ỦY BAN NHÂN DÂN TỈNH ĐỒNG THÁP

TRƯỜNG CAO ĐẲNG CỘNG ĐỒNG ĐỒNG THÁP

GIÁO TRÌNH

MÔN HỌC: LẬP TRÌNH CĂN BẢN

NGÀNH, NGHỀ: QUẢN TRỊ MẠNG MÁY TÍNH

TRÌNH ĐỘ: CAO ĐẲNG

(Ban hành kèm theo Quyết định số /QĐ-CĐCĐ ngày tháng năm 20…
của Hiệu trưởng trường Cao đẳng Cộng đồng Đồng Tháp)

Đồng Tháp, năm 2017

TUYÊN BỐ BẢN QUYỀN

LỜI NÓI ĐẦU

Tin học là một ngành khoa học mũi nhọn phát triển hết sức nhanh chóng trong vài chục năm lại đây và ngày càng mở rộng lĩnh vực nghiên cứu, ứng dụng trong mọi mặt của đời sống xã hội.

Ngôn ngữ lập trình là một loại công cụ giúp con người thể hiện các vấn đề của thực tế lên máy tính một cách hữu hiệu. Với sự phát triển của tin học, các ngôn ngữ lập trình cũng dần cải tiến để đáp ứng các thách thức mới của thực tế.

Khoảng cuối những năm 1960 đầu 1970 xuất hiện nhu cầu cần có các ngôn ngữ bậc cao để hỗ trợ cho những nhà tin học trong việc xây dựng các phần mềm hệ thống, hệ điều hành. Ngôn ngữ C ra đời từ đó, nó đã được phát triển tại phòng thí nghiệm Bell. Đến năm 1978, giáo trình " Ngôn ngữ lập trình C " do chính các tác giả của ngôn ngữ là Dennish Ritchie và B.W. Kernighan viết, đã được xuất bản và phổ biến rộng rãi.

C là ngôn ngữ lập trình vạn năng. Ngoài việc C được dùng để viết hệ điều hành UNIX, người ta nhanh chóng nhận ra sức mạnh của C trong việc xử lý cho các vấn đề hiện đại của tin học. C không gắn với bất kỳ một hệ điều hành hay máy nào, và mặc dầu nó đã được gọi là " ngôn ngữ lập trình hệ thống" vì nó được dùng cho việc viết hệ điều hành, nó cũng tiện lợi cho cả việc viết các chương trình xử lý số, xử lý văn bản và cơ sở dữ liệu.

Toàn bộ giáo trình gồm sáu chương chứa đựng tương đối đầy đủ các vấn đề cơ bản nhất của ngôn ngữ lập trình C, các loại ví dụ và bài tập chọn lọc cùng một số vấn đề liên quan, giúp người học có khả năng sử dụng thành thạo ngôn ngữ này trong việc giải quyết một số lớp bài toán thông dụng trong thực tế.

Giáo trình được biên soạn cho đối tượng chính là học sinh THCN, kỹ thuật viên tin học, tuy nhiên nó cũng có thể là tài liệu tham khảo bổ ích cho bậc đại học và những người quan tâm.

mặc dù đã cố gắng nhiều trong quá trình biên soạn giáo trình này, nhưng chắc chắn không tránh khỏi có những thiếu sót. Rất mong nhận được ý kiến đóng góp của độc giả và các đồng nghiệp để giáo trình ngày càng hoàn thiện hơn.

Và bây giờ chúng ta đi tìm hiểu thế giới của ngôn ngữ C từ những khái niệm ban đầu cơ bản nhất.

MỤC LỤC

TRANG

MÔ ĐUN: LẬP TRÌNH CĂN BẢN

Mã mô đun: MH10

Vị trí, ý nghĩa, vai trò của mô đun:

- Vị trí:

+ Mô đun được bố trí sau khi học sinh học xong các môn học chung, trước các môn học/ mô đun đào tạo chuyên môn nghề.

- Tính chất:

+ Là mô đun cơ sở.

- Ý nghĩa và vai trò của mô đun:

+ Là mô đun không thể thiếu của nghề SCLR máy tính

+ Là mô đun tư duy logic về lập trình

Mục tiêu của mô đun:

- Hiểu được công dụng của ngôn ngữ lập trình, hiểu cú pháp, công dụng của các câu lệnh dùng trong ngôn ngữ lập trình.
- Phân tích được chương trình: xác định nhiệm vụ chương trình (phải làm gì).
- Vận dụng điều kiện, trợ giúp môi trường của ngôn ngữ lập trình, chẳng hạn: các thao tác biên tập chương trình, các công cụ, điều khiển, thực đơn lệnh trợ giúp, gỡ rối, bẫy lỗi, v. v.
- Viết chương trình và thực hiện chương trình trong máy tính.
- Tự tin khi tiếp cận các mã (code) chương trình.
- Loại bỏ tâm lý lo sợ khi gặp những công việc được lập trình hóa.

