dữ liệu
  - dữ liệu đưa vào máy tính có thể được xử lý ngay hoặc có thể được lưu trong bộ nhớ
  - khi cần chúng sẽ được lấy ra xử lý
- Trao đổi dữ liệu
  - trao đổi dữ liệu giữa các thành phần bên trong và bên ngoài máy tính -> Quá trình vào ra (input-output)
  - các thiết bị vào-ra: nguồn cung cấp hoặc nơi tiếp nhận dữ liệu
  - dữ liệu được vận chuyển trên khoảng cách xa gọi là truyền dữ liệu (data communication)
- Điều khiển: máy tính cần phải điều khiển được 3 chức năng trên

Chức năng cơ bản của máy tính(...)

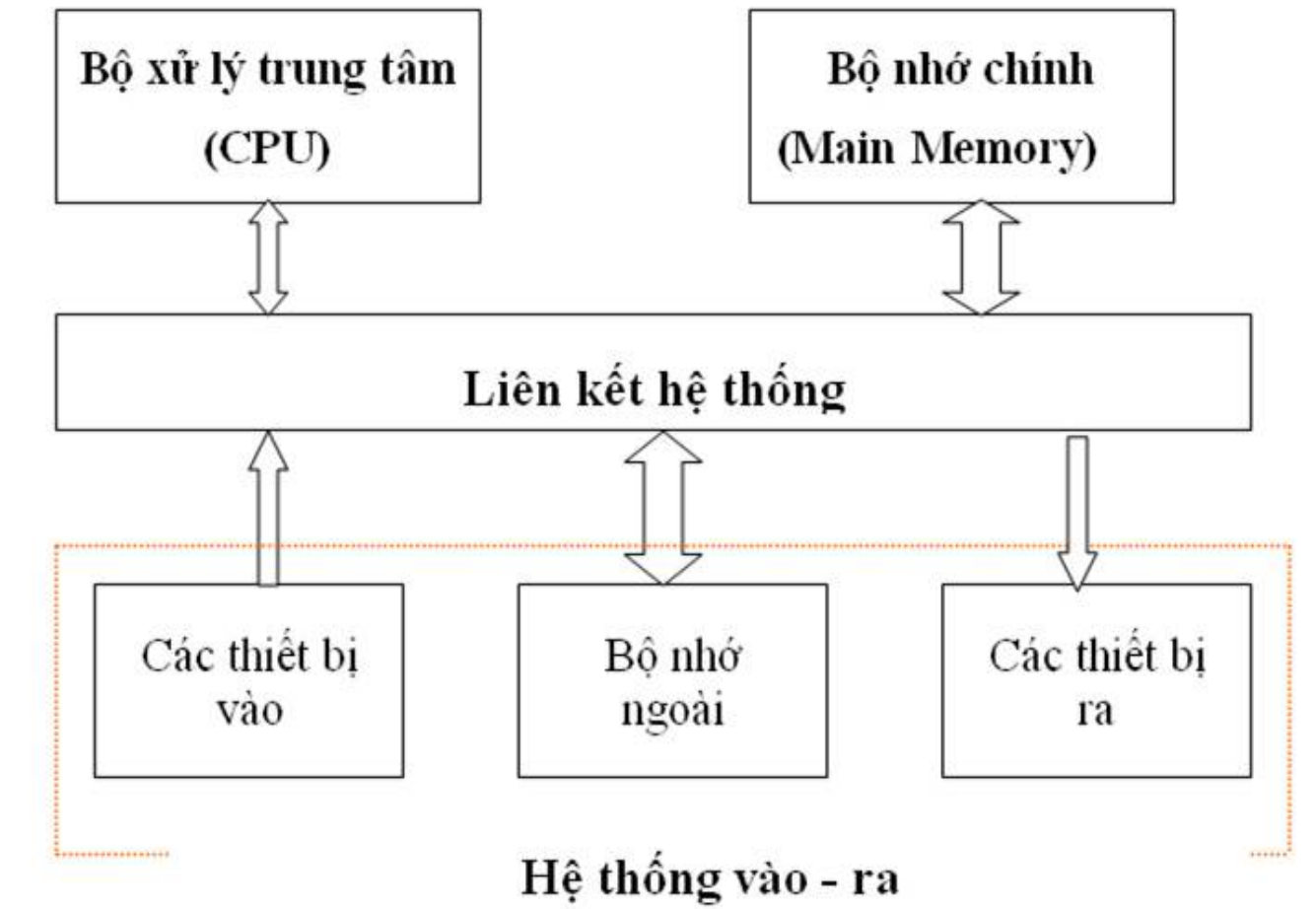
- Xử lý dữ liệu : chức năng quan trọng nhất
  - dữ liệu có thể có rất nhiều dạng khác nhau và có yêu cầu xử lý khác nhau
- Lưu trữ dữ liệu
  - dữ liệu đưa vào máy tính có thể được xử lý ngay hoặc có thể được lưu trong bộ nhớ
  - khi cần chúng sẽ được lấy ra xử lý
- Trao đổi dữ liệu
  - trao đổi dữ liệu giữa các thành phần bên trong và bên ngoài máy tính -> Quá trình vào ra (input-output)
  - các thiết bị vào-ra: nguồn cung cấp hoặc nơi tiếp nhận dữ liệu
  - dữ liệu được vận chuyển trên khoảng cách xa gọi là truyền dữ liệu (data communication)
- Điều khiển: máy tính cần phải điều khiển được 3 chức năng trên

Chức năng cơ bản của máy tính (...)



Cấu trúc máy tính

Các thành phần cơ bản của máy tính



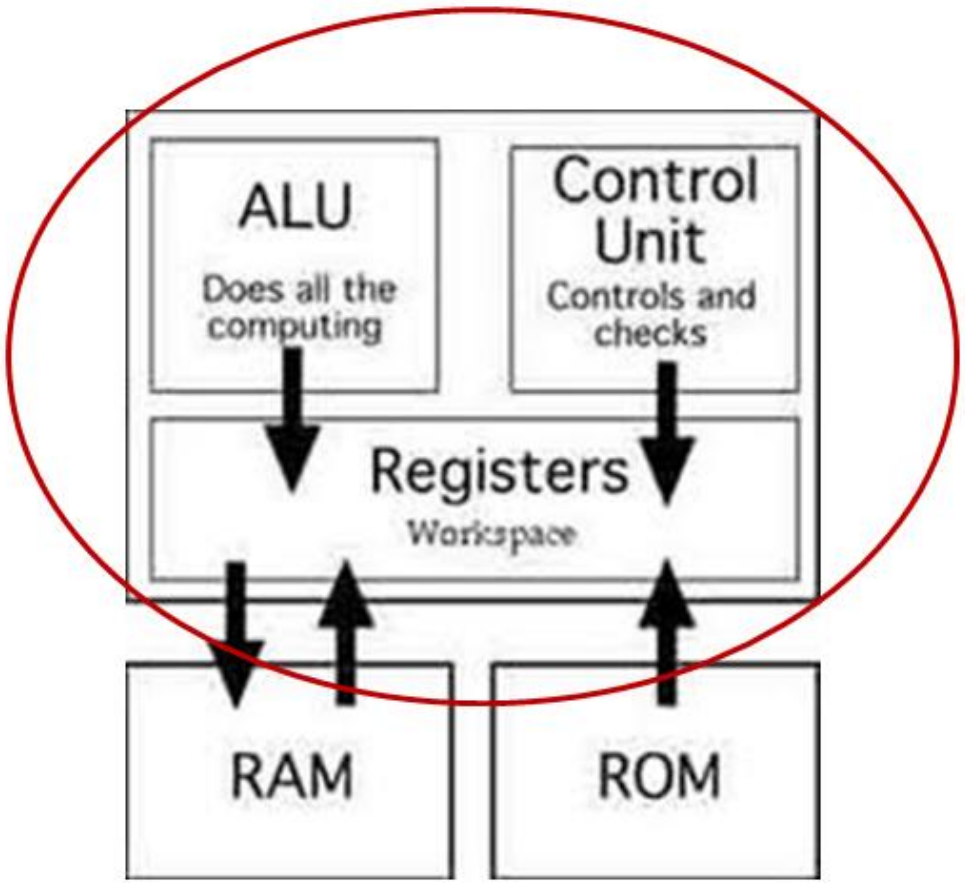
Cấu trúc máy tính(...)

- Bộ xử lý trung tâm – CPU (Central Processor Unit):
  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 máy tính và thế giới bên ngoài
- 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

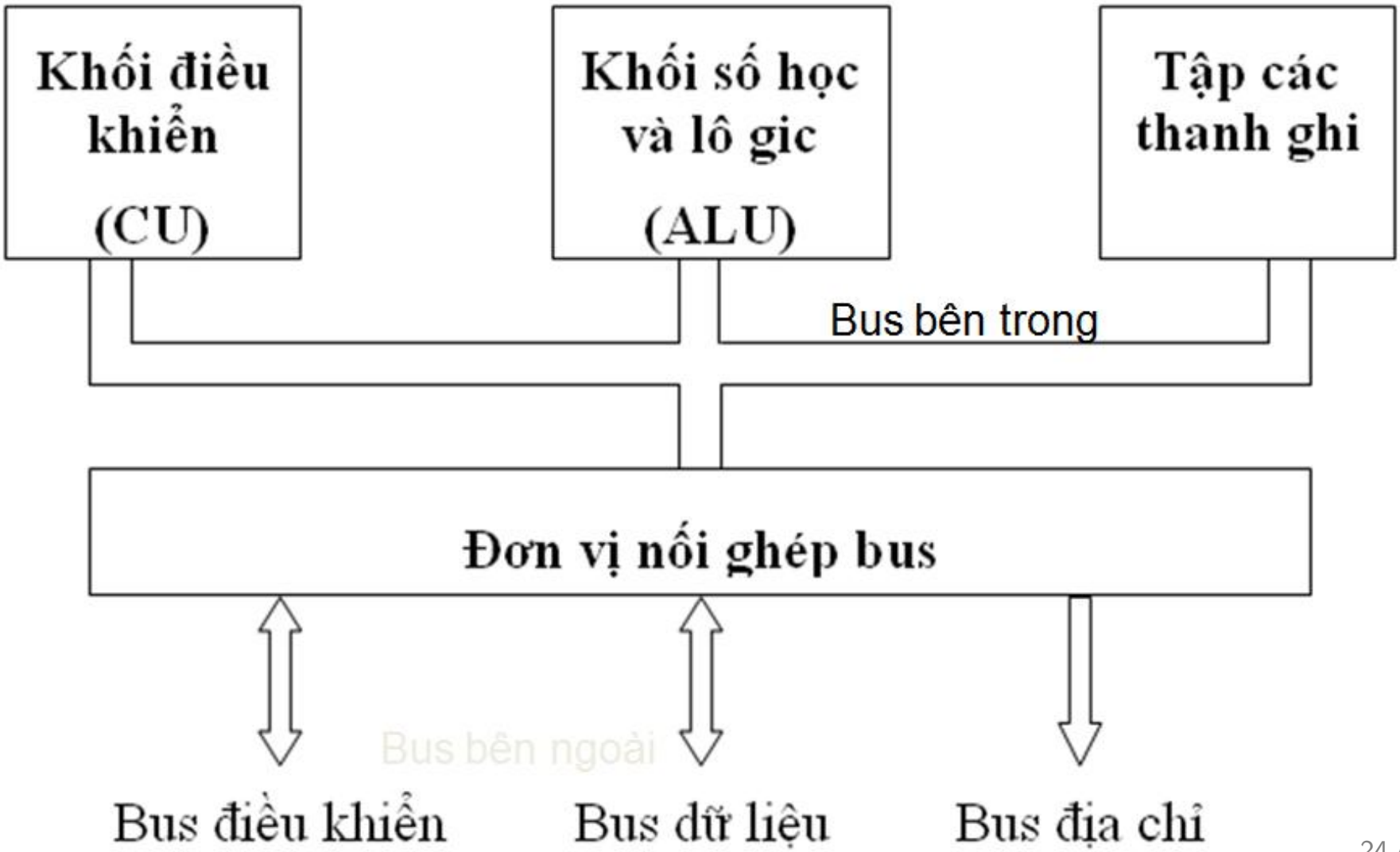
Bộ xử lý trung tâm – CPU



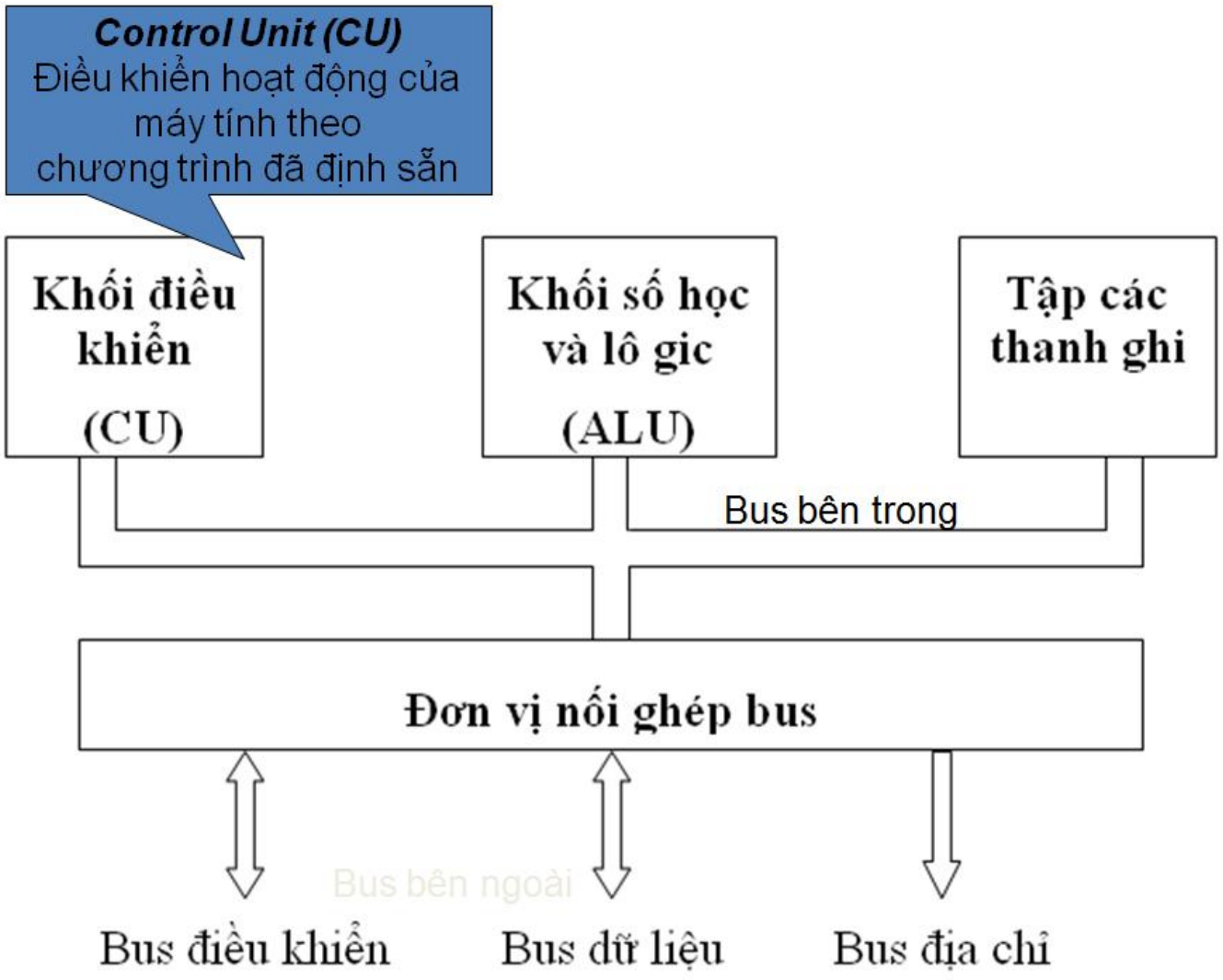
Chức năng

- Chức năng:
  - điều khiển hoạt động của toàn bộ hệ thống máy tính
  - xử lý dữ liệu
- Nguyên tắc hoạt động: theo chương trình nằm trong bộ nhớ chính bằng cách:
  - nhận lệnh từ bộ nhớ chính
  - giải mã lệnh và phát các tín hiệu điều khiển thực thi lệnh
  - CPU có thể trao đổi dữ liệu với bộ nhớ chính hay hệ thống vào-ra.
  - thực hiện lệnh
  - ghi kết quả

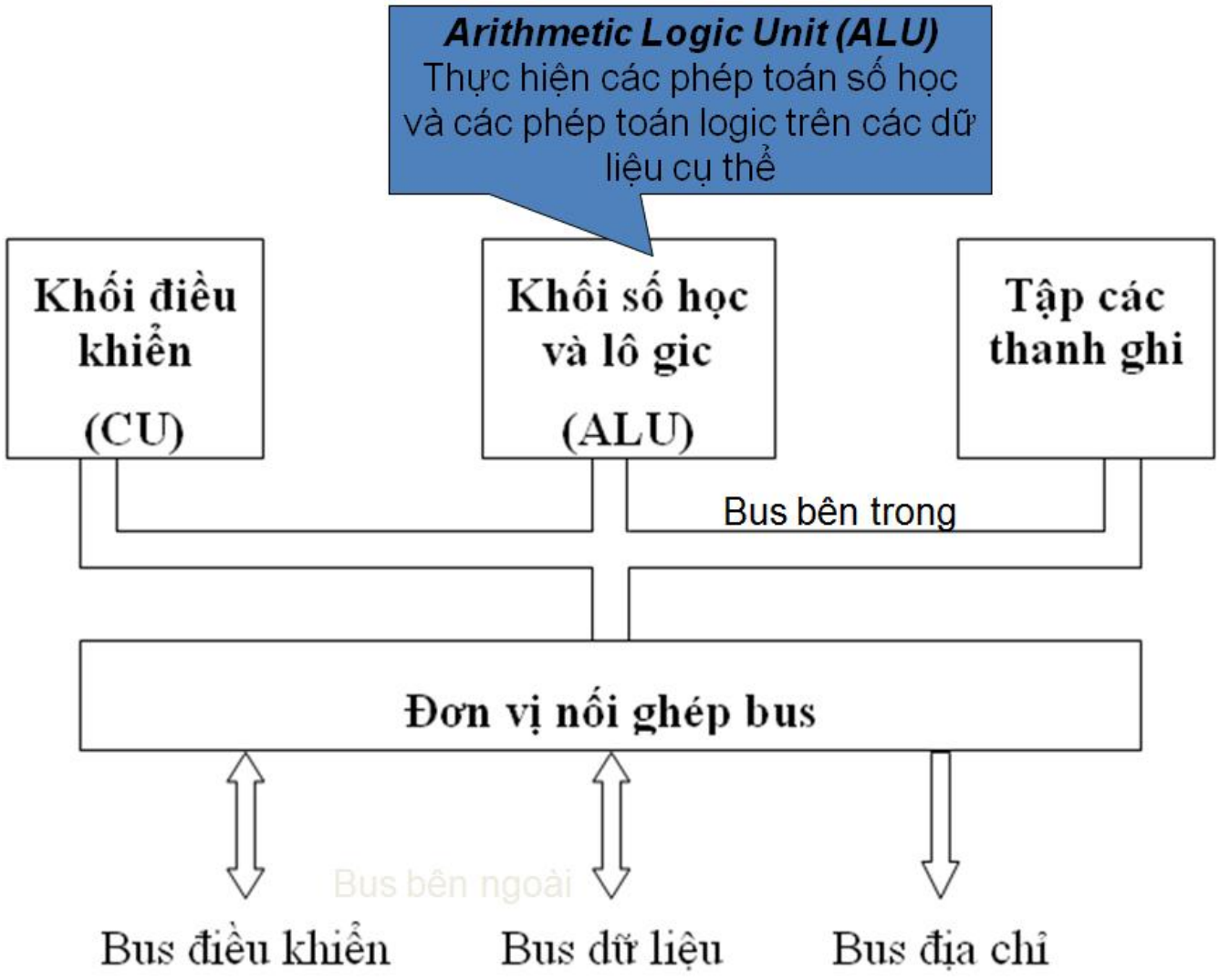
Các thành phần cơ bản



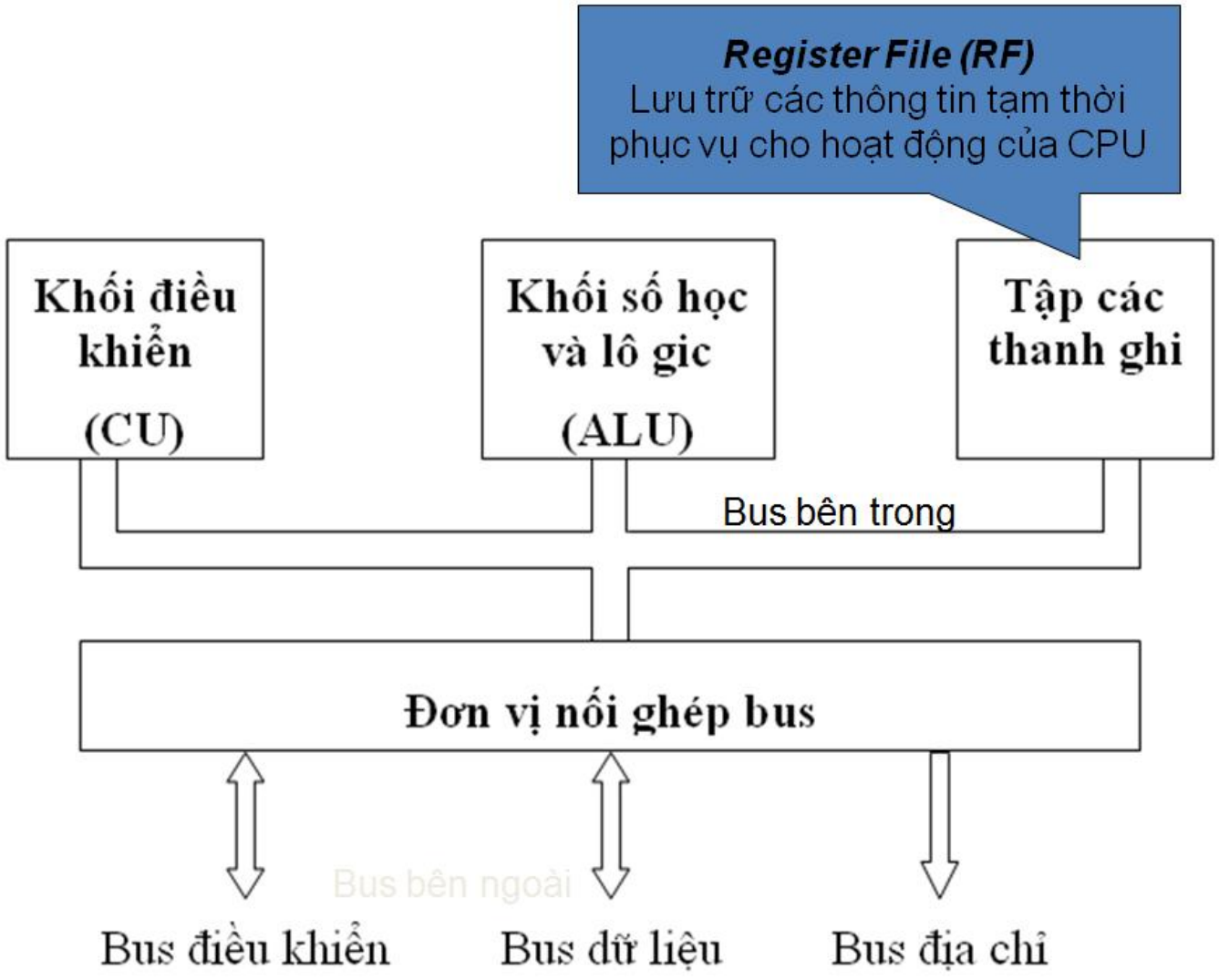
Các thành phần cơ bản



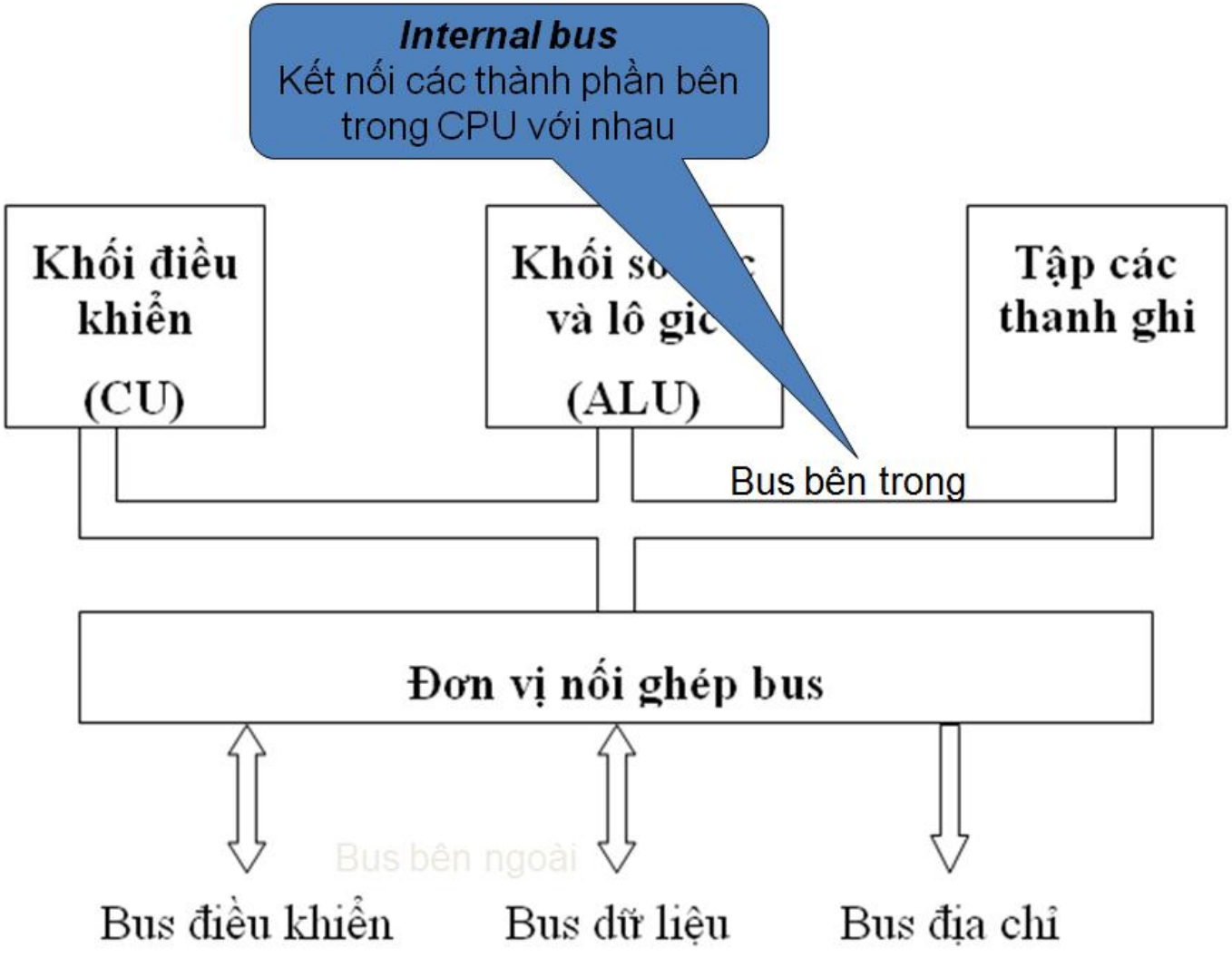
Các thành phần cơ bản



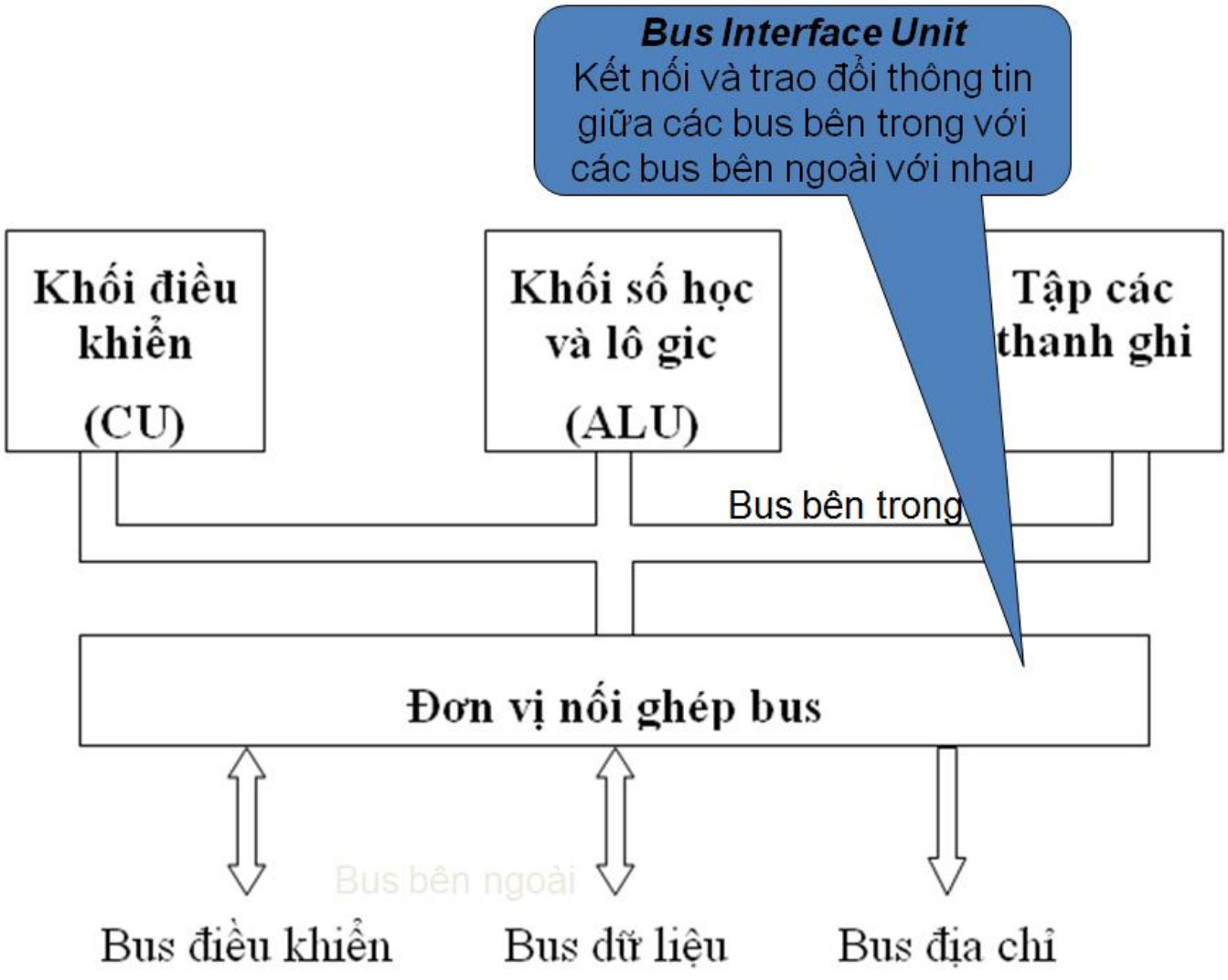
Các thành phần cơ bản



Các thành phần cơ bản



Các thành phần cơ bản



Các thành phần cơ bản

- Khối điều khiển (Control Unit – CU):

  điều khiển hoạt động của máy tính theo chương trình đã định sẵn
- Khối tính toán số học và logic (Arithmetic – Logic Unit - ALU)

  thực hiện các phép toán số học và các phép toán logic trên các dữ liệu cụ thể
- Tập các thanh ghi (Register File - RF)

  lưu trữ các thông tin tạm thời phục vụ cho hoạt động của CPU
- Bus bên trong (Internal Bus)

  kết nối các thành phần bên trong CPU với nhau
- Đơn vị ghép nối bus (Bus Interface Unit – BIU)

  kết nối và trao đổi thông tin với nhau giữa các bus bên trong với các bus bên ngoài

Bộ vi xử lý

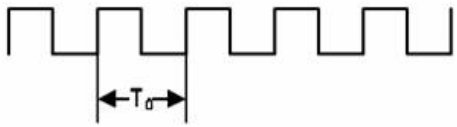
- 2 dòng chính:
  - Intel: Pentium, Core 2 Duo, Core i3, i5, i7, ...
  - AMD: Opteron, Athlon,...
- Bộ vi xử lý (Microprocessor)
  - là CPU được chế tạo trên một vi mạch
  - có thể gọi CPU là bộ vi xử lý. Tuy nhiên, các bộ vi xử lý hiện nay có cấu trúc phức tạp hơn nhiều so với một CPU cơ bản

Tốc độ bộ vi xử lý

- Tốc độ đo bằng:
  - số lệnh được thực hiện trong 1s (MIPS - Milliions of Intructions per Second)
  - khó đánh giá chính xác (còn phụ thuộc bộ nhớ, bo mạch đồ họa...)
- Tần số xung nhịp của bộ vi xử lý:
  - bộ xử lý hoạt động theo một xung nhịp (clock) có tần số xác định
  - tốc độ của bộ xử lý được đánh giá gián tiếp thông qua tần số xung nhịp

Tốc độ bộ vi xử lý

- Dạng xung nhịp:



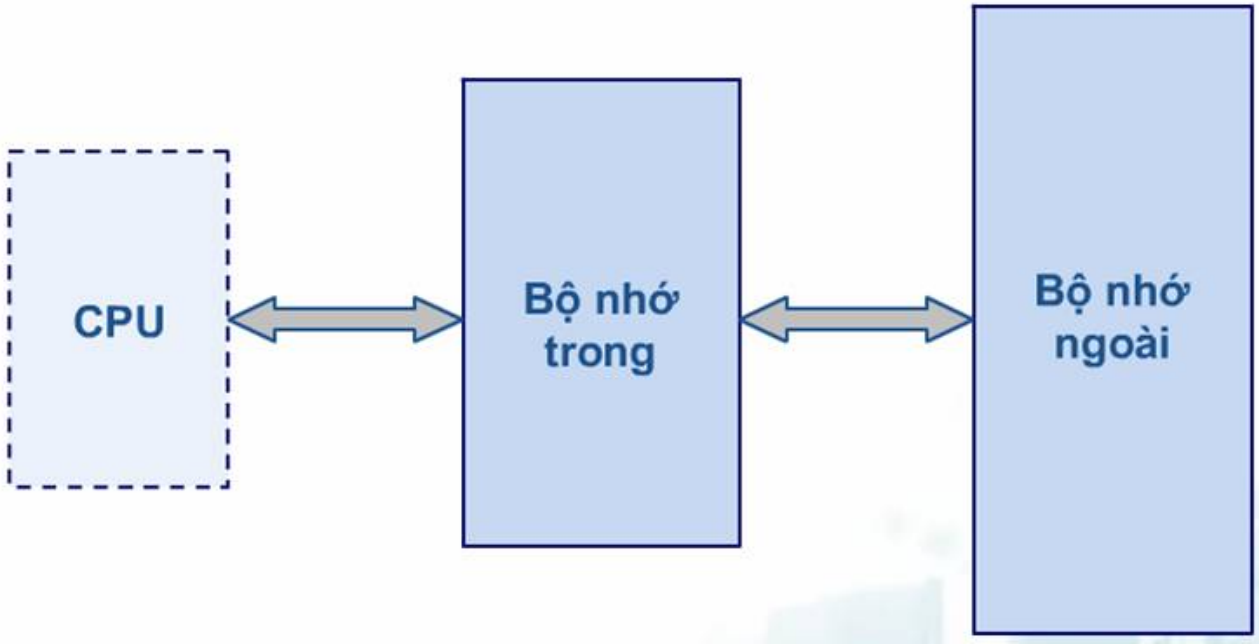
- T_0: chu kỳ xung nhịp
- Mỗi thao tác của bộ xử lý mất một số nguyên lần chu kỳ T_0
  $\Rightarrow T_0$ càng nhỏ thì bộ xử lý chạy càng nhanh
- Tần số xung nhịp: $f_0=1/T_0$ gọi là tần số làm việc của CPU
- VD: Máy tính dùng bộ xử lý Pentium IV 2GHz
  Ta có: $f_0 = 2GHz = 2 \times 10^9 Hz$
  $\rightarrow T_0 = 1/f_0 = 1 / (2 \times 10^9) = 0,5$ ns

Tốc độ bộ vi xử lý - ví dụ 1 số siêu máy tính

- Roadrunner –3rd ,IBM
  - 133 triệu USD
  - tốc độ: 1.04 petaflops (1.04 triệu tỷ phép tính/s)
  - 6 tỷ người dùng hand calculator * 24h/ngày * 7 ngày/tuần * 46 năm = 1 ngày Roadrunner
- Nabulae – 2nd ,China 1.2 petaflops
- Jaguar – 1st , USA 1.8 petaflops

Tổng quan

- Chức năng: lưu trữ chương trình và dữ liệu
- Các thao tác cơ bản với bộ nhớ: đọc (read) và ghi (write)
- Các thành phần chính
  - bộ nhớ trong (internal memory)
  - bộ nhớ ngoài (external memory)

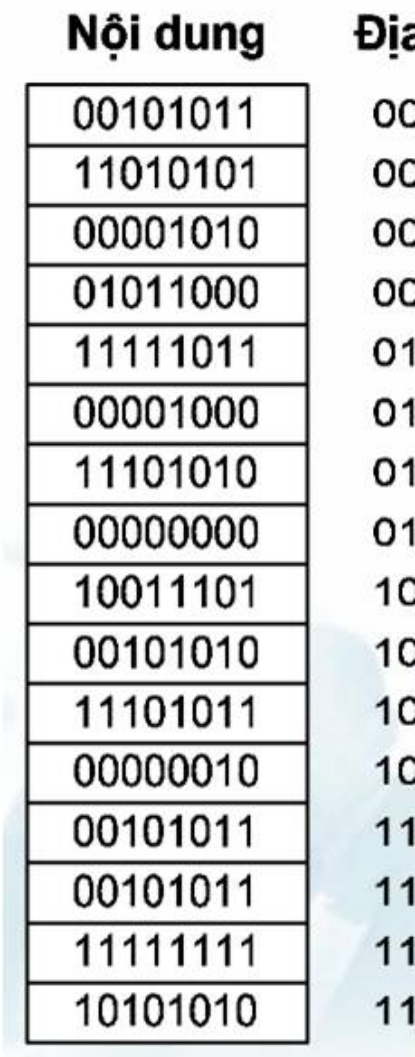


Bộ nhớ trong

- Chức năng: chứa các thông tin mà CPU có thể trao đổi trực tiếp
- Đặc điểm:
  - tốc độ nhanh
  - dung lượng không lớn
  - sử dụng bộ nhớ bán dẫn: ROM và RAM
- Các loại bộ nhớ trong:
  - bộ nhớ chính (main memory)
  - bộ nhớ đệm nhanh (cache memory)

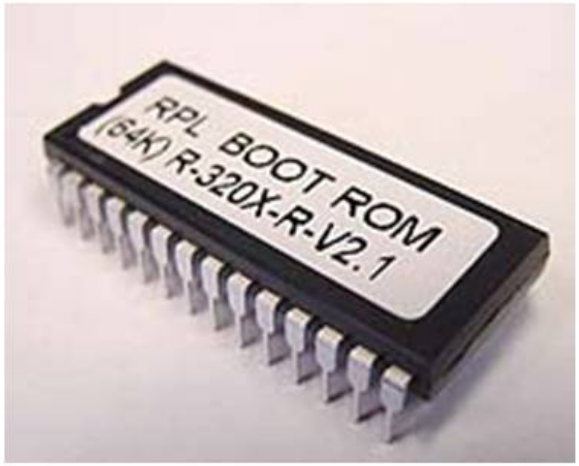
Bộ nhớ chính

- Là thành phần nhớ tồn tại trên mọi hệ thống máy tính
- Chứa các chương trình và dữ liệu đang được CPU sử dụng
- Tổ chức thành các ngăn nhớ được đánh địa chỉ, ngăn nhớ thường được tổ chức theo Byte
- Nội dung của ngăn nhớ có thể thay đổi, song địa chỉ vật lý của ngăn nhớ luôn cố định
- Thông thường, bộ nhớ chính gồm 2 phần: ROM và RAM



Bộ nhớ chính - ROM (Read Only Memory)

- Vùng nhớ chỉ đọc -> thông tin không bị mất khi mất nguồn điện
- Tích hợp trên các thiết bị
- Nội dung được cài đặt sẵn tại nơi sản xuất thiết bị
- Chức năng chính
  - chứa các phần mềm thực hiện các công việc của thiết bị (*firmware*)
  - đôi khi được gọi: ROM BIOS (*Basic Input/Output System*)



Bộ nhớ chính - RAM (Random Access Memory)

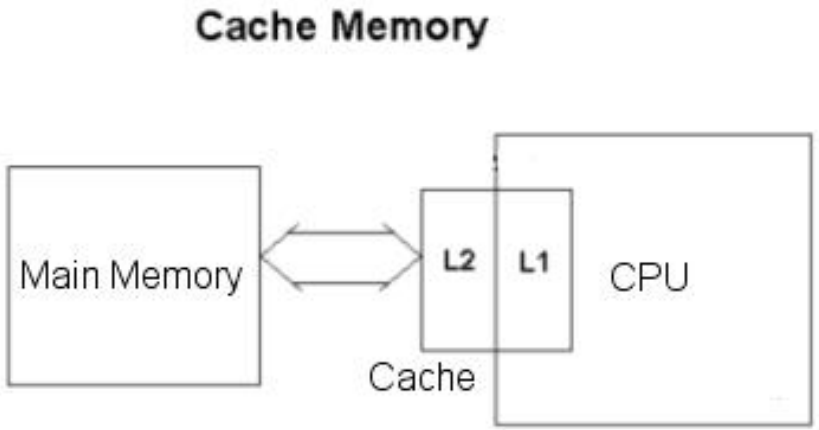
- Bộ nhớ truy cập ngẫu nhiên
  - không phải di chuyển tuần tự
  - được chia thành các ô nhớ có đánh địa chỉ
  - thời gian thực hiện thao tác đọc hoặc ghi đối với mỗi ô nhớ là như nhau, cho dù đang ở bất kỳ vị trí nào trong bộ nhớ
- Lưu trữ các thông tin thay đổi và các thông tin được sử dụng hiện hành
- Thông tin lưu trên RAM chỉ là tạm thời, chúng sẽ mất đi khi mất nguồn điện cung cấp

