

BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC CÔNG NGHỆ TP.HCM

LẬP TRÌNH C

Biên Soạn:

ThS. Nguyễn Thúy Loan

www.hutech.edu.vn

LẬP TRÌNH C

Ấn bản 2020

Các ý kiến đóng góp về tài liệu học tập này, xin gửi về e-mail của ban biên tập:
tailieuhocTap@hutech.edu.vn

HƯỚNG DẪN

MÔ TẢ MÔN HỌC

Môn Lập trình C cung cấp cho sinh viên những kiến thức cơ bản về máy tính, về lập trình thông qua ngôn ngữ lập trình C. Môn học này là nền tảng để tiếp thu hầu hết các môn học khác trong chương trình đào tạo. Mặt khác, nắm vững môn này là cơ sở để phát triển tư duy và kỹ năng lập trình để giải các bài toán và các ứng dụng trong thực tế.

Học xong môn này, sinh viên phải nắm được các vấn đề sau:

- Khái niệm về máy tính, hệ đếm, cách chuyển đổi giữa các hệ đếm
- Khái niệm về ngôn ngữ lập trình.
- Ngôn ngữ sơ đồ (lưu đồ), sử dụng lưu đồ để biểu diễn các giải thuật.
- Tổng quan về Ngôn ngữ lập trình C.
- Các kiểu dữ liệu trong C.
- Các lệnh có cấu trúc.
- Cách thiết kế và sử dụng các hàm trong C.
- Xử lý các bài toán trên mảng một chiều.
- Biết xây dựng và xử lý các bài toán trên dữ liệu có cấu trúc do người dùng định nghĩa.

NỘI DUNG MÔN HỌC

- Bài 1 CÁC KIẾN THỨC CƠ BẢN VỀ MÁY TÍNH
- Bài 2 CÁC KIẾN THỨC CƠ BẢN VỀ MÁY TÍNH
- Bài 3 GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH C
- Bài 4 KIỂU DỮ LIỆU VÀ BIỂU THỨC TRONG C
- Bài 5 CÁC TOÁN TỬ ĐIỀU KHIỂN
- Bài 6 CHƯƠNG TRÌNH CON (HÀM)
- Bài 7 MẢNG – CHUỖI KÝ TỰ

- Bài 8 KIỂU DỮ LIỆU CÓ CẤU TRÚC

YÊU CẦU MÔN HỌC

Có tư duy tốt về logic và kiến thức toán học cơ bản.

CÁCH TIẾP NHẬN NỘI DUNG MÔN HỌC

Lập trình C là môn học đầu tiên giúp sinh viên làm quen với khái niệm máy tính, phương pháp lập trình trên máy tính, giúp sinh viên có khái niệm cơ bản về cách tiếp cận và giải quyết các bài toán tin học, giúp sinh viên có khả năng tiếp cận với cách tư duy của người lập trình, và là tiền đề để tiếp cận với các học phần quan trọng còn lại của ngành Công nghệ Thông tin. Vì vậy, yêu cầu người học phải dự học đầy đủ các buổi lên lớp, làm bài tập đầy đủ ở nhà, nghiên cứu tài liệu trước khi đến lớp và gạch chân những vấn đề không hiểu khi đọc tài liệu để đến lớp trao đổi.

PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC

Để học tốt môn này, người học cần ôn tập các bài đã học, trả lời các câu hỏi và làm đầy đủ bài tập; đọc trước bài mới và tìm thêm các thông tin liên quan đến bài học.

Đối với mỗi bài học, người học đọc trước mục tiêu bài học, sau đó đọc nội dung bài học. Kết thúc mỗi ý của bài học, người đọc trả lời câu hỏi ôn tập và kết thúc toàn bộ bài học, người đọc làm các bài tập.

Điểm đánh giá: gồm 2 phần

1. Điểm quá trình (chuyên cần + bài tập)
2. Điểm cuối kỳ (bài thi trên giấy - tự luận).

BÀI 1: CÁC KIẾN THỨC CƠ BẢN VỀ MÁY TÍNH

Sau khi học xong bài này, sinh viên có thể nắm được về cơ bản:

- *Khái niệm cơ bản về ngành CNTT*
- *Khái niệm cơ bản về hệ thống máy tính*
- *Cách biểu diễn thông tin trong máy tính*
- *Khái niệm hệ thống mạng cơ bản*

1.1 MỘT SỐ KHÁI NIỆM

1.1.1 Máy tính là gì?

Trong thời đại tin học ngày nay, máy tính được dùng rộng rãi trong nhiều lĩnh vực nghề nghiệp như giáo dục, thông tin, giải trí, ngân hàng, kinh doanh, y tế, dự báo thời tiết, và nghiên cứu khoa học. Tại nhà, chúng ta dùng máy tính để liên lạc với người khác, giải trí, thực hiện việc tìm kiếm, viết và soạn bài tập, tạo ảnh, theo dõi tài chính cá nhân và rất nhiều việc khác. Máy tính (xem Hình 1.1) có thể được mô tả là một thiết bị điện tử thực hiện các thao tác toán học, logic học và đồ họa. Để thực hiện các thao tác này và các nhiệm vụ của người sử dụng, máy tính cần được trang bị một hệ điều hành và các chương trình phần mềm.



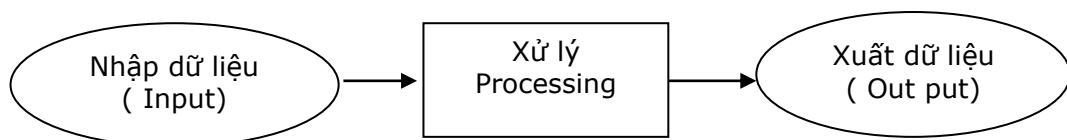
Hình 1.1: Máy tính

1.1.2 Khái niệm về thông tin

- **Dữ liệu** có thể là các kí tự, văn bản, chữ, số, hình ảnh, âm thanh, hoặc video chưa được tổ chức, xử lý và chưa có ý nghĩa.
- **Thông tin** là dữ liệu đã được xử lý, tổ chức, cấu trúc hoặc trình bày trong một bối cảnh cụ thể để làm cho nó hữu ích, có ý nghĩa.

Ví dụ: Trường HUTECH, Trường ĐHKHTN, UBND TP.HCM, bản tin tài chính....

- **Hệ thống thông tin** (information system) là một hệ thống ghi nhận dữ liệu, xử lý chúng để tạo nên thông tin có ý nghĩa hoặc dữ liệu mới (hình 1.2)



Hình 1.2: Hệ thống thông tin

1.1.3 Đơn vị đo thông tin

Đơn vị cơ sở dùng để đo thông tin được gọi là **BIT (BInary digiT)**. Một BIT là một chỉ thị hoặc một thông báo về sự kiện nào đó có 1 trong 2 trạng thái là: Tắt (Off) / Mở (On) hay Đúng (True) / Sai (False).

Số học nhị phân sử dụng hai ký số 0 và 1 để biểu diễn các số, nên số học nhị phân được dùng để biểu diễn trạng thái của 1 BIT.

Trong tin học, người ta thường sử dụng các đơn vị đo thông tin lớn hơn như sau:

Tên gọi	Ký hiệu	Giá trị	
Byte	B	8 bit	
KiloByte	KB	2^{10}	$B = 1024$ Byte
MegaByte	MB	2^{20}	KB
GigaByte	GB	2^{30}	MB
TetraByte	TB	2^{40}	GB
Petabyte	PB	2^{50}	TB
Exabyte	EB	2^{60}	PB
Zettabyte	ZB	2^{70}	EB
Yottabyte	YB	2^{80}	ZB
Brontobyte	BB	2^{90}	YB
Geopbyte	GeB	2^{100}	BB

1.1.4 Xử lý thông tin bằng máy tính điện tử

Bốn chức năng cơ bản của máy tính cũng được biết đến như là chu trình xử lý thông tin:

- **Nhập dữ liệu:** máy tính tập hợp dữ liệu hoặc cho phép người dùng nhập dữ liệu.
- **Xử lý:** dữ liệu được chuyển thành thông tin.
- **Xuất dữ liệu:** Kết quả xử lý được xuất ra từ máy tính.
- **Lưu trữ:** dữ liệu hoặc thông tin được lưu trữ để sử dụng trong tương lai.

1.2 BIỂU DIỄN THÔNG TIN TRONG MÁY TÍNH ĐIỆN TỬ

1.2.1 Biểu diễn số trong các hệ đếm

- Hệ đếm là tập hợp các ký hiệu và qui tắc, sử dụng tập ký hiệu quy tắc đó để biểu diễn và xác định giá trị của các số.
- Mỗi hệ đếm có một số ký số (digits) hữu hạn. Tổng số ký số của mỗi hệ đếm được gọi là **cơ số** (base hay radix), ký hiệu là b .
- **Hệ đếm cơ số b** ($b \geq 2$, b là số nguyên dương) mang tính chất sau :
 - Có b ký số để thể hiện giá trị số. Ký số nhỏ nhất là **0** và lớn nhất là **$b-1$** .
 - Giá trị vị trí thứ n trong một số của hệ đếm bằng cơ số b lũy thừa n : b^n

- Trong ngành toán - tin học hiện nay phổ biến 4 hệ đếm là hệ thập phân, hệ nhị phân, hệ bát phân và hệ thập lục phân.

1.2.1.1 Hệ đếm thập phân

Hệ đếm thập phân hay hệ đếm cơ số 10 ($b=10$) là một trong các phát minh của người Ả rập cổ, bao gồm 10 ký số theo ký hiệu sau: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

1.2.1.2 Hệ đếm nhị phân

Hệ đếm nhị phân hay hệ đếm cơ số 2 ($b=2$). Đây là hệ đếm đơn giản nhất với 2 chữ số là 0 và 1. Mỗi chữ số nhị phân gọi là BIT (BInary digiT). Để diễn tả một số lớn hơn thì cần kết hợp nhiều bit với nhau.

1.2.1.3 Hệ đếm bát phân

Hệ bát phân hay hệ đếm cơ số 8 ($b=8$). Hệ đếm này có 8 chữ số: 0, 1, 2, 3, 4, 5, 6, 7.

1.2.1.4 Hệ đếm thập lục phân

Hệ đếm thập lục phân hay hệ đếm cơ số 16 ($b=16$). Khi thể hiện ở dạng hexa-decimal, ta có 16 ký tự: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F để biểu diễn các giá trị số. Các chữ in A, B, C, D, E, F tương ứng với các giá trị số là 10, 11, 12, 13, 14, 15 trong hệ đếm thập phân.

Bảng 1.1, qui đổi tương đương 16 chữ số đầu tiên của 4 hệ đếm

Hệ 10	Hệ 2	Hệ 8	Hệ 16
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

1.2.2 Cách chuyển đổi cơ số

1.2.2.1 Đổi từ cơ số d sang cơ số 10

Để chuyển đổi một số ở hệ đếm cơ số d sang hệ đếm cơ số 10 người ta khai triển con số trong cơ số d dưới dạng đa thức theo cơ số của nó.

Ví dụ 1.1 Đổi số $1101_{(2)}$ ở hệ nhị phân sang hệ thập phân như sau: $1101_{(2)} = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13_{(10)}$

1.2.2.2 Đổi từ cơ số 10 sang cơ số d

Để chuyển đổi một số từ cơ số 10 sang cơ số d ($d = 2, 8, 16$) người ta lấy con số trong cơ số 10 chia liên tiếp cho d đến khi thương số bằng không thì dừng lại. Kết quả chuyển đổi có được trong hệ đếm cơ số d là tập hợp các số dư của phép chia được viết theo thứ tự ngược lại, nghĩa là số dư đầu tiên có trọng số nhỏ nhất.

Ví dụ : $6_{(10)} = 110_{(2)}$

1.3 HỆ THỐNG MÁY TÍNH

1.3.1 Máy vi tính và thiết bị cầm tay thông minh

Mặc dù các máy tính có kích cỡ và hình dạng khác nhau, nhưng các thành phần cơ bản cần thiết để hoàn thành chu trình xử lý thông tin phải có mặt trong đó. Ngoài máy vi tính (microcomputers) như máy tính để bàn và máy tính xách tay và các thiết bị di động mà chúng ta quen thuộc, cũng có máy tính đặc biệt bao gồm máy chủ (server), máy tính lớn (mainframes), siêu máy tính (supercomputers), và máy tính nhúng (embedded computers).

Máy vi tính (microcomputers) được phân loại như các máy tính nhỏ, rẻ tiền được thiết kế để sử dụng cá nhân và là các máy tính mà hầu hết mọi người thường sử dụng. Các máy vi tính trong phạm vi thể loại microcomputer có kích thước từ hệ thống máy tính để bàn lớn tới các thiết bị cầm tay phù hợp trong túi. Một số loại phổ biến nhất của máy vi tính bao gồm máy tính để bàn (Desktop computers), máy tính xách tay (Notebook computers), máy tính bảng (Tablet computers), các thiết bị di động (Mobile devices). Hình 1.3, mô tả các thiết bị này.



Hình 1.3: Các loại máy vi tính (microcomputers)

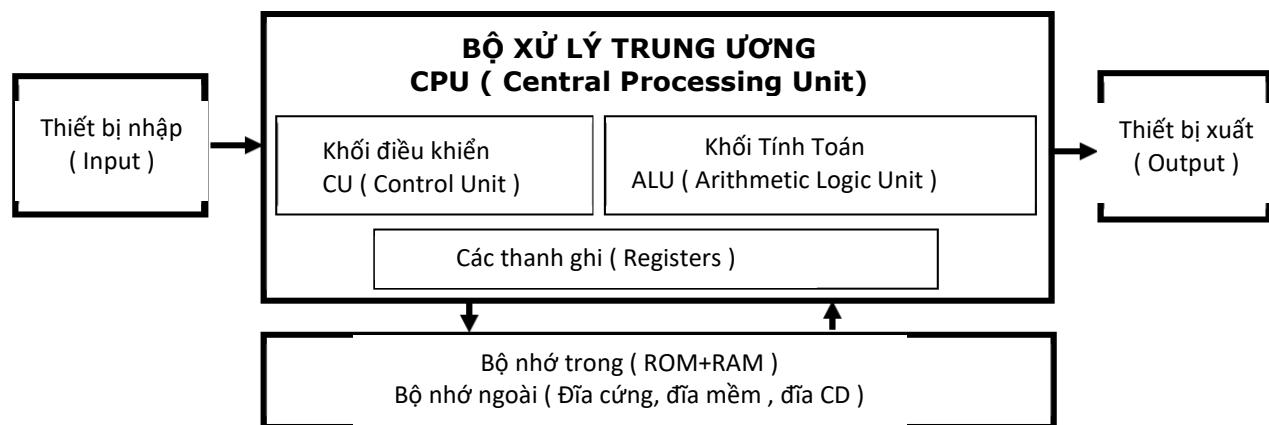
- Mỗi loại máy tính có thể có hình dạng hoặc cấu trúc khác nhau, tùy theo mục đích sử dụng, nhưng một cách tổng quát, máy tính điện tử là một hệ xử lý thông tin tự động gồm 2 phần chính: **phần cứng** và **phần mềm**.

1.3.2 Phần cứng

1.3.2.1 Phần cứng máy tính

Phần cứng máy tính là tập hợp tất cả những phần vật lý mà chúng ta có thể chạm đến. Phần cứng bao gồm 3 phần chính (hình 1.4):

- Bộ nhớ (Memory).
- Đơn vị xử lý trung ương (CPU - Central Processing Unit).
- Thiết bị nhập xuất (Input/ Output).



Hình 1.4: Cấu trúc phần cứng máy tính

1.3.2.2 Khối hệ thống máy tính (Computer System Unit)

Hay còn được gọi là thùng máy tính (computer case), là một cái thùng chứa đựng các thành phần chính của máy tính như Mainboard, CPU, RAM, đĩa cứng và các thành phần khác. Khi thùng máy tính được kết nối với các thiết bị ngoại vi (bàn phím, màn hình, chuột, máy in) tương thích, hệ thống máy tính có thể thực hiện bốn chức năng máy tính

cơ bản: nhập (input), xử lý (process), xuất (output) và lưu trữ (storage). Khi ta mở thùng máy tính, chúng ta sẽ thấy một số thành phần chính bên trong. Một trong những thành phần quan trọng nhất là chip vi xử lý, còn được gọi là bộ xử lý trung tâm (CPU). CPU nằm trên bo mạch chủ (motherboard), một bảng mạch in lớn mà tất cả các bo mạch khác trong máy tính được kết nối (hình 1.5)



Computer case



Mainboard

Hình 1.5: Computer case và Mainboard

1.3.2.2.1 Bộ xử lý trung ương (CPU)

Bộ xử lý trung ương chỉ huy các hoạt động của máy tính theo lệnh và thực hiện các phép tính. CPU có 3 bộ phận chính: khối điều khiển, khối tính toán số học và logic, và một số thanh ghi.

1.3.2.2.2 Khối điều khiển (CU: Control Unit): là trung tâm điều hành máy tính. Nó có nhiệm vụ giải mã các lệnh, tạo ra các tín hiệu điều khiển công việc của các bộ phận khác của máy tính theo yêu cầu của người sử dụng hoặc theo chương trình đã cài đặt.

1.3.2.2.3 Khối tính toán số học và logic (ALU: Arithmetic-Logic Unit): bao gồm các thiết bị thực hiện các phép tính số học (cộng, trừ, nhân, chia, ...), các phép tính logic (AND, OR, NOT, XOR) và các phép tính quan hệ (so sánh lớn hơn, nhỏ hơn, bằng nhau, ...)

1.3.2.2.4 Các thanh ghi (Registers): Được gắn chặt vào CPU bằng các mạch điện tử làm nhiệm vụ bộ nhớ trung gian. Các thanh ghi mang các chức năng chuyên dụng giúp tăng tốc độ trao đổi thông tin trong máy tính. Ngoài ra, CPU còn được gắn với một đồng hồ (clock) hay còn gọi là bộ tạo xung nhịp. Tần số đồng hồ càng cao thì tốc độ xử lý

thông tin càng nhanh. Thường thì đồng hồ được gắn tương xứng với cấu hình máy và có các tần số dao động (cho các máy Pentium 4 trở lên) là 2.0 GHz, 2.2 GHz, ... hoặc cao hơn. Bộ vi xử lý lõi kép (dual-core) hoặc đa lõi (multicore) được sản xuất bởi Intel và AMD. Các CPU này có nhiều hơn một bộ xử lý (hai cho một lõi kép, nhiều hơn cho một đa lõi) trên một chip duy nhất. Sử dụng nhiều bộ vi xử lý có nhiều lợi thế hơn một bộ xử lý CPU, trong đó có khả năng cải thiện đa nhiệm và hiệu suất hệ thống, tiêu thụ điện năng thấp hơn, giảm thiểu sử dụng tài nguyên hệ thống.

1.3.2.2.5 Bộ nhớ

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

1.3.2.2.5.1 **Bộ nhớ trong:** gồm ROM và RAM

+ ROM (Read Only Memory) là *Bộ nhớ chỉ đọc* thông tin, dùng để lưu trữ các chương trình hệ thống, chương trình điều khiển việc nhập xuất cơ sở (ROM-BIOS: ROM-Basic Input/ Output System). Thông tin được ghi vào ROM không thể bị thay đổi, không bị mất ngay cả khi không có điện.

+ RAM (Random Access Memory) là *Bộ nhớ truy xuất ngẫu nhiên*, được dùng để lưu trữ dữ kiện và chương trình trong quá trình thao tác và tính toán. RAM có đặc điểm là nội dung thông tin chứa trong nó sẽ mất đi khi mất điện hoặc tắt máy. Dung lượng bộ nhớ RAM cho các máy tính hiện nay thông thường vào khoảng 2GB đến 16GB và có thể hơn nữa.

1.3.2.2.5.2 Bộ nhớ ngoài: là thiết bị lưu trữ thông tin với dung lượng lớn, thông tin không bị mất khi không có điện, dữ liệu lưu trên bộ nhớ ngoài vẫn tồn tại cho tới khi người sử dụng xóa hoặc ghi đè lên. Bộ nhớ ngoài có thể cất giữ và di chuyển độc lập với máy tính. Hiện nay có các loại bộ nhớ ngoài phổ biến như: (hình 1.6)

- **Đĩa cứng (Hard Disk):** phổ biến là đĩa cứng có dung lượng từ 40 GB tới 2 TB và lớn hơn nữa.
- **Đĩa quang (Compact disk):** loại 4.72 inch, là thiết bị phổ biến dùng để lưu trữ các phần mềm mang nhiều thông tin, hình ảnh, âm thanh và thường được sử dụng trong các phương tiện đa truyền thông (multimedia). Có hai loại phổ biến là: đĩa CD (dung lượng khoảng 700 MB) và DVD (dung lượng khoảng 4.7 GB).

- Các loại bộ nhớ ngoài khác: như thẻ nhớ (Memory Stick, Compact Flash Card), USB Flash Drive có dung lượng phổ biến là từ 2GB đến 8GB và lớn hơn nữa.

Một số thuật ngữ với bộ nhớ ngoài:

- Số vòng quay mỗi phút (Revolutions Per Minute - RPM): RPM được sử dụng để giúp xác định thời gian truy cập vào ổ đĩa cứng của máy tính. RPM xác định số vòng quay của một đĩa cứng thực hiện trong mỗi phút. Các đĩa cứng có RPM càng cao, dữ liệu sẽ được truy cập nhanh hơn. Ví dụ, nếu so sánh hai ổ đĩa cứng với 5400 RPM và 7200 RPM, ổ đĩa cứng 7200 RPM sẽ có khả năng truy cập dữ liệu nhanh hơn nhiều so với đĩa cứng 5400 RPM.
- Số bit trên giây (bits per second – bps hoặc bit/sec): Trong giao tiếp dữ liệu. Bits per second là số đo tốc độ truyền dữ liệu giữa các thiết bị trong một hệ thống truyền dữ liệu. Bps là số bit được truyền đi hoặc nhận mỗi giây. Đôi khi đơn vị lớn hơn được sử dụng để biểu thị tốc độ dữ liệu cao. Một kilobit trên giây (viết tắt Kbps hoặc kbps) bằng 1.000 bps. Một megabit trên giây (Mbps) là bằng 1.000.000 bps hoặc 1.000 Kbps.



Hình 1.6: Một số loại bộ nhớ ngoài

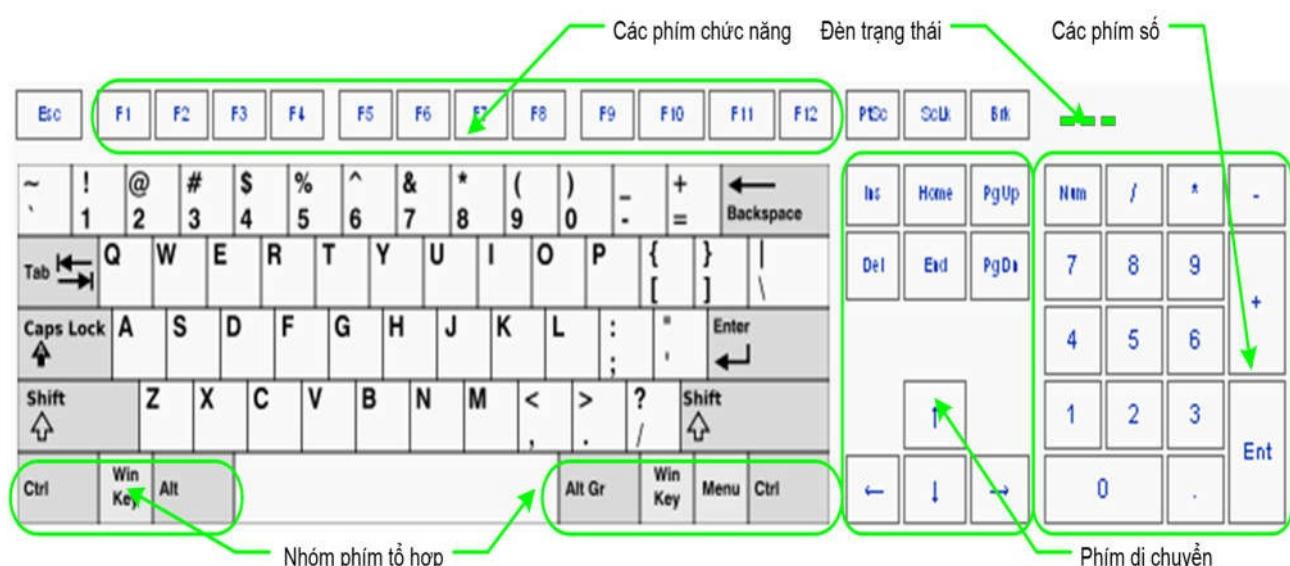
1.3.2.3 Các thiết bị xuất/ nhập

1.3.2.3.1 Các thiết bị nhập

- **Bàn phím** (Keyboard, thiết bị nhập chuẩn): là thiết bị nhập dữ liệu và câu lệnh, bàn phím máy vi tính phổ biến hiện nay là một bảng chứa 104 phím có các tác dụng khác nhau (hình 1.7).

Có thể chia làm 3 nhóm phím chính:

- **Nhóm phím đánh máy:** gồm các phím chữ, phím số và phím các ký tự đặc biệt (~, !, @, #, \$, %, ^, &, ?, ...).
- **Nhóm phím chức năng (function keypad):** gồm các phím từ F1 đến F12 và các phím như (phím di chuyển từng điểm), phím PgUp (lên trang màn hình), PgDn (xuống trang màn hình), Insert (chèn), Delete (xóa), Home (về đầu), End (về cuối)
- **Nhóm phím số (numeric keypad)** như NumLock (cho các ký tự số), CapsLock (tạo các chữ in), ScrollLock (chế độ cuộn màn hình) thể hiện ở các đèn chỉ thị.
- **Nhấn phím tổ hợp:** khi cần sử dụng phím tổ hợp, bạn cần nhấn và giữ phím tổ hợp điều khiển trước (Ctrl hoặc Alt hoặc Shift) sau đó bấm tiếp phím còn lại.
- **Tình trạng đèn báo sáng:** Đôi khi người sử dụng không chú ý và vô tình bật các tính năng hỗ trợ, chẳng hạn chế độ gõ chữ hoa, chế độ gõ số, chế độ khóa thanh cuộn. Các tính năng này, khi bật lên có thể làm cho thao tác của người sử dụng gặp khó khăn. Do đó khi gặp điều lạ khi gõ các phím, hãy nhìn khu vực đèn báo tình trạng bàn phím trước tiên.



Hình 1.7: Mô hình bàn phím

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

Hiện tại có 2 loại chuột thông dụng trên thị trường đối với các máy tính để bàn đó là **chuột dùng bi** và **chuột dùng cảm biến quang**. Các thành phần cơ bản của chuột máy tính như hình 1.8: (1) Nút nhấn trái (left button), (2) Bánh xe cuộn (Scroll wheel) và (3) Nút nhấn phải (right button). Tùy theo mục tiêu sử dụng mà có thể có thêm nhiều loại nút bấm khác được bố trí tại các vị trí khác nhau trên thân chuột.



Hình 1.8: Các loại chuột phổ biến và vị trí các nút chuột

- **Máy quét hình** (Scanner): là thiết bị dùng để nhập văn bản hay hình vẽ, hình chụp vào máy tính. Thông tin nguyên thủy trên giấy sẽ được quét thành các tín hiệu số tạo thành các tập tin ảnh (image file).
- **Cân điều khiển** (Joystick): là một thiết bị đầu vào thường được sử dụng để điều khiển trò chơi video và công nghệ hỗ trợ. Joystick bao gồm một chân đế, một tay đòn (stick) với một hay nhiều nút nhấn có thể được di chuyển bất kỳ hướng nào.
- **Webcam**: viết tắt của 'web camera', là một máy quay phim kỹ thuật số được kết nối với một máy tính. Nó có thể gửi hình ảnh trực tiếp từ bất cứ nơi nào nó được bố trí tới vị trí khác bằng phương thức Internet. Nhiều màn hình máy tính để bàn và máy tính xách tay có gắn sẵn Webcam và micro, tuy nhiên, chúng ta có thể gắn thêm một webcam riêng. Có nhiều loại Webcam khác nhau. Một số được cắm vào máy tính thông qua cổng USB, một số khác là không dây (wireless). (hình 1.9)



Máy quét (Scanner)



Cần điều khiển (Joystick)



Webcam

Hình 1.9: Các thiết bị nhập ngoại vi

1.3.2.3.2 Các thiết bị xuất

Thiết bị xuất cơ bản gồm các thiết bị sau: (hình 1.10)

- **Màn hình** (Screen hay Monitor, thiết bị xuất chuẩn): dùng để thể hiện thông tin cho người sử dụng xem. Thông tin được thể hiện ra màn hình bằng phương pháp ánh xạ bộ nhớ (memory mapping), với cách này màn hình chỉ việc đọc liên tục bộ nhớ và hiển thị (display) bất kỳ thông tin nào hiện có trong vùng nhớ ra màn hình. Màn hình phổ biến hiện nay trên thị trường là màn hình màu SVGA 15", 17", 19" với độ phân giải có thể đạt 1280 X 1024 pixel.
- **Máy in** (Printer): là thiết bị xuất để đưa thông tin ra giấy. Máy in phổ biến hiện nay là loại máy in ma trận điểm (dot matrix) loại 24 kim, máy in phun mực, máy in laser trắng đen hoặc màu. Tốc độ của một máy in được đo bởi các đơn vị sau: cps (ký tự trên mỗi giây), LPS (dòng trên mỗi giây) hoặc ppm (số trang mỗi phút)
- **Máy chiếu** (Projector): chức năng tương tự màn hình, thường được sử dụng thay cho màn hình trong các buổi Seminar, báo cáo, thuyết trình, ...



Màn
hình



Máy in (Printer)

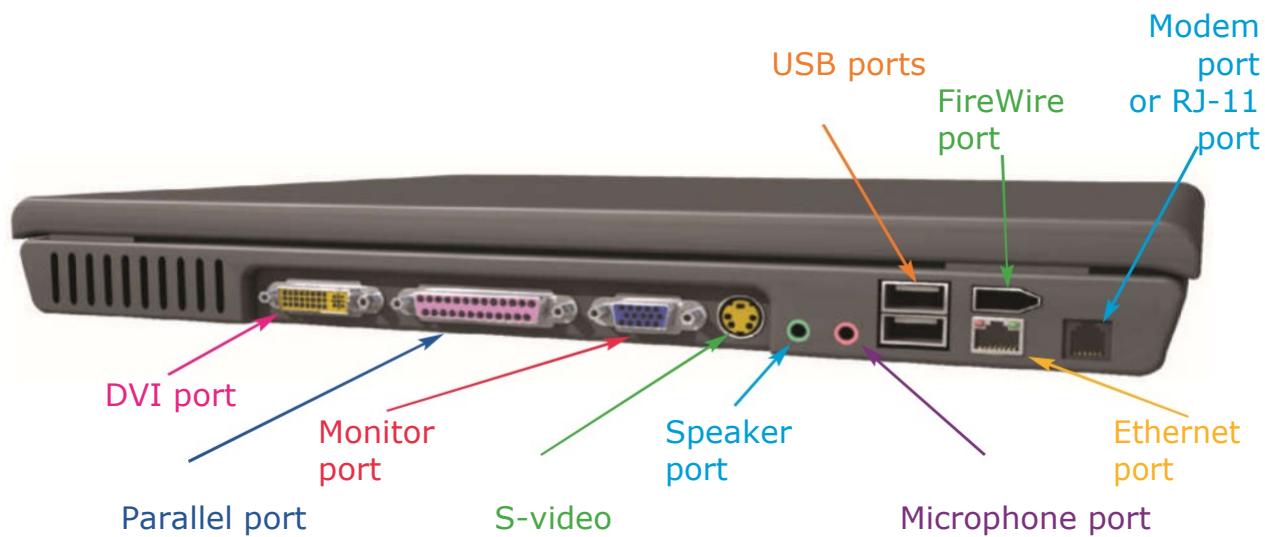


Máy
chiếu

Các thiết bị xuất ngoại vi

1.3.2.3.3 Cổng (Port)

Một cổng hoạt động như một mạch ghép nối (interface) giữa các thiết bị ngoại vi của hệ thống và máy tính, cho phép trao đổi dữ liệu khi chúng được kết nối. Như bạn có thể thấy trên mặt sau của máy tính xách tay hiện trong hình 1.11, cổng có thể có hình dạng và kích cỡ khác nhau.

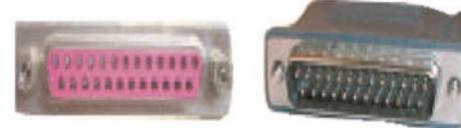


Hình 1.11: Các cổng thông thường trên máy tính xách tay

- **Cổng nối tiếp (serial port) và cổng song song (parallel port):** là hai trong số các loại cổng xưa nhất (hình 1.12) được tìm thấy trên một máy vi tính.
 - Cổng nối tiếp là cổng mà nó có thể gởi dữ liệu chỉ một bit tại một thời điểm, vì thế tốc độ trao đổi dữ liệu chậm so với các công nghệ mới hơn. Tốc độ tối đa mà một cổng nối tiếp chuẩn có thể truyền dữ liệu là 15 Kbps. Thiết bị chuột và modem là những ví dụ của thiết bị có thể sử dụng cổng nối tiếp.
 - Cổng song song là loại cổng có thể gởi dữ liệu trong nhóm các bit, tốc độ lên tới 500 Kbps, vì thế phương thức truyền dữ liệu **nhanh** hơn cổng nối tiếp. Các máy in cũ thường được nối với máy tính qua cổng song song.



Cổng và đầu nối nối tiếp



Cổng và đầu nối song song

Cổng USB (universal serial bus): cổng này có thể giao tiếp với nhiều thiết bị ngoại vi khác nhau. Cổng USB có thể truyền dữ liệu ở tốc độ cao. Cổng USB ban đầu được biết như USB 1.1, có tốc độ có thể 12 Mbps. Các phiên bản mới, USB 2.0 (hay còn gọi là Hi-Speed USB), có thể đạt tốc độ 480 Mbps – gấp 40 lần tốc độ USB 1.1

và hơn 400 lần một cổng nối tiếp. Thiết bị sử dụng cổng USB bao gồm bàn phím, chuột, máy in, máy nghe nhạc MP3 và PDA, ..., USB 3.0 (còn được gọi là SuperSpeed USB) có tốc độ băng thông tối đa là 5 Gbps (gigabits mỗi giây). Tức có thể đạt tốc độ truyền dữ liệu là 640 MBps (MB mỗi giây), nhanh hơn gấp 10 lần so với USB 2.0. (hình 1.13)



Hình 1.13 Cổng và đầu nối USB

Cổng FireWire (FireWire port): (hình 1.14) được phát triển bởi Apple và còn được gọi là IEEE 1394, là một phương tiện truyền tải dữ liệu nhanh chóng. FireWire 400 có tốc độ truyền dữ liệu 400 Mbps, trong khi FireWire 800 mới hơn truyền dữ liệu ở tốc 800 Mbps! Cổng này thường được sử dụng để kết nối các thiết bị cần chuyển một lượng lớn dữ liệu đến một máy tính một cách nhanh chóng, chẳng hạn như máy ảnh kỹ thuật số hoặc máy ghi video kỹ thuật số hoặc ổ đĩa cứng gắn ngoài. Cổng FireWire là tiêu chuẩn trên nhiều sản phẩm của Apple, nhưng thường chỉ được tìm thấy trên Windows PC và các thiết bị ngoại vi cao cấp. (hình 1.14)



Hình 1.14 Cổng và đầu nối FireWire

Các cổng nối kết mạng: (hình 1.15) chẳng hạn như cổng Ethernet và modem, được sử dụng để kết nối máy tính vào mạng nội bộ hoặc Internet. Một cổng Ethernet, cũng được biết đến như một jack RJ-45, tương tự như một jack cắm điện thoại chuẩn, nhưng lớn hơn một chút. Các cổng Ethernet được sử dụng để truy cập mạng và cũng có thể được sử dụng để kết nối với một modem hoặc router để truy cập Internet. Một cổng modem là cùng kích thước và hình dạng như một jack cắm điện thoại và được sử dụng để kết nối modem với một hệ thống điện thoại, cho phép quay số truy cập Internet. Tốc độ truyền dữ liệu tối đa đối với một modem 56 Kbps, trong khi tiêu chuẩn phổ biến nhất Ethernet, Fast Ethernet, truyền dữ liệu với tốc độ 100 Mbps. Tuy nhiên, Gigabit Ethernet, khả năng tốc độ truyền là 1.000 Mbps



1.3.3 Phân mềm

1.3.3.1 Khái niệm phần mềm

Phần cứng máy tính bao gồm các thành phần vật lý của hệ thống. Tuy nhiên, chưa có phần mềm, máy tính sẽ chỉ là một tập hợp các bộ phận cơ khí. Phần mềm cung cấp các hướng dẫn cho máy tính phải làm gì. Để thực hiện các nhiệm vụ khác nhau, máy tính đòi hỏi một bộ các hướng dẫn, gọi là chương trình. Các chương trình này cho phép người dùng sử dụng máy tính mà không cần kỹ năng lập trình đặc biệt. Chúng ta không thể thấy hoặc sờ được phần mềm, mặc dù ta có thể hiển thị được chương trình trên màn hình. Có hai loại phần mềm máy tính: phần mềm hệ thống và phần mềm ứng dụng

1.3.3.2 Phần mềm hệ thống (Operating System Software)

Phần mềm hệ thống cung cấp các hướng dẫn mà máy tính cần phải thực hiện. Nó chứa các hướng dẫn cần thiết để khởi động máy tính (được biết đến như quá trình khởi động), kiểm tra để đảm bảo mọi thứ đều trong làm việc tốt, và cho phép bạn giao tiếp máy tính và thiết bị ngoại vi của nó. Phần mềm hệ thống bao gồm hai loại chương trình chính: hệ điều hành (operating system) và các chương trình tiện ích (utility programs)

1.3.3.2.1 Hệ điều hành (operating system)

Hệ điều hành (OS) là một chương trình máy tính đặc biệt, nó có mặt trên tất cả các máy tính để bàn hoặc máy tính xách tay, từ máy tính lớn đến các thiết bị cầm tay thông minh. Các hệ điều hành điều khiển cách những máy tính làm việc từ khi nó được bật cho đến khi nó được tắt. Các **hệ** điều hành hệ thống quản lý các thành phần phần cứng khác nhau, bao gồm cả CPU, bộ nhớ, thiết bị lưu trữ, và các thiết bị ngoại vi. Nó còn phối hợp với các ứng dụng phần mềm khác để các phần mềm này có thể được chạy. Phần mềm hệ thống phổ biến hiện nay ở Việt Nam là 3 hệ điều hành chính: Microsoft Windows, Mac OS, and Linux.