Nội dung của mô đun:

Mã bài	Tên các bài trong mô đun	Thời lượng			
		Tổng số	Lý thuyết	Thực hành	Kiểm tra
MH13-01	Bài 1: Tổng quan về ngôn ngữ lập trình:	02	02		
MH13-02	Bài 2: Các thành phần cơ bản	08	02	06	
MH13-03	Bài 3: Các lệnh cấu trúc	16	04	10	02
MH13-04	Bài 4: Hàm	20	08	10	02
MH13-05	Bài 5: Mảng	13	04	08	01
MH13-06	Bài 6: Chuỗi ký tự	13	04	08	01
MH13-07	Bài 7: Mảng và biến con trỏ	18	04	12	02

BÀI 1
TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH C

Mã bài: MH 10-01

Giới thiệu:

Bài này nhằm cung cấp cho người học các kiến thức về lịch sử phát triển của ngôn ngữ, ứng dụng thực tế của ngôn ngữ, các cách khởi động và thoát khỏi chương trình.

Mục tiêu

- Biết được ngôn ngữ này có những ứng dụng thực tế như thế nào;

- Biết sử dụng được hệ thống trợ giúp của ngôn ngữ lập trình;

1.Giới thiệu lịch sử phát triển của ngôn ngữ lập trình C

Mục tiêu:

- Biết được lịch sử phát triển của ngôn ngữ lập trình;

Vào đầu những năm 70 tại phòng thí nghiệm Bell, Dennis Ritchie đã phát triển ngôn ngữ C. C được sử dụng lần đầu trên một hệ thống cài đặt hệ điều hành UNIX. C có nguồn gốc từ ngôn ngữ BCPL do Martin Richards phát triển. BCPL sau đó đã được Ken Thompson phát triển thành ngôn ngữ B, đây là người khởi thủy ra C.

Trong khi BCPL và B không hỗ trợ kiểu dữ liệu, thì C đã có nhiều kiểu dữ liệu khác nhau. Những kiểu dữ liệu chính gồm : kiểu ký tự (character), kiểu số nguyên (interger) và kiểu số thực (float).

C liên kết chặt chẽ với hệ thống UNIX nhưng không bị trói buộc vào bất cứ một máy tính hay hệ điều hành nào. C rất hiệu quả để viết các chương trình thuộc nhiều những lĩnh vực khác nhau.

C cũng được dùng để lập trình hệ thống. Một chương trình hệ thống có ý nghĩa liên quan đến hệ điều hành của máy tính hay những tiện ích hỗ trợ nó. Hệ điều hành (OS), trình thông dịch (Interpreters), trình soạn thảo (Editors), chương trình Hợp Ngữ (Assembly) là các chương trình hệ thống. Hệ điều hành UNIX được phát triển dựa vào C. C đang được sử dụng rộng rãi bởi vì tính hiệu quả và linh hoạt. Trình biên dịch (compiler) C có sẵn cho hầu hết các máy tính. Mã lệnh viết bằng C trên máy này có thể được biên dịch và chạy trên máy khác chỉ cần thay đổi rất ít hoặc không thay đổi gì cả. Trình biên dịch C dịch nhanh và cho ra mã đối tượng không lỗi.

C khi thực thi cũng rất nhanh như hợp ngữ (Assembly). Lập trình viên có thể tạo ra và bảo trì thư viện hàm mà chúng sẽ được tái sử dụng cho chương trình khác. Do đó, những dự án lớn có thể được quản lý dễ dàng mà tốn rất ít công sức.

2. Cách khởi động và thoát chương trình

Mục tiêu:

- Biết cách khởi động được và thoát khỏi chương trình C;

2.1. Khởi Động Chương Trình C

Nhập lệnh tại dấu nhắc DOS: gõ BC ↵ (Enter) (nếu đường dẫn đã được cài đặt bằng lệnh path trong đó có chứa đường dẫn đến thư mục chứa tập tin BC.EXE). Nếu đường dẫn chưa được cài đặt ta tìm xem thư mục BORLANDC nằm ở ổ đĩa nào. Sau đó ta gõ lệnh sau: <ổ đĩa>:\BORLANDC\BIN\BC ↵ (Enter)

Nếu bạn muốn vừa khởi động BC vừa soạn thảo chương trình với một tập tin có tên do chúng ta đặt, thì gõ lệnh: BC [đường dẫn]<tên file cần soạn thảo>, nếu tên file cần soạn thảo đã có thì được nạp lên, nếu chưa có sẽ được tạo mới.

Khởi động tại Windows: Bạn vào menu Start, chọn Run, bạn gõ vào hộp Open 1 trong
các dòng lệnh như nhập tại DOS. Hoặc bạn vào Window Explorer, chọn ổ đĩa chứa thư mục BORLANDC, vào thư mục BORLANDC, vào thư mục BIN, khởi động tập tin BC.EXE.