Phân loại RAM theo công nghệ

- SRAM (static RAM): RAM tĩnh
- DRAM (dynamic RAM): RAM động
  - SDRAM (Synchronous Dynamic RAM):
    - SDR (Single Data Rate): Đã lỗi thời
    - DDR (Double Data Rate): Đã được thay thế bởi DDR2
    - DDR2 (Double Data Rate 2), DDR3: Là thế hệ tiếp theo của DDR, hiện được sử dụng rộng rãi
  - RDRAM (Rambus Dynamic RAM): Ít người dùng vì không nhanh hơn SDRAM là bao nhưng lại đắt hơn nhiều

Bộ nhớ đệm nhanh (Cache Memory)

- Tốc độ xử lý của CPU > > tốc độ truy nhập dữ liệu từ RAM $\Rightarrow$ tăng tốc bằng cách sử dụng CACHE
  - dữ liệu nạp từ RAM vào CACHE khi cần
  - CPU thao tác với dữ liệu trên CACHE thay vì trên RAM
- Hiện nay CACHE được tích hợp trong chip vi xử lý
- CPU truy nhập dữ liệu trên CACHE nhanh hơn trên RAM
- CACHE chia làm một số mức: L1, L2, ...
- CACHE có thể có hoặc không



Bộ nhớ ngoài

- Lưu giữ tài nguyên phần mềm của máy tính, bao gồm: Hệ điều hành, các chương trình và dữ liệu
- Bộ nhớ ngoài được kết nối với hệ thống dưới dạng các thiết bị vào ra
- Dung lượng lớn
- Tốc độ truy cập dữ liệu chậm
- Phân loại:
  - Bộ nhớ từ: Đĩa cứng, đĩa mềm
  - Bộ nhớ quang: Đĩa CD, DVD,...
  - Bộ nhớ bán dẫn: Flash disk, memory card


Floppy disk


Compact disk

Compact Flash Card


USB Flash Drive

Bộ nhớ ngoài

- Đĩa mềm (Floppy disk)
  - Dung lượng: 1.44MB
  - Kích thước 3.5"
  - Có 2 mặt đĩa
  - Phạm vi xử dụng: hiện nay, không thông dụng
    - laptop: thường không có
    - desktop: ít sử dụng
- Đĩa cứng (Hard disk)
  - Dung lượng lớn
  - Là nơi cài đặt HĐH và các chương trình ứng dụng
  - Phạm vi sử dụng: rộng
  - Một máy tính: có thể có nhiều ổ cứng

Bộ nhớ ngoài

- Ổ cứng ngoài (External Hard Disk)
  - Dễ dàng mang từ nơi này sang nơi khác
  - Kết nối qua các giao tiếp:
    - USB 2.0
    - IEEE 1394, FireWare 800
    - Ethernet
  - Dung lượng lớn: 250GB, 500GB, ...
  - Sử dụng:
    - Lưu trữ dữ liệu
    - Lưu trữ trong mạng

Bộ nhớ ngoài

- CD-ROM (Compact disc read-only memory)
  - Kích thước thường 700MB
  - Đĩa quang, đầu đọc laze
  - Tốc đô chậm hơn so với đĩa từ (ổ cứng)
  - Phân loại:
    - CD-R (compact disc-recordable): chỉ ghi 1 lần
    - CD-RW (compact disc-rewritable): có thể xóa, ghi lại nhiều lần
  - Để ghi dữ liệu cần có ổ đọc/ghi CD và phần mềm hỗ trợ
- DVD (Digital Video Disc or Digital Versatile Disc)
  - 2 loại : một mặt (4.7GB) và 2 mặt (8.5GB)
  - Cần có ổ đọc DVD hoặc ổ đọc/ghi DVD và phần mềm hỗ trợ

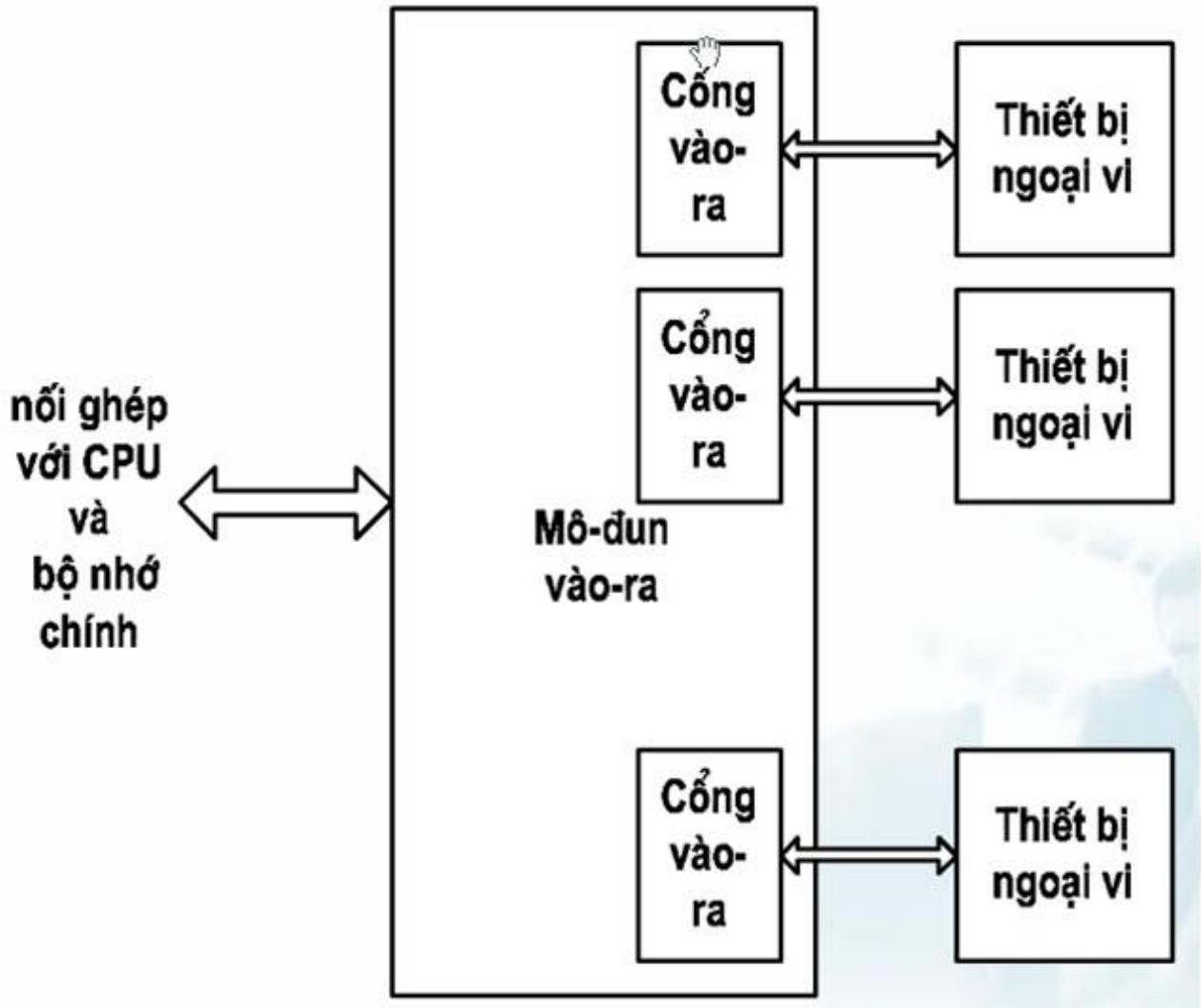
Bộ nhớ ngoài

- Flash sticks or memory - USB
  - Kết nối với máy tính qua cổng USB
  - Kích thước: Đa dạng 1G, 2G, 4G, 8G, ...
  - Sử dụng rộng rãi:
    - Lưu trữ dữ liệu cá nhân
    - Sử dụng trong các thiết bị nghe nhìn

Hệ thống vào-ra

- Chức năng: trao đổi thông tin giữa máy tính với thế giới bên ngoài
- Các thao tác cơ bản:
  - VÀO (input) dữ liệu
  - RA (output) dữ liệu
- Các thành phần chính
  - các thiết bị vào-ra *(IO devices)* dữ liệu hay còn gọi là thiết bị ngoại vi (Peripheral devices)
  - các mô-đun ghép nối vào-ra *(IO Interface modules)*

Cấu trúc cơ bản của hệ thống vào-ra



Thiết bị vào-ra

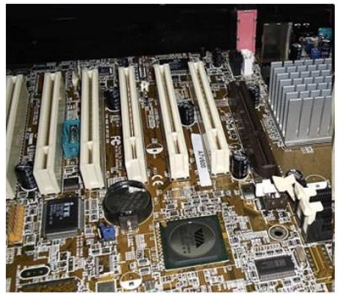
- Chức năng: chuyển đổi dữ liệu giữa bên trong và bên ngoài máy tính
- Các thiết bị ngoại vi cơ bản:
  - thiết bị vào: Bàn phím, chuột, máy quét,...
  - thiết bị ra: Màn hình, máy in,...
  - thiết bị nhớ: Các ổ đĩa,...
  - thiết bị truyền thông: Modem,...

Mô-đun ghép nối vào-ra

- Các thiết bị vào-ra được kết nối với CPU thông qua các mô-đun ghép nối vào-ra mà không được kết nối trực tiếp
- Trong các mô-đun ghép nối vào-ra có các cổng vào-ra (*IO Port*), mỗi cổng được đánh địa chỉ bởi CPU i.e mỗi cổng 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

Liên kết hệ thống (buses)

- CPU, bộ nhớ chính và hệ thống vào-ra cần phải kết nối với nhau để trao đổi thông tin $\Rightarrow$ kết nối = tập các đường kết nối gọi là bus
- Thực tế bus trong máy tính khá phức tạp, nó được thể hiện bằng các đường dẫn trên các bản mạch, các khe cắm trên bản mạch chính, các cáp nối,...
- Độ rộng của bus: số đường dây của bus có thể truyền thông tin đồng thời
- 3 loại: bus địa chỉ, bus dữ liệu và bus điều khiển

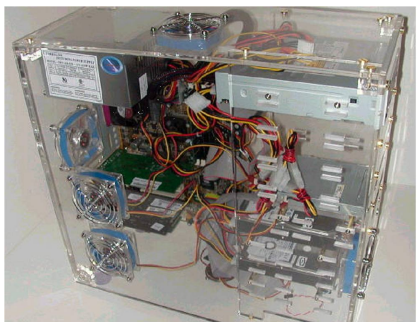


Máy tính

- Hộp máy tính (case)
- Các thiết bị ngoại vi

Hộp máy tính

- Bản mạch chính (Mainboard):
  - bộ vi xử lý
  - bộ nhớ hệ thống (bộ nhớ chính): chip nhớ ROM và các module nhớ RAM
  - các vi mạch điều khiển tổng hợp (chipset)
  - các kênh truyền tín hiệu (bus)
  - các khe cắm mở rộng
- Các loại ổ đĩa: ổ đĩa cứng, ổ đĩa mềm, ổ đĩa quang, ...
- Các cổng vào-ra
- Bộ nguồn và quạt



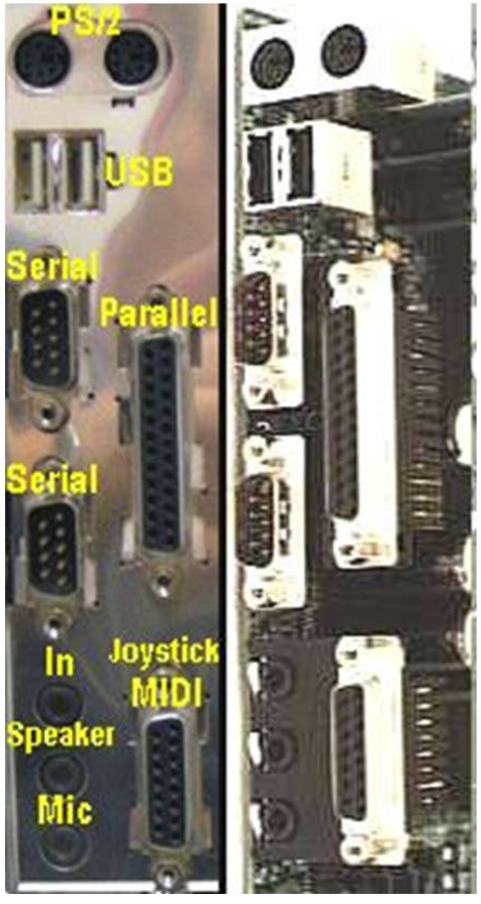
Hộp máy tính

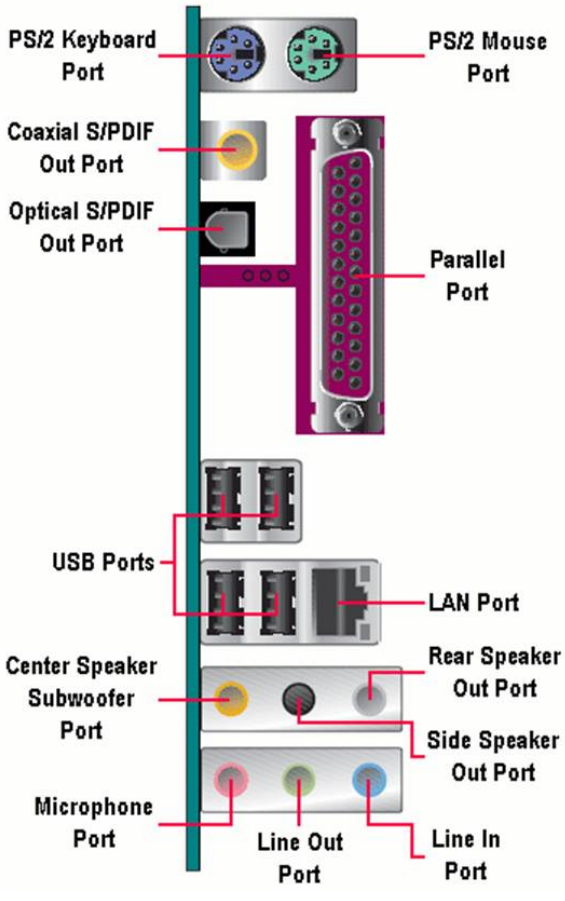
- Ổ đĩa:

- Bộ nguồn và quạt:

Hộp máy tính

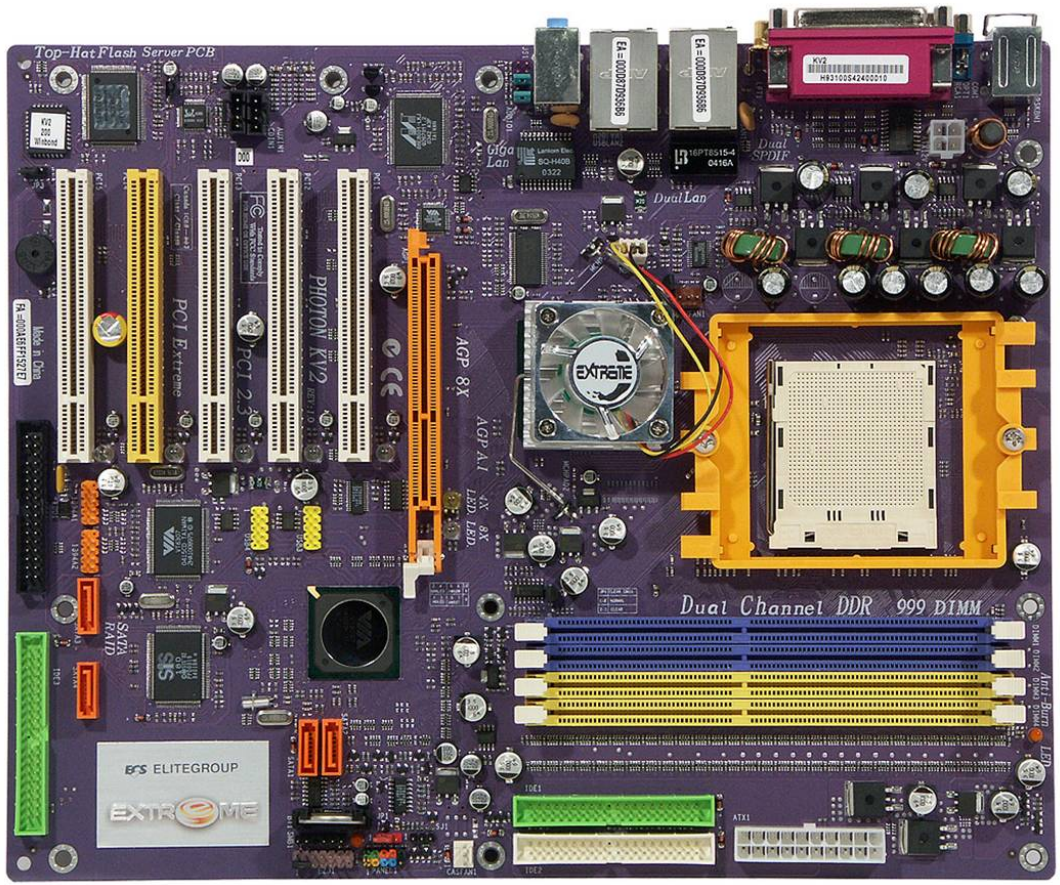
- Các cổng vào ra:





Hộp máy tính

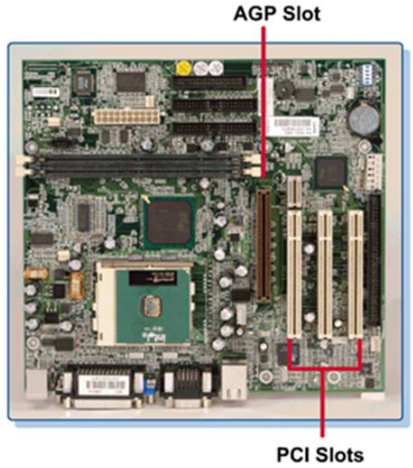
- Bản mạch chính:



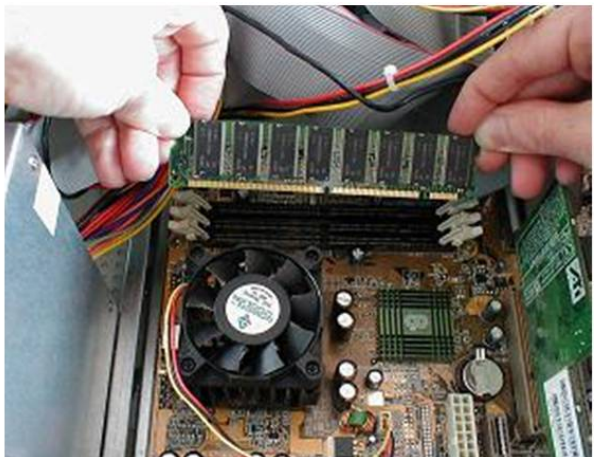
Một số linh kiện trên bản mạch chính

Bộ vi xử lý

Khe cắm mở rộng

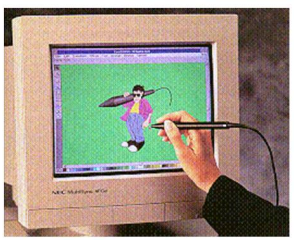


Bộ nhớ hệ thống



Các thiết bị ngoại vi

Màn hình (monitor), bàn phím (keyboard), chuột (mouse), loa (speaker), máy in (printer), máy quét ảnh (scanner), modem






Khái niệm

- Máy tính hoạt động theo một qui trình tự động đã định sẵn gọi là chương trình (*program*) hay còn gọi là phần mềm máy tính (*Software Computer*).
- Máy tính có thể hoạt động nếu thiếu phần mềm?
- Giá của một số phần mềm?
- Làm thế nào để viết ra phần mềm?

Phân loại theo phương thức hoạt động

- Phần mềm hệ thống
  - dùng để vận hành máy tính và các phần cứng máy tính
  - ví dụ: Các hệ điều hành máy tính: Windows XP, Vista, Ubuntu, ...
- Phần mềm ứng dụng
  - 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

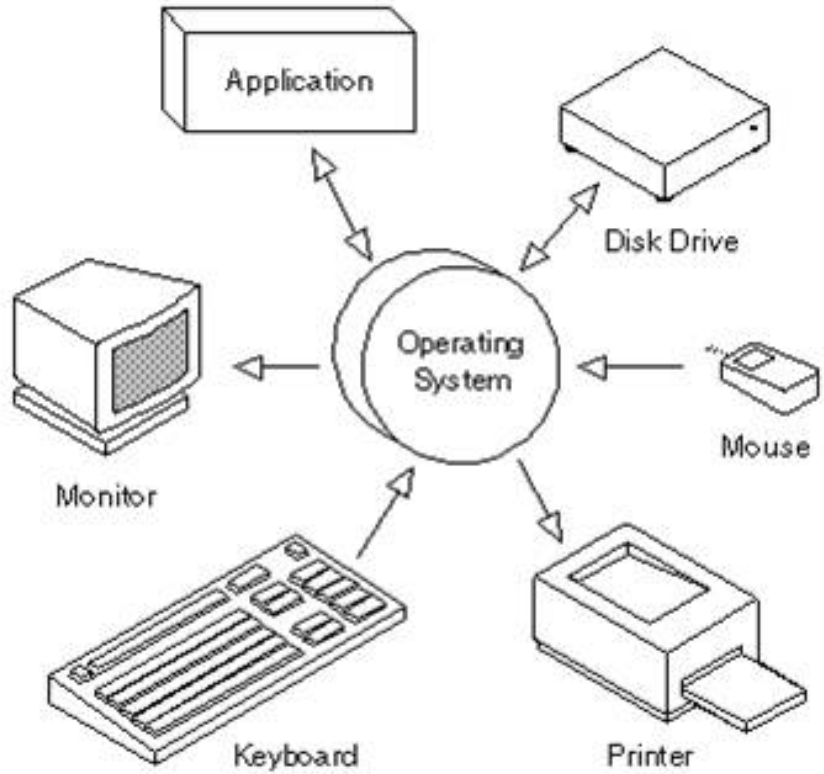
Phân loại theo đặc thù ứng dụng và môi trường

- 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 (AI SW)
- ...

Hệ điều hành

Là phần mềm hệ thống giúp:

- Điều khiển và kiểm soát hoạt động của các thiết bị (ổ đĩa, bàn phím, màn hình, máy in, ...)
- Quản lý việc cấp phát tài nguyên của máy tính như bộ xử lý trung tâm, bộ nhớ, các thiết bị vào ra, ...
- Sắp xếp sự thực thi của tất cả các phần mềm khác



Hệ điều hành

- Phần mềm đầu tiên được chạy khi máy khởi động
- Đóng vai trò trung gian trong việc giao tiếp giữa người sử dụng và phần cứng máy tính
- 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
  - 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, ...
  - máy tính Mac : MAC OS
  - ...

Tệp (*File, tập tin*)

- Tệp là tập hợp các dữ liệu có liên quan với nhau và được tổ chức theo 1 cấu trúc, thường được lưu trữ bên ngoài máy tính
- Nội dung của tệp có thể là chương trình, dữ liệu, văn bản,...
- Tên tệp tin thường có 2 phần:
  - phần tên (name)
  - phần mở rộng (extension)
  - giữa phần tên và phần mở rộng có một dấu chấm (.) ngăn cách

Tệp - Phần tên

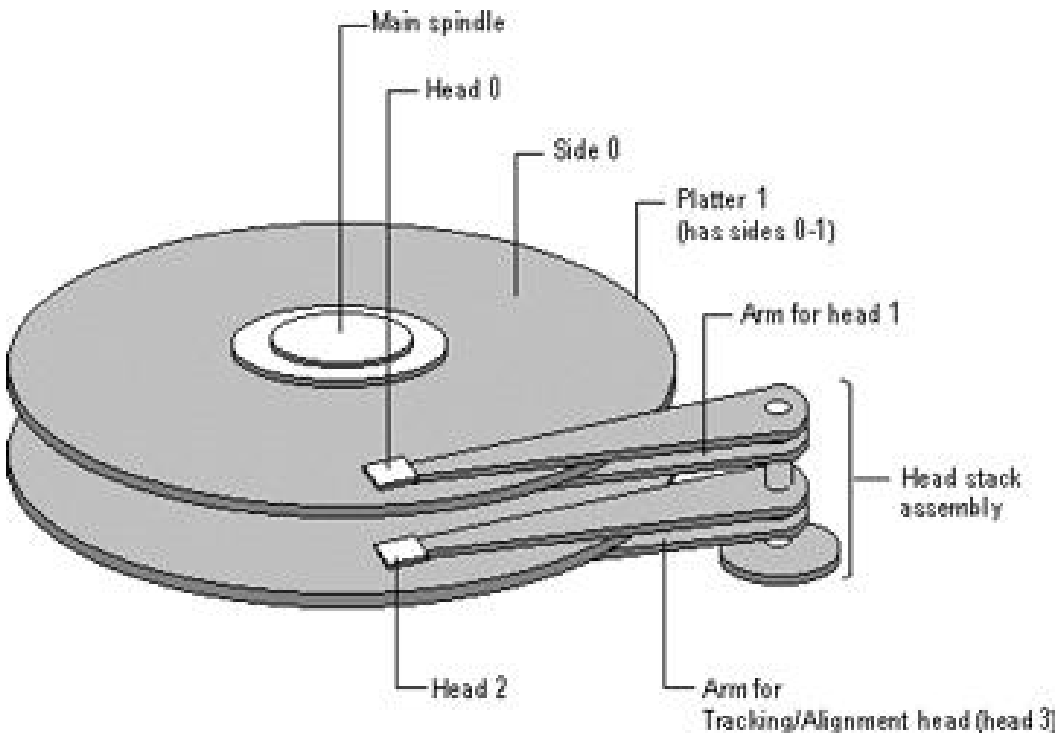
- Phần tên có thể gồm
  - Ký tự chữ từ A đến Z (hoa và thường)
  - Chữ số từ 0 đến 9
  - Ký tự khác: # , $, %, ~, ^, @, (,), !, _, khoảng trắng
- *Lưu ý*: Nên đặt tên mang tính gợi nhớ
- VD:
  - Tên file hợp lệ: dulieu100101.txt, dulieu$100101.dat
  - Tên file không hợp lệ: ‘dulieu100101.txt, ?abc.dat

Tệp - Phần mở rộng

- Phần mở rộng
  - Thường là 3 ký tự hợp lệ
  - Chương trình ứng dụng tạo ra tập tin tự đặt
    - COM, EXE : các file khả thi chạy trực tiếp
    - TXT, DOC, ... : các file văn bản
    - BMP, GIF, JPG, ... : các file hình ảnh
    - MP3, DAT, WMA, . . . : các file âm thanh, video
- Ký hiệu đại diện (*Wildcard*): dùng để chỉ một nhóm tập tin
  - dấu ?: đại diện cho một ký tự bất kỳ trong tên tập tin
  - dấu *: đại diện cho một chuỗi ký tự bất kỳ trong tên tập tin
    - Bai?.doc Bai1.doc, Bai6.doc, Baiq.doc, . . .
    - Bai*.doc Bai.doc, Bai6.doc, Bai12.doc, Bai Tap.doc, . . .

Quản lý tập tin của hệ điều hành

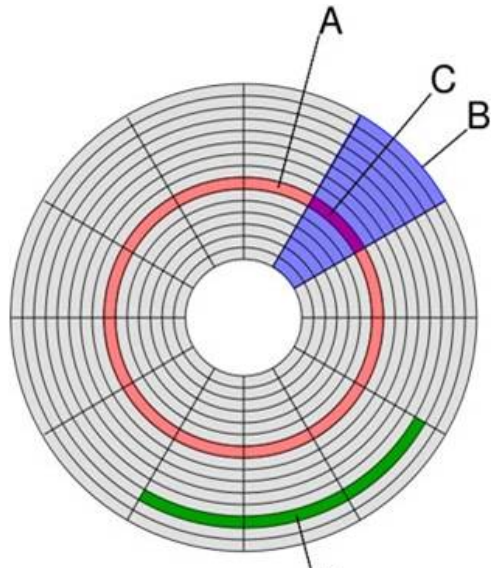
Cấu trúc đĩa từ : ổ đĩa từ gồm nhiều đĩa (platter) gắn đồng trục



Quản lý tập tin của hệ điều hành

Cấu trúc đĩa từ (...):

- Mỗi mặt đĩa chia thành các rãnh từ (*track*) các đường tròn đồng tâm, đánh số từ ngoài vào trong, bắt đầu từ 0
- Mỗi rãnh từ được chia thành các cung từ (*Sector*), dung lượng thường 512 byte
- Cylinder: các rãnh có cùng bán kính nằm trên các mặt đĩa khác nhau



- A - rãnh từ (*Track*)
- B - dải Cung từ (*Sector track*)
- C - Cung từ (*Sector*)
- D - liên cung (*Cluster*)

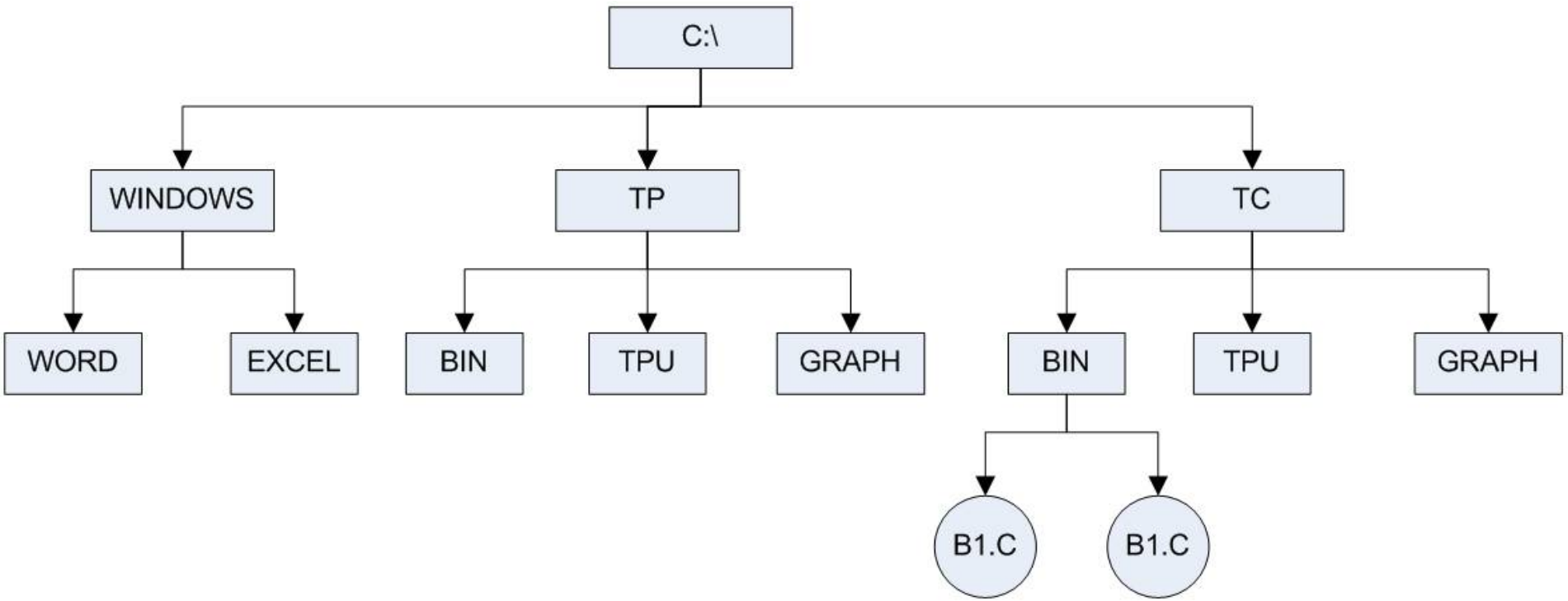
Quản lý tập tin của hệ điều hành

Tổ chức ghi thông tin trên đĩa:

- Thông tin lưu trữ trên đĩa dưới dạng các tệp
  - 1 tệp chiếm 1 hoặc nhiều sectors
- Hệ điều hành cho phép chia đĩa thành các vùng. Mỗi vùng:
  - chia thành các vùng con
  - có 1 vùng con chứ thông tin về vùng đó (thư mục (*Folder*, *Directory*))
  - tệp được lưu trữ ở các vùng, được tổ chức lưu trữ dạng cây (*Tree*)
- **Thư mục** là nơi lưu giữ các tập tin theo một chủ đề nào đó theo ý người sử dụng

Quản lý tập tin của hệ điều hành

- mỗi ổ đĩa (*ổ đĩa logic*) có một thư mục chung gọi là thư mục gốc
- thư mục gốc không có tên riêng và được ký hiệu là \
- thư mục có thể chứa các tệp và thư mục con



Quản lý tập tin của hệ điều hành

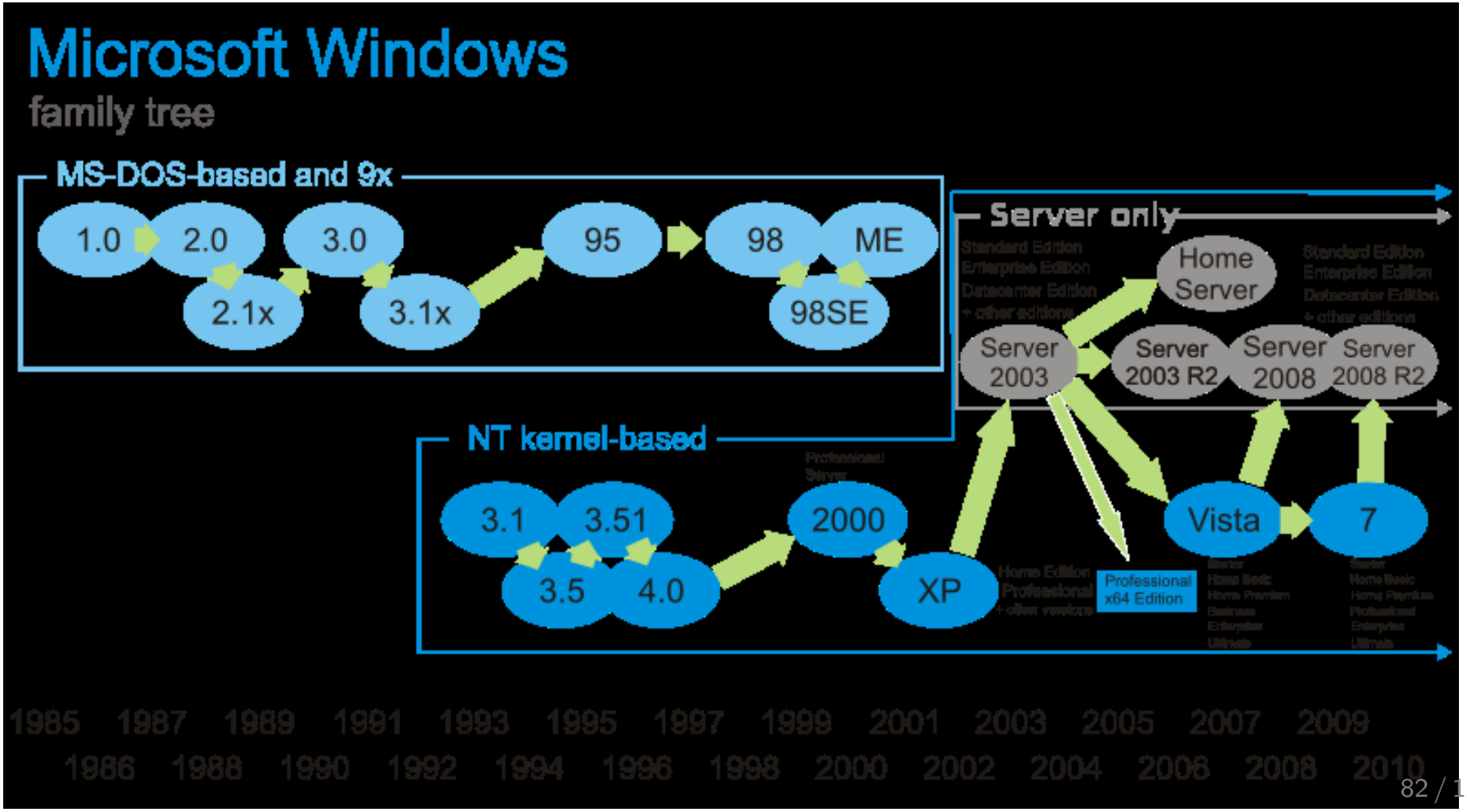
- Tên của thư mục: tuân thủ nguyên tắc đặt tên của tệp
- Xác định tên đầy đủ của tệp:
  - Tên tệp đầy đủ gồm nơi lưu trữ tệp (*đường dẫn từ gốc đến tệp (Path)*) + tên tệp
  - Ký hiệu "\" : ngăn cách tên các thư mục
  - Ví dụ : *C:\TC\BIN\B1.C*

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, xem nội dung
- Thao tác với đĩa: tạo khuôn (Format), sao chép đĩa

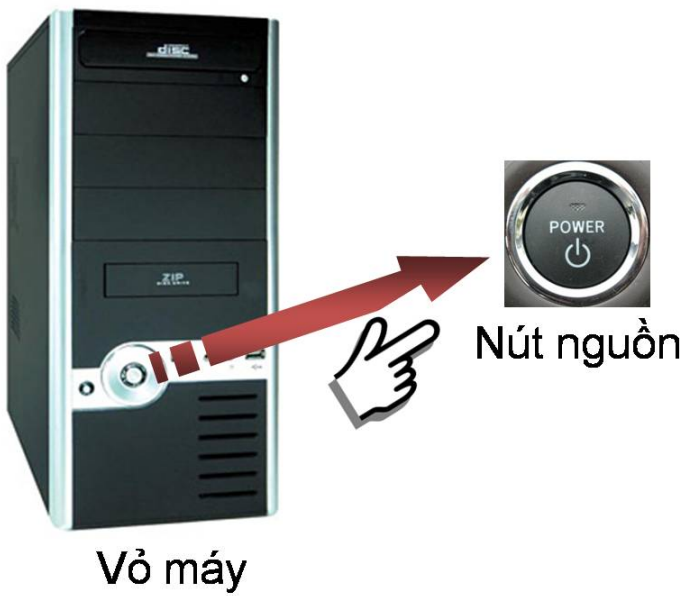
Sự ra đời và phát triển

Windows là một bộ chương trình do hãng Microsoft sản xuất



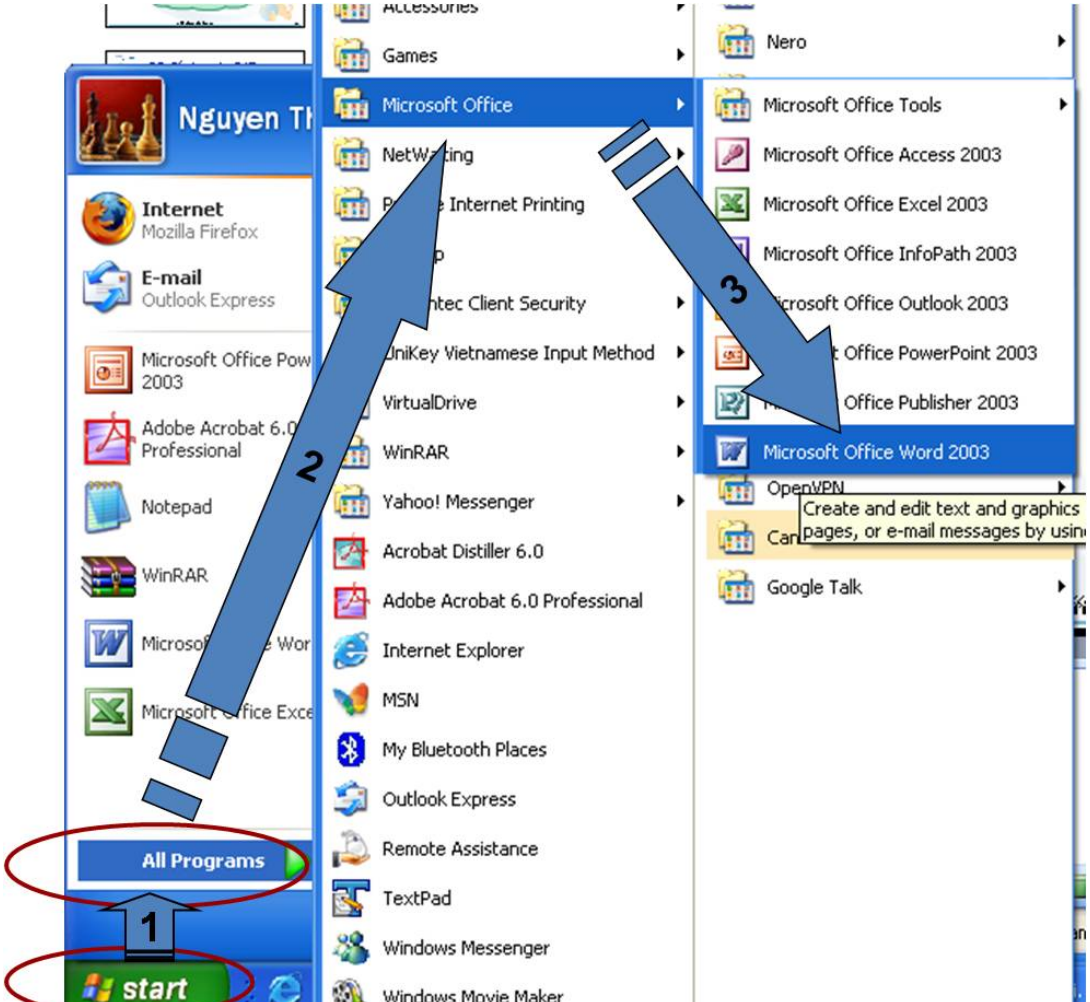
Khởi động

- Bật nguồn
- Chờ đến màn hình đăng nhập
- Nhập tên người dùng và mật khẩu



Khởi động chương trình ứng dụng

- Start $\Longrightarrow$ All Programs $\Longrightarrow$ Chọn chương trình cần khởi động