- **Phần mềm Microsoft Windows:** có thị phần lớn nhất trong ba hệ điều hành chính và được tìm thấy trên hầu hết các máy tính để bàn và máy tính xách tay hiện nay. Đã có

rất nhiều phiên bản của Microsoft Windows, bao gồm Windows 3.0, Windows 95, Windows 98, Windows ME và Windows Vista, Windows 7, Windows 8, Windows 10.

- **Phần mềm Mac OS:** Mac OS là một hệ điều hành được thiết kế đặc biệt cho máy tính Macintosh của Apple. Hệ điều hành Mac xuất hiện tương tự như Windows, vì nó cũng sử dụng một giao diện đồ họa. Trong thực tế, Apple là công ty đầu tiên giới thiệu hệ điều hành giao diện đồ họa mang tính thương mại. Nhưng, vì sự phổ biến áp đảo của các máy tính dựa trên Windows, Mac OS có một thị phần nhỏ hơn nhiều. Một số phiên bản Mac OS như Mac OS X Tiger, MacOS Sierra.
- **Phần mềm Linux:** Linux là một hệ điều hành thay thế dựa trên điều hành UNIX, được phát triển cho các máy tính lớn. Linux là một hệ điều hành mã nguồn mở, có nghĩa là nó không phải là thuộc sở hữu của một công ty và một số phiên bản có sẵn miễn phí. Một số hệ điều hành Linux phổ biến như: Ubuntu Linux, Linux Mint, Arch Linux, Deepin, Fedora, Debian, openSUSE.

1.3.3.2 Chương trình tiện ích (utility programs)

Phần mềm hệ điều hành là phần mềm quan trọng nhất trên máy tính, vì máy tính không thể hoạt động được khi không có phần mềm hệ điều hành. Tuy nhiên, các chương trình tiện ích là một thành phần quan trọng của phần mềm hệ thống. Chương trình tiện ích là ứng dụng nhỏ xử lý nhiều nhiệm vụ quan trọng liên quan đến việc quản lý và bảo trì hệ thống của bạn. Chương trình tiện ích có thể được sử dụng để sao lưu các tập tin quan trọng, loại bỏ các tập tin hoặc các chương trình không mong muốn từ hệ thống của bạn, và lịch các tác vụ khác nhau để giữ cho hệ thống của bạn chạy mượt. Một số các tiện ích được bao gồm trong các hệ điều hành, hoặc một số tiện ích có phiên bản độc lập mà bạn có thể mua hoặc tải về miễn phí. Ví dụ, trên môi trường Windows. Một số tiện ích quản lý file như Win Zip để nén tập tin, tiện ích Aladdin Easy Uninstall 2.0 gỡ bỏ các ứng dụng không mong muốn. Các tiện ích chuẩn đoán và bảo trì hệ thống như Task Manager của Windows, Norton Ghost, Norton SystemWorks, ...

1.3.3.3 Phần mềm ứng dụng (Application Software)

Phần mềm ứng dụng rất phong phú và đa dạng, bao gồm những chương trình được viết ra cho một hay nhiều mục đích ứng dụng cụ thể như phần mềm soạn thảo văn bản, phần mềm tài chính, phân tích số liệu, tổ chức hệ thống, bảo mật thông tin, đồ họa, chơi games, ...,

1.3.3.3.1 Phần mềm xử lý văn bản

Phần mềm xử lý văn bản được sử dụng để tạo, chỉnh sửa, định dạng, và lưu các tài liệu các định dạng tập tin văn bản khác nhau. Phần mềm xử lý văn bản Microsoft Word cho phép bạn tạo hoặc chỉnh sửa văn bản, các báo cáo, một cách nhanh chóng và dễ dàng. Tài liệu được tạo với phần mềm này cũng có thể bao gồm đồ họa, biểu đồ, và các yếu tố đồ họa khác. Microsoft Word, Lotus Pro, và Corel WordPerfect là những ví dụ của các chương trình xử lý văn bản. Một số phần mềm xử lý văn bản mã nguồn mở như LibreOffice Writer, OpenOffice Writer.

1.3.3.2 Phần mềm bảng tính

Phần mềm bảng tính cho phép bạn thực hiện các tính toán và các tác vụ toán học khác. Tương tự như các tài liệu được sử dụng bởi các kế toán, bảng tính chứa dữ liệu nhập vào trong các cột và các hàng và cho phép bạn thực hiện các phép tính, phân tích, tạo biểu đồ và đồ thị. Một tiện ích quan trọng của phần mềm bảng tính là khả năng tính toán lại bảng tính mà không cần người dùng can thiệp. Khi dữ liệu được sử dụng trong một phép tính hoặc sửa công thức được thay đổi, phần mềm bảng tính tự động cập nhật các bảng tính với kết quả đúng. Microsoft Excel, Lotus 1-2-3, và Corel Quattro Pro là những ví dụ của các chương trình bảng tính. Một số phần mềm xử lý văn bản mã nguồn mở như LibreOffice Calc, OpenOffice Calc.

1.3.3.3 Phần mềm cơ sở dữ liệu

Cơ sở dữ liệu được sử dụng để lưu trữ và tổ chức một lượng lớn dữ liệu. Thông thường, phần mềm cơ sở dữ liệu có thể được sử dụng để quản lý các loại thông tin khác nhau như hàng tồn kho, lịch sử đặt hàng, và lập hóa đơn. Cơ sở dữ liệu giúp bạn nhập, lưu trữ, sắp xếp, lọc, lấy, và tóm tắt các thông tin và sau đó tạo ra các báo cáo có ý nghĩa. Chương trình cơ sở dữ liệu phổ biến như Microsoft Access, Lotus Approach, và Corel Paradox.

1.3.3.4 Phần mềm trình chiếu

Phần mềm này được sử dụng để tạo các bài thuyết trình đồ họa, được gọi là slide show, nó có thể được chiếu lớn bằng phương tiện như máy chiếu hoặc hiển thị trên Web. Phần mềm trình chiếu cũng được sử dụng để tạo ra các tài liệu phân phát cho khán giả (handout), những ghi chú cho người thuyết trình và các tài liệu khác có thể được sử dụng trong một bài thuyết trình. Microsoft PowerPoint, Lotus Freelance Graphics, và Corel Presentations là

những ví dụ của các chương trình phần mềm trình chiếu. Một số phần mềm trình chiếu mã nguồn mở như LibreOffice Impress, OpenOffice Impress.

1.3.3.5 Phần mềm thương mại

- Phần mềm thương mại là phần mềm hoặc chương trình được thiết kế và phát triển cho việc cấp phép hoặc bán cho người dùng hoặc phục vụ mục đích thương mại. Phần mềm thương mại đã từng được coi là phần mềm thuộc quyền sở hữu, nhưng hiện nay có một số phần mềm ứng dụng được sử dụng miễn phí .
- Các sản phẩm của Microsoft như các hệ điều hành Windows và MS Office là một trong những ví dụ nổi tiếng nhất của phần mềm thương mại.
- Hầu hết các chương trình phần mềm thương mại yêu cầu người sử dụng đăng ký chương trình để công ty có thể theo dõi người dùng được ủy quyền. Một số chương trình phần mềm thương mại, chẳng hạn như các phiên bản mới hơn của chương trình Microsoft và Adobe, yêu cầu người dùng đăng ký các chương trình để tiếp tục sử dụng sau thời gian dùng thử (trial).

Trong khi hầu hết các chương trình phần mềm thương mại được bán thường chèa trong đĩa CD hoặc DVD và đi kèm trong một hộp vật lý, nhiều phần mềm thương mại ngày nay có sẵn bằng cách tải về từ trang web của công ty. Người dùng trả tiền cho phần mềm trực tiếp trên website và tải nó vào máy tính của mình.

1.3.3.6 Phần mềm mã nguồn mở

Phần mềm nguồn mở là phần mềm với mã nguồn mà ai cũng có thể sử dụng, nghiên cứu và đặc biệt là sửa đổi và nâng cao. Bất cứ ai cũng có thể sử dụng chương trình cho mục đích nào đó; không có lệ phí cấp giấy phép hoặc hạn chế khác về phần mềm. Tuy nhiên, với phần mềm miễn phí này, bạn không tự thể bán nó và nếu bạn thay đổi mã nguồn và cho ra phiên bản mới, phiên bản của bạn phải miễn phí. Ví dụ phần mềm nguồn mở điển hình nhất là hệ điều hành Linux.

1.3.4 Hiệu năng máy tính

Nhìn chung, hiệu năng của máy tính phụ thuộc vào các thiết bị hoạt động với nhau. Nếu chúng ta nâng cấp một phần thiết bị của máy tính trong khi để lại phần khác lỗi thời thì sẽ không cải thiện hiệu năng. Các yếu tố ảnh hưởng đến hiệu suất máy tính bao gồm tốc độ của bộ xử lý trung tâm (CPU), dung lượng và tốc bộ nhớ RAM, tốc độ đĩa cứng và card đồ họa (graphic card)

1.3.4.1 Tốc độ của bộ xử lý trung tâm (CPU): tốc độ CPU có liên hệ với tần số đồng hồ làm việc của nó (tính bằng các đơn vị như MHz, GHz, v.v..). Đối với các CPU cùng loại, tần số này cao hơn cũng có nghĩa là tốc độ xử lý cao hơn. Đối với CPU khác loại, điều này chưa chắc đã đúng. Ví dụ: CPU Intel Core 2 Duo có tần số 2,6 GHz có thể xử lí nhanh hơn CPU Intel Pentium 4 có tần số 3,4 GHz. Tốc độ CPU còn phụ thuộc vào bộ nhớ đệm - bộ nhớ dùng để lưu các lệnh/dữ liệu thường dùng hay có khả năng sẽ được dùng trong tương lai gần, giúp giảm bớt thời gian chờ đợi của CPU. Ví dụ: Intel Core 2 Duo sử dụng chung cache L2 (shared cache, L1 có vùng bộ nhớ đệm 128 KB, L2 là 256-512KB) giúp cho tốc độ xử lý của hệ thống hai nhân mới này cao hơn so với hệ thống hai nhân thế hệ thứ nhất (Intel Pentium D) với mỗi nhân từng bộ nhớ đệm L2 riêng biệt.

1.3.4.2 Dung lượng và tốc bộ nhớ RAM: dung lượng và tốc độ của bộ nhớ RAM trong máy tính của bạn tạo sự khác biệt rất lớn cách thực hiện máy tính của bạn. Nếu bạn đang cố gắng để chạy Windows XP với 64 MB bộ nhớ RAM thì có lẽ nó không làm việc. Khi các máy tính sử dụng hết tất cả dung lượng RAM có sẵn, máy tính bắt đầu sử dụng ổ cứng để làm bộ nhớ ảo. Việc chuyển đổi dữ liệu giữa RAM và bộ nhớ ảo (bộ nhớ ổ đĩa cứng) làm chậm máy tính xuống đáng kể. Đặc biệt là khi cố gắng tải các ứng dụng hay tập tin. Ngoài ra tốc độ của bộ nhớ RAM của bạn có thể bị ảnh hưởng. Tốc độ bình thường của bộ nhớ RAM trong hầu hết các máy tính hiện nay là PC100 (100MHz). Điều này chạy tốt cho hầu hết các ứng dụng. Các máy cao cấp có thể được sử dụng DDR (double data rate - tốc độ dữ liệu kép) RAM. Nó mới hơn và đắt tiền hơn, nhưng chạy nhanh hơn đáng kể (266MHz).

1.3.4.3 Tốc độ đĩa cứng (disk speed): yếu tố lớn nhất trong hoạt động của máy tính của bạn là tốc độ đĩa cứng, tốc độ các ổ đĩa cứng có thể tìm, đọc, viết, và chuyển dữ liệu sẽ làm cho một khác biệt lớn trong cách máy tính của bạn thực hiện. Hầu hết các ổ cứng ngày nay quay tại 7.200 rpm, máy tính xách tay vẫn quay tại 5.200 rpm, đó là một trong những lý do máy tính xách tay thường xuất hiện chậm chạp với một máy tính để bàn tương đương. Dung lượng của ổ đĩa cứng của bạn đóng một vai trò trong hiệu năng của một máy tính. Miễn là bạn có đủ không gian trống cho bộ nhớ ảo và giữ cho đĩa chống phân mảnh máy tính sẽ thực hiện cũng không có vấn đề gì kích thước.

1.3.4.4 Tốc độ card đồ họa: nếu bạn chạy các trò chơi 3-D hoặc các chương trình chỉnh sửa video, tốc độ card đồ họa của bạn có thể trở nên quan trọng ảnh hưởng hiệu năng của máy tính. Card đồ họa giúp cho hiệu suất của máy tính bằng cách nhận trách nhiệm xử lý đồ họa 3D và các nhiệm vụ phức tạp khác.

1.3.5 Mạng máy tính và truyền thông

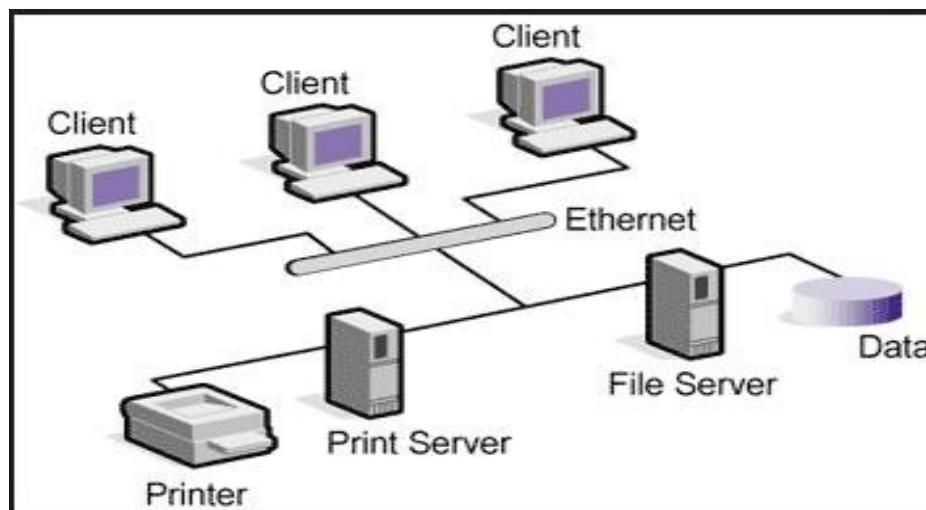
1.3.5.1 Khái niệm mạng máy tính

Một mạng máy tính là một hệ thống trong đó nhiều máy tính được kết nối với nhau để chia sẻ thông tin và nguồn tài nguyên. Khi các máy tính được nối trong một mạng, mọi người có thể chia sẻ tập tin và thiết bị ngoại vi như modem, máy in, băng đĩa sao lưu, hoặc ổ đĩa CD-ROM. Khi các mạng được nối với Internet, người dùng có thể gửi e-mail, tiến hành hội nghị video trong thời gian thực với những người dùng khác từ xa. ..., cho phép chia sẻ các chương trình phần mềm hoặc điều hành trên hệ thống từ xa.

Mạng có thể được cấu hình theo nhiều cách. Có hai loại chính: Mạng ngang hàng (peer to peer - P2P) và mạng máy khách - máy chủ (client - server).

Mạng ngang hàng: phổ biến nhất được tìm thấy trong cơ quan và doanh nghiệp nhỏ. Trong một mạng ngang hàng, mỗi nút (node) trên mạng có thể giao tiếp với tất cả các nút khác. Một nút (node) có thể là một máy tính, máy in, máy quét, modem, hoặc bất kỳ thiết bị ngoại vi khác có thể được kết nối với một máy tính. Mạng ngang hàng là tương đối dễ dàng thiết lập, nhưng có xu hướng là khá nhỏ.

Mạng Máy khách – Máy chủ: thường có hai loại máy tính khác nhau. Các máy chủ là máy tính cung cấp nguồn tài nguyên của nó, thường được lập trình để chờ đợi cho đến khi một người nào đó yêu cầu tài nguyên của nó. Các khách hàng là máy tính trong đó sử dụng các nguồn tài nguyên và gửi một yêu cầu đến máy chủ đang chờ đợi. (hình 1.16)



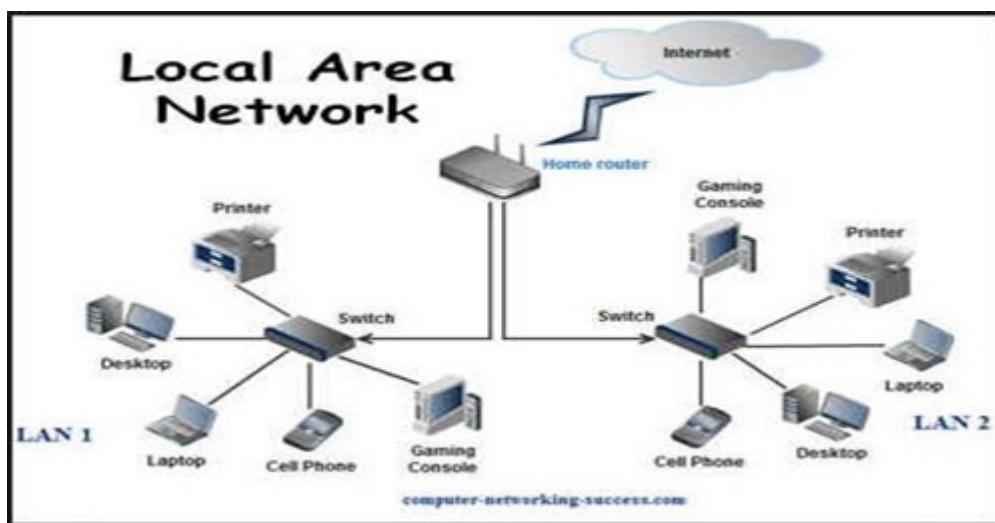
Hình 1.16: mạng Client – Server

- Các máy khách (client) là các máy tính được sử dụng máy trạm (workstation) để viết thư, gửi e-mail, hóa đơn, hoặc thực hiện bất kỳ nhiều nhiệm vụ. Các máy khách là một trong các máy mà hầu hết người dùng tương tác trực tiếp với nó,

Các máy tính server thường được giữ ở một nơi an toàn và được sử dụng để quản lý tài nguyên mạng. Nếu một máy chủ được phân công giải quyết một nhiệm vụ cụ thể, nó được biết đến như một máy chủ chuyên dụng. Ví dụ, một máy chủ Web được sử dụng để cung cấp các trang Web, một máy chủ tập tin được sử dụng để lưu trữ và kho lưu trữ tập tin, và một máy chủ in quản lý các nguồn tài nguyên in ấn cho mạng. Mỗi trong số này là một máy chủ chuyên dụng. Mô hình mạng client – server thường được sử dụng khi số nút (node) được nối kết trong mô hình vượt quá 10.

1.3.5.2 Mạng cục bộ (LAN) và mạng diện rộng (WAN)

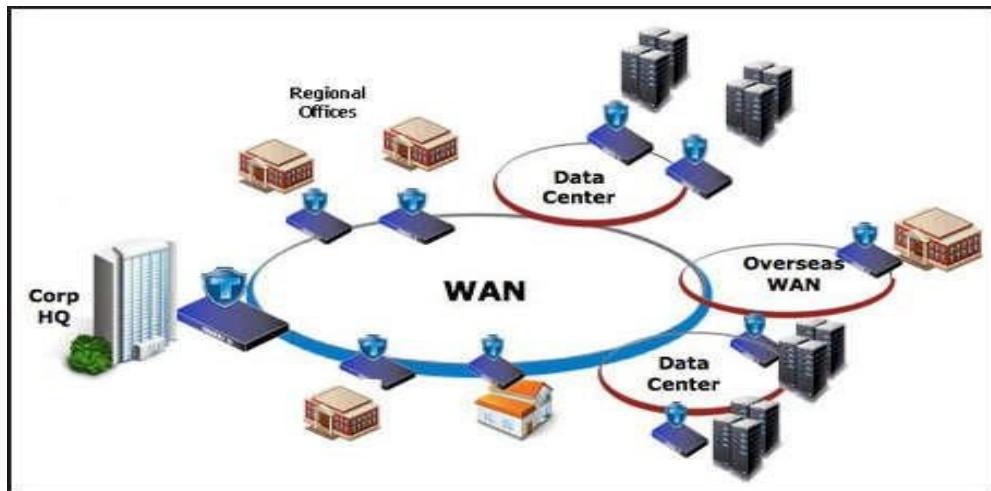
Mạng cục bộ (local area network - LAN): là mạng thường giới hạn trong một khu vực địa lý, chẳng hạn như một tòa nhà đơn lẻ hoặc một trường đại học (hình 1.17). Mạng LAN có thể phục vụ vài người sử dụng (ví dụ, trong một mạng lưới văn phòng nhỏ) hoặc vài trăm người sử dụng trong một văn phòng lớn hơn. Mạng LAN bao gồm cáp, switch, router và các thành phần khác cho phép người dùng kết nối đến các máy chủ nội bộ, các trang web và các mạng LAN khác thông qua các mạng diện rộng (wide area network - WAN).



Hình 1.17: Mạng LAN

Mạng diện rộng (Wide Area Network – WAN): một mạng diện rộng (WAN) là mạng tồn tại trên một khu vực vùng địa lý lớn thường cho quốc gia hay cả lục địa, phạm vi vài trăm cho đến vài ngàn km (hình 1.18). Mạng WAN kết nối các mạng LAN khác nhau và

được sử dụng cho các khu vực địa lý rộng lớn hơn. Mạng WAN tương tự như một hệ thống ngân hàng, nơi hàng trăm chi nhánh ở các thành phố, quốc gia khác nhau được kết nối với nhau để chia sẻ dữ liệu của họ.



Hình 1.18: Mạng WAN

Internet: là một liên kết các mạng trên phạm vi toàn thế giới. Với sự gia tăng nhanh chóng nhu cầu kết nối, Internet đã trở thành một xa lộ thông tin liên lạc cho hàng triệu người sử dụng. Internet ban đầu được hạn chế cho các tổ chức quân sự và học tập, ngày nay Internet có hàng tỷ trang web được tạo ra bởi con người và các công ty từ khắp nơi trên thế giới. Internet cũng có hàng ngàn dịch vụ giúp cho cuộc sống thuận tiện hơn. Ví dụ, nhiều tổ chức tài chính cung cấp cho ngân hàng trực tuyến cho phép người dùng quản lý và xem tài khoản trực tuyến của họ....

Mạng nội bộ (Intranet): là một mạng riêng trong một doanh nghiệp, một tổ chức. Nó có thể bao gồm nhiều mạng cục bộ liên kết với nhau. Tất cả các đối tượng muốn truy cập vào hệ thống intranet đang hoạt động đều cần phải có những yếu tố xác thực thông tin chính xác, ví dụ như Username (tài khoản) và Password (mật khẩu). Các hoạt động trên intranet đều có quy trình tương tự như các website thông thường trên internet khác, nhưng điểm khác biệt ở đây là intranet được bảo vệ bởi 1 lớp Firewall, để nhằm nâng cao bảo mật thông tin cho khách hàng tránh khỏi những yếu tố truy cập bất thường ở ngoài hệ thống mà không rõ nguồn gốc. Mục đích chính của một mạng nội bộ là để chia sẻ thông tin công ty và các tài nguyên máy tính giữa các nhân viên. Một mạng nội bộ cũng có thể được sử dụng để tạo điều kiện làm việc theo nhóm và hội nghị từ xa.

Mạng extranet: một extranet là giống như một mạng nội bộ, nhưng cung cấp truy cập được kiểm soát từ bên ngoài đối với khách hàng, các nhà cung cấp, đối tác, hoặc những người khác bên ngoài. Extranet là phần mở rộng, hoặc các phân đoạn của mạng intranet

tư nhân được xây dựng bởi các doanh nghiệp để chia sẻ thông tin và thương mại điện tử.

CÂU HỎI ÔN TẬP

Mục đích yêu cầu

- Làm quen với các khái niệm về máy tính, cách lưu trữ và xử lý thông tin trong máy tính.
- Biết cách chuyển đổi số giữa các cơ số thập phân, bát phân, nhị phân, thập lục phân.
- Biết và xác định được các cấu hình phần cứng trong máy vi tính.
- Biết phân biệt các phần mềm hệ thống, phần mềm ứng dụng...
- Hiểu sơ về các hình thức lắp đặt mạng .

Câu 1: Hãy cho biết tại sao em chọn ngành CNTT ?

Câu 2 :Trình bày khái niệm thông tin và quá trình xử lý thông tin

Câu 3: Hệ đếm là gì ? Trong ngành toán - tin học hiện nay phổ biến có mấy hệ đếm?

Câu 4: Hãy cho biết các loại máy vi tính (microcomputers)

Câu 5: Phân biệt sự giống và khác nhau giữa máy tính để bàn (desktop computers), máy tính xách tay (notebook computers) và máy tính bảng (tablet computers)

Câu 6: Phân biệt sự giống và khác nhau giữa bộ nhớ trong và bộ nhớ ngoài

Câu 7: Phân biệt sự khác và giống nhau giữa bộ nhớ ROM và bộ nhớ RAM

Câu 8: Hãy trình bày các thiết bị nhập xuất

Câu 9: Hãy so sánh tốc độ truyền của các loại cổng (port) giao tiếp

Câu 10: Phần mềm hệ thống là gì? Phần mềm ứng dụng là gì? Trình bày các loại phần mềm ứng dụng

Câu 11: Các yếu tố nào ảnh hưởng đến hiệu suất máy tính?

Câu 12: Phân biệt mạng ngang hàng (peer to peer - P2P) và mạng máy khách - máy chủ (client - server)

Câu 13: Phân biệt mạng cục bộ và mạng diện rộng

Chọn câu trả lời đúng nhất cho các câu hỏi sau

Câu 14: Đơn vị dùng để đo thông tin được gọi:

- Bit
- Byte
- Kilobyte
- Dữ liệu

Câu 15: Số 1110 trong hệ nhị phân tương đương số nào trong các số sau đây của hệ thập phân:

- a 12
- b 13
- c 10
- d 14

Câu 16: Hệ thống thông tin là gì?

- a Là hệ thống xử lý dữ liệu
- b Là hệ thống lưu trữ, xử lý và xuất dữ liệu
- c Là hệ thống tin tức, lưu trữ dữ liệu
- d Là hệ thống xuất dữ liệu

Câu 17: Thuật ngữ nào dưới đây trình bày cho tốc độ CPU nhanh nhất

- a. 733 MHz
- b. 286 MHz
- c. 2 GHz
- d. 2 GB

Câu 18: Những thiết bị nào sau đây không phải là một thiết bị nhập (Input)

- a. Bàn phím (keyboard)

b. Loa (speaker)

c. Chuột (mouse)

d. bút kỹ thuật số

Câu 19 : Hãy đổi tất cả các số từ 1 đến 20 từ hệ thập phân ra hệ nhị phân, bát phân và thập lục phân.

Câu 20 : Hãy chuyển đổi số 101010 từ hệ nhị phân ra các hệ còn lại.

BÀI 2: CÁC KIẾN THỨC CƠ BẢN VỀ MÁY TÍNH

Sau khi học xong bài này, sinh viên có thể nắm được về cơ bản:

- *Khái niệm về ngôn ngữ lập trình.*
- *Khái niệm về kiểu dữ liệu*
- *Khái niệm về giải thuật*
- *Ngôn ngữ biểu diễn giải thuật.*
- *Ngôn ngữ sơ đồ (lưu đồ), sử dụng lưu đồ để biểu diễn các giải thuật.*
- *Các kiểu dữ liệu cơ bản*
- *Ngôn ngữ lập trình*

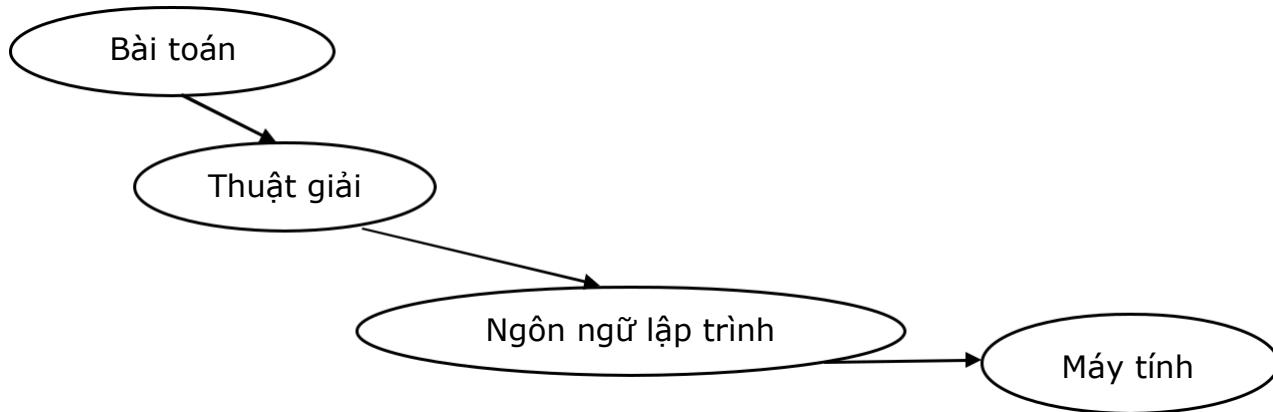
2.1 TỪ BÀI TOÁN ĐẾN CHƯƠNG TRÌNH

Giả sử chúng ta cần viết một chương trình để giải phương trình bậc 2 có dạng $ax^2 + bx + c = 0$, hay viết một chương trình kiểm tra xem một số tự nhiên nào đó có phải là số nguyên tố hay không? Công việc đầu tiên là chúng ta phải hiểu và biết cách giải bài toán bằng lời giải thông thường của người làm toán. Để giải được bài toán trên bằng máy tính (lập trình cho máy tính giải) thì chúng ta cần phải thực hiện qua các bước như sau:

1. Mô tả các bước giải bài toán.
 2. Vẽ sơ đồ xử lý dựa trên các bước.
 3. Dựa trên sơ đồ xử lý để viết chương trình xử lý bằng ngôn ngữ giả (ngôn ngữ bình thường của chúng ta).
 4. Chọn ngôn ngữ lập trình và chuyển chương trình từ ngôn ngữ giả sang ngôn ngữ lập trình để tạo thành một chương trình hoàn chỉnh.
-

5. Thực hiện chương trình: nhập vào các tham số, nhận kết quả.

Trong nhiều trường hợp, từ bài toán thực tế chúng ta phải xây dựng mô hình toán học rồi mới xác định được các bước để giải. Vấn đề này sẽ được trình bày chi tiết trong môn Cấu Trúc Dữ Liệu.



2.2 GIẢI THUẬT

2.2.1 Khái niệm giải thuật

Giải thuật là một hệ thống chặt chẽ và rõ ràng các quy tắc nhằm xác định một dãy các thao tác trên những dữ liệu vào sao cho sau một số hữu hạn bước thực hiện các thao tác đó ta thu được kết quả của bài toán.

Ví dụ 1: Giả sử có hai bình A và B đựng hai loại chất lỏng khác nhau, chẳng hạn bình A đựng rượu, bình B đựng nước mắm. Giải thuật để hoán đổi (swap) chất lỏng đựng trong hai bình đó là:

* **Yêu cầu phải có thêm một bình thứ ba gọi là bình C.**

- Bước 1: Đổ rượu từ bình A sang bình C.
- Bước 2: Đổ nước mắm từ bình B sang bình A.
- Bước 3: Đổ rượu từ bình C sang bình B.

Ví dụ 2: Một trong những giải thuật tìm ước chung lớn nhất của hai số a và b là:

- Bước 1: Nhập vào hai số a và b.

- Bước 2: So sánh 2 số a,b chọn số nhỏ nhất gán cho UCLN.
- Bước 3: Nếu một trong hai số a hoặc b không chia hết cho UCLN thì thực hiện bước 4, ngược lại (cả a và b đều chia hết cho UCLN) thì thực hiện bước 5.
- Bước 4: Giảm UCLN một đơn vị và quay lại bước 3
- Bước 5: In UCLN - Kết thúc.

2.2.2 Các đặc trưng của giải thuật

Khác với các bài toán thuần túy toán học chỉ cần xác định rõ giả thiết và kết luận chứ không cần xác định yêu cầu về lời giải.

Tuy nhiên đối với những bài toán tin học ứng dụng trong thực tế chúng ta chỉ cần tìm lời giải tốt tới mức nào đó, thậm chí tồi ở mức chấp nhận được nếu lời giải tốt đòi hỏi thời gian và chi phí.

Ví dụ 3: khi cài đặt các hàm số phức tạp trên máy tính. Nếu tính bằng cách khai triển chuỗi vô hạn thì độ chính xác cao hơn nhưng tốc độ chậm hơn rất nhiều so với phương pháp xấp xỉ

Việc xác định đúng yêu cầu bài toán là rất quan trọng bởi nó ảnh hưởng tới cách thức giải quyết và chất lượng của lời giải. Một bài toán thực tế thường cho bởi những thông tin khá mơ hồ và hình thức, ta phải phát biểu lại một cách chính xác và chặt chẽ để hiểu đúng bài toán.

Ví dụ 4: Lập trình tự động cho bài toán giặt máy.

Việc xác định độ dơ của quần áo cần giặt thì khá mơ hồ.

Trên thực tế, ta nên xét một vài trường hợp cụ thể để thông qua đó hiểu được bài toán rõ hơn và thấy được các bước cần thiết phải tiến hành.

Khái niệm giải thuật hay thuật giải mà nhiều khi còn được gọi là thuật toán dùng để chỉ phương pháp hay cách thức (method) để giải quyết vấn đề.

Một thuật giải tốt là thuật giả phải bảo đảm:

1. **Tính kết thúc:** Giải thuật phải dừng sau một số hữu hạn bước.
2. **Tính xác định:** Các thao tác máy tính phải thực hiện được và các máy tính khác nhau thực hiện cùng một bước của cùng một giải thuật phải cho cùng một kết quả.

3. **Tính phổ dụng:** Giải thuật phải "vết' hết các trường hợp và áp dụng cho một loạt bài toán cùng loại.
4. **Tính hiệu quả:** Một giải thuật được đánh giá là tốt nếu nó đạt hai tiêu chuẩn sau:
 - Thực hiện nhanh, tốn ít thời gian.
 - Tiêu phí ít tài nguyên của máy, chẳng hạn tốn ít bộ nhớ.

Giải thuật tìm UCLN nếu trên đạt tính kết thúc bởi vì qua mỗi lần thực hiện bước 4 thì UCLN sẽ giảm đi một đơn vị cho nên trong trường hợp xấu nhất thì UCLN=1, giải thuật phải dừng. Các thao tác trình bày trong các bước, máy tính đều có thể thực hiện được nên nó có tính xác định. Giải thuật này cũng đạt tính phổ dụng vì nó được dùng để tìm UCLN cho hai số nguyên dương a và b bất kỳ. Tuy nhiên tính hiệu quả của giải thuật có thể chưa cao; cụ thể là thời gian chạy máy có thể còn tốn nhiều hơn một số giải thuật khác mà chúng ta sẽ có dịp trở lại trong phần lập trình C.

2.2.3 Ngôn ngữ biểu diễn giải thuật

Để biểu diễn giải thuật, cần phải có một tập hợp các ký hiệu dùng để biểu diễn, mỗi ký hiệu biểu diễn cho một hành động nào đó. Tập hợp các ký hiệu đó lại tạo thành ngôn ngữ biểu diễn giải thuật.

1. Ngôn ngữ tự nhiên (Natural language)

Là thứ ngôn ngữ mà chúng ta đang sử dụng, chúng ta có thể sử dụng ngôn ngữ tự nhiên để mô tả giải thuật giống như các ví dụ ở trên. nhưng có thể sự mô tả không rõ ràng trong ngôn ngữ dẫn đến hiểu sai giải thuật.

