TRƯỜNG CAO ĐẲNG NGHỀ CÔNG NGHIỆP HÀ NỘI

Chủ biên: Vũ Thị Kim Phượng

Đồng tác giả: Nguyễn Thái Hà

GIÁO TRÌNH

LẬP TRÌNH CĂN BẢN

(Lưu hành nội bộ)

Hà Nội năm 2011

Tuyên bố bản quyền

Giáo trình này sử dụng làm tài liệu giảng dạy nội bộ trong trường cao đẳng nghề Công nghiệp Hà Nội

Trường Cao đẳng nghề Công nghiệp Hà Nội không sử dụng và không cho phép bất kỳ cá nhân hay tổ chức nào sử dụng giáo trình này với mục đích kinh doanh.

Mọi trích dẫn, sử dụng giáo trình này với mục đích khác hay ở nơi khác đều phải được sự đồng ý bằng văn bản của trường Cao đẳng nghề Công nghiệp Hà Nội

Chương I. Giới thiệu về ngôn ngữ C

1. Giới thiệu

a. Tổng quan về 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, Dennish 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, Dennish 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.

Ngôn ngữ C có những đặc điểm cơ bản sau:

o 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.

o 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.

o 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.

o 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.

o 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.

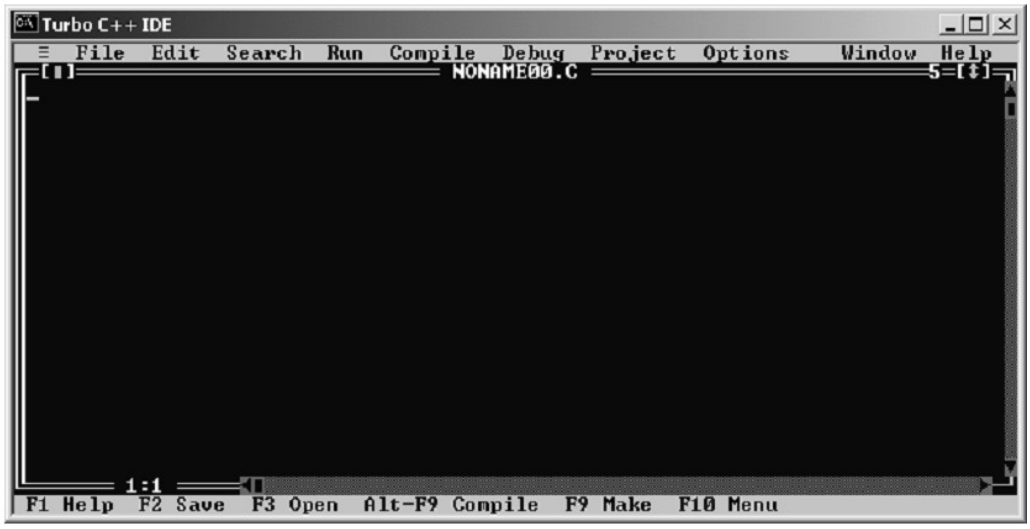
Ngày nay có một số ngôn ngữ lập trình cấp cao khác như C++, C#, … Đây là các ngôn ngữ lập trình hướng đối tượng và có thể xem là ngôn ngữ C nâng cấp. Do đó, toàn bộ những gì bạn học được trong ngôn ngữ C đều có thể áp dụng cho các ngôn ngữ nâng cấp đó.

b. Môi trường lập trình Turbo C

Turbo C là môi trường hỗ trợ lập trình C do hãng Borland cung cấp. Môi trường này cung cấp các chức năng như: soạn thảo chương trình, dịch, thực thi chương trình… Phiên bản được sử dụng ở đây là Turbo C 3.0.

+ Gọi Turbo C

Chạy Turbo C cũng giống như chạy các chương trình khác trong môi trường DOS hay Windows, màn hình sẽ xuất hiện menu của Turbo C có dạng 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.

Dòng dưới cùng ghi chức năng của một số phím đặc biệt. Chẳng hạn khi gõ phím F1 thì ta có được một hệ thống trợ giúp mà ta có thể tham khảo nhiều thông tin bổ ích

Muốn vào thanh menu ngang ta gõ phím F10. Sau đó dùng các phím mũi tên qua trái hoặc phải để di chuyển vùng sáng tới mục cần chọn rồi gõ phím Enter. Trong menu kéo xuống ta lại dùng các phím mũi tên lên xuống để di chuyển vùng sáng tới mục cần chọn rồi gõ Enter.

Ta cũng có thể chọn một mục trên thanh menu bằng cách giữ phím Alt và gõ vào một ký tự đại diện của mục đó (ký tự có màu sắc khác với

các ký tự khác). Chẳng hạn để chọn mục File ta gõ Alt-F (F là ký tự đại diện của File)

+ Soạn thảo chương trình mới

Muốn soạn thảo một chương trình mới ta chọn mục New trong menu File (File ->New)

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:

Các phím xem thông tin trợ giúp:

- F1: Xem toàn bộ thông tin trong phần trợ giúp.

- Ctrl/F1: Trợ giúp theo ngữ cảnh (tức là khi con trỏ đang ở trong một từ nào

đo, chẳng hạn int mà bạn gõ phím Ctrl-F1 thì bạn sẽ có được các thông tin về kiểu dữ liệu int)

Các phím di chuyển con trỏ trong vùng soạn thảo chương trình:

Phím	Ý nghĩa	Phím tắt (tổ hợp phím)
Enter	Đưa con trỏ xuống dòng	
Mũi tên đi lên	Đưa con trỏ lên hàng trước	Ctrl-E
Mũi tên đi xuống	Đưa con trỏ xuống hàng sau	Ctrl-X
Mũi tên sang trái	Đưa con trỏ sang trái một ký tự	Ctrl-S
Mũi tên sang phải	Đưa con trỏ sang phải một ký tự	Ctrl-D
End	Đưa con trỏ đến cuối dòng	
Home	Đưa con trỏ đến đầu dòng	
PgUp	Đưa con trỏ lên trang trước	Ctrl-R
PgDn	Đưa con trỏ xuống trang sau	Ctrl-C
	Đưa con trỏ sang từ bên trái	Ctrl-A
	Đưa con trỏ sang từ bên phải	Ctrl-F

Các phím xoá ký tự/ dòng:

Phím	Ý nghĩa	Phím tắt
Delete	Xoá ký tự tại ví trí con trỏ	Ctrl-G
BackSpace	Di chuyển sang trái đồng thời xoá ký tự đứng trước con trỏ	Ctrl-H
	Xoá một dòng chứa con trỏ	Ctrl-Y
	Xóa từ vị trí con trỏ đến cuối dòng	Ctrl-Q-Y
	Xóa ký tự bên phải con trỏ	Ctrl-T

Các phím chèn ký tự/ dòng:

Insert	Thay đổi viết xen hay viết chồng
Ctrl-N	Xen một dòng trống vào trước vị trí con trỏ

Sử dụng khối :

Khối là một đoạn văn bản chương trình hình chữ nhật được xác định bởi đầu khối là góc trên bên trái và cuối khối là góc dưới bên phải của hình chữ nhật. Khi một khối đã được xác định (trên màn hình khối có màu sắc khác chỗ bình thường) thì ta có thể chép khối, di chuyển khối, xoá khối... Sử dụng khối cho phép chúng ta soạn thảo chương trình một cách nhanh chóng. sau đây là các thao tác trên khối:

Phím tắt	Ý nghĩa
Ctrl-K-B	Đánh dấu đầu khối
Ctrl-K-K	Đánh dấu cuối khối
Ctrl-K-C	Chép khối vào sau vị trí con trỏ
Ctrl-K-V	Chuyển khối tới sau vị trí con trỏ
Ctrl-K-Y	Xoá khối
Ctrl-K-W	Ghi khối vào đĩa như một tập tin
Ctrl-K-R	Đọc khối (tập tin) từ đĩa vào sau vị trí con trỏ
Ctrl-K-H	Tắt/mở khối
Ctrl-K-T	Đánh dấu từ chứa chon trỏ
Ctrl-K-P	In một khối

Các phím, phím tắt thực hiện các thao tác khác:

Phím	Ý nghĩa	Phím tắt
F10	Kích hoạt menu chính	Ctrl-K-D, Ctrl-K-Q
F2	Lưu chương trình đang soạn vào đĩa	Ctrl-K-S
F3	Tạo tập tin mới	
Tab	Di chuyển con trỏ một khoảng đồng thời đẩy dòng văn bản	Ctrl-I
ESC	Hủy bỏ thao tác lệnh	Ctrl-U
	Đóng tập tin hiện tại	Alt-F3
	Hiện hộp thoại tìm kiếm	Ctrl-Q-F
	Hiện hộp thoại tìm kiếm và thay thế	Ctrl-Q-A
	Tìm kiếm tiếp tục	Ctrl-L

Ví dụ: Bạn hãy gõ đoạn chương trình sau:

```
#include <stdio.h>
#include<conio.h>
int  main ()
{
        char ten[50];
        printf("Xin cho biet ten cua ban !");
        scanf("%s",ten);
        printf("Xin chao ban %s",ten);
        getch();
         return 0;
}
```

+ Ghi chương trình đang soạn thảo vào đĩa

Sử dụng File/Save hoặc gõ phím F2. Có hai trường hợp xảy ra:

- Nếu chương trình chưa được ghi lần nào thì một hội thoại sẽ xuất hiện cho phép bạn xác định tên tập tin (FileName). Tên tập tin phải tuân thủ quy cách đặt tên của DOS và không cần có phần mở rộng (sẽ tự

động có phần mở rộng là .C hoặc .CPP sẽ nói thêm trong phần Option). Sau đó gõ phím Enter.

- Nếu chương trình đã được ghi một lần rồi thì nó sẽ ghi những thay đổi bổ sung lên tập tin chương trình cũ.

Chú ý: Để đề phòng mất điện trong khi soạn thảo chương trinh thỉnh thoảng bạn nên gõ phím F2.

Quy tắc đặt tên tập tin của DOS: Tên của tập tin gồm 2 phần: Phần tên và phần mở rộng.

o 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.

o Phần mở rộng: phần này dài tối đa 3 ký tự.

Ví dụ: Ghi chương trình vừa nhập ở trên với tên CHAO.C

+ Thực hiện chương trình

Để thực hiện chương trình hãy dùng Ctrl-F9 (giữ phím Ctrl và gõ phím F9).

Ví dụ: Thực hiện chương trình vừa soạn thảo xong và quan sát trên màn hình để

thấy kết quả của việc thực thi chương trình sau đó gõ phím bất kỳ để trở lại với Turbo.

+ Mở một chương trình đã có trên đĩa

Với một chương trình đã có trên đĩa, ta có thể mở nó ra để thực hiện hoặc sửa chữa bổ sung. Để mở một chương trình ta dùng File/Open hoặc gõ phím F3.Sau đó gõ tên tập tin vào hộp File Name hoặc lựa chọn tập tin trong danh sách các tập tin rồi gõ Enter.

Ví dụ: Mở tập tin CHAO.C sau đó bổ sung để có chương trình mới như sau:

#include <stdio.h>

```
#include<conio.h>
int main ()
{
        char ten[50];
        printf("Xin cho biet ten cua ban !");
        scanf("%s",ten);
        printf("Xin chao ban %s\n ",ten);
        printf("Chao mung ban den voi Ngon ngu lap trinh C");
        getch();
        return 0;
}
```

Ghi lại chương trình này (F2) và cho thực hiện (Ctrl-F9). Hãy so sánh xem có

gì khác trước?

+ Thoát khỏi Turbo C và trở về DOS (Windows)

Dùng File/Exit hoặc Alt-X.

2. Khởi động và thoát chương trình

- Khởi động vào chương trình Turbo C tương tự như vào các chương trình khác trên Windows bằng cách Click vào biểu tượng shortcut Turbo C trên Desktop, hoặc vào Window Explorer, chọn ổ đĩa chứa thư mục Turbo C, vào thư mục Turbo C, vào thư mục BIN, khởi động tập tin TC.EXE.
- Thoát khỏi chương trình: Dùng File/Exit hoặc ấn tổ hợp phím Alt-X.

Chương II. Các thành phần trong ngôn ngữ C

1. Từ khóa

Từ khóa (Keyword) là các từ dành riêng (reserved words) của C mà người lập trình có thể sử dụng nó trong chương trình, mỗi từ khóa có chức năng nhất định và khi sử dụng phải viết đúng cú pháp. Ta không được dùng từ khóa để đặt cho các biến, hàm, tên chương trình con. Từ khóa được chia thành các loại sau đây:

- Các từ khóa dùng để khai báo

```
const       enum        extern      register
signed      static      struct      typedef
union       unsigned    volatile
```

- Các từ khóa về kiểu dữ liệu

```
char        double      float       int
long        short       void
```

- Các từ khóa điều khiển

```
case        default     else        if
switch
```

- Các từ khóa vòng lặp

```
do          for         while
```

- Các từ khóa điều khiển

```
break       continue    goto        return
```

- Các từ khóa khác

```
asm         goto        sizeof
```

Các từ khóa phải viết bằng chữ thường

2. Tên

Tên hay định danh là khái niệm rất quan trọng trong quá trình lập trình, nó không những thể hiện rõ ý nghĩa trong chương trình mà còn dùng

để xác định các đại lượng khác nhau khi thực hiện chương trình. Tên là một dãy ký tự dùng để chỉ tên một hằng số, hằng ký tự, tên một biến, một kiểu dữ liệu, một hàm. Tên không được trùng với các từ khóa và được tạo thành từ các chữ cái và các chữ số. Chiều dài tối đa của tên là 32 ký tự.

Tên biến hợp lệ là một chuỗi ký tự liên tục gồm: ***Ký tự chữ, số và dấu gạch dưới***. Ký tự đầu của tên phải là ***chữ hoặc dấu gạch dưới***. Khi đặt tên không được đặt trùng với các từ khóa.

Ví dụ 1 :

Các tên đúng: delta, a_1, Num_ODD, Case

Các tên sai:

3a_1 (ký tự đầu là số)

num-odd (sử dụng dấu gạch ngang)

int (đặt tên trùng với từ khóa)

del ta (có khoảng trắng)

f(x) (có dấu ngoặc tròn)

Lưu ý: Trong C, tên phân biệt chữ hoa, chữ thường

Ví dụ 2 : number khác Number

case khác Case

(case là từ khóa, do đó bạn đặt tên là Case vẫn đúng)

3. Kiểu dữ liệu

a. Các kiểu dữ liệu cơ bản:

Có 4 kiểu dữ liệu cơ bản trong C là: kiểu số nguyên, kiểu số thực, kiểu luận lý, kiểu ký tự.

Kiểu số nguyên: là các kiểu dữ liệu mà giá trị của nó là số nguyên. Dữ liệu kiểu số nguyên lại chia ra thành hai loại như sau:

• Các số nguyên có dấu (signed) để chứa các số nguyên âm hoặc dương.

Bao gồm kiểu char, int, short, long.

• Các số nguyên không dấu (unsigned) để chứa các số nguyên dương (kể cả số 0). Bao gồm kiểu unsigned char, unsigned int, unsigned short, unsigned long.

Kiểu số thực: Đây là các kiểu dữ liệu mà giá trị của nó là số thực. Trong C định nghĩa các kiểu số thực chuẩn như sau: *float, double*. Kiểu *float* là kiểu số thực có độ chính xác đơn (single-precision floating-point), kiểu *double* là kiểu số thực có độ chính xác kép (double-precision floating-point). Số thực kiểu *float* có độ chính xác là 6 chữ số sau dấu thập phân, còn kiểu số thực *double* có độ chính xác là 15 chữ số sau dấu thập phân, do vậy khi sử dụng nếu yêu cầu giá trị lớn, độ chính xác cao thì nên dùng *double*, ngược lại dùng *float.*

Kiểu luận lý: Trong C không hỗ trợ kiểu luận lý tường minh mà chỉ ngầm hiểu một cách không tường minh như sau:

• false (sai) là giá trị 0.

• true (đúng) là giá trị khác 0, thường là 1.

Các ngôn ngữ lập trình nâng cấp khác của C như C++ định nghĩ kiểu luận lý tường minh có tên là bool (2 giá trị false/true).

Kiểu ký tự: Đây chính là kiểu dữ liệu số nguyên char có độ lớn 1 byte và miền giá trị là 256 ký tự trong bảng mã ASCII.

TT	Kiểu dữ liệu (Type)	Kích thước (Length)	Miền giá trị (Range)
1	unsigned char	1 byte	0 đến 255
2	char	1 byte	– 128 đến 127
3	enum	2 bytes	– 32,768 đến 32,767
4	unsigned int	2 bytes	0 đến 65,535
5	short int	2 bytes	– 32,768 đến 32,767
6	int	2 bytes	– 32,768 đến 32,767
7	unsigned long	4 bytes	0 đến 4,294,967,295
8	long	4 bytes	– 2,147,483,648 đến 2,147,483,647
9	float	4 bytes	$3.4 * 10^{-38}$ đến $3.4 * 10^{38}$
10	double	8 bytes	$1.7 * 10^{-308}$ đến $1.7 * 10^{308}$
11	long double	10 bytes	$3.4 * 10^{-4932}$ đến $1.1 * 10^{4932}$

b. Kiểu enum

Kiểu enum trong ngôn ngữ lập trình C là một loại kiểu liệt kê dùng để khai báo các biến có chứa các đối tượng kiểu đếm được có giá trị thuộc một miền thứ tự được chỉ rõ trong lúc khai báo. Để tạo ra một dữ liệu kiểu enum ta sử dụng câu lệnh có cú pháp sau đây:

Enum ten_kieu{danh sách các phần tử}

Trong đó ten_kieu là tên của kiểu dữ liệu liệt kê mới vừa được tạo ra, danh sách các phần tử là các giá trị liệt kê mà các biến có thể nhận được, các phần tử phân cách nhau bởi dấu phẩy.