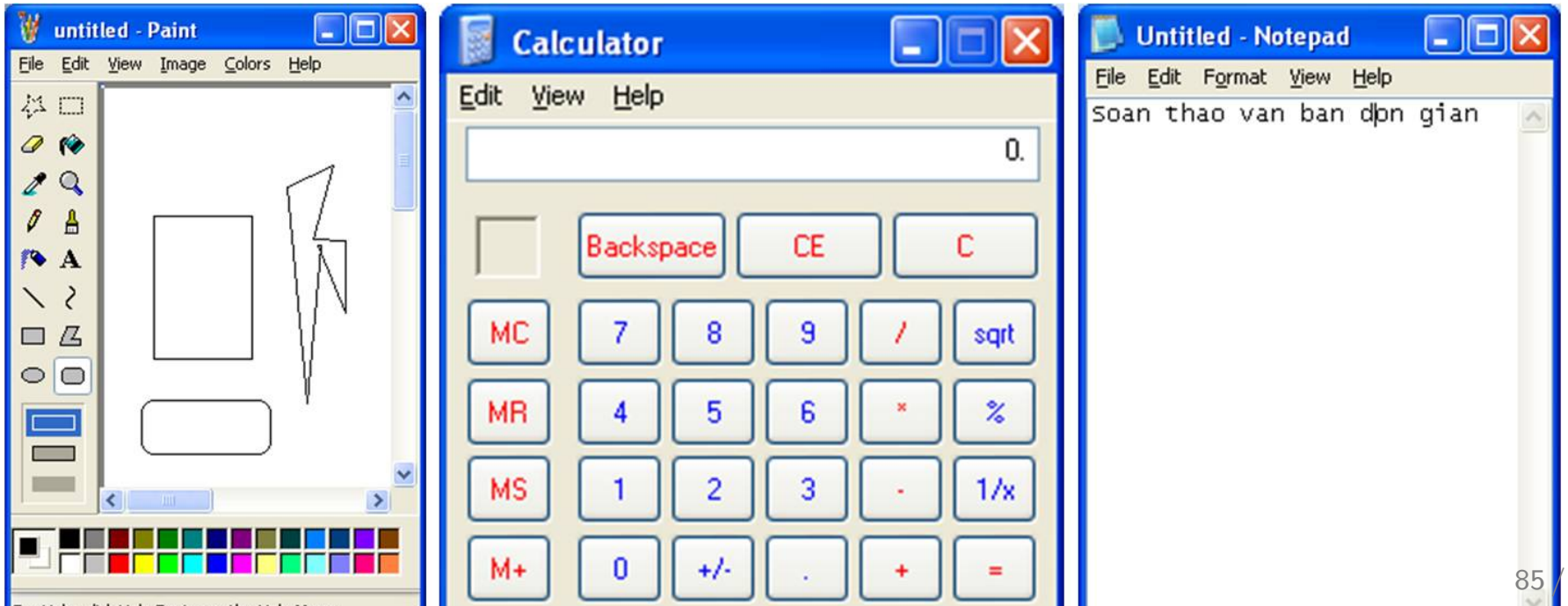


Khởi động chương trình ứng dụng

Một số chương trình có sẵn trong HĐH Windows:

Start $\Longrightarrow$ All Programs $\Longrightarrow$ Accessories $\Longrightarrow$

- Paint: Chương trình vẽ đơn giản
- Calculator: Máy tính tính toán các phép toán đơn giản
- Notepad: Soạn thảo văn bản đơn giản

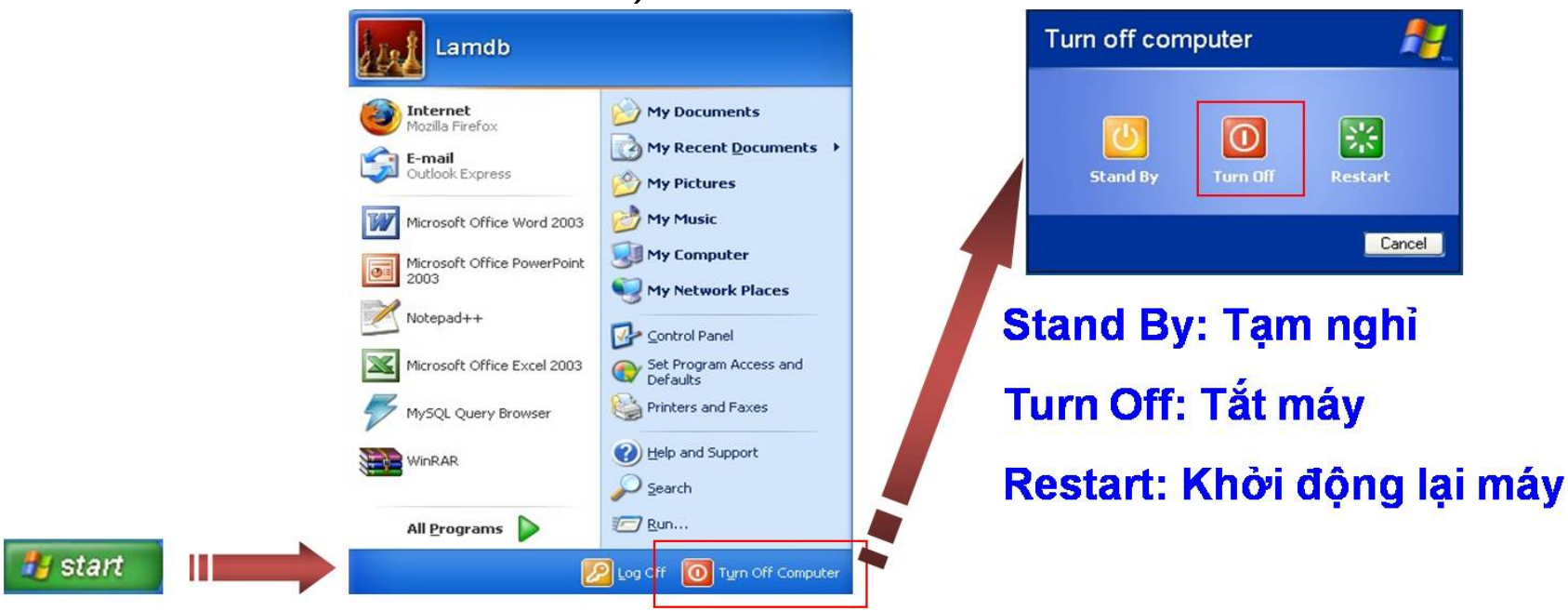


Thoát khỏi người dùng hiện tại

- lưu lại các tập tin và đóng tất cả các ứng dụng đang mở trước khi thoát khỏi phiên làm việc của người dùng hiện tại

Thoát khỏi HĐH (Tắt máy)

- Tắt máy thông thường (trước đó hãy lưu lại các tập tin và đóng tất cả các ứng dụng đang mở) :



- Tắt máy áp đặt: nhấn và giữ nút nguồn trong 5-10 giây

Các loại đối tượng: Tệp, thư mục, biểu tượng

- Tệp tin:

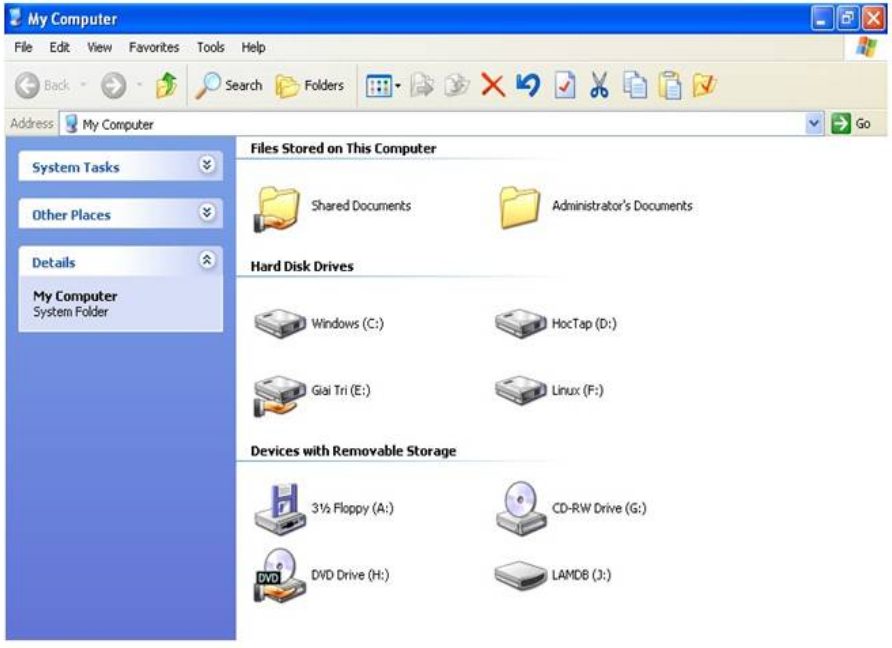
  một tập các thông tin có liên quan với nhau mà máy tính có thể truy nhập thông qua tên
- Thư mục:

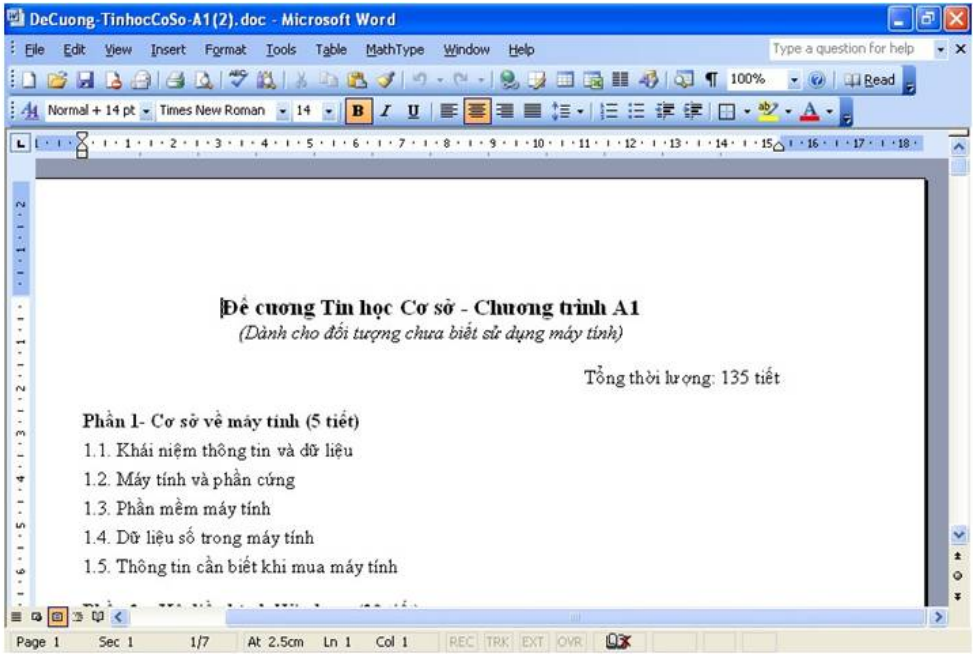
  là một vùng lưu trữ các tệp tin, một thư mục có thể có nhiều thư mục con
- Biểu tượng:

  là những hình ảnh nhỏ biểu diễn tệp tin, thư mục, phần cứng, ...

Các loại đối tượng: Cửa sổ

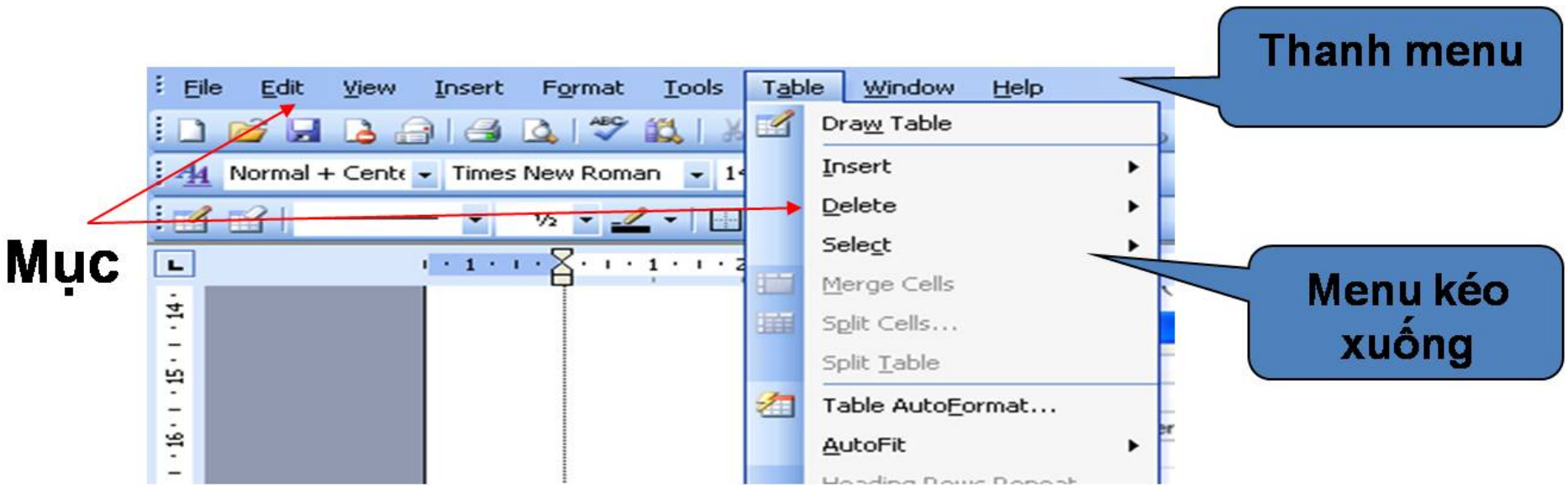
- Cửa sổ: thường là một hình chữ nhật, hiển thị đầu ra hoặc cho phép nhập dữ liệu, ...





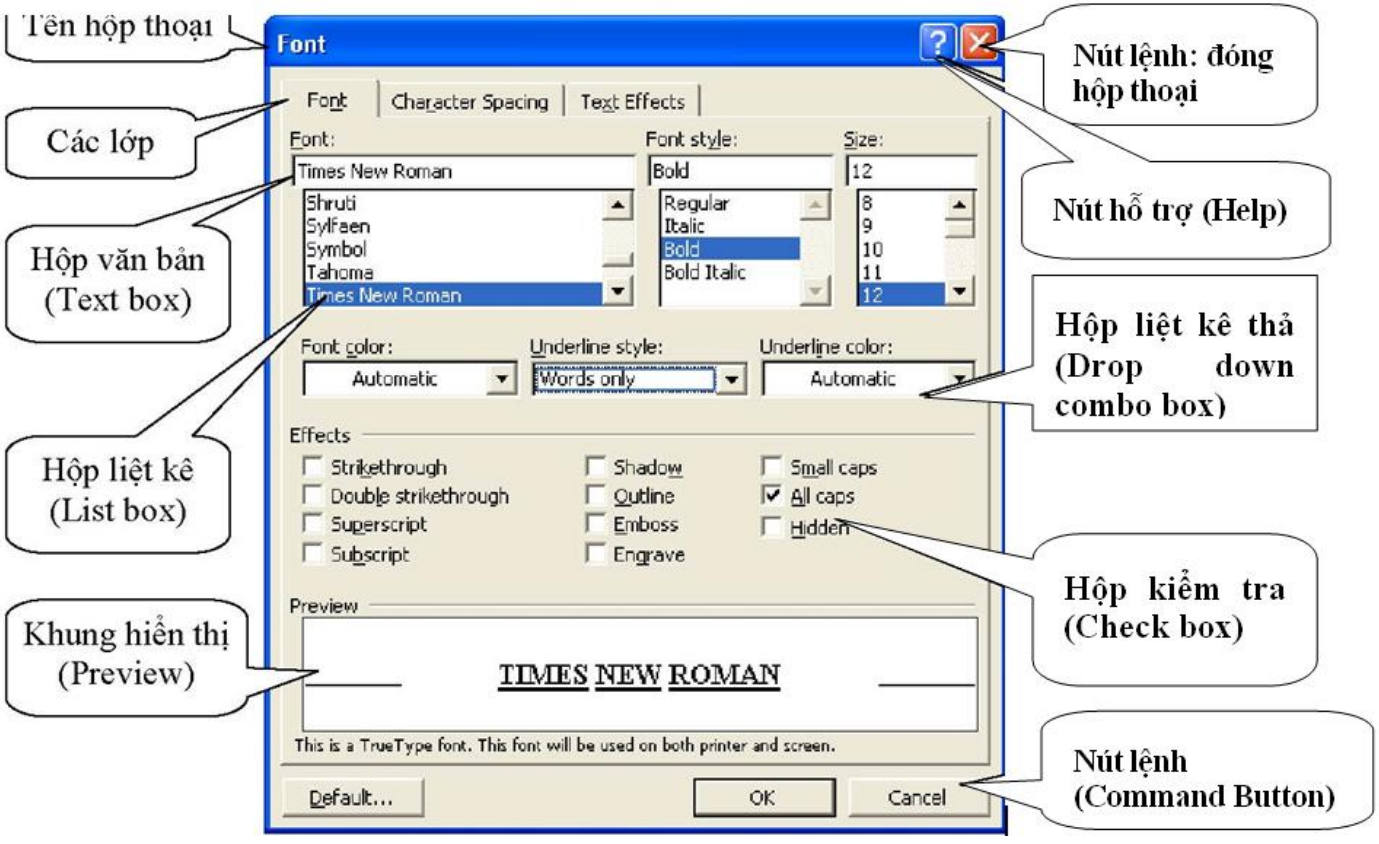
Các loại đối tượng: Bảng chọn (Menu)

- Bảng chọn: tập các thao tác được hiện thị trên màn hình mà người sử dụng có thể lựa chọn
- Mục (Item): một thao tác trên menu



Các loại đối tượng: Hộp thoại và nút

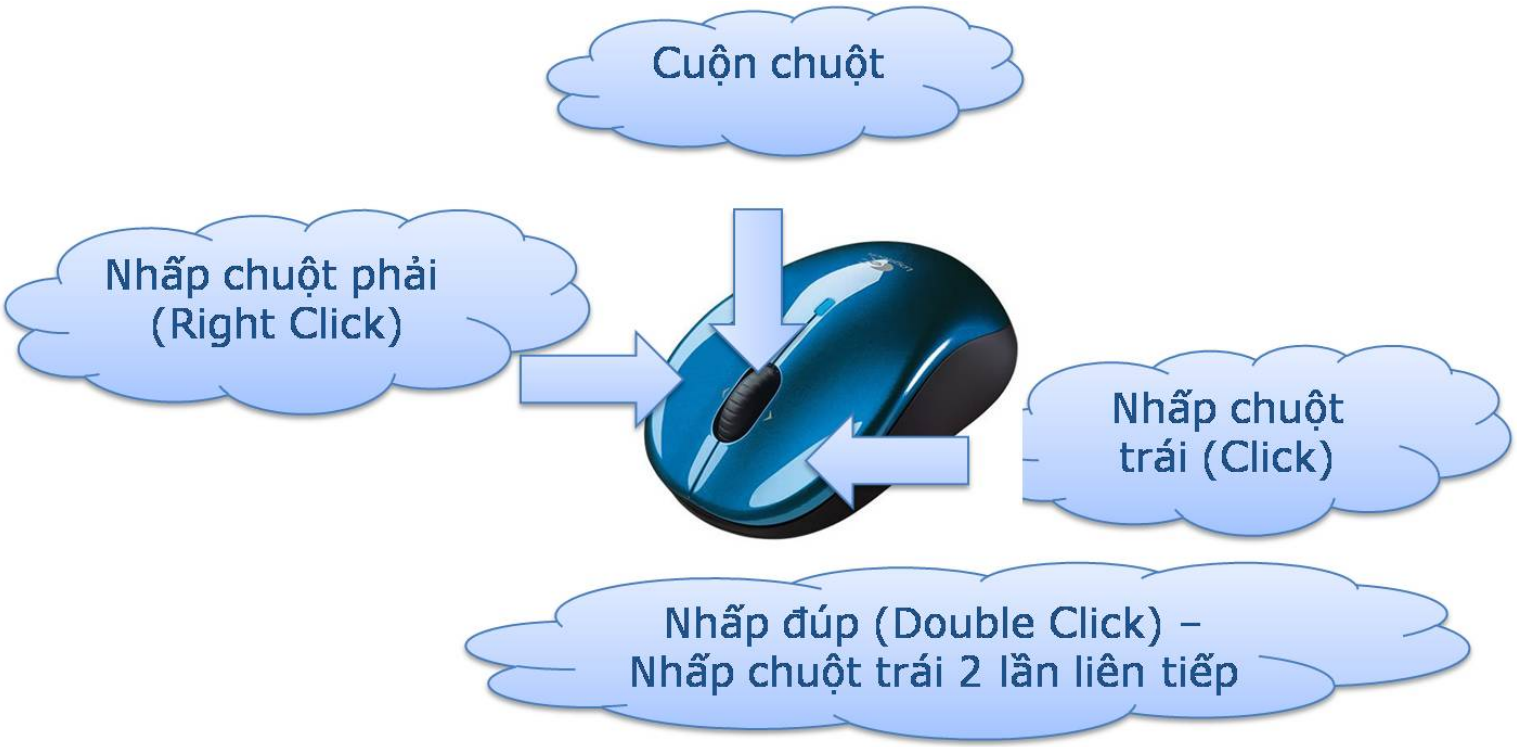
- Hộp thoại: cửa sổ nhỏ giao tiếp giữa người dùng & chương trình
- Nút : cung cấp cho người sử dụng một cách đơn giản để kích hoạt một thao tác



Sử dụng chuột

- là thiết bị vào
- Có dây hoặc không dây
- Điều khiển con trỏ chuột để tương tác với các đối tượng
- Có 2 nút bấm và có thể có 1 nút cuộn hoặc đẩy (nút giữa)
  - nút trái : chọn đối tượng, rê, kéo đối tượng, ...
  - nút phải : hiển thị danh sách công việc (menu tắt) tương ứng với vùng hoặc đối tượng được chọn
  - nút giữa : cuộn màn hình

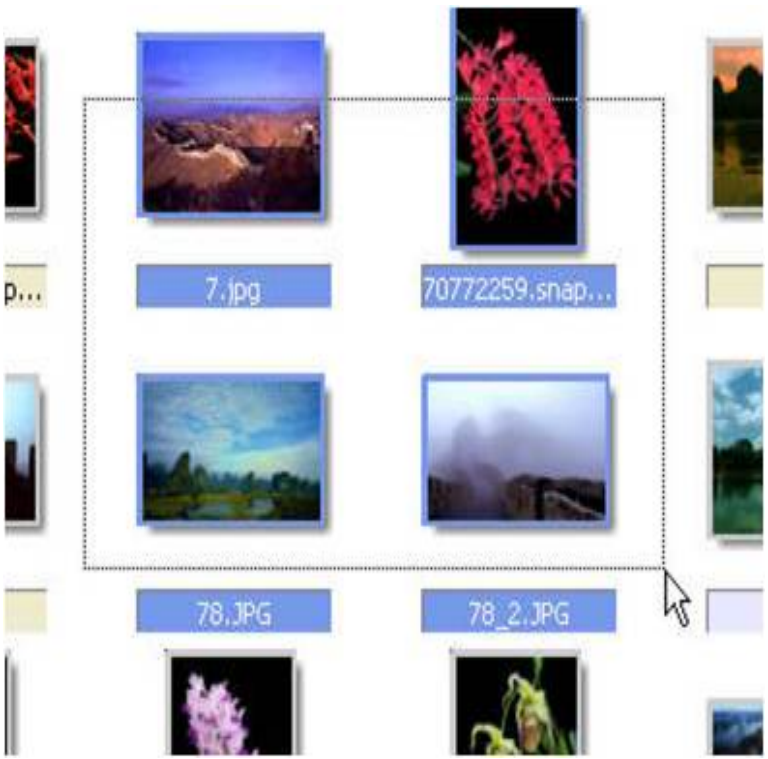
Thao tác cơ bản với chuột



- Chọn đối tượng: di chuyển con trỏ chuột đến đối tượng và nhấp trái chuột
- Kéo thả (drap and drop): chọn đối tượng $\Rightarrow$ nhấn, giữ nút chuột trái và kéo đến vị trí cần thả $\Rightarrow$ nhả chuột trái

Thao tác cơ bản với chuột

- Chọn nhiều đối tượng liên tiếp:
  - C1: nhấp chuột trái, giữ và kéo con trỏ chuột phủ hết bề mặt các đối tượng cần chọn
  - C2: chọn đối tượng đầu, vừa nhấn giữ phím SIFT vừa chọn đối tượng cuối
- Chọn nhiều đối tượng không liên tiếp: vừa nhấn giữ phím Ctrl vừa nhấp chọn từng đối tượng

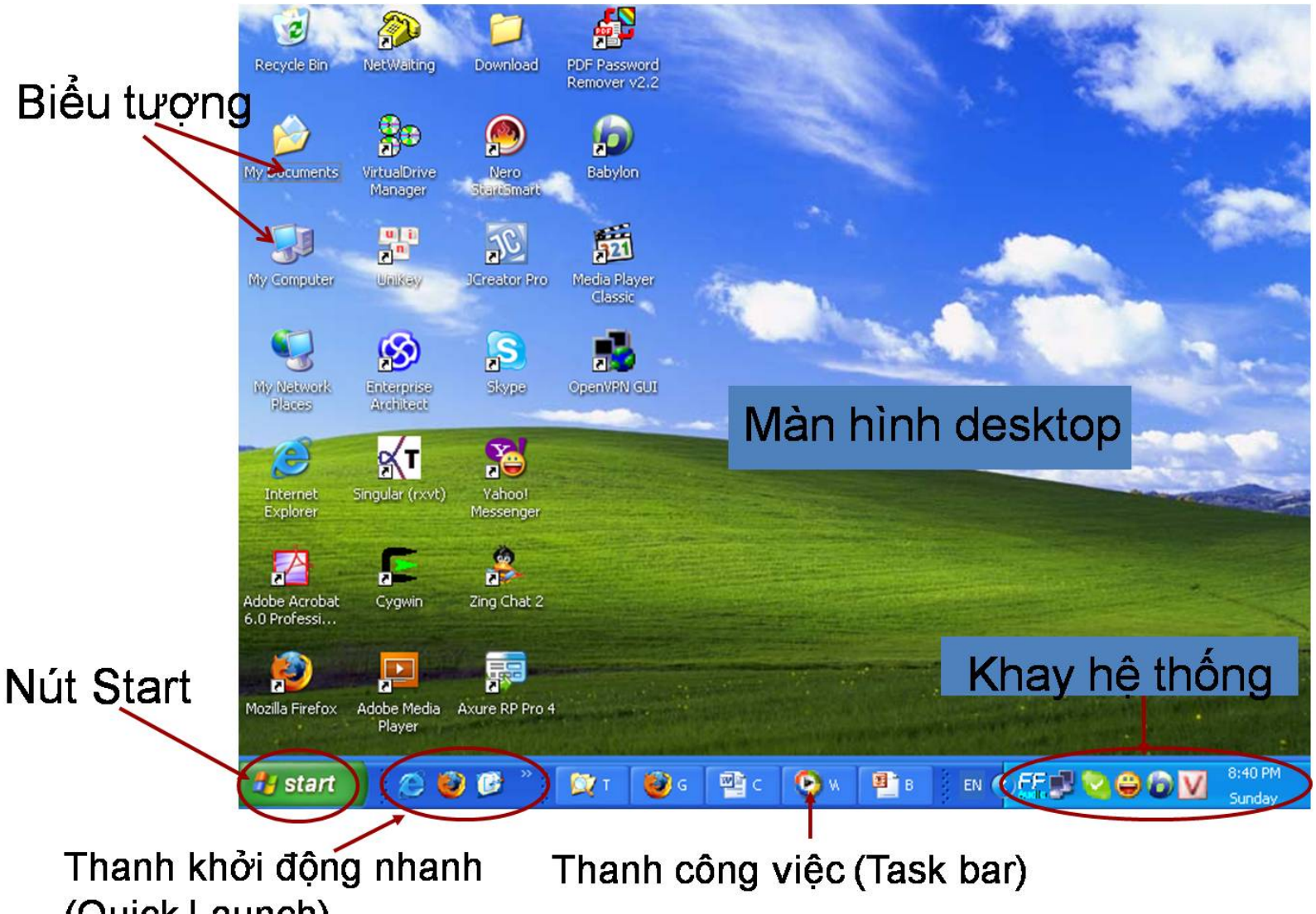


Thao tác cơ bản với chuột

Một số hình dạng thông dụng của con trỏ chuột:

Biểu tượng	Tên
	Hình dạng thông thường - **Normal Select**
	Đang bận - Busy
	Thay đổi kích cỡ theo chiều dọc - **Vertical Resize** Thay đổi kích cỡ theo chiều ngang - **Horizontal Resize**
	Thay đổi kích cỡ chéo – **Diagonal Resize**
	Chọn đoạn văn bản – **Text Select**
	Chọn mở liên kết – **Link Select**

Desktop


Biểu tượng
Màn hình desktop
Nút Start
Khay hệ thống
Thanh khởi động nhanh (Quick Launch)
Thanh công việc (Task bar)
Recycle Bin
NetWaiting
Download
PDF Password Remover v2.2
My Documents
VirtualDrive Manager
Nero StartSmart
Babylon
My Computer
Unikey
JCreator Pro
Media Player Classic
My Network Places
Enterprise Architect
Skype
OpenVPN GUI
Internet Explorer
Singular (rxvt)
Yahoo! Messenger
Adobe Acrobat 6.0 Professi...
Cygwin
Zing Chat 2
Mozilla Firefox
Adobe Media Player
Axure RP Pro 4
start
EN
8:40 PM
Sunday

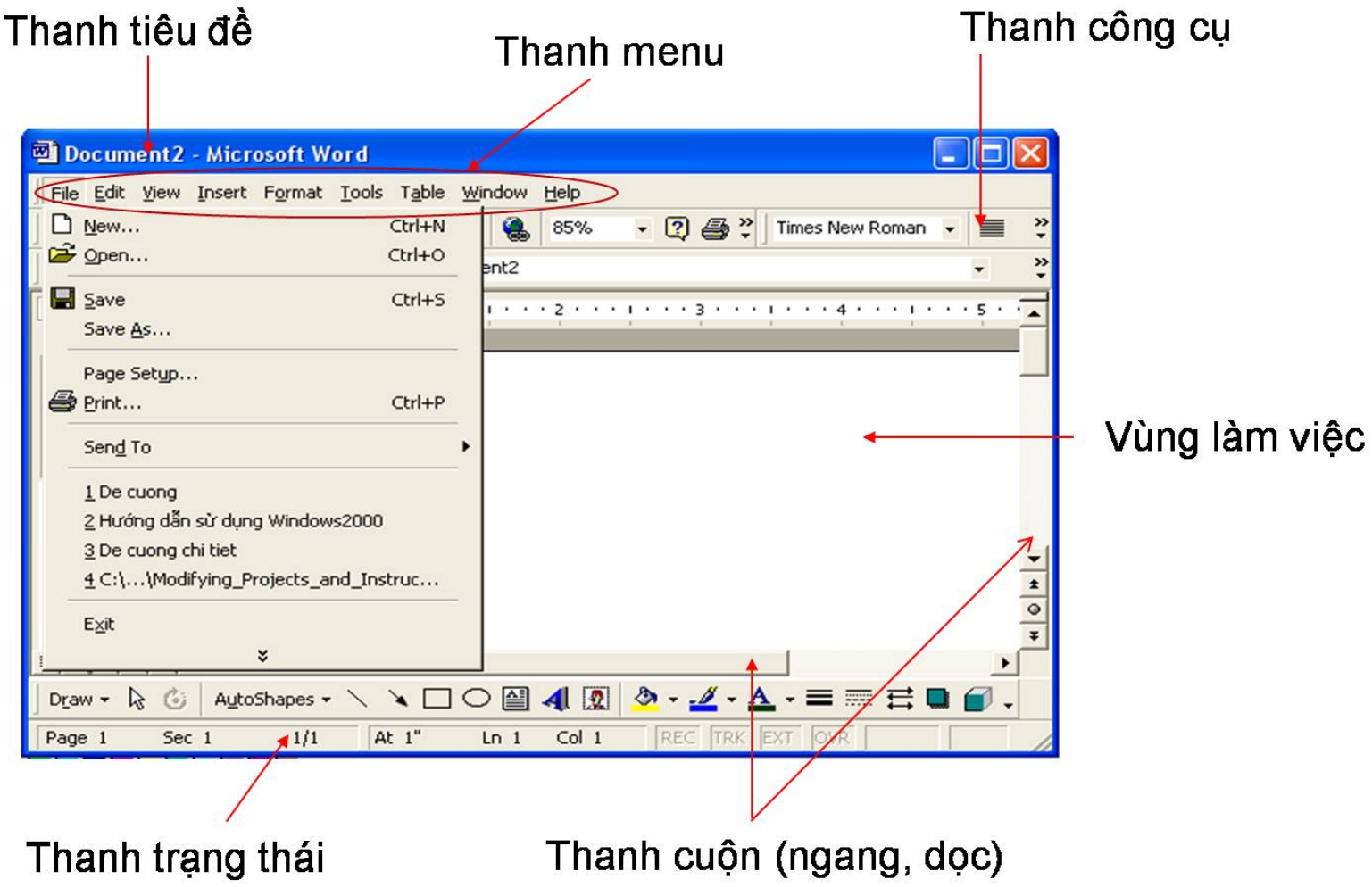
Desktop

Nhấp đúp chuột vào các biểu tượng để MỞ chương trình ứng dụng hoặc các cửa sổ tương ứng

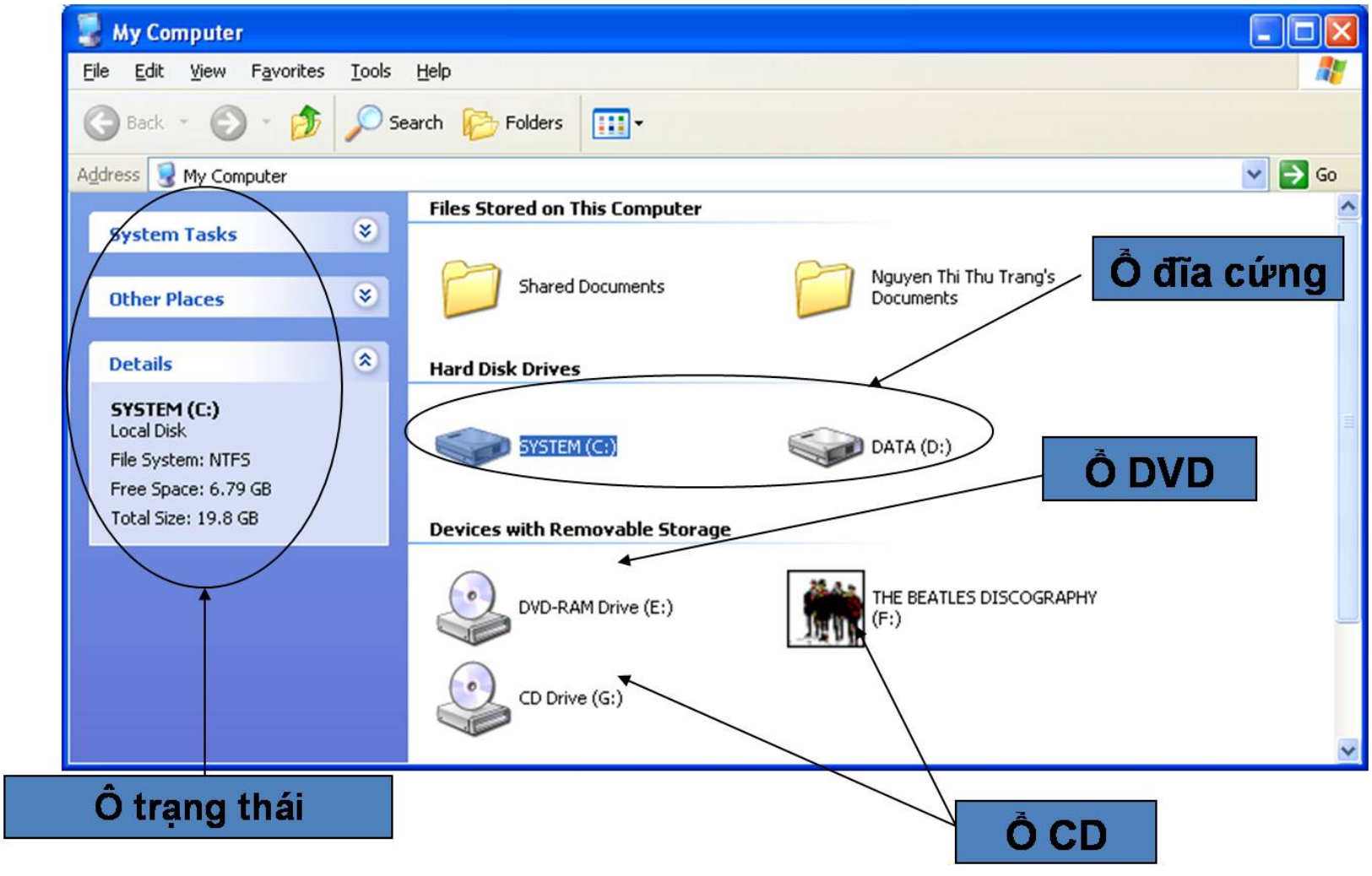
Desktop

- Có thể thay đổi, sắp xếp lại các biểu tượng trên desktop
- Có thể thay đổi nền của màn hình desktop
- Có thể thêm, xóa các biểu tượng vào thanh khởi động nhanh
- ...

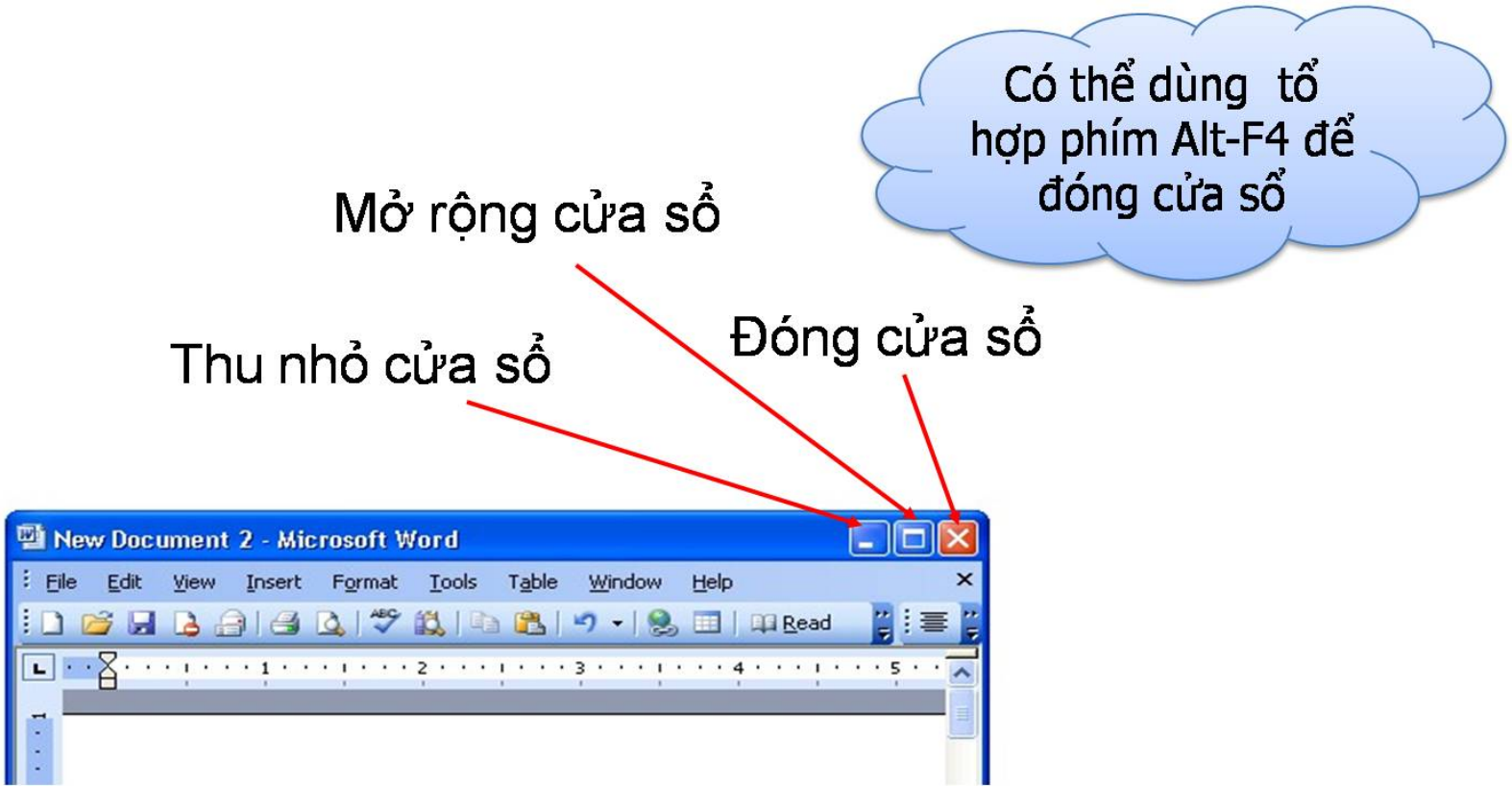
Làm việc với cửa sổ



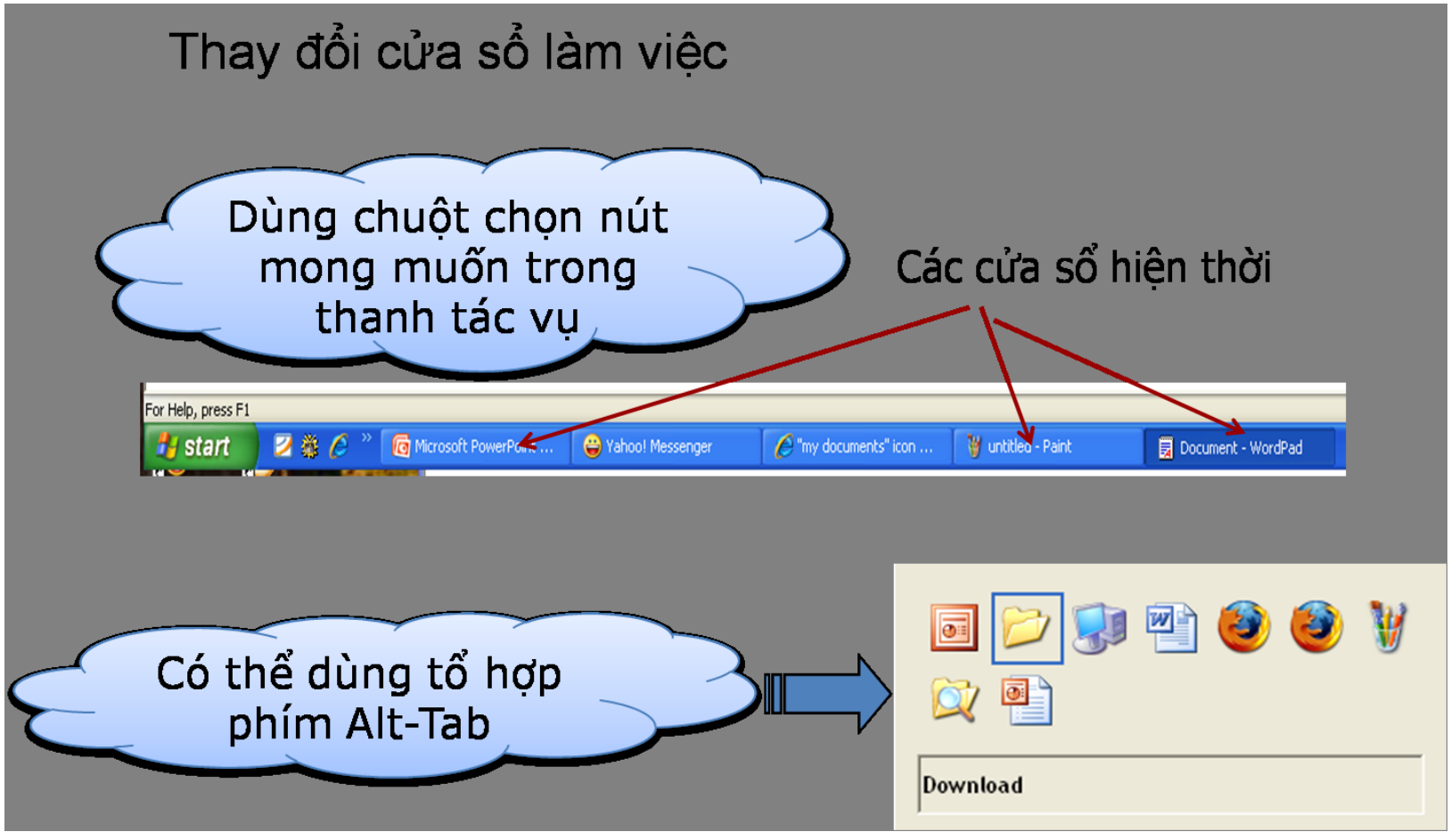
Làm việc với cửa sổ



Làm việc với cửa sổ



Làm việc với cửa sổ



Cấu hình Windows (*Control Panel*)

Một chương trình cho phép: xem và chỉnh sửa các tham số của hệ thống máy tính, thiết lập hoặc thay đổi cấu hình, thiết lập phần cứng, phần mềm, ...