Ví dụ 5: Ta có giải thuật giải phương trình bậc 1 $ax + b = 0$ như sau:

- Bước 1: Nhận giá trị của các tham số a, b.
- Bước 2: Xét giá trị của a xem có bằng 0 hay không? Nếu a=0 thì làm bước 3, Nếu a khác không thì làm bước 4.
- Bước 3: (a bằng 0) Nếu b bằng 0 thì ta kết luận phương trình vô số nghiệm, nếu b khác 0 thì ta kết luận phương trình vô nghiệm.

- Bước 4: (a khác 0) Ta kết luận phương trình có nghiệm $x = -b/a$

2. Mã giả pseudocode)

Chúng ta cũng có thể dùng một ngôn ngữ giả lập ngôn ngữ lập trình gọi là **mã giả (pseudocode)** để biểu diễn giải thuật cho một bài toán

3. Ngôn ngữ sơ đồ (Lưu đồ - flow chart)

Lưu đồ thuật toán là công cụ dùng để **biểu diễn thuật toán dưới dạng sơ đồ**, việc mô tả dữ liệu **nhập** (input), dữ liệu **xuất** (output) và luồng xử lý thông qua các **ký hiệu hình học**.

Một dạng quy ước chuẩn cho phép mô tả cách xử lý của chương trình một cách trực quan.

Các ký hiệu thường sử dụng để vẽ lưu đồ

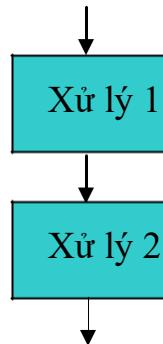
STT	KÝ HIỆU	DIỄN GIẢI
1		Bắt đầu chương trình
2		Kết thúc chương trình
3		Luồng xử lý
4		Điều khiển lựa chọn
5		Nhập
6		Xuất
7		Xử lý, tính toán hoặc gán
8		Trả về giá trị (return)
9		Điểm nối liên kết tiếp theo (Sử dụng khi lưu đồ vượt quá trang)

2.2.4 Các cấu trúc suy luận cơ bản của giải thuật

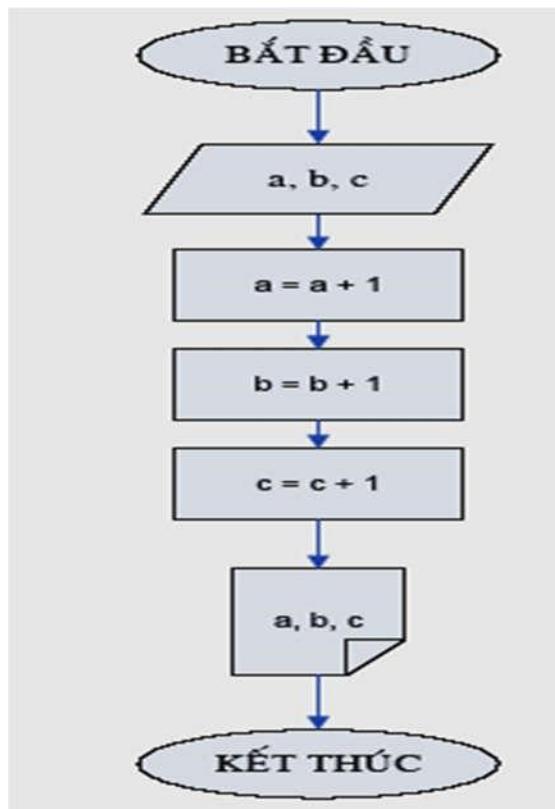
Giải thuật được thiết kế theo ba cấu trúc suy luận cơ bản sau đây:

2.2.4.1 Cấu trúc Tuần tự (Sequential)

Các công việc được thực hiện một cách tuần tự từ trên xuống, công việc này nối tiếp công việc kia.



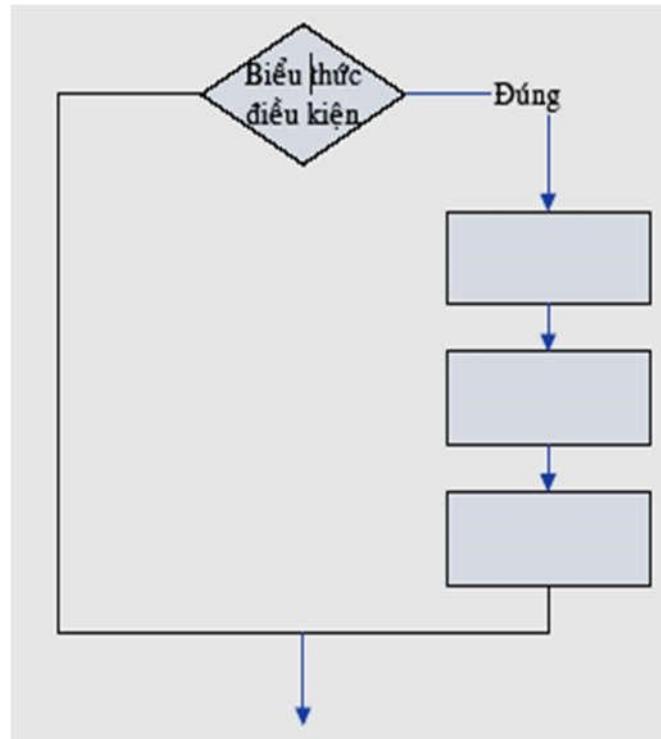
Bài toán 1: Nhập vào 3 số nguyên **a**, **b**, **c** và xuất ra màn hình với giá trị của mỗi số **tăng lên 1**.



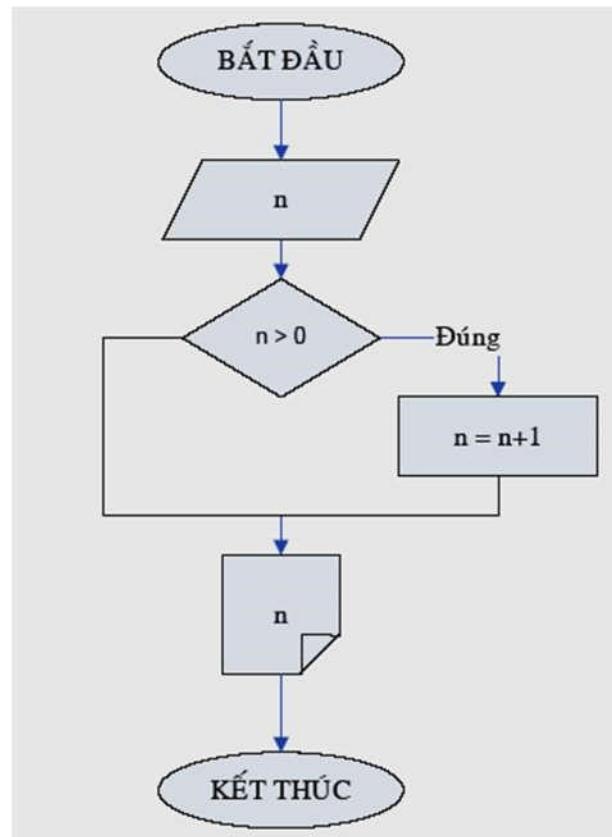
2.2.4.2 Cấu trúc lựa chọn (Selection)

Lựa chọn một công việc để thực hiện căn cứ vào một điều kiện nào đó. Có một số dạng như sau:

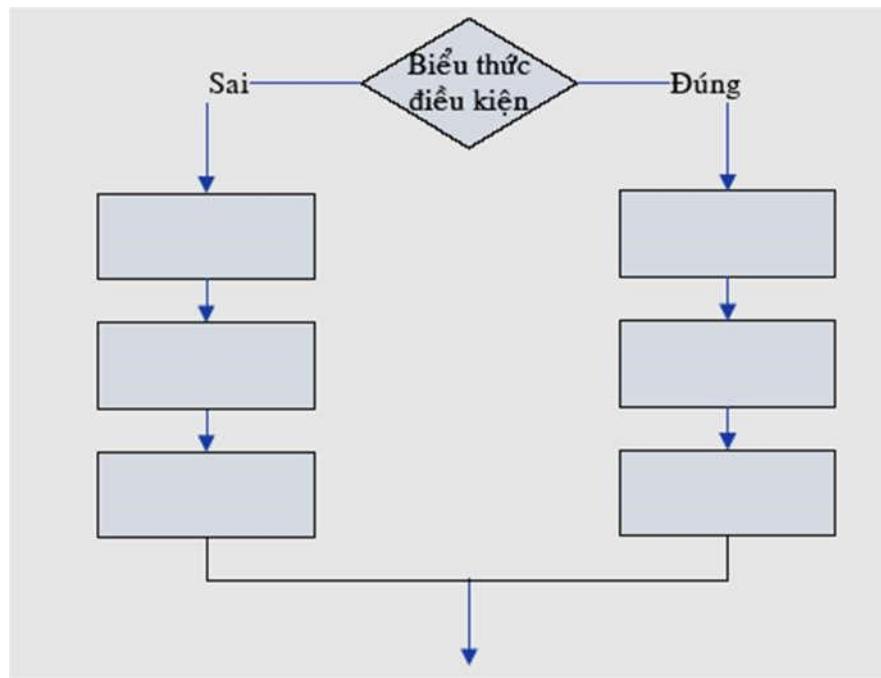
❖ **Cấu trúc 1:** Nếu <điều kiện> (đúng) thì thực hiện <công việc>.



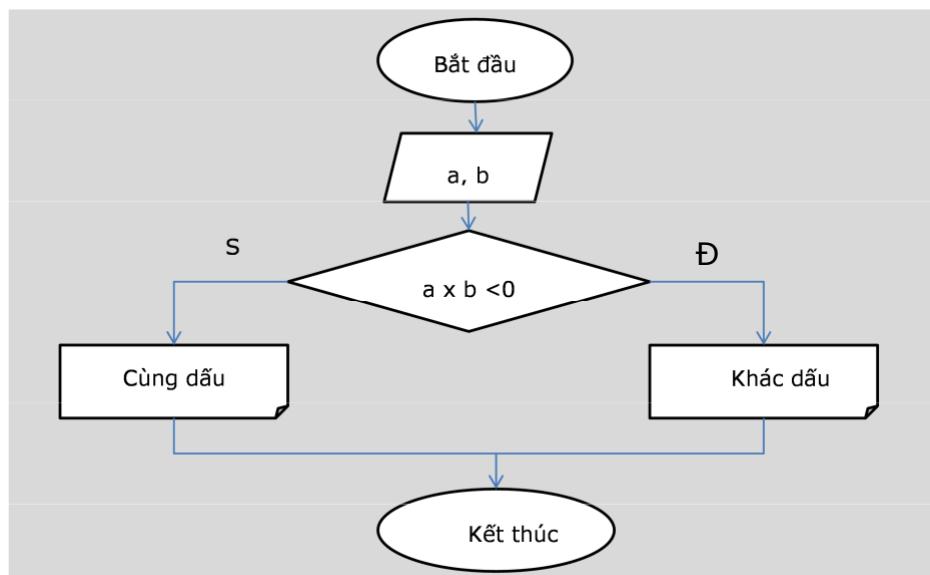
Bài toán 2: Nhập 1 số n, nếu số đó >0 thì tăng 1. Sau đó hiển thị



- ❖ **Cấu trúc 2:** Nếu <điều kiện> (đúng) thì thực hiện <công việc 1>, ngược lại (điều kiện sai) thì thực hiện <công việc 2>

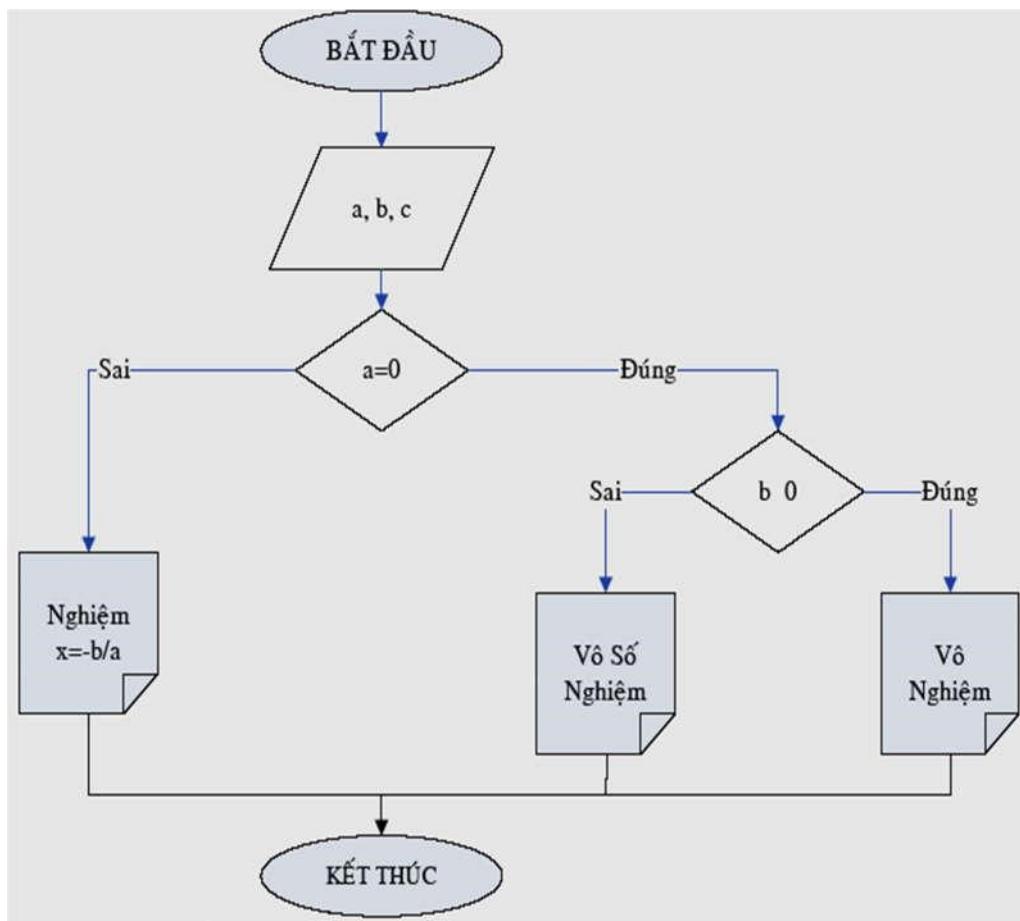


Bài toán 3: Viết chương trình nhập vào 2 số thực, cho biết 2 số đó cùng dấu hay khác dấu



* **Cấu trúc 3:** Trường hợp $<i>$ thực hiện $<\text{công việc } i>$

Bài toán 4: Viết chương trình giải và biện luận phương trình bậc nhất $ax+b=0$



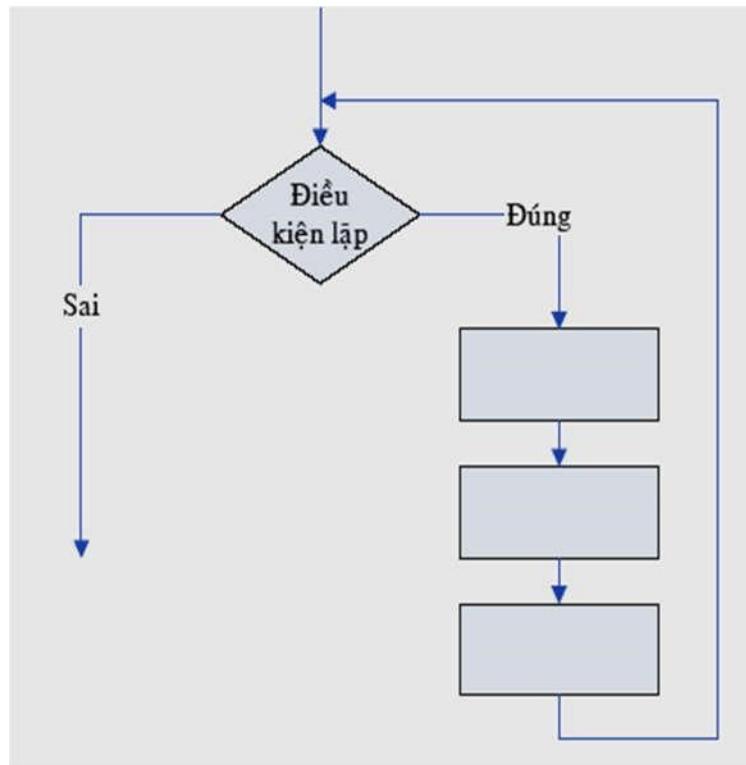
2.2.4.3 Cấu trúc lặp (Repeating)

Thực hiện lặp lại một công việc không hoặc nhiều lần căn cứ vào một điều kiện nào đó. Có hai dạng như sau:

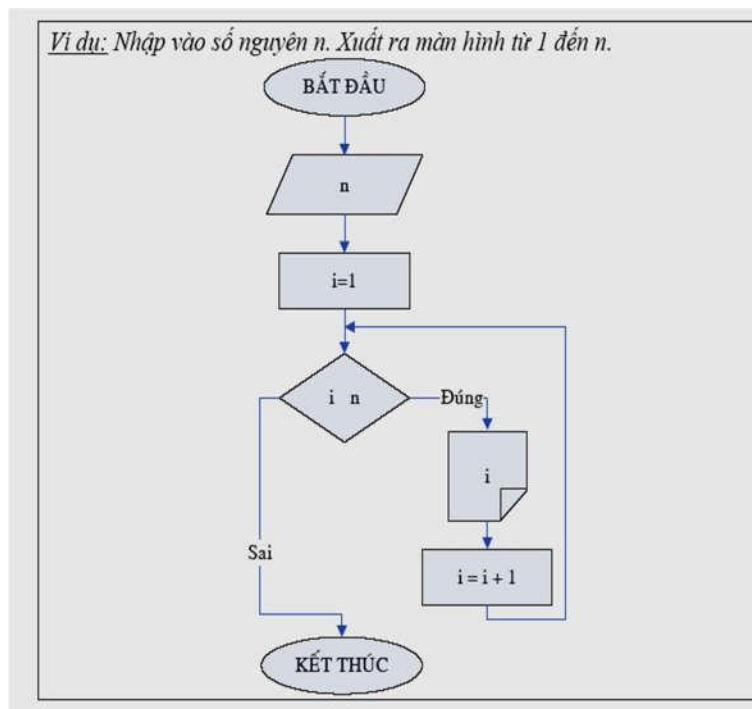
- **Lặp xác định:** là loại lặp mà khi viết chương trình, người lập trình đã xác định được công việc sẽ lặp bao nhiêu lần.

- **Lặp không xác định:** là loại lặp mà khi viết chương trình người lập trình chưa xác định được công việc sẽ lặp bao nhiêu lần. Số lần lặp sẽ được xác định khi chương trình thực thi.

Trong một số trường hợp người ta cũng có thể dùng các cấu trúc này để diễn tả một giải thuật.



Bài toán 5: Nhập vào một số nguyên n, xuất ra màn hình các số từ 1 đến n



Bước 1: nhập số n

Bước 2: xuất ra màn hình các số từ 1 đến n thì dừng.

2.3 KIỂU DỮ LIỆU

- Trong khoa học máy tính và lập trình máy tính, một **kiểu dữ liệu (Data type)** hay đơn giản **type** là một cách phân loại dữ liệu cho trình biên dịch hoặc thông dịch hiểu các lập trình viên muốn sử dụng dữ liệu loại gì. Hầu hết các ngôn ngữ hỗ trợ nhiều kiểu dữ liệu khác nhau, như số Thực, số Nguyên hay Boolean. Một kiểu dữ liệu cung cấp một bộ các giá trị mà từ đó một biểu thức (ví dụ như biến, hàm...) có thể lấy giá trị của nó. Kiểu định nghĩa các toán tử có thể được thực hiện trên dữ liệu của nó (ví dụ: $y=a+b$), và nó có thể được lưu trữ trong máy tính.

- Có 2 kiểu dữ liệu là kiểu dữ liệu cơ bản và kiểu dữ liệu phức hợp

Trong lĩnh vực khoa học máy tính, **kiểu dữ liệu cơ bản** (primitive data type) là kiểu được định nghĩa bởi một ngôn ngữ lập trình làm gốc để xây dựng các kiểu dữ liệu phức hợp khác.

Ví dụ : Các kiểu dữ liệu cơ bản trong ngôn ngữ C

Kiểu	Dung lượng xấp xỉ (đơn vị là bit)	Phạm vi
char	8	-128 tới 127
unsigned	8	0 tới 255
signed char	8	-128 tới 127
int	16	-32,768 tới 32,767
unsigned int	16	0 tới 65,535
signed int	16	Giống như kiểu int
short int	16	-128 tới 127
unsigned short int	16	0 tới 65,535
signed short int	16	Giống như kiểu short int
long int	32	-2,147,483,648 tới 2,147,483,647
signed long int	32	Giống như kiểu long int
unsigned long int	32	0 tới 4,294,967,295

float	32	6 con số thập phân
double	64	10 con số thập phân
long double	128	10 con số thập phân

Kiểu dữ liệu phức hợp do người dùng định nghĩa, được tạo ra để giải quyết một bài toán, một vấn đề nào đó mà kiểu dữ liệu cơ bản không giải quyết được (chúng ta sẽ nghiên cứu nó ở những phần sau) .

2.4 NGÔN NGỮ LẬP TRÌNH (Programming Language)

Là dạng ngôn ngữ máy tính sử dụng để phát triển các chương trình phần mềm, tập lệnh hoặc các chuẩn hóa theo một hệ thống các quy tắc riêng để máy tính thực thi.

Hiện nay có rất nhiều ngôn ngữ lập trình đang được sử dụng. Mặc dù các ngôn ngữ cũng có điểm chung tương đồng nhưng mỗi ngôn ngữ lại có các cú pháp sử dụng riêng. Công việc của các lập trình viên là họ phải học các quy tắc, cú pháp và cấu trúc ngôn ngữ rồi thực hiện viết mã nguồn trong một trình soạn thảo hoặc IDE và biên dịch code thành ngôn ngữ máy để máy tính có thể hiểu được. Các ngôn ngữ script không yêu cầu trình biên dịch mà sử dụng các trình thông dịch để thực thi script.



2.4.1 Các loại ngôn ngữ lập trình

Các ngôn ngữ lập trình có thể được chia thành một số mô hình sau đây. Mỗi ngôn ngữ có thể thuộc nhiều loại.

- Ngôn ngữ Bậc cao (High-level) - Ngôn ngữ Bậc thấp (Low-level)
- Lập trình Khai báo (Declarative) - Lập trình Mệnh lệnh (Imperative) - Lập trình Thủ tục (Procedural)
- Lập trình Đa năng (General-purpose) - Lập trình Chuyên biệt (Domain-specific)
- Lập trình Hướng đối tượng (Object-oriented) - Lập trình Đồng thời (Concurrent)
- Ngôn ngữ Dòng lệnh (Command) - Ngôn ngữ Biên dịch (Compiled) - Ngôn ngữ Thông dịch

Chú ý: Còn rất nhiều mô hình khác có thể được sử dụng để phân loại ngôn ngữ lập trình.

2.4.2 Danh sách ngôn ngữ lập trình máy tính

Ngày nay, có hàng trăm ngôn ngữ lập trình khác nhau. Dưới đây là những cái tên nổi bật nhất.

A-C	D-K	L-Q	R-Z
ActionScript	D	LeLisp	R
ALGOL	DarkBASIC	Lisp	Racket
Ada	Dart	LiveScript	Reia
AIML *	Datalog	LOGO	RPG
Altair BASIC	dBASE	Lua	Ruby
Assembly	Dylan	MACLISP	Rust
AutoHotkey	EuLisp	Matlab	Scala
Babel	Elixir	Metro	Scheme
BASIC	F	MUMPS	Scratch
Batch file	F#	Nim	SGML
BCPL	FORTRAN	Objective-C	Simula
BeanShell	FoxPro	OCaml	Smalltalk

A-C	D-K	L-Q	R-Z
Brooks	Franz Lisp	Pascal	SPL
C	Go	Perl	SQL
C#	GW Basic	PHP	Stanford LISP
C++	Haskell	Pick	Swift
CL	HDML *	PureBasic	Tcl
Clojure	HTML *	Python	Turbo Pascal
COBOL	InterLisp	Prolog	True BASIC
CoffeeScript	ksh	QBasic	VHDL
Common Lisp	Java		Visual Basic
CPL	JavaScript		Visual FoxPro
CSS *	JCL		WML
Curl	Julia		WHTML
Curry	Kotlin		XLISP
			XML
			YAML
			ZetaLisp

Lưu ý: Các ngôn ngữ được đánh dấu hoa thị (*) trong danh sách trên **không phải là ngôn ngữ lập trình**; chúng có thể là *ngôn ngữ đánh dấu (markup)*, *ngôn ngữ định kiểu (style sheet)* hoặc *ngôn ngữ quản lý cơ sở dữ liệu (database management)*, nhưng vẫn được liệt kê trong danh sách vì một số trường hợp có thể coi chúng là ngôn ngữ lập trình.

Nhìn danh sách trên, có thể bạn sẽ cảm thấy khá choáng ngợp về sự "đông đảo" của các ngôn ngữ, và thật khó để bạn biết bắt đầu từ đâu nếu bạn là một người đang mong muốn bước chân vào lĩnh vực lập trình máy tính. Vì vậy, chúng ta sẽ phân loại một số lĩnh vực lập trình khác nhau và các ngôn ngữ sử dụng cho từng lĩnh vực.

2.4.2.1 Phát triển Ứng dụng và chương trình máy tính

Các ứng dụng và chương trình máy tính là những thứ bạn sử dụng để làm việc, học tập, giải trí hằng ngày. Ví dụ: trình duyệt Internet bạn đang sử dụng để xem một trang web nào đó được coi là một chương trình. Nếu quan tâm đến việc phát triển một chương trình ứng dụng, bạn nên xem xét các ngôn ngữ sau:

- **C**
- **C#**
- **C++**
- **D**
- **Java**
- **Swift**
- **Tcl**
- **Visual Basic**

2.4.2.2 Phát triển Trí tuệ nhân tạo

Lĩnh vực này liên quan đến Trí tuệ nhân tạo, hay hướng tới tạo ra các nhân vật có thể tương tác trong các trò chơi máy tính, các chương trình đưa ra quyết định, chatbot... Nếu bạn quan tâm đến việc phát triển AI và lĩnh vực này, hãy xem xét các ngôn ngữ sau:

- **AIML**
- **C**
- **C#**
- **C++**
- **Prolog**
- **Python**

2.4.2.3 Phát triển Cơ sở dữ liệu

Dành cho các nhà phát triển, nghiên cứu, duy trì cơ sở dữ liệu. Nếu bạn quan tâm đến cơ sở dữ liệu, hãy xem xét các ngôn ngữ sau đây:

- **DBASE**
- **FoxPro**
- **MySQL**
- **SQL**
- **Visual FoxPro**

2.4.2.4 Phát triển chương trình game

Phát triển trò chơi liên quan đến việc tạo trò chơi trên máy tính hoặc phần mềm giải trí khác. Nếu bạn quan tâm đến việc phát triển trò chơi, nên xem xét các ngôn ngữ sau:

- **C**
- **C#**
- **C++**
- **DarkBASIC**
- **Java**

2.4.2.5 Phát triển Driver máy tính

Nếu bạn quan tâm đến việc phát triển driver hoặc giao diện phần mềm cho các thiết bị phần cứng, bạn nên xem xét các ngôn ngữ sau:

- **Assembly**
- **C**

2.4.2.6 Phát triển website và Internet

Phát triển Internet và trang web là bản chất của Internet. Không có những nhà phát triển, Internet sẽ không tồn tại. Nếu bạn quan tâm đến việc tạo các trang web, ứng dụng Internet hoặc các tác vụ khác liên quan đến Internet, bạn nên xem xét các ngôn ngữ sau:

- **HDML**
- **HTML**
- **Java**
- **JavaScript**
- **Perl**
- **PHP**
- **Python**
- **XML**

2.4.2.7 Phát triển Script

Mặc dù việc nghiên cứu, phát triển script không có khả năng trở thành một nghề nghiệp chính, nhưng nếu bạn biết cách tạo và phát triển các tập lệnh, bạn có thể dễ dàng tăng năng suất cho bản thân và công ty của mình, giúp tiết kiệm vô số thời gian. Nếu bạn quan tâm đến việc phát triển các script, hãy xem xét các ngôn ngữ sau:

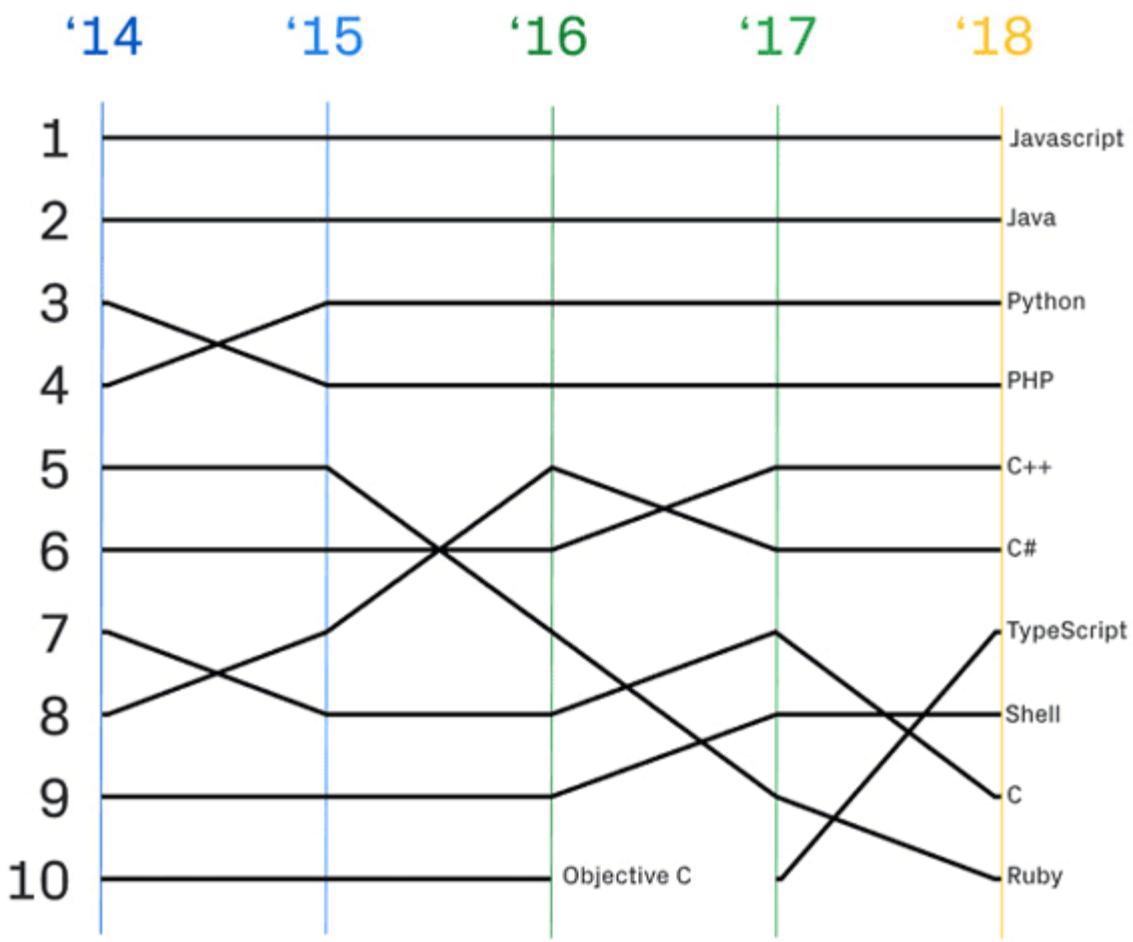
- **AutoHotkey**

- **awk**
- **bash**
- **Batch file**
- **Perl**
- **Python**
- **Tcl**

2.4.3 Ngôn ngữ lập trình nào phổ biến nhất?

Có rất nhiều đáp án khác nhau để trả lời câu hỏi này. Tuy nhiên, phương pháp tốt nhất là **dựa vào dữ liệu từ GitHub** - dịch vụ lưu trữ hơn 96 triệu dự án phần mềm khác nhau để có kết quả chính xác nhất. Trong biểu đồ dưới đây, GitHub đưa ra **10 cái tên hàng đầu dành cho ngôn ngữ lập trình phổ biến nhất từ năm 2014 đến 2018.**

Top 10 programming languages on GitHub



Trong biểu đồ trên, bạn có thể thấy 10 ngôn ngữ hàng đầu là: **JavaScript, Java, Python, PHP, C++, C#, TypeScript, Shell, C** và **Ruby**.

CÂU HỎI ÔN TẬP

Mục đích yêu cầu

- * Làm quen và nắm vững các cách mô tả giải thuật; từ đó đứng trước một bài toán cụ thể, sinh viên có thể mô tả thật chi tiết các bước để giải quyết vấn đề.

Lưu ý: Sinh viên cần xác định:

Input (dữ liệu nhập).

Output (dữ liệu xuất).

Tìm thuật toán phù hợp.

- * Biết phân biệt một số loại ngôn ngữ lập trình và hướng ứng dụng của nó, từ đó đặt ra mục tiêu cho chính mình.

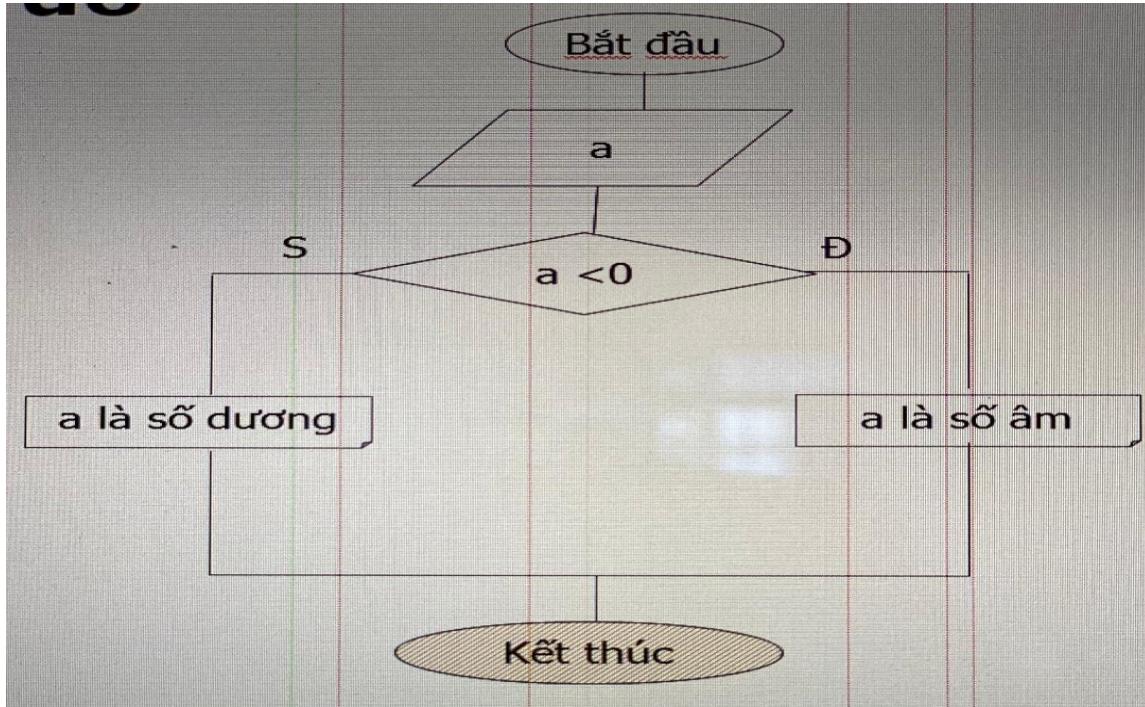
Câu 1 : Hãy cho biết lĩnh vực lập trình nào mà em thích ? kế hoạch thực thi nó .

Câu 2: Hãy làm quen và liệt kê các chức năng trên thanh toolbar của Dev C/ Cfree

Bằng ngôn ngữ tự nhiên và lưu đồ, anh (chị) hãy mô tả giải thuật cho các bài toán sau:

Câu 3 : Nhập vào một số nguyên a, cho biết a là số dương hay số âm
Hướng dẫn :

So sánh a với số 0 . Nếu $a > 0$ thì a dương, nếu $a < 0$ thì a là số âm.



Câu 4: Giải và biện luận phương trình bậc nhất $ax + b = 0$.