Ví dụ để làm việc với các ngày trong tuần ta có thể dùng kiểu Weekday như sau:

Enum Weekday {SUNDAY, MONDAY, TUESDAY, WEDSDAY, THURSDAY, FRIDAY, SATURDAY}Day1;

Enum Weekday Day2;

Khi đó Day1, Day2 là các biến kiểu Weekday và chúng có thể nhận các giá trị đã được liệt kê. Các câu lệnh sau đây là hợp lệ:

Day1=SUNDAY;

Day2=FRIDAY;

Thực chất các biến kiểu enum của ngôn ngữ lập trình C được coi là biến nguyên, chúng được cấp phát 2 byte bộ nhớ và có thể nhận một giá trị nguyên nào đó. Mỗi khi một dữ liệu kiểu enum được định nghĩa ra thì các phần tử trong **danh sách các phần tử** sẽ được gán cho các giá trị nguyên liên tiếp bắt đầu từ 0. Ví dụ như kiểu Weekday ở trên thì SUNDAY sẽ có giá trị là 0, MONDAY sẽ có giá trị là 1,…Do đó hai câu lệnh sau đây có kết quả giống nhau:

Day1=0; và *Day1*=SUNDAY;

c. Kiểu tự định nghĩa

Trong ngôn ngữ lập trình C ta có thể tự định nghĩa ra các kiểu dữ liệu của riêng mình bằng cách thêm từ khóa ***typedef*** vào trước một khai báo nào đó.

Ví dụ Định nghĩa kiểu bằng ***typedef.***

Để khai báo một biến nguyên có tên là *nguyen* ta có thể viết như sau:

int nguyen;

Nhưng nếu ta thêm từ khóa ***typedef*** vào trước của khai báo đó:

typedef int nguyen;

Thì lúc này *nguyen* đã trở thành một kiểu dữ liệu mới và câu lệnh sau đây là hoàn toàn đúng: nguyen i,j; tương tự ta cũng có thể định nghĩa ra một kiểu dữ liệu mới có tên là ***Mangnguyen50*** dùng để khai báo các biến mảng nguyên có kích thước là 50 như sau:

typedef int MangNguyen50[50];

Sau câu lệnh này MangNguyen50 sẽ trở thành một kiểu dữ liệu mới và ta có thể dùng nó để khai báo cho các biến tương tự như việc khai báo các biến có kiểu định nghĩa sẵn. Câu lệnh sau sẽ tạo ra hai biến kiểu

MangNguyen50 là m1,m2, mỗi biến là một mảng kiểu số nguyên có kích thước 50.

MangNguyen50 m1,m2;

4. Ghi chú

Trong khi lập trình cần phải ghi chú để giải thích các biến, hằng, thao tác xử lý giúp cho chương trình rõ ràng dễ hiểu, dễ nhớ, dễ sửa chữa và để người khác đọc vào dễ hiểu. Trong C có các ghi chú sau: // hoặc /* nội dung ghi chú */

Ví dụ 3 :

```
void main()
{
    int a, b;    //khai bao bien t kieu int
    a = 1;    //gan 1 cho a
    b =3;     //gan 3 cho b
 /*  thuat toan tim so lon nhat la
  neu a lon hon b thi a lon nhat
  nguoc lai b lon nhat */
    if (a > b) printf("max: %d", a);
    else printf("max: %d", b);
}
```

Khi biên dịch chương trình, C gặp cặp dấu ghi chú sẽ không dịch ra ngôn ngữ máy.

Tóm lại, đối với ghi chú dạng // dùng để ghi chú một hàng và dạng /* …. */ có thể ghi chú một hàng hoặc nhiều hàng.

Ví dụ 4 :

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

```
int  main ()
{
        char ten[50]; /* khai bao bien ten kieu char 50 ky tu */
                        /*Xuat chuoi ra man hinh*/
        printf("Xin cho biet ten cua ban !");
        scanf("%s",ten);   /*Doc vao 1 chuoi la ten cua ban*/
        printf("Xin chao ban %s\n ",ten);
        printf("Chao mung ban den voi Ngon ngu lap trinh C");
        /*Dung chuong trinh, cho go phim*/
        getch();
        return 0;
}
```

5. Khai báo biến

Biến là một vùng nhớ có kích thước và có một địa chỉ nhất định nằm trong bộ nhớ RAM. Biến dùng để lưu giữ một dữ liệu đầu vào, đầu ra hoặc một kết quả trung gian trong quá trình làm việc. Dữ liệu được lưu trong biến nên cách tổ chức thông tin trong biến là kiểu của dữ liệu.

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.

Để phân biệt các biến với nhau, mỗi biến sẽ được đặt một tên theo quy tắc đặt định danh và được gọi là định danh biến (Variable Indentifier). 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.

a. Cú pháp khai báo biến

<Kiểu dữ liệu> Danh sách các tên biến cách nhau bởi dấu phẩy;

Kiểu dữ liệu: một trong các kiểu ở mục 3

Danh sách tên các biến: gồm các tên biến có cùng kiểu dữ liệu, mỗi tên biến cùng kiểu dữ liệu cách nhau bởi dấu phấy (,), cuối cùng là dấu chấm phẩy (;). Các biến khác kiểu nhau được khai báo cách nhau bằng dấu chấm phẩy (;).

Ví dụ:

```
int   ia, ib, ic;        /*Ba biến a, b,c có kiểu int*/
int ituoi;     //khai báo biến ituoi có kiểu int
float  fTrongluong;  //khai báo biến fTrongluong có kiểu long
char ckitu1, ckitu2; //khai báo biến ckitu1, ckitu2 có kiểu char
```

Các biến khai báo trên theo quy tắc Hungarian Notation. Nghĩa là biến ituoi là kiểu int, bạn thêm chữ **i** (kí tự đầu của kiểu) vào đầu tên biến **tuoi** để trong quá trình lập trình hoặc sau này xem lại, sửa chữa… bạn dễ dàng nhận ra biến **ituoi** có kiểu **int** mà không cần phải di chuyển đến phần khai báo mới biết kiểu của biến này. Tương tự cho biến **fTrongluong**, bạn nhìn vào là biết ngay biến này có kiểu **float.**

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

Để xác định độ lớn của một biến (số byte mà biến chiếm giữ trong bộ nhớ) chúng ta sử dụng toán tử sau:

int sizeof(<Biến>)

b. Vừa khai báo vừa khởi gán

Có thể kết hợp việc khai báo với toán tử gán để biến nhận ngay giá trị cùng lúc với khai báo.

Ví dụ 5 :

Khai báo trước, gán giá trị sau:

```
void main()
{
        int a, b, c;
        a = 1;
        b = 2;
        c = 5;
          …
}
```

Vừa khai báo vừa gán giá trị:

```
void main()
{
        int a = 1, b = 2, c = 5;
          …
}
```

c. Phạm vi của biến

Trong ngôn ngữ lập trình C, ta phải nắm rõ phạm vi của biến. Nếu khai báo và sử dụng không đúng, không rõ ràng sẽ dẫn đến sai sót khó kiểm soát được, vì vậy bạn cần phải xác định đúng vị trí, phạm vi sử dụng biến trước khi sử dụng biến. Chúng ta có 2 cách đặt vị trí của biến như sau:

Khai báo biến ngoài (biến toàn cục): Vị trí biến đặt bên ngoài tất cả các hàm, cấu trúc... Các biến này có ảnh hưởng đến toàn bộ chương trình. Chu trình sống (thời gian được cấp phát bộ nhớ) của nó là bắt đầu chạy chương trình đến lúc kết thúc chương trình. Nghĩa là giá trị của biến sẽ được lưu trữ trong suốt thời gian mà chương trình hoạt động. Một biến ngoài có

thể được khởi gán một lần lúc dịch chương trình, nếu không được khởi gán máy sẽ mặc định gán cho giá trị 0 hoặc NULL (nếu là con trỏ).

Ví dụ:

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

Khai báo biến trong hay biến cục bộ: 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, … Chu trình sống (thời gian được cấp phát bộ nhớ) của nó là bắt đầu từ khi máy làm việc với khối lệnh đến khi ra khỏi khối lệnh đó. Nghĩa là khi ra khỏi khối lệnh, vùng nhớ được cấp phát cho biến sẽ bị xóa, do đó các giá trị của biến cũng mất đi. Một biến trong (*không áp dụng cho mảng*) nếu không được khởi gán một giá trị nào đó thì biến đó hoàn toàn không xác định (*nhận một giá trị ngẫu nhiên sẵn có trong bộ nhớ lúc được cấp phát)*

Ví dụ 6:

```
#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();
         i=1; j=2;
```

```
        bienngoai=3;
        printf("\n Gia7 tri cua i la %d",i);
    /*%d là số nguyên, sẽ biết sau */
        printf("\n Gia tri cua j la %d",j);
        printf("\n Gia tri cua bienngoai la %d",bienngoai);
        getch();
        return 0;
}
```

Ví dụ 7:

```
#include <stdio.h>
#include<conio.h>
int  main ()
{       int i, j;          /*Bien ben trong*/
        clrscr();
        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;       /*Bien ben trong */
                printf("\n Hieu so cua j tru i la %d",hieu);
         }
        else
        {
                int hieu=i-j      ; /*Bien ben trong*/
                printf("\n Gia tri cua i tru j la %d",hieu);
        }
         getch();
```

return 0;

}

6. Nhập/ xuất dữ liệu

a. Xuất dữ liệu ra màn hình (hàm printf)

Kết xuất dữ liệu được định dạng.

Cú pháp:

printf ("chuỗi định dạng"[, đối mục 1, đối mục 2,…]);

Khi sử dụng hàm phải khai báo tiền xử lý #include <stdio.h>

printf: tên hàm, phải viết bằng chữ thường.

đối mục 1,…: là các mục dữ kiện cần in ra màn hình. Các đối mục này có thể là biến, hằng hoặc biểu thức phải được định trị trước khi in ra.

chuỗi định dạng: được đặt trong cặp nháy kép (" "), là cách trình bày thông tin sẽ được xuất. Một chuỗi định dạng có ba thành phần:

+ Văn bản thường (literal text) trong chuỗi định dạng sẽ được xuất y hệt lúc gõ. Ví dụ sau sẽ xuất chuỗi Tin hoc co so A và chuỗi Chương 7 ra màn hình. Chuỗi sau xuất ngay sau chuỗi trước.

printf(“Tin hoc co so A”);

printf(“Chương 7”);

+ Đặc tả (conversion specifier) gồm dấu phần % và một ký tự. Phần này dùng để xác định kiểu của biến muốn xuất.Đối với những kí tự chuyển đổi dạng thức cho phép kết xuất giá trị của các đối mục ra màn hình tạm gọi là mã định dạng. Sau đây là các dấu mô tả định dạng:

%c : Ký tự đơn

%s : Chuỗi

%d : Số nguyên thập phân có dấu

%f : Số chấm động (ký hiệu thập phân)

%e : Số chấm động (ký hiệu có số mũ)

%g : Số chấm động (%f hay %g)

%x : Số nguyên thập phân không dấu

%u : Số nguyên hex không dấu

%o : Số nguyên bát phân không dấu

l : Tiền tố dùng kèm với %d, %u, %x, %o để chỉ số nguyên dài (ví dụ %ld)

+ Các ký tự điều khiển và ký tự đặc biệt:

\n : Nhảy xuống dòng kế tiếp canh về cột đầu tiên.

\t : Canh cột tab ngang.

\r : Nhảy về đầu hàng, không xuống hàng.

\a : Tiếng kêu bip.

\\ : In ra dấu \

\" : In ra dấu "

\' : In ra dấu '

%%: In ra dấu %

Ví dụ 8:

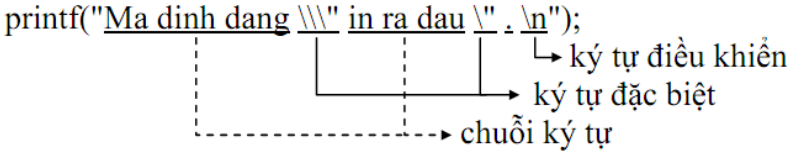
```
printf("Bai hoc C dau tien. \n");
                              ↳ ký tự điều khiển
       ↳ chuỗi ký tự
```

Kết quả in ra màn hình:

Bai học C dau tien.

_

Ví dụ 9:

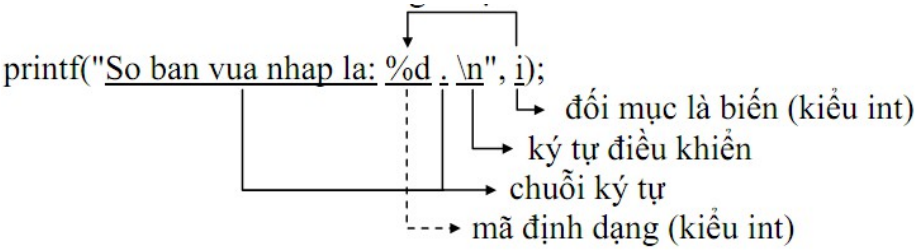


Kết quả in ra màn hình:

Ma dinh dang \"in ra dau".

–

Ví dụ 10: giả sử biến i có giá trị = 5 xuất giá trị biến i

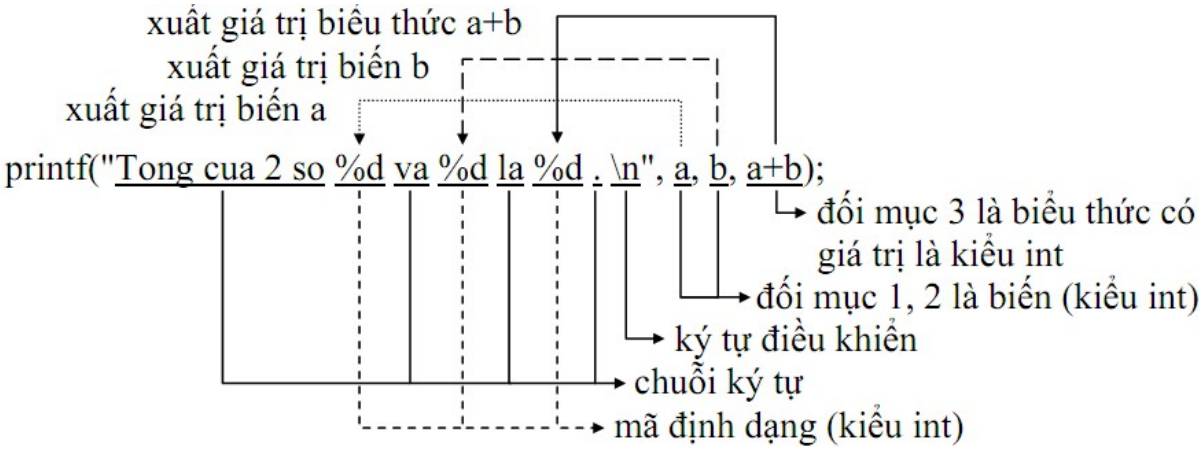


Kết quả in ra màn hình:

So ban vua nhap la: 5.

–

Ví dụ 11: giả sử biến a có giá trị = 7 và b có giá trị = 4

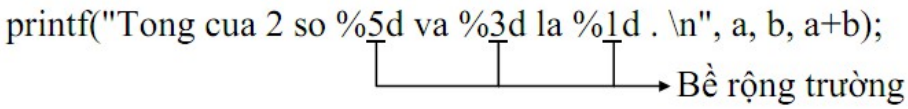


Kết quả in ra màn hình:

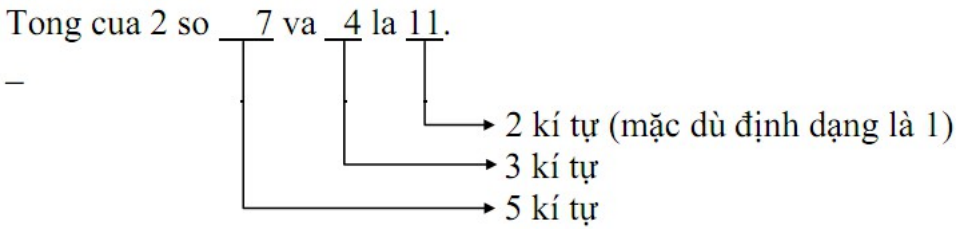
Tong cua 2 so 7 va 4 la 11.

–

Ví dụ 12: sửa lại ví dụ 11



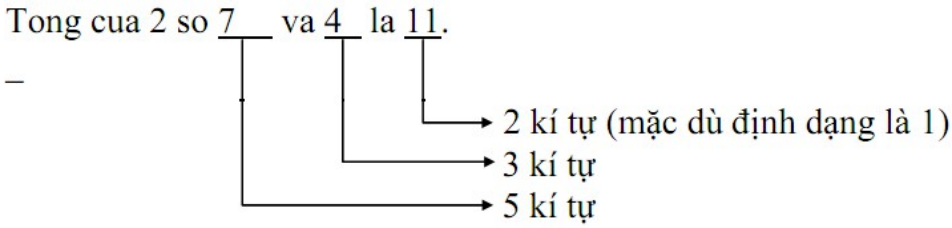
Kết quả in ra màn hình:



Ví dụ 13: sửa lại ví dụ 12

printf("Tong cua 2 so %-5d va %-3d la %-1d . \n", a, b, a+b);

Kết quả in ra màn hình:

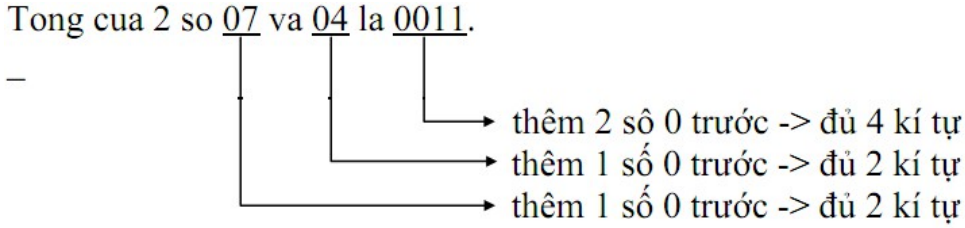


➔Dấu trừ trước bề rộng trường sẽ kéo kết quả sang trái

Ví dụ 14: sửa lại ví dụ 11

printf("Tong cua 2 so %02d va %02d la %04d . \n", a, b, a+b);

Kết quả in ra màn hình:

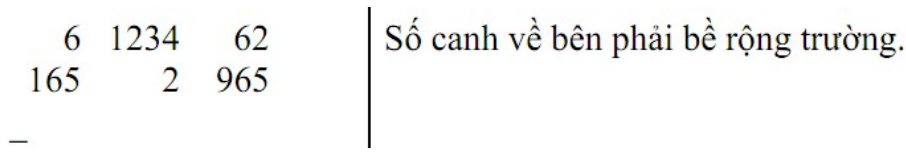


Ví dụ 15: giả sử int a = 6, b = 1234, c = 62

printf("%7d%7d%7d.\n", a, b, c);

printf("%7d%7d%7d.\n", 165, 2, 965);

Kết quả in ra màn hình:



printf("%-7d%-7d%-7d.\n", a, b, c);

printf("%-7d%-7d%-7d.\n", 165, 2, 965);

Kết quả in ra màn hình:

```
6      1234   62
165    2      965
_
```

Số canh về bên trái bề rộng trường.