Start ⇒ Settings ⇒ Control Panel

- Cài đặt và loại bỏ Font chữ
- Thay đổi dạng hiện màn hình desktop
- Cài đặt và loại bỏ chương trình
- Cấu hình ngày, giờ cho hệ thống
- Thay đổi thuộc tính của bàn phím và chuột
- Thay đổi thuộc tính vùng (Regional Settings)
- Cài đặt / loại bỏ máy in
- ...

  (chi tiết xem thêm trong tài liệu)

Windows Explorer

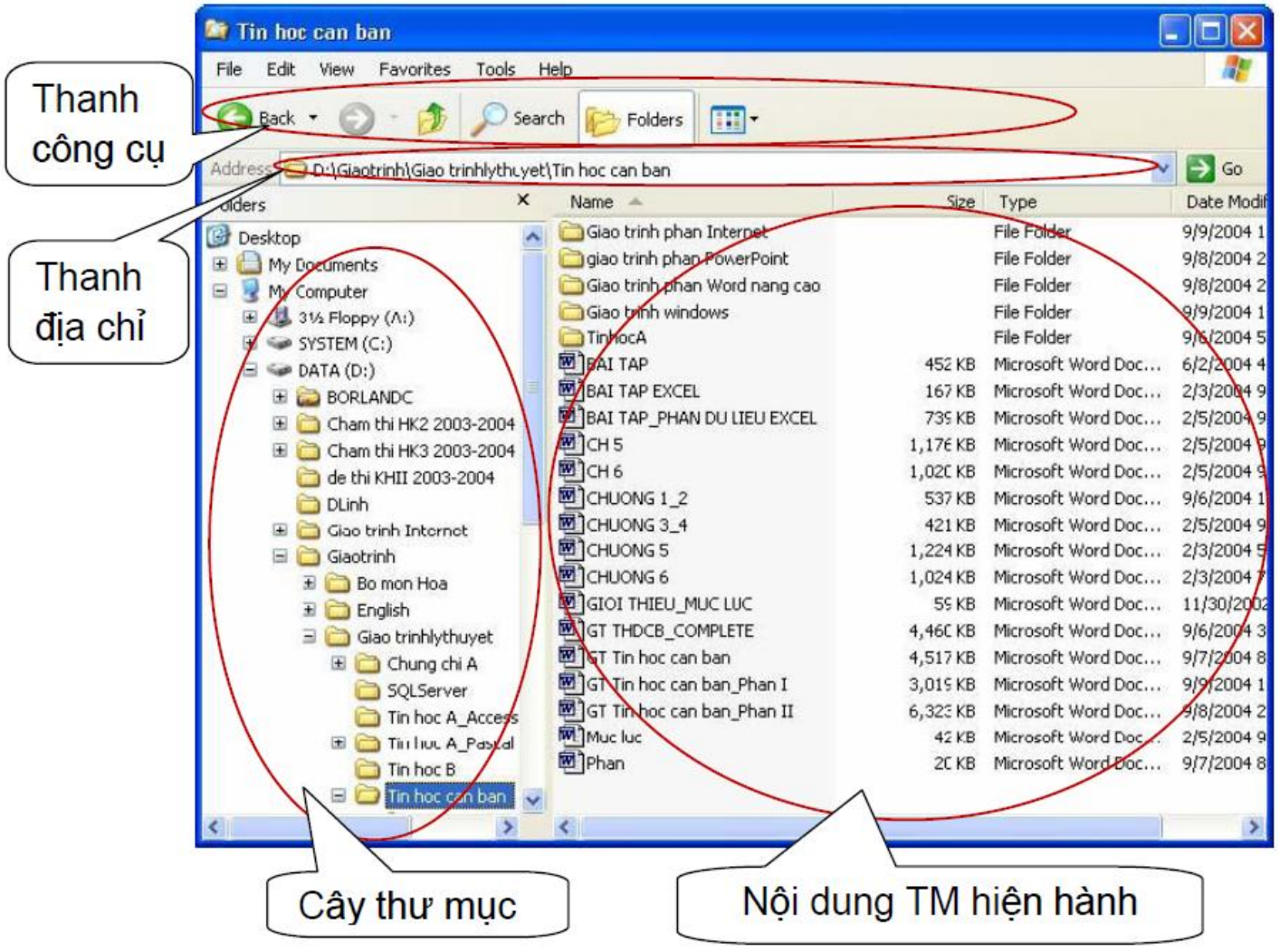
- Windows Explorer là một chương trình được hỗ trợ từ phiên bản Windows 95 cho phép người sử dụng thao tác với các tài nguyên có trong máy tính (*tập tin, thư mục, ổ đĩa, ...*) cũng như tài nguyên chia sẻ của các máy tính trong hệ thống mạng (nếu có nối mạng)
- Windows Explorer : các thao tác như *sao chép, xóa, đổi tên thư mục và tập tin,...* được thực hiện một cách thuận tiện và dễ dàng

Windows Explorer

Khởi động: 1 trong các cách sau:

-  + E
- Start ⇒ All Programs ⇒ Accessories ⇒ Windows Explorer
- Nhấp phải chuột (R_Click) lên Start ⇒ Explorer
- Nhấp phải chuột (R_Click) lên biểu tượng My Computer ⇒ Explorer

Windows Explorer



Thao tác với tệp, thư mục

- Mở tập tin/thư mục:
  - D_Click
  - hoặc R_Click → Open hoặc Chọn → nhấn Enter
- Chọn tập tin/thư mục: Click chuột lên biểu tượng
- Tạo thư mục: Chọn nơi chứa thư mục →
  - R_Click → New → Folder → nhập tên → nhấn Enter
  - hoặc dùng menu: File/New/Folder
- Sao chép thư mục/tập tin:
  - Nhấn Ctrl và Kéo (Drag) đối tượng đến nơi cần chép
  - hoặc Chọn → Ctrl + C → đến nơi cần chép → Ctrl +V
- Di chuyển thư mục/tập tin:
  - Drap and Drop
  - hoặc Chọn → Ctrl + X → đến nơi cần chép → Ctrl +V

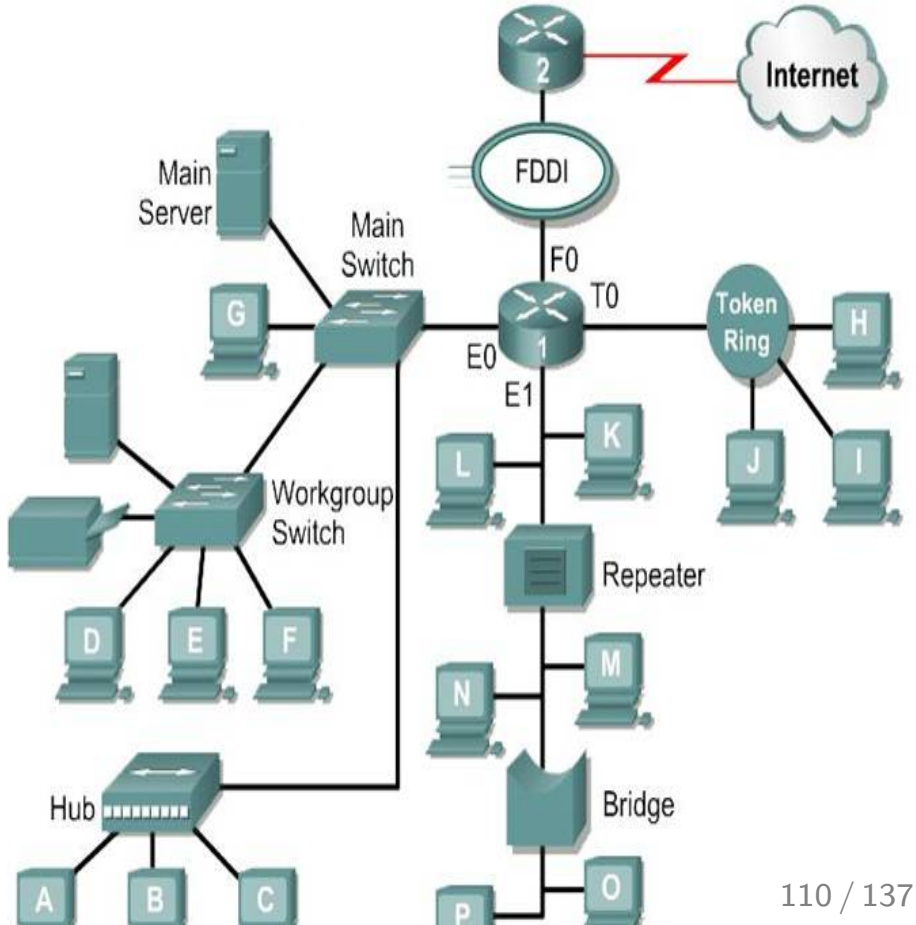
Thao tác với tệp, thư mục

- Xóa thư mục/tập tin:
  - Chọn → nhấn phím Delete
  - hoặc R_Click → Delete hoặc dùng menu File → Delete
- Phục hồi thư mục tập tin: D_Click lên biểu tượng Recycle Bin → chọn đối tượng cần phục hồi →
  - R_Click → Delete
  - hoặc dùng menu File → Restore
- Đổi tên thư mục/tập tin:
  - Chọn đối tượng
  - File/ Rename hoặc Nhấn F2 hoặc R_Click & chọn Rename
  - Nhập tên mới & gõ Enter để kết thúc
- Thay đổi thuộc tính tập tin/thư mục:
  - Chọn đối tượng → R_Click chọn → Properties
  - thay đổi thuộc tính → nhấn Apply

Khái niệm

là một tập hợp gồm nhiều máy tính hoặc thiết bị xử lý thông tin được kết nối với nhau qua các đường truyền

- Mục đích:
  - trao đổi thông tin giữa các máy tính
  - chia sẻ tài nguyên
- Ví dụ:
  - Mạng tại Trung tâm Máy tính, Viện CNTT &TT, Trường ĐHBK Hà Nội
  - Mạng LAN của quán Game
  - Mạng Internet



Lịch sử phát triển

- 1950s: máy tính xuất hiện
- 1960s: mạng máy tính xuất hiện. Ban đầu: 1 máy tính lớn kết nối nhiều trạm cuối (terminal)
- 1970s: mạng máy tính : kết nối các máy tính độc lập
- Quy mô và độ phức tạp của mạng càng tăng

Theo mối quan hệ giữa các máy trong mạng

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

Theo quy mô địa lý

- LAN (*Local Area Network*):
  - mạng cục bộ, trong phạm vi nhỏ (ví dụ bán kính từ vài mét đến 1km)
  - số máy tính không quá nhiều, mạng không quá phức tạp
  - VD: mạng tại quán Internet, Game, mạng công ty, trường học, ...
- WAN (*Wide Area Network*)
  - mạng diện rộng
  - các máy tính có thể ở các thành phố khác nhau (bán kính: vài trăm đến vài ngàn km)
  - VD: mạng của Tổng cục thuế, bộ công an, ...
- GAN (*Global Area Network*):
  - mạng toàn cầu, kết nối máy tính ở nhiều nước khác nhau
  - thường là kết hợp của nhiều mạng con
  - VD: mạng Internet

Các thành phần cơ bản - Phần cứng

- Máy tính : nút mạng
- Vỉ mạng (card mạng) (*Network Interface Card, NIC*)
- Các thiết bị kết nối mạng: HUB, SWITCH, ROUTER, ...
- Đường truyền: có dây hoặc không dây

Các thành phần cơ bản - Phần cứng

Cạc mạng

Cạc mạng
(không dây)

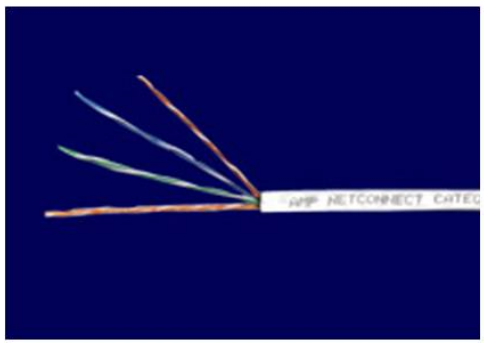
Bộ chuyển mạch
(switch)

Bộ định tuyến
(router)

Các thành phần cơ bản - Phần cứng

Đường truyền:

- Môi trường truyền thông giữa các máy tính
- hữu tuyến (cáp truyền) hoặc vô tuyến (ăng ten thu-phát)
- không dây (*wireless (wifi)*)



Cáp đồng

Cáp quang

Ăng-ten

Các thành phần cơ bản - Phần mềm

- Hệ điều hành mạng: phần mềm điều khiển hoạt động của mạng
- Các phần mềm mạng cho máy tính: thường đã có sẵn trong hệ điều hành
- Các ứng dụng trên mạng: Email, hệ quản trị CSDL, ...

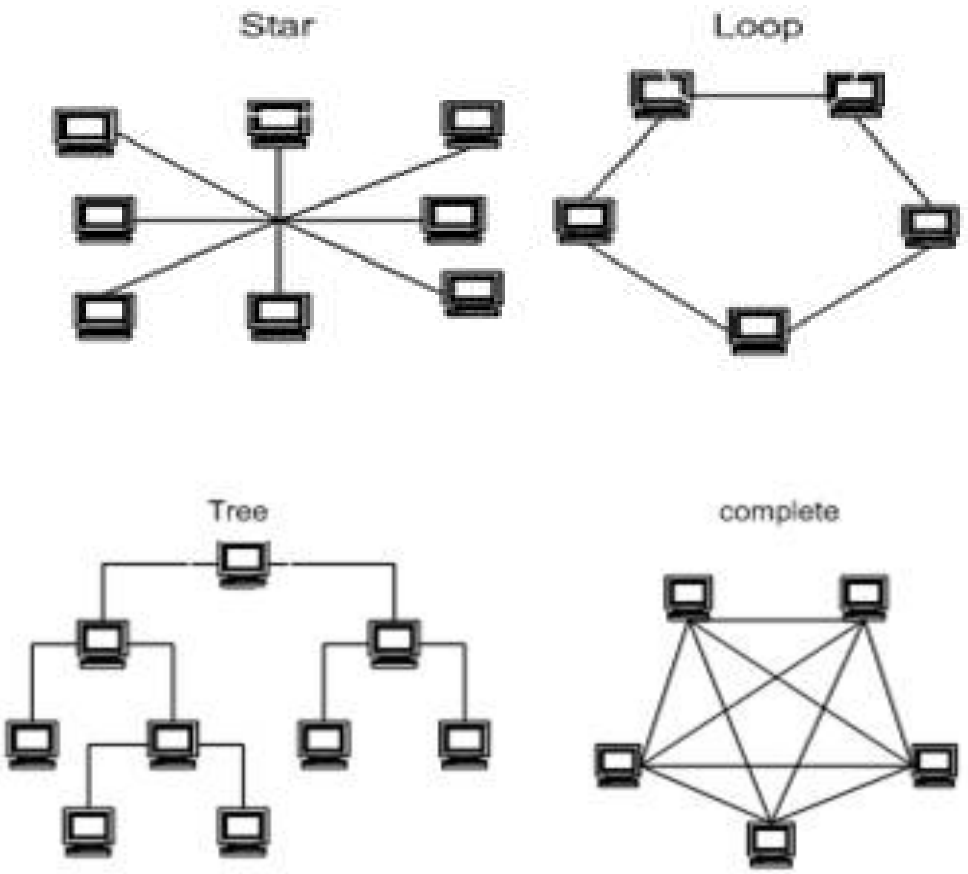
Kiến trúc mạng máy tính (Network architecture)

thể hiện :

- cách kết nối máy tính với nhau (*topology*): điểm-điểm (point-to-point) và quảng bá (broadcast)
- qui ước truyền dữ liệu (giao thức- *protocol*) giữa các máy tính

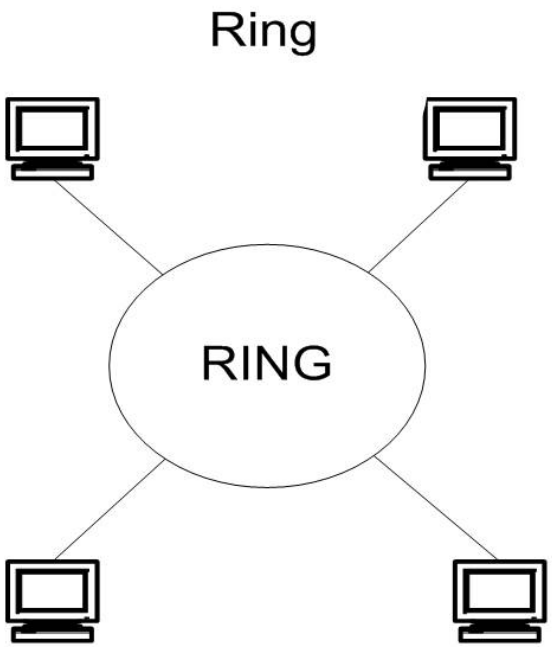
Kiến trúc mạng máy tính

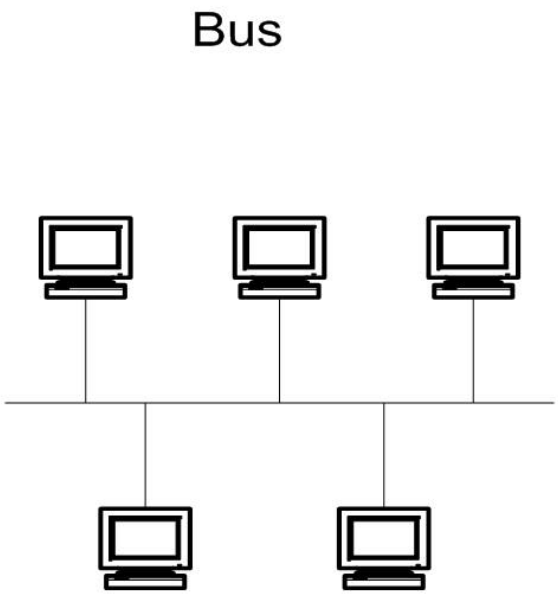
Điểm-điểm: các đường truyền nối các nút thành từng cặp



Kiến trúc mạng máy tính

Quảng bá:
các nút nối vào đường truyền chung $\Rightarrow$ dữ liệu truyền đi có địa chỉ đích $\Rightarrow$ máy nhận được kiểm tra xem địa chỉ đích có phải mình không





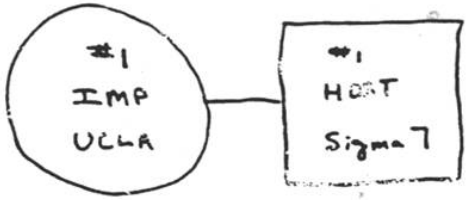
Khái niệm

- Internet là một mạng máy tính có quy mô toàn cầu (GAN), gồm rất nhiều mạng con và máy tính nối với nhau bằng nhiều loại phương tiện truyền
- Internet không thuộc sở hữu của ai cả. Chỉ có các uỷ ban điều phối và kỹ thuật giúp điều hành Internet
- Ban đầu là mạng ARPANET của Bộ Quốc phòng Mỹ (DoD)

Các dịch vụ chính

- Các dịch vụ chính:
  - 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:
  - truyền tin, phổ biến tin, thu thập tin, trao đổi tin nhanh chóng, thuận tiện, rẻ tiền
  - yếu tố không thể thiếu trong thời đại ngày nay ở mọi lĩnh vực

Sự phát triển của Internet



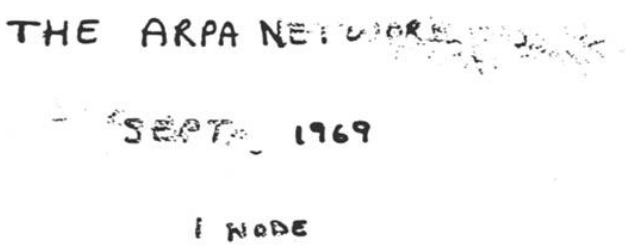
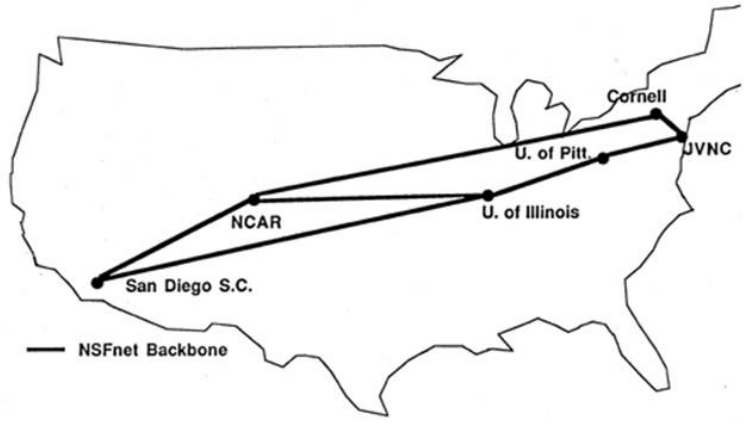
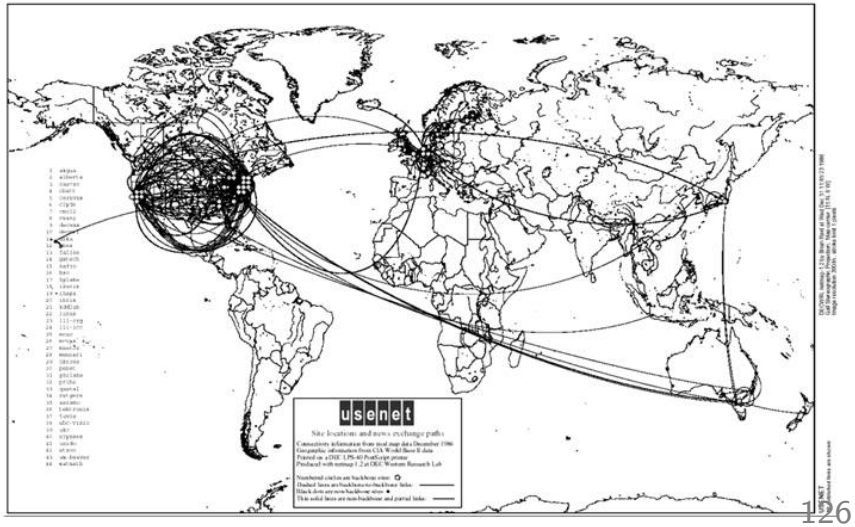


FIGURE 6.1 Drawing of September 1969 (Courtesy of Alex McKenzie)

Ý tưởng tại phòng thí nghiệm của ARPA (9/1969)





Sự phát triển của Internet

1974: khái niệm "Internet" xuất hiện

1983: ARPANET tách thành MILNET(quân đội) và NSFnet (nghiên cứu)

1987: NSFnet được mở cửa cho các cá nhân

1988: Internet hình thành

1997: Việt Nam kết nối Internet

http://www.youtube.com/watch?v=9hIQjrMHTv4

Làm sao để có được dịch vụ internet

- Máy tính kết nối với Modem/Router/USB3G
- Có thuê bao kết nối Internet với tài khoản đăng ký với 1 nhà cung cấp dịch vụ (Internet Service Provider - ISP)
  - Hình thức: qua đường điện thoại, đường thuê riêng, đường truyền hình cáp
  - Nhà cung cấp dịch vụ Internet: VNN, Viettel, FPT, EVN, Truyền hình cáp Việt Nam/Hà nội/TP HCM, ...
- Cài đặt các phần mềm Internet thông dụng:
  - trình duyệt web như IE, **FireFox**, **Google Chrome**, . . .
  - phần mềm nói chuyện trực tuyến: **Yahoo! Messenger**, **Skype**, **Gtalk**, Gaim, Pidgin, ...

Cơ bản về Word Wide Web (WWW)

- Gọi tắt là Web, ứng dụng chạy trên Internet
- Là một hệ thống bao gồm các tài liệu siêu văn bản (hypertext) có thể truy cập qua Internet
- Các tài liệu có thể chứa các liên kết tới các tài liệu khác có thể trên máy tính khácở bất kỳ nơi nào trên thế giới

Một số giao thức

- Giao thức (*Protocol*): Tập các qui ước truyền thông
- HTTP (*HyperText Transfer Protocol*): Giao thức được sử dụng để truyền các trang Web (siêu văn bản)
- HTTPS (*HyperText Transfer Protocol Secure*): Giao thức bảo mật cho các giao dịch bí mật (confidential)
- FTP (*File Transfer Protocol*): giao thức truyền tệp tin
- SMTP (*Simple Mail Transfer Protocol*) : giao thức truyền thư tín đơn giản
- POP3 (*Post Office Protocol 3*), IMAP (*Internet Message Access Protocol*): giao thức để lấy thư điện tử từ server
- ...

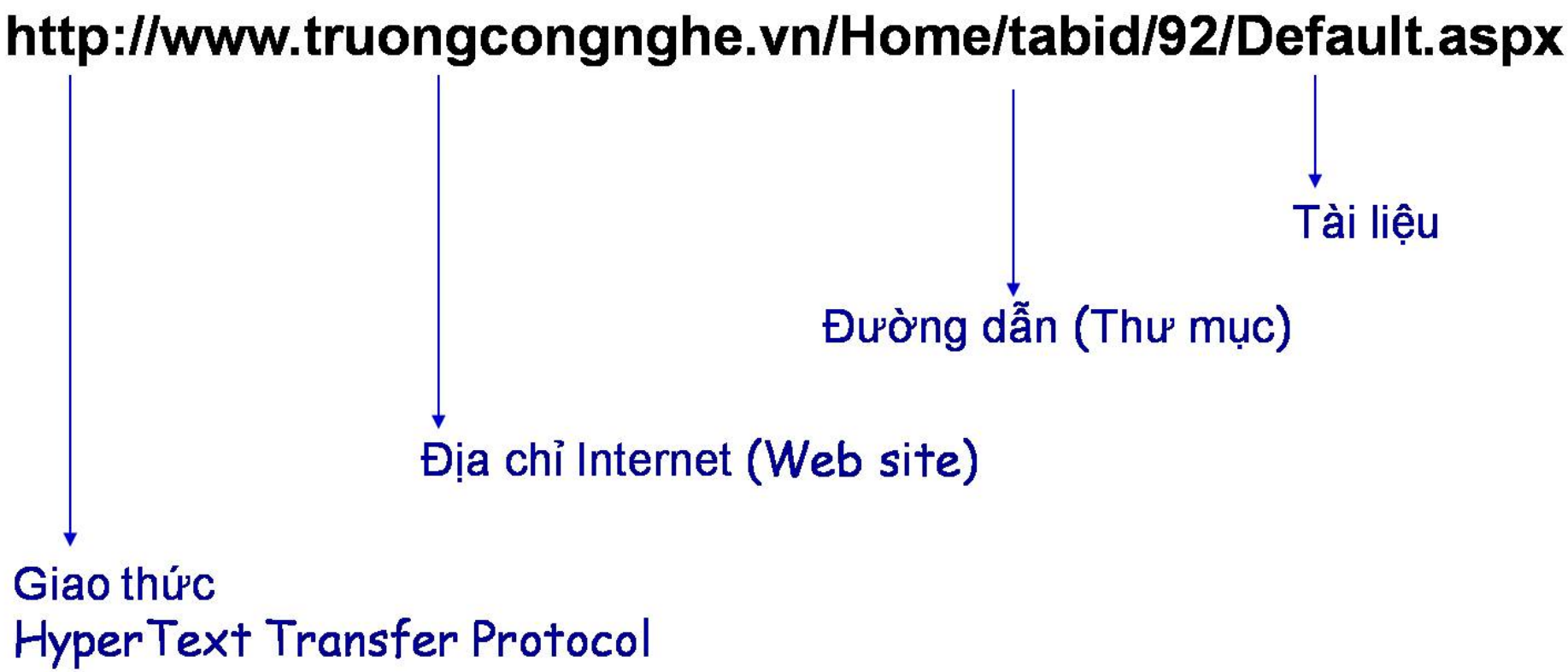
Website và URL

- Website: Tập hợp nhiều trang web (webpage) được đặt trên 1 máy tính trong mạng Internet và có chung một địa chỉ Internet

  ví dụ: `http://www.hut.edu.vn` , `http://www.google.com`

- Một trang web có một địa chỉ URL duy nhất, URL (Uniform Resource Locator):
  - tham chiếu đến các tài nguyên trên Internet
  - địa chỉ của các tài nguyên trên Internet

Cấu trúc URL


http://www.truongcongnghe.vn/Home/tabid/92/Default.aspx
Tài liệu
Đường dẫn (Thư mục)
Địa chỉ Internet (Web site)
Giao thức
HyperText Transfer Protocol

Một số website phổ biến

- Tìm kiếm: giúp tìm kiếm nhanh chóng các thông tin, tài liệu liên quan đến một hoặc vài từ khóa hoặc các tiêu chí khác
  - http://www.google.com : trang tìm kiếm khổng lồ và được sử dụng nhiều nhất, thường xuyên được nâng cấp, cải tiến.
  - http://scholar.google.com : hỗ trợ tìm kiếm chuyên về các tài liệu khoa học.
  - http://www.search.yahoo.com/
  - http://baamboo.com/
  - http://www.altavista.com/
  - http://www.tineye.com/, http://www.gazopa.com/, http://labs.ideeinc.com/, ...: tìm kiếm ảnh dựa trên nội dung

Một số website phổ biến

- từ điển trực tuyến:
  - Việt-Anh/Pháp/Nhật/,...: http://www.vdict.com, http://www.tratu.baamboo.com, ...
  - Anh-Anh: http://dictionary.msn.com
  - Anh-Pháp-Nhật -Hàn-Đức-Tây Ban Nha-...: http://www.wordreference.com
- Từ điển Bách khoa toàn thư mở:
  - cung cấp các khái niệm, giải thích về các từ hoặc cụm từ nào đó, được *chính những người duyệt web xây dựng nội dung*
  - www.wikipedia.org
  - Tiếng Việt: www.vi.wikipedia.org

Một số website phổ biến

- mạng giáo dục: http://www.edu.net.vn
- thư điện tử miễn phí: http://mail.google.com, http://mail.yahoo.com, http://www.hotmail.com, ...
- nghe nhạc trực tuyến: http://mp3.zing.vn, http://www.nghenhac.info, http://www.nhacso.net, ...
- đọc truyện trực tuyến: http://www.vnthuquan.net, ...
- báo điện tử: http://www.vnexpress.net, http://www.vietnamnet.vn, http://www.dantri.com.vn, www.24h.com.vn, http://www.hanoimoi.com.vn, ...
- xem truyền hình trực tuyến: http://www.vtc.com.vn/, ...
- xem video: http://www.youtube.com, ...

Một số website phổ biến

- Chia sẻ dữ liệu trực tuyến
  - chia sẻ dữ liệu lớn cho nhiều người
  - nếu dùng miễn phí: hạn chế dung lượng tối đa (100MB), thời gian tối đa thường 7 đến 10 ngày
  - ví dụ:
    - http://www.yousendit.com/
    - http://www.esnips.com/
    - http://www.box.net/
    - www.rapidshare.com
    - www.megaupload.com
    - ...

Questions & Answers

Tin học đại cương

Bài 3: Các hệ thống ứng dụng

NGUYỄN Thị Oanh

oanhnt@soict.hut.edu.vn

Bộ môn Hệ thống thông tin - Viện CNTT và Truyền Thông
Đại học Bách Khoa Hà nội

2010 - 2011

Nội dung

Khái niệm

- *Khái niệm 1*: HTTTQL là một **lĩnh vực khoa học quản lý** nhằm nghiên cứu việc phát triển, ứng dụng, duy trì các HTTT vi 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**
- *Khái niệm 2*: 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

Phân loại

- Phân loại theo cấp bậc quản lý
- Phân loại theo chức năng nghiệp vụ
- Phân loại theo quy mô tích hợp

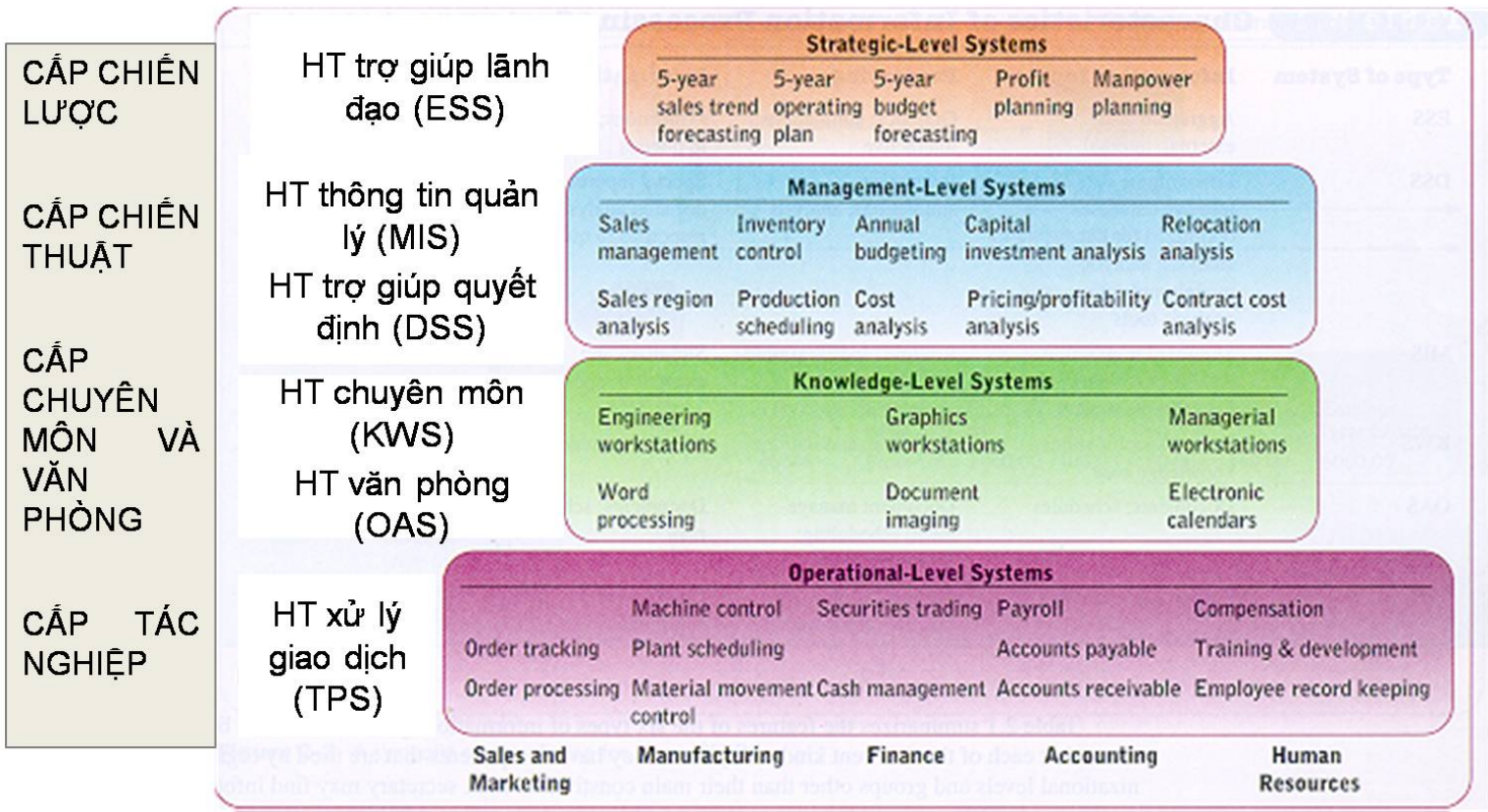
Phân loại theo cấp bậc quản lý

4 cấp từ thấp đến cao:

- Cấp tác nghiệp
- Cấp chuyên gia và văn phòng
- Cấp chiến thuật
- Cấp chiến lược

Phân loại theo cấp bậc quản lý

Mỗi cấp là các loại HTTTQL riêng biệt



Hệ thống xử lý giao dịch (TPS - Transaction Processing Systems)

- 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 - Office Automation Systems)

- 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 - Knowledge Work Systems)

- là hệ thống hỗ trợ lao động có trình độ cao trong công việc chuyên môn hàng ngày cuả 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 hoạ, kế hoạch*
  - người dùng: **chuyên gia, kỹ thuật viên**

Hệ thống trợ giúp ra quyết định (DSS - Decision Support Systems)

- là hệ thống 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 - Management Information Systems)

- 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 HT 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 - Executive Support Systems)

- 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**

Phân loại theo chức năng nghiệp vụ

Hệ thống quản lý Marketing

- là hệ thống trợ giúp các hoạt động của chức năng marketing
- Ví dụ về HTQL 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 DL về nhân khẩu, thị trường thái độ của người tiêu dùng và các xu hướng	Chuyên môn và văn phòng
Phân tích giá cả	Đánh giá cho sản phẩm hoặc dịch vụ	Chiến thuật
Dự báo chiều hướng doanh số	Chuẩn bị kế hoạch 5 năm dự báo doanh số	Chiến lược

Hệ thống quản lý sản xuất

- hệ thống trợ giúp các hoạt động cuả chức năng sản xuất
- Ví dụ về HTTTQL 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 nên sản xuất	Chiến thuật
Định vị khu sản xuất	Quyết định đặt các khu sản xuất mới ở đâu	Chiến lược

Hệ thống quản lý tài chính kế toán

- hệ thống trợ giúp các hoạt động cuả chức năng tài chính, kế toán
- Ví dụ về HTTTQL 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	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

- hệ thống trợ giúp các hoạt động cuả chức năng tổ chức, nhân sự
- Ví dụ về HTTTQL 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, phúc lợi	Chiến thuật
Hoạch định nhân sự	Hoạch định nhu cầu về nhân sự lâu dài của doanh nghiệp	Chiến lược

Phân loại theo quy mô tích hợp

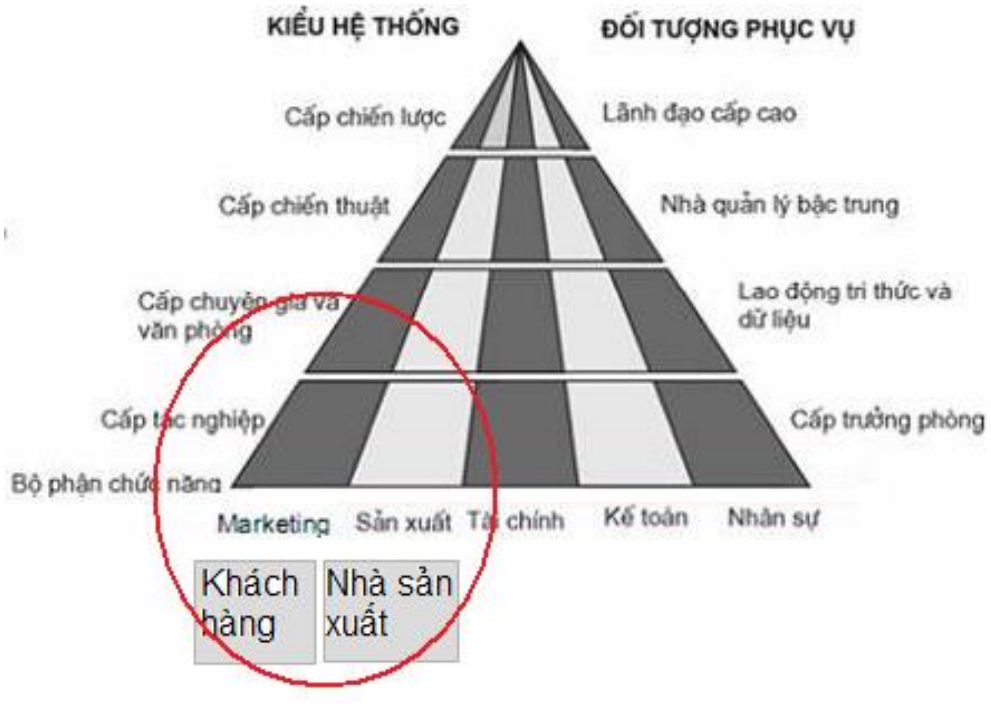
- Khái niệm: HT doanh nghiệp tích hợp là những hệ thống *liên kết xuyên suốt* nhiều bộ phận *chức năng, cấp bậc tổ chức* và *đơn vị* kinh doanh