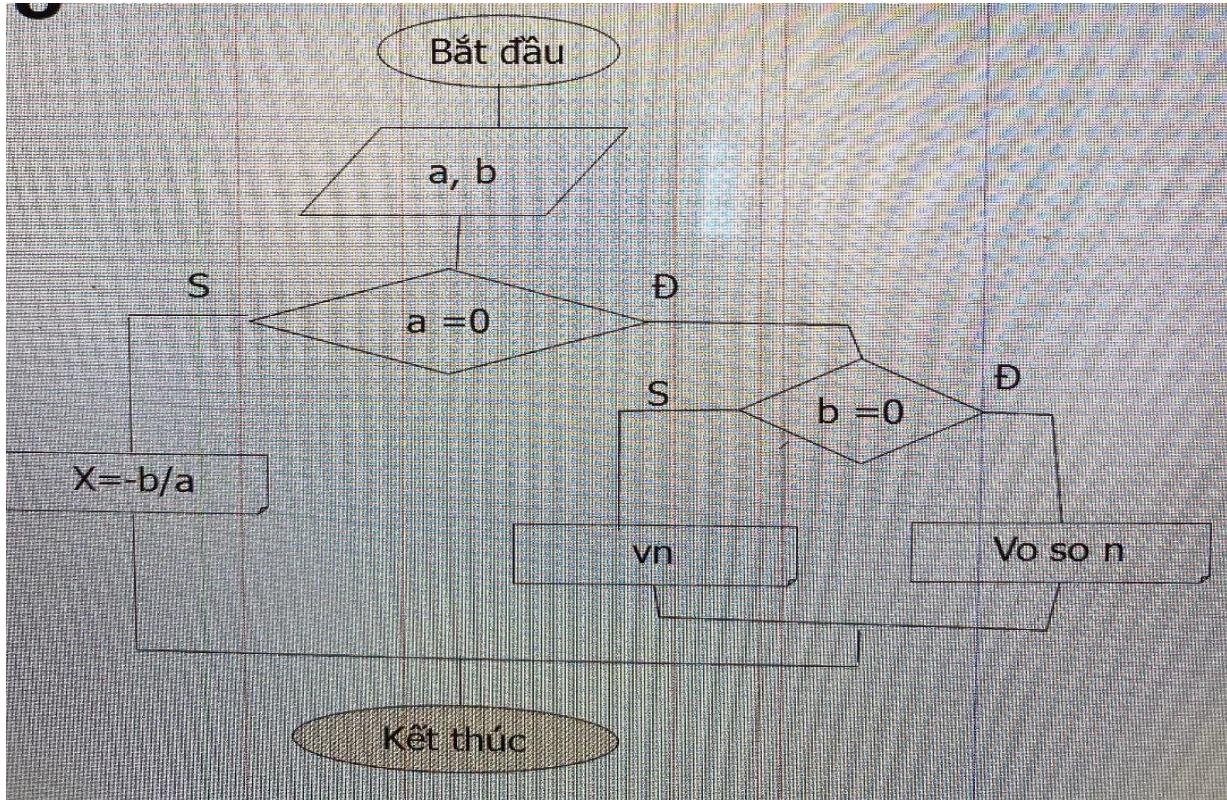
Hướng dẫn:

- Khai báo hai biến a, b kiểu số thực để lưu hệ số của phương trình do người dùng nhập tùy ý từ bàn phím. Nhập a, b .
- Giải thuật: Xét đủ 2 trường hợp $a = 0$ và $a \neq 0$.

$a=0 \rightarrow$ phương trình có dạng $0x+b=0$ chúng ta phải xét tiếp trường hợp của b

$b=0 ???$

$b \neq 0????$



Câu 5: Giải và biện luận phương trình bậc 2 $ax^2 + bx + c = 0$.

Hướng dẫn:

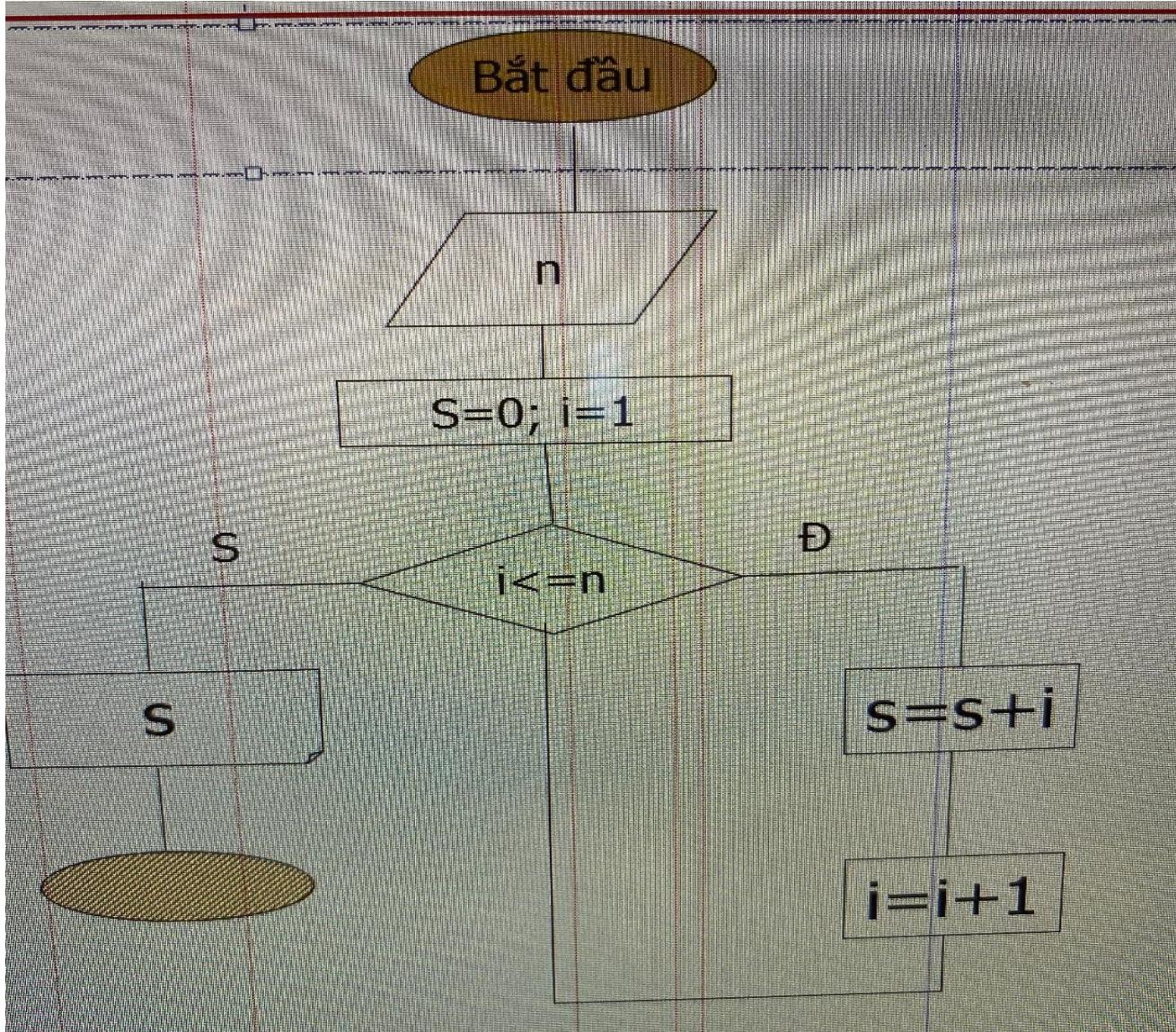
- Khai báo 3 biến a, b, c kiểu số thực. Nhập các hệ số a, b, c .
- Xét $a = 0$. Phương trình trở thành bậc nhất $bx+c=0$, giải và biện luận theo b, c (tương tự bài 1).
- Xét $a \neq 0$. Tính delta, $d = b*b - 4*a*c$. Xét các trường hợp $d = 0, d < 0$ và $d > 0$.

Câu 6: Tính tổng của n số tự nhiên đầu tiên $S = 1 + 2 + 3 + \dots + n$

Hướng dẫn

- Cho người dùng nhập giá trị n
 - o ví dụ $n=5$ ta phải tính tổng của các số $1+2+3+4+5$
- Khai báo 1 biến S để lưu trữ tổng
- Lần lượt cộng dồn các số $1, 2, 3, \dots$ vào s
 - o Tức là :
 - cộng 1 vào và lưu trữ vào biến $s \rightarrow s = 1$
 - cộng 2 vào và lưu trữ vào biến $s \rightarrow s = s + 2$
 - cộng 3 vào và lưu trữ vào biến $s \rightarrow s = s + 3$

Nhận xét : ta thấy có sự lặp đi lặp lại công việc cộng dồn các số 1,2,3...vào biến S từ đây chúng ta suy ra công thức chung $S=S+i$, i chính là các số chạy từ 1 đến n. Giải thuật dừng khi $i>n$.



Câu 7: Tính tổng của n số chẵn tự nhiên đầu tiên $S = 2+ 4+....+2n$

BÀI 3: GIỚI THIỆU VỀ NGÔN NGỮ LẬP TRÌNH C

3.1 KHÁI NIỆM VỀ NGÔN NGỮ LẬP TRÌNH

3.1.1 Khái niệm ngôn ngữ lập trình

Ngôn ngữ lập trình là một ngôn ngữ dùng để viết chương trình cho máy tính. Ta có thể chia ngôn ngữ lập trình thành các loại sau: ngôn ngữ máy, hợp ngữ và ngôn ngữ cấp cao.

Ngôn ngữ máy (machine language): Là các chỉ thị dưới dạng nhị phân, can thiệp trực tiếp vào trong các mạch điện tử. Chương trình được viết bằng ngôn ngữ máy thì có thể được thực hiện ngay không cần qua bước trung gian nào. Tuy nhiên chương trình viết bằng ngôn ngữ máy dễ sai sót, công kẽm và khó đọc, khó hiểu vì toàn những con số 0 và 1.

Hợp ngữ (assembly language): Bao gồm tên các câu lệnh và quy tắc viết các câu lệnh đó. Tên các câu lệnh bao gồm hai phần: phần mã lệnh (viết tựa tiếng Anh) chỉ phép toán cần thực hiện và địa chỉ chứa toán hạng của phép toán đó.

Ví dụ:

INPUT a ; Nhập giá trị cho a từ bàn phím LOAD a ; Đọc giá trị a vào thanh ghi tổng A PRINT a; Hiển thị giá trị của a ra màn hình. INPUT b

ADD b; Cộng giá trị của thanh ghi tổng A với giá trị b

Trong các lệnh trên thì INPUT, LOAD, PRINT, ADD là các mã lệnh còn a, b là địa chỉ. Để máy thực hiện được một chương trình viết bằng hợp ngữ thì chương trình đó phải được dịch sang ngôn ngữ máy. Công cụ thực hiện việc dịch đó được gọi là Assembler.

Ngôn ngữ cấp cao (High level language): Ra đời và phát triển nhằm phản ánh cách thức người lập trình nghĩ và làm. Rất gần với ngôn ngữ con người (Anh ngữ) nhưng chính xác như ngôn ngữ toán học. Cùng với sự phát triển của các thế hệ máy tính, ngôn ngữ lập trình cấp cao cũng được phát triển rất đa dạng và phong phú, việc lập trình cho máy tính vì thế mà cũng có nhiều khuynh hướng khác nhau: lập trình cấu trúc, lập trình hướng đối tượng, lập trình logic, lập trình hàm... Một chương trình viết bằng ngôn ngữ cấp cao được gọi là chương trình nguồn (source programs). Để máy tính "hiểu" và thực hiện được các lệnh trong chương trình nguồn thì phải có một chương trình dịch để dịch chương trình nguồn (viết bằng ngôn ngữ cấp cao) thành dạng chương trình có khả năng thực thi.

3.1.2 Chương trình dịch

Như trên đã trình bày, muốn chuyển từ chương trình nguồn sang chương trình đích phải có chương trình dịch. Thông thường mỗi một ngôn ngữ cấp cao đều có một chương trình dịch riêng nhưng chung quy lại thì có hai cách dịch: Thông dịch và biên dịch

Thông dịch (interpreter): Là cách dịch từng lệnh một, dịch tới đâu thực hiện tới đó. Chẳng hạn ngôn ngữ LISP sử dụng trình thông dịch.

Biên dịch (compiler): Dịch toàn bộ chương trình nguồn thành chương trình đích rồi sau đó mới thực hiện. Các ngôn ngữ sử dụng trình biên dịch như Pascal, C...

Giữa thông dịch và biên dịch có khác nhau ở chỗ: Do thông dịch là vừa dịch vừa thực thi chương trình còn biên dịch là dịch xong toàn bộ chương trình rồi mới thực thi nên chương trình viết bằng ngôn ngữ biên dịch thực hiện nhanh hơn chương trình viết bằng ngôn ngữ thông dịch.

Một số ngôn ngữ sử dụng kết hợp giữa thông dịch và biên dịch chẳng hạn như Java. Chương trình nguồn của Java được biên dịch tạo thành một chương trình đối tượng (một dạng mã trung gian) và khi thực hiện thì từng lệnh trong chương trình đối tượng được thông dịch thành mã máy.

3.1.3 Ngôn ngữ lập trình C

C là ngôn ngữ lập trình cấp cao, được sử dụng rất phổ biến để lập trình hệ thống cùng với Assembler và phát triển các ứng dụng.

Vào những năm cuối thập kỷ 60 đầu thập kỷ 70 của thế kỷ XX, Dennis Ritchie (làm việc tại phòng thí nghiệm Bell) đã phát triển ngôn ngữ lập trình C dựa trên ngôn

ngữ BCPL (do Martin Richards đưa ra vào năm 1967) và ngôn ngữ B (do Ken Thompson phát triển từ ngôn ngữ BCPL vào năm 1970 khi viết hệ điều hành UNIX đầu tiên trên máy PDP-7) và được cài đặt lần đầu tiên trên hệ điều hành UNIX của máy DEC PDP-11.

Năm 1978, Dennis Ritchie và B.W Kernighan đã cho xuất bản quyển "Ngôn ngữ lập trình C" và được phổ biến rộng rãi đến nay.

Lúc ban đầu, C được thiết kế nhằm lập trình trong môi trường của hệ điều hành Unix nhằm mục đích hỗ trợ cho các công việc lập trình phức tạp. Nhưng về sau, với những nhu cầu phát triển ngày một tăng của công việc lập trình, C đã vượt qua khuôn khổ của phòng thí nghiệm Bell và nhanh chóng hội nhập vào thế giới lập trình để rồi các công ty lập trình sử dụng một cách rộng rãi. Sau đó, các công ty sản xuất phần mềm lần lượt đưa ra các phiên bản hỗ trợ cho việc lập trình bằng ngôn ngữ C và chuẩn ANSI C cũng được khai sinh từ đó.

Ngôn ngữ lập trình C là một ngôn ngữ lập trình hệ thống rất mạnh và rất "mềm dẻo", có một thư viện gồm rất nhiều các hàm (function) đã được tạo sẵn. Người lập trình có thể tận dụng các hàm này để giải quyết các bài toán mà không cần phải tạo mới. Hơn thế nữa, ngôn ngữ C hỗ trợ rất nhiều phép toán nên phù hợp cho việc giải quyết các bài toán kỹ thuật có nhiều công thức phức tạp. Ngoài ra, C cũng cho phép người lập trình tự định nghĩa thêm các kiểu dữ liệu trừu tượng khác. Tuy nhiên, điều mà người mới vừa học lập trình C thường gặp "rắc rối" là "hơi khó hiểu" do sự "mềm dẻo" của C. Dù vậy, C được phổ biến khá rộng rãi và đã trở thành một công cụ lập trình khá mạnh, được sử dụng như là một ngôn ngữ lập trình chủ yếu trong việc xây dựng những phần mềm hiện nay.

Những đặc điểm cơ bản của ngôn ngữ C

- Tính cô đọng (compact):** C chỉ có 32 từ khóa chuẩn và 40 toán tử chuẩn, nhưng hầu hết đều được biểu diễn bằng những chuỗi ký tự ngắn gọn.
- Tính cấu trúc (structured):** C có một tập hợp những chỉ thị của lập trình như cấu trúc lựa chọn, lặp... Từ đó các chương trình viết bằng C được tổ chức rõ ràng, dễ hiểu.
- Tính tương thích (compatible):** C có bộ tiền xử lý và một thư viện chuẩn vô cùng phong phú nên khi chuyển từ máy tính này sang máy tính khác các chương trình viết bằng C vẫn hoàn toàn tương thích.

4. Tính linh động (flexible): C là một ngôn ngữ rất uyển chuyển và cú pháp, chấp nhận nhiều cách thể hiện, có thể thu gọn kích thước của các mã lệnh làm chương trình chạy nhanh hơn.

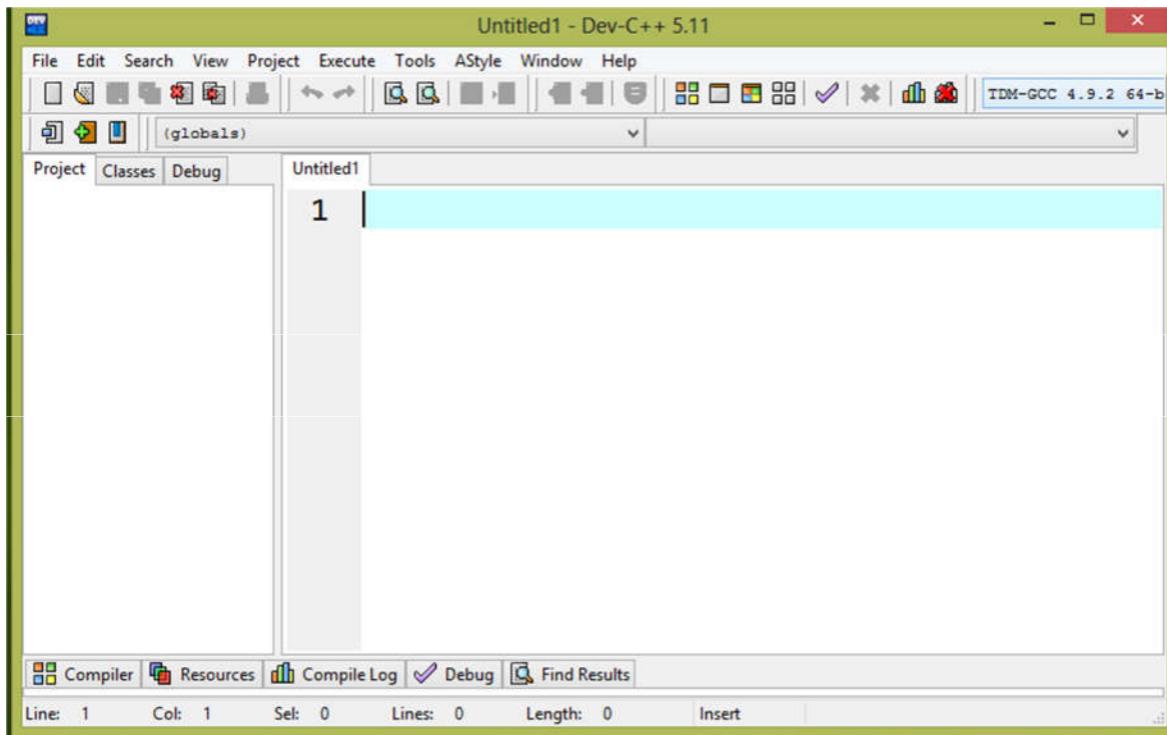
5. Biên dịch (compile): C cho phép biên dịch nhiều tập tin chương trình riêng rẽ thành các tập tin đối tượng (object) và liên kết (link) các đối tượng đó lại với nhau thành một chương trình có thể thực thi được (executable) thống nhất.

3.2 MÔI TRƯỜNG DEV-C

Dev-C là phần mềm cơ bản để học lập trình C. Ngoài ra còn phần mềm khác có thể sử dụng là Turbo C, Borland C, mặc dù tương đối “thân thuộc” với những người đã từng học Pascal (vì cùng họ “Turbo”); tuy nhiên, chương trình này trên giao diện Console không thân thiện cho lăm và đồng thời sẽ gặp trục trặc với những những hệ điều hành 64-bit. Chương trình Dev-C++ hỗ trợ đầy đủ các chức năng như: soạn thảo chương trình, dịch, thực thi chương trình... Phiên bản được sử dụng ở đây là Dev-C 5.11.

❖ Mở Dev-C

Chạy Dev-C cũng giống như chạy các chương trình khác trong môi trường Windows, màn hình sẽ xuất hiện cửa sổ Dev-C như sau:



Dòng trên cùng gọi là thanh menu (**menu bar**). Mỗi mục trên thanh menu lại có thể có nhiều mục con nằm trong một menu kéo xuống.

❖ Soạn thảo chương trình mới

Muốn soạn thảo một chương trình mới ta chọn **File → New → Source File**. Trên màn hình sẽ xuất hiện một vùng trống để cho ta soạn thảo nội dung của chương trình. Trong quá trình soạn thảo chương trình ta có thể sử dụng các phím sau:

❖ Quy tắc đặt tên tập tin của ngôn ngữ C:

Tên của tập tin gồm 2 phần: Phần tên và phần mở rộng.

- **Phần tên** của tập tin phải bắt đầu là 1 ký tự từ a..z (không phân biệt hoa thường), theo sau có thể là các ký tự từ a..z, các ký số từ 0..9 hay dấu gạch dưới (_), phần này dài tối đa là 8 ký tự.
- **Phần mở rộng: .CPP** phần này dài tối đa 3 ký tự.

Ví dụ: bai_tap.cpp

❖ Thực hiện chương trình:

Sau khi chúng ta viết hoàn thiện một chương trình, nếu muốn chạy thử và xem kết quả của chương trình hãy nhấn tổ hợp phím **F11**.

3.3 MỘT SỐ CHƯƠNG TRÌNH C ĐƠN GIẢN

Ví dụ 1 : Viết chương trình xuất chuỗi " Xin chào bạn " ra màn hình .

Mở C-free/Dev-C, vào File/New/Source file.

Gõ đoạn code sau và lưu file với phần mở rộng là .cpp (Ví dụ: Cau1.cpp)

```
#include<stdio.h> //Khai báo thư viện stdio.h vì nó chứa hàm printf.  
int main()  
{  
    printf(" Xin chao ban !");  
    printf(" Xin chao ban!\n");  
    printf(" Chao ban !\n Toi ten la Hoa ");  
    printf(" Chao ban !\t Toi ten la Hoa !\n ");  
    return 0;  
}
```

Hãy biên dịch và chạy chương trình, xem kết quả hiện ra màn hình và rút ra nhận xét.

Phím tắt để biên dịch và chạy chương trình (Compile and Run):

C-free: F5

Dev-C: F11

Hướng dẫn: Tìm hiểu cách xuất chuỗi, \t (tab), \n (xuống dòng) trong C .

Ví dụ 2 : Viết chương trình thực hiện:

- Nhập một số nguyên. Xuất ra màn hình số nguyên vừa nhập.
- Nhập một số thực. Xuất ra màn hình số thực vừa nhập.
- Nhập một kí tự. Xuất ra màn hình kí tự vừa nhập.
- Nhập hai số nguyên. Hãy tính tổng, hiệu, tích, thương của hai số đó và xuất kết quả ra màn hình.

```
#include<stdio.h>
int main()
{
    //câu a
    int a;
    printf("Moi nhap 1 so nguyen: "); scanf("%d", &a);
    printf("So nguyen da nhap la %d", a);
    //câu b
    float b;
    printf("Moi nhap 1 so thuc : "); scanf("%f", &b);
    printf("So thuc da nhap la %f", b);
    //câu c
    char z;
    printf("Moi nhap 1 kí tự: ");
    fflush(stdin); //xoá bộ nhớ đệm
    scanf("%c", &z);
    printf("Ki tu da nhap la %c", z);
    //câu d
    int x, y, z;
    printf("Moi nhap hai so nguyen de cong: ");
    scanf("%d%d", &x, &y);
    z = x + y;
    printf("Tong hai so la %d\n", z);
```

```

//hoặc printf("Tong hai so la %d\n", x+y);
/*tương tự với phép hiệu, tích, thương. Lưu ý kiểu dữ liệu kết quả của phép
chia */
return 0;
}

```

Ví dụ 3 :Viết chương trình nhập và một số nguyên, cho biết nó là số âm hay số dương .

Ví dụ 4 :Viết chương trình nhập và hai số nguyên, cho biết hai số đó cùng dấu hay khác dấu .

CÂU HỎI ÔN TẬP

Mục đích yêu cầu

- * Làm quen và nắm vững các cách mô tả giải thuật; từ đó đứng trước một bài toán cụ thể, sinh viên có thể mô tả thật chi tiết các bước để giải quyết vấn đề.

Lưu ý: Sinh viên cần xác định:

Input (dữ liệu nhập).

Output (dữ liệu xuất).

Tìm thuật toán phù hợp.

- * Làm quen với phần mềm C, biết thao tác và cài đặt một số bài toán đơn giản trong C

Câu 1: Làm quen với cấu trúc chung của một chương trình C đơn giản:

Mở C-free/Dev-C, vào File/New/Source file.

Gõ đoạn code sau và lưu file với phần mở rộng là .cpp (Ví dụ: Cau1.cpp)
//Khai báo thư viện stdio.h vì nó chứa hàm printf.

```

#include<stdio.h>
int main()
{
    printf("Hello, World!");
    printf("Hello, World!\n");
}

```

```
    printf("Hello, \nWorld!\n");
    printf("Hello, \tWorld!\n");
    return 0;
}
```

Hãy biên dịch và chạy chương trình, xem kết quả hiện ra màn hình và rút ra nhận xét.

Phím tắt để biên dịch và chạy chương trình (Compile and Run):

- * C-free: F5
- * Dev-C: F11

Câu 2: Viết chương trình in lên màn hình một thiệp mời dự sinh nhật có dạng:

```
*****
```

THIEP MOI

Than moi ban: "Le Loi"

Toi du le sinh nhat cua minh

Vao luc 19h ngay 20/10/2016

Tai: 05/42 Vinh Vien – TP. HCM

Rat mong duoc don tiep!

Ho Le Thu

```
*****
```

- Hướng dẫn: Áp dụng \t (tab), \n (xuống dòng), \" (in ra dấu ")

Câu 3: Viết chương trình thực hiện:

- Nhập một số nguyên. Xuất ra màn hình số nguyên vừa nhập.
- Nhập một số thực. Xuất ra màn hình số thực vừa nhập.
- Nhập một kí tự. Xuất ra màn hình kí tự vừa nhập.
- Nhập hai số nguyên. Hãy tính tổng, hiệu, tích, thương của hai số đó và xuất kết quả ra màn hình.

Câu 4: Viết chương trình nhập vào bán kính r của một hình tròn. Tính chu vi và diện tích của hình tròn. In các kết quả lên màn hình.

Hướng dẫn:

Khai báo biến r để lưu trữ bán kính của hình tròn

- Khai báo biến: **Kiểu_dữ_liệu Tên_bien;** VD: float r;
Khai báo hằng số PI = 3.14
 - Cách 1: **#define Tên_hằng Giá_trị** VD: #define MAX 100
- 1.** Cách 2: **const Kiểu_dữ_liệu Tên_hằng=Giá trị;**
- VD: const int MAX=100;
Diện tích hình tròn: PI*r*r
Chu vi hình tròn: 2*PI*r.

Câu 5: Viết chương trình thực hiện:

- Nhập vào hai số nguyên. Xuất ra màn hình giá trị lớn nhất.
- Nhập vào ba số nguyên. Xuất ra màn hình giá trị lớn nhất.

Hướng dẫn:

- Nhập vào hai số nguyên. Xuất ra màn hình giá trị lớn nhất.
- Khai báo hai biến *a*, *b* để lưu hai số nguyên.
- Thông báo và cho người dùng nhập 2 số nguyên *a* và *b* (dùng printf, scanf).
- Sử dụng một biến *max* để lưu giá trị lớn nhất trong hai số, khai báo *max*.
- So sánh hai số *a* và *b*, nếu số nào lớn hơn thì gán *max* = số đó. - Xuất ra màn hình giá trị lớn nhất tìm được: xuất giá trị biến *max*
- Nhập vào ba số nguyên. Xuất ra màn hình giá trị lớn nhất.
- Tương tự, khai báo 3 biến *a*, *b*, *c* và *max* lưu giá trị lớn nhất.
- Giả sử *a* là số lớn nhất, tức *max* = *a*.
- Lần lượt so sánh hai số còn lại *b*, *c* với *max*, nếu số nào lớn hơn *max* thì cập nhật *max* = số đó như sau: Nếu *b*>*max* thì *max*=*b*. Nếu *c*>*max* thì *max*=*c*.
- Xuất ra màn hình giá trị lớn nhất tìm được: xuất giá trị biến *max*.

BÀI TẬP NÂNG CAO

Câu 6: Nhập vào 3 số nguyên dương a, b, c . Kiểm tra xem 3 số đó có lập thành tam giác không? Nếu có hãy tính chu vi và diện tích của tam giác theo công thức:

- Chu vi CV = $a+b+c$.
- Diện tích S = $\sqrt{p*(p-a)*(p-b)*(p-c)}$, trong đó: $p = \frac{CV}{2}$. Xuất các kết quả ra màn hình.

Hướng dẫn:

- a, b, c là số nguyên dương \Rightarrow khai báo a, b, c kiểu *unsigned int*.
 - Điều kiện để 3 số lập thành tam giác: tổng 2 cạnh phải lớn hơn cạnh còn lại. Vậy: a, b, c lập thành 3 cạnh của tam giác khi và chỉ khi $a+b>c$ và $a+c>b$ và $b+c>a$.
 - Tính diện tích và chu vi theo công thức đã cho. Hàm căn bậc hai \sqrt{x} trong thư viện *<math.h>*.

Câu 7: Viết chương trình đảo ngược một số nguyên dương có đúng 3 chữ số.

VD: Nhập vào n=234 \rightarrow In ra: 432

Hướng dẫn:

6. Lần lượt lấy các chữ số (sử dụng phép chia / và phép chia lấy phần dư %) và in ra màn hình theo thứ tự:

Chữ số hàng đơn vị

Chữ số hàng chục

Chữ số hàng trăm.

BÀI 4: KIỂU DỮ LIỆU VÀ BIẾU THỨC TRONG C

Học xong bài này, sinh viên sẽ nắm rõ các vấn đề sau:

- Các ký tự và từ khóa dùng trong C.
- Cách đặt tên trong C
- Khai báo và sử dụng các thư viện trong C
- Các kiểu dữ liệu chuẩn (*int, long, char, float...*).
- Các phép toán và các hàm chuẩn của ngôn ngữ lập trình C.
- Câu lệnh là gì? Cách sử dụng câu lệnh gán giá trị của một biểu thức cho một biến.
- Cách sử dụng lệnh *scanf* để nhập giá trị cho biến.
- Cách sử dụng lệnh *printf* để xuất giá trị của biểu thức lên màn hình.
- Thực hiện viết các chương trình hoàn chỉnh sử dụng các lệnh đơn giản và các kiểu dữ liệu chuẩn đó.

4.1 KIỂU DỮ LIỆU CƠ BẢN, BIẾN, HÃNG

4.1.1 Các ký tự dùng trong ngôn ngữ C

Mọi ngôn ngữ lập trình đều được xây dựng từ một tập ký tự nào đó. Các ký tự này được ghép với nhau để tạo thành các từ ; Sau đó các từ được liên kết lại theo một qui tắc nào đó để tạo thành một câu lệnh. Một chương trình bao gồm nhiều câu lệnh và được diễn đạt theo một thuật toán nào đó nhằm để giải quyết một bài toán cụ thể nào đó.

Tập ký tự trong ngôn ngữ C bao gồm những ký tự, ký hiệu sau: (*phân biệt chữ in hoa và in thường*):

- 26 chữ cái latinh lớn A,B,C...Z
- 26 chữ cái latinh nhỏ a,b,c...z.
- 10 chữ số thập phân 0,1,2...9.
- Các ký hiệu toán học: +, -, *, /, =, <, >, (,)
- Các ký hiệu đặc biệt: ., ; " ' _ @ # \$! ^ [] { } ...
- Dấu cách hay khoảng trắng.

4.1.2 Các từ khóa trong C

Từ khóa được sử dụng để khai báo các kiểu dữ liệu, viết các toán tử và các câu lệnh.

Asm	Auto	break	case	Cdecl	char
Class	Const	continue	_cs	Default	delete
Do	Double	_ds	else	Enum	_es
Extern	_export	far	_fastcall	Float	for
Friend	Goto	huge	if	Inline	int
Interrupt	_loadds	long	near	New	operator
Pascal	Private	protected	public	register	return
_saveregs	_seg	short	signed	Sizeof	_ss
Static	Struct	switch	template	This	typedef
Union	unsigned	virtual	void	Volatile	while

Ví dụ 1: Có thể định nghĩa các kiểu riêng bằng từ khóa **typedef**.

```
#define TRUE 1
```

```
#define FALSE 0
```

```
typedef int boolean;
```

Chú ý:

- Không được sử dụng từ khóa để đặt tên cho các hằng, biến, mảng, hàm, ...
- Các từ khóa phải được viết bằng những ký tự thường.

4.1.3 Các kiểu dữ liệu cơ bản

❖ Kiểu kí tự

Kiểu char chiếm 1 byte và được biểu diễn thông qua bảng mã ASCII. Có hai kiểu char:

Kiểu dữ liệu	Miền giá trị (Domain)
unsigned char	Từ 0 đến 255 (tương đương 256 ký tự trong bảng mã ASCII)
Char	Từ - 128 đến 127

❖ Kiểu số nguyên

Kiểu dữ liệu	Kích thước	Miền giá trị (Domain)
char	1 byte	-128 đến 127 hoặc 0 đến 255
unsigned char	1 byte	0 đến 255
signed char	1 byte	-128 đến 127
int	2 bytes	-32,768 đến 32,767
unsigned int	2 bytes	0 đến 65,535
long	4 bytes	-2,147,483,648 đến 2,147,483,647
unsigned long	4 bytes	0 đến 4,294,967,295

❖ **Kiểu số thực:** Kiểu số thực dùng để lưu các số thực hay các số có dấu chấm thập phân gồm có 3 kiểu sau:

Kiểu dữ liệu	Kích thước (Size)	Miền giá trị (Domain)
Float	4 bytes	Từ $3.4 * 10^{-38}$ đến $3.4 * 10^{38}$
Double	8 bytes	Từ $1.7 * 10^{-308}$ đến $1.7 * 10^{308}$
long double	10 bytes	Từ $3.4 * 10^{-4932}$ đến $1.1 * 10^{4932}$

Ngoài ra ta còn có kiểu dữ liệu **void**, kiểu này mang ý nghĩa là kiểu rỗng không chứa giá trị gì cả.

4.1.4 Hằng (Constant)

Là đại lượng không đổi trong suốt quá trình thực thi của chương trình.

Hằng có thể là một chuỗi ký tự, một ký tự, một con số xác định. Chúng có thể được biểu diễn hay định dạng (Format) với nhiều dạng thức khác nhau.

❖ Khai báo hằng

Cú pháp: #define <ten hang> <gia tri hang>

Ví dụ:

```
#define MAX 253 /* khai báo một hằng MAX có giá trị là 253 */
#define PI 3.141593 /* khai báo một hằng PI có giá trị là 3.141593 */
```

❖ Hằng số thực

Số thực bao gồm các giá trị kiểu float, double, long double được thể hiện theo 2 cách sau:

- **Cách 1:** (dạng thập phân): số thực gồm 2 phần là: phần nguyên và phần dấu chấm thập phân. Ví dụ: 152.25 -3.726 275.0
- **Cách 2:** số thực gồm 2 phần là: phần định trị và phần mũ. Phần định trị là một số nguyên hoặc là số thực dạng thập phân, còn phần mũ là một số nguyên. Hai phần này cách nhau bởi chữ E hoặc chữ e.

Ví dụ:

183.21E-5	có giá trị là 183.21×10^{-5}
0.24e+6	có giá trị là 0.24×10^6
-32.7E-2	có giá trị là -32.7×10^{-2}

21e3

có giá trị là 21×10^3

❖ Hằng số nguyên

Hằng số nguyên (2 byte) hệ thập phân:

Ví dụ: 546 -27 5021

Hằng số nguyên (4 byte) long: là số nguyên được viết thêm ký tự L hoặc l vào tận cùng bên phải

Ví dụ: 564L -27456l

- **Chú ý:** nếu một hằng số nguyên vượt quá miền giá trị của int thì cũng được xem là hằng long.

Ví dụ: 123456L và 123456 là hai hằng long có cùng 1 giá trị.

❖ Hằng số nguyên hệ 8 (bát phân):

Cách viết: 0c1c2c3... Trong đó: ci là các ký tự số từ 0 đến 7.

Ví dụ: 05345 0721 là các hằng nguyên int hệ 8.

❖ Hằng số nguyên hệ 16 (thập lục phân):

Cách viết: 0xc1c2c3... hoặc 0Xc1c2c3...

Trong đó: ci là các ký tự từ 0 đến 9, và từ A đến F (hoặc từ a đến f)

Ví dụ: 0x79 0X5A1 0X129 0x6c8

là các hằng số nguyên hệ 16.

❖ Hằng ký tự

Hằng ký tự là một ký tự riêng biệt được viết trong cặp dấu nháy đơn (''). Mỗi một ký tự tương ứng với một giá trị trong bảng mã ASCII. Hằng ký tự cũng được xem như trị số nguyên.

Ví dụ: 'a', 'A', '0', '9'

Chúng ta có thể thực hiện các phép toán số học trên 2 ký tự (thực chất là thực hiện phép toán trên giá trị ASCII của chúng)

❖ Hằng chuỗi ký tự

Hằng chuỗi ký tự là một chuỗi hay một xâu ký tự được đặt trong cặp dấu nháy kép ("").