Ví dụ: Bạn gõ D:\BORLANDC\BIN\BC E:\BAITAP_BC\VIDU1.CPP

Câu lệnh trên có nghĩa khởi động BC và nạp tập tin VIDU1.CPP chứa trong thư mục BAITAP_BC trong ổ đĩa E. Nếu tập tin này không có sẽ được tạo mới.

2.2. Thoát khỏi chương trình C

Ấn phím F10 (kích hoạt Menu), chọn menu File, chọn Quit;

Hoặc ấn tổ hợp phím Alt – X.

3.Hệ thống thông tin giúp đỡ

Mục tiêu:

- *Trình bày được hệ thống trợ giúp của C*
- Ấn phím F1 để kích hoạt màn hình Help chính.
- Muốn xem Help của hàm trong soạn thảo, di chuyển con trỏ đến vị trí hàm đó ấn tổ hợp
phím Ctrl - F1
- Ấn tổ hợp phím Shift - F1 để xem danh sách các mục Help
- Ấn tổ hợp phím Alt - F1 để quay về màn hình Help trước đó.

BÀI 2
CÁC THÀNH PHẦN CƠ BẢN

Mã bài: MH 10-02

Giới thiệu

Cũng như những ngôn ngữ lập trình khác, ngôn ngữ lập trình c cũng có hệ thống ký hiệu và từ khóa, các kiểu dữ liệu. Bài học này sẽ cho người học các kiến thức về các loại hằng, biến, các lệnh, khối lệnh và thực hiện việc chạy chương trình.

Mục tiêu:

- Hiểu và vận dụng được các lệnh cấu trúc**:** cấu trúc lựa chọn, cấu trúc lặp xác định và lặp vô định.
- Hiểu và vận dụng được các lệnh bẻ vòng lặp
- Rèn luyện thói quen suy luận logic.

1. Hệ thống ký hiệu và từ khóa

Mục tiêu:

- Trình bày được tập kí tự dùng trong ngôn ngữ C

- Trình bày được các quy tắc cần nhớ khi viết chương trình

- Tập ký tự dùng trong ngôn ngữ C

Mọi ngôn ngữ lập trình đều được xây dựng từ một bộ ký tự nào đó. Các ký tự được nhóm lại theo nhiều cách khác nhau để tạo nên các từ. Các từ lại được liên kết với nhau theo một qui tắc nào đó để tạo nên các câu lệnh. Một chương trình bao gồm nhiều câu lệnh và thể hiện một thuật toán để giải một bài toán nào đó. Ngôn ngữ C được xây dựng trên bộ ký tự sau :

26 chữ cái hoa : A B C .. Z
26 chữ cái thường : a b c .. z
10 chữ số : 0 1 2 .. 9
Các ký hiệu toán học : + - * / = ( )
Ký tự gạch nối : _
Các ký tự khác : . , : ; [ ] {} ! \ & % # $ ...

Dấu cách (space) dùng để tách các từ. Ví dụ chữ VIET NAM có 8 ký tự, còn VIETNAM chỉ có 7 ký tự.

Chú ý :

Khi viết chương trình, ta không được sử dụng bất kỳ ký tự nào khác ngoài các ký tự trên.

Ví dụ như khi lập chương trình giải phương trình bậc hai $ax^2+bx+c=0$, ta cần tính biệt thức Delta $\Delta= b^2 - 4ac$, trong ngôn ngữ C không cho phép dùng ký tự Δ, vì vậy ta phải dùng ký hiệu khác để thay thế.

- Từ khóa

Từ khoá là những từ được sử dụng để khai báo các kiểu dữ liệu, để viết các toán tử và các câu lệnh. Bảng dưới đây liệt kê các từ khoá của TURBO C :

1.	asm	break	case	cdecl
2.	char	const	continue	default
3.	do	double	else	enum
4.	extern	far	float	for
5.	goto	huge	if	int
6.	interrupt	long	near	pascal
7.	register	return	short	signed
8.	sizeof	static	struct	switch
9.	tipedef	union	unsigned	void
10.	volatile	while		

Ý nghĩa và cách sử dụng của mỗi từ khoá sẽ được đề cập sau này, ở đây ta cần chú ý :

- Không được dùng các từ khoá để đặt tên cho các hằng, biến, mảng, hàm ...

- Từ khoá phải được viết bằng chữ thường, ví dụ : viết từ khoá khai báo kiểu nguyên là int chứ không phải là INT.

- Tên

Tên là một khái niệm rất quan trọng, nó dùng để xác định các đại lượng khác nhau trong một chương trình. Chúng ta có tên hằng, tên biến, tên mảng, tên hàm, tên con trỏ, tên tệp, tên cấu trúc, tên nhãn,...