Ví dụ 16: giả sử float a = 6.4, b = 1234.56, c = 62.3

```
printf("%7.2d%7.2d%7.2d.\n", a, b, c);
                  số số lẻ
```

Kết quả in ra màn hình:

```
  6.40  1234.56  62.30
_
          7 kí tự
```

Số canh về bên phải bề rộng trường.

➔ Bề rộng trường bao gồm: phần nguyên, phần lẻ và dấu chấm động

b. Hàm scanf

Định dạng khi nhập liệu.

Cú pháp:

scanf ("chuỗi định dạng"[, đối mục 1, đối mục 2,…]);

Khi sử dụng hàm phải khai báo tiền xử lý #include <stdio.h>

scanf: tên hàm, phải viết bằng chữ thường.

khung định dạng: được đặt trong cặp nháy kép (" ") là hình ảnh dạng dữ liệu nhập vào.

đối mục 1,…: là danh sách các đối mục cách nhau bởi dấu phẩy, mỗi đối mục sẽ tiếp nhận giá trị nhập vào.

Ví dụ 17:

```
scanf("%d", &i);
        đối mục 1
      mã định dạng
```

Nhập vào 12abc, biến i chỉ nhận giá trị 12. Nhập 3.4 chỉ nhận giá trị 3.

Ví dụ 18: **scanf("%d%d", &a, &b);**

Nhập vào 2 số a, b phải cách nhau bằng khoảng trắng hoặc enter.

Ví dụ 19:

scanf("%d/%d/%d", &ngay, &thang, &nam);

Nhập vào ngày, tháng, năm theo dạng ngay/thang/nam (20/12/2002)

Ví dụ 20:

scanf("%d%*c%d%*c%d", &ngay, &thang, &nam);

Nhập vào ngày, tháng, năm với dấu phân cách /, -,…; ngoại trừ số.

Ví dụ 21:

scanf("%2d%2d%4d", &ngay, &thang, &nam);

Nhập vào ngày, tháng, năm theo dạng dd/mm/yyyy.

Chương III. Cấu trúc rẽ nhánh có điều kiện

1. Lệnh và khối lệnh

1.1. Lệnh

Là một tác vụ, biểu thức, hàm, cấu trúc điều khiển…

Ví dụ 1:

```
x = x + 2;
printf("Day la mot lenh\n");
```

1.2. Khối lệnh

Là một dãy các câu lệnh được bọc bởi cặp dấu { }, các lệnh trong khối lệnh phải viết thụt vào 1 tab so với cặp dấu { }

Ví dụ 2:

```
{ //dau khoi
    a = 5;
    b = 6;
    printf("Tong %d + %d = %d", a, b, a+b);
} //cuoi khoi
```

2. Lệnh if

Câu lệnh if cho phép lựa chọn một trong hai nhánh tùy thuộc vào giá trị của biểu thức luận lý là đúng (true) hay sai (false) hoặc khác không hay bằng không.

2.1. Dạng 1 (if thiếu)

Quyết định sẽ thực hiện hay không một khối lệnh.

Cú pháp lệnh:

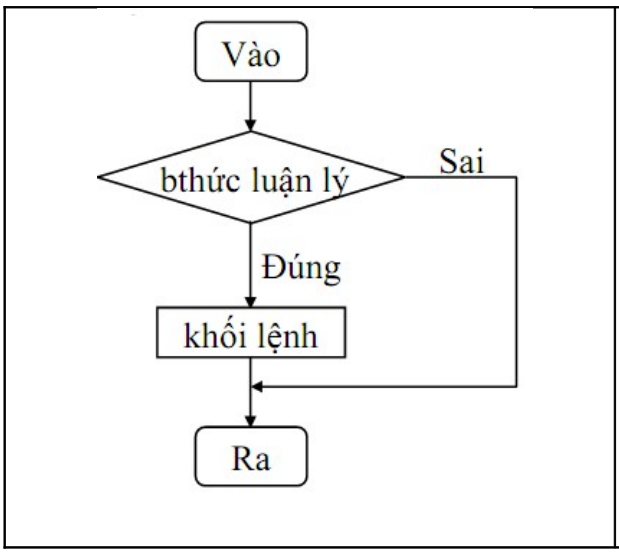
if (biểu thức luận lý)

khối lệnh;

Từ khoá if phải viết bằng chữ thường

Kết quả của biểu thức luận lý phải là đúng (≠ 0) hoặc sai (= 0)

Lưu đồ:



Nếu biểu thức luận lý đúng thì thực hiện khối lệnh và thoát khỏi if,

ngược lại

Không làm gì cả và thoát khỏi if.

Hình 2.1. Sơ đồ hoạt động của lệnh if thiếu

Nếu khối lệnh bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

Diễn giải:

+ Biểu thức luận lý có thể là biểu thức bất kì (biểu thức nguyênm thực, quan hệ và logic,…) miễn sao cứ trả về giá trị luận lý là đúng (≠ 0) hoặc sai (=0).

+ Khối lệnh là một lệnh ta viết lệnh if như sau:

```
if (biểu thức luận lý)
 lệnh;
```

+ Khối lệnh bao gồm nhiều lệnh: lệnh 1, lệnh 2..., ta viết lệnh if như sau:

```
if (biểu thức luận lý)
{
        lệnh 1;
        lệnh 2;
```

...

}

Chu´y´:

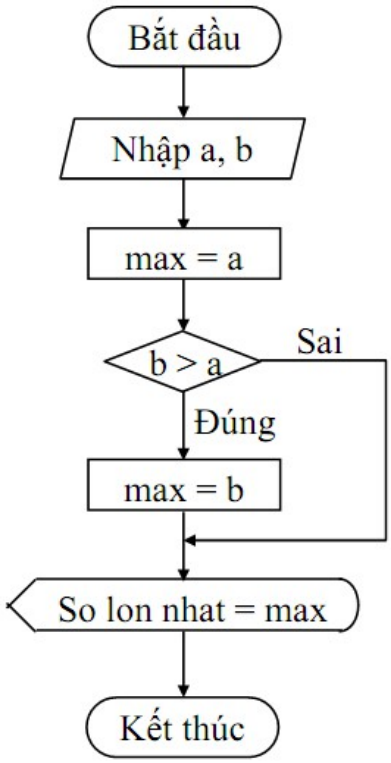
Không đặt dấu chấm phẩy sau câu lệnh if.

Ví dụ: if(biểu thức luận lý);

→ trình biên dịch không báo lỗi nhưng khối lệnh không được thực hiện cho dù điều kiện đúng hay sai

Ví dụ 3: Viết chương trình nhập vào 2 số nguyên a, b. Tìm và in ra số lớn nhất.

Mô tả bằng lưu đồ:



Viết chương trình:

```
/* Chuong trinh tim so lon nhat tu 2 so nguyen a, b */
 #include <stdio.h>
#include <conio.h>
 void main(void)
{
```

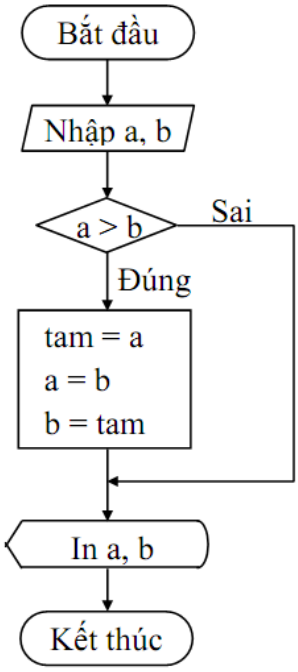
```
        int ia, ib, imax;
        printf("Nhap vao so a: ");
        scanf("%d", &ia);
         printf("Nhap vao so b: ");
        scanf("%d", &ib);
        imax = ia;
        if (ib>ia)
                imax = ib;
        printf("So lon nhat = %d.\n", imax);
        getch();
}
```

Kết quả in ra màn hình

```
        Nhap vao so a : 10
        Nhap vao so b : 8
        So lon nhat = 10.
        _
```

Ví dụ 4: Viết chương trình nhập vào 2 số nguyên a, b. Nếu a lớn hơn b thì hoán đổi giá trị a và b, ngược lại không hoán đổi. In ra giá trị a, b.

Mô tả bằng lưu đồ



Viết chương trình

```
/* Chuong trinh hoan vi 2 so a, b neu a > b */
 #include <stdio.h>
#include <conio.h>
 void main(void)
{
        int ia, ib, itam;
        printf("Nhap vao so a: ");
        scanf("%d", &ia);
         printf("Nhap vao so b: ");
         scanf("%d", &ib);
        if (ia>ib)
         {
                itam = ia;  //hoan vi a va b
                ia = ib;
                 ib = itam;
        }
```

```
        printf("%d, %d.\n", ia, ib);
        getch();
}
```

Kết quả in ra màn hình

```
Nhap vao so a : 10
Nhap vao so b : 8
8, 10
```

2.2. Dạng 2 (if đủ)

Quyết định sẽ thực hiện 1 trong 2 khối lệnh cho trước

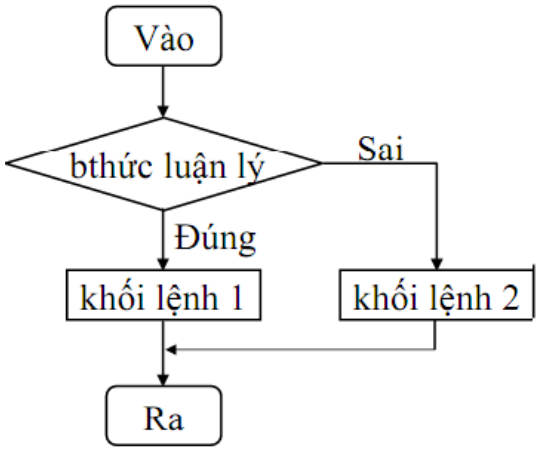
Cú pháp lệnh

```
if (biểu thức luận lý)  )
        khối lệnh 1;
 else
        khối lệnh 2;
```

Từ khóa if, else phải viết bằng chữ thường

Kết quả của biểu thức luận lý phải là đúng (≠ 0) hoặc sai (= 0)

Lưu đồ`



Hình 2.2. Sơ đồ hoạt động của lệnh if đủ

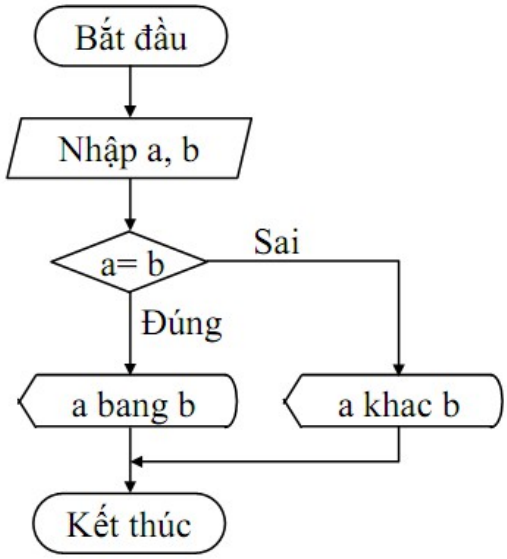
Nếu biểu thức luận lý đúng thì thực hiện khối lệnh 1 và thoát khỏi if

Ngược lại thực hiện khối lệnh 2 và thoát khỏi if.

Nếu khối lệnh 1, khối lệnh 2 bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

Ví dụ 4: Viết chương trình nhập vào 2 số nguyên a, b. Nếu a lớn hơn b thì hoán đổi giá trị a và b, ngược lại không hoán đổi. In ra giá trị a, b.

Mô tả bằng lưu đồ



Viết chương trình

```
/* Chuong trinh in ra thong bao "a bang b" neu a = b, nguoc lại in ra "a khac b" */
 #include <stdio.h>
#include <conio.h>
 void main(void)
{
        int ia, ib;
        printf("Nhap vao so a: ");
       scanf("%d", &ia);
```

```
        printf("Nhap vao so b: ");
        scanf("%d", &ib);
        if (ia == ib)
            printf("a bang b\n");
        else
            printf("a khac b\n");
        getch();
}
```

Kết quả in ra màn hình

```
Nhap vao so a : 10
Nhap vao so b : 8
a khac b.
```

Lưu ý:

- Sau else không có dấu chấm phẩy.

  Ví dụ: else; printf('a khac b\n");

  → trình biên dịch không báo lỗi, lệnh printf("a khac b\n"); không thuộc else

- Các toán tử if có thể lồng nhau, tuy nhiên khi số từ khóa ***else*** ít hơn số từ khóa if thì từ khóa else sẽ được gắn với if gần nhất trước đó.

  Ví dụ Với đoạn chương trình sau:

```
if (a>0)
    if(b>0)
        c=2009
    else
        c=1999
```

  Thì ***else*** sẽ được gắn với từ khóa if bên dưới. Để tránh nhầm lẫn trong khi sử dụng các toán tử if lồng nhau ta nên sử dụng khối lệnh

để xác định phạm vi cho toán tử if trong chương trình. Ví dụ trên có thể viết lại như sau:

```
if (a>0)
{
        if(b>0)
                c=2009
        else
                c=1999
}
```

2.3. Câú truć else if

Quyết định sẽ thực hiện 1 trong n khối lệnh cho trước.

Cu´phaṕ lệnh

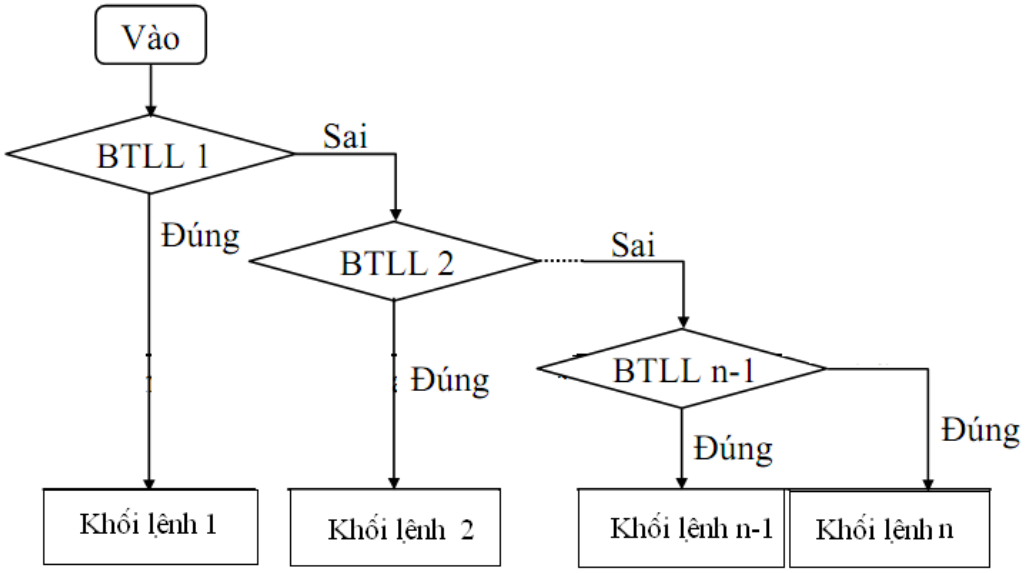
```
if (biểu thức luận lý 1)
   khối lệnh 1;
else if (biểu thức luận lý 2)
   khối lệnh 2;
…
else if (biểu thức luận lý n-1)
  khối lệnh n-1;
else
   khối lệnh n;
```

Từ khóa if, else if, else phải viết bằng chữ thường

Kết quả của biểu thức luận lý 1, 2..n phải là đúng (≠ 0) hoặc sai (= 0)

Nếu khối lệnh 1, 2…n bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

Lưu đồ`



Hình 2.3. Sơ đồ hoạt động của cấu trúc else if

Nếu **biểu thức luận lý 1** đúng thì thực hiện khối lệnh 1 và thoát khỏi cấu trúc if

Ngược lại Nếu **biểu thức luận lý 2** đúng thì thực hiện khối lệnh 2 và thoát khỏi cấu trúc if

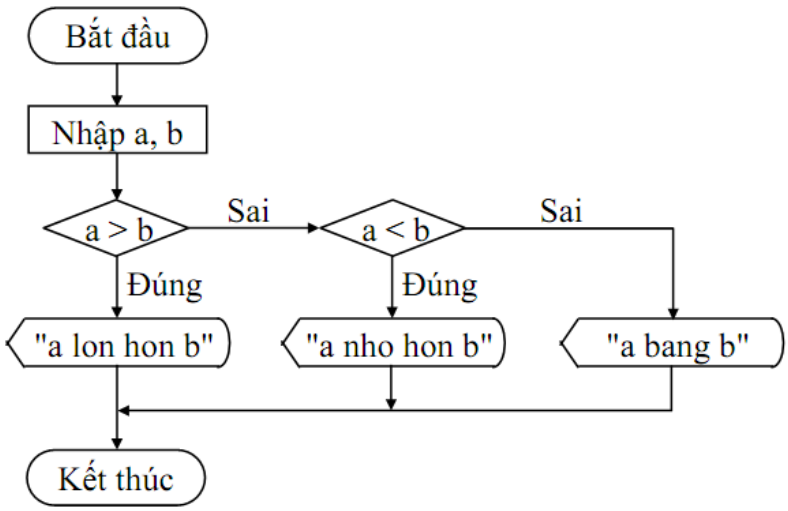
…

Ngược lại Nếu **biểu thức luận lý n-1** đúng thì thực hiện khối lệnh n-1 và thoát khỏi cấu trúc if

Ngược lại thì thực hiện khối lệnh n.

Vi´dụ 6: Viết chương trình nhập vào 2 số nguyên a, b. In ra thông báo "a lớn hơn b" nếu a>b, in ra thông báo "a nhỏ hơn b" nếu a<b, in ra thông báo "a bằng b" nếu a=b.