Ví dụ: "Ngon ngu lap trinh C", "Khoa CNTT-DHCT", "NVLinh-DVHieu"

Chú ý:

1. Một chuỗi không có nội dung "" được gọi là chuỗi rỗng.
2. Khi lưu trữ trong bộ nhớ, một chuỗi được kết thúc bằng ký tự NULL ('\0': mã ASCII là 0).
3. Cần phân biệt 'A' và "A". 'A' là một ký tự và được lưu trữ trong vùng nhớ 1 byte, còn "A" là một chuỗi ký tự và được lưu trữ trong vùng nhớ 2 byte.
4. Để biểu diễn ký tự đặc biệt bên trong chuỗi ta phải thêm dấu \ phía trước

Ví dụ: " I'm a Teacher " ta phải viết " I \ 'm a Teacher ".

" đây là " **ngôn ngữ** "của" thì phải viết" đây là \ " **ngôn ngữ** \ " của "

4.1.5 Biến

Biến là một đại lượng được người lập trình định nghĩa và được đặt tên thông qua việc khai báo biến. Biến dùng để chứa dữ liệu trong quá trình thực hiện chương trình và giá trị của biến có thể bị thay đổi trong quá trình này. Cách đặt tên biến giống như cách đặt tên đã nói trong phần trên.

Mỗi biến thuộc về một kiểu dữ liệu xác định và có giá trị thuộc kiểu đó.

Biến được xem như là một vùng nhớ được đặt tên. Mọi biến trước khi sử dụng cần phải được khai báo.

❖ Khai báo biến

Cú pháp: <kiểu dữ liệu> <tên biến>;

Ví dụ: int a; /* khai báo biến a có kiểu dữ liệu là int */
float x; /* khai báo biến x có kiểu dữ liệu là float */
double y, z; /* khai báo 2 biến y, z có kiểu dữ liệu là double */
long m, n, i; /* khai báo 3 biến m, n, i có kiểu dữ liệu là long */

Lưu ý: Để kết thúc 1 lệnh phải có dấu chấm phẩy (;) ở cuối lệnh.

❖ Vị trí khai báo biến trong C

Trong ngôn ngữ lập trình C, ta phải khai báo biến đúng vị trí. Nếu khai báo (đặt các biến) không đúng vị trí sẽ dẫn đến những sai sót ngoài ý muốn mà người lập trình không lường trước (hiệu ứng lề). Chúng ta có 2 cách đặt vị trí của biến như sau:

❖ Khai báo biến ngoài

Các biến này được đặt bên ngoài tất cả các hàm và nó có tác dụng hay ảnh hưởng đến toàn bộ chương trình (còn gọi là biến toàn cục).

Ví dụ:

```
int i;      /*Bien ben ngoai */  
float pi;    /*Bien ben ngoai*/  
int main()  
{ ... }
```

❖ Khai báo biến trong

Các biến được đặt ở bên trong hàm, chương trình chính hay một khối lệnh. Các biến này chỉ có tác dụng hay ảnh hưởng đến hàm, chương trình hay khối lệnh chứa nó. Khi khai báo biến, phải đặt các **biến này ở đầu của khối lệnh**, trước các lệnh gán, ...

Ví dụ 1:

```
#include <stdio.h>  
#include<conio.h>  
int bienngoai;        /*khai bao bien ngoai*/  
int main ()  
{  
    int j,i ; /*khai bao bien ben trong chuong trinh chinh*/  
    clrscr() ; /* lệnh xóa màn hình */  
    i=1; j=2;  
    bienngoai=3;  
    printf ("\n Gia tri cua i la : %d ", i);  
    printf ("\n Gia tri cua bienngoai la %d ", bienngoai);  
    ;  
    return 0;  
}
```

Ví dụ 2:

```
#include <stdio.h>
#include<conio.h>
int      main ()
{
    int i, j; /*Bien ben trong*/
    i=4;
    j=5;
    printf("\n Gia tri cua i la %d",i);
    printf("\n Gia tri cua j la %d",j);
    if(j>i)
    {
        int hieu = j-i ; /* biến bên trong */
        printf (" hiệu số của j trừ i là: %d ", hieu);
    }
    else
    {
        int hieu = i - j ; /* biến bên trong */
        printf (" hiệu số của i trừ j là: %d ", hieu);
    }
    return 0 ;
}
```

❖ Khởi gán giá trị ban đầu cho biến:

Ví dụ: int a=10; /* khai báo biến a có kiểu int và được khởi gán giá trị ban đầu bằng 10 */

float x, y=3.1, z; /* khai báo 3 biến x, y, z có kiểu float và biến y được khởi gán giá trị ban đầu bằng 3.1 */

❖ Lấy địa chỉ của biến: để lấy địa chỉ của biến ta dùng phép toán &. Ví

dụ: float x ;

Biểu thức **&x** trả về địa chỉ của biến x.

4.2 CÁCH ĐẶT TÊN TRONG C

Tên được dùng để xác định các thành phần khác nhau trong một chương trình. Chúng ta có các tên như: tên hằng, tên biến, tên mảng, tên cấu trúc, ...

4.2.1 Qui tắc đặt tên trong C

Tên là một dãy các ký tự chữ, số và dấu gạch nối dưới, nhưng phải bắt đầu bằng một chữ cái hoặc dấu gạch nối dưới và không được trùng với từ khóa.

Ví dụ 2: Các tên đúng

Bien_dem

X

_a1

Delta

pt_bac_2

Ví dụ 3: Các tên sai

5Bien bắt đầu bằng ký số 5

n! sử dụng ký tự chấm than !

while trùng với từ khóa

f(x) sử dụng ký tự ngoặc tròn ()

del ta sử dụng ký tự trắng

Chú ý:

- Trong ngôn ngữ C tên có phân biệt chữ thường và chữ hoa. **Ví dụ** tên **AB** khác với tên **ab**, tên **Delta** khác với tên **delta**.
- Tên do người sử dụng đặt ra nên phải được đặt sao cho chúng có ý nghĩa với đối tượng mà chúng hiển thị.

4.2.2 Cặp dấu ghi chú thích

Khi viết chương trình đôi lúc ta cần phải có vài lời ghi chú về một đoạn chương trình nào đó để dễ nhớ và dễ điều chỉnh sau này. phần nội dung ghi chú này phải không thuộc về chương trình (khi biên dịch phần này bị bỏ qua). Trong ngôn ngữ lập trình C, nội dung chú thích phải được viết trong cặp dấu /* và */ hoặc //.

Ví dụ 4: Viết chương trình cho xuất ra màn hình chuỗi ký tự "Hello, World !".

```
/* Chú thích trên nhiều dòng
Đây là chương trình minh họa
*/
#include "stdio.h" //chú thích trên 1 dòng
#include "conio.h"
int main() // chương trình chính
{
    printf("\nHello, World!"); // xuất chuỗi Hello, World! Ra màn hình
    return 0;
}
```

4.3 BIỂU THỨC, PHÉP TOÁN, CHUYỂN ĐỔI KIỂU DỮ LIỆU

4.3.1 Khái niệm biểu thức

Biểu thức là một sự kết hợp giữa các toán tử (operator) và các toán hạng (operand) theo đúng một trật tự nhất định, dùng để diễn đạt một công thức toán học nào đó.

Phép toán (toán tử) : +, -, *, /, ...
Toán hạng: hằng, biến, ...

Trong biểu thức ta có thể dùng các dấu ngoặc tròn để thể hiện đúng trình tự thực hiện của các phép toán.

Ví dụ:

Biểu thức tính tổng của n số tự nhiên đầu tiên: $n*(n+1)/2$;

Biểu thức tính nửa chu vi của một tam giác: $p=(a+b+c)/2$;

Biểu thức tính diện tích của một tam giác: $s=\sqrt{p*(p-a)*(p-b)*(p-c)}$;

- Mỗi biểu thức có một giá trị duy nhất.
- Hằng, biến cũng được xem là một biểu thức.

4.3.2 Các phép toán số học

❖ Phép toán 2 ngôi

Phép toán	Ý nghĩa
+	Phép cộng
-	Phép trừ
*	Phép nhân
/	Phép chia
%	Lấy phần dư

Phép toán % chỉ áp dụng cho số nguyên, nó không sử dụng được cho các số thực. Ví

dụ: biểu thức $25\%3$ có giá trị là 1

❖ Phép toán 1 ngôi

- (số âm). Ví

dụ: -3

❖ Thứ tự thực hiện các phép toán

1. Các phép toán +, - có cùng độ ưu tiên và có độ ưu tiên nhỏ hơn các phép toán *, /, %
2. Các phép toán *, /, % có cùng độ ưu tiên và có độ ưu tiên nhỏ hơn phép toán một ngôi -
3. Các phép toán số học có cùng độ ưu tiên thì được thực hiện theo thứ tự từ trái sang phải.

4.3.3 Phép gán (lệnh gán)

Lệnh gán (assignment statement) dùng để gán giá trị của một biểu thức cho một biến.

Cú pháp: < tên biến > = < biểu thức >;

```

Ví dụ:    int main ()
{
    int x,y;
    x = 5 ;      //Gán giá trị 5 cho biến x
    y = 2*x; //Gán giá trị 2*x = 2*5 =10 cho biến y
    return 0;
}

```

Nguyên tắc khi dùng lệnh gán thì kiểu của biến và kiểu của biểu thức phải giống nhau, gọi là có sự tương thích giữa các kiểu dữ liệu. Chẳng hạn ví dụ sau cho thấy một sự không tương thích về kiểu:

```

int main()
{
    int x, y;
    x = 10; //Gán hằng số 10 cho biến x
    y = " hoa "; //y có kiểu int, còn " hoa " có kiểu chuỗi ký tự
    return 0;
}

```

Khi biên dịch chương trình này, C sẽ báo lỗi "*Cannot convert 'char *' to 'int'*" tức là C không thể tự động chuyển đổi kiểu từ char * (chuỗi ký tự) sang int.

Chú ý:

- Khi một biểu thức được gán cho một biến thì giá trị của nó sẽ thay thế giá trị cũ mà biến đã lưu giữ trước đó.
- Trong câu lệnh gán, dấu = là một toán tử; do đó nó có thể được sử dụng là một thành phần của biểu thức. Trong trường hợp này giá trị của biểu thức gán chính là giá trị của biến.

Ví dụ:

```

int x, y;
y = x = 3;           // y lúc này cùng bằng 3

```

Ta có thể gán trị cho biến lúc biến được khai báo theo cách thức sau:

<Tên kiểu> <Tên biến> = <Biểu thức>;

Ví dụ: int x = 10, y=x;

Giả sử ta có câu lệnh gán sau: <bien> = <bien> + <bieu thuc>;

Ta có thể viết gọn lại: <bien> += <bieu thuc>;

Theo cách đó ta có các toán tử gán số học sau: +=, -=, *=, /=, %=

Ví dụ:

int i=5;

i += 3; // nghĩa là i=i+3, sau khi thực hiện câu lệnh thì i=8

float x=1.1, y=3.0;

y *= x+1;

// nghĩa là y=y*(x+1), sau khi thực hiện câu lệnh thì y=6.3 và x=1.1

4.3.4 Các phép toán quan hệ và lý luận

❖ Các phép toán quan hệ:

Ý tưởng chính của phép toán quan hệ và toán tử Logic là đúng hoặc sai. Trong C mọi giá trị khác 0 được gọi là đúng, còn sai là 0. Các biểu thức sử dụng các toán tử quan hệ và Logic trả về 0 nếu sai và trả về 1 nếu đúng.

- Các dữ liệu thuộc kiểu dữ liệu số nguyên, số thực, ký tự có thể được so sánh với nhau bằng các phép toán quan hệ.
- Kết quả của phép toán quan hệ cho ta giá trị đúng (bằng 1) hoặc giá trị sai (bằng 0).
- Các phép toán:

Phép toán	Ý nghĩa
>	Lớn hơn
>=	Lớn hơn hay bằng
<	Nhỏ hơn
<=	Nhỏ hơn hay bằng

$==$	Bằng nhau
$!=$	Khác nhau

Ví dụ:

5>7	có giá trị 0(sai)
10>=10	có giá trị 1 (đúng)
6<6	có giá trị 0(sai)
7<=10	có giá trị 1 (đúng)
12==8	có giá trị 0(sai)
25!=9	có giá trị 1 (đúng)
'A'>'F'	có giá trị 0(sai)

- a. Các phép toán $>$, \geq , $<$, \leq có cùng độ ưu tiên.
- b. Các phép toán $=$, \neq có cùng độ ưu tiên
- c. Các phép toán ở mục (1) có độ ưu tiên cao hơn các phép toán ở mục (2).
- d. Các phép toán quan hệ có độ ưu tiên thấp hơn so với các phép toán số học.

❖ **Các phép toán luận lý:**

Kết quả của phép toán luận lý cho ta giá trị đúng (bằng 1) hoặc giá trị sai (bằng 0).

Phép phủ định: (toán tử một ngôi) toán tử !

Phép và: (toán tử 2 ngôi) toán tử **&&**

Phép hoặc: (toán tử 2 ngôi) toán tử **||**

A	B	!A	A&&B	A B
Đúng (khác 0)	Đúng (khác 0)	Sai (bằng 0)	Đúng (bằng 1)	Đúng (bằng 1)
Đúng (khác 0)	Sai (bằng 0)	Sai (bằng 0)	Sai (bằng 0)	Đúng (bằng 1)
Sai (bằng 0)	Đúng (khác 0)	Đúng (bằng 1)	Sai (bằng 0)	Đúng (bằng 1)
Sai (bằng 0)	Sai (bằng 0)	Đúng (bằng 1)	Sai (bằng 0)	Sai (bằng 0)

5 && 8	có giá trị 1
! 12.5	có giá trị 0
!(9 < 2)	có giá trị 1
('A' == 'B') (7 > 3)	có giá trị 1
('A' > 'Y') && (6 >= 1)	có giá trị 0

- Các phép toán quan hệ có độ ưu tiên nhỏ hơn phép toán ! (phủ định), nhưng lớn hơn so với các phép toán && (và), || (hoặc).

❖ **Tóm tắt: thứ tự ưu tiên của các phép toán**

1. !, - (số âm)
2. *, /, %
3. +, -
4. . >, >=, <, <=
5. ==, !=
6. &&, ||

4.3.5 Chuyển đổi kiểu giá trị

❖ **Cú pháp:**

<(type)> <biểu thức>

Ý **nghĩa:** kiểu giá trị của biểu thức được đổi thành kiểu giá trị **type**

Ví dụ:

```
float x;
int n=5;
x= (float) n;          /* giá trị của biến n có kiểu int được đổi sang kiểu
                           float và gán cho biến x */
```

Phép ép kiểu có độ ưu tiên như toán tử một ngôi.

Ví dụ:

(int) 1.4*10	có giá trị 10
(int) (1.4*10)	có giá trị 14
(float) 21/6 + 8.0	có giá trị 11.5

❖ Chuyển đổi kiểu trong biểu thức:

Khi 2 toán hạng trong một phép toán có kiểu giá trị khác nhau thì kiểu giá trị thấp hơn sẽ được tự động đổi sang kiểu giá trị cao hơn trước khi thực hiện phép toán. Kết quả của phép toán là một giá trị có kiểu giá trị cao nhất.

Ví dụ: Kiểu int và long thì được đổi thành kiểu long.

Kiểu int và kiểu float thì được đổi thành kiểu float.

Kiểu float và kiểu double thì được đổi thành kiểu double.

Ví dụ: Biểu thức

11/2	có giá trị 5
1.5*(11/2)	có giá trị 7.5
1.5*11/2	có giá trị 8.25
21/6 + 8.0	có giá trị 11.0

❖ Chuyển đổi kiểu được thực hiện thông qua phép gán:

Kiểu giá trị của biểu thức bên vẽ phải được đổi sang kiểu giá trị của biến bên vẽ trái và đó cũng là kết quả của phép gán.

Kiểu float được đổi thành kiểu int bằng cách bỏ đi phần sau dấu chấm thập phân.

Kiểu double được đổi thành kiểu float bằng cách làm tròn.

Ví dụ:

```
int n;  
n = (float) 20/6 + 8.3;           biến n có giá trị 11
```

4.3.6 Phép toán tăng giảm

❖ Phép toán tăng toán hạng lên một đơn vị: ++

- Phép toán tăng sau (`a++`): `a` được tăng lên một đơn vị sau khi giá trị của nó được sử dụng.

Ví dụ 1:

```
int n, i=3;
```

```
n = i++;
```

/* giá trị của biến `i` được gán cho biến `n`, sau đó biến `i` được tăng lên một đơn vị; sau khi thực hiện câu lệnh thì `n=3` và `i=4` */

- Phép toán tăng trước (`++a`): `a` được tăng lên một đơn vị trước khi giá trị của nó được sử dụng.

Ví dụ 2:

```
int n, i=3;
```

```
n = ++i;
```

/* giá trị của biến `i` được tăng lên một đơn vị, sau đó giá trị của biến `i` được gán cho biến `n`; sau khi thực hiện câu lệnh thì `n=4` và `i=4` */

❖ Phép toán giảm toán hạng đi một đơn vị: `--` Ví

dụ 1:

```
int n, i=2;
```

```
n = i--;
```

/* giá trị của biến `i` được gán cho biến `n`, sau đó biến `i` bị giảm đi một đơn vị; sau khi thực hiện câu lệnh thì `n=2` và `i=1` */

Ví dụ 2:

```
int n, i=2;
```

```
n = --i;
```

/* giá trị của biến `i` bị giảm đi một đơn vị, sau đó giá trị của biến `i` được gán cho biến `n`; sau khi thực hiện câu lệnh thì `n=1` và `i=1` */

Chú ý: Các phép toán tăng giảm một đơn vị chỉ được sử dụng với biến, mà không sử dụng được với hằng.

4.3.7 Biểu thức điều kiện

Biểu thức điều kiện có dạng:

<biểu thức 1> ? <biểu thức 2>: <biểu thức 3>

Trong đó:

- <biểu thức 1>, <biểu thức 2>, <biểu thức 3> là các biểu thức.
- Giá trị trả về của biểu thức điều kiện bằng giá trị của <biểu thức 2> nếu giá trị của <biểu thức 1> khác 0 (đúng) và bằng giá trị của <biểu thức 3> nếu giá trị của <biểu thức 1> bằng 0 (sai).
- Kiểu trả về của biểu thức điều kiện là kiểu cao nhất trong các kiểu của <biểu thức 2> và <biểu thức 3>.

Ví dụ:

```
int a;  
  
float b;  
(a > b) ? a:b
```

Biểu thức trả về giá trị lớn nhất trong 2 giá trị a và b.

Kiểu giá trị của biểu thức điều kiện là float.

4.3.8 Các phép toán thao tác trên bit

Để thao tác trên từng bit của một số nguyên ta có thể dùng các phép toán sau:

PHÉP TOÁN	Ý NGHĨA
&	Phép và theo bit (AND)
	Phép hoặc theo bit (OR)
^	Phép hoặc loại trừ theo bit (XOR)
<<	Phép dịch trái theo bit
>>	Phép dịch phải theo bit
~	Phép lấy phần bù theo bit

Ví dụ:

$$\begin{array}{l} 1\&1=1 \\ 1\&0=0 \\ 0\&1=0 \\ 0\&0=0 \end{array}$$

$$\begin{array}{l} 1|1=1 \\ 1|0=1 \\ 0|1=1 \\ 0|0=0 \end{array}$$

$$\begin{array}{l} 1^1=0 \\ 1^0=1 \\ 0^1=1 \\ 0^0=0 \end{array}$$

```
a << n      /*dịch trái giá trị a đi n bit*/
a >> n      /*dịch phải giá trị a đi n bit*/
~1=0        /*phủ định của 1 là 0*/
~0=1        /*phủ định của 0 là 1*/
```

Chú ý:

- Các phép toán theo tác trên bit chỉ dùng cho kiểu số nguyên, chứ không dùng cho kiểu số thực (float, double).
- Các phép dịch chuyển được phân thành hai loại: Phép dịch chuyển số học và phép dịch chuyển logic.
- Phép dịch chuyển số học được thực hiện trên giá trị kiểu int, và bảo toàn bit dấu.
- Phép dịch chuyển logic được thực hiện trên giá trị kiểu unsigned.
 - Ví dụ:
 - $0x5b1c \& 0x9a = 0x18$
 - $0x5b1c | 0x9a = 0x5b9e$
 - $0x5b1c ^ 0x9a = 0x5b86$
 - $\sim 0x5b1c = 0xa4e3$
 - int k=0xffe0;
 - $k << 2 = 0xff80$
 - unsigned int m=0x5b1c;
 - $k << 3 = 0xd8e0$

4.3.9 Toán tử con trỏ & và *

- Một con trỏ là địa chỉ trong bộ nhớ của một biến. Một biến con trỏ là một biến được khai báo riêng để chứa một con trỏ đến một đối tượng của kiểu đã chỉ ra nó. Ta sẽ tìm hiểu kỹ hơn về con trỏ trong chương về con trỏ. Ở đây, chúng ta sẽ đề cập ngắn gọn đến hai toán tử được sử dụng để thao tác với các con trỏ.
- **Toán tử thứ nhất là &** là một toán tử quy ước trả về địa chỉ bộ nhớ của hệ số của nó.
 - Ví dụ: $m = \&count$
 - Đặt vào biến m địa chỉ bộ nhớ của biến count.
 - Chẳng hạn, biến count ở vị trí bộ nhớ 2000, giả sử count có giá trị là 100. Sau câu lệnh trên m sẽ nhận giá trị 2000.
 - **Toán tử thứ hai là *** là một bổ sung cho &; đây là một toán tử quy ước trả về giá trị của biến được cấp phát tại địa chỉ theo sau đó.
 - Ví dụ: $q = *m$
 - Sẽ đặt giá trị của count vào q. Bây giờ q sẽ có giá trị là 100 vì 100 được lưu trữ tại địa chỉ 2000.

4.3.10 Toán tử dấu phẩy (,)

Toán tử dấu, được sử dụng để kết hợp các biểu thức lại với nhau. Bên trái của toán tử dấu, luôn được xem là kiểu int. Điều đó có nghĩa là biểu thức bên phải trở thành giá trị của tổng các biểu thức được phân cách bởi dấu phẩy.

Ví dụ: $x = (y=3,y+1);$

Trước hết gán 3 cho y rồi gán 4 cho x. Cặp dấu ngoặc đơn là cần thiết vì toán tử dấu, có độ ưu tiên thấp hơn toán tử gán.

CÁC VÍ DỤ:

Ví dụ 1:

Cho biết giá trị của biểu thức $5.6+2.7+20/6+8.0$

Đáp số: Giá trị của biểu thức là: 19.3

Ví dụ 2:

Cho biết int $x=10, a;$

Hãy cho biết giá trị của các biến x, a sau khi thực hiện câu lệnh:

a. $a=3*x++ + 8;$

b. $a=3* ++x + 8;$

Đáp số:

Câu a: $x=11$ và $a=38$

Câu b: $x=11$ và $a=41$

Ví dụ 3:

Hãy viết một câu lệnh để gán giá trị x cho y chỉ khi x nằm trong khoảng từ 1 đến 20. Không thay đổi y nếu x không thuộc khoảng trên.

Đáp số: Câu lệnh

```
y = (x >= 1 && x <= 20) ? x: y;
```

4.4 NHẬP XUẤT DỮ LIỆU

4.4.1 Lệnh nhập giá trị từ bàn phím cho biến (hàm scanf)

Là hàm cho phép đọc dữ liệu từ bàn phím và gán cho các biến trong chương trình khi chương trình thực thi. Trong ngôn ngữ C, đó là hàm scanf nằm trong thư viện stdio.h.

Cú pháp: `scanf ("mã định dạng", địa chỉ của các biến);`

Giải thích:

Mã định dạng: dùng để qui định kiểu dữ liệu, cách biểu diễn, độ rộng, số chữ số thập phân...

Một số định dạng khi nhập kiểu số nguyên, số thực, ký tự.

Định dạng	Ý nghĩa
%d	Nhập số nguyên kiểu 2 byte
%f	Nhập số thực
%c	Nhập một ký tự
%ld	Nhập số nguyên kiểu 4 byte

Địa chỉ của các biến: là địa chỉ (&) của các biến mà chúng ta cần nhập giá trị cho nó. Được viết như sau: &<địa chỉ biến>.

Ví dụ:

```
scanf ("%d",&bien1); /*Doc gia tri cho bien1 co kieu nguyen*/
scanf("%f ",&bien2); /*Doc gia tri cho bien2 co kieu thuc*/
scanf("%d%f ",&bien1,&bien2);
/*Doc gia tri cho bien1 co kieu nguyen, bien2 co kieu thuc*/
scanf("%d%f%c ",&bien1,&bien2,&bien3); /*bien3 co kieu char*/
```

Lưu ý:

- Chuỗi định dạng phải đặt trong cặp dấu nháy kép ("").
- Các biến (địa chỉ biến) phải cách nhau bởi dấu phẩy (,).
- Có bao nhiêu biến thì phải có bấy nhiêu định dạng.
- Thứ tự của các định dạng phải phù hợp với thứ tự của các biến.
- Để nhập giá trị kiểu char được chính xác, nên dùng hàm **fflush(stdin)** để loại bỏ các ký tự còn nằm trong vùng đệm bàn phím trước hàm scanf().
- Để nhập vào một chuỗi ký tự (không chứa khoảng trắng hay *kết thúc bằng khoảng trắng*), chúng ta phải khai báo kiểu *mảng ký tự* hay *con trỏ ký tự*, sử dụng định dạng %s và *tên biến thay cho địa chỉ biến*.
- Để đọc vào một chuỗi ký tự có chứa khoảng trắng (*kết thúc bằng phím Enter*) thì *phải dùng hàm gets()*.

Ví dụ:

Khai báo biến:

```
int biennguyen;
float bienthuc;
char bienchar;
char chuoi1[20], *chuoi2;
```

Nhập giá trị cho các biến:

```
scanf ("%d",&biennguyen);
```

```

scanf ("%f ",&bienthuc);
scanf ("%d%f ",&biennghyen, &bienthuc);
scanf("%d%f%c",&biennghyen, &bienthuc,&bienchar) ;

```

4.4.2 Lệnh xuất giá trị của biểu thức lên màn hình (hàm printf)

Hàm printf (nằm trong thư viện **stdio.h**) dùng để xuất giá trị của các biểu thức lên màn hình.

Cú pháp: **printf("Chuỗi định dạng ", Các biểu thức);**

Giải thích:

- **Chuỗi định dạng:** dùng để qui định kiểu dữ liệu, cách biểu diễn, độ rộng, số chữ số thập phân...

Một số định dạng khi đối với số nguyên, số thực, ký tự.

Mã định dạng	Ý nghĩa
%d	Xuất số nguyên
%[.số chữ số thập phân] f	Xuất số thực có <số chữ số thập phân>.
%o	Xuất số nguyên hệ bát phân
%x	Xuất số nguyên hệ thập lục phân
%c	Xuất một ký tự
%s	Xuất chuỗi ký tự
%e hoặc %E hoặc %g hoặc %G	Xuất số nguyên dạng khoa học (nhân 10 mũ x)
%p	Xuất địa chỉ của biến con trả

Các biểu thức: là các biểu thức mà chúng ta cần xuất giá trị của nó lên màn hình, mỗi biểu thức phân cách nhau bởi dấu phẩy (,).

Ví dụ:

```

# include<stdio.h>

int main()
{
    int bien_nguyen=1234, i=65;

```

```
float bien_thuc=123.456703;  
  
printf("Gia tri nguyen cua bien nguyen =%d\n",bien_nguyen);  
  
printf("Gia tri thuc cua bien thuc =%f\n",bien_thuc);  
  
printf("Truoc khi lam tron=%f \n Sau khi lam tron=%f ",bien_thuc, bien_thuc);  
  
return 0;  
}
```

Kết quả in ra màn hình như sau:

```
Gia tri nguyen cua bien nguyen =1234  
Gia tri thuc cua bien thuc =123.456703  
Truoc khi lam tron=123.456703  
Sau khi lam tron=123.46
```

Nếu ta thêm vào dòng sau trong chương trình:

```
printf("\nKy tu co ma ASCII %d la %c",i,i);
```

Kết quả ta nhận được thêm:

```
Ky tu co ma ASCII 65 la A_
```

```
printf(" So nguyen la %d \n So thuc la %f ",i, (float)i);
```

```
So nguyen la 65  
So thuc la 65.000000
```

```
printf("\n So thuc la %f \n So nguyen la %d",bien_thuc, (int)bien_thuc);
```

```
So thuc la 123.456703  
So nguyen la 123_
```

```
printf("\n Viet binh thuong =%f \n Viet kieu khoa hoc=%e",bien_thuc, bien_thuc);
```

Kết quả in ra màn hình:

```
Viет binh thường=123.456703
Việt kieu khoa hoc=1.234567e+02
```

Lưu ý: Đối với các ký tự điều khiển, ta không thể sử dụng cách viết thông thường để hiển thị chúng.

Ký tự điều khiển là các ký tự dùng để điều khiển các thao tác xuất, nhập dữ liệu.

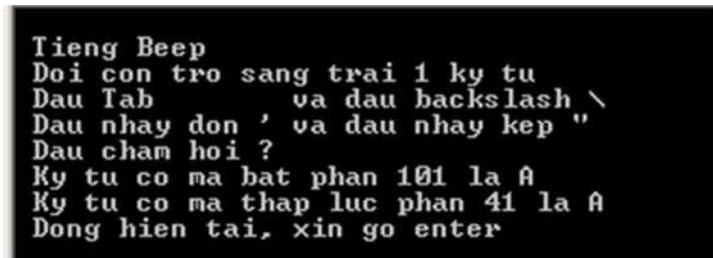
Một số ký tự điều khiển được mô tả trong bảng:

Ký tự điều khiển	Giá trị thập lục phân	Ký tự được hiển thị	Ý nghĩa
\a	0x07	BEL	Phát ra tiếng chuông
\b	0x08	BS	Di chuyển con trỏ sang trái 1 ký tự và xóa ký tự bên trái (backspace)
\f	0x0C	FF	Sang trang
\n	0x0A	LF	Xuống dòng
\r	0x0D	CR	Trở về đầu dòng
\t	0x09	HT	Tab theo cột (giống gõ phím Tab)
\\\	0x5C	\	Dấu \
\'	0x2C	'	Dấu nháy đơn (')
\"	0x22	"	Dấu nháy kép (")
\?	0x3F	?	Dấu chấm hỏi (?)
\ddd	Ddd		Ký tự có mã ACSII trong hệ bát phân là số ddd
xHHH	oxHHH		Ký tự có mã ACSII trong hệ thập lục phân là HHH

Ví dụ

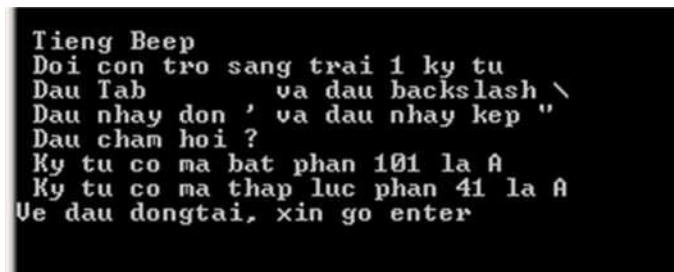
```
#include <stdio.h>
#include<conio.h>
int main ()
{
    printf("\n Tieng Beep \a");
    printf("\n Doi con tro sang trai 1 ky tu\b");
    printf("\n Dau Tab \tva dau backslash \\");
    printf("\n Dau nhay don \' va dau nhay kep '\"");
    printf("\n Dau cham hoi \?");
    printf("\n Ky tu co ma bat phan 101 la \101"); printf("\n
        Ky tu co ma thap luc phan 41 la \x041");
    printf("\n Dong hien tai, xin go enter");
    getch();
    printf("\r\nVe dau dong");
    return 0;
}
```

Kết quả trước khi gõ phím Enter:



```
Tieng Beep
Doi con tro sang trai 1 ky tu
Dau Tab      va dau backslash \
Dau nhay don ' va dau nhay kep "
Dau cham hoi ?
Ky tu co ma bat phan 101 la A
Ky tu co ma thap luc phan 41 la A
Dong hien tai, xin go enter
```

Kết quả sau khi gõ phím Enter:



```
Tieng Beep
Doi con tro sang trai 1 ky tu
Dau Tab      va dau backslash \
Dau nhay don ' va dau nhay kep "
Dau cham hoi ?
Ky tu co ma bat phan 101 la A
Ky tu co ma thap luc phan 41 la A
Ve dau dongtai, xin go enter
```

4.5 CẤU TRÚC MỘT CHƯƠNG TRÌNH C

4.5.1 Tiền xử lý và biên dịch

Trong C, việc dịch (translation) một tập tin nguồn được tiến hành trên hai bước hoàn toàn độc lập với nhau:

- Tiền xử lý.
- Biên dịch.

Hai bước này trong phần lớn thời gian được nối tiếp với nhau một cách tự động theo cách thức mà ta có ấn tượng rằng nó đã được thực hiện như là một xử lý duy nhất. Nói chung, ta thường nói đến việc tồn tại của một bộ tiền xử lý (preprocessor?) nhằm chỉ rõ chương trình thực hiện việc xử lý trước. Ngược lại, các thuật ngữ trình biên dịch hay sự biên dịch vẫn còn nhập nhằng bởi vì nó chỉ ra khi thì toàn bộ hai giai đoạn, khi thì lại là giai đoạn thứ hai.

Bước tiền xử lý tương ứng với việc cập nhật trong văn bản của chương trình nguồn, chủ yếu dựa trên việc diễn giải các mã lệnh rất đặc biệt gọi là các chỉ thị dẫn hướng của bộ tiền xử lý (destination directive of preprocessor);

Các chỉ thị này được nhận biết bởi chúng bắt đầu bằng ký hiệu (symbol) #.

Hai chỉ thị quan trọng nhất là:

- Chỉ thị sự gộp vào của các tập tin nguồn khác: #include
- Chỉ thị việc định nghĩa các macros hoặc ký hiệu: #define

Chỉ thị đầu tiên được sử dụng trước hết là nhằm gộp vào nội dung của các tập tin cần có (header file), không thể thiếu trong việc sử dụng một cách tốt nhất các hàm của thư viện chuẩn, phổ biến nhất là:

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

Chỉ thị thứ hai rất hay được sử dụng trong các tập tin thư viện (header file) đã được định nghĩa trước đó và thường được khai thác bởi các lập trình viên trong việc định nghĩa các ký hiệu như là:

```
#define NB_COUPS_MAX 100
```

```
#define SIZE 25
```

4.5.2 Cấu trúc một chương trình C

Một chương trình C bao gồm các phần như: Các chỉ thị tiền xử lý, khai báo biến ngoài, các hàm tự tạo, chương trình chính (hàm main).

Cấu trúc có thể như sau:

Các chỉ thị tiền xử lý (Preprocessor directives)

```
#include <Tên tập tin thư viện>  
#define ....
```

Định nghĩa kiểu dữ liệu (phần này không bắt buộc): dùng để đặt tên lại cho một kiểu dữ liệu nào đó để gợi nhớ hay đặt 1 kiểu dữ liệu cho riêng mình dựa trên các kiểu dữ liệu đã có.

Cú pháp: typedef <Tên kiểu cũ> <Tên kiểu mới>

Ví dụ: `typedef int SoNguyen ; // Kiểu SoNguyen là kiểu int`

Khai báo các prototype (tên hàm, các tham số, kiểu kết quả trả về,... của các hàm (*sẽ cài đặt trong phần sau, phần này không bắt buộc*):

Phần khai báo này chỉ là các khai báo đầu hàm, không phải là phần định nghĩa hàm.

Khai báo các biến ngoài (các biến toàn cục) *phần này không bắt buộc*: phần này khai báo các biến toàn cục được sử dụng trong cả chương trình.

Chương trình chính phần này bắt buộc phải có

```
<Kiểu dữ liệu trả về> main()
```

```
{
```

Các khai báo cục bộ trong hàm main: Các khai báo này chỉ tồn tại trong hàm mà thôi, có thể là khai báo biến hay khai báo kiểu.

Các câu lệnh dùng để định nghĩa hàm main return

```
<kết quả trả về>; // Hàm phải trả về kết quả
```

Cài đặt các hàm

```
<Kiểu dữ liệu trả về> Tên hàm (các tham số)
```

```
{
```

```
Các khai báo cục bộ trong hàm.  
Các câu lệnh dùng để định nghĩa  
hàm return <kết quả trả về>;  
}
```

Một chương trình C bắt đầu thực thi từ hàm main (thông thường là từ câu lệnh đầu tiên đến câu lệnh cuối cùng).