Tên được đặt theo qui tắc sau :

Tên là một dãy các ký tự bao gồm chữ cái, số và gạch nối. Ký tự đầu tiên của tên phải là chữ hoặc gạch nối. Tên không được trùng với khoá. Độ dài cực đại của tên theo mặc định là 32 và có thể được đặt lại là một trong các giá trị từ 1 tới 32 nhờ chức năng : Option-Compiler-Source-Identifier length khi dùng TURBO C.

Ví dụ :

Các tên đúng :

a_1 delta x1 _step GAMA

Các tên sai :

3MN	Ký tự đầu tiên là số
m#2	Sử dụng ký tự #
f(x)	Sử dụng các dấu ( )
do	Trùng với từ khoá
te ta	Sử dụng dấu trắng
Y-3	Sử dụng dấu -

Chú ý :

Trong TURBO C, tên bằng chữ thường và chữ hoa là khác nhau ví dụ tên AB khác với ab. trong C, ta thường dùng chữ hoa để đặt tên cho các hằng và dùng chữ thường để đặt tên cho hầu hết cho các đại lượng khác như biến, biến mảng, hàm, cấu trúc. Tuy nhiên đây không phải là điều bắt buộc.

- Một số quy tắc cần nhớ khi viết chương trình

Qui tắc đầu tiên cần nhớ là :

Mỗi câu lệnh có thể viết trên một hay nhiều dòng nhưng phải kết thúc bằng dấu;

Qui tắc thứ hai là :

C*ác lời giải thích cần được đặt giữa các dấu /* và */ và có thể được viết*

Trên một dòng

Trên nhiều dòng

Trên phần còn lại của dòng

Qui tắc thứ ba là :

Trong chương trình, khi ta sử dụng các hàm chuẩn, ví dụ như printf(), getch(),... mà các hàm này lại chứa trong file stdio.h trong thư mục của C, vì vậy ở đầu chương trình ta phải khai báo sử dụng ;

#include "stdio.h "

Qui tắc thứ tư là :

Một chương trình có thể chỉ có một hàm chính (hàm main()) hoặc có thể có thêm vài hàm khác.

2. Các kiểu dữ liệu: kiểu số, chuỗi, ký tự

Mục tiêu:

- Trình bày được các kiểu dữ liệu.

- Kiểu ký tự

Một giá trị kiểu char chiếm 1 byte (8 bit) và biểu diễn được một ký tự thông qua bảng mã ASCII. Ví dụ:

Ký tự	Mã ASCII
0	048
1	049
2	050
A	065
B	066
A	097
B	098

Có hai kiểu dữ liệu char : kiểu signed char và unsigned char.

	Kiểu	Phạm vi biểu diễn	Số ký tự	Kích thước	
	Char (Signed char)	-128 đến 127	256	1 byte	
	Unsigned char	0 đến 255	256	1 byte	

Ví dụ sau minh hoạ sự khác nhau giữa hai kiểu dữ liệu trên : Xét đoạn chương trình sau :

```
char  ch1;
unsigned char ch2;
.....
ch1=200; ch2=200;
```

Khi đó thực chất :

```
ch1=-56;
ch2=200;
```

Nhưng cả ch1 và ch2 đều biểu diễn cùng một ký tự có mã 200.

Phân loại ký tự :

Có thể chia 256 ký tự làm ba nhóm :

Nhóm 1: Nhóm các ký tự điều khiển có mã từ 0 đến 31. Chẳng hạn ký tự mã 13 dùng để chuyển con trỏ về đầu dòng, ký tự 10 chuyển con trỏ xuống dòng dưới (trên cùng một cột). Các ký tự nhóm này nói chung không hiển thị ra màn hình.

Nhóm 2 : Nhóm các ký tự văn bản có mã từ 32 đến 126. Các ký tự này có thể được đưa ra màn hình hoặc máy in.

Nhóm 3 : Nhóm các ký tự đồ hoạ có mã số từ 127 đến 255. Các ký tự này có thể đưa ra màn hình nhưng không in ra được (bằng các lệnh DOS).

- Kiểu số nguyên

Trong C cho phép sử dụng số nguyên kiểu int, số nguyên dài kiểu long và số nguyên không dấu kiểu unsigned. Kích cỡ và phạm vi biểu diễn của chúng được chỉ ra trong bảng dưới đây :

	Kiểu	Phạm vi biểu diễn	Kích thước
1	int	-32768 đến 32767	2 byte

2	unsigned int	0 đến 65535	2 byte
3	long	-2147483648 đến 2147483647	4 byte
4	unsigned long	0 đến 4294967295	4 byte

Chú ý :

Kiểu ký tự cũng có thể xem là một dạng của kiểu nguyên.