Mô tả băng lưu đồ`



Viết chương trình

```
/* Chuong trinh nhap vao 2 so nguyen a, b. In ra thong bao a > b, a < b, a = b */
 #include <stdio.h>
#include <conio.h>
 void main(void)
{
        int ia, ib;
        printf("Nhap vao so a: ");
        scanf("%d", &ia);
        printf("Nhap vao so b: ");
        scanf("%d", &ib);
        if (ia>ib)
                printf("a lon hon b.\n");
        else if (ia<ib)
                printf("a nho hon b.\n");
        else
                printf("a bang b.\n");
        getch();
}
```

Kết quả in ra màn hình

Nhap vao so a : 5

Nhap vao so b : 7

a nho hon b

Lưu ý:

Cũng như if, không đặt dấu chấm phẩy sau câu lệnh else if.

Ví dụ: else if(c >= 'A' && c <= 'Z');

→ trình biên dịch không báo lỗi nhưng khối lệnh sau else if không được thực hiện.

2.4. Câú truć if lôǹg nhau

Quyết định sẽ thực hiện 1 trong n khối lệnh cho trước.

Cu ́phaṕ lệnh

Cú pháp là một trong 3 dạng trên, nhưng trong 1 hoặc nhiều khối lệnh bên trong phải chứa ít nhất một trong 3 dạng trên gọi là cấu trúc if lồng nhau. Thường cấu trúc if lồng nhau càng nhiều cấp độ phức tạp càng cao, chương trình chạy càng chậm và trong lúc lập trình dễ bị nhầm lẫn.

Lưu ý: Các lệnh **if…else lồng nhau** thì else sẽ luôn luôn kết hợp với if nào chưa có else gần nhất. Vì vậy khi gặp những lệnh if không có else, ta phải đặt chúng trong những khối lệnh rõ ràng để tránh bị hiểu sai câu lệnh.

Ví dụ 7: Ta viết các dòng lệnh sau:

```
…
 if (n > 0)
        if (a > b)
                x = a;
 else
```

```
        x = b;
    …
```

Mặc dù ta viết lệnh else thẳng hàng với if (n > 0), nhưng lệnh else ở đây được hiểu đi kèm với if (a > b), vì nó nằm gần với if (a > b) nhất và if (a > b) chưa có else. Để dễ nhìn và dễ hiểu hơn ta viết lại như sau:

```
  …
    if (n > 0)
        if (a > b)
            x = a;
        else
            x = b;
        …
```

Còn nếu ta muốn lệnh else là của if (n > 0) thì phải đặt if (a > b) x = a trong một khối lệnh. Bạn viết lại như sau:

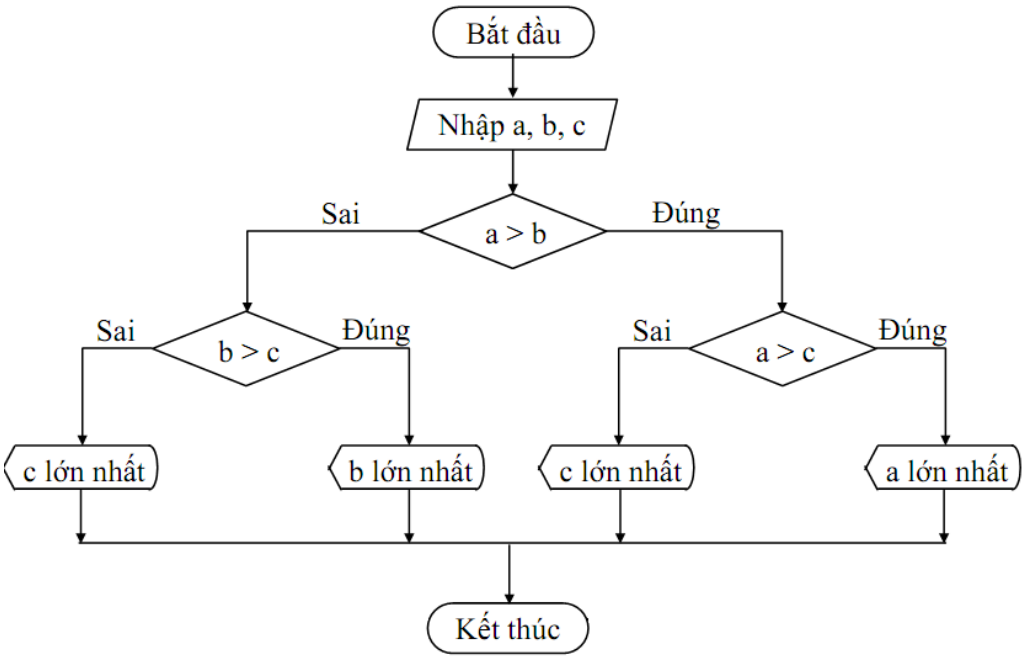
```
  …
 if (n > 0)
  {
        if (a > b)
            x = a;
  }
  else
        x = b;
  …
```

Lưu đô`

Tương tự 3 dạng trên. Nhưng trong mỗi khối lệnh có thể có một (nhiều) cấu trúc if ở 3 dạng trên.

Ví dụ 8: Viết chương trình nhập vào 3 số nguyên a, b, c. Tìm và in ra số lớn nhất.

Mô tả bằng lưu đồ



Viết chương trình

```
/* Chuong trinh nhap vao 2 so nguyen a, b, c. Tim, in ra so lon nhat */
#include <stdio.h>
#include <conio.h>
void main(void)
{
    int ia, ib, ic;
    printf("Nhap vao so a: ");
    scanf("%d", &ia);
    printf("Nhap vao so b: ");
    scanf("%d", &ib);
    printf("Nhap vao so c: ");
    scanf("%d", &ic);
    if (ia > ib)
```

```
            if (ia > ic)
                printf("%d lon nhat.\n", ia);
            else
                printf("%d lon nhat.\n", ic);
    else
            if (ib > ic)
                printf("%d lon nhat.\n", ib);
            else
                printf("%d lon nhat.\n", ic);
            getch();
}
```

Kết quả in ra màn hình

```
Nhap vao so a: 4
Nhap vao so b: 5
Nhap vao so c: 3
5 lon nhat.
_
```

3. Lệnh switch

Lệnh switch cũng giống cấu trúc ***else if***, nhưng nó mềm dẻo hơn và linh động hơn nhiều so với sử dụng ***if.*** Tuy nhiên, nó cũng có mặt hạn chế là kết quả của biểu thức phải là giá trị hằng nguyên (có giá trị cụ thể). Một bài toán sử dụng lệnh switch thì cũng có thể sử dụng ***if***, nhưng ngược lại còn tùy thuộc vào giải thuật của bài toán.

3.1. Câú truć switch….case (switch thiếú)

Chọn thực hiện 1 trong n lệnh cho trước

Cu´phaṕ lệnh

switch (biểu thức)

{

case giá trị n1 : lệnh 1;

break;

case giá trị n2 : lệnh 2;

break;

…

case giá trị nk : lệnh n;

[break;]

}

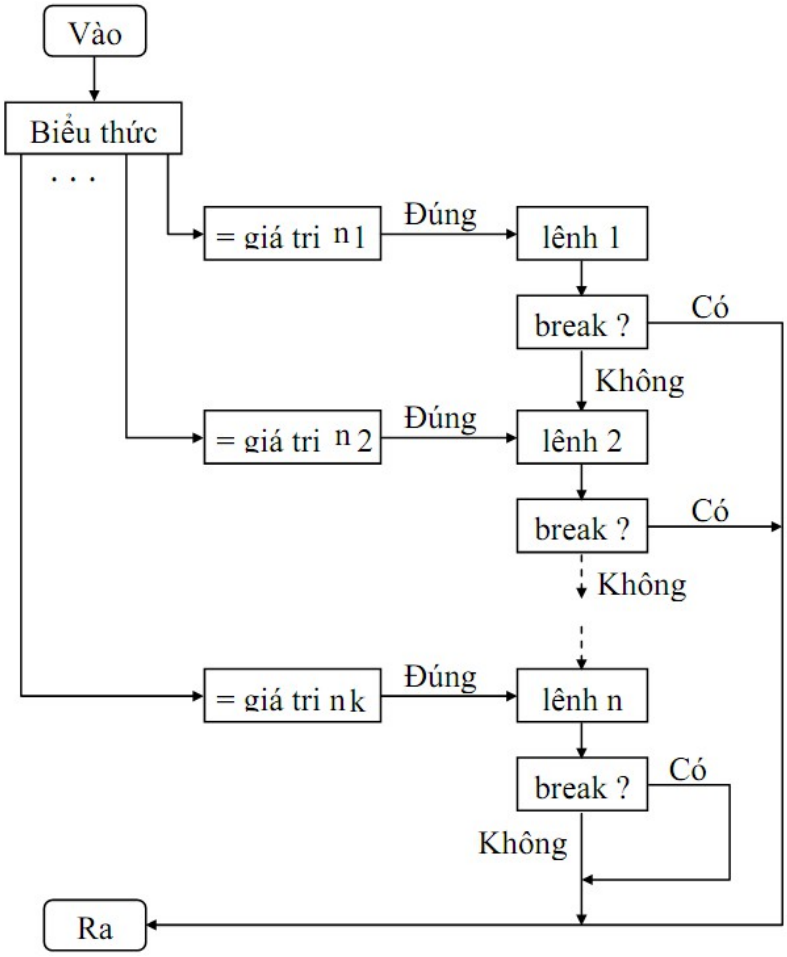
Từ khóa switch, case, break phải viết bằng chữ thường

Biểu thức phải là có kết quả là giá trị hằng nguyên (char, int, long,…)

Giá trị n1, giá trị n2,… (hay ni) là các số nguyên, hằng kí tự hoặc biểu thức hằng. Các giá trị này phải khác nhau.

Lệnh 1, 2…n có thể gồm nhiều lệnh, nhưng không cần đặt trong cặp dấu { }

Lưu đồ`



Hình 3.1. Sơ đồ hoạt động của cấu trúc switch…case(switch thiếu)

Trước tiên máy sẽ xét giá trị của biểu thức , tùy theo giá trị của biểu thức này mà nó quyết định nhảy tới đâu.

Nếu giá trị này bằng ni máy sẽ nhảy tới câu lệnh nằm sau nhãn *case ni* và bắt đầu thực hiện các lệnh từ đó cho đến khi gặp một toán tử ***break, goto, return***, hoặc dấu kết thúc '}'.

Chú ý:

- Máy sẽ thoát khỏi toán tử switch chỉ khi nó gặp một toán tử ***break*** hoặc dấu đóng ngoặc '}'chỉ sự kết thúc của toán tử ***switch***. Do đó các toán tử ***break*** là không thể vắng mặt mỗi khi một nhánh nào đó đã được lựa chọn.

- Trong thân của lệnh ***switch*** ta cũng có thể sử dụng toán tử goto để nhảy tới một câu lệnh bất kỳ bên ngoài ***switch***.

- Khi ***switch*** nằm trong thân của một hàm nào đó, ta cũng có thể sử dụng một toán tử ***return*** trong thân của switch để thoát khỏi hàm đó.

Ví dụ 9: Viết chương trình nhập vào số 1, 2, 3. In ra tương ứng 1, 2, 3 sao.

Viết chương trình

```
/* Chuong trinh nhap vao so 1, 2, 3. In ra so sao tuong ung */
 #include <stdio.h>
#include <conio.h>
 void main(void)
{
        int i;
       printf("Nhap vao so 1, 2 hoặc 3: ");
       scanf("%d", &i);
        switch(i)
       {
              case 3: printf("*");
              case 2: printf("*");
              case 1: printf("*");
        };
        printf("An phim bat ky de ket thuc!\n");
       getch();
}
```

Kết quả in ra màn hình

```
Nhap vao so 1, 2 hoặc 3: 2
**
```

Lưu ý:

Không đặt dấu chấm phẩy sau câu lệnh switch.

Ví dụ: switch(i); → trình biên dịch không báo lỗi nhưng các lệnh trong switch không được thực hiện.

3.2. Câú truć switch….case (switch đủ)

Chọn thực hiện 1 trong n + 1 lệnh cho trước.

Cúpháp lệnh

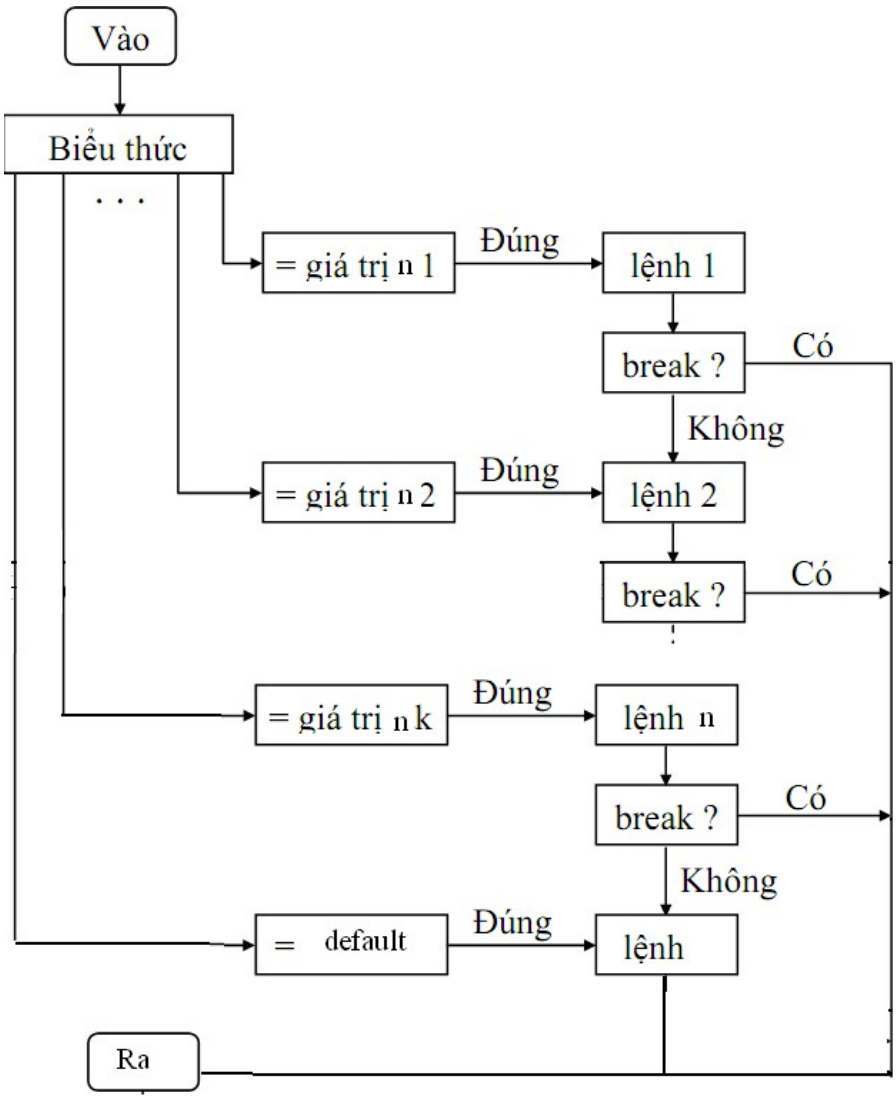
```
switch (biểu thức)
{
   case giá trị 1 :    lệnh 1;
                       break;
   case giá trị 2 :    lệnh 2;
                       break;
   …
   case giá trị n :    lệnh n;
                       break;
   default :           lệnh;
                       [break;]
}
```

Từ khóa **switch, case, break, default** phải viết bằng chữ thường

Biểu thức phải là có kết quả là giá trị nguyên (char, int, long, …)

Lệnh 1, 2…n có thể gồm nhiều lệnh, nhưng không cần đặt trong cặp dấu { }

Lưu đô`



Hình 3.2. Sơ đồ hoạt động của cấu trúc switch…case(switch đủ)

Tương tự như hoạt động của cấu trúc switch…case (switch thiếu) như trên. Ở cấu trúc này có thêm câu lệnh nằm sau từ khóa ***default.***

Khi giá trị của biêu thức khác tất cả các giá trị ni, i=1…k thì sự hoạt động của switch lúc này phụ thuộc vào sự có mặt hay vắng mặt của ***default.*** Nếu có mặt ***default*** thì các câu lệnh nằm sau ***default*** sẽ được thực hiện, ngược lại máy sẽ thoát khỏi ***switch*** và bắt đầu thực hiện các lệnh nằm sau dấu '}' của thân ***switch.***

Ví dụ 10: Viết chương trình nhập vào số 1, 2, 3. In ra tương ứng 1, 2, 3 sao.

Viết chương trình

```
/* Chuong trinh nhap vao so 1, 2, 3. In ra so sao tuong ung */
 #include <stdio.h>
#include <conio.h>
 void main(void)
        {
                int i;
                printf("Nhap vao so 1, 2 hoặc 3: ");
                scanf("%d", &i);
                 switch(i)
                      {
                        case 3: printf("*");
                        case 2: printf("*");
                        case 1: printf("*");
                                 break;
                        default: printf("Ban nhap phai nhap vao so 1, 2 hoac
                        3.\n");
                       };
                getch();
        }
```

Kết quả in ra màn hình

```
                Nhap vao so 1, 2 hoặc 3: 3
                ***
                _
```

3.3. Câú truć switch lôǹg nhau

Quyết định sẽ thực hiện 1 trong n khối lệnh cho trước.

Cu´phaṕ lệnh

Cú pháp là một trong 2 dạng trên nhưng trong 1 hoặc nhiều lệnh bên trong phải chứa ít nhất một trong 2 dạng trên gọi là cấu trúc **switch lồng nhau**. Thường cấu trúc switch lồng nhau càng nhiều cấp độ phức tạp càng cao, chương trình chạy càng chậm và trong lúc lập trình dễ bị nhầm lẫn.

Lưu đô`

Tương tự 2 dạng trên. Nhưng trong mỗi lệnh có thể có một (nhiều) cấu trúc switch ở 2 dạng trên.