4.5.3 Các tập tin thư viện thông dụng

Đây là các tập tin chứa các hàm thông dụng khi lập trình C, muốn sử dụng các hàm trong các tập tin header này thì phải khai báo #include <Tên tập tin> ở phần đầu của chương trình

- 1. stdio.h:** Tập tin định nghĩa các hàm vào/ra chuẩn (standard input/output). Gồm các hàm in dữ liệu (printf()), nhập giá trị cho biến (scanf()), nhận ký tự từ bàn phím (getc()), in ký tự ra màn hình (putc()), nhận một dãy ký tự từ bàn phím (gets()), in chuỗi ký tự ra màn hình (puts()), xóa vùng đệm bàn phím (fflush()), fopen(), fclose(), fread(), fwrite(), getchar(), putchar(), getw(), putw()...)
 - 2. conio.h:** Tập tin định nghĩa các hàm vào ra trong chế độ DOS (DOS console). Gồm các hàm clrscr(), , getch(), getpass(), cgets(), cputs(), putch(), clreol(),...
 - 3. math.h:** Tập tin định nghĩa các hàm tính toán gồm các hàm abs(), sqrt(), log(). log10(), sin(), cos(), tan(), acos(), asin(), atan(), pow(), exp(),...
 - 4. alloc.h:** Tập tin định nghĩa các hàm liên quan đến việc quản lý bộ nhớ. Gồm các hàm calloc(), realloc(), malloc(), free(), farmalloc(), farcalloc(), farfree(),...
 - 5. io.h:** Tập tin định nghĩa các hàm vào ra cấp thấp. Gồm các hàm open(), _open(), read(), _read(), close(), _close(), creat(), _creat(), creatnew(), eof(), filelength(), lock(),...
 - 6. graphics.h:** Tập tin định nghĩa các hàm liên quan đến đồ họa. Gồm initgraph(), line(), circle(), putpixel(), getpixel(), setcolor(), ...
- Còn nhiều tập tin khác nữa.

4.5.4 Cú pháp khai báo các phần bên trong một chương trình C

☞ **Chỉ thị #include để sử dụng tập tin thư viện.**

Cú pháp:

#include <Tên tập tin> // Tên tập tin được đặt trong dấu <>

hay **#include "Tên đường dẫn"**

Menu Option của Borland C có mục INCLUDE DIRECTORIES, mục này dùng

để chỉ định các tập tin thư viện được lưu trữ trong thư mục nào.

Nếu ta dùng #include<Tên tập tin> thì Borland C sẽ tìm tập tin thư viện trong thư mục đã được xác định trong INCLUDE DIRECTORIES.

Ví dụ: include <stdio.h>

Nếu ta dùng #include"Tên đường dẫn" thì ta phải chỉ rõ tên ở đâu, tên thư mục và tập tin thư viện.

Ví dụ: #include"C:\\TC\\math.h"

Trong trường hợp tập tin thư viện nằm trong thư mục hiện hành thì ta chỉ cần đưa tên tập tin thư viện

Ví dụ:

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

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

```
#include "math.h"
```

❖ Chỉ thị #define để định nghĩa hằng số

Cú pháp:

#define <Tên hằng> <Giá trị>

Ví dụ:

```
#define pi 3.14
```

❖ Khai báo các prototype của hàm

Cú pháp:

<Kiểu kết quả trả về> Tên hàm (danh sách đối số)

Ví dụ:

```
long giaithua (int n); //Hàm tính giai thừa của số nguyên
```

```
n double x_mu_y (float x, float y); /*Hàm tính x mũ y* /
```

4.5.5 Cấu trúc một chương trình C đơn giản

```

//Khai báo thư viện
#include<tên tập tin thư viện>
//Định nghĩa hằng (không bắt buộc)
//Chương trình chính
int main()
{
    Khai báo biến – nếu có (chỉ tồn tại trong hàm); Các câu lệnh dùng để định nghĩa hàm
    main; return 0;
}

```

4.5.6 Một vài chương trình C đơn giản

Ví dụ 1 : Viết chương trình nhập vào một số thực x, sau đó tính giá trị của biểu thức ($x^2 - 5x + 4$) và xuất kết quả ra màn hình.

```

#include "stdio.h"

int main()
{
    float x, y;
    printf("\nNhap x = ");
    scanf("%f ",&x);
    y= x*x-5*x+4;
    printf("Gia tri bieu thuc: %f", y);
}

```

Ví dụ 2 : Viết chương trình nhập vào một chuỗi ký tự, sau đó xuất chuỗi ký tự đó ra màn hình.

```

#include "stdio.h"

int main()
{
    char ChuoiKyTu[50];
    printf("\nNhap chuoi ky tu: ");
    gets(ChuoiKyTu);
    printf("Chuoi da nhap: %s",ChuoiKyTu);
}

```

CÂU HỎI ÔN TẬP

Câu 1: Cho biết chỗ sai của chương trình sau:

```
int main()
{
    n=22;
    printf("%d",n);
}
```

Câu 2: Cho biết chỗ sai của chương trình sau:

```
int main()
{
    float x;
    scanf("%f",x);
}
```

Câu 3: Hãy cho biết các biểu thức sau đây đúng hay sai; Nếu đúng hãy cho biết giá trị của biểu thức đó.

37/(5*2)

37/5/2

37(5/2)

37%(5%2)

37%5%2

37-5-2

(37-5)2

37/(5*2)

Câu 4: Giả sử m=5, n=2. Hãy cho biết giá trị của m và n sau khi thực thi mỗi câu lệnh sau:

m *= n++ ;

m += --n;

Câu 5: Liệt kê tất cả các ước số của số nguyên dương N.

Ví dụ : Nhập n=6

Kết quả: Các ước số của 6 là: 1 2 3 6

Hướng dẫn

Ta lấy số n lần lượt chia cho các số từ 1 đến n . Nếu số nào chia hết thì nó chính là ước của n (phép chia hết là phép chia lấy phần dư ==0)

$6 \% 1 ==0 \rightarrow 1$ là ước của 6

$6 \% 2 ==0 \rightarrow 2$ là ước của 6

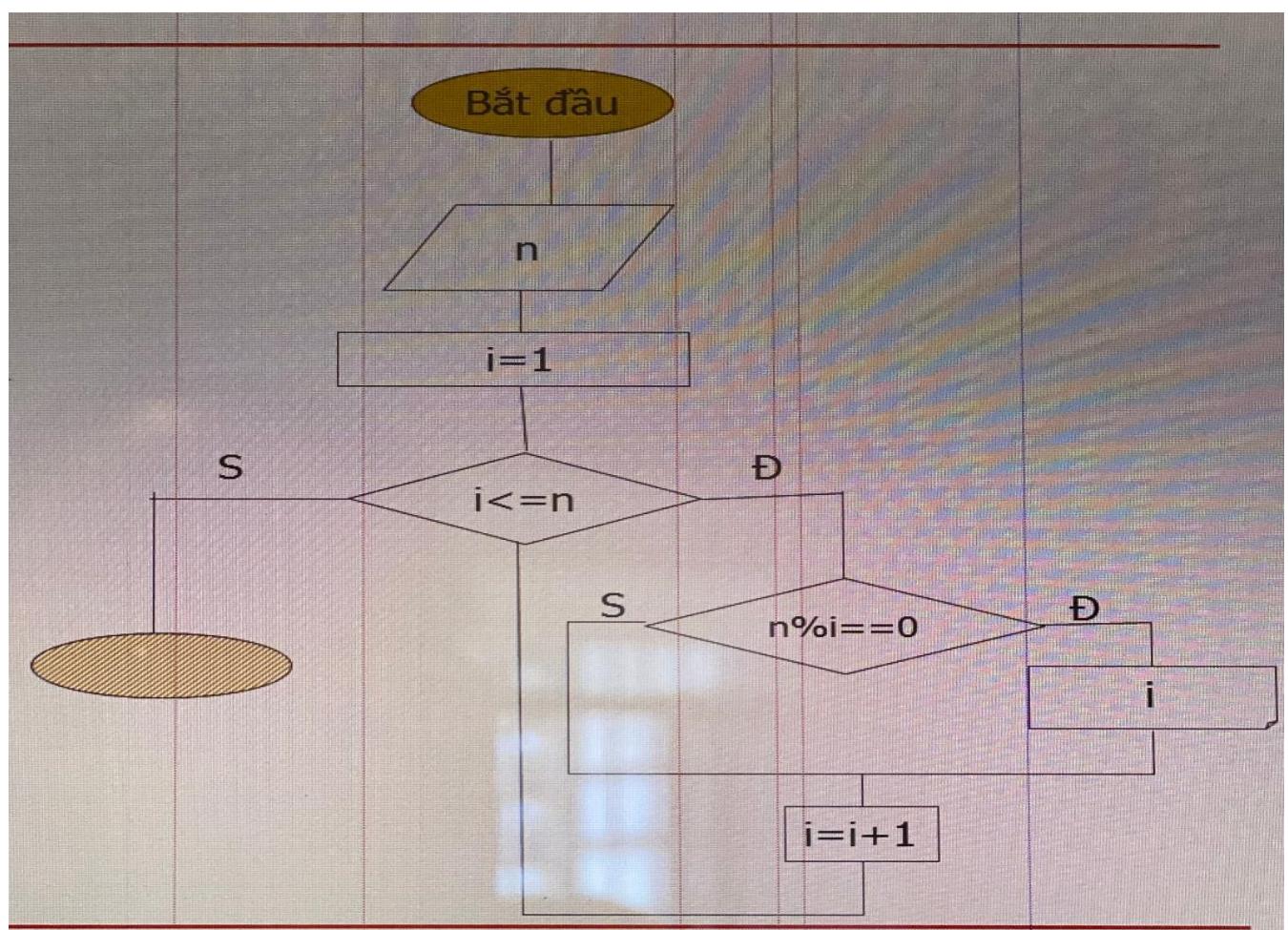
$6 \% 3 ==0 \rightarrow 3$ là ước của 6

$6 \% 4 !=0 \rightarrow 4$ không là ước của 6

$6 \% 5 !=0 \rightarrow 5$ không là ước của 6

$6 \% 6 ==0 \rightarrow 6$ là ước của 6

Nhân xét : ta thấy lặp đi lặp lại 1 công việc là lấy n chia lấy phần dư cho các số từ 1 đến n. Nếu dư ==0 thì kết luận số đó là ước của n .



Câu 6: Đếm các ước của một số nguyên dương N.

Ví dụ : n=6

Kết quả: Số ước số của 6 là: 4

Hướng dẫn

Ta lấy số n lần lượt chia cho các số từ 1 đến n . Nếu số nào chia hết thì nó

chính là ước của n , lập tức ta tăng biến đếm ước lên 1 đơn vị

$6 \% 1 ==0 \rightarrow 1$ là ước của 6 \rightarrow đếm +1

$6 \% 2 ==0 \rightarrow 2$ là ước của 6 \rightarrow đếm +1

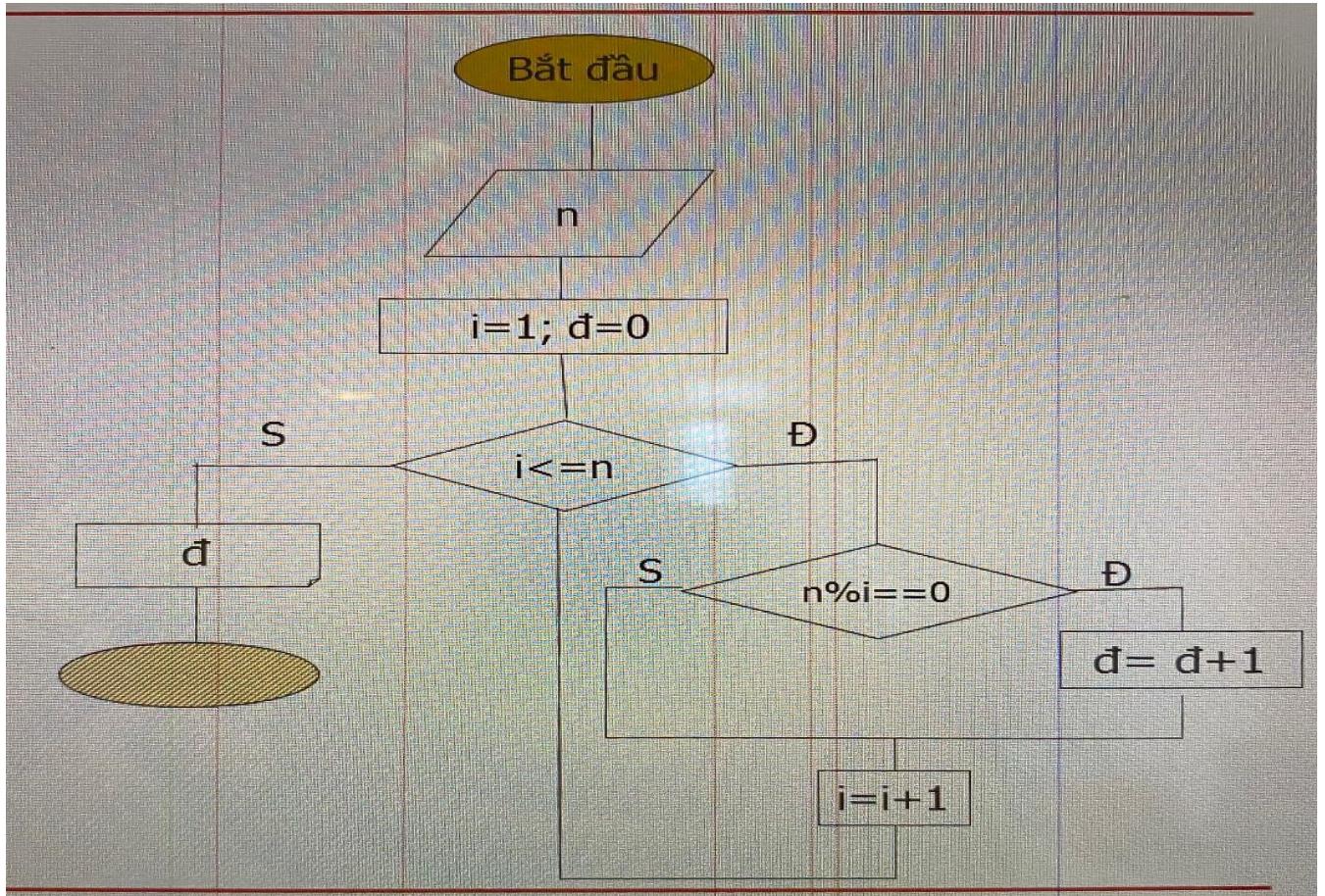
$6 \% 3 ==0 \rightarrow 3$ là ước của 6 \rightarrow đếm +1

$6 \% 4 !=0 \rightarrow 4$ không là ước của 6

$6 \% 5 !=0 \rightarrow 5$ không là ước của 6

$6 \% 6 ==0 \rightarrow 6$ là ước của 6 \rightarrow đếm +1

Nhân xét : ta thấy lặp đi lặp lại 1 công việc là lấy n chia lấy phần dư cho các số từ 1 đến n. Nếu dư ==0 thì ta tăng biến đếm lên 1 đơn vị .



Câu 7: Vẽ lưu đồ tính tổng các ước số của số nguyên dương N.

Ví dụ $n = 6$ tổng các ước $s = 1+2+3+6$

Hướng dẫn

Ta lấy số n lần lượt chia cho các số từ 1 đến n . Nếu số nào chia hết thì nó chính là ước của n , lập tức ta cộng dồn các ước đó vào biến tổng

$6 \% 1 == 0 \rightarrow 1$ là ước của $6 \rightarrow S + 1$

$6 \% 2 == 0 \rightarrow 2$ là ước của $6 \rightarrow S + 2$

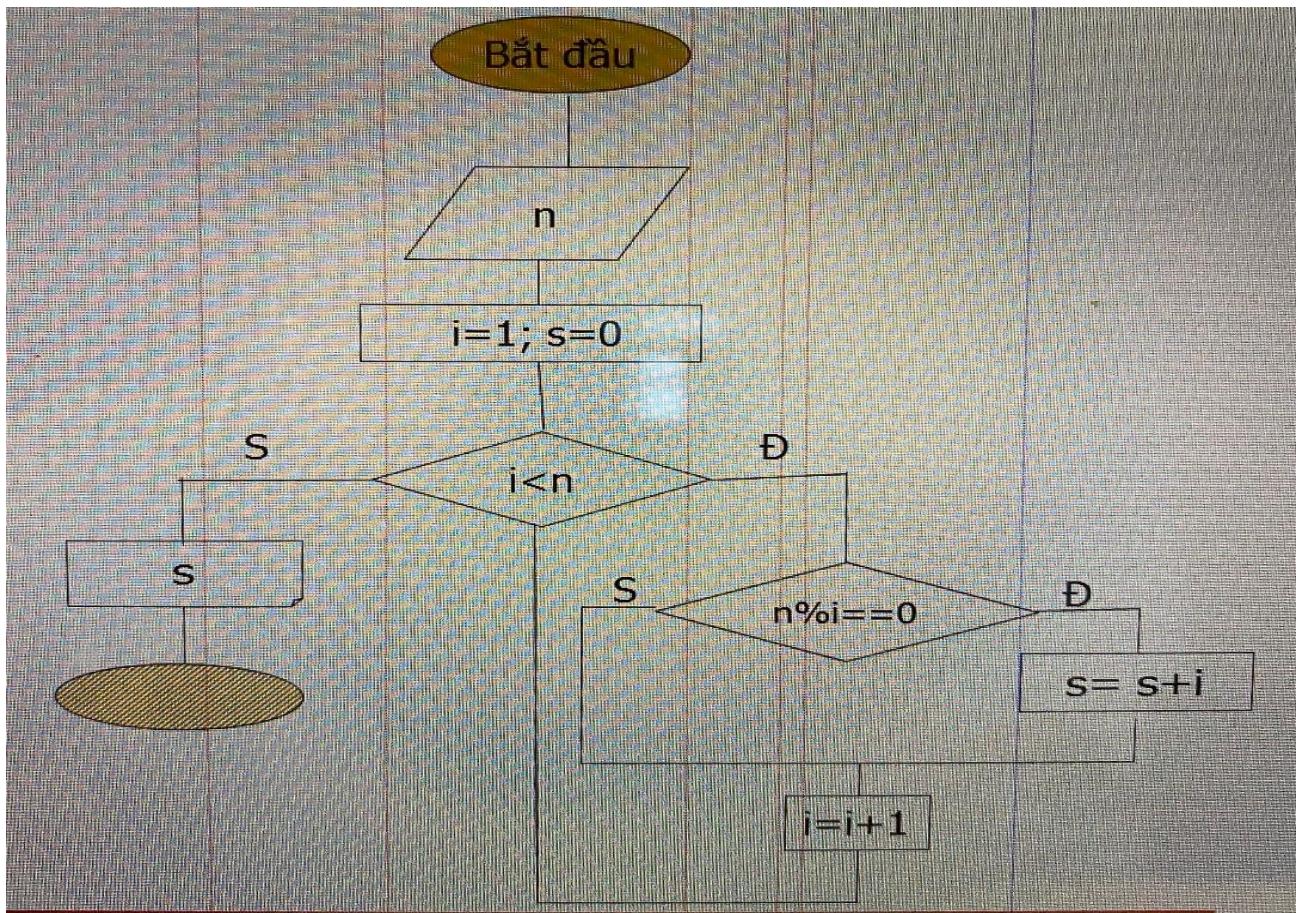
$6 \% 3 == 0 \rightarrow 3$ là ước của $6 \rightarrow S + 3$

$6 \% 4 != 0 \rightarrow 4$ không là ước của 6

$6 \% 5 != 0 \rightarrow 5$ không là ước của 6

$6 \% 6 == 0 \rightarrow 6$ là ước của $6 \rightarrow S + 6$

Nhận xét : ta thấy lặp đi lặp lại 1 công việc là lấy n chia lấy phần dư cho các số từ 1 đến n . Nếu dư == 0 thì ta lập tức cộng dồn ước đó vào biến tổng.



Câu 8: Hoàn thiện các bài tập ở các bài trước.

Câu 9: Viết chương trình tính $\log_a x$ với a, x là các số thực nhập vào từ bàn phím, và $x > 0, a > 0, a \neq 1$ (dùng $\log_a x = \ln x / \ln a$).

Câu 10: Viết chương trình nhập vào tọa độ của hai điểm (x_1, y_1) và (x_2, y_2)

- Tính hệ số góc của đường thẳng đi qua hai điểm đó theo công thức: Hệ số góc = $(y_2 - y_1) / (x_2 - x_1)$
- Tính khoảng cách giữa hai điểm.

Câu 11: Viết chương trình nhập vào một ký tự:

- In ra mã Ascii của ký tự đó.

b. In ra ký tự kế tiếp của nó.

Câu 12: Viết chương trình nhập vào điểm ba môn Toán, Lý, Hóa của một học sinh. In ra điểm trung bình của học sinh đó với hai số lẻ thập phân.

BÀI 5: CÁC TOÁN TỬ ĐIỀU KHIỂN

Sau khi học xong bài này, Sinh viên có thể nắm được:

- *Câu lệnh là gì? Phân loại câu lệnh.*
- *Cấu trúc rẽ nhánh.*
- *Cấu trúc lựa chọn.*
- *Cấu trúc vòng lặp.*
- *Các câu lệnh "đặc biệt".*

5.1 KHÁI NIỆM CÂU LỆNH VÀ KHỐI LỆNH

5.1.1 Khái niệm câu lệnh

- Một câu lệnh (statement) xác định một công việc mà chương trình phải thực hiện để xử lý dữ liệu đã được mô tả và khai báo.
- Mỗi câu lệnh có thể được viết trên một hoặc nhiều dòng và được kết thúc bằng dấu chấm phẩy (;).

5.1.2 Phân loại

Có 3 loại lệnh: lệnh đơn, khối lệnh và lệnh có cấu trúc.

- **Lệnh đơn** là một lệnh không chứa các lệnh khác. Các lệnh đơn gồm: lệnh gán (mục 2.4.3), các câu lệnh nhập xuất dữ liệu (mục 2.5), vv...
- **Khối lệnh** là một dãy các câu lệnh được đặt trong các dấu { và }.

Ví dụ: khối lệnh

```
{  
    float x;
```

```

    printf("\nNhap x=");
    scanf("%f",&x);
}

```

Lưu ý

- Không được đặt dấu chấm phẩy sau dấu ngoặc kết thúc một khối.
- Khối lệnh tương đương với một câu lệnh riêng lẻ về mặt cú pháp; Nghĩa là nơi nào đặt được một câu lệnh thì nơi đó có thể viết được bằng một khối lệnh.
- Khi khối lệnh chỉ gồm một câu lệnh thì ta có thể bỏ các dấu ngoặc ở đầu và cuối; Nghĩa là có thể xem câu lệnh là một trường hợp riêng của khối lệnh.

Một khối lệnh có thể chứa bên trong nó nhiều khối lệnh khác gọi là khối lệnh lồng nhau. Sự lồng nhau của các khối lệnh là không hạn chế.

Lưu ý về phạm vi tác động của biến trong khối lệnh lồng nhau: Trong các khối lệnh khác nhau hay các khối lệnh lồng nhau có thể khai báo các biến cùng tên.

Ví dụ 1:

```

{
    ... lệnh;
{
    int a,b; /*biến a, b trong khối lệnh thứ nhất*/
    ... lệnh;
}
...lệnh;
{
    int a,b; /*biến a,b trong khối lệnh thứ
hai*/ ... lệnh;
}
}

```

Ví dụ 2:

```

{
    int a, b;      /*biến a,b trong khối lệnh "bên ngoài"*/
    ... lệnh;
{
    int a,b; /*biến a,b bên trong khối lệnh con*/
}
}

```

- Nếu một biến được khai báo bên ngoài khối lệnh và không trùng tên với biến bên trong khối lệnh thì nó cũng được sử dụng bên trong khối lệnh.

- Một khối lệnh con có thể sử dụng các biến bên ngoài, Nhưng các lệnh bên ngoài không thể sử dụng các biến bên trong khối lệnh con.
- **Lệnh có cấu trúc** là lệnh trong đó chứa các lệnh khác. Lệnh có cấu trúc bao gồm: cấu trúc điều kiện rẽ nhánh, cấu trúc điều kiện lựa chọn, cấu trúc lặp và cấu trúc lệnh hợp thành. Lệnh hợp thành (khối lệnh) là một nhóm bao gồm nhiều khai báo biến và các lệnh được gom vào trong cặp dấu {}.

5.2 CÁC LỆNH RẼ NHÁNH, LỰA CHỌN

5.2.1 Cấu trúc rẽ nhánh

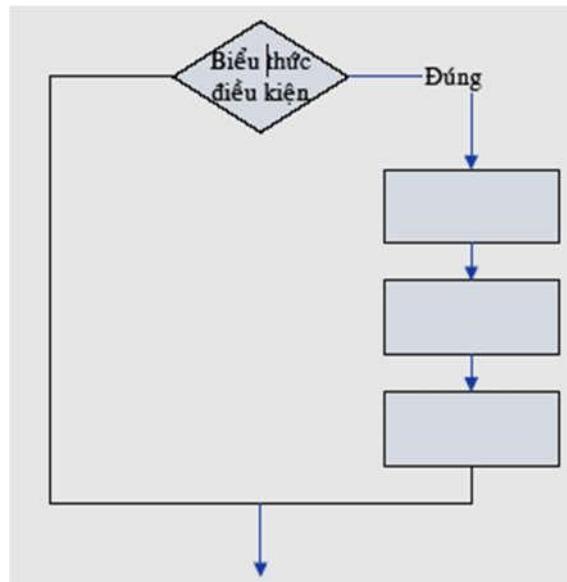
Cấu trúc rẽ nhánh là một cấu trúc được dùng rất phổ biến trong các ngôn ngữ lập trình nói chung. Trong C, có hai dạng: dạng không đầy đủ và dạng đầy đủ.

❖ Dạng không đầy đủ

Cú pháp:

```
if (<Biểu thức điều kiện>)
{
    <Công việc>
}
```

Lưu đồ cú pháp:



Giải thích:

<Công việc> được thể hiện bằng 1 câu lệnh hay 1 khối lệnh.

Kiểm tra Biểu thức điều kiện trước.

Nếu điều kiện đúng ($\neq 0$) thì thực hiện câu lệnh hoặc khối lệnh liền sau điều kiện.
Nếu điều kiện sai thì bỏ qua lệnh hoặc khối lệnh liền sau điều kiện (những lệnh và
khối lệnh sau đó vẫn được thực hiện bình thường vì nó không phụ thuộc vào điều kiện
sau if).

Ví dụ 1: Yêu cầu người thực hiện chương trình nhập vào một số thực a. In ra màn
hình kết quả nghịch đảo của a khi a $\neq 0$

```
#include <stdio.h>
#include <conio.h>
int main ()
{
    float a;
    printf("Nhập a = "); scanf("%f",&a);
    if (a !=0)
        printf("Nghịch đảo của %f là %f ",a, 1/a);
    ;
    return 0;
}
```

❖ **Dạng đầy đủ**

Cú pháp:

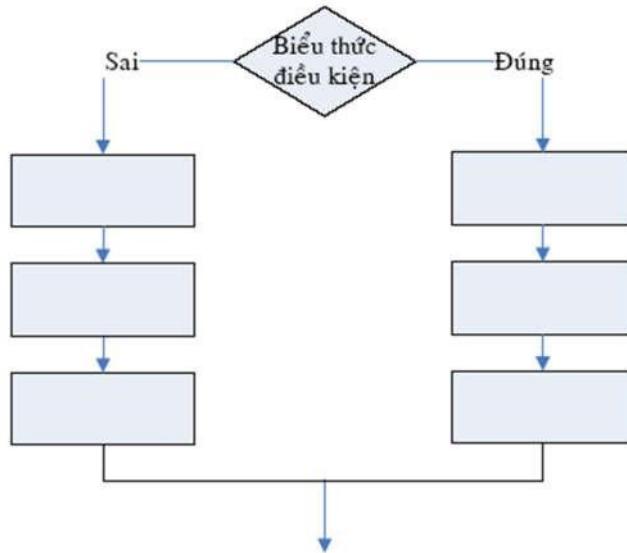
if (<Biểu thức điều kiện>)

<Công việc 1>

else

<Công việc 2 >

Lưu ý:



Giải thích:

Công việc 1, công việc 2 được thể hiện là 1 câu lệnh hay 1 khối lệnh.

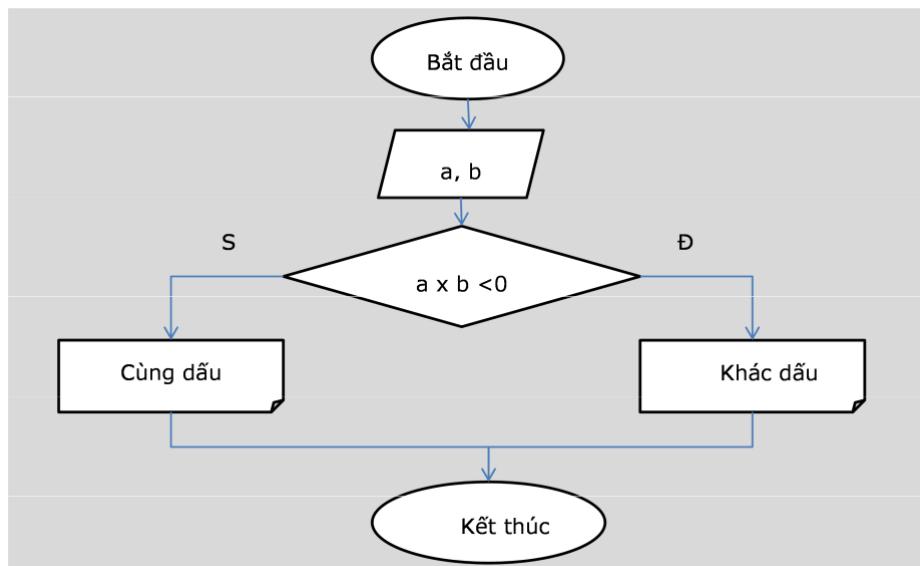
Đầu tiên Biểu thức điều kiện được kiểm tra trước.

Nếu điều kiện đúng thì thực hiện công việc 1.

Nếu điều kiện sai thì thực hiện công việc 2.

Các lệnh phía sau công việc 2 không phụ thuộc vào điều kiện.

Ví dụ: Viết chương trình nhập vào 2 số thực, cho biết 2 số đó cùng dấu hay khác dấu.



```

#include< stdio.h>           // thư viện phải khai báo
#include<conio.h>
int main()                  // chương trình chính
{
    float a,b;             // khai báo biến
    printf (" nhap a: "); scanf("%f", &a);
    printf (" nhap b: "); scanf("%f", &b);
    if (a*b<0)
        printf (" %f va %f khac dau ",a,b);
    else
        printf (" %f va %f cùng dấu ", a,b);
    return 0;
}

```

Lưu ý:

- Ngôn ngữ C cho phép sử dụng các toán tử if lồng nhau; Nghĩa là các <khoi lenh 1> và <khoi lenh 2> có thể chứa các toán tử if khác.
- Nếu số từ khóa else ít hơn số từ khóa if thì else được gắn với if không có từ khóa else gần nhất trước đó.
- Ví dụ:

```

if (n > 0)
    if (a > b)
        z = a;
    else z = b;

```

- Để chương trình rõ ràng, dễ kiểm tra, và tránh nhầm lẫn chúng ta nên viết chương trình theo các qui tắc sau:
 - Các câu lệnh và khối lệnh nằm trong một toán tử điều khiển thì viết tüt vào bên phải.
 - Các câu lệnh, khối lệnh cùng cấp thì viết trên cùng một cột.
 - Điểm đầu và điểm cuối của một khối lệnh cũng phải thẳng cột.

Ví dụ:

```
if (n > 0)
{
    if (a > b)
        z = a;
    else
        z = b;
}
```

5.2.2 Cấu trúc lựa chọn switch

Cấu trúc lựa chọn cho phép lựa chọn một trong nhiều trường hợp. Trong C, đó là câu lệnh switch.

Cú pháp:

```
switch (<Biểu thức>)
{
    case giá trị 1:
        Khối lệnh thực hiện công việc 1;
        break;
    ...
    case   giá trị n:
        Khối lệnh thực hiện công việc n;
        break;
    default:
        Khối lệnh thực hiện công việc mặc định;
        break;
}
```

Trong đó:

- Giá trị của biểu thức <biểu thức> là một số nguyên.
- Các <giá trị i> là các số nguyên, hằng ký tự, hoặc biểu thức hằng và cần có giá trị khác nhau.
- default là thành phần không bắt buộc.
- Trước hết giá trị của biểu thức <biểu thức> được tính, sau đó được so sánh với các giá trị <giá trị 1>, <giá trị 2>, ..., <giá trị n> và thực hiện khối lệnh tương ứng.
- Khi giá trị <biểu thức> khác tất cả các giá trị <giá trị i> và trong câu lệnh có thành phần default thì khối lệnh <khoi lenh thực hiện công việc mặc định> được thực hiện.

Ví dụ: Nhập vào một số nguyên, chia số nguyên này cho 2 lấy phần dư. Kiểm tra nếu phần dư bằng 0 thì in ra thông báo "số chẵn", nếu số dư bằng 1 thì in thông báo "số lẻ".