- Kiểu số phẩy động độ chính xác đơn (float), kép (double)

Trong C cho phép sử dụng ba loại dữ liệu dấu phảy động, đó là float, double và long double. Kích cỡ và phạm vi biểu diễn của chúng được chỉ ra trong bảng dưới đây :

Kiểu	Phạm vi biểu diễn	Số chữ số có nghĩa	Kích thước
Float	3.4E-38 đến 3.4E+38	7 đến 8	4 byte
Double	1.7E-308 đến 1.7E+308	15 đến 16	8 byte
long double	3.4E-4932 đến 1.1E4932	17 đến 18	10 byte

Giải thích :

Máy tính có thể lưu trữ được các số kiểu float có giá trị tuyệt đối từ 3.4E-38 đến 3.4E+38. Các số có giá trị tuyệt đối nhỏ hơn3.4E-38 được xem bằng 0. Phạm vi biểu diễn của số double được hiểu theo nghĩa tương tự.

3. Biến, hằng, biểu thức

3.1. Hằng

Hằng là các đại lượng mà giá trị của nó không thay đổi trong quá trình tính toán.

- Tên hằng :

Nguyên tắc đặt tên hằng ta đã xem xét trong mục 1.3.

Để đặt tên một hằng, ta dùng dòng lệnh sau :

#define tên hằng giá trị

Ví dụ :

```
#define MAX 1000
```

Lúc này, tất cả các tên MAX trong chương trình xuất hiện sau này đều được thay bằng 1000. Vì vậy, ta thường gọi MAX là tên hằng, nó biểu diễn số 1000.

Một ví dụ khác : #define pi 3.141593 Đặt tên cho một hằng float là pi có giá trị là 3.141593.

- Các loại hằng:

+ Hằng int:

Hằng int là số nguyên có giá trị trong khoảng từ -32768 đến 32767.

Chú ý :

Cần phân biệt hai hằng 5056 và 5056.0 : ở đây 5056 là số nguyên còn 5056.0 là hằng thực.

+ Hằng long :

Hằng long là số nguyên có giá trị trong khoảng từ -2147483648 đến 2147483647.
Hằng long được viết theo cách :
1234L hoặc 1234l
(thêm L hoặc l vào đuôi)
Một số nguyên vượt ra ngoài miền xác định của int cũng được xem là long.
Ví dụ :

#define sl 8865056L	Định nghiã hằng long sl có giá trị là 8865056
#define sl 8865056	Định nghiã hằng long sl có giá trị là 8865056

+ Hằng int hệ 8 :
Hằng int hệ 8 được viết theo cách 0c1c2c3....Ở đây ci là một số nguyên dương trong khoảng từ 1 đến 7. Hằng int hệ 8 luôn luôn nhận giá trị dương.

Ví dụ :

#define h8 0345	Định nghiã hằng int hệ 8 có giá trị là 3*8*8+4*8+5=229

+ Hằng int hệ 16 :
Trong hệ này ta sử dụng 16 ký tự : 0,1..,9,A,B,C,D,E,F.

Cách viết	Giá trị
a hoặc A	10
b hoặc B	11
c hoặc C	12
d hoặc D	13
e hoặc E	14
f hoặc F	15

Hằng số hệ 16 có dạng 0xc1c2c3... hoặc 0Xc1c2c3... Ở đây ci là một số trong hệ 16.

Ví dụ :
#define h16 0xa5
#define h16 0xA5
#define h16 0Xa5
#define h16 0XA5
Cho ta các hắng số h16 trong hệ 16 có giá trị như nhau. Giá trị của chúng trong hệ 10 là :
10*16+5=165.
+ Hằng ký tự :

Hằng ký tự là một ký tự riêng biệt được viết trong hai dấu nháy đơn, ví dụ 'a'.

Giá trị của 'a' chính là mã ASCII của chữ a. Như vậy giá trị của 'a' là 97. Hằng ký tự có thể tham gia vào các phép toán như mọi số nguyên khác. Ví dụ :

'9'-'0'=57-48=9

Ví dụ :

#define kt 'a' Định nghiã hằng ký tự kt có giá trị là 97

Hằng ký tự còn có thể được viết theo cách sau :

' \c1c2c3'

Trong đó c1c2c3 là một số hệ 8 mà giá trị của nó bằng mã ASCII của ký tự cần biểu diễn.

Ví dụ : chữ a có mã hệ 10 là 97, đổi ra hệ 8 là 0141. Vậy hằng ký tự 'a' có thể viết dưới dạng '\141'. Đối với một vài hằng ký tự đặc biệt ta cần sử dụng cách viết sau (thêm dấu\ ):

Cách viết	Ký tự
'\''	'
'\"'	"
'\\'	\
'\n'	\n (chuyển dòng)
'\0'	\0 (null)
'\t'	Tab
'\b'	Backspace
'\r'	CR (về đầu dòng)
'\f'	LF (sang trang)

Chú ý :

Cần phân biệt hằng ký tự '0' và '\0'. Hằng '0' ứng với chữ số 0 có mã ASCII là 48, còn hằng '\0' ứng với kýtự \0 (thường gọi là ký tự null) có mã ASCII là 0.