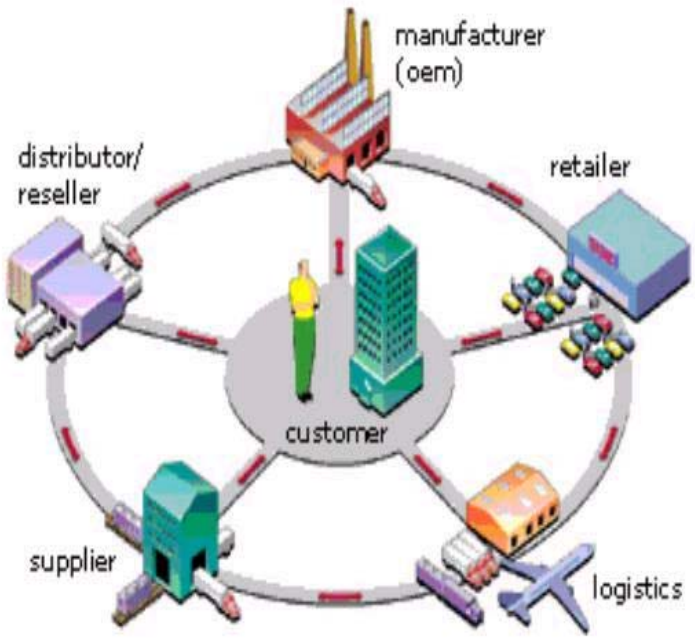
Hệ thống quản lý nguồn lực (ERP - Enterprise Resource Planning)

- là hệ thống tích hợp và phối hợp hầu hết các quy trình tác nghiệp chủ yếu cuả doanh nghiệp
- là hệ thống ứng dụng đa phân hệ giúp tổ chức, doanh nghiệp quản lý các nguồn lực và điều hành tác nghiệp

Hệ thống quản lý chuỗi cung ứng (SCM - Supply Chain Management)

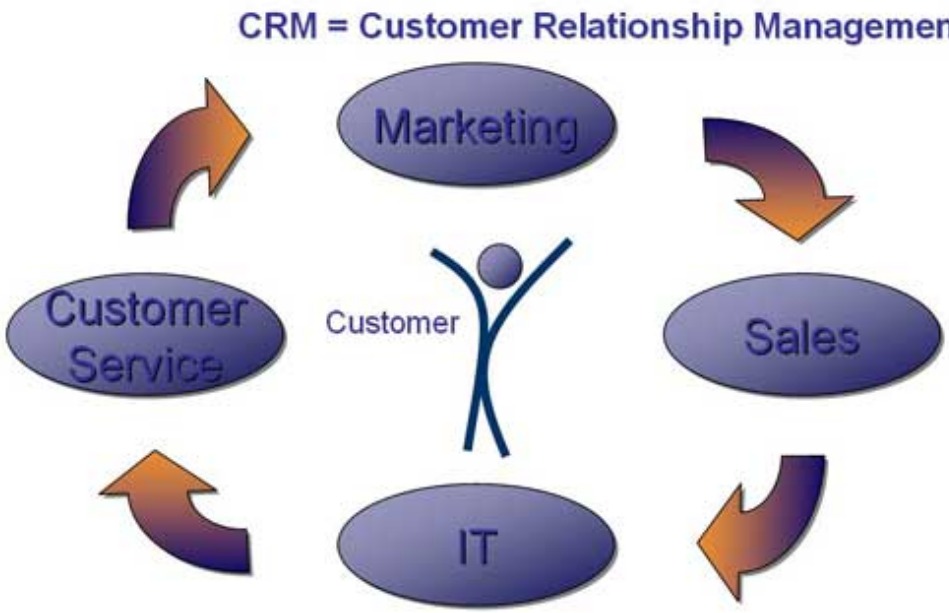
là hệ thống tích hợp giúp quản lý và liên kết các bộ phận **sản xuất**, **khách hàng** và **nhà cung cấp**





Hệ thống quản lý quan hệ khách hàng (CRM - Customer Relationship Management)

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 - Knowledge Management)

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

Hệ thông tin bảng tính

- Máy tính: Hỗ trợ việc tính toán, nhất là kế toán và phân tích thống kê
- Phần mềm trợ giúp tính toán thông dụng: Phần mềm bảng tính (PMBT) (*spreadsheet software*)
- PMBT: giúp tính toán các số liệu, từ đó cho phép xây dựng và làm việc với những tình huống mô phỏng thế giới thực

Hệ thông tin bảng tính

- **Bảng tính - phần mềm của dự toán**
  - Tạo **thay đổi lớn** trong hoạt động kinh doanh
  - Giúp **thao tác với con số**, các phương thức khó làm bằng tay hoặc nhàm chán
  - **Rút ngắn thời gian** thực hiện, **giảm nhầm lẫn**
  - Giúp khám phá **mối liên hệ giữa các con số** => cơ sở dự đoán tương lai

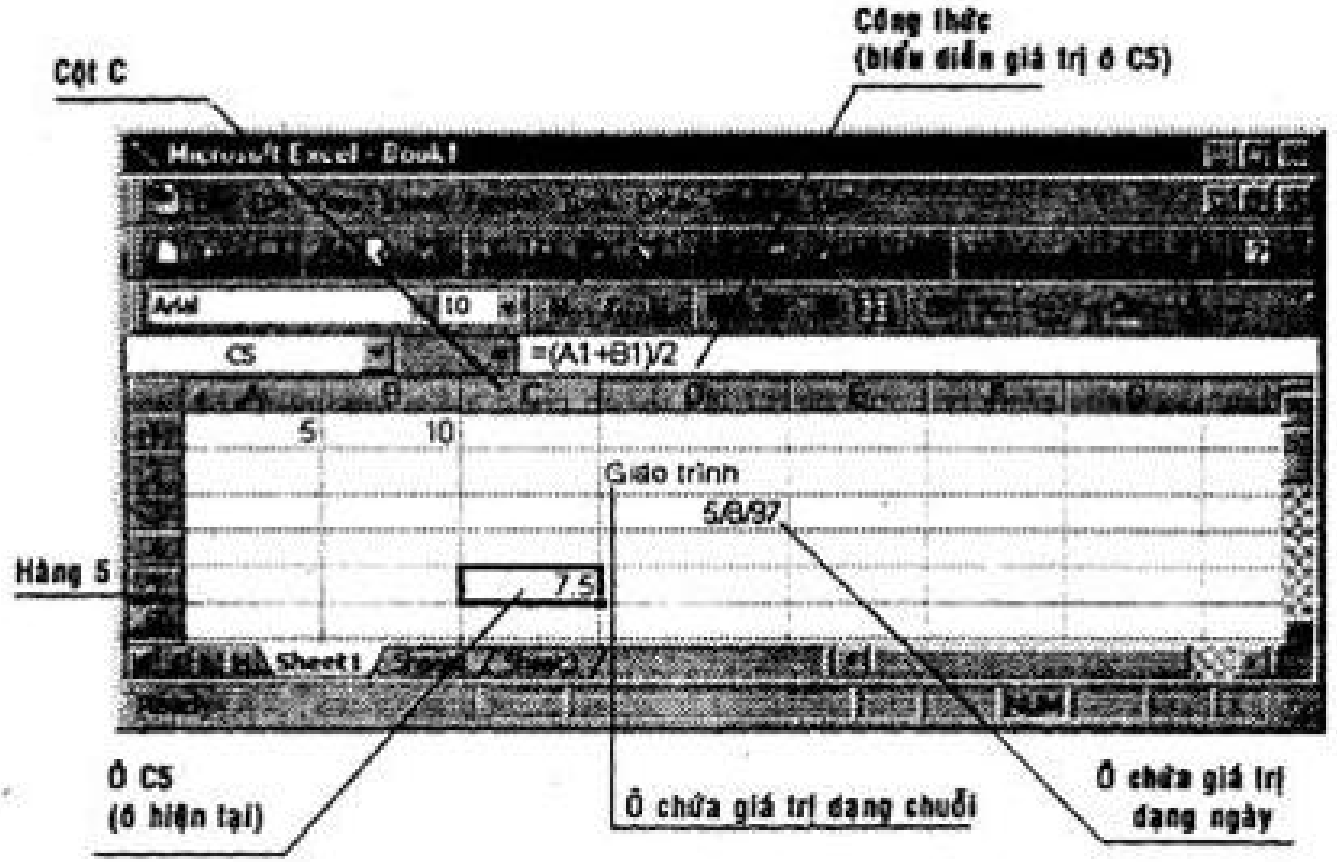
Hệ thông tin bảng tính

- **Bảng tính: những ô lưới linh động**
  - Dạng ô lưới gồm: các hàng đánh số từ 1 và các cột đánh số từ chữ A
  - **Ô** là giao của 1 hàng và 1 cột, được đánh **địa chỉ**. Ví dụ ô A1 là giao của hàng 1 và cột A
  - Mỗi ô có thể chứa dữ liệu dạng **số**, **chuỗi kí tự** hoặc **công thức** hiển thị liên hệ giữa các con số
  - **Giá trị số** là vật liệu thô để tính toán

Hệ thông tin bảng tính

- **Bảng tính: những ô lưới linh động**
  - Để dễ hiểu ý nghĩa của các con số => dùng thêm chuỗi **tiêu đề** (label) ở đầu các cột hoặc bên trái các hàng
    - Tiêu đề là dạng **chỉ dẫn người đọc** chứ KHÔNG phải yêu cầu máy tính thực hiện
  - Để thực hiện tính toán => **cần đưa vào công thức** hướng dẫn máy tính thực hiện

Hệ thông tin bảng tính



Demo

Địa chỉ tuyệt đối và tương đối

Để tham chiếu đến 1 ô trong bảng tính: sử dụng địa chỉ của ô đó

- Địa chỉ **tương đối**: **thay đổi** khi sao chép công thức từ ô này sang ô khác
  - được sử dụng nhiều
- Địa chỉ **tuyệt đối**: **không thay đổi** khi sao chép công thức từ ô này sang ô khác (ví dụ: B1)
  - tuyệt đối theo hàng: **không thay đổi chỉ số hàng** khi sao chép (ví dụ: B$1)
  - tuyệt đối theo cột: **không thay đổi chỉ số cột** khi sao chép (ví dụ: $C1)

Các chức năng cơ bản

- **Tự động lặp** các giá trị, tiêu đề và công thức
  - hiện các giá trị đã tồn tại trong bảng tính gần giống với dữ liệu đang được nhập
  - khái niệm tham chiếu tương đối: cho phép copy các công thức => tham chiếu tương đối thay đổi (demo)
- **Tự động tính lại**
- Có các **hàm thư viện**
- **Macro**
- **Bảng tính mẫu**
- **Liên kết**: cho phép tạo các liên kết động giữa các bảng tính (giữa các sheet trong *1 file* hoặc *nhiều file*)
- Chức năng **cơ sở dữ liệu**

Các chức năng cơ bản

- **Tự động lặp** các giá trị, tiêu đề và công thức
- **Tự động tính lại**
  - thay đổi giá trị ở 1 ô $\Rightarrow$ các tính toán liên quan cũng thay đổi
  - chức năng này có thể bật /tắt (Formulas/Calculation/Calculation Options)
- Có các **hàm thư viện**
- **Macro**
- **Bảng tính mẫu**
- **Liên kết**: cho phép tạo các liên kết động giữa các bảng tính (giữa các sheet trong *1 file* hoặc *nhiều file*)
- Chức năng **cơ sở dữ liệu**

Các chức năng cơ bản

- **Tự động lặp** các giá trị, tiêu đề và công thức
- **Tự động tính lại**
- Có các **hàm thư viện**
  - thực hiện một số các công việc định sẵn
  - tiết kiệm thời gian và nguy cơ gây lỗi
  - ví dụ: SUM, AVERAGE, IF, COUNT, SQRT, VLOOKUP, ...
- **Macro**
- **Bảng tính mẫu**
- **Liên kết**: cho phép tạo các liên kết động giữa các bảng tính (giữa các sheet trong *1 file* hoặc *nhiều file*)
- Chức năng **cơ sở dữ liệu**

Các chức năng cơ bản

- **Tự động lặp** các giá trị, tiêu đề và công thức
- **Tự động tính lại**
- Có các **hàm thư viện**
- **Macro**
  - 1 macro chứa các thao tác lặp đi lặp lại và được người dùng "thu" lại
  - định nghĩa bằng ngôn ngữ macro hoặc dùng bộ thu macro để thu lại các thao tác từ chuột và bàn phím (demo - hàm thư viện và macro)
- **Bảng tính mẫu**
- **Liên kết**: cho phép tạo các liên kết động giữa các bảng tính (giữa các sheet trong *1 file* hoặc *nhiều file*)
- Chức năng **cơ sở dữ liệu**

Các chức năng cơ bản

- **Tự động lặp** các giá trị, tiêu đề và công thức
- **Tự động tính lại**
- Có các **hàm thư viện**
- **Macro**
- **Bảng tính mẫu**
  - bảng tính chỉ chứa tiêu đề và công thức, chưa có dữ liệu
  - *tính năng tự động tính* lại sẽ giúp ta có kết quả khi đưa dữ liệu vào
  - tiết kiệm thời gian và công sức
- **Liên kết**: cho phép tạo các liên kết động giữa các bảng tính (giữa các sheet trong *1 file* hoặc *nhiều file*)
- Chức năng **cơ sở dữ liệu**

Các chức năng cơ bản

- **Tự động lặp** các giá trị, tiêu đề và công thức
- **Tự động tính lại**
- Có các **hàm thư viện**
- **Macro**
- **Bảng tính mẫu**
- **Liên kết**: cho phép tạo các liên kết động giữa các bảng tính (giữa các sheet trong *1 file* hoặc *nhiều file*)
- Chức năng **cơ sở dữ liệu**

Các chức năng cơ bản

- **Tự động lặp** các giá trị, tiêu đề và công thức
- **Tự động tính lại**
- Có các **hàm thư viện**
- **Macro**
- **Bảng tính mẫu**
- **Liên kết**: cho phép tạo các liên kết động giữa các bảng tính (giữa các sheet trong *1 file* hoặc *nhiều file*)
- Chức năng **cơ sở dữ liệu**
  - thực hiện được những chức năng CSDL cơ bản: lưu trữ và truy cập thông tin, tìm kiếm sắp xếp, phát sinh báo cáo, trộn thư(mail megre)

    $\Rightarrow$ đáp ứng những yêu cầu cơ sở dữ liệu phức tạp hơn, PMBT vẫn có thể hữu dụng
  - nhiều PMBT hỗ trợ kết nối 2 chiều với các phần mềm CSDL

Những đặc điểm nổi bật khác

- Vẽ đồ thị: từ các con số chuyển thành đồ thị để biểu đạt thông tin: đồ thị tròn, đồ thị đường, đồ thị cột, ...
- Công cụ giải phương trình, những bài toán tối ưu
- Lotus hỗ trợ Multimedia, Excel sử dụng trí tuệ nhân tạo, ... => tương lai của PMBT ?

Kinh nghiệm

- Hãy **hình dung bảng tính** trước khi bạn đưa ra các giá trị và công thức vào: mục tiêu ?
- Kiểm tra nhiều lần mỗi công thức và giá trị: **GIGO** (rác vào thì rác ra – *Garbage In Garbage Out*)
- Làm bảng tính trở nên **dễ đọc**
- **Kiểm tra kết quả** bằng những cách khác
- Xây dựng các hàm **kiểm tra chéo**
- **Đổi giá trị đầu vào** và **quan sát kết quả**
- Hãy tận dụng những **hàm có sẵn**
- PMBT hỗ trợ quyết định chứ **không thay quyết định**

Khái niệm

- Cơ sở dữ liệu (CSDL):
  - tập hợp thông tin có liên quan mà chúng ta quan tâm,
  - tập thông tin có cấu trúc
  - được lưu trên các thiết bị lưu trữ thông tin
  - ví dụ:
    - trang niên giám điện thoại
    - danh sách sinh viên
    - hệ thống tài khoản ngân hàng
- Các chương trình CSDL được thiết kế để **quản lý** CSDL

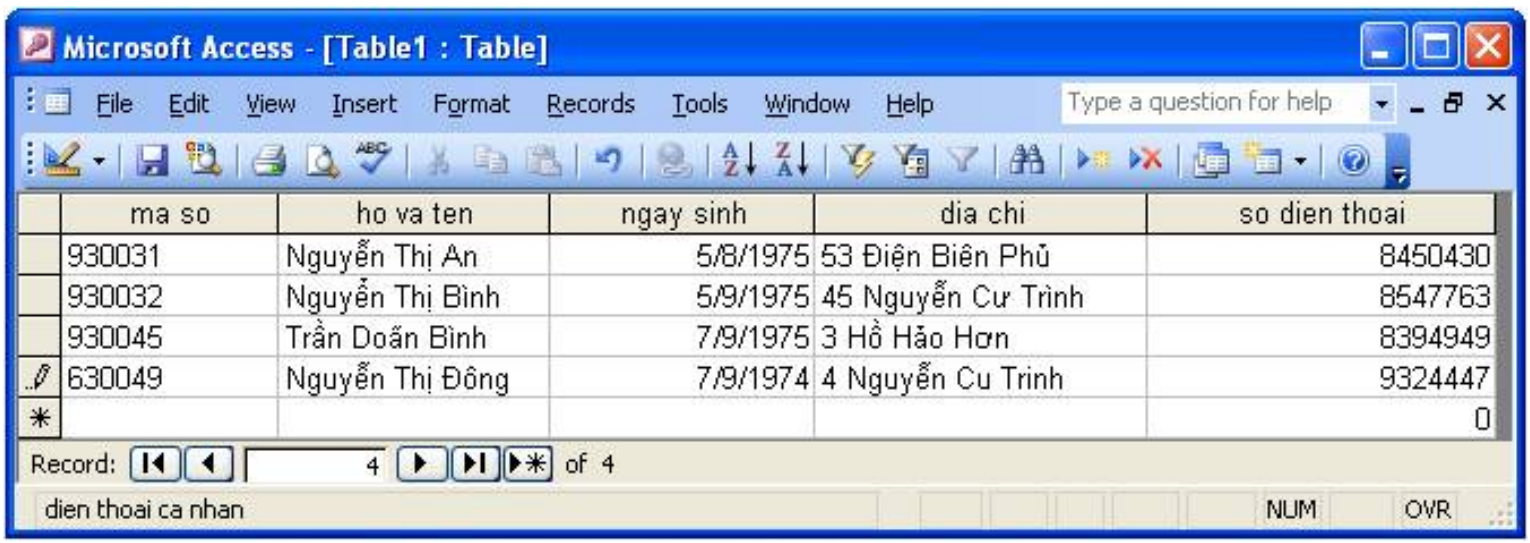
Ưu điểm khi sử dụng CSDL

- Việc lưu trữ một **lượng thông tin khổng lồ** trở nên **dễ dàng**
- Giúp *nhanh chóng* và *mềm dẻo* trong việc **tra cứu thông tin**
- Giúp dễ dàng **sắp xếp và tổ chức thông tin**
- Giúp **in** và **phân phối thông tin** theo nhiều cách

Bên trong cơ sở dữ liệu

- CSDL được hình thành từ **các file chứa một tập thông tin có liên quan**
- 1 file (tệp tin) trong CSDL gồm:
  - Nhiều **bản ghi** (*record*): 1 bản ghi là thông tin liên quan đến 1 người, 1 sản phẩm hoặc 1 sự kiện nào đó
  - Nhiều **trường** (*field*): Mỗi 1 đoạn thông tin riêng rẽ trong 1 record là 1 trường
  - Mỗi trường được xác định bằng **kiểu cụ thể**: ngày, chữ, số, ...
  - Ví dụ: 1 file về sách trong CDSL thư viện
    - **1 record** có các field cho **tác giả, tựa đề sách, nhà XB, địa chỉ, năm xuất bản, ...**
    - *tác giả*: kiểu chữ; *năm xuất bản*: kiểu số, ...

Bên trong cơ sở dữ liệu



Chức năng của chương trình CSDL

- Truy vấn cơ sở dữ liệu (*query*): một yêu cầu về tìm kiếm thông tin
  - truy vấn có thể **đơn giản** (*tìm sinh viên có mã số ABC*) hoặc **phức tạp** thỏa mãn nhiều điều kiện (*tìm sinh viên có điểm TB $<$ 5 và còn nợ trên 2 môn*)
  - **ngôn ngữ** thường được sử dụng: SQL (Structured Query Language)
  - ví dụ: xác định tất cả record của những người nam giới trong độ tuổi từ 18 đến 35

    *Select * from Population Where Sex = M and Age $>$ 18 and Age $<$ 35*
- Sắp xếp dữ liệu
- Báo cáo

Các chương trình CSDL chuyện biệt

- lập trình sẵn cho một **kiểu dữ liệu** xác định và **một cách truy cập xác định**
- người dùng: **KHÔNG** định nghĩa các cấu trúc file hay giao diện hiển thị
- **tên** được đặt để phản ánh mục đích sử dụng: *danh bạ điện thoại điện tử, hệ thông tin địa lý (Geographical Information System - GIS), ...*
- nhiều dạng chương trình CSDL được bán dưới dạng **hệ quản trị thông tin cá nhân** (*Personal Information Manager - PIM*) thực hiện các chức năng:
  - sổ địa chỉ, danh bạ điện thoại
  - lịch hẹn
  - danh sách công việc phải làm
  - ghi chú cá nhân, nhật ký, ...

Khái niệm

- **Hệ quản trị cơ sở dữ liệu** là phần mềm cung cấp một môi trường thuận lợi và hiệu quả để **tạo lập, lưu trữ và tìm kiếm thông tin** của cơ sở dữ liệu
- Hệ quản trị CSDL + CSDL => được gọi chung là **HỆ CSDL**

Các chức năng cơ bản

- Tạo lập CSDL

  ⇒ người dùng sử dung **ngôn ngữ định nghĩa dữ liệu**: khai báo kiểu và cấu trúc của DL, khai báo các ràng buộc trên DL

- Cập nhật dữ liệu, tìm kiếm và kết xuất thông tin

  ⇒ người dùng sử dung **ngôn ngữ thao tác dữ liệu**: cập nhật (nhập, xóa, sửa) DL, tìm kiếm thông tin

- Cung cấp công cụ kiểm soát, điều khiển việc truy cập vào CSDL để đảm bảo:
  - phát hiện và ngăn chặn truy cập trái phép
  - duy trì tính nhất quán của dữ liệu
  - tổ chức, điều khiển các truy cập cùng lúc
  - khôi phục CDSL khi gặp sự cố
  - quản lý các mô tả dữ liệu

Quản trị file vs. quản trị CSDL

Bài toán quản lý sinh viên. Các file dùng để chứa thông tin liên quan đến sinh viên:

- Thông tin sinh viên
- Thông tin tài chính
- Danh sách lớp

Quản trị file vs. quản trị CSDL

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

...

...

- **Tính quan hệ** trong CSDL: sự ràng buộc giữa các file dữ liệu
- Tính **nhiều mặt**: Với cùng 1 CSDL, mỗi người dùng (*nhân viên bán hàng, nhà quản lý kế toán, ...*) có:
  - nhu cầu cần truy cập thông tin khác nhau
  - cách thức làm việc khác nhau
    (*ví dụ về con voi*)
- Xu thế **Client/Server**
  tận dụng được lợi điểm **giao diện đơn giản** của các máy PC + **sức mạnh tính toán** của các máy tính cỡ lớn với các CSDL khổng lồ
- CSDL của ngày mai: CSDL **hướng đối tượng**

Questions & Answers

Tin học đại cương

Bài 4: Giải quyết bài toán - Phần 1: Thuật toán

NGUYỄN Thị Oanh
oanhnt@soict.hut.edu.vn

Bộ môn Hệ thống thông tin - Viện CNTT và Truyền Thông
Đại học Bách Khoa Hà nội

2010 - 2011

Nội dung

Định nghĩa thuật toán

- là khái niệm cơ sở trong tin học và toán học
- "Algorithm" ra đời dựa theo cách phiên âm tên của nhà toán học Trung á (Abu Abd - Allah ibn Musa al'Khwarizmi)
- ĐN chung: thuật toán bao gồm một **dãy hữu hạn** các chỉ thị rõ ràng, **có trình tự** và có thể **thi hành được** để hướng dẫn thực hiện hành động nhằm đạt được mục tiêu đề ra
- Thuật toán được xem như là **sự thể hiện của 1 phương án** giải quyết 1 vấn đề nào đó

Định nghĩa thuật toán

- ĐN trong khoa học máy tính:
  - gồm các dãy các chỉ thị không mập mờ và có thể thực thi được (tính **xác định**)
    không mập mờ: tại mỗi bước, hành động kế tiếp phải được xác định một cách duy nhất theo chỉ thị hành động và dữ liệu tại thời điểm đó
  - quá trình hoạt động theo thuật toán phải dừng (tính **hữu hạn**) và cho kết quả như mong muốn (tính **đúng**)

Thuật toán ?

Chỉ thị **không mập mờ** ?

- thuật toán chọn lớp trưởng

- VD1:
  1. lập danh sách tất các SV trong lớp
  2. sắp thứ tự danh sách SV
  3. chọn học sinh đứng đầu danh sách để làm lớp trưởng
- VD2:
  1. lập danh sách tất các SV trong lớp theo hai thông tin: Họ và Tên, Tổng điểm thi đầu vào
  2. sắp thứ tự danh sách SV theo thứ tự tổng điểm giảm dần, 2 học sinh có cùng điểm TB có cùng hạng
  3. nếu có 1 HS đứng hạng nhất chọn HS đó làm lớp trường, nếu không thì tiến hành bốc thăm cho các SV được xếp hạng nhất

Thuật toán ?

- Chỉ thị **thực thi** được: xét trong điều kiện hiện tại của bài toán
  - VD1: $sqrt(-1)$
  - VD2: bài toán chỉ đường:

    ...

    bước n: rẽ vào đường X (X đường **ngược chiều**)

    ...
- Tính **hữu hạn** (dừng): dễ bị vi phạm nhất, thường do sai sót khi trình bày thuật toán

  B1 Nhập **n**

  B2 **i = n**

  B3 Gán **i = i-2**

  B4 Nếu **(i < 0)** thì sang **B6**

  B5 Quay lại **B3**

  B6 Hiển thị giá trị của **i**

Thuật toán - Ví dụ

Tìm giá trị lớn nhất trong một dãy hữu hạn số nguyên

1. Đặt **giá trị lớn nhất tạm thời** = **số nguyên đầu tiên**
2. **So sánh** giá trị số nguyên kế tiếp với giá trị lớn nhất tạm thời, *nếu* nó **lớn hơn** *thì* **gán lại giá trị lớn nhất tạm thời** bằng giá trị số nguyên này
3. **Lặp 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, **giá trị lớn nhất** trong dãy chính là **giá trị lớn nhất tạm thời** lúc này

Các đặc trưng của thuật toán

- Tính xác định: các bước trong thuật toán phải **chính xác, rõ ràng**
- Tính hữu hạn: thuật toán phải cho kết quả (lời giải) sau một **số hữu hạn** các bước
- Tính đúng: thuật toán phải cho **kết quả đúng**

- Nhập: các giá trị nhập vào (input values) từ một tập nào đó
- Xuất: giá trị vào + thuật toán => giá trị ra (output values) : thể hiện lời giải cho bài toán
- Tính hiệu quả: dựa trên **khối lượng** tính toán, **không gian và thời gian** sử dụng,...
- Tính tổng quát: phải áp dụng cho **tất cả các bài toán có dạng như mong muốn** chứ không phải chỉ cho 1 trường hợp riêng lẻ nào

Cách biểu diễn thuật toán

- Ngôn ngữ tự nhiên
- Ngôn ngữ lưu đồ (lưu đồ/sơ đồ khối)
- Ngôn ngữ tựa lập trình (mã giả -*pseudocode*) gọi là ngôn ngữ mô phỏng chương trình PDL (*Programming Description Language*)
- Ngôn ngữ lập trình: Pascal, C/C++ hay Java, ...

Biểu diễn bằng ngôn ngữ tự nhiên

- Sử dụng một loại ngôn ngữ tự nhiên (Anh, Pháp, Việt, ...) để liệt kê các bước của thuật toán
- Ví dụ: đưa ra kết luận về tương quan giữa hai số a và b (>, < hay =)
  - Đầu **vào**: Hai số a và b
  - Đầu **ra**: Kết luận a>b hay a<b hay a=b
  - **Ý tưởng**: So sánh a và b rồi đưa ra kết luận
  - **Thuật toán**:

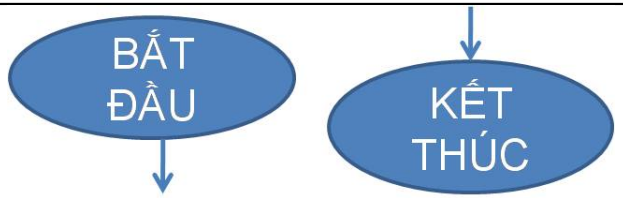
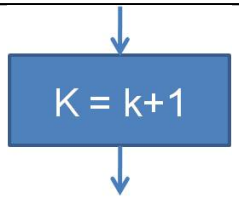
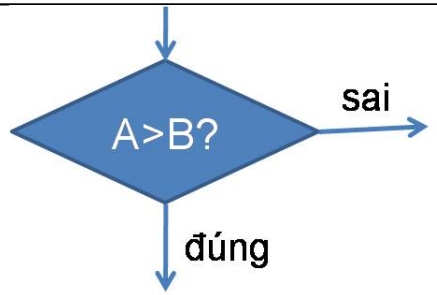
1. Nhập số a và số b
2. Nếu a> b, hiển thị "a>b" và kết thúc
3. Nếu a=b, hiển thị "a=b" và kết thúc
4. Nếu (a<b) hiển thị "a<b" và kết thúc

Biểu diễn bằng ngôn ngữ tự nhiên

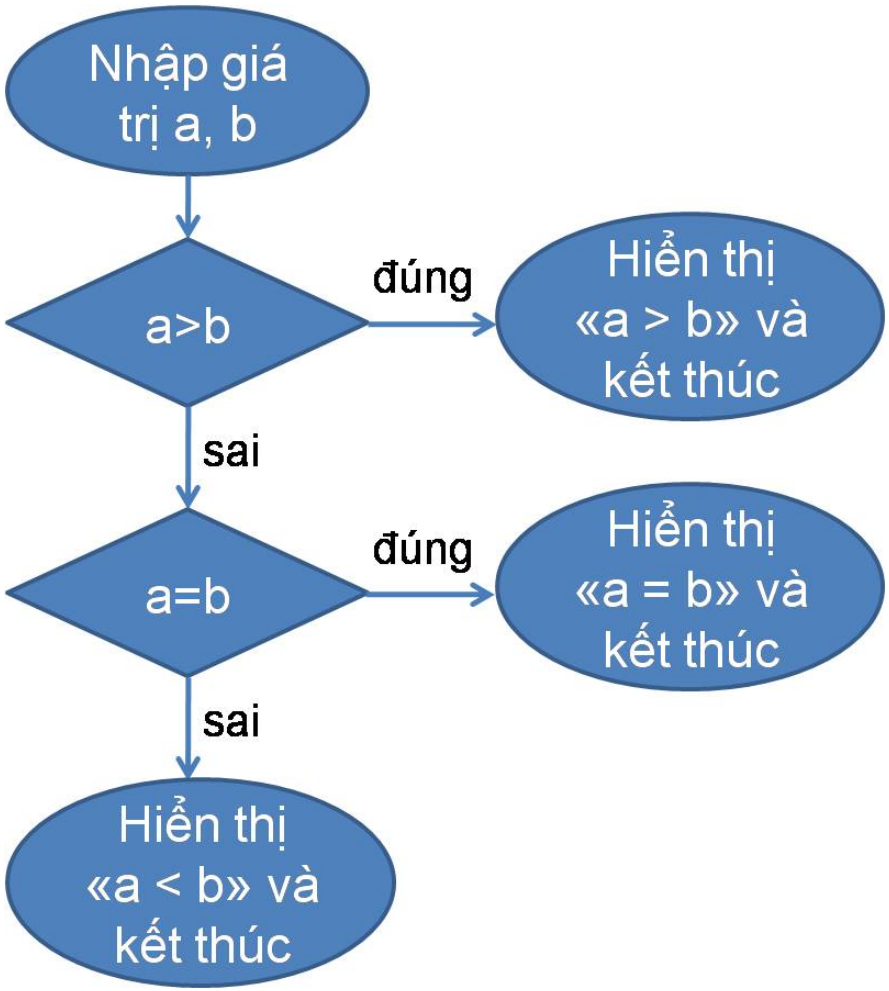
- Ưu điểm:
  - Đơn giản
  - Không yêu cầu người viết và người đọc phải có kiến thức nền tảng
- Nhược điểm:
  - Dài dòng
  - Không làm nổi bật cấu trúc của thuật toán
  - Khó biểu diễn với những bài toán phức tạp

Biểu diễn bằng lưu đồ (sơ đồ khối)

Lưu đồ: 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**

nút giới hạn: hình oval và có chữ ghi bên trong	
nút thao tác: hình chữ nhật, bên trong ghi các lệnh	
nút điều kiện: hình thoi, bên trong ghi điều kiện, có 2 cung ra chỉ điều kiện đúng/sai	
cung: đường nối từ nút này đến nút khác của lưu đồ	

Biểu diễn bằng lưu đồ (sơ đồ khối)



Biểu diễn bằng lưu đồ (sơ đồ khối)

- Ưu điểm:
  - trực quan, dễ diễn đạt, dễ hiểu
  - cung cấp toàn cảnh, tổng quan về thuật toán
- Nhược điểm:
  - cồng kềnh nhất là với bài toán phức tạp

Biểu diễn bằng mã giả (pseudo code)

- Ngôn ngữ **tựa** (gần giống) **với ngôn ngữ lập trình** được gọi là mã giả
  - sử dụng mệnh đề có cấu trúc
  - sử dụng ngôn ngữ tự nhiên
  - sử dụng các biến, các kí hiệu toán học, các cấu trúc kiểu thủ tục, ...
- Ưu điểm: tiện lợi, đơn giản, dễ hiểu, dễ diễn đạt
- Nhược điểm: người đọc/viết phải hiểu được cấu trúc trong lập trình và cách biểu diễn

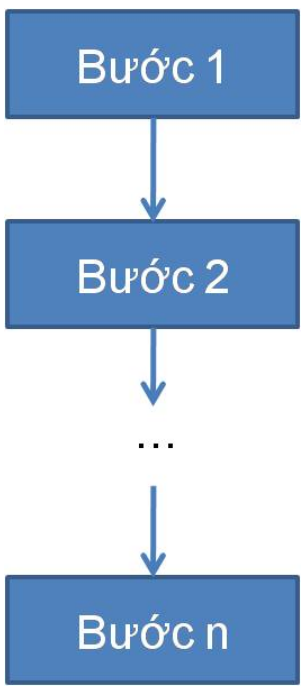
Cấu trúc cơ bản trong lập trình

3 cấu trúc cơ bản:

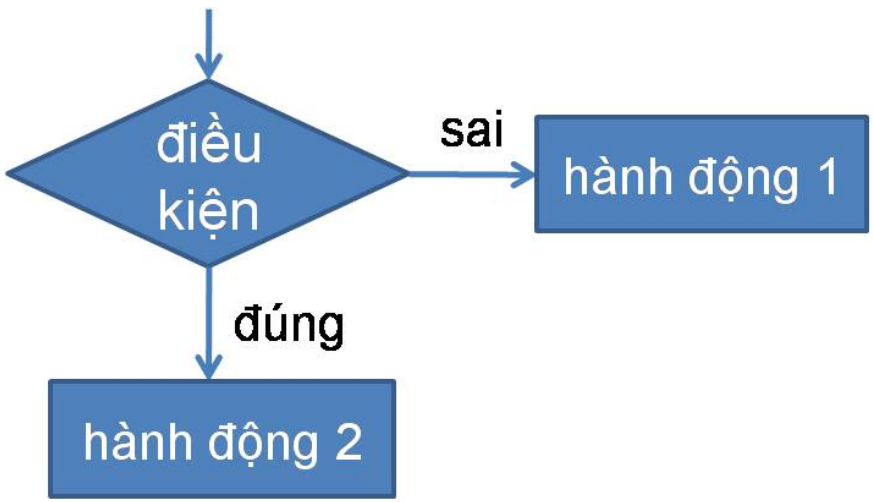
- cấu trúc **tuần tự**
- cấu trúc **rẽ nhánh**
- cấu trúc **lặp**

Cấu trúc cơ bản trong lập trình

- Cấu trúc tuần tự: các bước thực hiện theo tuần tự tuyến tính, hết bước này đến bước khác



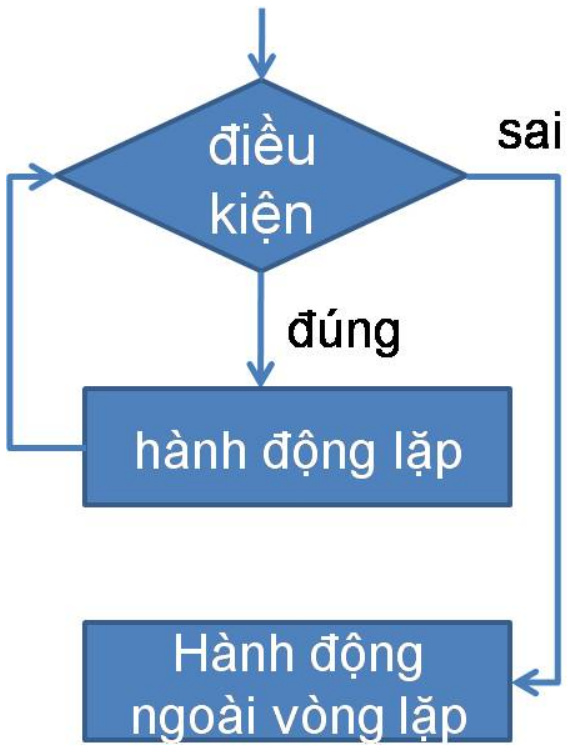
- Cấu trúc rẽ nhánh: việc thực hiện bước nào phục thuộc vào điều kiện xác định



Cấu trúc cơ bản trong lập trình

Cấu trúc lặp:

- Một hoạt động/ nhiệm vụ có thể được thực hiện lặp nhiều lần
- Số lần lặp có thể biết trước hoặc không biết trước, tuy nhiên số lần lặp phải hữu hạn



Biểu diễn bằng mã giả (pseudo code)

- Phép gán: $:=, \leftarrow$

  $i := i + 1$

  $i \leftarrow 0$

- Cấu trúc chọn (rẽ nhánh):
  If (điều kiện) **then** (hành động) **[endif]**
  hoặc
  If (điều kiện) **then** (hành động 1) **else** (hành động 2) **[endif]**
- Cấu trúc nhảy:
  goto nhãn x;

Biểu diễn bằng mã giả (pseudo code)

- Cấu trúc lặp:

  hoặc **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 cuối) **downto** (giá trị đầu) **do**
  (hành động)

- Để rõ ràng dãy các hành động có thể **nhóm** lại bằng:

  begin *(dãy hành động)* **end** *hoặc*
  { *dãy hành động* **}**

Biểu diễn bằng mã giả (pseudo code)

Nhập a, b;
If (*a>b*) **then**
    in("a > b");
else
    If *(a=b)* **then**
        in("a = b");
    else
        in("a < b");
    endif;
endif;

Biểu diễn bằng ngôn ngữ lập trình

- Định nghĩa: *Ngôn ngữ lập trình* là một hệ thống được ký hiệu hóa để miêu tả những tính toán (qua máy tính) trong một dạng mà cả **con người và máy** đều có thể **đọc và hiểu được**
- C, java, Pascal, C#, ...