Vi´dụ 11: Viết chương trình menu 2 cấp

Viêt chương trình

```
/* Chuong trinh menu 2 cap */
#include <stdio.h>
#include <conio.h>
 void main(void)
        {
                int imenu, isubmenu;
               printf("------------------------\n");
               printf("     MAIN MENU     \n");
                printf("------------------------\n");
               printf("1. File\n");
                printf("2. Edit\n");
               printf("3. Search\n");
               printf("Chon muc tuong ung: ");
               scanf("%d", &imenu);
```

```
            switch(imenu)
             {
                  case 1: printf("------------------------\n");
                          printf("      MENU FILE      \n");
                          printf("------------------------\n");
                          printf("1. New\n");
                          printf("2. Open\n");
                          printf("Chon muc tuong ung: ");
                          scanf("%d", &isubmenu);
                          switch(isubmenu)
                           {
                              case 1: printf("Ban da chon chuc nang New File\n");
                                      break;
                             case 2: printf("Ban da chon chuc nang Open File\n");
                           }
                           break;  //break cua case 1 – switch(imenu)
                  case 2: printf("Ban da chon chuc nang Edit\n");
                          break;
                  case 3: printf("Ban da chon chuc nang Search\n");
            };
            getch();
        }
```

Kết quả in ra màn hình

```
------------------------
   MAIN MENU
------------------------
```

1. File

2. Edit

3. Search

Chon muc tuong ung: 1

MENU FILE

---

1. New

2. Open

Chon muc tuong ung: 2

Ban da chon chuc nang Open File

_

Bai tập hết chương

Sử dụng lệnh IF

Bai 1. Viết chương trình nhập vào số nguyên dương, in ra thông báo số chẵn hay lẻ.

Hướng dẫn: Nhập vào số nguyên dương x. Kiểm tra nếu x chia chẵn cho hai thì x là số

chẵn (hoặc chia cho 2 dư 0) ngược lại là số lẻ.

Bai 2. Viết chương trình nhập vào 4 số nguyên. Tìm và in ra số lớn nhất.

Hướng dẫn: Ta có 4 số nguyên a, b, c, d. Tìm 2 số nguyên lớn nhất x, y của 2 cặp (a, b)

và (c, d). Sau đó so sánh 2 số nguyên x, y để tìm ra số nguyên lớn nhất.

Bai 3. Viết chương trình giải phương trình bậc 2: ax2 + bx + c = 0, với a, b, c nhập vào từ bàn phím.

Hướng dẫn: Nhập vào 3 biến a, b, c.

*Tính Delta = b*b - 4*a*c*

Nếu Delta < 0 thì

Phương trình vô nghiệm

Ngược lại

Nếu Delta = 0 thì

*x1 = x2 = - b/(2*a)*

Ngược lại

*x1 = (- b - sqrt(Delta))/(2*a)*

*x2 = (- b + sqrt(Delta))/(2*a)*

Hết Nếu

Hết Nếu

Bai 4. Viết chương trình nhập vào giờ phút giây (hh:mm:ss). Cộng thêm số giây nhập vào và in ra kết quả dưới dạng hh:mm:ss.

Hướng dẫn: Nhập vào giờ phút giây vào 3 biến gio, phut, giay và nhập và giây công thêm vào biến them:

Nếu giay + them < 60 thì

giay = giay + them

Ngược lại

giay = (giay + them) - 60

phut = phut + 1

Nếu phut >= 60 thì

phut = phut - 60

gio = gio + 1

Hết nếu

Hết nếu

Sử dụng lệnh switch

Bai 5. Viết chương trình nhập vào tháng, in ra tháng đó có bao nhiêu ngày.

Hướng dẫn: Nhập vào tháng

Nếu là tháng 1, 3, 5, 7, 8, 10, 12 thì có 30 ngày

Nếu là tháng 4, 6, 9, 11 thì có 31 ngày

Nếu là tháng 2 và là năm nhuận thì có 29 ngày ngược lại 28 ngày

(Năm nhuận là năm chia chẵn cho 4)

Bai 6. Viết chương trình trò chơi One-Two-Three ra cái gì ra cái này theo điều kiện:

- Búa (B) thắng Kéo, thua Giấy.
- Kéo (K) thắng Giấy, thua Búa.
- Giấy (G) thắng Búa, thua Kéo.

Hướng dẫn: Dùng lệnh switch lồng nhau

Bai 7. Viết chương trình xác định biến ký tự color rồi in ra thông báo

- RED, nếu color = 'R' hoặc color = 'r'
- GREEN, nếu color = 'G' hoặc color = 'g'

- BLUE, nếu color = 'B' hoặc color = 'b'
- BLACK, nếu color có giá trị khác.

Bai 8. Viết chương trình nhập vào 2 số x, y và 1 trong 4 toán tử +, -, *, /. Nếu là + thì in ra kết quả x + y, nếu là – thì in ra x – y, nếu là * thì in ra x * y, nếu là / thì in ra x / y (nếu y = 0 thì thông báo không chia được)

Chương IV. Câú truć voòng lặp

Trong lập trình có cấu trúc, thông thường ta cần tạo ra các đoạn mã mà hoạt động của nó lặp đi lặp lại theo một quy luật nào đó. Để thuận tiện cho việc tạo ra các chu trình trong khi viết chương trình, Turbo C đưa ra các loại cấu trúc lặp bao với mục đích khác nhau, đó là cấu trúc lặp xác định (dùng toán tử điều khiển for) và cấu trúc lặp không xác định (dùng toán tử điều khiển while và do...while). Cấu trúc lặp xác định hay vòng lặp xác định thường hay dùng cho các thao tác lặp đi lặp lại với số lần lặp đã được biết trước. Còn cấu trúc lặp không xác định hay vòng lặp không xác định thường hay dùng trong các trường hợp ta chưa biết trước sẽ phải lặp bao nhiêu lần, số lần lặp này thường phụ thuộc vào giá trị tại thời điểm hiện tại của một hay nhiều biểu thức nào đó.

1. Lệnh for

Vòng lặp xác định thực hiện lặp lại một số lần xác định của một (chuỗi hành động)

Cu´pháp lệnh

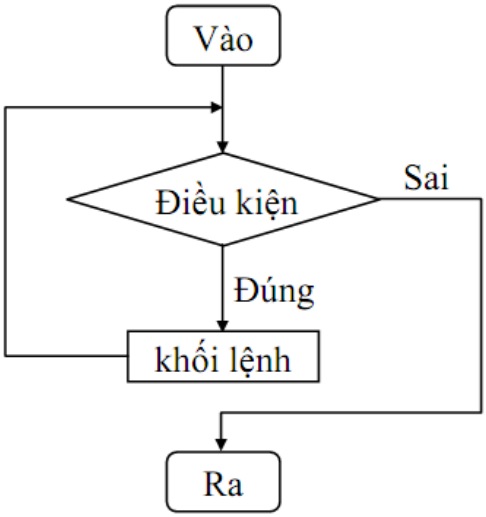
for (biểu thức 1; biểu thức 2; biểu thức 3)

khối lệnh;

Từ khóa **for** phải viết bằng chữ thường

Nếu khối lệnh bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

Lưu đồ`



Kiểm tra điều kiện **nếu đúng** đúng thì thực hiện khối lệnh;

lặp lại kiểm tra điều kiện **nếu sai** thoát khỏi vòng lặp

Biểu thức 1: khởi tạo giá trị ban đầu cho biến điều khiển.

Biểu thức 2: là quan hệ logic thể hiện điều kiện tiếp tục vòng lặp.

Biểu thức 3: phép gán dùng thay đổi giá trị biến điều khiển.

Nhận xet:

Biểu thức 1 bao giờ cũng chỉ được tính toán một lần khi gọi thực hiện for.

Biểu thức 2, 3 và thân for có thể thực hiện lặp lại nhiều lần.

Lưu ý:

+ Biểu thức 1, 2, 3 phải phân cách bằng dấu chấm phẩy (;)

+ Nếu biểu thức 2 không có, vòng ***for*** được xem là luôn luôn đúng. Muốn thoát khỏi

vòng lặp for phải dùng một trong 3 lệnh ***break, goto*** hoặc ***return.***

+ Với mỗi biểu thức có thể viết thành một dãy biểu thức con phân cách nhau bởi dấu phẩy. Khi đó các biểu thức con được xác định từ trái sang phải. Tính đúng sai của dãy biểu thức con trong biểu thức thứ 2 được xác định bởi biểu thức con cuối cùng.

+ Trong thân ***for*** (khối lệnh) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.

+ Khi gặp lệnh ***break,*** cấu trúc lặp sâu nhất sẽ thoát ra.

+ Trong thân for có thể dùng lệnh goto để thoát khỏi vòng lặp đến vị trí mong muốn.

+ Trong thân ***for*** có thể sử dụng ***return*** để trở về một hàm nào đó.

+ Trong thân ***for*** có thể sử dụng lệnh ***continue*** để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

Ví dụ 1: Viết chương trình in ra câu "Vi du su dung vong lap for" 3 lần.

Viết chương trình

```
/* Chuong trinh in ra cau "Vi du su dung vong lap for" 3 lan */
 #include <stdio.h>
#include <conio.h>
 #define MSG "Vi du su dung vong lap for.\n"
 void main(void)
        {
                int i;
                for(i = 1; i<=3; i++)    /hoac for(i = 1; i<=3; i+=1)
                printf("%s", MSG);
                getch();
    }
```

Kết quả in ra màn hình

```
Vi du su dung vong lap for.
Vi du su dung vong lap for.
Vi du su dung vong lap for.
```

Lưu y´1: Có dấu chấm phẩy sau lệnh for(i=1; i<=3; i++); → các lệnh thuộc vòng lặp for sẽ không được thực hiện.

Ví dụ 2: Viết chương trình nhập vào 3 số nguyên. Tính và in ra tổng của chúng.

Viết chương trình

```
/* Chuong trinh nhap vao 3 so va tinh tong */
 #include <stdio.h>
#include <conio.h>
 void main(void)
        {
                int i, in, is;
                is = 0;
               for(i = 1; i<=3; i++)
               {
                        printf("Nhap vao so thu %d :", i);
                        scanf("%d", &in);
                        is = is + in;
                }
                printf("Tong: %d", is);
                getch();
        }
```

Kết quả in ra màn hình

```
Nhap vao so thu 1: 5
Nhap vao so thu 2: 4
Nhap vao so thu 3: 2
Tong: 11.
_
```

Lưu y´2:

Trong vòng lặp for có sử dụng từ 2 lệnh trở lên, nhớ sử dụng cặp ngoặc { } để bọc các lệnh đó lại. Dòng 12, 13, 14 thuộc vòng for dòng 10 do được bọc bởi

cặp ngoặc { }. Nếu 3 dòng này không bọc bởi cặp ngoặc { }, thì chỉ dòng 12 thuộc vòng lặp for, còn 2 dòng còn lại không thuộc vòng lặp for.

Ví dụ 3: Viết chương trình nhập vào số nguyên n. Tính tổng các giá trị lẻ từ 0 đến n.

Viết chương trình

```
/* Chuong trinh nhap vao 3 so va tinh tong */
 #include <stdio.h>
#include <conio.h>
 void main(void)
        {
                int i, in, is = 0;
                printf("Nhap vao so n: ");
                scanf("%d", &in);
                is = 0;
                for(i = 0; i<=in; i++)
                        {
                                if (i % 2 != 0)   //neu i la so le
                                    is = is + i;    //hoac is += i;
                        }
                 printf("Tong: %d", is);
                getch();
        }
```

Kết quả in ra màn hình

```
Nhap vao so n : 5
Tong: 9.
```

Lưu y´3:

Ta có thể viết gộp các lệnh trong thân for vào trong lệnh for. Tuy nhiên, khi lập trình ta nên viết lệnh for có đủ 3 biểu thức đơn và các lệnh thực hiện trong thân for mỗi lệnh một dòng để sau này có thể đọc lại dễ hiểu, dễ sửa chữa.

Ví dụ 4: Một vài ví dụ thay đổi biến điều khiển vòng lặp.

- Thay đổi biến điều khiển từ 1 đến 100, mỗi lần tăng 1:

for(i = 1; i <= 100; i++)

- Thay đổi biến điều khiển từ 100 đến 1, mỗi lần giảm 1:

for(i = 100; i >= 1; i--)

- Thay đổi biến điều khiển từ 7 đến 77, mỗi lần tăng 7:

for(i = 7; i <= 77; i += 7)

- Thay đổi biến điều khiển từ 20 đến 2, mỗi lần giảm 2:

for(i = 20; i >= 2; i –= 2)

Ví dụ 5: Đọc vào một loạt kí tự trên bàn phím, đếm số kí tự nhập vào. Kết thúc khi gặp dấu chấm '.' .

Viết chương trình

```
/* Doc vao 1 loat ktu tren ban phim, dem so ktu nhap vao. Ket thuc khi gap dau cham */
 #include <stdio.h>
#include <conio.h>
 #define  DAU_CHAM     '.'
 void main(void)
        {
                char c;
                 int idem;
                 for(idem = 0; (c = getchar()) != DAU_CHAM; )
                        idem++;
                 printf("So ki tu: %d.\n", idem);
```

```
        getch();
    }
```

Kết quả in ra màn hình

```
        afser.
        So ki tu: 5.
```

Lưu y´4: Vòng lặp for vắng mặt biểu thức 3

Ví dụ 6: Đọc vào một loạt kí tự trên bàn phím, đếm số kí tự nhập vào. Kết thúc khi gặp dấu chấm '.' ..

Viết chương trình

```
/* Doc vao 1 loat ktu tren ban phim, dem so ktu nhap vao. Ket thuc khi gap dau cham */
 #include <stdio.h>
#include <conio.h>
 #define  DAU_CHAM     '.'
 void main(void)
    {
        char c;
        int idem = 0;
        for(; ;)
            {
                c = getchar();
                if (c == DAU_CHAM)  //nhap vao dau cham
                    break;       //thoat vong lap
                idem++;
            }
        printf("So ki tu: %d.\n", idem);
        getch();
    }
```

Kết quả in ra màn hình

afser.

So ki tu: 5.

Lưu ý5: Vòng lặp for vắng mặt cả ba biểu thức.

2. Lệnh break

Thông thường lệnh ***break*** dùng để thoát khỏi vòng lặp không xác định điều kiện dừng hoặc bạn muốn dừng vòng lặp theo điều kiện do bạn chỉ định. Việc dùng lệnh **break** để thoát khỏi vòng lặp thường sử dụng phối hợp với lệnh if. Lệnh ***break*** dùng trong **for, while, do…while, switch**. Lệnh ***break*** thoát khỏi vòng lặp chứa nó.

Chú ý:

Khi có nhiều vòng lặp hoặc switch lồng nhau thì câu lệnh break chỉ thoát khỏi vòng lặp sâu nhất (trong cùng) có chứa toán tử break đó mà thôi.

Mọi toán tử break đều có thể thay bằng toán tử goto với nhãn thích hợp

break thường được sử dụng khi vòng lặp for có thể thực hiện số lần lặp ít hơn số lần lặp đã được xác định trước, trong thân của switch và trong các trường hợp ta phải kiểm tra điều kiện kết thúc của vòng lặp while bên trong thân của nó.

Ví dụ 7 : Như ví dụ 6

Sử dụng lệnh **break** trong **switch** để nhảy bỏ các câu lệnh kế tiếp còn lại.

3. Lệnh continue

Trái với toán tử break (dùng để thoát khỏi vòng lặp) câu lệnh continue dùng để bắt đầu một vòng lặp mới của chu trình sâu nhất chứa toán tử đó. Được dùng trong vòng lặp ***for, while, do…while***. Khi lệnh ***continue*** thi hành quyền

điều khiển sẽ trao qua cho biểu thức điều kiện của vòng lặp gần nhất. Nghĩa là lộn ngược lên đầu vòng lặp, tất cả những lệnh đi sau trong vòng lặp chứa ***continue*** sẽ bị bỏ qua không thi hành.

Khi gặp toán tử ***continue*** bên trong thân của một toán tử for, máy sẽ bỏ qua các câu lệnh còn lại trong thân for (sau ***continue)*** chuyển sang thực hiện biểu thức 3 để khởi đầu cho vòng lặp mới tiếp theo.

Khi gặp toán tử ***continue*** bên trong thân của vòng lặp while hoặc do… while máy sẽ bỏ qua các lệnh còn lại trong thân vòng lặp (sau ***continue)*** chuyển tới ***Kiểm tra biểu thức*** sau từ khóa while để khởi động cho vòng lặp tiếp theo (nếu *Biểu thức còn đúng*) hoặc thoát khỏi vòng lặp (nếu *Biểu thức sai*)

Ví dụ 8: Nhập vào 1 dãy số nguyên từ bàn phím đến khi gặp số 0 thì dừng. In ra tổng các số nguyên dương.

Viết chương trình

```
/* Nhap vao 1 day so nguyen tu ban phim den khi gap so 0 thi dung. In ra tong
cac so nguyen duong */
#include <stdio.h>
#include <conio.h>
 void main(void)
    {
          int in, itong = 0;
           for(; ;)
                 {
                       printf("Nhap vao 1 so nguyen: ");
                       scanf("%d", &in);
                       if (in < 0)
                             continue;  //in < 0 quay nguoc len dau vong lap
```

```
            if (in == 0)
                break;   //in = 0 thoat vong lap
            itong += in;
        }
    printf("Tong: %d.\n", itong);
    getch();
}
```

Kết quả in ra màn hình

```
Nhap vao 1 so nguyen: -8
Nhap vao 1 so nguyen: 9
Nhap vao 1 so nguyen: -7
Nhap vao 1 so nguyen: 3
Nhap vao 1 so nguyen: 0
Tong: 12
_
```

4. Lệnh while

Vòng lặp thực hiện lặp lại trong khi biểu thức còn đúng.

Cu´pháp lệnh

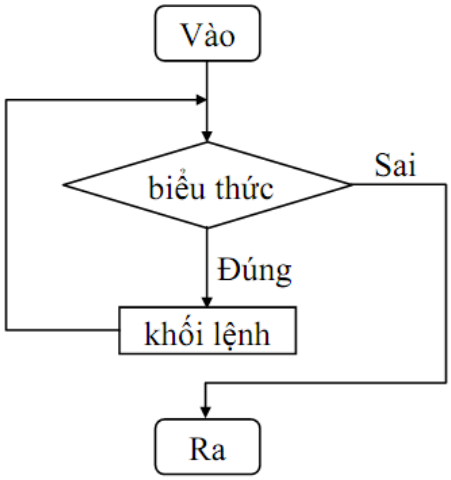
while (biểu thức)

khối lệnh;

Từ khóa while phải viết bằng chữ thường

Nếu khối lệnh bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

Lưu đồ`



Trước tiên **biểu thức** được kiểm tra **nếu sai** thì kết thúc vòng lặp while

(khối lệnh không được thi hành 1 lần nào)

Nếu đúng thực hiện khối lệnh;

Lặp lại kiểm tra biểu thức

+ Biểu thức: có thể là một biểu thức hoặc nhiều biểu thức con. Nếu là nhiều biểu thức con thì cách nhau bởi dấu phẩy (,) và tính đúng sai của biểu thức được quyết định bởi biểu thức con cuối cùng.