Hằng ký tự thực sự là một số nguyên, vì vậy có thể dùng các số nguyên hệ 10 để biểu diễn các ký tự, ví dụ lệnh printf("%c%c",65,66) sẽ in ra AB.

+ Hằng xâu ký tự :

Hằng xâu ký tự là một dãy ký tự bất kỳ đặt trong hai dấu nháy kép.

Ví dụ :

```
#define xau1 "Ha noi"
#define xau2 "My name is Giang"
```

Xâu ký tự được lưu trữ trong máy dưới dạng một bảng có các phần tử là các ký tự riêng biệt. Trình biên dịch tự động thêm ký tự null \0 vào cuối mỗi xâu (ký tự \0 được xem là dấu hiệu kết thúc của một xâu ký tự).

Chú ý :

Cần phân biệt hai hằng 'a' và "a". 'a' là hằng ký tự được lưu trữ trong 1 byte, còn "a" là hằng xâu ký tự được lưu trữ trong 1 mảng hai phần tử : phần tử thứ nhất chứa chữ a còn phần tử thứ hai chứa \0.

3.2. Biến

Mỗi biến cần phải được khai báo trước khi đưa vào sử dụng. Việc khai báo biến được thực hiện theo mẫu sau :

Kiểu dữ liệu của biến tên biến ;

Ví dụ :

int a,b,c;	Khai báo ba biến int là a,b,c
long dai,mn;	Khai báo hai biến long là dai và mn
char kt1,kt2;	Khai báo hai biến ký tự là kt1 và kt2
float x,y	Khai báo hai biến float là x và y
doublecanh1,canh 2;	Khai báo hai biến double là canh1 và canh2

Biến kiểu int chỉ nhận được các giá trị kiểu int. Các biến khác cũng có ý nghĩa tương tự. Các biến kiểu char chỉ chứa được một ký tự. Để lưu trữ được một xâu ký tự cần sử dụng một mảng kiểu char.

Vị trí của khai báo biến :

Các khai báo cần phải được đặt ngay sau dấu { đầu tiên của thân hàm và cần đứng trước mọi câu lệnh khác. Sau đây là một ví dụ về khai báo biến sai :
(Khái niệm về hàm và cấu trúc chương trình sẽ nghiên cứu sau này)

```
main()
    {
        int a,b,c;
        a=2;
        int d; /* Vị trí của khai báo sai */
        .....
    }
```

Khởi đầu cho biến :

Nếu trong khai báo ngay sau tên biến ta đặt dấu = và một giá trị nào đó thì đây chính là cách vừa khai báo vừa khởi đầu cho biến.

Ví dụ :

```
        int a,b=20,c,d=40;
        float e=-55.2,x=27.23,y,z,t=18.98;
```

Việc khởi đầu và việc khai báo biến rồi gán giá trị cho nó sau này là hoàn toàn tương đương.

Lấy địa chỉ của biến :

Mỗi biến được cấp phát một vùng nhớ gồm một số byte liên tiếp. Số hiệu của byte đầu chính là địa chỉ của biến. Địa chỉ của biến sẽ được sử dụng trong một số hàm ta sẽ nghiên cứu sau này (ví dụ như hàm scanf).

Để lấy địa chỉ của một biến ta sử dụng phép toán :

& tên biến

3.3. Biểu thức :

Biểu thức là một sự kết hợp giữa các phép toán và các toán hạng để diễn đạt một công thức toán học nào đó. Mỗi biểu thức có sẽ có một giá trị. Như vậy hằng, biến, phần tử mảng và hàm cũng được xem là biểu thức.

Trong C, ta có hai khái niệm về biểu thức :

Biểu thức gán.

Biểu thức điều kiện .

Biểu thức được phân loại theo kiểu giá trị : nguyên và thực. Trong các mệnh đề logic, biểu thức được phân thành đúng (giá trị khác 0) và sai (giá trị bằng 0).

Biểu thức thường được dùng trong :

Vế phải của câu lệnh gán.

Làm tham số thực sự của hàm.

Làm chỉ số.

Trong các toán tử của các cấu trúc điều khiển.

Tới đây, ta đã có hai khái niệm chính tạo nên biểu thức đó là toán hạng và phép toán. Toán hạng gồm: hằng, biến, phần tử mảng và hàm trước đây ta đã xét. Dưới đây ta sẽ nói đến các phép toán.

4. Lệnh và khối lệnh

Mục tiêu:

- Phân biệt được lệnh và khối lệnh;

- Trình bày được phạm vi hoạt động của biến và mảng;

4.1. Lệnh:

Một biểu thức kiểu như x=0 hoặc ++i hoặc scanf(...) trở thành câu lệnh khi có đi kèm theo dấu ;

Ví dụ :

```
x=0;
++i;
scanf(...);
```

Trong chương trình C, dấu ; là dấu hiệu kết thúc câu lệnh.