  (chi tiết trong phần sau (phần lập trình) của môn học)

Ví dụ 2: Tính tổng, tích, hiệu, thương

Bài toán: Tính tổng, tích, hiệu, thương của 2 số a,b

- Đầu vào (*Input*): hai số a, b
- Đầu ra (*Output*): tổng, hiệu, tích, thương của a và b
- Ý tưởng:
  - tính tổng, hiệu, tích của a, b
  - nếu b khác 0 thì tính thương
  - nếu b bằng 0 thì đưa thông báo không thực hiện được phép chia

Ví dụ 2 (...) - ngôn ngữ tự nhiên

B1: Nhập số a và số b

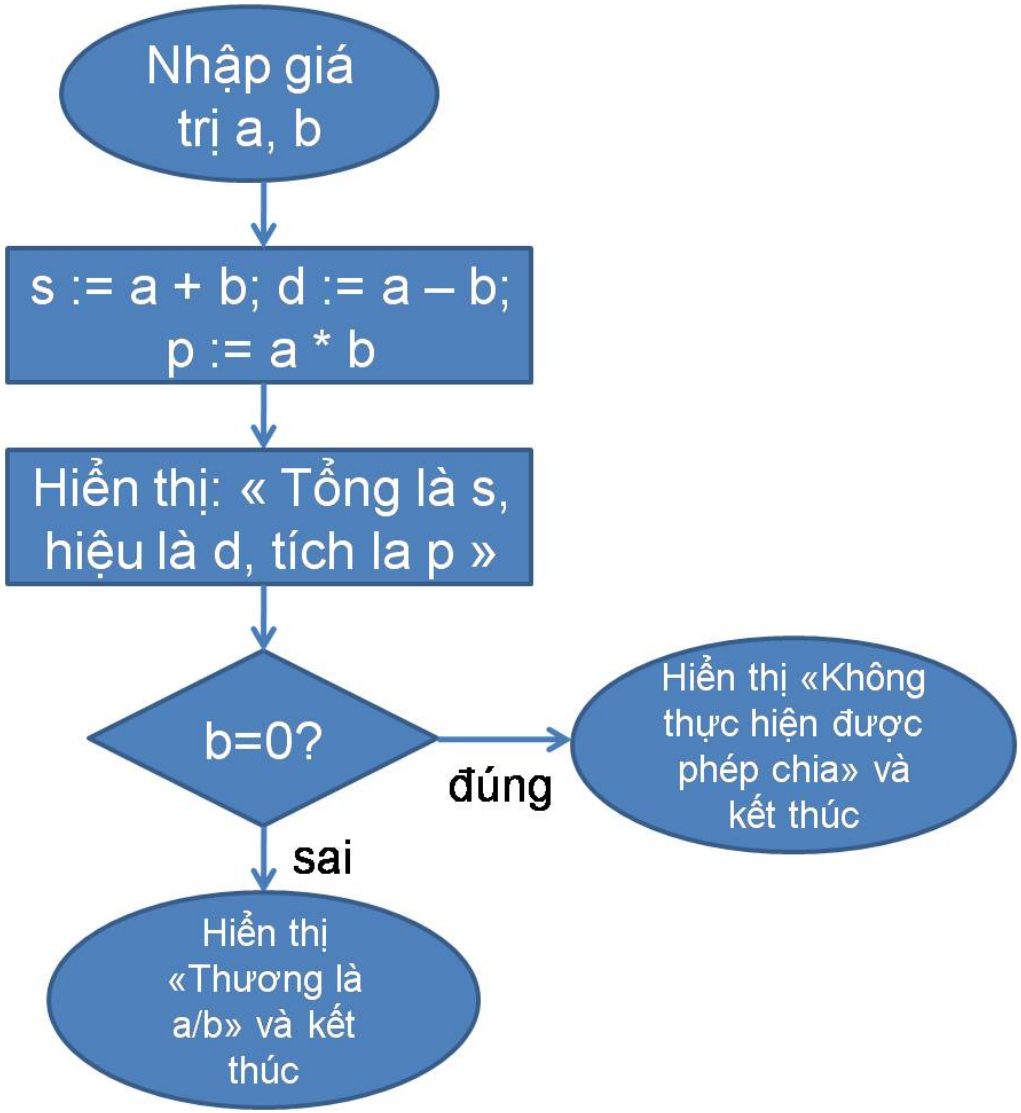
B2: Tính $s \leftarrow a + b; d \leftarrow a - b; p \leftarrow a * b$

B3: Hiển thị: Tổng là s, Hiệu là d, Tích là p

B4: Nếu b $=$ 0, hiển thị "Không thực hiện được phép chia" và kết thức

B5: Nếu b<> 0, hiển thị "Thương là a/b" và kết thúc

Ví dụ 2 (...) - Lưu đồ


Nhập giá trị a, b
s := a + b; d := a – b; p := a * b
Hiển thị: « Tổng là s, hiệu là d, tích la p »
b=0?
đúng
Hiển thị «Không thực hiện được phép chia» và kết thúc
sai
Hiển thị «Thương là a/b» và kết thúc

Ví dụ 3: Giải phương trình bậc 1

Bài toán: Giải phương trình bậc 1 : $ax + b = 0$

- Đầu vào: 2 số a, b
- Đầu ra: Nghiệm của phương trình $ax + b = 0$
- Ý tưởng:
  - lần lượt xét $a = 0$ rồi xét $b = 0$ để xét các trường hợp của phương trình

Ví dụ 3 (...) - Ngôn ngữ tự nhiên

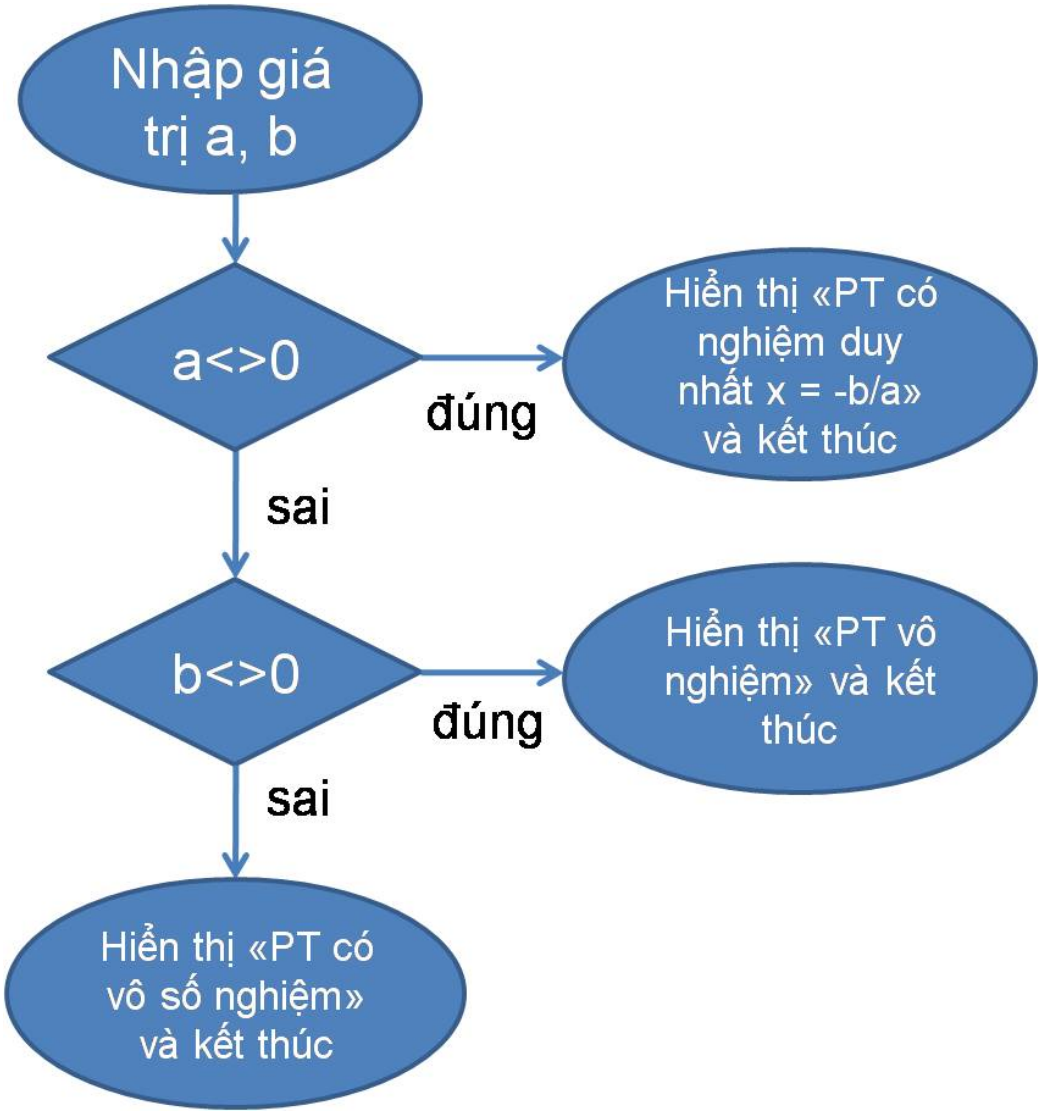
B1 Nhập a và b

B2 Nếu **a <> 0** thì hiển thị "Phương trình có nghiệm duy nhất x = -b/a" và kết thúc

B3 Nếu **a = 0 và b <> 0** thì hiểu thị "Phương trình vô nghiệm" và kết thúc

B4 Nếu **a = 0 và b = 0** thì hiểu thị "Phương trình có vô số nghiệm" và kết thúc

Ví dụ 3 (...) - Lưu đồ


Nhập giá trị a, b
a<>0
đúng
Hiển thị «PT có nghiệm duy nhất x = -b/a» và kết thúc
sai
b<>0
đúng
Hiển thị «PT vô nghiệm» và kết thúc
sai
Hiển thị «PT có vô số nghiệm» và kết thúc

Ví dụ 4: Tìm giá trị lớn nhất

Bài toán: Tìm giá trị lớn nhất của một dãy số nguyên có N số

- Vào: Số nguyên dương N và dãy N số nguyên $a_1, a_2, \ldots, a_N$
- Ra: giá trị lớn nhất của dãy
- Ý tưởng:
  - khởi tạo giá trị $Max = a_1$
  - lần lượt so sánh Max với a_i với i=2,3,. . . , N; nếu $a_i > Max$ ta gán giá trị mới cho Max

Ví dụ 4 (...) - Ngôn ngữ tự nhiên

B1 Nhập số nguyên dương N và dãy số $a_1, a_2, \ldots, a_N$

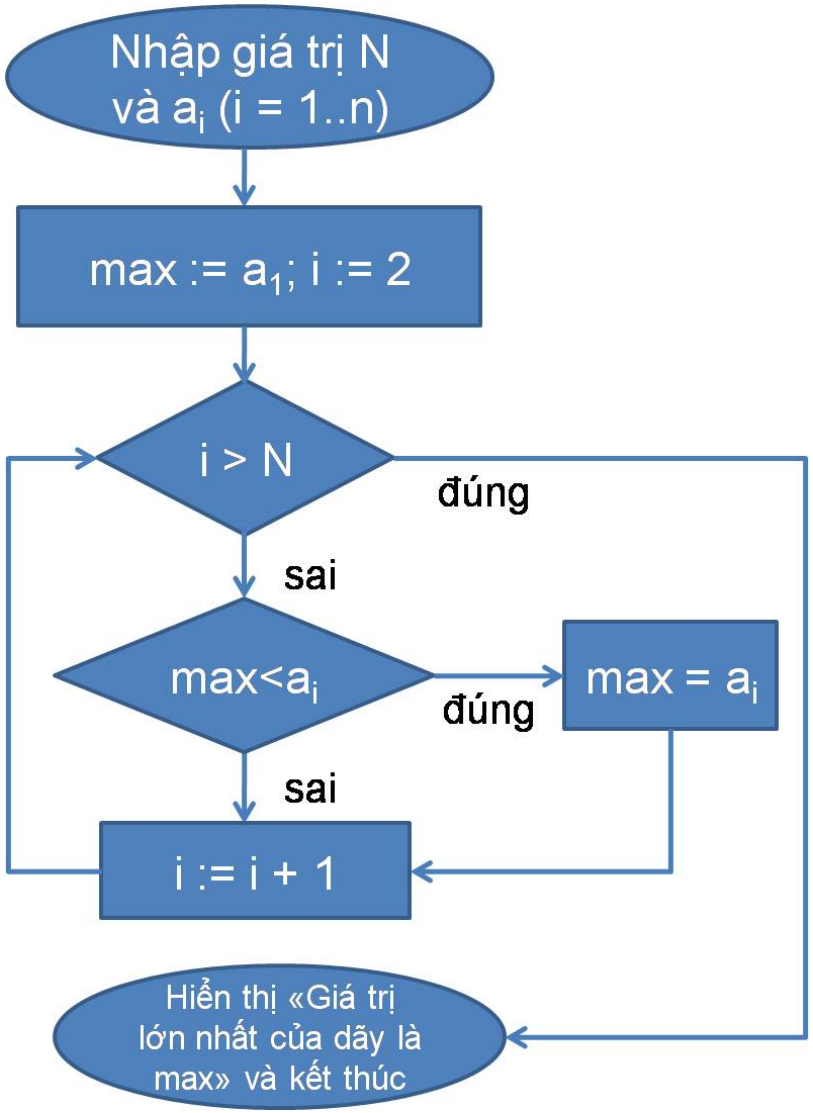
B2 gán max bằng a_1 và i bằng 2

B3 Nếu $i > N$, Hiển thị Max là giá trị lớn nhất của dãy và kết thúc

B4 Nếu $a_i > Max$, $Max \leftarrow a_i$

B5 Tăng i lên 1 đơn vị và quay lên **B3**

Ví dụ 4 (...) - Lưu đồ


Nhập giá trị N và a_i (i = 1..n)
max := a_1; i := 2
i > N
đúng
sai
max<a_i
đúng
max = a_i
sai
i := i + 1
Hiển thị «Giá trị lớn nhất của dãy là max» và kết thúc

Ví dụ 5: Sắp xếp

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

- Đầu vào: Dãy A gồm N số nguyên $a_1, a_2, \ldots, a_N$
- Đầu ra: Dãy A dược sắp lại theo **thứ tự không giảm**
- Ý tưởng:
  - Với mỗi **cặp số liên tiếp** trong dãy, nếu *số trước lớn hơn số sau* ta **đổi chỗ** chúng cho nhau
  - **Lặp** cho đến khi không có sự đổi chỗ nào cho nhau

Ví dụ 5 (...) - Ngôn ngữ tự nhiên

B1 Nhập số N và dãy số $a_1, a_2, \ldots, a_N$

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

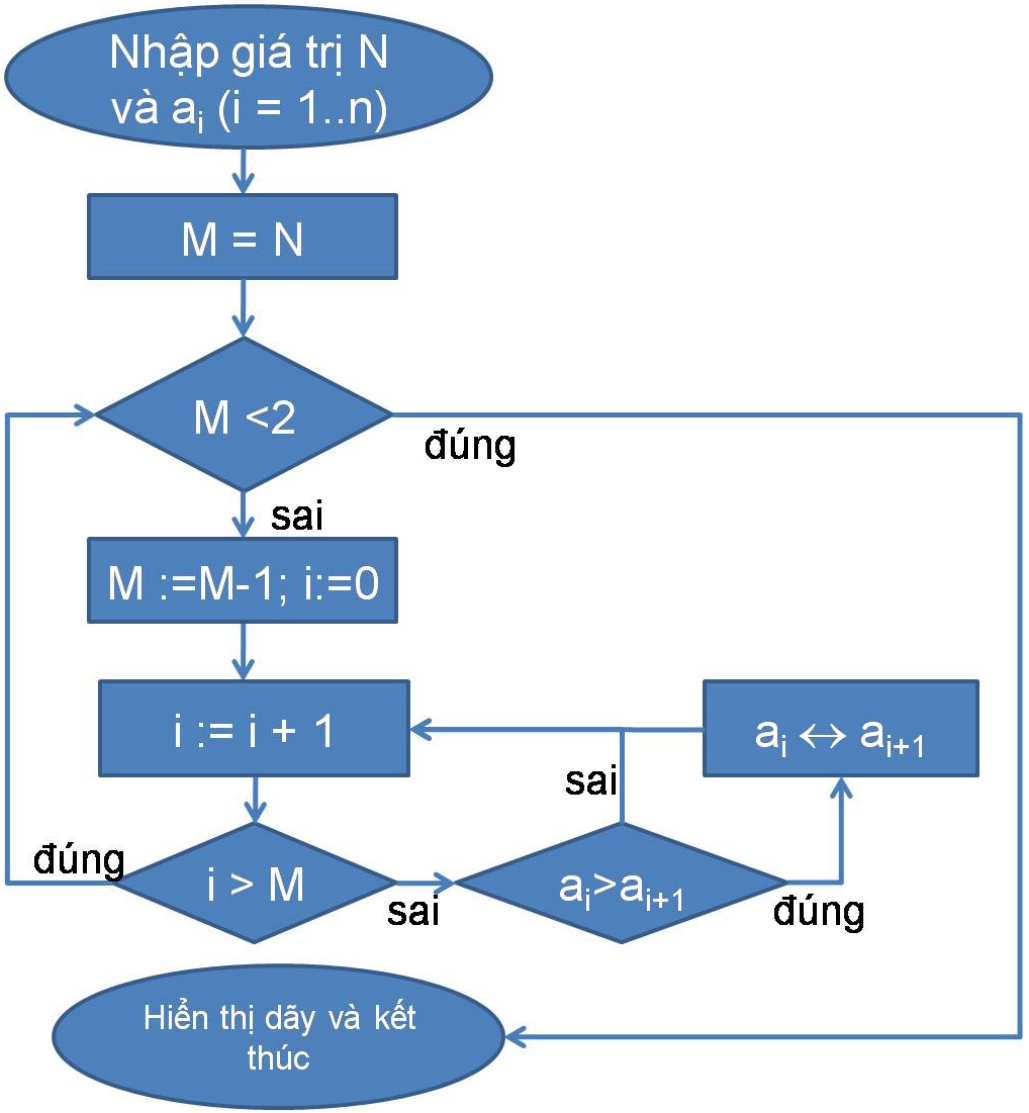
B6 Nếu $i > M$ thì **quay lại B3**

B7 Nếu $a_i > a_{i+1}$ thì tráo đổi hai số đó cho nhau

B8 **Quay lên B5**

$\Rightarrow$ sắp xếp "nổi bọt" (Bubble Sort)

Ví dụ 5 (...) - Lưu đồ


Nhập giá trị N và a_i (i = 1..n)
M = N
M <2
đúng
sai
M :=M-1; i:=0
i := i + 1
$a_i \leftrightarrow a_{i+1}$
sai
đúng
i > M
$a_i > a_{i+1}$
sai
đúng
Hiển thị dãy và kết thúc

Bài tập

Giải phương trình bậc 2 : $ax^2 + bx + c = 0$

Bài toán về số học

Bài toán về số học:

- Xác định một số nguyên có phải là **số nguyên tố**/hợp số hay không
- Tìm **USCLN**, **BSCNN** của 2 số nguyên
- ...

Bài toán kiểm tra số nguyên tố

Bài toán: Cho một số nguyên dương p => p có phải là số nguyên tố hay không ?

- **Input:** p nguyên dương
- **Output:** kết luận về tính nguyên tố của p
- **Ý tưởng?**
  - p = 1? ⇒ Không phải số nguyên tố
  - p > 1?
  * Kiểm tra từ 2 đến p-1 có phải là ước số của p không
  * **Nếu có** thì kết luận p **không là số nguyên tố**, ngược lại **không có** số nào thì kết luận **p là số nguyên tố**

Thuật toán kiểm tra số nguyên tố

```
Nhập p;
if p=1 then begin
                Xuất: p không nguyên tố;
                Dừng thuật toán;
                end
flag := TRUE;
for k:=2 to p-1 do
    if (k là ước số của p) then begin
                                flag:=FALSE;
                                break; { ngắt vòng lặp FOR }
                                end
if flag=TRUE then Xuất: p là số nguyên tố;
             else Xuất: p không là số nguyên tố;
```

Thuật toán về dãy

- Làm việc với một dãy số
- Các bài toán điển hình:
  - Tìm số **lớn nhất**, **nhỏ nhất** trong dãy
  - **Kiểm tra** dãy có phải là dãy tăng hoặc dãy giảm
  - **Sắp xếp** dãy tăng dần hoặc giảm dần
  - **Tìm** trong dãy có phần tử nào bằng một giá trị cho trước
  - **Tính** trung bình cộng của dãy
  - ...

Ví dụ tìm số lớn nhất trong dãy

- Input: dãy số $a_1, a_2, a_3, \ldots a_n$
- Output: max là giá trị lớn nhất trong dãy số đã cho
- **Thuật toán**:

  Nhập số nguyên dương n và dãy $a_1, a_2, \ldots, a_n$;
  $max := a_1$;
  for i:=2 **to** n **do**
  if $max < a_i$ **then** $max := a_i$;
  Xuất: max là giá trị lớn nhất trong dãy số;

Bài tập

Bài 1. Xây dựng thuật toán tìm phần tử có giá trị truyệt đối lớn nhất trong dãy gồm n phần tử

Bài 2. Xây dựng thuật toán tìm tổng của các số chẵn và tổng của các số lẻ trong dãy gồm n phần tử được nhập vào từ bàn phím

Bài 3. Xây dựng thuật toán kiểm tra xem một dãy số gồm n phần tử được nhập vào từ bàn phím có phải là dãy số tăng (hoặc giảm) không

Bài 4. Xây dựng thuật toán tính trung bình cộng của các số dương trong dãy gồm n số được nhập vào từ bàn phím

Thuật toán đệ quy

- Nhắc lại: tính chất của thuật toán
  - tính xác định: các bước rõ ràng, chính xác
  - tính dừng: thuật toán phải dừng sau 1 số bước hữu hạn
  - tính đúng: kết quả đúng
- Một số trường hợp tìm ra thuật toán có tính chất như trên **khó khăn phức tạp** nhưng có thể có các giải khác **vi pham t/c thuật toán** nhưng **đơn giản** và **được chấp nhận**

$\Longrightarrow$ **Mở rộng** khái niệm thuật toán : thuật toán đệ quy / thuật giải đệ quy **là 1 sự mở rộng**

Thuật toán đệ quy

- Một số bài toán có thể được **phân tích** thành các **bài toán cùng loại** nhưng ở *mức độ thấp hơn* (độ lớn DL nhập ít hơn, giá trị tính toán nhỏ hơn, ...)
  Ví dụ:

  Giai thừa: $n! = (n-1)! * n$

  Dãy số Fibonacci: 0, 1, 1, 2, 3, 5, 8...
  $F(n) = F(n-1) + F(n-2)$
  $\Rightarrow$ thuật toán để giải: dựa trên **chính nó** $\Rightarrow$ gọi là thuật toán đệ quy
- Thuật toán đệ quy: **trong** các bước của **thuật toán** có thể có trường hợp **thực hiện lại thuật toán** đó

Thuật toán đệ quy tính giai thừa của 1 số tự nhiên

Input: số tự nhiên n
Output: GT(n)=n!
Thuật giải:

Nhập số tự nhiên n;
GT:=1;
if n>0 **then**
GT := GT(n-1)*n;
Xuất GT;

2 thành phần của thuật toán đệ quy

Để xây dựng thuật toán đệ quy, phải xác định:

- Phần cơ bản: các trường hợp **không cần thực hiện lại thuật toán** (không có yêu cầu gọi đệ quy)
- Phần quy nạp (phần tổng quát): có yêu cầu **gọi đệ quy**
  - cần xác định nguyên lý đưa trường hợp **tổng quát về trường hợp cơ bản**
  - đảm bảo **tính dừng** của giải thuật đệ quy: chắc chắn từ trường hợp tổng quát sẽ đến được trường hợp cơ bản

Tràn "STACK"

Khi cài đặt :

- Các biến cục bộ trong hàm/thủ tục đệ quy : lưu trong vùng nhớ "STACK"
  => dễ gây tràn "STACK"
- Nhiều trường hợp có thể viết lại thuật toán đệ quy dưới dạng lặp

Thuật giải - Khái niệm

- **Tình huống** có thể xảy ra:
  - 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
  - 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

Thuật giải - Khái niệm

⇒ **Mở rộng** tiêu chuẩn cho thuật toán:

- tính xác định => thuật toán đệ quy
- tính đúng: chấp nhận cách giải cho kết quả gần đúng => cách giải heuristic

⇒ **Thuật giải:** cách giải chấp nhận được nhưng không hoàn toàn đáp ứng đầy đủ các tiêu chuẩn của thuật toán

Thuật giải heuristic

- Thuật giải heuristic là sự mở rộng của khái niệm thuật toán: dùng **"mẹo"** để giải
- Đặc tính:
  - thường tìm được lời giải tốt (nhưng **không chắc là tốt nhất**)
  - thực hiện thường dễ dàng và nhanh chóng hơn so với **giải thuật tối ưu**
  - thường thể hiện một cách hành động khá tự nhiên, gần gũi với cách suy nghĩ và hành động của con người

Nguyên lý cơ sở để thiết kế thuật giải heuristic

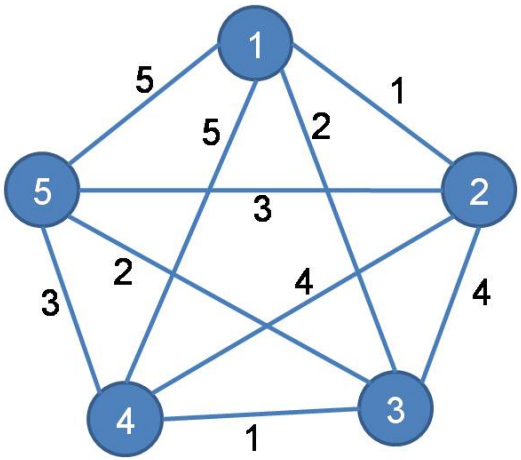
- Nguyên lý vét cạn thông minh: bài toán có không gian tìm kiếm lớn, dựa vào đặc thù của btoán
  - giới hạn không gian tìm kiếm
  - **hoặc** thực hiện kiểu dò tìm đặc biệt
- Nguyên lý tham lam: lấy **tiêu chuẩn tối ưu** (trên phạm vi toàn cục) của bài toán để làm **tiêu chuẩn chọn lựa hành động cho phạm vi cục bộ** của từng bước (hay từng giai đoạn) trong quá trình tìm kiếm lời giải
- Nguyên lý thứ tự: thực hiện hành động dựa trên một **cấu trúc thứ tự hợp lý** của không gian khảo sát nhằm nhanh chóng đạt được một lời giải tốt
- thường dùng các hàm heuristic: **f(trạng thái/tình huống ở từng bước thực hiện)**, hàm đánh giá thô => lựa chọn các bước tiếp hợp lý

Ví dụ: bài toán người bán hàng

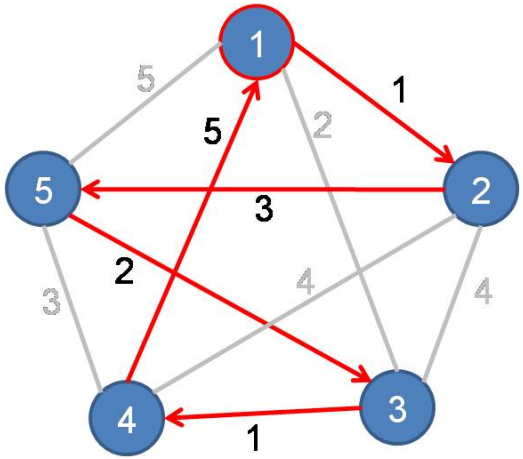
Bài toán người bán hàng - sử dụng nguyên lý tham lam:

- Bài toán: tìm hành trình ngắn nhất cho người bán hành đi qua n đại lý cho ở các thành phố khác nhau
- Thuật toán tìm **lời giải tối ưu**:
  - **duyệt tất cả các khả năng có thể** (n! hành trình) và so sánh giá của các hành trình
  - độ phức tạp lớn $O(n!)$ => tăng nhanh nếu n tăng
- **Thuật giải heuristic** dùng nguyên lý tham lam ($O(n^2)$):
  - từ điểm khởi đầu $\Rightarrow$ liệt kê tất cả các con đường đến các đại lý còn lại $\Rightarrow$ chọn điểm đến gần nhất
  - khi đã đến 1 đại lý, chọn điểm đến tiếp theo nguyên tắc trên
  - lặp cho đến khi nào không còn đại lý nào

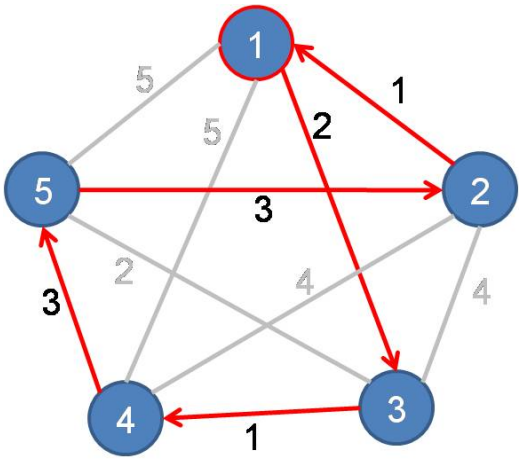
Ví dụ: bài toán người bán hàng



Bài toán



Lời giải theo nguyên lý tham lam (độ dài:12)



Phương án tối ưu (độ dài:10)

Thuật giải này *đôi khi* đưa ra kết quả *không tốt*, thậm chí *rất tệ* (ví dụ: đổi độ dài cạnh 1-4 thành 100)

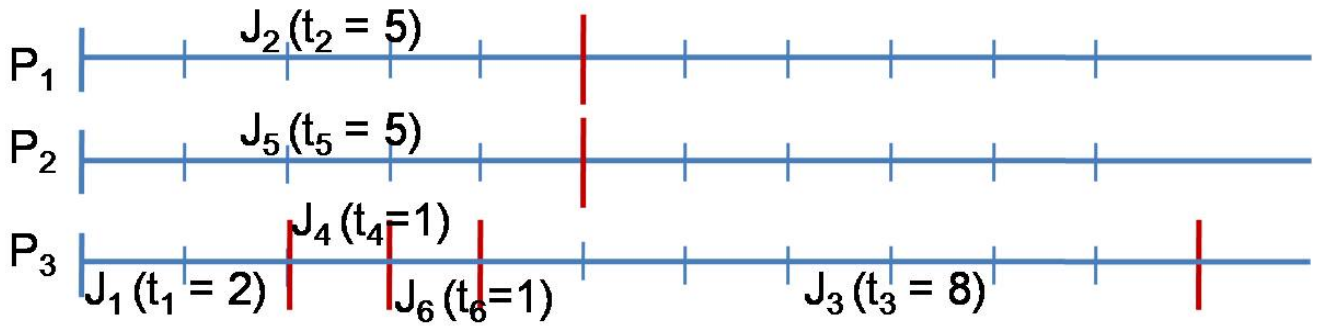
Ví dụ: bài toán phân việc

Bài toán phân việc (bài toán lập lịch) – ứng dụng của nguyên lý thứ tự:

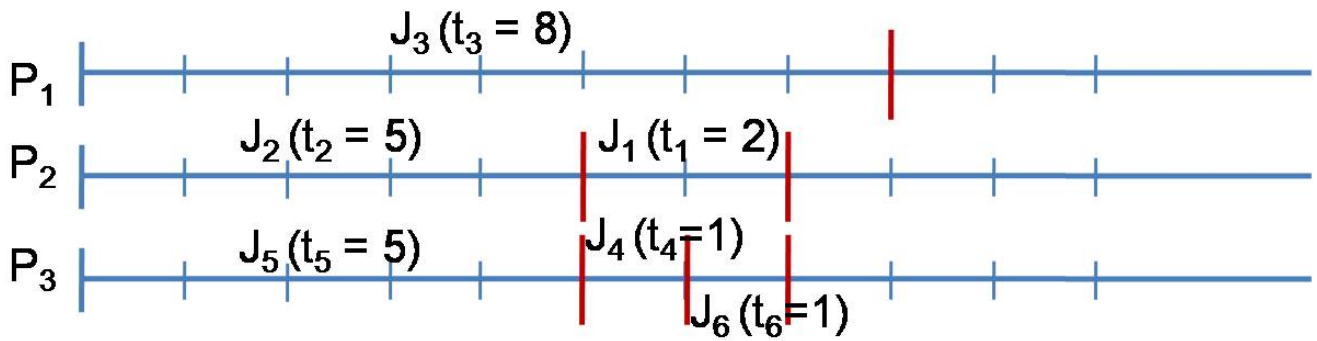
- Bài toán:
  - m công việc $J_1, J_2, ..., J_m$ thực hiện trên n máy $P_1, P_2, ..., P_n$
  - thời gian thực hiện mỗi công việc J_i là t_i, giống nhau trên các máy
- Tìm **phương án tối ưu**: khó, phải dùng kỹ thuật phức tạp
- Thuật giải **heuristic**:
  1. sắp xếp các công việc theo thứ tự giảm dần về thời gian gia công
  2. lần lượt sắp xếp các việc theo thứ tự đó vào máy còn dư nhiều thời gian nhất

Ví dụ: bài toán phân việc

- Ví dụ 1 phương án ($m = 6, n = 3, t_i = \{2, 5, 8, 1, 5, 1\}$):



- Phương án theo thuật giải heuristic:



(phương án tối ưu)

- Sai số tối đa: $T/T^0 \leq 4/3 - 1/3n$ (hiếm khí xảy ra)

$\Rightarrow$ Lời giải tương đối tốt

Ví dụ: bài toán ta-canh

Bài toán Ta-canh (9-puzzle)[1] - ứng dụng của hàm Heuristic:

- **chưa có** lời giải chính xác, tối ưu
- **thuật giải heuristic** đơn giản:
  - ở mỗi trạng thái k tính giá trị hàm heuristic F_k cho 4 trạng thái tiếp ($F_{KT}, F_{KD}, F_{KTr}, F_{KP}$) theo có thể có
  - chọn các đi có giá trị hàm heuristic nhỏ nhất, nếu có nhiều hơn 1 phương án có giá trị nhỏ nhất => chọn ngẫu nhiên từ các phương án đó

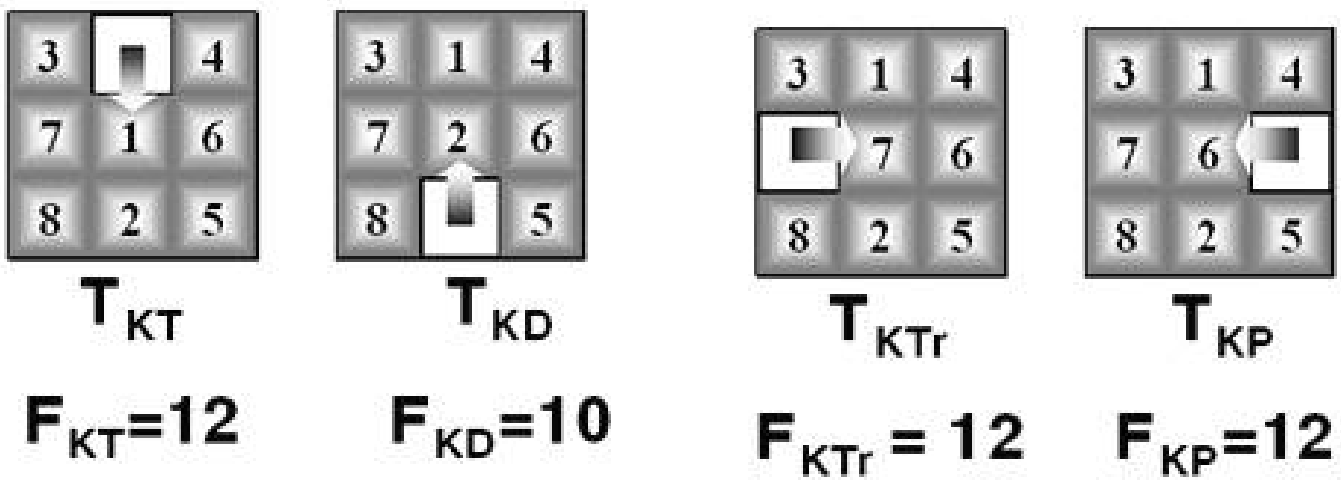
1

http://www4.hcmut.edu.vn/~huynhqlinh/TinhocDC/THDC13/Bai03_6.htm#tacanh

Ví dụ: bài toán ta-canh

- Hàm heuristic: $F_k = \sum_{i,j=1, V(i,j)<>0}^{3} d(i,j)$

  $V(i,j)$: giá trị tại ô (i,j) , ô trống: V(i,j) = 0
  $d(i,j)$: số ô cần di chuyển để đưa con số ở ô (i,j) về đúng vị trí của nó



Tin học đại cương

Bài 4: Giải quyết bài toán - Phần 2: Giải quyết bài toán

NGUYỄN Thị Oanh

oanhnt@soict.hut.edu.vn

Bộ môn Hệ thống thông tin - Viện CNTT và Truyền Thông
Đại học Bách Khoa Hà nội

2010 - 2011

Nội dung

Khái niệm

- "Bài toán" vs. "Vấn đề"
  - *vấn đề* có nghĩa rộng hơn *bài toán*
  - 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, ...
- Pitago chia mọi vấn đề thành 2 loại:
  - *Theorema*: vấn đề cần được khẳng định tính đúng / sai, ví dụ: chứng minh định lý trong toán học
  - *Problema*: vấn đề cần tìm được giải pháp để đạt được một mục tiêu xác định từ những điều kiện ban đầu nào đó, ví dụ: bài toán dựng hình, tìm đường đi ngắn nhất, ...

Khái niệm

- Biểu diễn *vấn đề - bài toán*:

  $A \rightarrow B$

  A: giả thiết, điều kiện ban đầu

  B: kết luận, mục tiêu cần đạt được

  $\rightarrow$: suy luận và giải pháp cần xác định

- Giải quyết *vấn đề - bài toán*:
  - từ A dùng một số hữu hạn các bước suy luận có lý hoặc hành động thích hợp để đạt được B
  - trong tin hoc:

    A: **đầu vào** (*input*) B: **đầu ra** (*output*)

    $\rightarrow$: chương trình tạo thành từ các lệnh cơ bản của máy tính cho phép biến đổi A thành B (cách mã hóa lại thuật toán hay thuật giải)

Giải quyết bài toán bằng máy tính

- Máy tính không thể dùng, ít nhất là trực tiếp, để giải quyết các vấn đề liên quan đến hành động vật lý hoặc biểu thị cảm xúc
- Máy tính chỉ làm được những gì mà nó được bảo phải làm. Máy tính không tự thông minh, nó không thể tự phân tích vấn đề và đưa ra giải pháp
- Con người (lập trình viên) là người **phân tích vấn đề**, tạo ra **các chỉ dẫn** để giải quyết vấn đề (chương trình), và máy tính sẽ thực hiện các chỉ dẫn đó

Thiết kế thuật toán

- Tính **không xác định** của *vấn đề-bài toán*:
  - **A, B** không đầy đủ, rõ ràng
  - thông báo về các **điều kiện** đặt ra cho giải thuật thường được **nêu trong bài toán**
- **Thiết kế** thuật toán/ thuật giải:
  - chủ yếu là do con người: thông tin (tiềm ẩn hoặc rõ ràng) trong A, B + tri thức => thuật giải
  - **tự động hóa** xây dựng thuật toán / thuật giải: biểu diễn bài toán + tri thức liên quan **tường minh** và **đầy đủ** ==> xây dựng **trí tuệ nhân tạo** cho máy tính

Các bước giải quyết bài toán bằng máy tính

Không chỉ đơn giản là lập trình, phức tạp, gồm nhiều bước:

- B1: Xác định yêu cầu 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
- B3: Xây dựng thuật toán
- B4: Lập trình: mô tả thuật giải bằng chương trình
- B5: Kiểm thử và hiệu chỉnh chương trình:
- B6: Triển khai và bảo trì

Các bước giải quyết bài toán bằng máy tính

Không chỉ đơn giản là lập trình, phức tạp, gồm nhiều bước:

- B1: Xác định yêu cầu bài toán
- B2: Lựa chọn phương pháp giải
  - 1 bài toán có nhiều phương án khác nhau về **thời gian thực hiện, chi phí lưu trữ DL, độ chính xác, ...**
  - **phương pháp phù hợp** tùy theo nhu cầu khả năng xử lý tự động, ...
- B3: Xây dựng thuật toán
- B4: Lập trình: mô tả thuật giải bằng chương trình
- B5: Kiểm thử và hiệu chỉnh chương trình:
- B6: Triển khai và bảo trì

Các bước giải quyết bài toán bằng máy tính

Không chỉ đơn giản là lập trình, phức tạp, gồm nhiều bước:

- B1: Xác định yêu cầu bài toán
- B2: Lựa chọn phương pháp giải
- B3: Xây dựng thuật toán
  xây dựng mô hình **chặt chẽ, chính xác, chi tiết** cho phương pháp đã chọn: dữ liệu vào ? các bước và trình tự thực hiện?
- B4: Lập trình: mô tả thuật giải bằng chương trình
- B5: Kiểm thử và hiệu chỉnh chương trình:
- B6: Triển khai và bảo trì

Các bước giải quyết bài toán bằng máy tính

Không chỉ đơn giản là lập trình, phức tạp, gồm nhiều bước:

- B1: Xác định yêu cầu bài toán
- B2: Lựa chọn phương pháp giải
- B3: Xây dựng thuật toán
- B4: Lập trình: mô tả thuật giải bằng chương trình
- B5: Kiểm thử và hiệu chỉnh chương trình:
- B6: Triển khai và bảo trì

Các bước giải quyết bài toán bằng máy tính

Không chỉ đơn giản là lập trình, phức tạp, gồm nhiều bước:

- B1: Xác định yêu cầu bài toán
- B2: Lựa chọn phương pháp giải
- B3: Xây dựng thuật toán
- B4: Lập trình: mô tả thuật giải bằng chương trình
- B5: Kiểm thử và hiệu chỉnh chương trình: chạy thử và hiệu chỉnh sai sót
- B6: Triển khai và bảo trì

Các bước giải quyết bài toán bằng máy tính

Không chỉ đơn giản là lập trình, phức tạp, gồm nhiều bước:

- B1: Xác định yêu cầu bài toán
- B2: Lựa chọn phương pháp giải
- B3: Xây dựng thuật toán
- B4: Lập trình: mô tả thuật giải bằng chương trình
- B5: Kiểm thử và hiệu chỉnh chương trình:
- B6: Triển khai và bảo trì

2 giao đoạn chính để hiện thực hóa bài toán

- Giai đoạn quan niệm:
  - phân tích
  - lựa chọn mô hình
  - xây dựng giải thuật
  - cài đặt
- Giai đoạn khai thác và bảo trì: sử dụng => cải tiến, mở rộng

Các phương pháp giải quyết bài toán bằng máy tính

Giải bài toán theo hướng trực tiếp xác định lời giải

- Xác định lời giải thông qua:
  - các thủ tục tính toán (công thức, hệ thức, định lý, ...)
  - hoặc một dãy hữu hạn các thao tác sơ cấp (dựng hình, phản ứng hóa học, ...)
- Ví dụ:
  - giải PT bậc 2 theo định lý VIET
  - phương pháp lặp để toán tính nghiệm gần đúng, ...

Giải bài toán theo hướng tìm kiếm lời giải

- Nguyên lý thử-sai
- Ứng dụng rộng rãi cho các bài toán-vấn đề phức tạp
- Một số phương pháp điển hình:
  - phương pháp **liệt kê** (hay vét bạn): thử tất cả các khả năng có thể để kiểm tra
  - phương pháp **thử ngẫu nhiên**:
    - dựa trên một số khả năng được chọn ngẫu nhiên
    - khả năng thành công phụ thuộc: chiến lược chọn và điều kiện cụ thể của bài toán
  - phương pháp **quay lui**:
    - đánh dấu các thử nghiệm thất bại và thử khả năng khác (vd: tìm đường trong mê cung)
    - khi không thể xác định hoặc liệt kê từ trước tất cả các khả năng có thể
- Ví dụ: bài toán 8 hậu, phân tích ra thừa số nguyên tố, ...