+ Trong thân while (khối lệnh) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.

+ Trong thân ***while*** có thể sử dụng lệnh ***continue*** để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

+ Muốn thoát khỏi vòng lặp ***while*** tùy ý có thể dùng các lệnh ***break***, ***goto, return*** như lệnh ***for.***

Ví dụ 9: Viết chương trình in ra câu "Vi du su dung vong lap while" 3 lần.

Viết chương trình

```
/* Chuong trinh in ra cau "Vi du su dung vong lap while" 3 lan */
 #include <stdio.h>
#include <conio.h>
```

```
#define MSG "Vi du su dung vong lap while.\n"
void main(void)
   {
          int i = 0;
          while (i++ < 3)
                printf("%s", MSG);
          getch();
   }
```

Kết quả in ra màn hình

```
          Vi du su dung vong lap while.
          Vi du su dung vong lap while.
          Vi du su dung vong lap while.
```

Ví dụ 10: Viết chương trình tính tổng các số nguyên từ 1 đến n, với n được nhập vào từ bàn phím.

Viết chương trình

```
/* Chuong trinh tính tong cac so nguyen tu 1 den n */
#include <stdio.h>
#include <conio.h>
void main(void)
   {
          int i = 0, in, is = 0;
          printf("Nhap vao so n: ");
          scanf("%d", &in);
          while (i++ < in)
                is = is + i;    //hoac is += i;
          printf("Tong: %d", is);
          getch();
   }
```

Kết quả in ra màn hình

Nhap vao so n : 5

Tong: 15.

Ví dụ 11: Thay dòng for(; (c = getchar()) != DAU_CHAM;) ở ví dụ 4 thành dòng while ((c = getchar()) != DAU_CHAM)) Chạy lại chương trình, quan sát và nhận xét kết quả.

5. Lệnh do…while

Vòng lặp thực hiện lặp lại cho đến khi biểu thức sai.

Cu´pháp lệnh

do

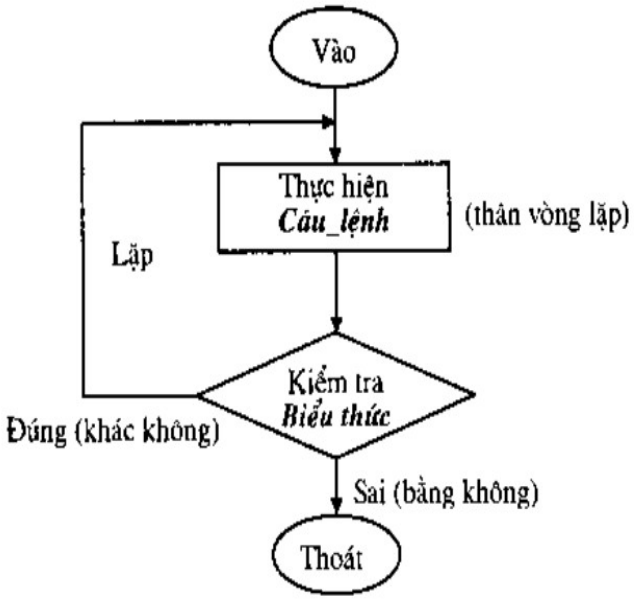
khối lệnh;

while (biểu thức);

Từ khóa do, while phải viết bằng chữ thường

Nếu khối lệnh bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

Lưu đồ`



Bước 1. Thực hiện khối lệnh

Bước 2. Kiểm tra biểu thức

+ Nếu đúng (có giá trị khác 0) thì quay lại bước 1

+ Nếu sai thì chuyển sang bước 3.

Bước 3. Kết thúc vòng lặp

Biểu thức: có thể là một biểu thức hoặc nhiều biểu thức con. Nếu là nhiều biểu thức con thì cách nhau bởi dấu phẩy (,) và tính đúng sai của biểu thức được quyết định bởi biểu thức con cuối cùng.

Trong thân ***do…while*** (khối lệnh) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.

Trong thân ***do…while*** có thể sử dụng lệnh ***continue*** để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

Muốn thoát khỏi vòng lặp ***do…while*** tùy ý có thể dùng các lệnh ***break, goto, return.***

Vi dụ 11: Viết chương trình kiểm tra password.

Viết chương trình

```
/* Chuong trinh kiem tra mat khau */
#include <stdio.h>
 # define PASSWORD  12345
 void main(void)
    {
            int in;
           do
           {
                printf("Nhap vao password: ");
                scanf("%d", &in);
           } while (in != PASSWORD)
```

}

In kết quả ra màn hình

Nhap vao password: 1123

Nhap vao password: 12346

Nhap vao password: 12345

Ví dụ 12: Viết chương trình nhập vào năm hiện tại, năm sinh. In ra tuoi.

Viết chương trình

```
* Chuong trinh in tuoi */


#include <stdio.h>
# define CHUC  "Chuc ban vui ve (: >\n"
void main(void)
   {
        unsigned char choi;
        int inamhtai, inamsinh;
        do
        {
             printf("Nhap vao nam hien tai: ");
             scanf("%d", inamhtai);
             printf("Nhap vao nam sinh: ");
             scanf("%d", inamsinh);
             printf("Ban %d tuoi, %s", inamhtai – inamsinh, CHUC);
             printf("Ban co muon tiep tuc? (Y/N)\n");
             choi = getch();
         } while (choi == 'y' || choi == 'Y');
   }
```

In kết quả ra màn hình

Nhap vao nam hien tai: 2002

Nhap vao nam sinh: 1980

Ban 22 tuoi, chuc ban vui ve (:>

Ban co muon tiep tuc? (Y/N)

_ (nếu gõ y hoặc Y tiếp tục thực hiện

chương trình, ngược lại gõ các phím

khác chương trình sẽ thoát)

6. Vòng lặp lồng nhau

Ví dụ 13: Vẽ hình chữ nhật đặc bằng các dấu ''*

Viết chương trình

```
/* Ve hinh chu nhat dac */
 #include <stdio.h>
#include <conio.h>

void main(void)
    {
            int i, ij, idai, irong;
            printf("Nhap vao chieu dai: ");
            scanf("%d", &idai);
            printf("Nhap vao chieu rong: ");
            scanf("%d", &irong);
            for (i = 1; i <= irong; i++)
            {
                    for (ij = 1; ij <= idai; ij++)  //in mot hang voi chieu dai dau *
                            printf("*");
                    printf("\n");       //xuong dong khi in xong 1 hang
            }
    getch();
```

```
}
```

In kết quả ra màn hình

```
Nhap vao chieu dai: 10
Nhap vao chieu rong: 5
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
```

Ví dụ 14: Vẽ hình chữ nhật đặc có chiều rộng = 10 hàng. Hàng thứ 1 = 10 số 0, hàng thứ 2 = 10 số 1…

Viết chương trình

```
/* Ve hinh chu nhat bang cac so tu 0 den 9 */
 #include <stdio.h>
#include <conio.h>
 void main(void)
{
        int i = 0, ij;
        while (i <= 9)
    {
        ij = 0;       //khoi tao lai ij = 0 cho lan in ke tiep
        while (ij++ <= 9)   //in 1 hang 10 so i
                printf("%d", i);
        printf("\n");     //xuong dong khi in xong 1 hang
        i++;        //tang i len 1 cho vong lap ke tiep
    }
    getch();
}
```

In kết quả ra màn hình

0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1

2 2 2 2 2 2 2 2 2 2

3 3 3 3 3 3 3 3 3 3

4 4 4 4 4 4 4 4 4 4

5 5 5 5 5 5 5 5 5 5

6 6 6 6 6 6 6 6 6 6

7 7 7 7 7 7 7 7 7 7

8 8 8 8 8 8 8 8 8 8

9 9 9 9 9 9 9 9 9 9

Lưu ý: Các lệnh lặp for, while, do…while có thể lồng vào chính nó, hoặc lồng vào lẫn

nhau. Nếu không cần thiết không nên lồng vào nhiều cấp dễ gây nhầm lẫn khi lập trình

cũng như kiểm soát chương trình.

7. So sánh sự khác nhau của các vòng lặp

- Vòng lặp for thường sử dụng khi biết được số lần lặp xác định.
- Vòng lặp thường while, do…while sử dụng khi không biết rõ số lần lặp.
- Khi gọi vòng lặp while, do…while, nếu biểu thức sai vòng lặp while sẽ không được

thực hiện lần nào nhưng vòng lặp do…while thực hiện được 1 lần.

➔ Số lần thực hiện ít nhất của while là 0 và của do…while là 1

Bai tập hết chương

Bai 1. Viết chương trình in ra bảng mã ASCII

Bai 2. Viết chương trình tính tổng bậc 3 của N số nguyên đầu tiên.

Bai 3. Viết chương trình nhập vào một số nguyên rồi in ra tất cả các ước số của số đó.

Bai 4. Viết chương trình vẽ một tam giác cân bằng các dấu *

Bai 5. Viết chương trình tính tổng nghịch đảo của N số nguyên đầu tiên theo công thức

S = 1 + 1/2 + 1/3 + … + 1/N

Bai 6. Viết chương trình tính tổng bình phương các số lẻ từ 1 đến N.

Bai 7. Viết chương trình nhập vào N số nguyên, tìm số lớn nhất, số nhỏ nhất.

Bai 8. Viết chương trình nhập vào N rồi tính giai thừa của N.

Bai 9. Viết chương trình tìm USCLN, BSCNN của 2 số.

Bai 10. Viết chương trình vẽ một tam giác cân rỗng bằng các dấu *.

Bai 11. Viết chương trình vẽ hình chữ nhật rỗng bằng các dấu *.

Bai 12. Viết chương trình nhập vào một số và kiểm tra xem số đó có phải là số nguyên tố hay không?

Bai 13. Viết chương trình tính số hạng thứ n của dãy Fibonaci.

Dãy Fibonaci là dãy số gồm các số hạng p(n) với:

p(n) = p(n-1) + p(n-2) với n>2 và p(1) = p(2) = 1

Dãy Fibonaci sẽ là: 1 1 2 3 5 8 13 21 34 55 89 144…

Bai 14. Viết chương trình tính giá trị của đa thức

$Pn = a_n x^n + a_{n-1} x^{n-1} \ldots\ldots\ldots\ldots + a_1 x^{1+} a_0 x^0$

Hướng dẫn đa thức có thể viết lại

$Pn = (\ldots(a_n x + a_{n-1})x + a_{n-2})x + \ldots + a0$

Như vậy trước tiên tính $a_n x + a_{n-1}$, lấy kết quả nhân với x, sau đó lấy kết quả nhân với

x cộng thêm a_{n-2} , lấy kết quả nhân với x … n gọi là bậc của đa thức.

Chương V. Hàm

Trong thực tế khi ta muốn giải quyết một công việc phức tạp nào đó, ta thường chia nhỏ công việc đó thành các công việc nhỏ hơn và tất nhiên những công việc nhỏ này lại thực hiện dễ dàng hơn rất nhiều. Thực vậy, trong lập trình ta cũng có nhu cầu chia nhỏ chương trình phức tạp thành những chương trình nhỏ hơn, đơn giản và dễ hiểu. Mỗi chương trình nhỏ đó được gọi là hàm.

1. Các ví dụ về hàm

a. Khái niệm

Hàm là một đoạn chương trình có tên và có chức năng giải quyết một số vấn đề chuyên biệt cho chương trình chính, nó có thể được gọi nhiều lần với các tham số khác nhau và trả lại một giá trị nào đó cho chương trình gọi nó.

Hàm thường được sử dụng khi:

- **Nhu cầu tái sử dụng:** có một số công việc được thực hiện ở nhiều nơi (cùng một chương trình hoặc ở nhiều chương trình khác nhau), bản chất không đổi nhưng giá trị các tham số cung cấp khác nhau ở từng trường hợp.
- **Nhu cầu sửa lỗi và cải tiến:** giúp phân đoạn chương trình để chương trình được trong sáng, dễ hiểu và do đó rất dễ dàng phát hiện lỗi cũng như cải tiến chương trình.

b. Cú pháp

Hàm có cấu trúc tổng quát như sau:

```
<kiểu trả về> <tên hàm>([<danh sách tham số>])
{
     <các câu lệnh>
     [return <giá trị>;]
}
```

Trong đó:

- <kiểu trả về>: là bất kỳ kiểu dữ liệu nào của C như char, int, long, float hay double… Nếu hàm đơn thuần chỉ thực hiện một số câu lệnh mà không cần trả về cho chương trình gọi nó thì kiểu trả về này là void.
- <tên hàm>: là tên gọi của hàm và được đặt theo quy tắc đặt tên/định danh.
- <danh sách tham số>: xác định các đối số sẽ truyền cho hàm. Các tham số này giống như khai báo biến và cách nhau bằng dấu phẩy. Hàm có thể không có đối số nào.
- <các câu lệnh>: là các câu lệnh sẽ được thực hiện mỗi khi hàm được gọi.
- <giá trị>: là giá trị trả về cho hàm thông qua câu lệnh return.

Ví dụ 1: Hàm sau đây có tên là Tong, nhận vào hai đối số kiểu nguyên và trả về tổng của hai số nguyên đó.

```
/* Ham ten Tong
Nhan vao hai so nguyen va tra ve mot so nguyen */
int Tong(int a, int b)
{
        return a + b;
}
```

Ví dụ 2: Hàm sau đây có tên là Xuat, nhận vào một đối số kiểu nguyên và xuất số nguyên đó ra màn hình. Hàm này không trả về gì cả.

```
void Xuat(int n)
{
        printf("%d", n);
}
```

Ví dụ 3: Hàm sau đây có tên là Nhap, không nhận đối số nào cả và trả về giá trị số nguyên người dùng nhập vào.

```
int Nhap()
```

```
{
        int n;
        printf("Nhap mot so nguyen: ");
       scanf("%d", &n);
       return n;
}
```

Lưu ý:

- Hàm phải được khai báo và định nghĩa trước khi sử dụng và thường đặt ở trên hàm chính (hàm main).

```
int Tong(int a, int b)
{
        return a + b;
}
void main()
{
       int a = 2912, b = 1706;
        int sum = Tong(a, b); // Loi goi ham
}
```

- Thông thường, trước hàm main ta chỉ xác định tên hàm, các tham số và giá trị trả về của hàm để thông báo cho các hàm bên dưới biết cách sử dụng của nó còn phần định nghĩa hàm sẽ được đưa xuống dưới cùng. Phần ở trên này được gọi là nguyên mẫu hàm (function prototype). Nguyên mẫu hàm chính là tiêu đề hàm được kết thúc bằng dấu chấm phẩy.

```
int Tong(int a, int b); // prototype ham Tong


void main()
{
```

```
    int a = 2912, b = 1706;
    int sum = Tong(a, b); // Loi goi ham
}

int Tong(int a, int b) // Mo ta ham Tong
{
    return a + b;
}
```

- Trên thực tế, nguyên mẫu hàm không cần thiết phải giống tuyệt đối tiêu đề hàm. Tên tham số có thể khác hoặc bỏ luôn miễn là cùng kiểu. Tuy nhiên, không nên để chúng khác nhau vì như vậy sẽ gây rối cho chương trình.

  Ví dụ sau cho thấy có thể bỏ hẳn tên tham số:

```
int Tong(int, int); // prototype ham Tong
…
```

c. Phạm vi của biến và hàm

Là phạm vi hiệu quả của biến hoặc hàm. Phạm vi này bao gồm bản thân khối đó và các khối con bên trong nó. Các khối cha hoặc các khối ngang hàng sẽ không thuộc phạm vi này.

Ví dụ 4:

```
int a;
        int Ham1()
        {
                int a1;
        }

        int Ham2();
        {
                {
                        int a21;
                }
        }

        void main()
        {
                int a3;
        }
```

Nhận xét:

- Các hình chữ nhật bao quanh tạo thành một khối. Mộtkhối có thể chứa khối con trong nó.
- Biến khai báo trong khối nào thì chỉ có tác dụng trong khối đó và các khối con của nó, không có tác dụng với khối cùng cấp.
- Biến khai báo trong khối lớn nhất (chứa tất cả các khốikhác) là biến toàn cục.
- Biến khai báo trong các hàm (hoặc khối) là cục bộ, sẽ bị mất khi kết thúc hàm (hoặc khối).
- Hàm cùng một khối (cùng cấp) có thể gọi lẫn nhau nhưng phải tuân theo thứ tự khai báo.
- Các biến cục bộ nên đặt khác tên với các biến ở khối cha để tránh nhầm lẫn. Trong trường hợp đặt trùng tên thì biến được ưu tiên là biến cục bộ của khối con.

Giải thích chương trình trên: *a* là biến toàn cục, có thể sử dụng ở bất cứ đâu. *a1, a2, a21, a3* là các biến cục bộ do được khai báo trong hàm (hoặc khối). *a1* chỉ có tác dụng trong hàm *Ham1; a2* có tác dụng trong thủ tục *Ham2* và khối trong *Ham2; a21* chỉ có tác dụng trong khối mà nó khai báo; a2 chỉ có tác dụng trong hàm *main*.