4.2. Khối lệnh :

Một dãy các câu lệnh được bao bởi các dấu { } gọi là một khối lệnh. Ví dụ :

```
{
    a=2;
    b=3;
    printf("\n%6d%6d",a,b);
}
```

TURBO C xem khối lệnh cũng như một câu lệnh riêng lẻ. Nói cách khác, chỗ nào viết được một câu lệnh thì ở đó cũng có quyền đặt một khối lệnh.

Khai báo ở đầu khối lệnh :

Các khai báo biến và mảng chẳng những có thể đặt ở đầu của một hàm mà còn có thể viết ở đầu khối lệnh :

```
{
    int a,b,c[50];
    float x,y,z,t[20][30];
    a==b==3;
    x=5.5; y=a*x;
    z=b*x;
    printf("\n y= %8.2f\n z=%8.2f",y,z);
}
```

Sự lồng nhau của các khối lệnh và phạm vi hoạt động của các biến và mảng:

Bên trong một khối lệnh lại có thể viết lồng khối lệnh khác. Sự lồng nhau theo cách như vậy là không hạn chế.

Khi máy bắt đầu làm việc với một khối lệnh thì các biến và mảng khai báo bên trong nó mới được hình thành và được hình thành và được cấp phát bộ nhớ. Các biến này chỉ tồn tại trong thời gian máy làm việc bên trong khối lệnh và chúng lập tức biến mất ngay sau khi máy ra khỏi khối lệnh. Vậy :

Giá trị của một biến hay một mảng khai báo bên trong một khối lệnh không thể đưa ra sử dụng ở bất kỳ chỗ nào bên ngoài khối lệnh đó.

ở bất kỳ chỗ nào bên ngoài một khối lệnh ta không thể can thiệp đến các biến và các mảng được khai báo bên trong khối lệnh

Nếu bên trong một khối ta dùng một biến hay một mảng có tên là a thì điều này không làm thay đổi giá trị của một biến khác cũng có tên là a (nếu có) được dùng ở đâu đó bên ngoài khối lệnh này.

Nếu có một biến đã được khai báo ở ngoài một khối lệnh và không trùng tên với các biến khai báo bên trong khối lệnh này thì biến đó cũng có thể sử dụng cả bên trong cũng như bên ngoài khối lệnh.

Ví dụ :

Xét đoạn chương trình sau :

```
 {
        int a=5,b=2;
          {
                int a=4;
                b=a+b;
                printf("\n a trong =%3d  b=%3d",a,b);
          }
        printf("\n a ngoai =%3d  b=%3d",a,b);
 }
```

Khi đó đoạn chương trình sẽ in kết quả như sau :

a trong =4 b=6

a ngoài =5 b=6

Do tính chất biến a trong và ngoài khối lệnh.

5. Lệnh gán, lệnh xuất nhập, lệnh gán kết hợp

Mục tiêu:

- Trình bày được cú pháp của các lệnh gán, lệnh xuất nhập;

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

Cú pháp: <Tên biến> = <biểu thức>

Ví dụ:

```
int main() {
        int x,y;
        x =10; /*Gán hằng số 10 cho biến x*/
        y = 2*x; /*Gán giá trị 2*x=2*10=20 cho x*/
        return 0;
```

```
}
```

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

```
int main() {
        int x,y;
        x = 10; /*Gán hằng số 10 cho biến x*/
        y = “Xin chao”;
        /*y có kiểu int, còn “Xin chao” có kiểu char* */
        return 0;
    }
```

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

Tuy nhiên trong đa số trường hợp sự tự động biến đổi kiểu để sự tương thích về kiểu sẽ được thực hiện.

Ví dụ:

```
int main() {
        int x,y;
        float r;
        char ch;
        r = 9000;
        x = 10; /* Gán hằng số 10 cho biến x */
        y = 'd'; /* y có kiểu int, còn ‘d’ có kiểu char*/
        r = 'e'; /* r có kiểu float, ‘e’ có kiểu char*/
        ch = 65.7; /* ch có kiểu char, còn 65.7 có kiểu float*/
        return 0;
    }
```

Trong nhiều trường hợp để tạo ra sự tương thích về kiểu, ta phải sử dụng đến cách thức chuyển đổi kiểu một cách tường minh. Cú pháp của phép toán này như sau:

(Tên kiểu) <Biểu thức>

Chuyển đổi kiểu của <Biểu thức> thành kiểu mới <Tên kiểu>. Chẳng hạn như:

```
float f;
f = (float) 10 / 4; /* f lúc này là 2.5*/
```

Chú ý:

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

Ví dụ:

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

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

<Tên kiểu> <Tên biến> = <Biểu thức>;

Ví dụ: int x = 10, y=x;

2.5.2. Lệnh xuất,nhập

a. Lệnh nhập

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

Cú pháp:

scanf("Chuỗi định dạng", địa chỉ của các biến);

Giải thích:

- Chuỗi định dạng: dùng để qui định kiểu dữ liệu, cách biểu diễn, độ rộng, số chữ số thập phân... Một số định dạng khi nhập kiểu số nguyên, số thực, ký tự.