Độ phức tạp thuật toán

- Độ phức tạp thuật toán
  - không gian: tùy thuộc vào cấu trúc dữ liệu sử dụng khi cài đặt
  - thời gian

$\Rightarrow$ Độ phức tạp về thời gian của thuật toán: khó xác định chính xác

  - *số lượng thao tác* được sử dụng trong thuật toán: phép so sánh 2 số nguyên, cộng, nhân, chia hai số nguyên hay bất kỳ một thao tác cơ bản
  - số lượng thao tác này phụ thuộc vào *độ lớn của dữ liệu* nhập (n)

  $\Rightarrow$ $t(n)$

- Thời gian nhỏ nhất: thời gian thực hiện TT trong trường hợp **tốt nhất** đối với dữ liệu có dữ liệu nhập có độ lớn n
- Thời gian lớn nhất: thời gian thực hiện TT trong trường hợp **xấu nhất** đối với dữ liệu có dữ liệu nhập có độ lớn n

Độ phức tạp thuật toán

- Ví dụ: TT tìm giá trị lớn nhất trong dãy gồm n số nguyên
  - số các thao tác: số lần các phép so sánh và phép gán
  - trường hợp xấu nhất: $t(n) = 1 + 2(n-1) = 2n + 1$
  - trường hợp tốt nhất: $T(n) = 1 + (n-1) = n$

  $\Rightarrow$ Độ phức tạp TT: $O(n)$
- Ví dụ khác:
  - $t(n) = 20n^2 + 5n + 1$
  - $T(n) = n^2 + 10n + 1$

  $\Rightarrow$ $t(n), T(n)$ tăng giống n^2 khi n tăng $\Rightarrow$ độ phức tạp TT: $O(n^2)$

Độ phức tạp thuật toán

- **Định nghĩa**: Cho $f(n), g(n), n \in N$, 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 **f** va **g**

- **Định lý**: Nếu $f(n)$ là hàm đa thức bậc k thì $f(n) = O(n^k)$
- Một số thuật ngữ:
  - Độ phức tạp **hằng**: $O(1)$
  - Độ phức tạp **logarith**: $O(logn)$
  - Độ phức tạp **tuyến tính**: $O(n)$
  - Độ phức tạp **n log n**: $O(nlogn)$
  - 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**: $O(n!)$

Phân loại

- Bài toán **đa thức**
- Bài toán **không đa thức**
- Bài toán **NP**

Phân loại

Bài toán **đa thức**:

- bài toán có thể giải được với giải thuật có độ phức tạp bị chặn bởi 1 đa thức (*lớp đa thức*) => thời gian chạy đa thức
- VD: bài toán sắp xếp
- bài toán có độ phức tạp $O(1), O(logn), O(n), O(nlogn), O(n^b)$ THUỘC lớp bài toán đa thức
- bài toán đa thức được xem là các bài toán có *lời giải thực tế*

Phân loại

Bài toán **không đa thức**:

- bài toán có thể giải được nhưng không thuộc lớp đa thức. VD:
  - bài toán **xác định tất cả** các tập con không rỗng từ tập **n phần tử**
  - số bước cần thiết ít nhất $2^n - 1$
  - $n = 8 \Rightarrow \sim 65000$ bước
  - $n = 32 \Rightarrow \sim 4$ tỷ bước, $n = 33 \Rightarrow \sim 8$ tỷ bước !!!!!!

Phân loại

Bài toán **NP**:

- Định nghĩa: Một bài toán khi *được giải bằng một thuật toán KHÔNG tự quyết* mà có độ phức tạp thuộc lớp ĐA THỨC thì được gọi là một bài toán **đa thức không tự quyết** hay viết tắt là **bài toán NP**
- Thuật toán *KHÔNG tự quyết*: tại 1 bước của cách giải, có một số lựa chọn mà thuật toán không thể tự xác định được
- Ví dụ: bài toán người bán hàng phải đi đến n đại lý, và tổng số km không vượt ngưỡng cho trước
  - C1: liệt kê => độ phức tạp KHÔNG đa thức
  - C2: thuật toán KHÔNG TỰ QUYẾT, độ phức tạp đa thức

  1. Chọn 1 con đường và tính độ dài L
  2. Nếu L <= ngưỡng thì báo là thành công, ngược lại báo chọn lựa sai

Questions & Answers

Tin học đại cương

Bài 5: Tổng quan về ngôn ngữ C

NGUYỄN Thị Oanh

oanhnt@soict.hut.edu.vn

Bộ môn Hệ thống thông tin - Viện CNTT và Truyền Thông
Đại học Bách Khoa Hà nội

2010 - 2011

Nội dung

Lịch sử phát triển

- Ra đời tại phòng thí nghiệm BELL của tập đoàn AT&T (Hoa Kỳ)
- Do Brian W. Kernighan và Dennis M. Ritchie phát triển vào đầu 1970, hoàn thành 1972
- C dựa trên nền các ngôn ngữ **BCPL** (*Basic Combined Programming Language*) và ngôn ngữ **B**
- Tên là ngôn ngữ C như là sự tiếp nối ngôn ngữ B

Lịch sử phát triển

- 1978: C được giới thiệu trong phiên bản đầu của cuốn sách "*The C programming language*"
- Sau đó, C được bổ sung thêm những tính năng và khả năng mới
  $\Rightarrow$ Đồng thời tồn tại nhiều phiên bản nhưng không tương thích nhau
- 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**: ANSI C hay C chuẩn hay C89

Lịch sử phát triển

- Các phiên bản ngôn ngữ C
  - ANSI C: C chuẩn (1989)
  - Các phiên bản khác thường bổ sung thêm thư viện của ANSI C
- Hiện nay cũng có nhiều phiên bản của ngôn ngữ C khác nhau, gắn liền với một bộ **chương trình dịch cụ thể** của ngôn ngữ C
  - Turbo C++ và Borland C++ của Borland Inc
  - MSC và VC của Microsoft Corp
  - GCC của GNU project ...

Đặc điểm của ngôn ngữ lập trình C

- Đặc điểm:
  - Ngôn ngữ lập trình hệ thống
  - Tính khả chuyển, linh hoạt cao
  - Có thế mạnh trong xử lý dữ liệu số, văn bản, cơ sở dữ liệu
- C thường được sử dụng để viết các chương trình hệ thống
  - Hệ điều hành Unix có 90% mã C, 10% hợp ngữ
  - Các trình điều khiển thiết bị (*device driver*)
  - Xử lý ảnh...

Ví dụ

```
#include<stdio.h>
#include<conio.h>

void main(){
        printf("Hello World!\n");
        getch();

}
```

Demo

Tập ký tự

- Tập ký tự là tập các phần tử cơ bản tạo nên chương trình
  - Tổ hợp các ký tự → **từ** (*include*, *void*, *main*, *printf*, ...)
  - Liên kết các từ theo cú pháp → **câu lệnh**
  - Tổ chức các câu lệnh → **chương trình**
- Tập các ký tự trong C:
  - các chữ cái hoa và thường: A ... Z a ... z
  - 10 chữ số: 0 1 2 ... 9
  - các ký hiệu toán học: + - * / = < >
  - dấu ngăn cách: . ; , : space tab
  - các dấu ngoặc: ( ) [ ] { }
  - các kí hiệu đặc biệt: _ ? $ & # ^{} ' " ~ .v.v.

Từ khóa (Keyword)

- Các từ có sẵn trong ngôn ngữ lập trình
- Dùng dành riêng cho các mục đích xác định
  - 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**, ...
- Trong **C** các từ khóa đều dùng viết bằng **chữ thường**

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	singed	void
default	goto	sizeof	volatile
do	if	static	while

Định danh (identifier - Tên)

- Tên : dãy các ký tự dùng để đặt tên cho các đối tượng (*biến, hằng số, hàm, kiểu dữ liệu, ...*) trong chương trình
- Tên được đặt bởi:
  - ngôn ngữ lập trình (các từ khóa)
  - người lập trình

Quy tắc đặt tên, trong C

- Quy tắc:
  - Các ký tự được sử dụng: **chữ cái**, **chữ số** và **dấu gạch dưới** (_)
  - Ký tự **bắt đầu**: chữ cái hoặc _
  - KHÔNG được trùng với **từ khóa**
  - Turbo C++: không quy định độ dài, nhưng chỉ lấy **32 ký tự đầu**
- C phân biệt chữ hoa và chữ thường: ABC khác Abc

- VD tên **hợp lệ**: gia_tri_1, x, i, danh_sach, _MY_CONSTANT, PI, ...
- VD tên **không hợp lệ**: 1A, 55x, danh sach, danh-sach

Lưu ý:

- Nên đặt tên dễ gợi nhớ
- Dùng dấu _khi có nhiều từ để dễ đọc
- Một số quy ước thường được dùng:
  - HẰNG: dùng chữ cái HOA. VD: PI, _MY_CONSTANT, HANG_SO
  - biến, hàm, cấu trúc: dùng chữ cái thường. VD: a, i, giai_thua, sinh_vien

Kiểu dữ liệu

- Định nghĩa:
  - 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: Kiểu **int** trong C
  - số nguyên, có giá trị từ $-32,768$ (-2^{15}) đến $32,767$ $(2^{15}-1)$
  - các phép toán: - (đảo dấu), **+**, **-** , ***, **/** (chia lấy phần nguyên), **%** (chia lấy phần dư), **>**, **<**, **>=**, **<=**, **==**, **!=**

Ví dụ

```
#include<stdio.h>
#include<conio.h>
#define PI 3.14

void main(){
        float r, s;
        printf("Nhap ban kinh hinh tron ");
        scanf("%f",&r);
        s = PI * r * r;
        printf("Dien tich hinh tron %f",s);
        getch();
}
```

Hằng

- Hằng (*constant*): là đại lượng có giá trị không đổi trong chương trình
- Biểu diễn **hằng số nguyên**:

Số thập phân	Số bát phân	Số thập lục phân
2007	03727	0x7D7
396	0614	0x18C

- Biểu diễn **hằng số thực**:
  - số thực dấu phẩy tĩnh : 3.14, 123.45
  - số thực dấu phẩy động: 31.4 E-1 , 1.2345 E+2

Hằng

- Biểu diễn **hằng ký tự**:
  - C1: ký tự đặt giữa 2 dấu nháy đơn: 'A' , '\'', 'z'
  - C2: số thứ tự trong bảng mã ASCII: 65, **0x**41, **0**101

Ký tự cần biểu diễn	**C1**	**C2**
chữ A	'A'	65, **0x**41, **0**101
dấu nháy đơn	'\''	39, **0x**27, **0**47
dấu tab	'\t'	0, **0x**09, **0**11

- Biểu diễn **hằng xâu ký tự**:
  - sử dụng dấu nháy kép:
  - ví dụ: "Đại học Bách Khoa Hà nội", "Tin học đại cương"

Ví dụ

```
#include<stdio.h>
#include<conio.h>
void main(){
   float a, b, x;
   printf("Nhap he so a khac 0, a =");
   scanf("%f",&a);
   printf("Nhap he so b =");
   scanf("%f",&b);
   x = -b/a;
   printf("Nghiem cua phuong trinh %f",x);
   getch();
}
```

? a, b, x là hằng số ?

Biến

- **Biến** (*variable*) là đại lượng mà giá trị có thể thay đổi trong chương trình
- Chú ý:
  - Hằng số và biến được sử dụng để **lưu trữ dữ liệu** trong chương trình
  - Hằng số và biến phải thuộc một **kiểu dữ liệu** nào đó
  - Hằng số và biến đều phải **đặt tên theo quy tắc**

Hàm

- **Hàm** (*function*) là một chương trình con có chức năng nhận dữ liệu đầu vào (các tham số đầu vào), thực hiện một chức năng nào đó và đưa ra các kết quả
- Ví dụ:

Hàm	**Ý nghĩa**	**Ký hiệu toán học**	**Ví dụ**
pow(x,y)	x mũ y	x^y	pow(2,3) = 8
sin(x)	sin của x	$\sin(x)$	sin(0) = 0
sqrt(x)	căn bậc 2 của x	$\sqrt{x}$	sqrt(4) = 2

Biểu thức

- Định nghĩa: Biểu thức là sự ghép nối các toán tử (*operator*) và các toán hạng (*operand*) theo một quy tắc xác định
- Các toán hạng có thể là biến, hằng
- Các toán tử: +, -, *, /, ...

- Ví dụ: **chieu_dai * chieu_rong * chieu_cao**

Câu lệnh (*Statement*)

- **Câu lệnh** (*statement*) diễn tả một hoặc một nhóm các thao tác trong giải thuật
- Cuối mỗi câu lệnh bắt buộc có dấu chấm phẩy '**;**' để đánh dấu kết thúc câu lệnh: $x = a + b;$
- Phân nhóm:
  - câu lệnh **đơn**: gán, +, -, ...
  - câu lệnh **phức**: câu lệnh chứa các câu lệnh khác (lệnh khối, if ... else ..., for ... , while ...,)
    lệnh khối : các lệnh được đặt trong **{ }**
- **Chương trình** được tạo thành từ **dãy các câu lệnh**

Chú thích (*Comment*)

- Chú thích (*comment*):
  - Lời mô tả, giải thích vắn tắt cho một câu lệnh, một đoạn chương trình hoặc cả chương trình
  - Giúp việc đọc và hiểu chương trình dễ dàng hơn
  - Chú thích **không phải là câu lệnh** $\Rightarrow$ không ảnh hưởng tới chương trình
- Trong C có hai cách viết:
  - chú thích trên **1 dòng**:

    ```
    // nội dung chú thích được tính từ dấu "//" đến cuối dòng
    ```
  - chú thích trên **nhiều dòng**:

    ```
    /* nội dung chú thích,
    có thể viết trên nhiều dòng */
    ```

```c
/* Chuong trinh sau se nhap vao tu ban phim 2 so nguyen
va hien thi ra man hinh tong, hieu cua 2 so nguyen vua nhap vao */
#include<stdio.h>
#include<conio.h>

void main()
{
    // khai bao cac bien trong chuong trinh
   int a, b, tong, hieu;
   // Nhap vao tu ban phim 2 so nguyen
   printf("\n Nhap vao so nguyen thu nhat: "); scanf("%d",&a);
   printf("\n Nhap vao so nguyen thu hai: "); scanf("%d",&b);
   // Tinh tong, hieu, tich cua 2 so vua nhap
   tong = a + b;
   hieu = a - b;
   // Hien thi cac gia tri ra man hinh
   printf("\n Tong cua 2 so vua nhap la %d", tong);
   printf("\n Hieu cua 2 so vua nhap la %d", hieu);
   getch(); // Cho nguoi su dung an phim bat ki de ket thuc
}
```

```c
// PHAN 1:  Khai bao tap tieu de
#include<stdio.h>

// PHAN 2: Dinh nghia kieu du lieu mo
typedef struct{
    char* name;
    int age;
}person;

// PHAN 3: Khai bao cac ham nguyen mau
int power(int, int);

// PHAN 4: Khai bao cac bien toan cuc
float y;

// PHAN 5 (PHAN BAT BUOC): Ham main
int main() {
    ... // Cac khai bao, cac lenh
}
// PHAN 6: Noi dung cac ham da khai bao nguyen mau
int power(int x, int y){
    return (x*y);
}
```

Cấu trúc cơ bản của chương trình C

- PHẦN 1: Khai báo **các tập tiêu đề**
  - thông báo cho chương trình dịch biết những thư viện sẽ sử dụng trong CT
  - Ví dụ:

    #include <stdio.h> // *chứa các hàm thực hiện thao tác vào ra DL*

    #include <conio.h> // *chứa các hàm của DOS*
- PHẦN 2: **Định nghĩa các kiểu dữ liệu** mới
- PHẦN 3: **Khai báo các hàm nguyên mẫu**
  - thông báo cho chương trình dịch biết các hàm và các thông tin cơ bản về các hàm sẽ được xây dựng trong CT

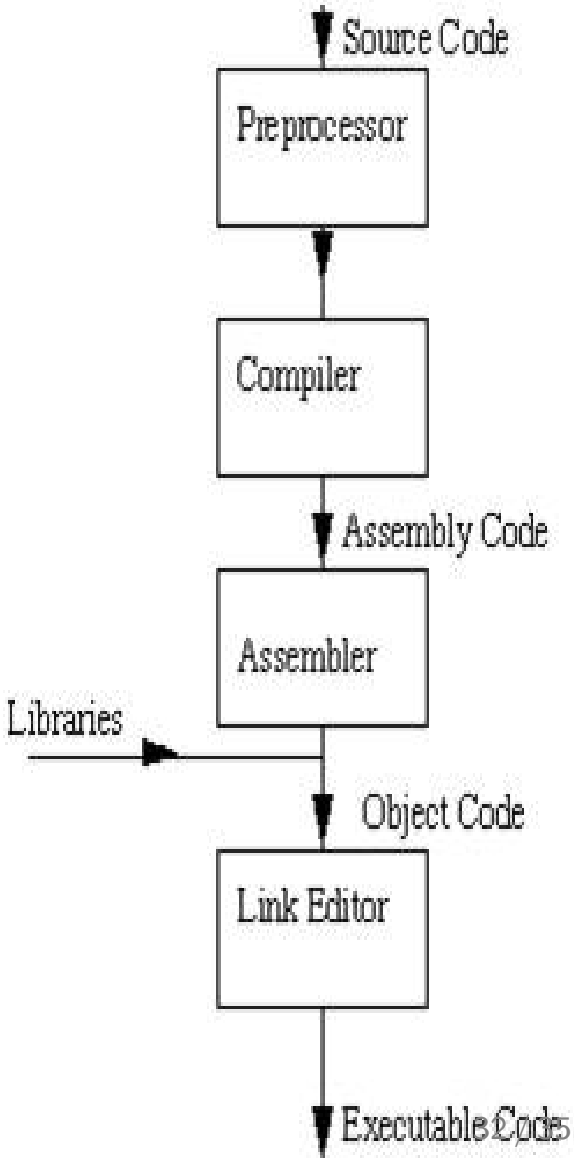
Cấu trúc cơ bản của chương trình C

- PHẦN 4: Khai báo **các biến toàn cục**
  - Ví dụ:
    ```
    int a;
    float x, y;
    ```
- PHẦN 5: Hàm **main**
  - các lệnh trong hàm **main** sẽ được thực hiện đầu tiên
  - trong hàm **main** mới gọi các hàm khác
- PHẦN 6: **Định nghĩa các hàm đã khai báo nguyên mẫu**
  - cài đặt (viết mã) cho các hàm đã khai báo

Biên dịch chương trình C

- Preprocessor
  - loại bỏ các chú thích
  - dịch các chị thị tiền xử lý bắt đầu là #
- C Compiler
  - biên dịch mã nguồn thành mã assembly.
- Assembler
  - tạo ra mã object
  - trên UNIX → file .o, trên MS-DOS → file.OBJ
- Link Editor
  - *Link editor* kết hợp các hàm thư viện với hàm **main()** để tạo ra tệp có thể thực thi được

  → trong MS-DOS là file .exe



Trình biên dịch Turbo C++

- Trình biên dịch (*compiler*): dịch mã nguồn (*source code*) thành file thực thi
- Các trình biên dịch C phổ biến:
  - **Turbo C++** của hãng Borland
  - MSC của Microsoft
  - GCC của GNU
  - Dev C++ của Bloodshed Software
- Turbo C++ có nhiều phiên bản
  - phiên bản lựa chọn: Turbo C++ 3.0

Cài đặt và sử dụng

- Cài đặt: *www.google.com* :)
- Sử dụng:
  - mở file mới để soạn thảo CT: chọn **File/New**
  - lưu chương trình với 1 tên file: nhấn **F2**
  - biên dich: nhấn **F9**
  - chạy chương trình: **Ctrl + F9**

Demo

Questions & Answers

Tin học đại cương

Bài 6: Kiểu dữ liệu và biểu thức trong C

NGUYỄN Thị Oanh

oanhnt@soict.hut.edu.vn

Bộ môn Hệ thống thông tin - Viện CNTT và Truyền Thông
Đại học Bách Khoa Hà nội

2010 - 2011

Nội dung

Các kiểu dữ liệu chuẩn trong C

Kiểu DL	**Ý nghĩa**	**Kích thước**[1]	**Miền giá trị**
unsigned char	*ký tự không dấu*	1 byte	$0 \div 255$
char	*Kí tự*	1 byte	$-128 \div 127$
unsigned int	*Số nguyên không dấu*	2 bytes	$0 \div 65,535$
short int	*Số nguyên có dấu*	2 bytes	$-32,768 \div 32,767$
int	*Số nguyên có dấu*	2 bytes	$-32,768 \div 32,767$

[1]compiler 16bit, dùng **sizeof**(kieu du lieu) để biết kích thước chính xác

Các kiểu dữ liệu chuẩn trong C

unsigned long	*Số nguyên không dấu*	4 bytes	$0 \div 4,294,967,295$
long	*Số nguyên có dấu*	4 bytes	$-2,147,483,648 \div 2,147,483,647$
float	*Số thực dấu phẩy động, độ chính xác đơn*	4 bytes	$\pm 3.4E-38 \div \pm 3.4E+38$
double	*Số thực dấu phẩy động độ chính xác kép*	8 bytes	$\pm 1.7E-308 \div \pm 1.7E+308$
long double	*Số thực dấu phẩy động*	10 bytes	$\pm 3.4E-4932 \div \pm 1.1E+4932$

Khai báo biến

- Một biến trước khi sử dụng phải được khai báo
- Cú pháp khai báo:
  - hoặc: **kiểu_dữ_liệu** *tên_biến*;
  - hoặc: **kiểu_dữ_liệu** *tên_biến_1*, *tên_biến_2*, ..., *tên_biến_n*;
- Ví dụ:
  - **int** *a*;
  - **float** *x*, *y*, *z*;

Kết hợp khai báo và khởi tạo biến

- Cú pháp:
  - hoặc: **kiểu_dữ_liệu** *tên_biến = giá_trị_khởi_tạo*;
  - hoặc:
    kiểu_dữ_liệu *tên_biến_1= giá_trị_khởi_tạo_1, ...,*
    tên_biến_n = giá_trị_khởi_tạo_n;
- Ví dụ:
  - **int** *a = 5*;
  - **float** *x=5.0, y=7.6*;

Khai báo hằng

- Cách 1: dùng từ khóa **#define**
  - Cú pháp khai báo: **#define** *TÊN_HẰNG giá_trị*
  - **KHÔNG** có dấu **chấm phẩy (;)** ở cuối
- Ví dụ:

```
#define MAX_SINHVIEN 50
#define CNTT "Công nghệ thông tin"
#define DIEM_CHUAN 23.5
```

Khai báo hằng

- Cách 2: dùng từ khóa **const**
  - Cú pháp khai báo: **const** *kiểu_dữ_liệu TÊN_HẰNG = giá_trị* **;**
  - **CÓ** dấu **;** ở cuối lệnh
- Ví dụ:

const int *MAX_SINHVIEN = 50* **;**
const char *CNTT[20] = "Công nghệ thông tin"* **;**
const float *DIEM_CHUAN = 23.5* **;**

Chú ý

- Giá trị của các hằng phải được xác định ngay khi khai báo
- Trong chương trình, KHÔNG thể thay đổi được giá trị của hằng
- #define là chỉ thị tiền xử lý (*preprocessing directive*)
  - **dễ đọc**, **dễ thay đổi**
  - **dễ chuyển đổi** giữa các nền tảng phần cứng hơn
  - **tốc độ nhanh** hơn

Biểu thức số học

- là biểu thức mà giá trị của nó là cái đại lượng số học (số nguyên, số thực)
- các toán tử là các phép toán số học (cộng, trừ, nhân, chia...), các toán hạng là các đại lượng số học (**số, biến, hằng**)
- Ví dụ: a, b, c là các biến thuộc một kiểu dữ liệu số nào đó

  $3 * 3.7$

  $8 + 6/3$

  $a + b - c, \ldots$

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)
  - Giá trị nguyên **khác 0**: Đúng (**TRUE**)
  - Giá trị **0**: Sai (**FALSE**).
- Các phép toán logic gồm có;
  - AND: VÀ logic, kí hiệu là &&
  - OR: HOẶC logic, kí hiệu là ||
  - NOT: PHỦ ĐỊNH, kí hiệu là !

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, ...
- chỉ có thể nhận giá trị là một trong 2 giá trị: Đúng (**TRUE**) hoặc Sai (**FALSE**)

$\Rightarrow$ Biểu thức quan hệ là một trường hợp riêng của BIỂU THỨC LOGIC

Ví dụ- Biểu thức logic

Biểu thức logic	Giá trị
$5 > 7$	FALSE
$9 != 10$	?
$2 >= 2$	?
$a > b$	
$a + 1 > a$	?
$(5 > 7)\&\&(9 != 10)$	?
$0 \| \| 1$	?

Biểu thức logic	Giá trị
$(5 > 7) \| \| (9 != 10)$	?
0	FALSE
$!0$	?
3	?
$!3$	?
$(a > b)\&\&(a < b)$	?

Sử dụng biểu thức

- Làm vế phải của **lệnh gán**
- Làm toán hạng trong các **biểu thức**
- Làm tham số thực trong **lời gọi hàm**
- Làm chỉ số cho các cấu trúc lặp: **for**, **while**, **do ... while**
- Làm biểu thức kiểm tra cho các cấu trúc rẽ nhánh **if**, **switch**

Các phép toán trong C

- 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**
- Các phép toán khác: phép **gán**, **lấy địa chỉ**, ...

Phép toán số học

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)

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 1)
			`1 | 0`	(có giá trị 1)
			`1 | 1`	(có giá trị 1)
			`101 | 110`	(có giá trị 111)

Phép toán trên bit

^	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 << n	(có giá trị $a*2^n$)
			101 << 2	(có giá trị 10100)
>>	Phép DỊCH PHẢI nhị phân	Số nhị phân	a >> n	(có giá trị $a/2^n$)
			101 >> 2	(có giá trị 1)
~	Phép ĐẢO BIT nhị phân (lấy Bù 1)	Số nhị phân	~ 0	(có giá trị 1)
			~ 1	(có giá trị 0)
			~ 110	(có giá trị 001)

Phép toán quan hệ

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 >= 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)

Phép toán logic

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 0) 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)

Phép gán

- Cú pháp:

  tên_biến = biểu_thức;

- Lấy *giá trị* của **biểu_thức** gán cho **tên_biến**
- Ví dụ:

  int a, b, c;

  $a = 3;$

  $b = a + 5;$

  $c = a * b;$

Phép gán

- Biểu thức gán là biểu thức nên nó cũng **có giá trị**
- **Giá trị của biểu thức gán** bằng giá trị của **biểu_thức**

  $\Longrightarrow$ 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);
```

Phép gán

- Phép toán gán thu gọn:

  $x = x + y;$ giống như : $x \mathrel{+}= y;$

- Dạng lệnh gán thu gọn này còn áp dụng được với các phép toán khác: $+, -, *, /, \%, >>, <<, \&, |, \hat{}$

Thứ tự ưu tiên các phép toán

Mức	Các toán tử	Trật tự kết hợp
1	() [] . -> ++ (hậu tố) -- hậu tố	------>
2	! ~ ++ (tiền tố) -- (tiền tố) - *(dereference) & sizeof	<------
3	* / %	------>
4	+ -	------>
5	<< >>	------>
6	< <= > >=	------>
7	== !=	------>
8	&	------>
9	^	------>
10	|	------>
11	&&	------>
12	||	------>
13	?:	<------
14	= += -=	<------

http://www.difranco.net/cop2220/op-prec.htm

Thứ tự ưu tiên các phép toán

- Ví dụ 1:

int x=5, z=7, y;

$y = x * z{+}{+};$ $\Rightarrow$ $y = ?, z = ?, x = ?$

$y = x * {+}{+}z;$ $\Rightarrow$ $y = ?, z = ?, x = ?$

$y = x * (x + z);$ $\Rightarrow$ $y = ?, z = ?, x = ?$

$y = x * x + z;$ $\Rightarrow$ $y = ?, z = ?, x = ?$

- Ví dụ 2:

a < 10 && 2 * b < c $\Rightarrow$ *(a < 10) && ((2 * b) < c)*

Phép tăng/giảm 1 đơn vị

- Tăng 1 đơn vị:

```
ten_bien = ten_bien+1;
ten_bien++;
++ten_bien;
```

- Giảm 1 đơn vị:

```
ten_bien = ten_bien-1;
ten_bien-- ;
--ten_bien;
```

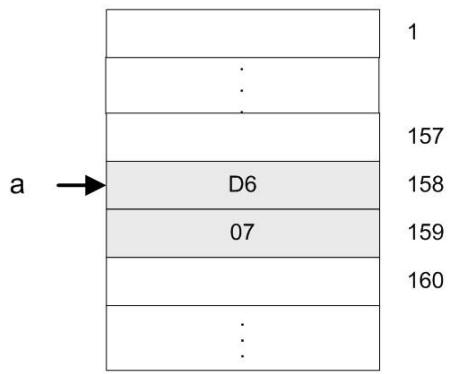
Phép tăng giảm 1 đơn vị

- Biểu thức dạng tiền tố:
  - thay đổi giá trị của biến **trước khi sử dụng**
- Biểu thức dạng hậu tố:
  - 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;
```

Phép toán lấy địa chỉ (&)

- **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**
  $\Longrightarrow$ Mọi biến đều có địa chỉ : địa chỉ của **ô nhớ đầu tiên** trong vùng nhớ dành cho biến
- Cú pháp : **&ten_bien;**
- Ví dụ:



int a= 2006; // = 0x07D6
$\Longrightarrow$ giá trị của **&a** là **158** hay **9E**

Phép chuyển đổi kiểu bắt buộc

- Khi cần thiết chương trình dịch có thể tự động chuyển kiểu: **char** $\rightarrow$ **int** $\rightarrow$ **long int** $\rightarrow$ **float** $\rightarrow$ **double** $\rightarrow$ **long double**
- Các trường hợp khác, nếu muốn chuyển đổi kiểu $\Rightarrow$ sử dụng phép chuyển đổi kiểu (**ép kiểu**):
  - dễ gây ra kết quả không mong muốn (sai dữ liệu):

```
long int a = 50000;
int b;                  // số nguyên: −32768 ÷ +32767
b= (int) a;             // b = −15536!!!!!!!
```

  - chỉ sử dụng khi thật cần thiết; những người mới bắt đầu nên tránh dùng ép kiểu
- Cú pháp:

(kiểu_dữ_liệu_mới) biểu_thức;

Phép chuyển đổi kiểu bắt buộc

```
#include <stdio.h>
#include <conio.h>
void main()
{
    long int li;    int i;  float f; clrscr();
    li = 50000; f = 123.456;
    i = (int) li;
    printf("\n li = %ld; i = %d",li, i);

    i = (int) f;
    printf("\n f = %f; i = %d",f, i);
    getch();
}
```

Kết quả:

li = 50000; i = -15536

f= 123.456001; i = 123

Biểu thức điều kiện

- Cú pháp:
  biểu_thức_1 ? biểu_thức_2 : biểu_thức_3
- Giá trị của BT ĐK:
  - là giá trị của **biểu_thức_2** nếu giá trị của **biểu_thức_1 khác 0** (giá trị logic: ĐÚNG)
  - là giá trị của **biểu_thức_3** nếu giá trị của **biểu_thức_1 bằng 0** (giá trị logic: SAI)
- Ví dụ:

```
float x, y, z;          // khai báo biến
x = 3.8; y = 7.6;       // 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
```

Các lệnh vào ra dữ liệu với các biến

- C cung cấp 2 hàm vào ra cơ bản:
  - printf(): hàm ra
  - scanf(): hàm vào
- Muốn sử dụng 2 hàm **printf()** và **scanf()** ta cần khai báo tệp tiêu đề **stdio.h**:

  #include <stdio.h>
  hoặc
  #include "stdio.h"

Hàm **printf**

- Mục đích:
  - Hiển thị ra màn hình các loại dữ liệu cơ bản như: Số, kí tự và xâu kí tự
  - Một số hiệu ứng hiển thị đặc biệt: *xuống dòng, sang trang,...*
- Cú pháp:
  printf(xau_dinh_dang [, danh_sach_tham_so]);

  xau_dinh_dang: quy định cách thức hiển thị dữ liệu ra màn hình máy tính
  danh_sach_tham_so: danh sách các biến sẽ hiển thị lên màn hình theo cách thức quy định trong *xau_dinh_dang*

Ví dụ

```
#include <conio.h>
#include <stdio.h>
void main()
{   int a = 5;
    float x = 1.234;
    printf("Hien thi mot so nguyen %d va mot so thuc %f",a,x);
    getch();
}
```

Hàm **printf**

- *xau_dinh_dang* chứa:
  - Các kí tự **thông thường**: Được hiển thị ra màn hình
  - Các nhóm kí tự **định dạng**: Xác định quy cách hiển thị các tham số trong phần danh_sach_tham_so
  - Các kí tự **điều khiển**: Dùng để tạo các hiệu ứng hiển thị đặc biệt: xuống dòng ('\n') hay sang trang ('\f'), ...
- **1 nhóm kí tự định dạng** chỉ dùng cho **1 kiểu dữ liệu**. Ví dụ:
  - **%d** dùng cho kiểu nguyên
  - **%f** dùng cho kiểu thực
- 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

Hàm **printf**

- *danh_sach_tham_so* phải phù hợp với các nhóm kí tự định dạng trong *xau_dinh_dang* về :
  - Số lượng
  - Kiểu dữ liệu
  - Thứ tự

Một số nhóm ký tự định dạng phổ biến

Nhóm kí tự định dạng	Kiểu dữ liệu	Kết quả
%c	int, char	Kí tự đơn lẻ
%i, %d	int, char	Số thập phân
%o	int, char	Số bát phân (không có 0 đằng trước)
%x, %X	int, char	Số hexa (chữ thường/chữ hoa)
%u	unsigned int/char	Số thập phân

Một số nhóm ký tự định dạng phổ biến

Nhóm kí tự định dạng	Kiểu dữ liệu	Kết quả
%ld, %li	long	Số thập phân
%lo	long	Số bát phân (không có 0 đằng trước)
%lx, %LX	long	Số hexa (chữ thường/chữ hoa)
%lu	unsigned long	Số thập phân

Một số nhóm ký tự định dạng phổ biến

Nhóm kí tự định dạng	Kiểu dữ liệu	Kết quả
%s	char []	Hiển thị xâu kí tự kết thúc bởi '\0'
%f	float/double	Số thực dấu phẩy tĩnh
%e, %E	float/double	Số thực dấu phẩy động
%		Hiển thị kí tự %

Độ rộng hiển thị tối thiểu

- C tự động xác định chỗ cần thiết để hiển thị đủ nội dung
- Số nguyên, ký tự, xâu ký tự: %**m**d, %**m**c, %**m**s ($m >= 0$). Ví dụ:
  int a = 1234;
  printf("%5d", a); // dành 5 chỗ để hiển thị a
  printf("\n%5d", 34); // xuống dòng, dành 5 chỗ để hiển thị giá trị 34
- Số thực: **%m.nf** ($m, n >= 0$)

  m: độ rộng cho **cả số thực** (phần nguyên, phần thập phân, dấu chấm)
  n: độ rộng cho phần thập phân
  Ví dụ:*printf("%5.2d", 31.23);*
  Ví dụ:*printf("%.2d", 31.23);*
  Ví dụ:*printf("%7.2d", 31.23);*

Độ rộng hiển thị tối thiểu

Lưu ý:

- Khi độ rộng chỉ ra trong phần định dạng nhỏ hơn số chỗ thực sự cần thiết: **C tự động cấp thêm chỗ mới** để hiển thị
- Nếu không: các vị trí còn dư thì được hiểu thị bằng dấu trắng (space)
- Ví dụ:

Lệnh	**Hiển thị**
printf("%f", 12.345);	12.345
printf("%6.3f", 12.345);	12.345
printf("%8.3f", 12.345);	␣␣12.345
printf("%8.5f", 12.345);	12.34500
printf("%8.1f", 12.345);	␣␣␣␣12.3
printf("%8.2f", 12.345);	␣␣␣12.35

Căn lề

- Căn lề **phải**: mặc định
- Căn lề **trái**: thêm dấu **-** ngay sau dấu **%** trong phần định dạng

Lệnh	**Hiển thị**
printf("%8.3f", 12.345);	␣␣12.345
printf("%-8.3f", 12.345);	12.345␣␣
printf("%8.1f", 12.345);	␣␣␣␣12.3
printf("%-8.1f", 12.345);	12.3␣␣␣␣

printf("%-15s %5.2f", 9, "Nguyen Van A", 7.5,);
printf("%15s %5.2f", 9, "Nguyen Van A", 7.5,);
Kết quả hiển thị:

```
Nguyen Van A␣␣␣␣7.50
␣␣␣Nguyen Van A␣7.50
```

Hàm **scanf()**

- Mục đích:

  hàm scanf() dùng để **nhập dữ liệu từ bàn phím**
- Cú pháp:
  scanf(xau_dinh_dang,[danh_sach_dia_chi]);
- Ví dụ:
  scanf("%d %f", &a, &b);

Hàm **scanf()**

- **xau_dinh_dang**:
  - gồm các ký tự được qui định cho từng **loại dữ liệu** được nhập vào (định dạng giống trong hàm **printf()**)
  - Ví dụ: kiểu nguyên: **%d**
- **danh_sach_dia_chi**:
  - địa chỉ các biến (toán tử **&**) phân tách nhau bởi dấu **,**
- **danh_sach_dia_chi** phải phù hợp với các nhóm kí tự định dạng trong **xau_dinh_dang** về:
  - Số lượng
  - Kiểu dữ liệu
  - Thứ tự
- Ví dụ:
  scanf("%d %f", &a, &b);

Hàm **scanf()** - Quy tắc đọc

- Đọc SỐ:
  - mọi ký tự số, dấu chấm (.): các ký tự hợp lệ
  - khi gặp các dấu phân cách: *tab*, *xuống dòng* hay *dấu cách* (*space bar*) thì scanf() sẽ hiểu là **kết thúc** nhập dữ liệu cho một số
- Đọc KÝ TỰ:

  mọi ký 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*
- Đọc XÂU:
  - 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 xâu** kí tự
- $\Rightarrow$ Trước khi nhập dữ liệu kí tự hay xâu kí tự ta nên dùng lệnh **fflush(stdin);** để xóa bộ đệm

Ví dụ

```
#include <conio.h>
#include <stdio.h>
void main() {
    // khai bao bien
    int a; float x; char ch; char str[30];
    // Nhap du lieu
    printf(" Nhap vao mot so nguyen: "); scanf("%d",&a);
    printf("\n Nhap vao mot so thuc: "); scanf("%f",&x);
    printf("\n Nhap vao mot ki tu: ");
    fflush(stdin); scanf("%c",&ch);
    printf("\n Nhap vao mot xau ki tu: ");
    fflush(stdin); scanf("%s",str);

    // Hien thi du lieu vua nhap vao
    printf("\n Nhung du lieu vua nhap vao");
    printf("\n So nguyen: %d",a);
    printf("\n So thuc : %.2f",x);
    printf("\n Ki tu: %c",ch);
    printf("\n Xau ki tu: %s",str);
    getch();
}
```

Một số hàm vào ra khác

- Hàm **gets()**:
  - dùng để nhập vào từ bàn phím một xâu kí tự bao gồm cả dấu cách (hàm **scanf()** KHÔNG làm được)
  - Cú pháp: **gets (ten_bien);**
  - Ví dụ:

```
char str[40];
printf("Nhap vao mot xau ki tu:");
fflush(stdin); gets(str);
```

Một số hàm vào ra khác

- Hàm **puts**(*xâu_kí_tự*):
  - Hiển thị ra màn hình nội dung *xâu_kí_tự* và sau đó đưa con trỏ xuống dòng mới
  - Cú pháp: **puts(xâu_kí_tự);**
  - Ví dụ:
    puts("Nhap vao xau ki tu:");
    $\Longleftrightarrow$ *printf("%s\n","Nhap vao xau ki tu:");*

Một số hàm vào ra khác

- Hàm **getch()**:
  - thường dùng để chờ người sử dụng ấn một phím bất kì rồi sẽ kết thúc chương trình
  - Cú pháp: **getch();**
- Để sử dụng các hàm **gets()**, **puts()**, **getch()** ta cần khai báo tệp tiêu đề **conio.h**
  #include<conio.h>
  hoặc
  #include"conio.h"

Questions & Answers

Tin học đại cương

Bài 7: Các cấu trúc lập trình trong C

NGUYỄN Thị Oanh

oanhnt@soict.hut.edu.vn

Bộ môn Hệ thống thông tin - Viện CNTT và Truyền Thông
Đại học Bách Khoa Hà nội

2010 - 2011

Nội dung

Cấu trúc lệnh khối

- Thể hiện **cấu trúc tuần tự**
- Lệnh khối là dãy các câu lệnh được đặt trong cặp dấu ngoặc nhọn **{...}**

```
{
    lenh_1;
    lenh_2;
    ...
    lenh_n;
}
```

- **C** cho phép khai **báo biến trong lệnh khối**, nhưng phần khai báo phải nằm trước câu lệnh

Cấu trúc lệnh khối

- Lệnh khối lồng nhau
  - Trong một lệnh khối có thể chứa lệnh khối khác
  - Sự lồng nhau là không hạn chế

```
{
    lenh_1;
    ...
    {
        lenh_11;
        ...
        lenh_1n;
    }
    ...
    lenh_n;
}
```

```c
#include <conio.h>
#include <stdio.h>
void main()
//Noi dung cua ham main() cung la mot khoi lenh
{
   // khai bao bien
   int c;
   c = 10;
   printf("Gia tri cua c = %d day la c ngoai",c);
   // bat dau mot khoi lenh khac
   {
      int c;
      c = 10;
      printf("\n Gia tri cua c = %d day la c trong",c);
      printf("\n Tang gia tri cua c them 10 don vi");
      c = c + 10;
      printf("\n Gia tri cua c = %d day la c trong",c);
   }
   printf("\n Gia tri cua c = %d day la c ngoai",c);
   getch();
}// ket thuc khoi lenh cua ham main()
```

Kết quả ?

Cấu trúc if, if . . . else

- Cú pháp cấu trúc **if**:

```
if (bieu_thuc_dieu_kien)
   lenh;
```

```
if (bieu_thuc_dieu_kien)
      {
            ...
      }
```

- Cú pháp cấu trúc **if . . . else**:

```
if (bieu_thuc_dieu_kien)
   lenh_1;
else
   lenh_2;
```

```
if (bieu_thuc_dieu_kien)
      {
            ...
      }
else
      {
            ...
      }
```

Ví dụ: tìm số lớn nhất

```
#include <conio.h>
#include <stdio.h>
void main()
{
   // khai bao bien
   float a, b;
   float max;
   printf("Nhap gia tri a va b: ");
   scanf("%f %f",&a,&b);
    if(a<b)
       max = b;
   else
       max = a;
   printf("\nSo lon nhat trong 2 so %.0f va %.0f la %.0f",a,b,max);
   getch();
} //ket thuc ham main()
```

Cấu trúc lựa chọn switch

- Cú pháp cấu trúc **switch**:

switch (*bieu_thuc*)
{
case gia_tri_1: lenh_1; [break];
...
case gia_tri_n: lenh_n; [break];
[**default**]: lenh_n+1; [break];]
}

Cấu trúc lựa chọn switch

- Giá trị của biểu thức kiểm tra (bieu_thuc) phải là **số nguyên**:

  Phải có kiểu dữ liệu là *char*, *int*, *long*
- Tương ứng các giá trị sau **case** (gia_tri_1, gia_tri_2,. . .) cũng phải là **số nguyên**
- **Ví dụ:** Nhập vào số nguyên không âm, đưa ra ngày trong tuần tương ứng (theo số dư khi chia cho 7)

Cấu trúc lựa chọn switch

```
#include <conio.h>
#include <stdio.h>
void main()
{
    int a;
    printf("\nNhap mot gia tri so nguyen khong am: ");
    scanf("%d",&a);
    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();
}
```

Cấu trúc lựa chọn switch

- Bài tập:
  - Trong một năm các tháng có 30 ngày là 4, 6, 9, 11 còn các tháng có 31 ngày là 1, 3, 5, 7, 8, 10, 12. Riêng tháng hai có thể có 28 hoặc 29 ngày
  - Hãy viết chương trình nhập vào 1 tháng, sau đó đưa ra kết luận tháng đó có bao nhiêu ngày.