Hàm *main* có thể gọi *Ham1, Ham2. Hàm 2* có thể gọi *Ham1*.

d. Các ví dụ về hàm

Ví dụ 5:

Chương trình

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  // khai bao prototype
5  void line();
6
7  // ham in 1 dong dau
8  void line()
9  {
10     int i;
11     for(i = 0; i < 19; i++)
12        printf("*");
13     printf("\n");
14 }
15
16 void main(void)
17 {
18     line();
19     printf("* Minh hoa ve ham *");
20     line();
21     getch();
22 }
```

Kết quả in ra màn hình

```
******************
* Minh hoa ve ham *
******************

_
```

Giải thích chương trình

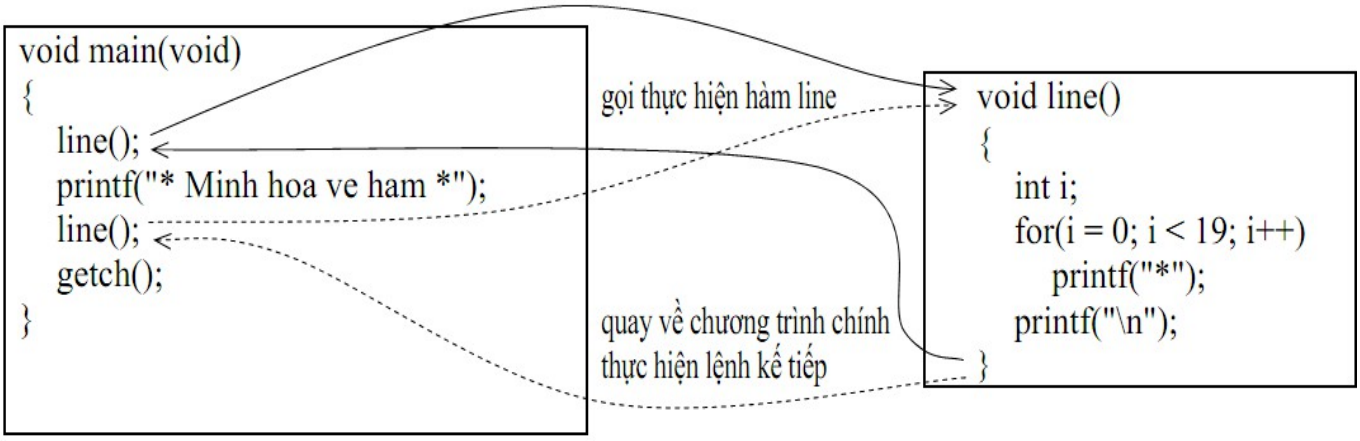
Dòng 8 đến dòng 14: định nghĩa hàm line, hàm này không trả về giá trị, thực hiện công việc in ra 19 dấu sao.

Dòng 5: khai báo prototype, sau tên hàm phải có dấu chầm phẩy

Trong hàm line có sử dụng biến i, biến i là biến cục bộ chỉ sử dụng được trong phạm vi hàm line.

Dòng 18 và 20: gọi thực hiện hàm line.

Trình tự thực hiện chương trình



Lưu ý: Không có dấu chấm phẩy sau tên hàm, phải có cặp dấu ngoặc ( ) sau tên hàm nếu

hàm không có tham số truyền vào. Phải có dấu chấm phẩy sau tên hàm khai báo prototype. Nên khai báo prototype cho dù hàm được gọi nằm trước hay sau câu lệnh gọi nó.

Ví dụ 6:

Chương trình

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  // khai bao prototype
5  int power(int, int);
6
7  // ham tinh so mu
8  int power(int ix, int in)
9  {
10    int i, ip = 1;
11    for(i = 1; i <= in; i++)
12       ip *= ix;
13    return ip;
14 }
15
16 void main(void)
17 {
18    printf("2 mu 2 = %d.\n", power(2, 2));
19    printf("2 mu 3 = %d.\n", power(2, 3));
20    getch();
   }
```

Kết quả in ra màn hình

2 mu 2 = 4.

2 mu 3 = 8.

–

Giải thích chương trình

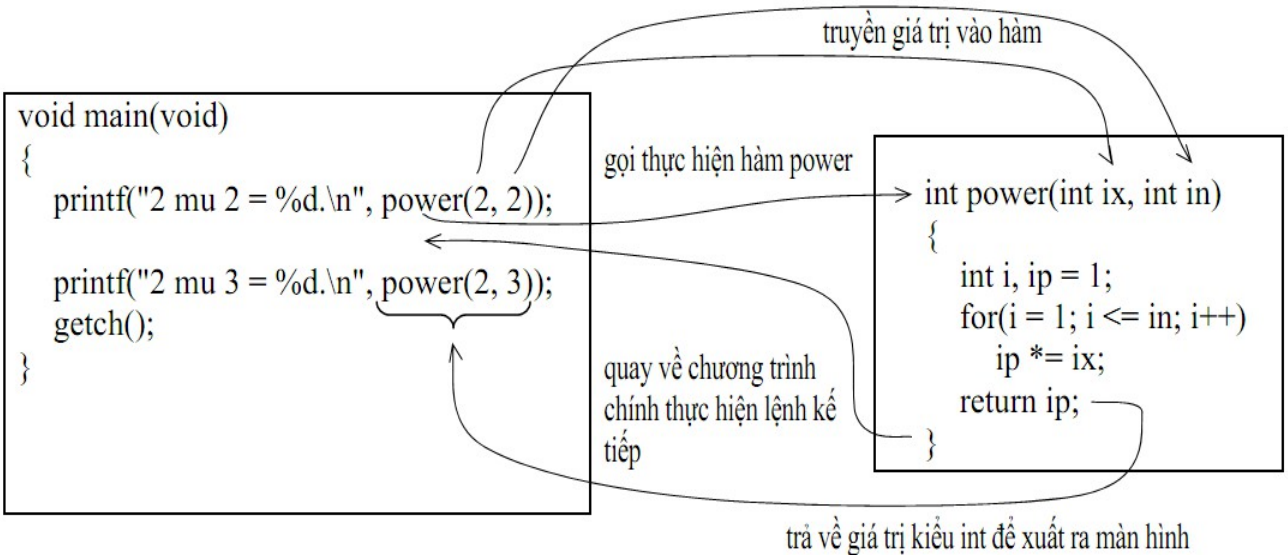
Hàm power có hai tham số truyền vào là ix, in có kiểu int và kiểu trả về cũng có kiểu int.

Dòng 13: return ip: trả về giá trị sau khi tính toán

Dòng 18: đối mục 2 và 3 có kiểu trả về là int sau khi thực hiện gọi power.

Hai tham số ix, in của hàm power là dạng truyền tham trị.

Trình tự thực hiện chương trình



Lưu ý: Quy tắc đặt tên hàm giống tên biến, hằng… Mỗi đối số cách nhau = dấu phẩy kèm

theo kiểu dữ liệu tương ứng.

Ví dụ 7:

Chương trình

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  // khai bao prototype
5  void time(int & , int &);
6
7  // ham doi phut thanh gio:phut
8  void time(int &ig, int &ip)
9  {
10    ig = ip / 60;
11    ip %= 60;
12 }
13
14 void main(void)
15 {
16    int igio, iphut;
```

```
17      printf("Nhap vao so phut : ");
18      scanf("%d", &iphut);
19      time(igio, iphut);
20      printf("%02d:%02d\n", igio, iphut);
21      getch();
22  }
```

Kết quả in ra màn hình

Nhap vao so phut: 185

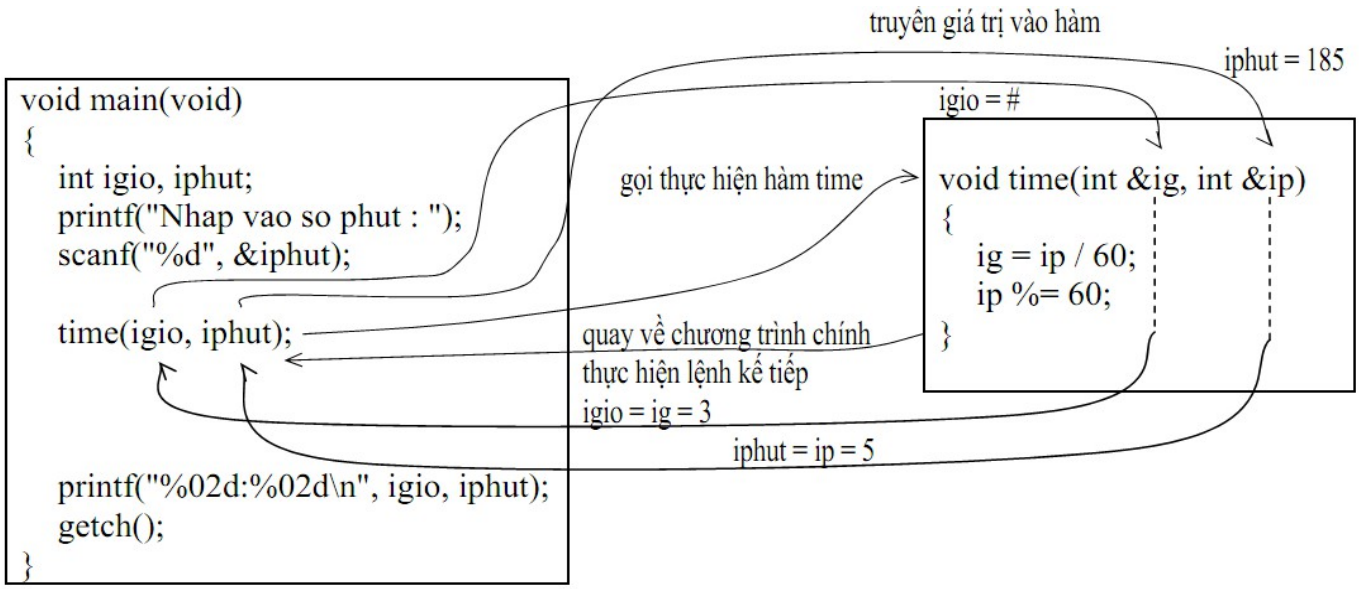
03:05

_

Giải thích chương trình

Hàm time có hai tham số truyền vào là ix, in có kiểu int. 2 tham số này có toán tử địa chỉ & đi trước cho biết 2 tham số này là dạng truyền tham biến.

Trình tự thực hiện chương trình



2. Tham sô´dạng tham biến va`tham trị

Ví dụ 8:

Chương trình

```
void thamtri(int ix, int iy)
    {
```

```
        ix += 1;  //cong ix them 1
        iy += 1;  //cong iy them 1
    }
void thambien(int &ix, int &iy)
    {
        ix += 1;  //cong ix them 1
        iy += 1;  //cong iy them 1
    }
void main(void)
    {
        int ia = 5, ib = 5;
        thamtri(ia, ib);
        printf("a = %d, b = %d", ia, ib);
        thambien(ia, ib);
        printf("a = %d, b = %d", ai, ib);
}
```

Kết quả in ra màn hình

```
a = 5, b = 5

a = 6, b = 6
```

Lưu ý: Đối với hàm sử dụng lệnh ***return*** bạn chỉ có thể trả về duy nhất 1 giá trị mà thôi. Để có thể trả về nhiều giá trị sau khi gọi hàm bạn sử dụng hàm truyền nhiều tham số dạng tham biến.

3. Sử dụng biến toàn cục

Vì các hàm đều có thể truy nhập và thay đổi giá trị trên các biến toàn cục, cho nên kết quả của hàm này có thể truyền sang hàm khác.

Đặc điểm: không cần truyền tham số cho hàm thông qua các tham số hình thức (vì các hàm xử lý trực tiếp trên các biến toàn cục) cho nên đơn giản. Tuy nhiên, khi viết chương trình ta nên hạn chế sử dụng biến toàn cục nếu không cần thiết (vì dễ gây ra hiệu ứng lề không mong muốn, cấu trúc chương trình không sáng sủa và mất tính riêng tư của các hàm).

Ví dụ 9:

Chương trình

```
#include <stdio.h>
#include <conio.h>
 // khai bao prototype
void oddeven();
void negative();
//khai bao bien toan cuc
int inum;
 void main(void)
    {
            printf("Nhap vao 1 so nguyen : ");
            scanf("%d", &inum);
            oddeven();
            negative();
            getch();
    }
 // ham kiem tra chan le
void oddeven()
    {
            if (inum % 2)
                    printf("%d la so le.\n", inum);
            else
```

```
            printf("%d la so chan.\n", inum);
    }
 //ham kiem tra so am
void negative()
    {
        if (inum < 0)
            printf("%d la so am.\n", inum);
        else
            printf("%d la so duong.\n", inum);
    }
```

Kết quả in ra màn hình

```
Nhap vao 1 so nguyen: 3
3 la so le.
3 la so duong.
```

Giải thích chương trình

Chương trình trên gồm 2 hàm oddeven và negative, 2 hàm này bạn thấy không có tham số để truyền biến inum vào xử lý nhưng vẫn cho kết quả đúng. Do chương trình sử dụng biến inum toàn cục (dòng.9) nên biến này có ảnh hưởng đến toàn bộ chương trình mỗi khi gọi và sử dụng nó. Xét tình huống sau: Giả sử trong hàm negative ta khai báo biến inum có kiểu int như sau:

```
void negative()
{
 int inum;
 ….
}
```

Khi đó chương trình sẽ cho kết quả sai! Do các câu lệnh trong hàm negative sử dụng biến inum sẽ sử dụng biến inum khai báo trong hàm negative và lúc này biến inum toàn cục không có tác dụng đối với các câu lệnh trong hàm này. Biến inum khai báo trong hàm negative chỉ có ảnh hưởng trong phạm vi hàm và chu trình sống của nó bắt đầu từ lúc gọi hàm đến khi thực hiện xong.

Lưu ý: Cẩn thận khi đặt tên biến, xác định rõ phạm vi của biến khi sử dụng để có thể dễ dàng kiểm soát chương trình.

4. Dùng dẫn hướng #define

Sau đây là một vài ví dụ dùng dẫn hướng #define để định nghĩa hàm đơn giản

```
#define AREA_CIRCLE (frad)  (4*PI*frad*frad)  //tinh dien tich hinh cau
#define SUM (x, y)      (x + y)      //cong 2 so
#define SQR (x)       (x*x)       //tinh x binh phuong
#define MAX(x, y)       (x > y) ? x : y    //tim so lon nhat giua x va y
#define ERROR (s)       printf("%s.\n", s)  //in thong bao voi chuoi s
```

Ví dụ 10:

Chương trình

```
#include <stdio.h>
#include <conio.h>
#define MAX(x, y)  (x > y) ? x : y
 void main(void)
    {
          float a = 4.5, b = 6.1;
          printf("So lon nhat la: %5.2f.\n", MAX(a, b));
          getch();
```

}

Kết quả in ra màn hình

So lon nhat la: 6.10

_

Ví dụ 11:

Thêm vào dòng 8 giá trị c = 10

Sửa lại dòng 9: MAX(a, b) thành MAX(MAX(a, b), c)

Chạy lại chương trình, quan sát và nhận xét kết quả

Bai tập hết chương

Bai 1. Viết hàm tính n!

Bai 2. Viết hàm tính tổng S = 1+2+….+n.

Bai 3. Viết hàm kiểm tra số nguyên tố.

Bai 4. Viết hàm tính số hạng thứ n trong dãy Fibonacci.

Bai 5. Viết hàm tìm số lớn nhất trong 2 số.

Chương VI. Mảng va`chuôĩ

1. Mảng

Là tập hợp các phần tử có cùng dữ liệu. Mỗi phần tử của mảng được truy xuất thông qua các chỉ số mô tả vị trí của phần tử đó trong mảng.

Giả sử bạn muốn lưu n số nguyên để tính trung bình, bạn không thể khai báo n biến để lưu n giá trị rồi sau đó tính trung bình.

Ví dụ 1 : Ta muốn tính trung bình 10 số nguyên nhập vào từ bàn phím, bạn sẽ khai báo 10 biến: a, b, c, d, e, f, g, h, i, j có kiểu int và lập thao tác nhập cho 10 biến này như sau:

```
printf("Nhap vao bien a: ");
scanf("%d", &a);
```

10 biến bạn sẽ thực hiện 2 lệnh trên 10 lần, sau đó tính trung bình:

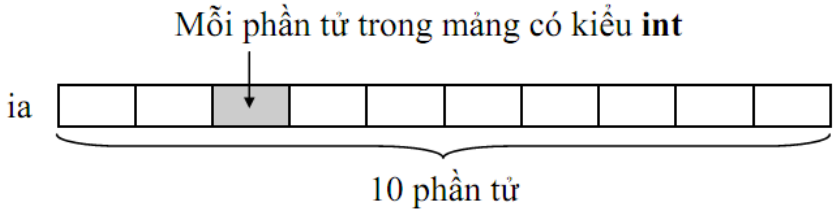
```
(a + b + c + d + e + f + g + h + i + j)/10
```

Điều này chỉ phù hợp với n nhỏ, còn đối với n lớn thì khó có thể thực hiện được. Vì vậy khái niệm mảng được sử dụng

1.1. Cách khai báo mảng

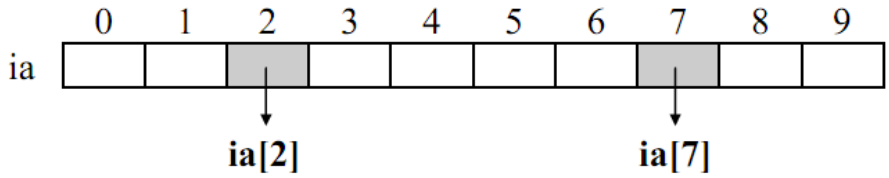
Ví dụ 2 : int ia[10]; với int là kiểu mảng, ia là tên mảng, 10 số phần tử mảng

Ý nghĩa: Khai báo một mảng số nguyên gồm 10 phần tử, mỗi phần tử có kiểu int.



1.2. Tham chiếu đến từng phần tử mảng

Sau khi mảng được khai báo, mỗi phần tử trong mảng đều có chỉ số để tham chiếu. Chỉ số bắt đầu từ 0 đến n-1 (với n là kích thước mảng). Trong ví dụ trên, ta khai báo mảng 10 phần tử thì chỉ số bắt đầu từ 0 đến 9.



ia[2], ia[7]… là phần tử thứ 3, 8… trong mảng xem như là một biến kiểu int.

1.3. Nhập dữ liệu cho mảng

```
for (i = 0; i < 10; i++)   //vòng for có giá trị i chạy từ 0 đến 9
{
  printf("Nhap vao phan tu thu %d: ", i + 1);
  scanf("%d", &ia[i]);
}
```

1.4. Đọc dữ liệu từ mảng

```
for(i = 0; i < 10; i++)
 printf("%3d ", ia[i]);
```

Ví dụ 3 : Viết chương trình nhập vào n số nguyên. Tính và in ra trung bình cộng.

Chương trình

```
/* Tinh trung binh cong n so nguyen */
 #include <stdio.h>
#include <conio.h>
 void main(void)
    {
            int ia[50], i, in, isum = 0;
```

```
            printf("Nhap vao gia tri n: ");
            scanf("%d", &in);
    //Nhap du lieu vao mang
            for(i = 0; i < in; i++)
            {
                printf("Nhap vao phan tu thu %d: ", i + 1);
                scanf("%d", &ia[i]);  //Nhap gia tri cho phan tu thu i
            }
    //Tinh tong gia tri cac phan tu
            for(i = 0; i < in; i++)
                isum += ia[i];    //cong don tung phan tu vao isum
            printf("Trung binh cong: %.2f\n", (float) isum/in);
            getch();
    }
```

Kết quả in ra màn hình