Định dạng	Ý nghĩa
%[số ký số]d	Nhập số nguyên có tối đa <số ký số>
%[số ký số] f	Nhập số thực có tối đa <số ký số> tính cả dấu chấm
%c	Nhập một ký tự
Ví dụ:	
%d	Nhập số nguyên
%4d	Nhập số nguyên tối đa 4 ký số, nếu nhập nhiều hơn 4 ký số thì chỉ nhận được 4 ký số đầu tiên
%f	Nhập số thực
%6f	Nhập số thực tối đa 6 ký số (tính luôn dấu chấm), nếu nhập nhiều hơn 6 ký số thì chỉ nhận được 6 ký số đầu tiên (hoặc 5 ký số với dấu chấm)

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

Ví dụ:

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

Lưu ý:

- Chuỗi định dạng phải đặt trong cặp dấu nháy kép ("").
- Các biến (địa chỉ biến) phải cách nhau bởi dấu phẩy (,).
- Có bao nhiêu biến thì phải có bấy nhiêu định dạng.
- Thứ tự của các định dạng phải phù hợp với thứ tự của các biến.
- Để nhập giá trị kiểu char được chính xác, nên dùng hàm fflush(stdin) để loại bỏ các ký tự còn nằm trong vùng đệm bàn phím trước hàm scanf().

- Để nhập vào một chuỗi ký tự (không chứa khoảng trắng hay kết thúc bằng khoảng trắng), chúng ta phải khai báo kiểu mảng ký tự hay con trỏ ký tự, sử dụng định dạng %s và tên biến thay cho địa chỉ biến.
- Để đọc vào một chuỗi ký tự có chứa khoảng trắng (kết thúc bằng phím Enter) thì phải dùng hàm gets().

Ví dụ:

int biennguyen;
float bienthuc;
char bienchar;
char chuoi1[20], *chuoi2;
Nhập giá trị cho các biến:
scanf("%3d",&biennguyen);

Nếu ta nhập 1234455 thì giá trị của biennguyen là 3 ký số đầu tiên (123). Các ký số còn lại sẽ còn nằm lại trong vùng đệm.

scanf("%5f",&bienthuc);

Nếu ta nhập 123.446 thì giá trị của bienthuc là 123.4, các ký số còn lại sẽ còn nằm trong vùng đệm.

scanf("%2d%5f",&biennguyen, &bienthuc);

Nếu ta nhập liên tiếp 2 số cách nhau bởi khoảng trắng như sau: 1223 3.142325

- 2 ký số đầu tiên (12) sẽ được đọc vào cho biennguyen.
- 2 ký số tiếp theo trước khoảng trắng (23) sẽ được đọc vào cho bienthuc.

scanf("%2d%5f%c",&biennguyen, &bienthuc,&bienchar)

Nếu ta nhập liên tiếp 2 số cách nhau bởi khoảng trắng như sau: 12345 3.142325:

- 2 ký số đầu tiên (12) sẽ được đọc vào cho biennguyen.
- 3 ký số tiếp theo trước khoảng trắng (345) sẽ được đọc vào cho bienthuc.
- Khoảng trắng sẽ được đọc cho bienchar.

Nếu ta chỉ nhập 1 số gồm nhiều ký số như sau: 123456789:

- 2 ký số đầu tiên (12) sẽ được đọc vào cho biennguyen.
- 5 ký số tiếp theo (34567) sẽ được đọc vào cho bienthuc.
- bienchar sẽ có giá trị là ký số tiếp theo '8'.

scanf("%s",chuoi1); hoặc scanf("%s",chuoi2)

thì↵ Nếu ta nhập chuỗi như sau: Nguyen Van Linh giá trị của biến chuoi1 hay chuoi2 chỉ là Nguyen .

scanf("%s%s",chuoi1, chuoi2);

thì giá↵Nếu ta nhập chuỗi như sau: Duong Van Hieu trị của biến chuoi1 là Duong và giá trị của biến chuoi2 là Van.

Vì sao như vậy? C sẽ đọc từ đầu đến khi gặp khoảng trắng và gán giá trị cho biến đầu tiên, phần còn lại sau khoảng trắng là giá trị của các biến tiếp theo.

gets(chuoi1);

Nếu nhập chuỗi : Nguyen Van Linh thì giá trị của biến chuoi1 là Nguyen Van Linh