Cấu trúc lựa chọn switch

Chữa

Vòng lặp for

- Mục đích: thực hiện **lặp đi lặp lại** một công việc nào đó với **số lần lặp xác định**
- Cú pháp:

```
for (bieu_thuc_1; bieu_thuc_2; bieu_thuc_3)
{
    day_cac_lenh;
}
```

- Trong đó:

bieu_thuc_1: Khởi tạo giá trị ban đầu cho vòng lặp
bieu_thuc_2: Điều kiện tiếp tục vòng lặp
bieu_thuc_3: Thực hiện bước tăng của vòng lặp
Chú ý các biểu thức 1, 2, 3 có thể CÓ hoặc KHÔNG

Vòng lặp for - Ví dụ

Ví dụ: Đưa ra màn hình các số nguyên lẻ nhỏ hơn 100

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i;
    for(i = 1;i<100;i++)
    {
        if(i%2 == 1)
            printf("%5d",i);
        if((i+1)%20 == 0)
            printf("\n");
    }
    getch();
}
```

Vòng lặp for - Ví dụ

Ví dụ: Đưa ra màn hình các số nguyên lẻ nhỏ hơn 100
Cải tiến ?

Vòng lặp **while**

- Mục đích: thực hiện **lặp đi lặp lại** một công việc nào đó với **số lần lặp không xác định**
- Cú pháp: 2 dạng

```
while (bieu_thuc_dieu_kien)
{
    lenh;
}
```

hoặc

```
do
{
    lenh;
}
while (bieu_thuc_dieu_kien)
```

while vs. do... while

- **while**:
  - Kiểm tra điều kiện vòng lặp (tức là giá trị của biểu thức) trước rồi mới thực hiện lệnh
  - Các **lenh** sau **while** có thể không được thực hiện lần nào

- **do... while**:
  - Thực hiện *lenh* trước rồi mới kiểm tra *bieu_thuc_dieu_kien* của vòng lặp
  - Các *lenh* sau **do** được thực hiện ít nhất 1 lần dù *bieu_thuc_dieu_kien* có giá trị như thế nào

Ví dụ

- Ví dụ: Nhập vào điểm của một sinh viên, nếu điểm đó $\notin [0, 10]$ thì thông báo cho người dùng **nhập lại**

- Cách làm:
  - Nếu dùng lệnh **if** $\Rightarrow$ Chỉ kiểm tra được 1 lần
  - Không dùng **for** được vì chưa biết trước số lần lặp
    $\Rightarrow$ sử dụng vòng lặp **while**

Ví dụ

```c
#include <stdio.h>
#include <conio.h>
void main()
{
    float diem;
    printf("Chuong trinh nhap diem sinh vien\n");
    do
    {
        printf("Nhap diem (0<=diem<=10):");
        scanf("%f",&diem);
        if (diem < 0 || diem > 10)
            printf("\nBan nhap khong dung!\n");
    }
    while (diem < 0 || diem > 10);
    printf("\nDiem ban vua nhap la: %.2f", diem);
    getch();
}
```

Ví dụ

- sử dụng while ?
- sửa chương trình xác định ngày trong tháng ? Yêu cầu tháng nhập vào phải từ 1 đến 12

Các lệnh thay đổi cấu trúc lập trình

- Dùng cho các lệnh lặp: **while**, **do** ... **while**, **for**
- Thay đổi việc thực hiện lệnh trong vòng lặp

  $\Rightarrow$ C cung cấp:

  continue;

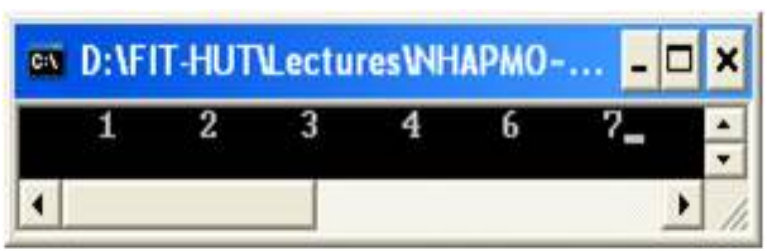
  break;

Các lệnh thay đổi cấu trúc lập trình

- **continue**
  - **Bỏ qua** việc thực hiện các **câu lệnh nằm sau** lệnh **continue** trong thân vòng lặp
  - Chuyển sang thực hiện một **vòng lặp mới**
- **break**
  - Thoát **khỏi vòng lặp** ngay cả khi biểu thức điều kiện của vòng lặp vẫn còn được thỏa mãn

Ví dụ

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i; clrscr();
    for(i = 1;i<=10;i++)
    {
        if(i == 5)
            continue;
        printf("%5d",i);
        if(i==7)
            break;
    }
    getch();
}
```



Bài tập

Nhập vào 1 số nguyên. Kết luận số đó là số nguyên tố hay là hợp số?

Questions & Answers

Tin học đại cương

Bài 8: Mảng và xâu ký tự

NGUYỄN Thị Oanh

oanhnt@soict.hut.edu.vn

Bộ môn Hệ thống thông tin - Viện CNTT và Truyền Thông
Đại học Bách Khoa Hà nội

2010 - 2011

Nội dung

Khái niệm mảng

- Tập hợp hữu hạn các phần tử **cùng kiểu**, **lưu trữ kế tiếp** nhau trong bộ nhớ
- Các phần tử trong mảng có **cùng tên** (là tên mảng) nhưng **phân biệt** với nhau ở **chỉ số** cho biết **vị trí** của nó trong mảng
- Ví dụ:
  - Bảng điểm của sinh viên
  - Vector
  - Ma trận

Khai báo và sử dụng mảng

- Khai báo mảng (một chiều):
  kiểu_dữ_liệu *tên_mảng* [kích_thước_mảng];
- Trong đó:
  - kiểu_dữ_liệu: kiểu dữ liệu của các phần tử trong mảng
  - tên_mảng: tên của mảng
  - kích_thước_mảng: số phần tử trong mảng
- Ví dụ:
  int *mang*[6]; // khai báo mảng 6 phần tử có kiểu dữ liệu int

Khai báo và sử dụng mảng

- Cấp phát bộ nhớ:
  - được cấp phát các **ô nhớ kế tiếp nhau** trong bộ nhớ
  - biến mảng lưu trữ địa chỉ của ô nhớ đầu tiên được cấp phát
- Ngôn ngữ C **đánh chỉ số** các phần tử bắt đầu từ **0**:
  - phần tử thứ *i* của mảng *mang* được xác định bởi: *mang[i-1]*

mang[0]	mang[1]	mang[2]	mang[3]	mang[4]	mang[5]

Khai báo và sử dụng mảng

- Mỗi phần tử của mảng cũng là một mảng $\Rightarrow$ mảng nhiều chiều
- Khai báo: *kieu_du_lieu ten_mang* $[size_1][size_2]...[size_k]$;
- Ví dụ:

```
int a[6][5]; /* mảng a gồm 6 phần tử, mỗi phần tử là một mảng số nguyên gồm 5 phần tử */

int b[3][4][5]; // b là mảng 3 chiều
```

Khai báo và sử dụng mảng

- Sử dụng mảng
  - Truy cập vào phần tử thông qua **tên mảng** và **chỉ số** của phần tử trong mảng: *mang[chỉ_số_phần_tử]*
  - Chú ý: chỉ số **bắt đầu từ 0**
- Ví dụ:

  int a[4]; int b[3][4];

  - phần tử *đầu tiên* (thứ nhất) của mảng: a[0]
  - phần tử *cuối cùng (thứ tư)* của mảng: a[3]
  - *a[i]*: là phần tử thứ *i+1* của a

  - phần tử *đầu tiên* (thứ nhất) của mảng b: b[0]
  - phần tử *đầu tiên* (thứ nhất) của mảng b[0]: b[0][0]
  - *b[i][j]*: là phần tử thứ **j+1** của b[i], **b[i]** là phần tử thứ **i + i** của **b**

Nhập dữ liệu

- Khởi tạo giá trị của mảng ngay khi khai báo:

  int a[3] = {1, 2, 3};
  int b[2][3] = {{1 2 3}, {4, 5, 6}};

  - số lượng giá trị khởi tạo $\leq$ số lượng phần tử trong mảng
  - nếu nhỏ hơn, các phần tử còn lại **nhận giá trị 0** (NULL, '\0')
- Có thể xác định kích thước mảng thông qua số giá trị khởi tạo:

  int array1 [8] = {2, 4, 6, 8, 10, 12, 14, 16};
  int array2 [] = {2, 4, 6, 8, 10, 12, 14, 16};

Nhập dữ liệu

- Nhập dữ liệu từ bàn phím bằng hàm **scanf**:

  int a[10];
  - Nhập dữ liệu cho a[1]: **scanf("%d", &a[1]);**
  - Nhập dữ liệu cho toàn bộ phần tử của mảng a $\Rightarrow$ sử dụng vòng lặp **for**
- Lưu ý:
  - Tên mảng là một *hằng (hằng con trỏ)* do đó **không thể thực hiện phép toán với tên mảng** như phép gán sau khi đã khai báo
  - Nếu số phần tử của mảng được nhập từ bàn phím $\Rightarrow$ khai báo số phần tử tối đa

Nhập dữ liệu

```
#include<stdio.h>
#include<conio.h>
void main(){
    int a[100];
    int n, i;
    do{
        printf("\n Cho biet so phan tu cua mang n ( 0 < n <= 100): "
            );
        scanf("%d",&n);
    } while (n>100||n<=0);
    for(i = 0; i < n; i++){
        printf("a[%d] = ", i);
        scanf("%d",&a[i]);
    }
    getch();
}
```

Nhập dữ liệu

- Xuất ra màn hình:
  - Dùng hàm **printf()**
  - Để hiển thị tất cả các phần tử: dùng vòng **for**
- Ví dụ:
  - Hiển thị một phần tử bất kì
  - Hiển thị tất cả các phần tử, mỗi phần tử trên một dòng
  - Hiển thị tất cả các phần tử trên một dòng, cách nhau 2 vị trí
  - Hiển thị từng k phần tử trên một dòng

Xuất dữ liệu

```
#include <stdio.h>
#define  MONTHS 12
void main(){
    int rainfall[MONTHS], i,max;

    // Nhap du lieu
    for ( i=0; i < MONTHS; i++ ){
        printf("Luong mua (mm) cua thang %d: ", i+1);
        scanf("%d", &rainfall[i] );
    }
    // Xuat du lieu
    printf("\n");
    for (i=0; i < MONTHS; i++ )
        printf( "%2d " , rainfall[i]);

   printf("\n");
   getch();
}
```

Tìm giá trị lớn nhất, nhỏ nhất

- Tìm giá trị **lớn nhất**:
  - Giả sử phần tử đó là phần tử đầu tiên
  - Lần lượt so sánh với các phần tử còn lại
  - Nếu lớn hơn hoặc bằng => so sánh tiếp
  - Nếu nhỏ hơn => coi phần tử này là phần tử lớn nhất và tiếp tục so sánh
  - Cách làm?
- Tìm giá trị **nhỏ nhất**: tương tự

Tìm giá trị lớn nhất, nhỏ nhất

```
// Tim kiem phan tu lon nhat
max = rainfall[0];
for(i = 1; i < MONTHS; i++)
    if(max < rainfall[i])
        max = rainfall[i];

printf("\n Luong mua nhieu nhat la: %d\n", max);
```

Tìm 1 giá trị trên mảng

- Bài toán:
  - Cho mảng dữ liệu a và một giá trị k
  - Tìm các phần tử trong mảng a có giá trị bằng (giống) với k. Nếu có in ra vị trí (chỉ số) các phần tử này. Ngược lại thông báo không tìm thấy
- Cách làm:
  - Duyệt toàn bộ các phần tử trong mảng $\Rightarrow$ vòng lặp for (while, do while)
  - Nếu *a[i]* bằng (giống) k thì lưu lại chỉ số *i* $\Rightarrow$ sử dụng mảng lưu chỉ số
  - Sử dụng một biến để xác định tìm thấy hay không tìm thấy $\Rightarrow$ giá trị **0** hoặc **>= 1**

Tìm 1 giá trị trên mảng I

```c
#include <stdio.h>
#include <conio.h>
void main(){
    int a[100], chi_so[100];
    int n;//n la so phan tu trong mang
    int i, k, kiem_tra;
    // Nhap so phan tu va du lieu trong mang
    do{
        printf("\n Cho biet so phan tu cua mang: ");
        scanf("%d",&n);
    } while (n>100||n<=0);
    for(i = 0; i < n; i++){
        printf("a[%d] = ", i);
        scanf("%d",&a[i]);
    }
    // Gia tri can tim kiem
    printf("\nNhap vao gia tri tim kiem: ");
    scanf("%d",&k);
```

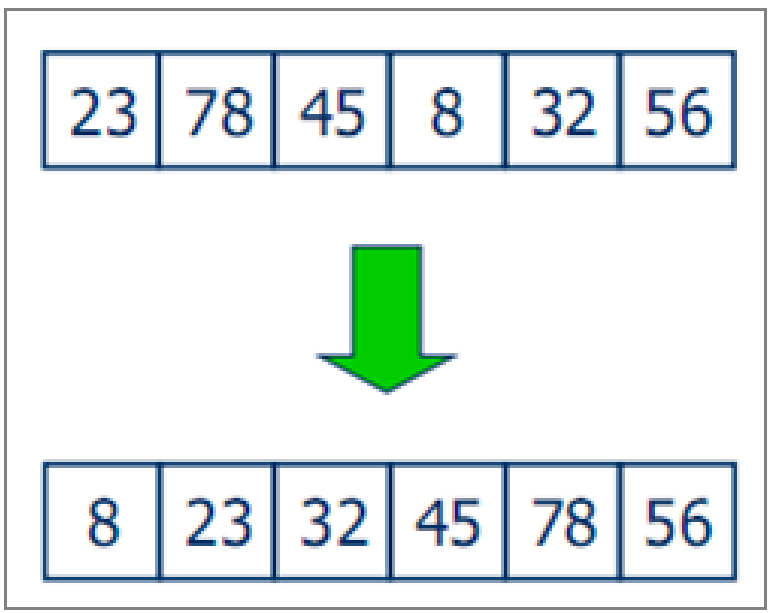
Tìm 1 giá trị trên mảng II

```
    kiem_tra = 0;
    // Duyet qua tat ca cac phan tu trong mang
    for(i = 0;i<n;i++)
        if(a[i] == k){
            chi_so[kiem_tra] = i;
            kiem_tra ++;
        }
     if(kiem_tra > 0){
            printf("Trong mang co %d phan tu co gia tri bang %d",
                kiem_tra,k);
            printf("\nChi so cua cac phan tula:");
            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();
}
```

Sắp xếp các phần tử trên mảng

Bài toán:

- Cho mảng a gồm **n** phần tử. Sắp xếp các phần tử của mảng a theo thứ tự tăng dần/giảm dần

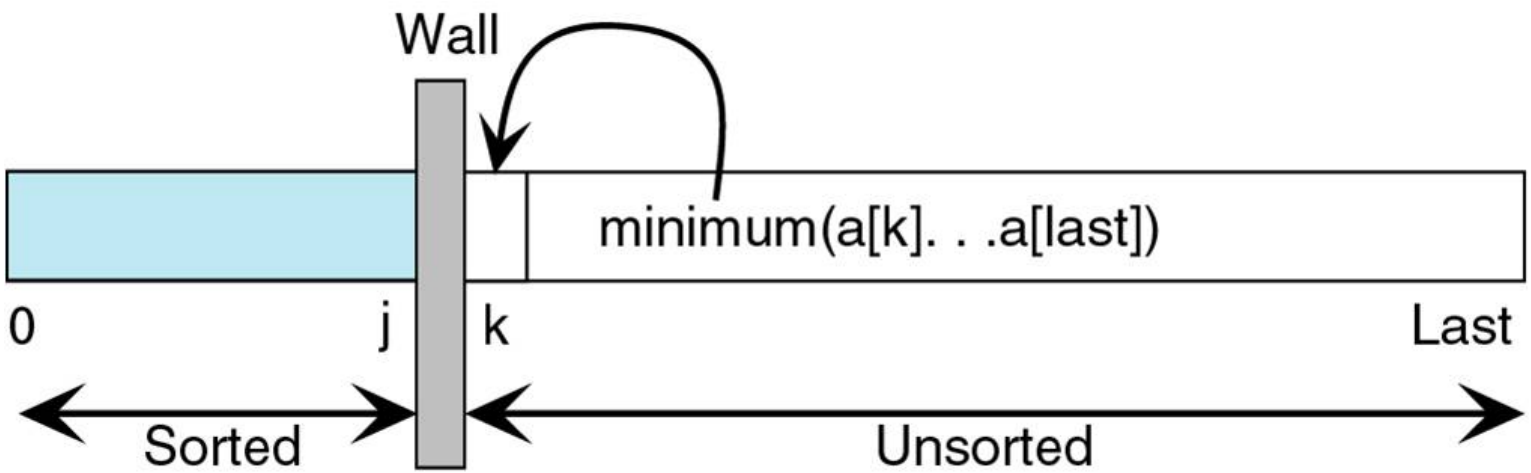


Sắp xếp các phần tử trên mảng

- Các giải thuật sắp xếp:
  - Sắp xếp thêm dần (insertion sort)
  - Sắp xếp lựa chọn (selection sort)
  - Sắp xếp nổi bọt (bubble sort)
  - Sắp xếp vun đống (heap sort)
  - Sắp xếp nhanh (quick sort)
  - Sắp xếp trộn (merge sort)
  - ...

Sắp xếp các phần tử trên mảng

- Giải thuật sắp xếp lựa chọn
  - Tìm phần tử nhỏ nhất chưa được sắp xếp trong mảng
  - Đổi chỗ nó với phần tử đầu tiên trong phần chưa được sắp



Giải thuật sắp xếp lựa chọn

- Lần sắp xếp thứ 1:
  - So sánh $a[0]$ với các $a[i], i = 1..n-1$
  - $a[0] > a[i] \Rightarrow$ **đổi chỗ** $a[0]$ và $a[i]$
  - Thu được a[0] là phần tử nhỏ nhất
- Lần sắp xếp thứ 2:
  - So sánh $a[1]$ với các $a[i], i = 2..n-1$
  - $a[1] > a[i] \Rightarrow$ **đổi chỗ** $a[1]$ và $a[i]$
  - Thu được a[1] là phần tử nhỏ thứ 2
- ...
- Lần sắp xếp thứ **k**:
  - So sánh $a[k-1]$ với các $a[i], i = k..n-1$
  - $a[k-1] > a[i] \Rightarrow$ **đổi chỗ** $a[k-1]$ và $a[i]$
  - Thu được a[k-1] là phần tử nhỏ thứ k
- ...
- Lặp đến $k = n-1$

Giải thuật sắp xếp lựa chọn

Ví dụ: a= { 12, 5, 3, 4 };

	Lượt 1	Lượt 2	Lượt 3
12	**3**	**3**	**3**
5	12	**4**	**4**
3	5	12	**5**
4	4	5	12

Giải thuật sắp xếp lựa chọn

```
//Khai bao cac bien
int a[100], temp;
int i, j;

//Sap xep
for (i = 0; i < n-1; i++)
  for (j = i+1; j < n ; j++)
    if ( a[i] > a[j]){
        tmp = a[i];
        a[i]= a[j];
        a[j]= tmp;
    }
```

VD: Sắp xếp I

```c
#include <stdio.h>
#include <conio.h>
void main(){

    // Khai bao bien
    int m[100],temp;
    int n;    // n la so phan tu trong mang
    int i, j, k;

    // Nhap so phan tu mang
    printf("Cho biet so phan tu co trong mang: ");
    scanf("%d",&n);

     // Nhap gia tri cho cac phan tu trong mang
    for(i  = 0;i<n;i++){
        printf("\n Cho biet gia tri cua m[%d] = ",i);
        scanf("%d",&m[i]);
    }
```

VD: Sắp xếp II

```
    // Hien thi mang vua nhap vao
    printf("Mang truoc khi sap xep\n");
    for(i=0;i<n;i++)
        printf("%3d",m[i]);

    // Sap xep
    for(i = 0; i<n-1; i++){
          for(j = i+1; j<n; j++){
               if(m[i]>m[j]){
                    temp = m[j];m[j] = m[i];m[i] = temp;
               }
          }
          printf("\nMang o luot sap xep thu %d: ",i+1);
          for(k = 0;k < n ;k++)
                    printf("%3d",m[k]);
     }
     getch();
}
```

Khái niệm xâu ký tự

- Xâu ký tự (string) là một dãy các ký tự viết liên tiếp nhau
  - Độ dài xâu là số ký tự có trong xâu
  - Xâu rỗng là xâu không có ký tự nào
- Ví dụ: “Tin hoc”, “String”
- Lưu trữ: kết thúc xâu bằng ký tự **’\0’** hay **NUL**(mã ASCII là 0)

’T’	’i’	’n’		’h’	’o’	c	’\0’

Xâu ký tự vs. Mảng ký tự

- Xâu ký tự và Mảng ký tự
  - Tập hợp các ký tự viết liên tiếp nhau
  - Sự khác biệt: xâu ký tự có ký tự kết thúc xâu, mảng ký tự không có ký tự kết thúc xâu
- Xâu ký tự "A" và ký tự 'A'?
  - 'A' là 1 ký tự
  - "A" là 1 xâu ký tự, ngoài ký tự 'A' còn có ký tự '\0' => gồm 2 ký tự

Khai báo và sử dụng xâu

- Khai báo:
  char ten_xau[so_phan_tu_toi_da];

⇒ Lưu trữ: cần 1 mảng có kích thước: so_phan_tu_toi_da + 1

- Truy cập các phần tử trong xâu:
  ten_xau[chi_so_phan_tu];
- Ví dụ:

```
char quequan[10];
quequan = "Ha noi";
```

⟹ quequan[0] lưu 'H'
quequan[1] lưu 'a'
...
quequan[6] lưu '\0'

Các hàm xử lý ký tự

- Tệp tiêu đề sử dụng: **ctype.h**
- Một số hàm xử lý ký tự:
  - int toupper(int ch): chuyển kí tự thường thành kí tự hoa: **toupper('a') => 'A'**
  - int tolower(int ch): chuyển kí tự hoa thành kí tự thường: **tolower('B') => 'b'**

Các hàm xử lý ký tự

- Một số hàm kiểm tra ký tự:
  - int isalpha(int ch): kiểm tra xem kí tự có phải chữ cái hay không ('a',. . . , 'z', 'A',..., 'Z')
  - int isdigit(int ch): kiểm tra chữ số ('0', '1', ...,'9')
  - int islower(int ch): kiểm tra chữ thường
  - int isupper(int ch): kiểm trachữhoa
  - int iscntrl(int ch): kiểm tra kí tự điều khiển (0-31)
  - int isspace(int ch): kiểm tra dấu trắng: kí tự dấu cách (mã 32), xuống dòng ('\n' 10), đầu dòng ('\r' 13), tab ngang ('\t' 9), tab dọc ('\v' 11)

  trả về giá trị <> 0 nếu đúng, nếu sai trả về 0

Ví dụ xử lý ký tự

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
void main(){
    char ch;
    printf("Nhap vao mot ky tu: ");
    scanf("%c", &ch);

    if(isupper(ch)){
        printf("Ki tu nay la chu hoa\n");
        printf("Ki tu chu thuong tuong ung la: %c\n",tolower(ch));
    }else
        if(islower(ch)){
            printf("Ki tu nay la chu thuong\n");
            printf("Ki tu chu hoa tuong ung la: %c\n",toupper(ch));
        }
    getch();
}
```

Vào ra xâu kí tự

- Tệp tiêu đề: **stdio.h**
- Nhập xâu kí tự:
  - gets(tên_xâu);
  - scanf("%s",&tên_xâu);
- Hiển thị xâu kí tự:
  - puts(tên_xâu);
  - printf("%s",tên_xâu);
- Sự khác nhau giữa **gets** và **scanf**?

Các hàm xử lý xâu

- Tệp tiêu đề: **string.h**
- size_t strlen(char[] tên_xâu): trả về độ dài xâu
- char[] strcpy(char[] xâu_đích, char[] xâu_nguồn): sao chép xâu
- int strcmp(char[] xâu_thứ_nhất, char[] xâu_thứ_hai): so sánh hai xâu
  giá trị 0 : hai xâu giống nhau
  giá trị < 0: xâu thứ nhất nhỏ hơn xâu thứ hai
  giá trị > 0: xâu thứ nhất lớn hơn xâu thứ hai
- char[] strcat(char[] xâu_đích, char[] xâu_nguồn): ghép nối xâu nguồn vào ngay sau xâu đích

Các hàm xử lý xâu

- Tệp tiêu đề: **stdlib.h**
- int atoi(char[] str): chuyển một xâu kí tự thành một số nguyên tương ứng
- int atol(char[] str): chuyển thành số long int
- float atof(char[] str): chuyển thành số thực

  Không **thành công** cả 3 hàm: **trả về 0**

Ví dụ xử lý xâu ký tự

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main(){
    char str1[10] = "abc";
    char str2[10] = "def";
    printf(" str1: %s",str1);
    printf("\n str2: %s",str2);
    printf("\n strcmp(str1,str2)= %d", strcmp(str1,str2));

    printf("\n strcpy(str1,str2) = %s", strcpy(str1,str2));
   printf(" str1: %s",str1);
   printf("\n str2: %s",str2);
   strcpy(str1,"ab");strcpy(str2,"abc");
   printf("\n str1: %s",str1);
   printf(" str2: %s",str2);
   printf("\n strcmp(str1,str2) = %d", strcmp(str1,str2));
   getch();
}
```

Questions & Answers

Tin học đại cương

Bài 9: Cấu trúc

NGUYỄN Thị Oanh

oanhnt@soict.hut.edu.vn

Bộ môn Hệ thống thông tin - Viện CNTT và Truyền Thông
Đại học Bách Khoa Hà nội

2010 - 2011

Nội dung

Khái niệm cấu trúc

- Kiểu dữ liệu cấu trúc (*struct*):
  - Là kiểu dữ liệu phức hợp, bao gồm **nhiều thành phần** có thể thuộc các **kiểu dữ liệu khác nhau**
  - Các thành phần: gọi là **trường dữ liệu** (*field*)
- Ví dụ:
  - Thông tin về **kết quả học tập** môn Tin đại cương của sinh viên: TenSV, MaSV, Diem
  - Thông tin về **cầu thủ**: Ten, Tuoi, CLB, SoAo, Vitri,...

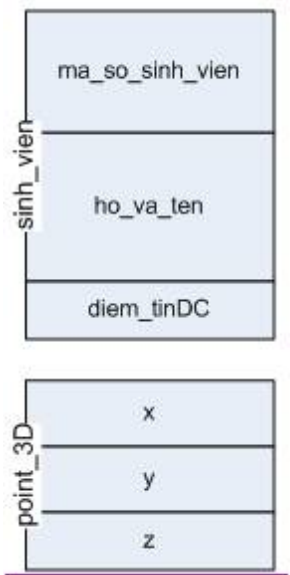
Khai báo **kiểu** dữ liệu cấu trúc

- Khai báo:

struct *ten_cau_truc*{

khai báo các trường DL;

}



- VD:

```
struct sinh_vien{
        char SBD[10];
        char ho_va_ten[40];
        float diem_thi[10];
}

struct point_3D{
        float x;
        float y;
        float z;
}
```

Khai báo **biến** cấu trúc

- Khai báo biến:

  struct *ten_cau_truc* ten_bien_cau_truc**;**

- VD:

  struct sinh_vien a, b, c**;**

- Kết hợp khai báo:

  struct [*ten_cau_truc*]**{**

  khai báo các trường DL;

  } *ten_bien_cau_truc***;**

Khai báo **biến** cấu trúc

- Các cấu trúc có thể được **khai báo lồng** nhau

```
struct diem_thi{
        float dToan, dLy, dHoa;
}
struct sinh_vien{
        char SBD[10];
        char ho_va_ten[40];
        struct diem_thi ket_qua;
} thi_sinh_1, thi_sinh_2;
```

```
struct sinh_vien{
        char SBD[10];
        char ho_va_ten[40];
        struct diem_thi {
                float dToan, dLy, dHoa;
        } ket_qua;
} thi_sinh_1, thi_sinh_2;
```

Định nghĩa kiểu dữ liệu cấu trúc với typedef

- Mục đích:
  - Đặt tên mới cho kiểu dữ liệu cấu trúc
  - Giúp khai báo biến "quen thuộc" và ít sai hơn
- Cú pháp:

  typedef struct *tên_cũ tên_mới*;

  hoặc

  typedef struct [*tên_cũ*] {

  khai báo các trường dữ liệu;

  } *danh_sách_các_tên_mới*;
- *tên_mới* có thể trùng *tên_cũ*

Ví dụ

- Khai báo cấu trúc:

```
struct point_3D {
    float x, y, z;
}
```

⇒ Khai báo biến:

struct point_3D p1;

- Định nghĩa kiểu dữ liệu:

```
typedef struct point_3D {
    float x, y, z;
} diem3D, ten_bat_ky;
```

⇒ Khai báo biến:

diem3D p1;
ten_bat_ky p2;

- Định nghĩa kiểu dữ liệu:
  typedef struct point_3D diem3D;

⇒ Khai báo biến: **diem3D** p1;

Ví dụ

- Lưu ý:
  - p1, p2: các **biến**
  - diem3D, ten_bat_ky: **kiểu dữ liệu** cấu trúc, không phải là tên biến
  - point_3D: **tên cấu trúc**

Truy cập các trường dữ liệu

- Cú pháp: **tên_biến_cấu_trúc.tên_trường**
- Lưu ý:
  - Dấu "**.**": **toán tử truy cập** vào trường dữ liệu trong cấu trúc
  - Nếu trường dữ liệu là một cấu trúc
    $\Rightarrow$ sử dụng tiếp dấu "**.**" để truy cập vào thành phần mức sâu hơn
- Ví dụ: p1.x, p1.y, p1.z

Ví dụ về cấu trúc

- Xây dựng một cấu trúc biểu diễn điểm trong không gian 2 chiều. Nhập giá trị cho một biến kiểu cấu trúc này, sau đó hiển thị giá trị các trường dữ liệu của biến này ra màn hình.
  - Cấu trúc: tên điểm, tọa độ x, tọa độ y
  - Nhập, hiển thị từng trường của biến cấu trúc như các biến dữ liệu khác

Ví dụ về cấu trúc

```
#include<stdio.h>
#include<conio.h>
typedef struct{
     char ten[10];
     int x,y;
}toado;

void main(){
     toado t;
     printf("Nhap thong tin toa do\n");

     printf("Ten diem: ");gets(t.ten);
     printf("Toa do x: ");scanf("%d",&t.x);
    printf("Toa do y: ");scanf("%d",&t.y);
    printf("Gia tri cac truong da nhap vao la:\n");
    printf("%-10s%3d%3d\n",t.ten,t.x,t.y);
    getch();
}
```

Phép gán giữa các biến cấu trúc

- Sao chép dữ liệu từ biến cấu trúc này sang biến cấu trúc khác **cùng kiểu**
  - gán lần lượt từng trường trong hai biến cấu trúc => "thủ công":

```
p2.x = p1.x;
p2.y = p1.y;
p2.z = p1.z;
```

  - C cung cấp phép gán hai biến cấu trúc cùng kiểu:
    bien_1 = bien_2;

Phép gán giữa các biến cấu trúc

```
#include<stdio.h>
#include<conio.h>
typedef struct{
    char hoten[20];
    int diem;
}sinhvien;

void main(){
    sinhvien a,b,c;
    printf("Nhap thong tin sinh vien\n");
    printf("Ho ten: ");gets(a.hoten);
    printf("Diem:");scanf("%d",&a.diem);
     b = a;
    strcpy(c.hoten,a.hoten);
     c.diem = a.diem;
    printf("Bien a: ");
    printf("%-20s%3d\n",a.hoten,a.diem);
    printf("Bien b: ");
    printf("%-20s%3d\n",b.hoten,b.diem);
    printf("Bien c: ");
    printf("%-20s%3d\n",c.hoten,c.diem);
    getch();
}
```

Mảng cấu trúc

- Là tập hợp các phần tử có **cùng kiểu dữ liệu là kiểu cấu trúc**
- Mục đích:
  - Lưu trữ một tập hợp các phần tử có cùng kiểu
  - Mỗi phần tử có các thành phần có thể có giá trị khác nhau: thông tin các sinh viên trong lớp, đội bóng...
- Khai báo:
  struct tên_cấu_trúc tên_biến_mảng [số phần tử];
  kiểu_cấu_trúc tên_biến_mảng [số phần tử];
- Ví dụ: **struct point_3D** mang_3D[10];
  diem3D mang_3D[10];

Bài tập

1. **Khai báo một cấu trúc** gồm: tên và điểm thi Tin đại cương
2. **Khai báo mảng** tên là sv thuộc kiểu trên
3. **Nhập** từ bàn phím số **n** là số sinh viên trong lớp
4. **Nhập thông tin n sinh viên** và **lưu** vào mảng sv
5. **Sắp xếp** mảng theo thứ tự tăng dần của điểm, tên
6. **Hiển thị** ra màn hình danh sách các **sinh viên có điểm >=8**

../VD_C/VD_Cautruc_Baitap.c

Questions & Answers

Tin học đại cương

Bài 10: Hàm

NGUYỄN Thị Oanh

oanhnt@soict.hut.edu.vn

Bộ môn Hệ thống thông tin - Viện CNTT và Truyền Thông
Đại học Bách Khoa Hà nội

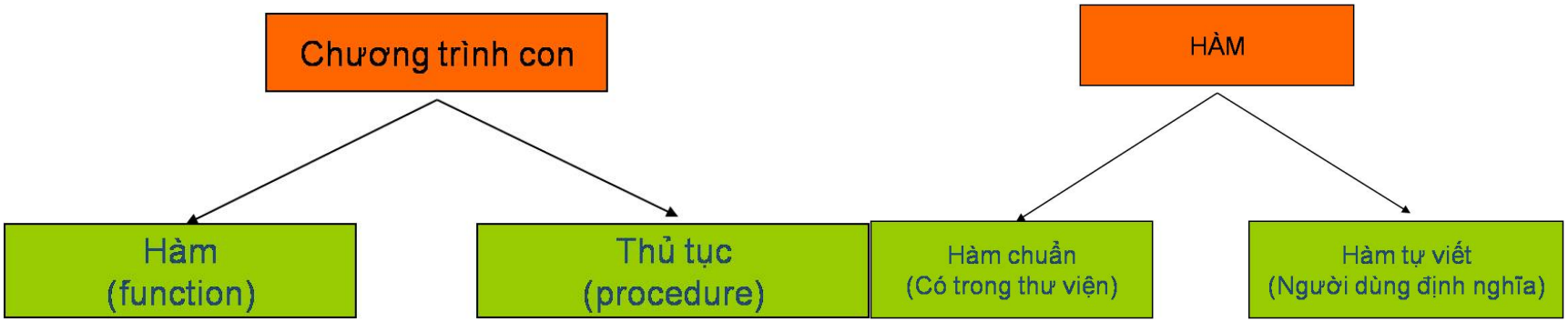
2010 - 2011

Nội dung

Khái niệm chương trình con

- Khái niệm
  - Là một chương trình nằm trong một chương trình lớn hơn nhằm thực hiện một nhiệm vụ cụ thể
- Vai trò:
  - Chia nhỏ chương trình ra thành từng phần để **dễ quản lý** => Phương pháp lập trình có cấu trúc
  - Có thể **sử dụng lại** nhiều lần: printf, scanf...
  - Chương trình **dễ dàng đọc** và bảo trì hơn

Phân loại chương trình con



- Hàm: trả về giá trị
- Thủ tục: không trả về giá trị
- Trong C:
  - Chỉ cho phép khai báo chương trình con là hàm.
  - Khi chương trình con không có giá trị trả về: sử dụng kiểu "*void*" với ý nghĩa "không là kiểu dữ liệu nào cả"

Khai báo hàm

✦ Cú pháp:

```
[kieu_gia_tri_tra_ve] ten_ham ([danh_sach_tham_so])
{
        [Cac_khai_bao]
        [Cac_cau_lenh]
 }
```

- Dòng đầu hàm:
  - là thông tin để trao đổi giữa các hàm. Phân biệt giữa các hàm với nhau
  - dùng để phân biệt các hàm => không có 2 hàm có dòng đầu hàm giống nhau
- Thân hàm:
  - chứa các khai báo và các câu lệnh
  - tồn tại ít nhất 1 lệnh **return**

Khai báo hàm

- Dòng đầu hàm:
  [kieu_gia_tri_tra_ve] ten_ham ([danh_sach_tham_so])
  - *ten_ham*: tên hàm là định danh hợp lệ, trong C tên hàm là duy nhất (không trùng nhau)
  - *kieu_gia_tri_tra_ve*: kiểu giá trị trả về
    - kiểu dữ liệu bất kì, không được là kiểu dữ liệu mảng
    - nếu không có kiểu giá trị trả về, trình biên dịch C ngầm hiểu kiểu DL trả về là **int**
  - *danh_sach_tham_so*:
    - các tham số (cách nhau bởi dấu phẩy) chứa DL vào cung cấp cho hàm
    - **tham số hình thức**: tham số trong lời khai báo hàm: phải có kiểu DL và tên tham số
    - **tham số thực**: các tham số cung cấp cho hàm khi thực hiện

Khai báo hàm

```
#include<stdio.h>
#include<conio.h>

// Khai bao va dinh nghia ham
    binhphuong
int binhphuong(int x)
{
    int y;
    y =  x * x;
    return y;
}

// Ham main
void main()
{
    // Noi dung ham main o day
    ...
}
```

```
#include<stdio.h>
#include<conio.h>

// Khai bao ham nguyen mau
int binhphuong(int);

// Ham main
void main()
{
    // Noi dung ham main o day
    ...
}

// Dinh nghia ham binhphuong
int binhphuong(int x)
{
    int y;
    y =  x * x;
    return y;
}
```

Khai báo hàm

- Ý nghĩa của nguyên mẫu hàm:
  - Cho phép **định nghĩa sau** khi sử dụng, nhưng phải khai báo trước
  - Cho phép đưa ra lời gọi đến một hàm mà **không cần biết định nghĩa**
  - Ví dụ:
    - khi gọi **printf, scanf** chúng ta chỉ cần quan tâm các tham số truyền cho hàm
    - tệp **stdio.h** chứa **nguyên mẫu hàm** của **printf** và **scanf**

Sử dụng hàm

- Lời gọi hàm: ten_ham([tham_so1, tham_so2, ...]);
- Thực hiện:
  - Nếu hàm có tham số => các tham số được gán giá trị thực tương ứng
  - Thực hiện các lệnh trong thân hàm
  - Hàm sẽ kết thúc và trở về chương trình gọi nó nếu:
    - thực hiện hết các lệnh trong hàm HOẶC
    - gặp lệnh **return** : **return [bieu_thuc];**
- Lưu ý chung:
  - dù không có tham số, sau tên hàm luôn có cặp dấu **( )**
  - một số ngôn ngữ cho phép khai báo các chương trình con **lồng nhau** (Pascal) nhưng **C thì không**

Sử dụng hàm

```
#include<stdio.h>
#include<conio.h>

// Khai bao ham nguyen mau
int binhphuong(int);

// Ham main
void main()
{
    int i;
    for (i=0; i<= 10; i++)
     printf("%4d", binhphuong(i));
    getch();
}

// Dinh nghia ham binhphuong
int binhphuong(int x)
{
    int y;
    y =  x * x;
    return y;
}
```

QA

- Trong chương trình lớn có nhiều chương trình con, điểm bắt đầu thực hiện chương trình sẽ thuộc chương trình con nào?
- **main** là một chương trình con?
- Khai báo các chương trình con độc lập nhau/lồng lẫn nhau?
- Muốn "lắp ráp" các công việc khác nhau để cùng thực hiện, cần phải đưa ra "lời gọi" hàm. "Lời gọi" cần cung cấp những gì?

Phạm vi của các biến

- Phạm vi:
  - khối lệnh, chương trình con, chương trình chính
- Biến khai báo trong phạm vi nào thì sử dụng trong phạm vi đó
- Trong *cùng một phạm vi* các biến có *tên khác nhau*
- Tình huống: Trong hai phạm vi khác nhau có hai biến cùng tên, phạm vi này nằm trong phạm vi kia?
  $\Longrightarrow$ chương trình đang thực thi trong phạm vi nào thì khai báo trong phạm vi đó có tác dụng

Phân loại biến

- Biến cục bộ:
  - biến được khai báo trong lệnh khối hoặc chương trình con, được đặt trước các câu lệnh
- Biến toàn cục:
  - biến được khai báo ngoài mọi hàm, được sử dụng ở các hàm đứng sau nó
  - Vị trí khai báo: sau phần khai báo tệp tiêu đề và khai báo hàm nguyên mẫu
- Ghi nhớ:
  - Hàm **main()** cũng là một chương trình con nhưng là nơi chương trình được bắt đầu cũng như kết thúc
  - Biến khai báo trong hàm **main()** cũng là biến cục bộ, chỉ có phạm vi trong hàm **main()**.

Một số lệnh đặc trưng trong C: static, register

- Biến **static**:
  - biến cục bộ: bộ nhớ được giải phóng sau khi ra khỏi phạm vi của biến

  ⇒ sử dụng biến lưu trữ lâu dài: từ khóa **static**. Khai báo:
  static kieu_du_lieu ten_bien;
  - Giống/khác với *biến toàn cục*?

```
# 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();
}
```

Một số lệnh đặc trưng trong C: static, register

- Biến **register**:
  - Thanh ghi có tốc độ truy cập nhanh hơn RAM, bộ nhớ ngoài

  ⇒ Lưu biến trong thanh ghi sẽ tăng tốc độ thực hiện chương trình.

  Khai báo:

  register kieu_du_lieu ten_bien;
  - Lưu ý: số lượng biến register **không nhiều** và thường chỉ với kiểu *dữ liệu nhỏ* như int, char

Questions & Answers