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    int songuyen, phandu;
    printf("\n Nhập vào số nguyên "); scanf("%d",&songuyen);
    phandu=(songuyen % 2);
    switch(phandu)
    {
        case 0: printf("%d là số chẵn ",songuyen);
        break;
        case 1: printf("%d là số lẻ ",songuyen);
        break;
    }
    return 0;
}
```

5.3 CÁC LỆNH LẶP

Cấu trúc vòng lặp cho phép lặp lại nhiều lần 1 công việc (được thể hiện bằng 1 câu lệnh hay 1 khối lệnh) nào đó cho đến khi thỏa mãn 1 điều kiện cụ thể.

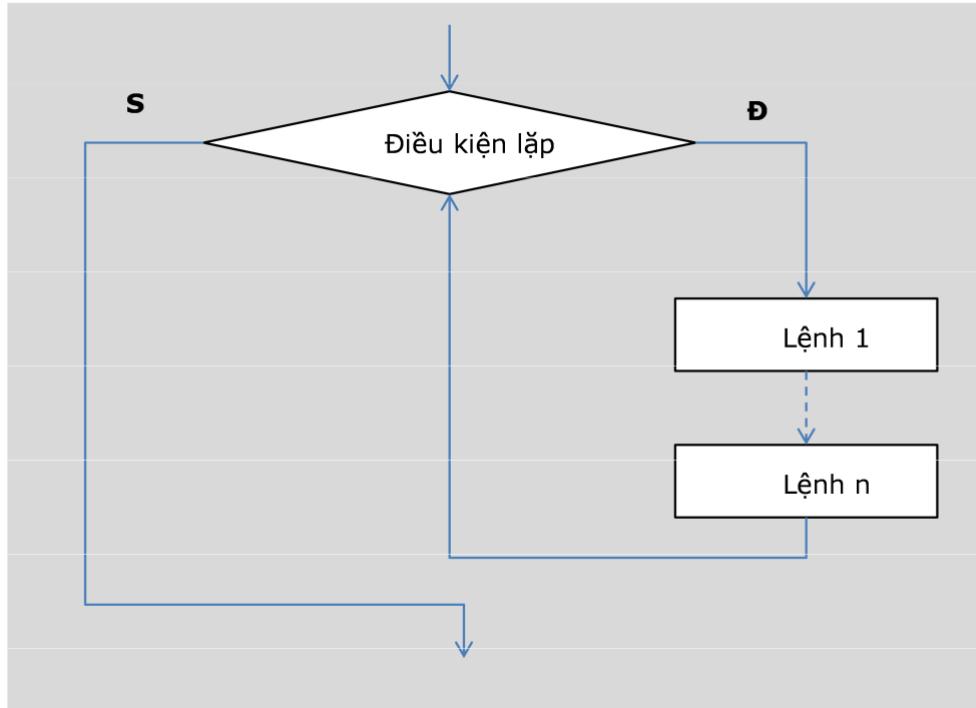
5.3.1 Cấu trúc lặp for

Vòng lặp for: cho phép lặp lại công việc cho đến khi điều kiện sai.

Cú pháp:

for (Biểu thức 1; biểu thức 2; biểu thức 3) { <Công việc> }

Lưu đồ:



Sự hoạt động của toán tử for:

Bước 1: Xác định <bieu thuc 1>

Bước 2: Xác định <bieu thuc 2>

Bước 3: Tùy thuộc vào giá trị của <bieu thuc 2> ta có lựa chọn sau:

- Nếu <bieu thuc 2> có giá trị 0 (sai) thì máy kết thúc lệnh for và thực hiện câu lệnh sau thân for.
- Nếu <bieu thuc 2> có giá trị khác 0 (đúng) thì máy thực các câu lệnh trong thân for. Khi gặp dấu ngoặc đóng } cuối cùng của thân for hoặc gặp câu lệnh continue thì máy chuyển sang bước 4.

Bước 4: Tính giá trị của <bieu thuc 3>, sau đó quay lại thực hiện bước 2 để bắt đầu một vòng lặp mới.

Chú ý:

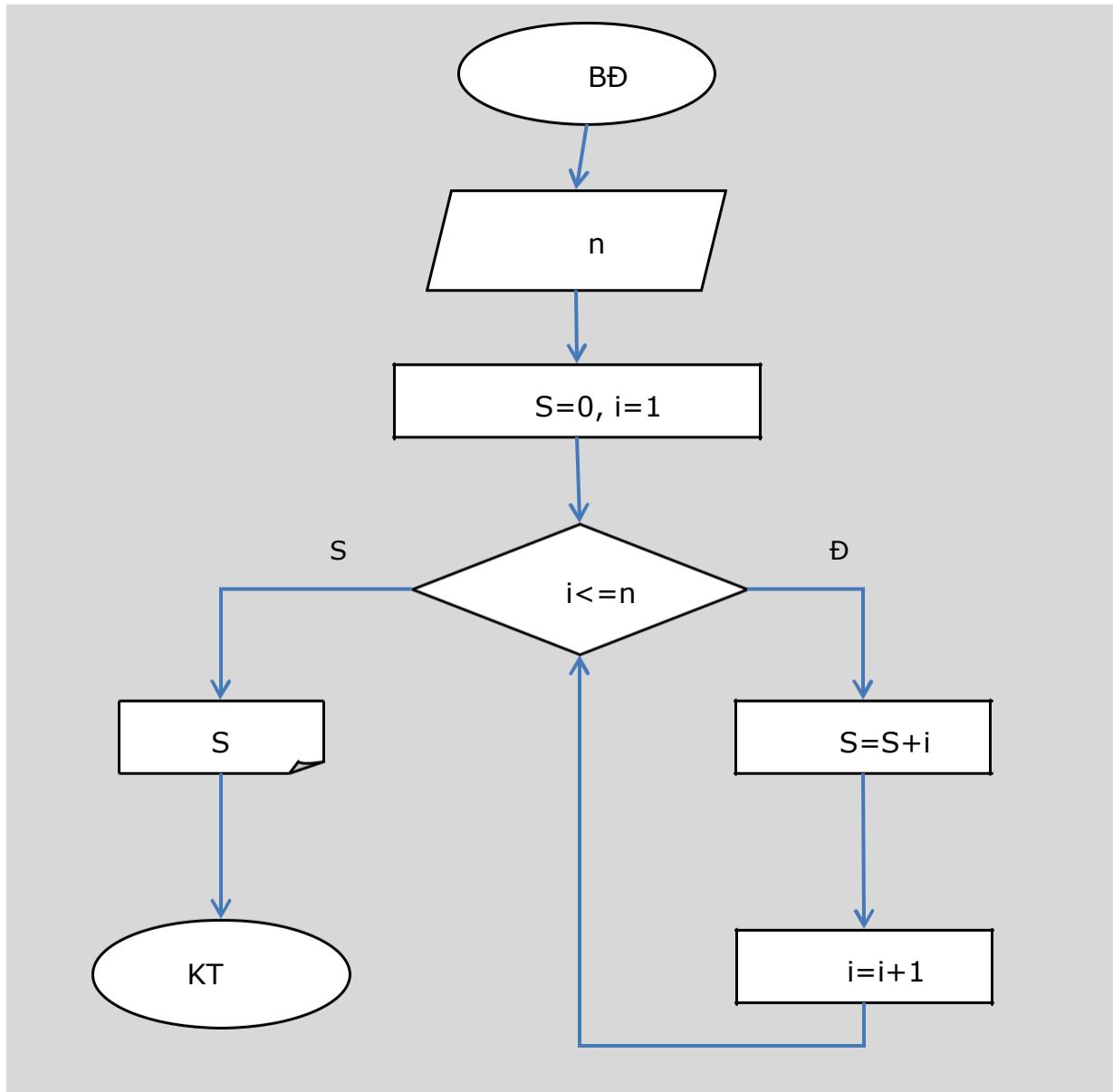
- <bieu thuc 1> thông thường là một toán tử gán dùng để khởi tạo giá trị ban đầu cho các biến điều khiển và nó chỉ được tính một lần.
- <bieu thuc 2> thông thường là một quan hệ logic và là điều kiện để tiếp tục một chu trình.

- <biểu thức 3> thông thường dùng để thay đổi giá trị của các biến điều khiển.
- Các <biểu thức 1>, <biểu thức 2>, <biểu thức 3> có thể vắng mặt trong câu lệnh for. Khi <biểu thức 2> vắng mặt trong câu lệnh thì nó luôn được xem là đúng; Trong trường hợp này để thoát khỏi vòng lặp for ta dùng các câu lệnh break hoặc return.
- Trong dấu ngoặc sau từ khoá for có 3 thành phần là <biểu thức 1>, <biểu thức 2>, <biểu thức 3> và các phần này ngăn cách với nhau bởi dấu chấm phẩy (;). Trong mỗi phần có thể viết một dãy biểu thức và được phân cách nhau bởi dấu phẩy (,). Khi đó giá trị của <biểu thức 2> được xác định bằng giá trị của biểu thức sau cùng của dãy biểu thức.
- Bên trong thân vòng lặp for có thể sử dụng các vòng lặp for khác.
- Khi gặp câu lệnh break trong thân vòng lặp for thì máy sẽ thoát khỏi vòng lặp for sâu nhất chứa câu lệnh này.
- Trong thân vòng lặp for ta có thể dùng câu lệnh return để kết thúc một hàm nào đó.
- Trong thân vòng lặp for ta có thể dùng câu lệnh continue để chuyển đến đầu vòng lặp mới; Nghĩa là máy sẽ bỏ qua những câu lệnh còn lại của thân vòng lặp và chuyển sang xét <biểu thức 3>.

Ví dụ 1: Viết đoạn chương trình in dãy số nguyên từ 1 đến 10.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    int i;
    printf("\n Day so tu 1 den 10:");
    for (i=1; i<=10; i++)
        printf("%d ",i);
    return 0;
}
```

Ví dụ 2: Viết chương trình nhập vào một số nguyên n. Tính tổng của các số nguyên từ 1 đến n



```

#include< stdio.h>
int main()
{
    int n;
    printf(" Nhập vào 1 số tự nhiên n:");
    scanf("%d",&n);
    long s=0;
  
```

```

for (int i=1; i<=n; i++)
{
    s= s+i;
}
printf(" tổng là s= %ld", s);
return 0;
}

```

5.3.2 Cấu trúc lặp while

Vòng lặp while giống như vòng lặp for, dùng để lặp lại một công việc nào đó cho đến khi điều kiện sai.

Cú pháp:

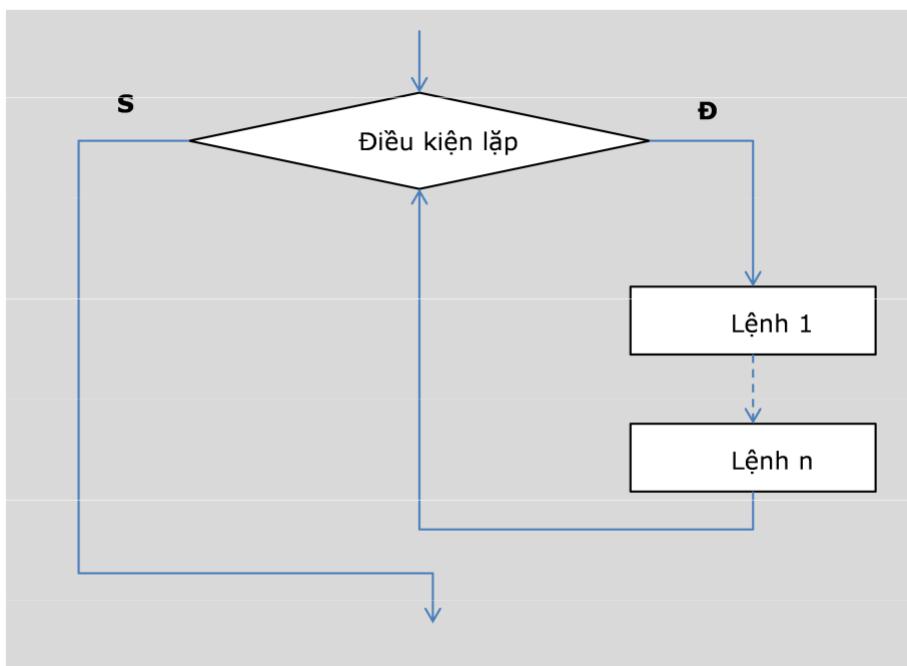
while (Biểu thức điều kiện)

{

<Công việc>

}

Lưu ý:



Sự hoạt động của toán tử while:

Bước 1: Xác định giá trị của <bieu thuc>

Bước 2:

- Nếu giá trị của <bieu thuc> là sai (bằng 0) thì máy sẽ thoát khỏi vòng lặp và thực hiện câu lệnh đứng sau lệnh while.
- Nếu giá trị của <bieu thuc> là đúng (khác 0) thì máy sẽ thực hiện các câu lệnh trong <khoi lenh>. Khi gặp dấu ngoặc đóng } cuối <khoi lenh> hoặc gặp lệnh continue thì máy trở lại bước 1.

Ví dụ: Viết chương trình nhập vào một số nguyên n. Tính tổng của các số nguyên từ 1 đến n.

Lưu đồ thuật toán như của vòng for.

```
#include< stdio.h>
int main()
{
    int n;
    printf(" Nhập n:");
    scanf("%d",&n);
    long s=0;
    int i=1;
    while(i<=n)
    {s=s+i;
        i++;
    }
    printf(" s= %ld", s);
    return 0;
}
```

Ví dụ 2: Viết chương trình tìm số nguyên dương nhỏ nhất n, sao cho $1+2+3+\dots+n > 10000$. Xuất ra màn hình tổng tìm được và số n.

```
#include "stdio.h"
int main()
{
    unsigned int n=0, s=0;
    while (s <= 10000)
        s += ++n;
    printf("\nSo n = %d \t Tong: %d", n, s);
}
```

5.3.3 Vòng lặp do... while

Vòng lặp **do ... while** giống như vòng lặp for, while, dùng để lặp lại một công việc nào đó khi điều kiện còn đúng.

Trong các toán tử while và for, việc kiểm tra biểu thức điều kiện kết thúc được đặt ở đầu vòng lặp. Khác với 2 toán tử trên, toán tử **do ... while** thực hiện việc kiểm tra biểu thức điều kiện ở cuối vòng lặp.

Cú pháp:

```
do{  
    <Công việc>  
}while (<Biểu thức điều kiện>);
```

Sự hoạt động của toán tử do...while:

Bước 1: Thực hiện các câu lệnh trong <khoi lenh công việc>.

Bước 2: Xác định giá trị của <bieu thuc điều kiện>. Tùy theo giá trị của <bieu thuc điều kiện > mà máy sẽ thực hiện rẽ nhánh:

- Nếu giá trị của <bieu thuc dk> là đúng (khác 0) thì máy quay lại bước 1.
- Nếu giá trị của <bieu thuc dk > là sai (bằng 0) thì máy kết thúc câu lệnh **do...while**, và thực hiện câu lệnh sau câu lệnh do...while.

Ví dụ: viết chương trình nhập vào 1 số tự nhiên <100. Nếu nhập sai thì thông báo sai và nhập lại.

```
do {  
    printf (" Nhập vào số phần tử ");  
    scanf(" %d", &n);  
    if(n<=0 || n >= 100)  
        printf("Nhập sai. Nhập lại");  
}while (n<=0 || n > = 100);
```

5.3.4 So sánh các vòng lặp

* **vòng lặp for, while:**

- Kiểm tra điều kiện trước thực hiện công việc sau nên đoạn lệnh thực hiện công việc có thể không được thực hiện.
- Vòng lặp kết thúc khi nào điều kiện sai.

* **Vòng lặp do...while:**

- Thực hiện công việc trước kiểm tra điều kiện sau nên đoạn lệnh thực hiện công việc được thực hiện ít nhất 1 lần.
- Vòng lặp kết thúc khi nào điều kiện sai.

5.4 CÁC CÂU LỆNH ĐẶC BIỆT

5.4.1 Lệnh break

Cú pháp: `break ;`

Dùng để thoát khỏi vòng lặp. Khi gặp câu lệnh này trong vòng lặp, chương trình sẽ thoát ra khỏi vòng lặp và chỉ đến câu lệnh liền sau nó. Nếu nhiều vòng lặp thì break sẽ thoát ra khỏi vòng lặp gần nhất. Ngoài ra, break còn được dùng trong cấu trúc lựa chọn switch.

Ví dụ:

```
#include "stdio.h"
int main()
{
    int i;
    for (i=0; i<10; i++)
        {if (i == 5) break; printf("%d
",i);}
}
```

Chương trình trên xuất ra màn hình các số: 0 1 2 3 4

5.4.2 Lệnh continue

Cú pháp: `continue`

- Khi gặp lệnh này trong các vòng lặp, chương trình sẽ bỏ qua phần còn lại trong vòng lặp và tiếp tục thực hiện lần lặp tiếp theo.
- Đối với lệnh for, *biểu thức 3* sẽ được tính trị và quay lại bước 2.
- Đối với lệnh while, do while; *biểu thức điều kiện* sẽ được tính và xét xem có thể tiếp tục thực hiện <*Công việc*> nữa hay không? (dựa vào kết quả của *biểu thức điều kiện*).

Ví dụ:

```
#include "stdio.h"
int main()
{
    int i;
    for (i=0 ;i <10; i++)
    {
        if (i%2 == 0) continue;
        printf("%d ",i);
    }
}
```

Chương trình trên xuất ra màn hình các số: 1 3 5 7 9

CÂU HỎI ÔN TẬP

Câu 1: Nhập vào 1 ký tự, nếu là chữ hoa xuất: CHUHOA, nếu là chữ thường xuất CHUTHUONG, nếu là số chẵn xuất SOCHAN và là số lẻ xuất SOLE

Câu 2: Viết chương trình nhập vào một số nguyên dương N. Tính tổng của các ký số có trong N.

Ví dụ: nhập 1234 → $1 + 2 + 3 + 4 = 10 \rightarrow$ in ra 10.

Câu 3: Viết chương trình cho phép nhập 1 số nguyên dương N. Xuất ra số lượng chữ số lẻ của N.

Câu 4: Viết chương trình tính tổng sau:

$$S(n) = 1 + 1.2 + 1.2.3 + \dots + 1.2.3\dots n, \text{ Với } n \text{ nhập từ bàn phím.}$$

Câu 5: Viết chương trình nhập 1 số nguyên có hai chữ số. Hãy in ra cách đọc của nó.

Ví dụ:	Số	Cách đọc
	95	Chín mươi lăm
	11	Mười một
	31	Ba mươi một

Câu 6: Viết chương trình nhập 1 số nguyên có ba chữ số. Hãy in ra cách đọc của nó.

Ví dụ:	Số	Cách đọc
	105	Một trăm lẻ năm
	101	Một trăm lẻ một
	111	Một trăm mười một
	131	Một trăm ba mươi một
	145	Một trăm bốn mươi lăm

BÀI 6: CHƯƠNG TRÌNH CON

Sau khi học xong bài này, sinh viên sẽ nắm được các vấn đề sau:

- *Khái niệm về hàm (function) trong C.*
- *Cách xây dựng và cách sử dụng hàm trong C.*

6.1 KHÁI NIỆM HÀM TRONG C

Trong những chương trình lớn, có thể có những đoạn chương trình viết lặp đi lặp lại nhiều lần, để tránh rườm rà và mất thời gian khi viết chương trình; người ta thường phân chia chương trình thành nhiều module, mỗi module giải quyết một công việc nào đó. Các module như vậy gọi là các chương trình con.

Một tiện lợi khác của việc sử dụng chương trình con là ta có thể dễ dàng kiểm tra xác định tính đúng đắn của nó trước khi ráp nối vào chương trình chính và do đó việc xác định sai sót để tiến hành hiệu đính trong chương trình chính sẽ thuận lợi hơn.

Trong C, chương trình con được gọi là hàm. Hàm trong C có thể trả về kết quả thông qua tên hàm hay có thể không trả về kết quả.

Hàm có hai loại: hàm chuẩn và hàm tự định nghĩa. Trong chương này, ta chú trọng đến cách định nghĩa hàm và cách sử dụng các hàm đó.

Một hàm khi được định nghĩa thì có thể sử dụng bất cứ đâu trong chương trình. Trong C, một chương trình bắt đầu thực thi bằng hàm main.

Ví dụ 1: Ta có hàm max để tìm số lớn giữa 2 số nguyên a, b như sau:

```
int max(int a, int b)
{
    return (a>b) ? a:b;
}
```

Ví dụ 2: Ta có chương trình chính (hàm main) dùng để nhập vào 2 số nguyên a,b và in ra màn hình số lớn trong 2 số.

```
#include <stdio.h>
#include <conio.h>
int max(int a, int b)
{
    return (a>b) ? a:b;
}

int main()
{
    int a, b, c;
    printf("\n Nhập vào 3 số a, b,c  ");
    scanf("%d%d%d",&a,&b,&c);
    printf("\n Số lớn là %d",max(a,
        max(b,c))); return 0;
}
```

6.2 HÀM THƯ VIỆN

Hàm thư viện là những hàm đã được định nghĩa sẵn trong một thư viện nào đó, muốn sử dụng các hàm thư viện thì phải khai báo thư viện trước khi sử dụng bằng lệnh

```
#include <tên thư viện.h>
```

Ý nghĩa của một số thư viện thường dùng:

- 1. stdio.h :** Thư viện chứa các hàm vào/ ra chuẩn (**standard input/output**). Gồm các hàm **printf()**, **scanf()**, **getc()**, **putc()**, **gets()**, **puts()**, **fflush()**, **fopen()**, **fclose()**, **fread()**, **fwrite()**, **getchar()**, **putchar()**, **getw()**, **putw()**...
- 2. conio.h :** Thư viện chứa các hàm vào ra trong chế độ DOS (DOS console). Gồm các hàm **clrscr()**, **getche()**, **getpass()**, **cgets()**, **cputs()**, **putch()**, **clreol()**...
- 3. math.h:** Thư viện chứa các hàm tính toán gồm các hàm **abs()**, **sqrt()**, **log()**, **log10()**, **sin()**, **cos()**, **tan()**, **acos()**, **asin()**, **atan()**, **pow()**, **exp()**...

4. alloc.h: Thư viện chứa các hàm liên quan đến việc quản lý bộ nhớ. Gồm các hàm **calloc()**, **realloc()**, **malloc()**, **free()**, **farmalloc()**, **farcalloc()**, **farfree()**, ...

5. io.h: Thư viện chứa các hàm vào ra cấp thấp. Gồm các hàm **open()**, **_open()**, **read()**, **_read()**, **close()**, **_close()**, **creat()**, **_creat()**, **creatnew()**, **eof()**, **filelength()**, **lock()**,...

6. graphics.h: Thư viện chứa các hàm liên quan đến đồ họa. Gồm **initgraph()**, **line()**, **circle()**, **putpixel()**, **getpixel()**, **setcolor()**, ...

Muốn sử dụng các hàm thư viện thì ta phải xem cú pháp của các hàm và sử dụng theo đúng cú pháp (xem trong phần trợ giúp của borland C).

6.3 HÀM NGƯỜI DÙNG

Hàm người dùng là những hàm do người lập trình tự tạo ra nhằm đáp ứng nhu cầu xử lý của mình.

6.3.1 Xây dựng một hàm

Cấu trúc của một hàm tự thiết kế:

```
<Kiểu kết quả> Tên hàm ([<kiểu t_ số> <tham số>] [,<kiểu t số><tham số>][...])  
{  
    [Khai báo biến cục bộ và các câu lệnh thực hiện hàm]  
    [return [<Biểu thức>];]  
}
```

Giải thích:

- **Kiểu kết quả:** là kiểu dữ liệu của kết quả trả về, có thể là: int, byte, char, float, int... Một hàm có thể có hoặc không có kết quả trả về. Trong trường hợp *hàm không có kết quả trả về ta nên sử dụng kiểu kết quả là void*.
- **Kiểu t số:** là kiểu dữ liệu của tham số.
- **Tham số:** là tham số truyền dữ liệu vào cho hàm, một hàm có thể có hoặc không có tham số. **Tham số này gọi là tham số hình thức, khi gọi hàm chúng ta phải truyền cho nó các tham số thực tế.** Nếu có nhiều tham số, mỗi tham số phân cách nhau dấu phẩy (,).

- Bên trong thân hàm (*phần giới hạn bởi cặp dấu {}*) là các khai báo cùng các câu lệnh xử lý. Các khai báo bên trong hàm được gọi là các khai báo cục bộ trong hàm và các khai báo này chỉ tồn tại bên trong hàm mà thôi.
- Khi định nghĩa hàm, ta thường sử dụng câu lệnh **return** để trả về kết quả thông qua tên hàm.
- Lệnh return dùng để thoát khỏi một hàm và có thể trả về một giá trị nào đó.

Cú pháp:

return ; //không trả về giá trị, chỉ thoát khỏi hàm **return**

<biểu thức>; //Trả về giá trị của biểu thức và thoát

- Nếu hàm có kết quả trả về, ta bắt buộc phải sử dụng câu lệnh return để trả về kết quả cho hàm.

Ví dụ: Viết hàm tìm số lớn giữa 2 số nguyên a và

```
b int max(int a, int b)
```

```
{
```

```
    return (a>b) ? a:b;
```

```
}
```

6.3.2 Sử dụng hàm

Một hàm khi định nghĩa thì chúng vẫn chưa được thực thi trừ khi ta có một lời gọi đến hàm đó.

Cú pháp gọi hàm: <Tên hàm>([Danh sách các tham số])

Ví dụ: Viết chương trình kiểm tra 1 số tự nhiên có phải là nguyên tố hay không.

```
int ktnt(int n)
{
    if (n<2)  return 0;
    if (n==2) return 1;
    if(n %2 == 0)  return 0; //n là số chẵn thì không nguyên tố
    for(int i =3; i< n/2 ; i=i+2) //nếu là số lẻ thì kiểm tra n có ước số nào không
        if (n%i ==0)
```

```

        return 0;
    }
}

int main()
{
    unsigned int a;
    printf("Nhập a: "); scanf("%d ",&a);
int nt= ktnt (a);
    if (nt== 1)
        printf("%d là số primo", a);
    else
        printf("%d không phải là số primo", a);
    return 0;
}

```

6.3.3 Nguyên tắc hoạt động của hàm

Trong chương trình, khi gặp một lời gọi hàm thì hàm bắt đầu thực hiện bằng cách chuyển các lệnh thi hành đến hàm được gọi. Quá trình diễn ra như sau:

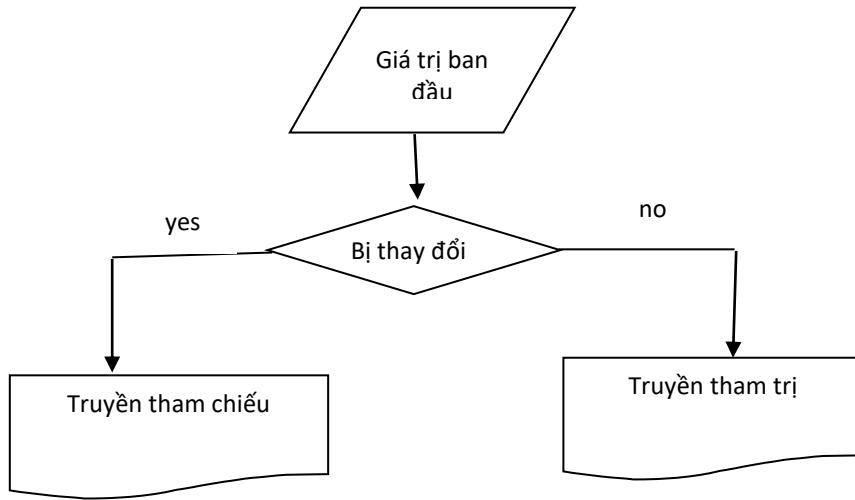
- Nếu hàm có tham số, trước tiên các tham số sẽ được *gán giá trị thực tương ứng*.
- Chương trình sẽ thực hiện tiếp các câu lệnh trong thân hàm bắt đầu từ lệnh đầu tiên đến câu lệnh cuối cùng.
- Khi gặp lệnh return hoặc dấu } cuối cùng trong thân hàm, chương trình sẽ thoát khỏi hàm để trở về chương trình gọi nó và thực hiện tiếp tục những câu lệnh của chương trình này.

6.4 Truyền tham số cho hàm

Một điều khá quan trọng trong C đó là **truyền tham chiếu và tham trị vào một phương thức (Function)**.

Truyền tham chiếu là truyền địa chỉ ô nhớ của biến, do đó khi thay đổi giá trị của biến bên trong phương thức thì giá trị của biến cũng bị thay đổi bên ngoài phương thức.

Truyền tham trị là truyền giá trị của biến (không phải là địa chỉ ô nhớ), khi đó phương thức sẽ tự động tạo ra một địa chỉ ô nhớ mới để lưu trữ giá trị này, do đó nó chỉ được thay đổi trong phương thức hiện hành và giá trị của biến không bị thay đổi bên ngoài phương thức hiện hành.



Ví dụ 1: Truyền tham trị trong C

```
# include<stdio.h>

void change(int num)
{
    num = num + 1;
}

int main()
{
    int x = 100;
    printf("Truoc khi goi phuong thuc x = %d \n", x);
    change(x); // truyen tham tri vao phuong thuc
    printf("Sau khi goi phuong thuc x = %d \n", x);
    return 0;
}
```

Kết quả

Truoc khi goi phuong thuc x = 100

Sau khi goi phuong thuc x = 100

Trong ví dụ trên, giá trị của biến x không bị thay đổi bên ngoài phương thức change(), mặc dù bên trong phương thức change() chúng ta đã cố gắng thay đổi bằng cách tăng m lên 1.

Ví dụ 2 : Truyền tham chiếu trong C

```
#include<stdio.h>

void change(int *num)
{
    *num = *num + 1;
}

int main()
{
    int x = 100;
    printf("Truoc khi goi phuong thuc x = %d \n", x);
    change(&x); // truyền tham chiếu vào phương thức
    printf("Sau khi goi phuong thuc x = %d \n", x);
    return 0;
}
```

Kết quả :

Truoc khi goi phuong thuc x = 100

Sau khi goi phuong thuc x = 101

Trong ví dụ trên, Giá trị của biến x bị thay đổi cả bên trong và bên ngoài phương thức change().

Mặc nhiên, việc truyền tham số cho hàm trong C là truyền theo giá trị; nghĩa là các giá trị thực (tham số thực) không bị thay đổi giá trị khi truyền cho các tham số hình thức

Ví dụ 3: Giả sử ta muốn in ra nhiều dòng, mỗi dòng 50 ký tự nào đó. Để đơn giản ta viết một hàm, nhiệm vụ của hàm này là in ra trên một dòng 50 ký tự nào đó. Hàm này có tên là InKT.

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

```

int InKT(char ch)
{
    int i;
    for(i=1;i<=50;i++) printf("%c",ch);
    printf("\n");
}
int main()
{
    char c = 'A';
    InKT('*'); /* In ra 50 dau * */
    InKT('+');
    InKT(c);
    return 0;
}

```

Lưu ý:

- Trong hàm InKT ở trên, biến ch gọi là tham số hình thức được truyền bằng giá trị (gọi là tham trị của hàm). Các tham trị của hàm coi như là một biến cục bộ trong hàm và chúng được sử dụng như là dữ liệu đầu vào của hàm.
- Khi chương trình con được gọi để thi hành, tham trị được cấp ô nhớ và nhận giá trị là bản sao giá trị của tham số thực. Do đó, mặc dù tham trị cũng là biến, nhưng việc thay đổi giá trị của chúng không có ý nghĩa gì đối với bên ngoài hàm, không ảnh hưởng đến chương trình chính, nghĩa là không làm ảnh hưởng đến tham số thực tương ứng.

Ví dụ 4: Ta xét chương trình sau đây:

```

#include <stdio.h>
#include <conio.h>
int hoanvi (int a, int b)
{
    int t;
    t=a; //Đoạn này hoá vị giá trị của 2 biến a, b
    a=b;
    b=t;

    printf("\Ben trong ham a=%d , b=%d",a,b);
    return 0;
}

```

```

int main()
{
    int a, b;
    printf("\n Nhap vao 2 so nguyen a, b:");
    scanf("%d%d",&a,&b);
    printf("\n Truoc khi goi ham hoan vi a=%d ,b=%d",a,b);
    hoanvi(a,b);
    printf("\n Sau khi goi ham hoan vi a=%d ,b=%d",a,b);
    return 0;
}

```

Kết quả thực hiện chương trình:

```

Nhap vao 2 so nguyen a, b:6 5
Truoc khi goi ham hoan vi a=6 ,b=5
Ben trong ham a=5 , b=6
Sau khi goi ham hoan vi a=6 ,b=5

```

❖ Lưu ý

Trong đoạn chương trình trên, nếu ta muốn sau khi kết thúc chương trình con giá trị của a, b thay đổi thì ta phải đặt tham số hình thức là các con trỏ, còn tham số thực tế là địa chỉ của các biến.

Lúc này mọi sự thay đổi trên vùng nhớ được quản lý bởi con trỏ là các tham số hình thức của hàm thì sẽ ảnh hưởng đến *vùng nhớ đang được quản lý bởi tham số thực tế tương ứng* (cần để ý rằng vùng nhớ này chính là các biến ta cần thay đổi giá trị). Người ta thường áp dụng cách này đối với các dữ liệu đầu ra của hàm.

6.5 TÂM VỰC CỦA BIẾN: BIẾN TOÀN CỤC, BIẾN CỤC BỘ

6.5.1 Biến toàn cục

Là biến được khai báo ở đầu chương trình chính, chúng tồn tại trong suốt thời gian chạy của chương trình. Biến toàn cục ảnh hưởng toàn bộ chương trình, cả chương trình chính lẫn chương trình con.

6.5.2 Biến cục bộ

Là biến được khai báo trong mỗi chương trình con . Chúng được hình thành khi chương trình con được gọi, và sẽ tự biến mất khi chương trình con kết thúc.

CÂU HỎI ÔN TẬP

Viết dưới dạng hàm. Rồi gọi thực hiện chúng trong hàm main

Câu 1: Viết chương trình nhập 3 số từ bàn phím, tìm số lớn nhất trong 3 số đó, in kết quả lên màn hình.

Câu 2: Viết chương trình tính chu vi, diện tích của tam giác với yêu cầu sau khi nhập 3 số a, b, c phải kiểm tra lại xem a, b, c có tạo thành một tam giác không? Nếu có thì tính chu vi và diện tích. Nếu không thì in ra câu " Không tạo thành tam giác".

Câu 3: Viết chương trình giải phương trình bậc nhất $ax + b = 0$ với a, b nhập từ bàn phím.

Câu 4: Viết chương trình giải phương trình bậc hai $ax^2 + bx + c = 0$ với a, b, c nhập từ bàn phím.

Câu 5: Viết chương trình nhập từ bàn phím 2 số a, b và một ký tự ch.

Nếu: ch là "+" thì thực hiện phép tính $a + b$ và in kết quả lên màn hình.

ch là "-" thì thực hiện phép tính $a - b$ và in kết quả lên màn hình.

ch là "*" thì thực hiện phép tính $a * b$ và in kết quả lên màn hình.

ch là "/" thì thực hiện phép tính a / b và in kết quả lên màn hình.

Câu 6: Viết hàm kiểm tra 1 số nguyên có phải là số nguyên tố hay không. Áp dụng hàm này viết chương trình xuất ra các số nguyên tố nhỏ hơn N (N nhập từ bàn phím)

Câu 7: Làm lại các bài tập cũ dưới dạng hàm.

BÀI 7: MẢNG- CHUỖI KÝ TỰ

Học xong bài này, sinh viên sẽ nắm được các vấn đề sau:

- Khái niệm về kiểu dữ liệu mảng cũng như ứng dụng của nó.
- Cách khai báo biến kiểu mảng và các phép toán trên các phần tử của mảng.
- Đi sâu vào các giải thuật trên mảng 1 chiều

7.1 KHÁI NIỆM

Mảng là một tập hợp các phần tử cố định có cùng một kiểu dữ liệu, gọi là kiểu phần tử. Kiểu phần tử có thể là có các kiểu bất kỳ: ký tự, số, chuỗi ký tự...; cũng có khi ta sử dụng kiểu mảng để làm kiểu phần tử cho một mảng (trong trường hợp này ta gọi là mảng của mảng hay mảng nhiều chiều).

Ta có thể chia mảng làm 2 loại: mảng một chiều và mảng nhiều chiều.

Mảng là kiểu dữ liệu được sử dụng rất thường xuyên. Chẳng hạn, người ta cần quản lý một danh sách họ và tên của khoảng 100 sinh viên trong một lớp. Nhận thấy rằng mỗi họ và tên để lưu trữ ta cần 1 biến kiểu chuỗi, như vậy 100 họ và tên thì cần khai báo 100 biến kiểu chuỗi. Nếu khai báo như thế này thì đoạn khai báo cũng như các thao tác trên các họ tên sẽ rất dài dòng và rắc rối. Vì thế, kiểu dữ liệu mảng giúp ích ta trong trường hợp này; chỉ cần khai báo 1 biến, biến này có thể coi như là tương đương với 100 biến kiểu chuỗi ký tự; đó là 1 mảng mà các phần tử của nó là chuỗi ký tự. Hay như

để lưu trữ các từ khóa của ngôn ngữ lập trình C, ta cũng dùng đến một mảng để lưu trữ chúng.

Kích thước của mảng là số phần tử của mảng. Kích thước này phải được biết ngay khi khai báo mảng.

Nếu xét dưới góc độ toán học, mảng 1 chiều giống như một vector. Mỗi phần tử của mảng một chiều có giá trị ***không phải là một mảng***

7.2 MẢNG MỘT CHIỀU

7.2.1 Khai báo

❖ **Khai báo tường minh (số phần tử xác định):**

Cú pháp: **<kiểu cơ sở> <tên mảng> [<số phần tử>];**

Ý nghĩa:

- **<Tên mảng>**: được đặt đúng theo quy tắc đặt tên của danh biểu. Tên này cũng mang ý nghĩa là tên của biến mảng.
- **[<Số phần tử>]**: là một hằng số nguyên, cho biết số lượng phần tử tối đa trong mảng là bao nhiêu (hay nói khác đi nó là kích thước của mảng).
- **<Kiểu cơ sở>**: là kiểu dữ liệu của mỗi phần tử của mảng.

Ví dụ:

Khai báo mảng một chiều có tên là **songuyen** gồm 10 phần tử và kiểu cơ sở là int
int songuyen [10] ;

Khai báo mảng một chiều có tên là **sothuc** gồm 15 phần tử và kiểu cơ sở là float
float sothuc [15] ;

Khai báo mảng một chiều có tên là **daykytu** gồm 30 phần tử và kiểu cơ sở là char
char daykytu [30] ;

❖ **Khai báo không tường minh(số phần tử không xác định)**

Cú pháp: **<kiểu cơ sở> <tên mảng> [];**

Khi khai báo, không cho biết rõ số phần tử của mảng, kiểu khai báo này thường được áp dụng trong các trường hợp: vừa khai báo vừa gán giá trị, hoặc khai báo mảng là tham số hình thức của hàm.

Cách 1. Vừa khai báo vừa gán giá trị

Cú pháp:

<Kiểu> <Tên mảng> [] = {Các giá trị cách nhau bởi dấu phẩy}

Nếu vừa khai báo vừa gán giá trị thì mặc nhiên C sẽ hiểu số phần tử của mảng là số giá trị mà chúng ta gán cho mảng trong cặp dấu {}.

Ví dụ: float x[] = {12.1, 7.23, 5.0, 27.6, 87.9, 9.31};

Chúng ta có thể sử dụng hàm **sizeof()** để biết số phần tử của mảng như sau:

Số phần tử=**sizeof(tên mảng)/ sizeof(kiểu)**

Cách 2. Khai báo mảng là tham số hình thức của hàm, trong trường hợp này ta không cần chỉ định số phần tử của mảng là bao nhiêu.

Ví dụ: void nhapmang (int a[], int n)

7.2.2 Truy cập vào các phần tử của mảng

Mỗi phần tử của mảng được truy xuất thông qua **Tên biến mảng** theo sau là **chỉ số** nằm trong cặp **dấu ngoặc vuông []**.

Chẳng hạn a[0] là phần tử đầu tiên của mảng a được khai báo ở trên.

Chỉ số của phần tử mảng là một biểu thức mà giá trị là kiểu số nguyên.

Với cách truy xuất theo kiểu này, **Tên biến mảng[Chỉ số]** có thể coi như là **một biến** có kiểu dữ liệu là **kiểu** được chỉ ra trong khai báo biến mảng.

Chỉ số của mảng có thể là một hằng, một biến hay một biểu thức đại số.

Một phần tử của mảng là một biến có kiểu dữ liệu là kiểu cơ sở nên các thao tác trên các biến cũng được áp dụng như trên các phần tử của mảng.

