UỶ BAN NHÂN DÂN TỈNH HẢI PHÒNG
TRƯỜNG CĐCN HẢI PHÒNG

GIÁO TRÌNH

Tên môn học: Lập trình C
NGHỀ: QUẢN TRỊ MẠNG

TRÌNH ĐỘ CAO ĐẲNG

HẢI PHÒNG

TUYÊN BỐ BẢN QUYỀN

LỜI GIỚI THIỆU

”Lập trình cơ bản” là môn học bắt buộc trong các trường nghề. Tuỳ thuộc vào đối tượng người học và cấp bậc học mà trang bị cho học sinh, sinh viên những kiến thức cơ bản nhất.

Để thống nhất chương trình và nội dung giảng dạy trong các nhà trường chúng tôi biên soạn cuốn giáo trình: Lập trình cơ bản. Giáo trình được biên soạn phù hợp với các nghề trong các trường đào tạo nghề phục vụ theo yêu cầu của thực tế xã hội hiện nay.

Tài liệu tham khảo để biên soạn gồm:

[1]. Phạm Văn Ất. *Kỹ thuật lập trình C cơ sở và nâng cao*. Nhà xuất bản Giao thông vận tải,2006

[2]. Ngô Trung Việt. *Ngôn ngữ lập trình C và C++*. Nhà xuất bản Thống kê,

Kết hợp với kiến thức mới có liên quan môn học và những vấn đề thực tế thường gặp trong sản xuất, đời sống để giáo trình có tính thực tế cao, giúp cho người học dễ hiểu, dễ dàng lĩnh hội được kiến thức môn học.

Trong quá trình biên soạn giáo trình kinh nghiệm còn hạn chế, chúng tôi rất mong nhận được ý kiến đóng góp của bạn đọc để lần hiệu đính sau được hoàn chỉnh hơn.

Tổ bộ môn ứng dụng tin học

MỤC LỤC

LẬP TRÌNH CƠ BẢN

I. Vị trí, tính chất của môn học:

- Vị trí: Là môn học được bố trí sau khi sinh viên học xong các môn chung, các môn học cơ sở chuyên ngành đào tạo chuyên môn nghề;
- Tính chất: Là môn học kỹ thuật cơ sở chương trình đào tạo nghề công nghệ thông tin (ƯDPM).

II. Mục tiêu môn học:

Kiến thức: Trình bày được các khái niệm cơ bản về lập trình, một số cấu trúc, câu lệnh trong lập trình C.

Kỹ năng: Sử dụng được môi trường lập trình C để soạn thảo, biên dịch và hiệu chỉnh chương trình; Hình thành kỹ thuật lập trình từ việc áp dụng các khái niệm cơ bản để giải quyết các bài toán.

Về năng lực tự chủ và trách nhiệm: Rèn luyện tư duy lập trình đối với sinh viên hệ cao đẳng.

III. NỘI DUNG

Chương 1: Tổng quan về ngôn ngữ lập trình c

1.1. Giới thiệu sơ lược về ngôn ngữ lập trình C
1.2. Các phần tử cơ bản của C
1.3. Cấu trúc cơ bản của chương trình C
1.4. Cài đặt và sử dụng phần mềm

Chương 2: Khởi đầu với lập trình c

2.1. Một số khai báo cơ bản
2.2. Các phép toán trong C
2.3. Các lệnh vào ra dữ liệu

TH bài 1: Cài đặt, sử dụng môi trường lập trình C

Chương 3: Các lệnh cấu trúc

3.1. Lệnh khối
3.2. Lệnh lựa chọn
3.3. Lệnh lặp
TH Bài 2: Lập trình với lệnh khối và lệnh lựa chọn
TH Bài 3: Lập trình với lệnh lặp