```
Nhap vao gia tri n: 3
Nhap vao phan tu thu 1: 7
Nhap vao phan tu thu 2: 3
Nhap vao phan tu thu 3: 6
Trung binh cong: 5.33
_
```

** Điều gì sẽ xảy ra cho đoạn chương trình trên nếu bạn nhập n > 50 trong khi bạn chỉ khai báo mảng ia tối đa là 50 phần tử. Bạn dùng lệnh if để ngăn chặn điều này trước khi vào thực hiện lệnh for. Thay dòng 7, 8 bằng đoạn lệnh sau :*

```
    do
      {
       printf("Nhap vao gia tri n: ");
       scanf("%d", &in);
```

```
} while (in <= 0 || in > 50);   //chi chap nhan gia tri nhap vao trong khoang
    1..50
```

Chạy chương trình và nhập n với các giá trị -6, 0, 51, 6. Quan sát kết quả.

1.5. Sử dụng kỹ thuật Sentinal

Sử dụng kỹ thuật này để nhập liệu giá trị cho các phần tử mảng mà không biết rõ số lượng phần tử sẽ nhập vào là bao nhiêu (không biết số n).

Ví dụ 4 : Viết chương trình nhập vào 1 dãy số dương rồi in tổng các số dương đó.

Phác họa lời giải: Chương trình yêu cầu nhập vào dãy số dương mà không biết trước số lượng phần tử cần nhập là bao nhiêu, vì vậy để chấm dứt nhập liệu khi thỏa mãn bằng cách nhập vào số âm hoặc không.

Chương trình

```
/* Nhap vao day so nguyen duong, in ra day chan, day le */
 #include <stdio.h>
#include <conio.h>
#define MAX 50
 void main(void)
    {
          float fa[MAX], fsum = 0;
         int i = 0;
          do
                {
                printf("Nhap vao phan tu thu %d: ", i  + 1);
                scanf("%f", &fa[i]);  //Nhap gia tri cho phan tu thu i
         } while (fa[i++] > 0);   //con nhap lieu khi gia tri phan tu > 0
         i--;        //giam i di 1 lan cuoi cung tang 1 truoc khi thoat
         //Tinh tong
```

```
        for(int ij = 0; ij < i; ij++)
                fsum += fa[ij];     //cong don tung phan tu vao isum
         printf("Tong : %5.2f\n", fsum);
        getch();
    }
```

Kết quả in ra màn hình

```
Nhap vao phan tu thu 1: 1.2
Nhap vao phan tu thu 2: 3
Nhap vao phan tu thu 3: 4.6
Nhap vao phan tu thu 4: -9
Tong : 8.80
_
```

** Điều gì sẽ xảy ra cho đoạn chương trình trên nếu bạn nhập số lượng phần tử vượt quá 50 trong khi bạn chỉ khai báo mảng fa tối đa là MAX = 50 phần tử. Bạn dùng lệnh break để thoát khỏi vòng lặp do…while trước khi bước sang phần tử thứ 51. Thêm đoạn lệnh sau vào trước dòng 11:*

```
if(i >= MAX)      //kiem tra phan tu buoc sang 51
   {
    printf("Mang da day!\n"); //thong bao "Mang da day"
    i++;        //tang i len 1 do dong 14 giam i xuong 1
    break;       //thoat khoi vong lap do…while
   }
```

1.6. Khởi tạo mảng

Ví dụ 5 : Có 4 loại tiền 1, 5, 10, 25 và 50 đồng. Hãy viết chương trình nhập vào số tiền sau đó cho biết số số tiền trên gồm mấy loại tiền, mỗi loại bao nhiêu tờ.

Phác họa lời giải: Số tiền là 246 đồng gồm 4 tờ 50 đồng, 1 tờ 25 đồng, 2 tờ 10 đồng, 0 tờ 5 đồng và 1 tờ 1 đồng, Nghĩa là bạn phải xét loại tiền lớn trước, nếu hết khả năng mới xét tiếp loại kế tiếp.

Chương trình

```
/* Nhap vao so tien va doi tien ra cac loai 50, 25, 10, 5, 1 */
 #include <stdio.h>
#include <conio.h>
#define MAX 5
 void main(void)
{
    int itien[MAX] = {50, 25, 10, 5, 1}; //Khai bao va khoi tao mang voi 5 phan tu
     int i , isotien, ito;
    printf("Nhap vao so tien: ");
    scanf("%d", &isotien);   //Nhap vao so tien
    for (i = 0; i < MAX; i++)
    {
            ito = isotien/itien[i];   //Tim so to cua loai tien thu i
            printf("%4d to %2d dong\n", ito, itien[i]);
            isotien = isotien%itien[i]; //So tien con lai sau khi da loai tru cac loai tien da co
    }
 getch();
}
```

Kết quả in ra màn hình

```
Nhap vao so tien: 246
 4 tờ 50 đồng
 1 tờ 25 đồng
```

2 tờ 10 đồng

0 tờ 5 đồng

1 tờ 1 đồng

–

* Điều gì sẽ xảy nếu số phần tử mảng lớn hơn số mục, số phần tử dôi ra không được khởi tạo sẽ điền vào số 0. Nếu số phần tử nhỏ hơn số mục khởi tạo trình biên dịch sẽ báo lỗi.

Ví dụ 6 :

int itien[5] = {50, 25}, phần tử itien[0] sẽ có giá trị 50, itien[1] có giá trị 25, itien[2], itien[3], itien[4] có giá trị 0.

int itien[3] = {50, 25, 10, 5, 1} → trình biên dịch báo lỗi

1.7. Khởi tạo mảng không bao hàm kích thước

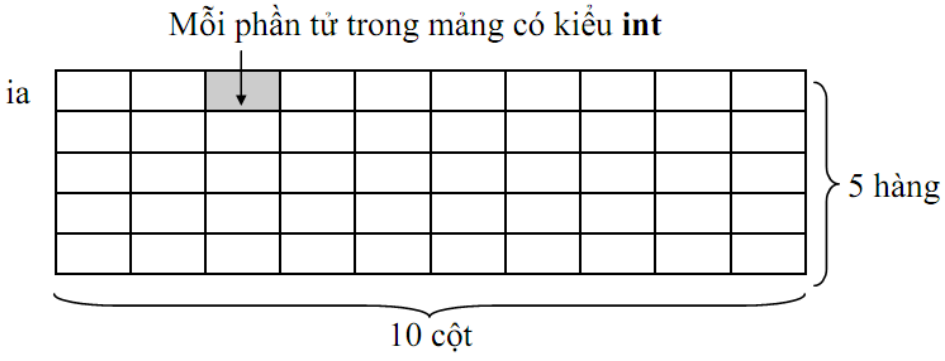
Trong ví dụ trên giả sử ta khai báo int itien[] = {50, 25, 10, 5, 1}. Khi đó trình biên dịch sẽ đếm số mục trong danh sách khởi tạo và dùng con số đó làm kích thước mảng.

1.8. Mảng nhiều chiều

Ví dụ 7 : khai báo mảng 2 chiều int ia[5][10]; với int là kiểu mảng, ia là tên mảng,

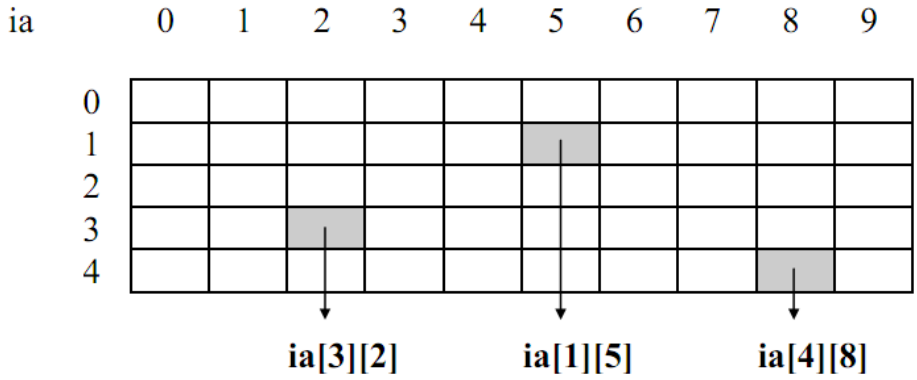
số phần tử mảng là 5 x 10.

Ý nghĩa: Khai báo một mảng 2 chiều số nguyên gồm 50 phần tử, mỗi phần tử có kiểu int.



1.9. Tham chiếu đến từng phần tử mảng 2 chiều

Sau khi được khai báo, mỗi phần tử trong mảng 2 chiều đều có 2 chỉ số để tham chiếu, chỉ số hàng và chỉ số cột. Chỉ số hàng bắt đầu từ 0 đến số hàng – 1 và chỉ số cột bắt đầu từ 0 đến số cột – 1. Tham chiếu đến một phần tử trong mảng 2 chiều ia: ia[chỉ số hàng][chỉ số cột]

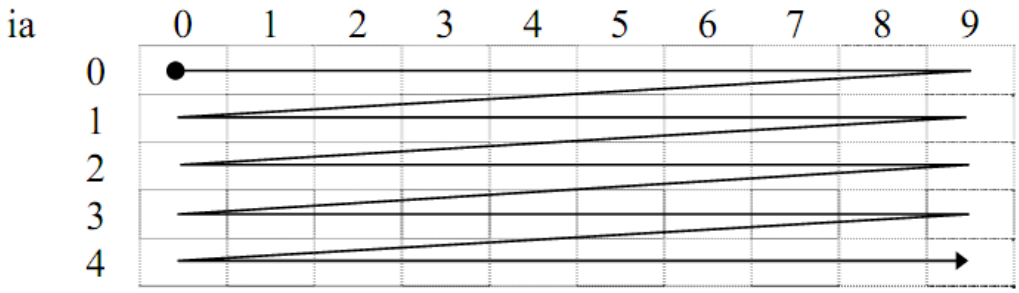


ia[3][2] là phần tử tại hàng 3 cột 2 trong mảng 2 chiều xem như là một biến kiểu int.

1.10. Nhập dữ liệu cho mảng 2 chiều

```
for (i = 0; i < 5; i++)     //vòng for có giá trị i chạy từ 0 đến 4 cho hàng
    for (ij = 0; ij < 10; ij++)   //vòng for có giá trị ij chạy từ 0 đến 9 cho cột
{
 printf("Nhap vao phan tu ia[%d][%d]: ", i + 1, ij + 1);
 scanf("%d", &ia[i][ij]);
}
```

* Thứ tự nhập dữ liệu vào mảng 2 chiều

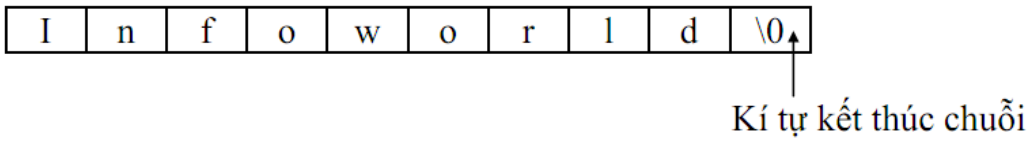


2. Chuỗi

Chuỗi được xem như là một mảng 1 chiều gồm các phần tử có kiểu char như mẫu tự, con số và bất cứ ký tự đặc biệt như +, -, *, /, $, #…

Theo quy ước, một chuỗi sẽ được kết thúc bởi ký tự null ('\0' : kí tự rỗng).

Ví dụ: chuỗi "Infoworld" được lưu trữ như sau:



2.1. Cách khai báo chuỗi

Ví dụ 8 : char cname[30];

Ý nghĩa: Khai báo chuỗi cname có chiều dài 30 kí tự. Do chuỗi kết thúc bằng kí tự null, nên khi bạn khai báo chuỗi có chiều dài 30 kí tự chỉ có thể chứa 29 kí tự.

Ví dụ 9 : Nhập vào in ra tên

Chương trình

```
/* Chuong trinh nhap va in ra ten*/
 #include <stdio.h>
#include <conio.h>
 void main(void)
{
    char cname[30];
    printf("Cho biet ten cua ban: ");
    scanf("%s", cname);
    printf("Chao ban %s\n", cname);
    getch();
}
```

Kết quả in ra màn hình

Cho biet ten cua ban: Minh

Chao ban Minh

Lưu ý: không cần sử dụng toán tử địa chỉ & trong cname trong lệnh scanf("%s", fname), vì bản thân fname đã là địa chỉ.

Dùng hàm scanf để nhập chuỗi có hạn chế như sau: Khi bạn thử lại chương trình trên với dữ liệu nhập vào là Mai Lan, nhưng khi in ra bạn chỉ nhận được Mai. Vì hàm scanf nhận vào dữ liệu đến khi gặp khoảng trắng thì kết thúc.

2.2. Hàm nhập (gets), xuất (puts)

Sử dụng hàm gets, puts phải khai báo #include <stdio.h>

- Hàm gets dùng để nhập một chuỗi kí tự từ bàn phím thông qua *stdin*

  Dạng hàm: char * gets(char *s);

  Hoạt động:

  Hàm tiến hành nhận một dãy kí tự từ stdin cho đến khi gặp kí tự '\n' (do đó nếu trong stdin đã có sẵn kí tự '\n' rồi thì hàm gets sẽ không chờ người sử dụng nhập dữ liệu vào nữa, ta nói hàm gets đã bị trôi). Kí tự '\n' sẽ loại khỏi stdin nhưng không được đặt vào chuỗi. Chuỗi nhận được sẽ tự động bổ sung thêm kí tự '\0' để đánh dấu sự kết thúc chuỗi rồi được đặt vào vùng nhớ do con trỏ s trỏ tới. Hàm trả về địa chỉ của chuỗi nhận được.

  Ví dụ để nhập từ bàn phím một chuỗi kí tự rồi lưu vào biến HoTen ta viết như sau:

  Char HoTen[25]; gets(HoTen);

- Hàm puts dùng để đưa một chuỗi kí tự ra ngoài màn hình thông qua stdout.

  Dạng hàm: int puts(const char *s);

  Hoạt động:

Hàm sẽ đưa chuỗi do con trỏ s quản lý và một kí tự '\n' lên stdout. Nếu thành công hàm sẽ trả về kí tự cuối cùng được xuất (chính là kí tự '\n'), ngược lại hàm trả về EOF.

Ví dụ câu lệnh puts("Hello");, sẽ đưa ra màn hình dòng chữ "Hello" sau đó xuống dòng. Tương tự câu lệnh printf("Hello\n");

Vi´dụ 10

Chương trình

```
/* Chuong trinh nhap va in ra ten*/
 #include <stdio.h>
#include <conio.h>
 void main(void)
{
    char cname[30];
    puts("Cho biet ten cua ban: ");
    gets(cname);
    puts("Chao ban ");
    puts(cname);
    getch();
}
```

Kết quả in ra màn hình

```
Cho biet ten cua ban:
Mai Lan
Chao ban
Mai Lan
_
```

Lưu y´: Đối với hàm puts kí tự kết thúc chuỗi null (\0) được thay thế bằng kí tự newline (\n). Hàm gets và puts chỉ có 1 đối số và không sử dụng dạng thức trong nhập liệu cũng như xuất ra màn hình.

2.3. Khơí tạo chuôĩ

Vi′dụ 11

Chương trình

```
/* Chuong trinh nhap va in ra ten*/
 #include <stdio.h>
 #include <conio.h>
 void main(void)
{
          char cname[30];
         char chao[] = "Chao ban";
          printf("Cho biet ten cua ban: ");
         gets(cname);
          printf("%s %s.\n", chao, cname);
          getch();
}
```

Kêt quả in ra màn hình

```
Cho biet ten cua ban: Mai Lan
Chao ban Mai Lan
_
```

Lưu ý: Chiều dài tối đa của chuỗi khởi tạo bằng số kí tự + 1 (kí tự null). Với chuỗi chao có chiều dài là 9.

2.4. Mảng chuôĩ

Vi′dụ 12

Chương trình

```
/* Chuong trinh nhap thang (so) và in ra thang (chu) tuong ung*/


#include <stdio.h>
```

```
#include <conio.h>
 void main(void)
{
        char cthang[12][15] = {"January", "February", "March", "April",
                               "May", "June", "July", "August", "September",
                               "October", "November", "December"};
        int ithang;
       printf("Nhap vao thang (1-12): ");
       scanf("%d", &ithang);
        printf("%s.\n", cthang[ithang-1]);
       getch();
}
```

Kết quả in ra màn hình

```
Nhap vao thang (1-12): 2
February
_
```

Bai tập hết chương

Bai 1. Viết hàm tìm số lớn nhất, nhỏ nhất trong một mảng n số nguyên.

Bai 2. Viết hàm sắp xếp tăng dần, giảm dần của một dãy số cho trước.

Bai 3. Viết hàm tách tên và họ lót từ một chuỗi cho trước.

Bai 4. Viết hàm cắt bỏ khoảng trắng thừa ở giữa, hai đầu.

Bai 5. Viết hàm chuyển đổi 1 chuỗi sang chữ thường và 1 hàm chuyển đổi sang chữ HOA.

Bai 6. Viết hàm chuyển đổi 1 chuỗi sang dạng Title Case (kí tự đầu của mỗi từ là chữ HOA, các kí tự còn lại chữ thường)

Bai 7. Viết chương trình nhập vào 1 chuỗi và in ra chuỗi đảo ngược.

Ví dụ: Nhập vào chuỗi "Lap trinh C can ban"

In ra "nab nac C hnirt paL"

Bai 8. Viết chương trình nhập vào một chuỗi ký tự rồi đếm xem trong chuỗi đó có bao nhiêu chữ 'th'.

Bai 9. Biết rằng năm 0 là năm Canh thân (năm kỵ nhau có chu kì là 3, năm hợp nhau có chu kì là 4). Hãy viết chương trình cho phép gõ vào năm dương lịch (ví dụ 1997), xuất ra năm âm lịch (Đinh sửu) và các năm kỵ và hợp.

Có 12 chi: Tý, Sửu, Dần, Mão, Thìn, Tỵ, Ngọ, Mùi, Thân, Dậu, Tuất, Hợi.

Có 10 can: Giáp, Ất, Bính, Đinh, Mậu, Kỷ, Canh, Tân, Nhâm, Quý.

Mục lục