Ví dụ Khai báo mảng số thực có 5 phần tử **float a [5] ;**

Khi đó Mảng số thực trên có các phần tử là:

a [0], a [1], a [2], a [3], a [4] và là những biến kiểu **float**

Và ta có thể thực hiện các phép toán:

```

float t=10.0;

int i=1;

a [0]=4.2;

a[2]=t;

a [i ]=(a [0] +a [2])/2;

printf("\nGia tri: %f ", a[1]);

scanf("%f ",&a [4]);

t= a [4];

a [2*i+1]= a [2*i]+ a [2*i+2];

```

Ta có thể khởi gán giá trị cho mảng:

```
float x[6] = {12.1, 7.23, 5.0, 27.6, 87.9, 9.31};
```

khi đó: x[0]=12.1, x[1]=7.23, x[2]=5.0, x[3]=27.6, x[4]=87.9, x[5]=9.31

7.2.3 Nhập dữ liệu cho mảng một chiều

Khai báo mảng a để lưu trữ 100 phần tử là các số nguyên: **int a [100]** khi đó máy sẽ cấp phát 200 byte để lưu trữ mảng a

Hình ảnh mảng a gồm n phần tử được lưu trong bộ nhớ

Vị trí	0	1	2	3	4	n-3	n-2	n-1
Giá trị a[]	7	3	9	4	5	8	12	2
Tên phần tử	a[0]	a[1]	a[2]	a[3]	a[4]	a[n-3]	a[n-2]	a[n-1]

Nhập dữ liệu cho các phần tử

```
a [ 0 ] = 7    scanf (" %d", & a[0 ]);
```

```
a [ 1 ] = 3    scanf (" %d", & a[1 ]);
```

```
a [ 2 ] = 9    scanf (" %d", & a[2 ]);
```

.....

```
a [ n-1 ] = 2    scanf (" %d", & a[n-1 ]);
```

```
// Nhập từng phần tử của mảng
```

```
for (int i = 0 ; i < n ; i++)  
{  
    printf (" nhap a[%d]: ", i) ;  
    scanf (" %d ", & a[ i ]);  
}
```

7.2.4 Xuất dữ liệu cho mảng một chiều

Khai báo mảng a để lưu trữ 100 phần tử là các số nguyên: int a [100], khi đó máy sẽ cấp phát 200 byte để lưu trữ mảng a.

Hình ảnh mảng a gồm n phần tử được lưu trong bộ nhớ

Vị trí	0	1	2	3	4	n-3	n-2	n-1
Giá trị a[]	7	3	9	4	5	8	12	2
Tên phần tử	a[0]	a[1]	a[2]	a[3]	a[4]	a[n-3]	a[n-2]	a[n-1]

Xuất dữ liệu cho từng phần tử

```
a [0] = 7      printf(" % 4d", a[0]);  
a [1] = 3      printf(" % 4d", a[1]);  
a [2] = 9      printf(" % 4d", a[2]);  
a [3] = 4      printf(" % 4d", a[3]);  
.....  
a [n-1] = 2    printf(" % 4d", a[n-1]);
```

```
// Xuất từng phần tử của mảng
```

```
for (int i = 0 ; i < n ; i++)  
{  
    printf (" % 4d ", a[ i ]);  
}
```

7.2.5 Một vài thuật toán trên mảng một chiều

Bài toán 1: Tính tổng các phần tử trong mảng một chiều các số nguyên.

Khai báo mảng a để lưu trữ 100 phần tử là các số nguyên: int a [100]

Khi đó máy sẽ cấp phát 200 byte để lưu trữ mảng a

Hình ảnh mảng a gồm n phần tử được lưu trong bộ nhớ

Vị trí	0	1	2	3	4	n-3	n-2	n-1
Giá trị a[]	7	3	9	4	5	8	12	2
Tên phần tử	a[0]	a[1]	a[2]	a[3]	a[4]	a[n-3]	a[n-2]	a[n-1]

Tính tổng: khai báo một biến s để lưu trữ tổng

$$S = a[0] + a[1] + a[2] + \dots + a[n - 1]$$

Giải thuật:

Đi từ đầu mảng đến cuối mảng

```
for (int i= 0 ; i<n ; i++)
```

Cộng dồn các phần tử a[i] vào biến s

```
S= S + a[i] ;
```

Hàm cài đặt

```
long tinh tong (int a[ ], int n)
```

```
{
```

```
    long s =0 ;
```

```
    for (int i = 0 ; i<n ; i++)
```

```
        s=s+ a[i] ;
```

```
    return s;
```

```
}
```

Bài toán 2: Tìm phần tử âm đầu tiên có trong mảng một chiều các số nguyên.

Hình ảnh mảng a gồm n phần tử được lưu trong bộ nhớ

Vị trí	0	1	2	3	4	n-3	n-2	n-1
a[]	7	3	-9	4	5	8	12	2

Giải thuật:

Đi từ đầu mảng đến cuối mảng

```
for (int i= 0 ; i<n ; i++)
Kiểm tra phần tử a[i] < 0 đầu tiên
```

```
if (a[i] < 0)
```

Nếu gặp thì xuất giá trị a[i] và dừng chương trình.

```
return a[i] ;
```

Hàm cài đặt

```
int amdau (int a[ ], int n)
{
    for (int i = 0 ; i<n ; i++)
        if (a[i] < 0)
            return a[i] ;
    return 1; // Tại sao lại phải thêm lệnh return 1?
}
```

7.3 CHUỖI KÝ TỰ (MẢNG MỘT CHIỀU CÁC KÝ TỰ)

Chuỗi là một dãy ký tự dùng để lưu trữ và xử lý văn bản như từ, tên, câu. Trong ngôn ngữ C không có kiểu chuỗi và chuỗi được thể hiện bằng mảng các ký tự (có kiểu cơ sở char), được kết thúc bằng ký tự '\0' (còn được gọi là ký tự NULL trong bảng mã Ascii).

Các hằng chuỗi ký tự được đặt trong cặp dấu nháy kép " ".

Chú ý: Chuỗi được khai báo là một mảng các ký tự nên các thao tác trên mảng có thể áp dụng đối với chuỗi ký tự.

7.3.1 Cách khai báo chuỗi

❖ Khai báo chuỗi: **char < tên biến> [chiều dài tối đa chuỗi];**

Ví dụ: char Hoten [20];

Khai báo như trên là khai báo 1 chuỗi chứa tối đa 19 ký tự (còn 1 ký tự cuối của chuỗi chứa NULL)

❖ Vừa khai báo vừa gán giá trị: **char <Biến> [] = <"Hằng chuỗi">;**

Ví dụ: char chuỗi [] = " Kỹ thuật lập trình ";
 char chuoi [50]= "CONG HOA XA HOI CHU NGHIA VIET NAM";
 char name []= {'K','C','N','T','T','\0'};
 char ten[10]={'h','o','a','h','o','n','g','\0'};

khi đó:

ten[0]= 'h'; ten[1]= 'o'; ten[2]= 'a'; ten[3]= 'h'; ten[4]= 'o';
ten[5]= 'n'; ten[6]= 'g'; ten[7]= \0';

Lỗi khi tạo một chuỗi

1. Chú ý: Không sử dụng toán tử gán = để chép nội dung của một chuỗi này sang chuỗi khác.

```
char      a[4] = "hi";
char    b[4];
b = a;      //??? Máy báo lỗi
```

2. Không dùng toán tử == để so sánh nội dung hai

```
chuỗi char a[] = "hi";
char b[] = "there";
if(a==b) //??? Máy báo lỗi
{ }
```

3. Phép gán trong kiểu dữ liệu chuỗi như thế này là sai

```
char ten[10];
ten = "hoahong"
```

7.3.2 Các thao tác trên chuỗi ký tự

Cách 1: Nhập xuất chuỗi dùng thư viện < stdio.h>

❖ Nhập: scanf

Ví dụ: scanf ("%s", & Hoten);

Đối với hàm scanf khi gặp phím space, tab, new line, Enter thì dừng, cho nên chỉ dùng **hàm scanf để nhập chuỗi không có khoảng trắng**.

❖ Xuất:

printf (xuất chuỗi xong không xuống dòng)

Ví dụ printf ("%s", Hoten);

printf (xuất chuỗi xong có xuống dòng)

Ví dụ printf ("%s \n ", Hoten);

Cách 2: Nhập xuất chuỗi dùng thư viện <string.h>

❖ **Nhập:** gets (Hoten);

Tiếp nhận được space, tab, new line.

Gặp Enter thì dừng, phải khai báo hàm xóa bộ đệm bàn phím trước khi dùng hàm gets: **fflush (stdin)** hay **flushall ()**.

❖ **Xuất:** puts (Hoten); (Xuất chuỗi xong tự động xuống dòng)

7.3.3 Một số hàm xử lý chuỗi (trong <string.h>)

1. Hàm kbhit: kiểm tra bộ đệm bàn phím.

Cú pháp: int kbhit (void)

Hàm trả về giá trị khác không nếu bộ đệm bàn phím khác rỗng, trả về giá trị không nếu ngược lại.

2. Hàm strcat: dùng để nối hai chuỗi lại với nhau.

Cú pháp: char * strcat(char* s1, char* s2)

Hàm có công dụng ghép nối hai chuỗi s1 và s2 lại với nhau; kết quả ghép nối được chứa trong s1.

Ví dụ:

```
#include "stdio.h"
#include "string.h"
void main()
{
    char s1[50],s2[50];
    printf("\nNhập chuỗi 1: ");    gets(s1);
    printf("Nhập chuỗi 2: ");    gets(s2);
    strcat(s1,s2);
    printf("Xuất chuỗi 1: %s",s1);
```

```
    printf("\nXuat chuoi 2: %s",s2);
}
```

3. Hàm strchr: Tìm lần xuất hiện đầu tiên của ký tự trong chuỗi.

Cú pháp: *char* strchr (char* ch, int kt)*

Hàm có tác dụng tìm lần xuất hiện đầu tiên của ký tự kt trong chuỗi ch. Nếu tìm thấy hàm trả về địa chỉ của ký tự được tìm thấy trong chuỗi ch, trái lại hàm trả về giá trị NULL.

Ví dụ:

```
#include "stdio.h"
#include "string.h"
void main()
{
    char s[50], ch, *p;
    printf("\nNhap chuoi: ");
    gets(s);
    printf("Nhap ky tu: ");
    ch=getche();
    p=strchr(s,ch);
    if (p != NULL) printf("\nchi so cua ky tu: %d", (int)(p-s));
    else printf("\nKhong tim thay!");
}
```

4. Hàm strcmp: so sánh hai chuỗi.

Cú pháp: *int strcmp (char* s1, char* s2)*

Hàm có công dụng so sánh hai chuỗi s1 và s2.

- Nếu hàm trả về giá trị <0 thì chuỗi s1 nhỏ hơn chuỗi s2.
- Nếu hàm trả về giá trị 0 nếu chuỗi s1 bằng chuỗi s2.
- Nếu hàm trả về giá trị >0 thì chuỗi s1 lớn hơn chuỗi s2.

5. So sánh chuỗi - Hàm strcmp()

Hàm này thực hiện việc so sánh trong n ký tự đầu tiên của 2 chuỗi s1 và s2, giữa chữ thường và chữ hoa không phân biệt.

Cú pháp: *int strcmp (const char *s1, const char *s2)*

Kết quả trả về tương tự như kết quả trả về của hàm strcmp()

6. Hàm strcpy: sao chép chuỗi.

Hàm được dùng để sao chép toàn bộ nội dung của chuỗi nguồn vào chuỗi đích.

Cú pháp: char *strcpy (char *Des, const char *Source)

Ví dụ:

```
#include "stdio.h"
#include "string.h"
void main()
{
    char s[50];
    strcpy(s,"Truong Dai hoc Ky thuat");
    printf("\nXuat chuoi: %s",s);
}
```

7. Hàm strncpy (): Sao chép một phần chuỗi

Hàm này cho phép chép n ký tự đầu tiên của chuỗi nguồn sang chuỗi đích.

Cú pháp: char *strncpy(char *Des, const char *Source, size_t n)

8. Hàm strchr(): Trích một phần chuỗi

Để trích một chuỗi con của một chuỗi ký tự bắt đầu từ một ký tự được chỉ định trong chuỗi cho đến hết chuỗi, ta sử dụng hàm strchr().

Cú pháp: char *strchr (const char *str, int c)

Ghi chú:

Nếu ký tự đã chỉ định không có trong chuỗi, kết quả trả về là NULL.

Kết quả trả về của hàm là một con trỏ, con trỏ này chỉ đến ký tự c được tìm thấy đầu tiên trong chuỗi str.

9. Tìm kiếm nội dung chuỗi - hàm strstr()

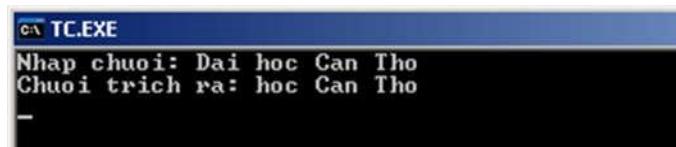
Hàm strstr() được sử dụng để tìm kiếm sự xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1.

Cú pháp: char *strstr(const char *s1, const char *s2)

Kết quả trả về của hàm là một con trỏ chỉ đến phần tử đầu tiên của chuỗi s1 có chứa chuỗi s2 hoặc giá trị NULL nếu chuỗi s2 không có trong chuỗi s1.

Ví dụ: Viết chương trình sử dụng hàm strstr() để lấy ra một phần của chuỗi gốc bắt đầu từ chuỗi "hoc".

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
int main()
{
    char Chuoi[255], *s ;
    printf("Nhập chuỗi: ");gets(Chuoi);
    s=strstr(Chuoi,"hoc");
    -printf("Chuỗi trích ra: ");puts(s);
    getch();
    return 0;
}
```



10. Lấy chiều dài chuỗi với hàm strlen()

Cú pháp: *int strlen (const char *s);*

```
#include <stdio.h>
#include <string.h>
void main()
{
    char string [ ] = "Borland International";
    printf("%d\n", strlen(string)); // kết quả 21
    getch ();
}
```

Ví dụ: Gán một chuỗi vào chuỗi khác

Gán từng ký tự trong chuỗi.

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

```

{
    clrscr() ;
        char newstr [35];
        char str[] = " Trường Đại học Công Nghệ Tp.HCM";
        for(int i = 0; i<strlen(str); i++)
            newstr[i] = str[i];
        newstr[i] = '\0';
        getch () ;
}

```

11. Đổi một ký tự thường thành ký tự hoa - Hàm toupper()

Hàm toupper() (trong ctype.h) được dùng để chuyển đổi một ký tự thường thành ký tự hoa.

Cú pháp: **char toupper (char c)**

12. Đổi chuỗi chữ thường thành chuỗi chữ hoa - Hàm strupr()

Hàm strupper() được dùng để chuyển đổi chuỗi chữ thường thành chuỗi chữ hoa, kết quả trả về của hàm là một con trỏ chỉ đến địa chỉ chuỗi được chuyển đổi.

Cú pháp: **char *strupr(char *s)**

Ví dụ: Viết chương trình nhập vào một chuỗi ký tự từ bàn phím. Sau đó sử dụng hàm strupr() để chuyển đổi chúng thành chuỗi chữ hoa.

```

#include<conio.h>
#include<stdio.h>
#include<string.h>
int main()
{
    char Chuoi[255],*s;
    printf("Nhập chuỗi: "); gets(Chuoi);
    s=strupr(Chuoi) ;
    printf("Chuỗi chữ hoa: "); puts(s);
    getch();
    return 0;
}

```

}

13. Đổi chuỗi chữ hoa thành chuỗi chữ thường - Hàm strlwr()

Muốn chuyển đổi chuỗi chữ hoa thành chuỗi toàn chữ thường, ta sử dụng hàm strlwr(), các tham số của hàm tương tự như hàm strupr()

Cú pháp: **char *strlwr (char *s)**

14. Đổi từ chuỗi ra số, hàm atoi(), atof(), atol() (trong stdlib.h)

Để chuyển đổi chuỗi ra số, ta sử dụng các hàm trên.

Cú pháp:

int atoi(const char *s): chuyển chuỗi thành số nguyên

long atol(const char *s): chuyển chuỗi thành số nguyên dài float

atof(const char *s): chuyển chuỗi thành số thực

Nếu chuyển đổi không thành công, kết quả trả về của các hàm là 0.

15. char* strtok (char *s1, const char *s2)

Xem s1 là 1 loạt chuỗi con, ngăn cách nhau bởi 1 hay nhiều ký tự có trong s2.

Ví dụ:

```
#include <string.h>
void main()
{
    char s[80], *p ;
    gets(s);
    p = strtok(s," ");
    if (p) printf("%s", p);
    while(p)
    {
        p = strtok(NULL," ");
        if (p) printf("%s", p);
    }
}
```



16. Đảo ngược chuỗi: char* strrev(char *s)

Ngoài ra, thư viện string.h còn hỗ trợ các hàm xử lý chuỗi khác, ta có thể đọc thêm trong phần trợ giúp.

CÂU HỎI ÔN TẬP

Câu 1: Viết hàm nhập vào một mảng một chiều các số nguyên gồm n phần tử ($0 < n \leq 100$).

Hướng dẫn

```
#include <conio.h>
#include <stdio.h>
#define MAX 100
/*-----
 *Hàm nhập số lượng phần tử cho mảng
 */
void NhapSL(int &n)
{
    do{
        printf ("Nhập số phần tử 0<sl<100: ");
        scanf (" %d ", &n);
    }while (n<=0 || n>100);
}
/*-----
 *Hàm nhập giá trị cho từng phần tử trong mảng */
void NhapMang (int a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        printf (" a[%d] = ", i);
        scanf (" %d ", &a[i]);
    }
}
/*-----
```

Câu 2: Viết hàm nhập vào một mảng một chiều các số thực gồm n phần tử ($0 < n \leq 100$).

Hướng dẫn

```

#include <conio.h>
#include <stdio.h>
#define MAX 100
/*-----
*Hàm nhập số lượng phần tử cho mảng
*/
void NhapSL(int &n)
{
    do{
        printf ("Nhập số phần tử 0<n<100: ");
        scanf (" %d ", &n);
    }while (n<=0 || n>100);
}
/*-----
*Hàm nhập giá trị cho từng phần tử trong mảng */
void NhapMang (float a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        printf (" a[%d] = ", i);
        scanf (" %f ", &a[i]);
    }
}
/*-----
```

Câu 3: Viết hàm xuất mảng số nguyên n phần tử vừa nhập ở trên.

Hướng dẫn

```

void XuatMang (int a[], int n)
{
    printf ("\nMảng gồm các phần tử:\n ");
    for (int i = 0; i < n; i++)
        printf (" %5d", a[i]);
}
```

Câu 4: Viết hàm xuất mảng số thực n phần tử vừa nhập ở trên.

Hướng dẫn

```
void XuatMang (float a[], int n)
{
    printf ("\nMang gom cac phan tu:\n ");
    for (int i = 0; i < n; i++)
        printf (" %8.2f ", a[i]);
}
```

Câu 5: Tính tổng các phần tử có trong mảng.

Hướng dẫn

```
long tinh tong ( int a[ ], int n )
{
    long s=0;
    for( int i=0 ; i<n ; i++ )
        s=s+a[i];
    return s;
}
```

Câu 6: Tính tổng các phần tử chẵn có trong mảng.

Câu 7: Tính tổng các phần tử lẻ có trong mảng.

Câu 8: Tính tổng các phần tử nguyên tố có trong mảng.

Câu 9: Tìm phần tử chẵn đầu tiên có trong mảng

Câu 10: Tìm phần tử lẻ đầu tiên có trong mảng

Câu 11: Tìm phần tử nguyên tố đầu tiên có trong mảng

Câu 12: Tìm phần tử chẵn cuối cùng có trong mảng

Câu 13: Tìm phần tử chính phương cuối cùng có trong mảng

Câu 14: Tìm giá trị lớn nhất của mảng

Hướng dẫn

```
int timmax( int a[ ], int n )
{
```

```
int max = a [ 0 ] ;  
for( int i=1; i<n ; i++)  
    if ( max < a[ i ] )  
        max = a[ i ] ;  
return max;  
}
```

Câu 15: Đếm số phần tử chẵn có trong mảng

Câu 16: Đếm số phần tử có giá trị là x có trong mảng

Câu 17: Đếm số phần tử lớn nhất có trong mảng

Câu 18: In ra vị trí của phần tử lớn nhất đầu tiên có trong mảng

Câu 19: In ra vị trí của phần tử có giá trị là x cuối cùng có trong mảng

Câu 20: Thêm một phần tử vào đầu mảng.

Câu 21: Thêm một phần tử vào cuối mảng.

Câu 22: Thêm một phần tử vào vị trí x trong mảng.

Câu 23: Xóa phần tử chẵn đầu tiên.

Câu 24: Xóa tất cả các phần tử lớn nhất trong mảng

Câu 25: Sắp xếp mảng tăng dần

Câu 26: Thêm một phần tử có giá trị là x vào trong mảng sao cho mảng vẫn có thứ tự tăng dần.

Câu 27: Đảo ngược mảng

Câu 28: Nhập chuỗi

Hướng dẫn

```
void nhapchuoi ( char x[ ] )  
{  
    flushall ( ) ;  
    printf ( " nhap chuoi : " );  
    gets ( x );  
}
```

Câu 29: Xuất chuỗi

Câu 30: Đếm số ký tự 'a' có trong chuỗi

Câu 31: Cắt khoảng trắng có trong chuỗi

Câu 32: Đếm khoảng trắng trong chuỗi.

Câu 33: Đếm số từ có trong chuỗi

Câu 34: Sắp xếp chuỗi tăng dần

BÀI 8: KIỂU DỮ LIỆU CÓ CẤU TRÚC

Sau khi học xong bài này, học viên có thể:

- *Nắm được các khái niệm cơ bản về kiểu dữ liệu có cấu trúc;*
- *Biết nhập, xuất dữ liệu có cấu trúc cho một phần tử;*
- *Biết nhập, xuất dữ liệu có cấu trúc và lưu trên mảng một chiều;*
- *Cách tìm kiếm và sắp xếp dữ liệu có cấu trúc trên mảng một chiều;*
- *Đi sâu vào các giải thuật trên mảng một chiều như tìm kiếm, sắp xếp, thêm phần tử, xóa phần tử theo từng thành phần của kiểu dữ liệu có cấu trúc.*

8.1 KHÁI NIỆM

Kiểu cấu trúc (Structure) là một kiểu dữ liệu do người dùng tự định nghĩa, là kiểu dữ liệu bao gồm nhiều thành phần, mỗi thành phần có một kiểu dữ liệu khác nhau, mỗi thành phần được gọi là một trường (field).

Sự khác biệt giữa kiểu cấu trúc và kiểu mảng là: các phần tử của mảng có cùng kiểu dữ liệu còn các phần tử của kiểu cấu trúc có thể có các kiểu dữ liệu khác nhau.

Hình ảnh của kiểu cấu trúc được minh họa:

1	2	3	4	5
Field	Field	Field	Field	Field

Đây là kiểu cấu trúc có 5 trường

Còn kiểu mảng có dạng

0	1	2	3	4	5	6
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]

8.2 CÁCH KHAI BÁO KIỂU CẤU TRÚC

❖ Cú pháp

```
struct <Tên cấu trúc>
{
    <Kiểu> <Trường 1> ;
    <Kiểu> <Trường 2> ;
    .....
    <Kiểu> <Trường n> ;
};

typedef struct <Tên cấu trúc> <tên mới>;
```

Trong đó:

1. **<Tên cấu trúc>**: là một tên được đặt theo quy tắc đặt tên của danh biểu; tên này mang ý nghĩa sẽ là tên kiểu cấu trúc.
2. **<Kiểu> <Trường i>** (*i* = 1... n): mỗi trường trong cấu trúc có dữ liệu thuộc kiểu gì (tên của trường phải là một tên được đặt theo quy tắc đặt tên của danh biểu).

Ví dụ 1: Khai báo một kiểu struct có tên **NgayThang** có các thành phần là: ngày, tháng, năm, và đặt tên mới là Date.

```
struct NgayThang
{
    int ngay;
    int thang;
    int nam;
};

typedef struct NgayThang Date ;
```

Ví dụ 2: Khai báo một kiểu struct có tên **DiemThi** có các thành phần là: Mã số sinh viên, Mã số môn học, Lần thi, điểm thi.

```
struct DiemThi
{
    char masv[8]; /* mã số sinh viên */
    char mamh[5]; /* mã số môn học */
    int lanthi; /* lần thi */
    float diem; /* điểm thi */
};

typedef struct DiemThi Diem ;
```

Ví dụ 3: Khai báo một kiểu struct để lưu thông tin của một sinh viên bao gồm: mã số sinh viên, họ lót, tên, ngày sinh, nơi sinh, địa chỉ. Trong đó, ngày sinh là kiểu struct đã được định nghĩa trước đó.

```
typedef struct SinhVien
{
    char masv[8];
    char holot[30];
    char ten[10];
Date ngaysinh;
    char noisinh[50];
    char diachi[50];
} SV;
```

❖ **Khai báo biến có kiểu cấu trúc**

Ví dụ: Khai báo 1 biến x để lưu thông tin của 1 sinh viên:

```
SV x;
```

8.3 TRUY CẬP VÀO TỪNG PHẦN TỬ CỦA CẤU TRÚC

❖ **Cú pháp:** Thông qua dấu chấm để truy cập đến từng thành phần của struct

<tên biến struct>.<tên thành phần của struct>

Ví dụ 1: Để xuất ra màn hình tên của sinh viên x, ta thực hiện câu lệnh sau:

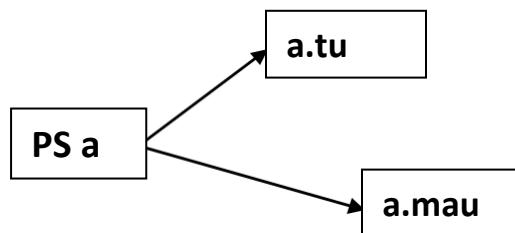
```
printf("%s", x.ten);
```

Ví dụ 2: Định nghĩa cấu trúc dữ liệu cho phân số

```
struct phanso
{
    int tu;
    int mau ;
};

typedef struct phanso PS;
```

Khi đó Nếu ta khai báo một biến a có kiểu dữ liệu là phân số thì a gồm 2 thành phần: a.tu, a.mau.



8.4 NHẬP, XUẤT DỮ LIỆU CHO KIỂU DỮ LIỆU CÓ CẤU TRÚC

8.4.1 Nhập dữ liệu

Để nhập dữ liệu cho biến kiểu cấu trúc, ta phải nhập cho từng thành phần của cấu trúc.

Ví dụ: Nhập dữ liệu cho một phân số có cấu trúc khai báo như trên:

```
void nhapphanso (PS &x)
```

```
{
```

```
    printf("Nhập tu so:");
```

```
    scanf("%d",&x.tu);
```

```
    do{
```

```
        printf("Nhập mau so khac 0:");
```

```
        scanf("%d",&x.mau);
```

```
    }while (x.mau ==0) ;
```

```
}
```

8.4.2 Xuất dữ liệu

Để xuất dữ liệu cho biến cấu trúc, ta xuất dữ liệu trong từng thành phần của cấu trúc.

Ví dụ: Xuất dữ liệu cho một phân số:

```
void xuatphanso (PS x) // VD 3/5
{
    printf ("phan so:%d / %d ", x.tu, x.mau);
}
```

8.5 MẢNG CẤU TRÚC

Bài toán minh họa: Viết chương trình

1. Nhập vào một dãy phân số.
2. Xuất ra dãy phân số vừa nhập.
3. Tính tổng các phân số có trong dãy.
4. Tìm phân số lớn nhất có trong dãy.
5. Đếm số phần tử lớn nhất có trong dãy.

Hình ảnh mảng một chiều các phân số

	0	1	2	3	4	5	6	7	8
a[]	9/1	6/2	1/2	5/3	1/7	1/3	1/2	8/2	3/2

Trong đó

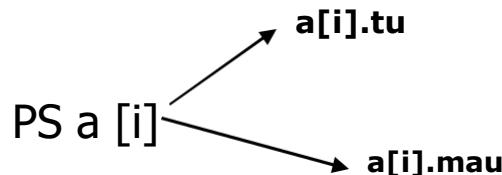
$$a[0] = 9/1$$

$$a[1] = 6/2$$

$$a[2] = 1/2$$

.....

$$a[8] = 3/2$$



Nhập mảng phân số

// hàm cài đặt

```
void nhapmang (PS a[], int n)
{
    for (int i=0 ; i<n ; i++)
        nhaphanso(a[i]);
}
```

Xuất mảng phân số

// hàm cài đặt

```
void xuatmang (PS a[], int n)
{
    for (int i=0 ; i<n ; i++)
        xuatphanso(a[i]);
}
```

//Hàm tìm ước chung lớn nhất-----

int UCLN(int a, int b)

```
{
    a=abs(a); b=abs(b); //lấy trị tuyệt đối
    Lặp chừng nào a ≠ b
        Nếu a>b thì a=a-b;
        Nếu b>a thì b=b-a;

    return a;
}
```

//Hàm rút gọn phân số-----

void RutGon(PS &x)

```
{
    if (x.tu==0) return; //nếu phân số có tử = 0 thì không cần rút gọn
    // lấy giá trị ucln của tử và mẫu của phân số x bằng cách gọi hàm UCLN ();
```

```
//tính lại x.tu và x.mau;  
}
```

//Hàm tính tổng hai phân số-----

PS Tong2ps(PS x, PS y)

```
{  
    PS tong; //phân số lưu kết quả sau khi  
    tong.tu= ....?;  
    tong.mau=..?;  
    RutGon(tong);  
    return tong;
```

```
}
```

//Hàm so sánh hai phân số-----

int SoSanh(PS x, PS y)

```
{  
}  
}
```

CÂU HỎI ÔN TẬP

Câu 1: Viết hàm Nhập vào một phân số

Hướng dẫn

```
void Nhap1ps (PS &x)
```

```
{
```

```
    printf("Nhập tu số: "); scanf("%d", &x.tu);
    //nhập mẫu số phải khác 0, nhập sai bắt nhập lại
    do{
        printf("Nhập mau số: "); scanf("%d", &x.mau);
        if ( x.mau==0)
            printf( "Mau so phai khac 0. Nhap sai! Nhap lai" );
        }while (x.mau==0);
}
```

Câu 2: Viết hàm Nhập vào một dãy phân số

Hướng dẫn

Cách 1

```
void nhapdayps( PS a[ ] , int n )
{
    for(int i = 0 ; i<n ; i++ )
    {
        printf ( " nhap tu : " );
        scanf ( " %d " , & a[ i ] . tu );
        printf ( " nhap mau : " );
        scanf ( " %d " , & a[ i ] . mau );
    }
}
```

Cách 2

```
void nhapdayps( PS a[ ] , int n )
{
    for(int i = 0 ; i<n ; i++ )

        Nhaph1ps ( a[i] );
}
```

Câu 3: Viết hàm xuất một phân số

Hướng dẫn

```
void xuat1phanso(PS x)
1. {
2.     printf("phan so :%d/%d",x.tu,x.mau);
3. }
```

Câu 4: Viết hàm xuất một dãy phân số

Hướng dẫn

```
void xuatdayps( PS a[ ] , int n )
{
    for(int i = 0 ; i<n ; i++ )
    {
        xuat1phanso(a[i]);
        printf(" ");
    }
}
```

Câu 5: Viết hàm tìm phân số lớn nhất trong dãy phân số

Câu 6: Viết hàm tính tổng các phân số có trong dãy

Câu 7: Viết hàm nhập dữ liệu cho một sinh viên, thông tin về một sinh viên gồm có:

1. Mã số sinh viên (chuỗi 10 ký tự).
2. Tên (là chuỗi tối đa 10 ký tự).

3. Ngày tháng năm sinh (theo kiểu int. Mở rộng dd/mm/yy).
4. Giới tính (Nam hoặc Nữ).
5. Lớp (chuỗi 7 ký tự trong đó 2 ký tự đầu là năm vào học, 1 ký tự tiếp là bậc học (D: Đại học, C: Cao đẳng), 2 ký tự tiếp là ngành học (TH: Tin Học, KT: Kế Toán, QT: Điện tử, ĐT: Điện tử...)).
6. Điểm toán, lý, tin (Kiểu số thực).

Hướng dẫn

1/. Khai báo cấu trúc sinh viên

```
struct sinhvien
{ char ten[11];
  char mssv [11];
  char lop [8];
  char gioitinh;
  int ntns;
  float toan,ly,tin, DTB ;
};

typedef struct sinhvien  sv;
// viết hàm nhập 1 sinh viên
```

Cách 1

```
void nhap1sv( sv &a)
{
  flushall();
  printf("nhap ten sinh vien :");    gets (a. ten);
  printf("nhap ma so sinh vien :");   gets(a. mssv);
  printf("nhap lop :");      scanf( "%s", &a. lop );
  printf (" nhap nam sinh :") ; scanf( "%d", &a. namsinh );
  flushall();
  printf(" gioi tinh ( Nam y/nu x ) :"); scanf( "%c", &a. gioitinh );
  printf("nhap diem toan:");   scanf("%f ", &a.toan) ;
  printf("nhap diem ly:");     scanf("%f ", &a.ly ) ;
  printf("nhap diem tin:");    scanf("%f ", &a.tin );
```

```
a.DTB= ( a.toan +a.ly +a.tin ) /3 ;  
}
```

Cách 2

```
void nhap1sv( sv &a)  
{ flushall();  
    printf("nhap ten sinh vien :"); gets (a. ten);  
    printf("nhap ma so sinh vien :"); gets(a. mssv);  
    printf("nhap lop :"); scanf( "%s", &a. lop );  
    printf (" nhap nam sinh :"); scanf( "%d", &a. namsinh );  
    printf(" gioi tinh ( Nam y/nu x ) :"); a. gioitinh= getche();  
    float t;  
    printf("nhap diem toan:"); scanf("%f ", &t) ; a.toan= t ;  
    printf("nhap diem ly:"); scanf("%f ", &t); a.ly = t ;  
    printf("nhap diem tin:"); scanf("%f ", &t); a.tin = t;  
    a.DTB= ( a.toan +a.ly +a.tin ) /3 ;  
}
```

Câu 8: Viết hàm xuất dữ liệu một sinh viên với thông tin vừa nhập ở trên.

Hướng dẫn

```
void xuat1sv( sv a)  
{  
    printf("Ten sinh vien :"); puts(a. ten);  
    printf(" Ma so sinh vien :"); puts(a. mssv);  
    printf(" Lop :"); puts(a. lop);  
    if(a.gioitinh=='x' )  
        printf(" gioi tinh : Nu ");  
    else  
        printf(" gioi tinh : Nam ");  
    printf(" ntns : %d, a. ntns);  
  
    printf("diem toan: %f " , a.toan ) ;  
    printf("diem ly: %f " , a.ly ) ;
```

```
    printf("diem tin: %f ", a.tin ) ;
    printf("diem TB : %f" , a.DTB ) ;
}
```

Câu 9: Viết hàm nhập danh sách sinh viên, lưu trên mảng một chiều.

Câu 10: Viết hàm xuất danh sách sinh viên.

Câu 11: Tìm sinh viên có tên là " X ". ("X" do người dùng nhập vào)

Hướng dẫn

```
void timtensv (sv a[ ] , int n, char x[])
{
    int flag = 0 ;
    for(int i=0;i<n;i++)
        if ( strcmp ( a[i].ten, x )==0 )
    {
        xuat1sv (a[i] ) ;
        flag = 1 ;
    }
    if ( flag == 0 )
        printf (" ko co ten sv can tim ");
}
```

Câu 12: Đếm số lượng sinh viên có điểm trung bình >5.

Hướng dẫn

```
int DemsvDTB>5 (sv a[ ], int n)
{
    int dem =0;
    for(int i=0;i<n;i++)
    {
        if ( a[i].DTB > 5)
            dem++;
    }
    return dem;
}
```

Câu 13: Xuất danh sách sinh viên thuộc ngành công nghệ thông tin.

Hướng dẫn

Câu 14: Xuất danh sách sinh viên Nữ thuộc ngành công nghệ thông tin.

Hướng dẫn

Câu 15: Sắp xếp danh sách sinh viên tăng dần theo cột DTB.

Hướng dẫn

```
void InterchangeSort( sv a[ ], int n )
{
    sv tam ;
    for ( int i = 0 ; i<n-1 ; i++ )
        for ( int j =i+1; j < n ; j++ )
            if( a[ i ].DTB> a[ j ].DTB )
            {
                tam = a[i] ;
                a[i]  = a[j] ;
                a[j] = tam;
            }
}
```

TÀI LIỆU THAM KHẢO

1. Phạm Văn Ất, Kỹ thuật Lập trình C- cơ bản và nâng cao, NXB KH & KT - 2003.
2. Quách Tuấn Ngọc, Tin học căn bản, Nhà xuất bản giáo dục - 1997.
3. Hoàng Kiếm, Nguyễn Đức Thắng, Đinh Nguyễn Anh Dũng, Giáo trình Tin học Đại cương, Nhà xuất bản giáo dục - 1999.
4. Nguyễn Tân Trần Minh Khang, Bài giảng Kỹ Thuật Lập trình, Khoa Công Nghệ Thông Tin, Đại học Khoa học Tự nhiên
5. Nguyễn Thanh Thủy (chủ biên), Nhập môn lập trình ngôn ngữ C, Nhà xuất bản Khoa học kỹ thuật – 2000.
6. Trần Minh Thái, Bài giảng và bài tập Lập trình căn bản; Khoa Công Nghệ Thông Tin, Đại học Khoa học Tự nhiên
7. Mai Ngọc Thu, Giáo trình C. Khoa Công Nghệ Thông Tin, Đại học Kỹ Thuật Công Nghệ.
8. Brian W. Kernighan & Dennis Ritchie, The C Programming Language, Prentice Hall Publisher, 1988.
9. Giáo trình ứng dụng CNTT- cơ bản, Trung tâm tin học. Nguồn Internet.
10. Nguyễn Thúy Loan – Văn Thị Thiên Trang ,Giáo trình KTLT khoa CNTT- HUTECH