Chương 4: Hàm và thư viện

4.1. Khái niệm về hàm và thư viện
4.2. Quy trình xây dựng hàm

Chương 5: Dữ liệu kiểu mảng

5.1. Khái niệm và khai báo mảng
5.2. Một số bài toán với mảng 1 chiều
5.3. Một số bài toán với mảng 2 chiều
TH Bài 4: Xây dựng hàm và xử lý mảng 1 chiều các hàm đơn giản
TH Bài 5: Xây dựng hàm và xử lý mảng 2 chiều các hàm đơn giản

Chương 6: Xâu ký tự (string)

6.1. Khái niệm và khai báo xâu
6.2. Một số hàm xử lý ký tự và xâu
6.3. Một số chương trình xử lý dữ liệu kiểu xâu

TH Bài 6: Mảng và xử lý xâu ký tự

Bài tập tổng hợp

CHƯƠNG 1: TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH C

Kiến thức:

Trình bày được lịch sử phát triển của ngôn ngữ C, các phần tử cơ bản trong ngôn ngữ C và cấu trúc cơ bản của một chương trình C.

Kĩ năng

Thực hiện được cài đặt trình biên dịch và viết được chương trình C cơ bản.

Thái độ

Vận dụng tích cực, linh hoạt các kiến thức đã học vào các bài học tiếp theo, áp dụng viết các chương trình cơ bản.

1.1. Giới thiệu sơ lược về ngôn ngữ lập trình C

1.1.1. Lịch sử phát triển ngôn ngữ C

Ngôn ngữ C ra đời tại phòng thí nghiệm BELL của tập đoàn AT&T (Hoa Kỳ) do Brian W. Kernighan và Dennis Ritchie phát triển vào đầu những năm 1970 và hoàn thành vào năm 1972.

C được phát triển dựa trên nền các ngôn ngữ BCPL (*Basic Combined Programming Language*) và ngôn ngữ B. Cũng vì được phát triển dựa trên nền ngôn ngữ B nên ngôn ngữ mới được Brian W. Kernighan và Dennis Ritchie đặt tên là ngôn ngữ C như là sự tiếp nối ngôn ngữ B.

C có các đặc điểm là một ngôn ngữ lập trình hệ thống mạnh, khả chuyển, có tính linh hoạt cao và có thế mạnh trong xử lí các dạng dữ liệu số, văn bản, cơ sở dữ liệu. Vì vậy C thường được dùng để viết các chương trình hệ thống như hệ điều hành (ví dụ hệ điều hành Unix có 90% mã nguồn được viết bằng C, 10% còn lại viết bằng hợp ngữ) và các chương trình ứng dụng chuyên nghiệp có can thiệp tới dữ liệu ở mức thấp như xử lí văn bản, cơ sở dữ liệu, xử lí ảnh…

W. Kernighan và Dennis Ritchie công bố ngôn ngữ C trong lần xuất bản đầu của cuốn sách "*The C programming language*" (1978). Sau đó người ta đã bổ sung thêm những yếu tố và khả năng mới vào trong ngôn ngữ C (ví dụ như đưa thêm kiểu liệt kê **enum**, cho phép kiểu dữ liệu trả về bởi hàm là kiểu **void**, **struct** hoặc **union**… và đặc biệt là sự bổ sung các thư viện cho ngôn ngữ. Lúc đó đồng thời tồn tại nhiều phiên bản khác nhau của ngôn ngữ C nhưng không tương thích với nhau. Điều này gây khó khăn cho việc trao đổi mã nguồn chương trình C viết trên các phiên bản ngôn ngữ C khác nhau (bạn sẽ rất khó đọc và hiểu chương trình của người khác, và khi bạn muốn sửa nó thành chương trình của mình dịch trên bộ dịch của mình thì sẽ tốn rất nhiều công sức) và dẫn đến nhu cầu chuẩn hóa ngôn ngữ C.

Hiện nay cũng có nhiều phiên bản của ngôn ngữ C khác nhau và mỗi phiên bản này gắn liền với một bộ chương trình dịch cụ thể của ngôn ngữ C. Các bộ chương trình dịch phổ biến của ngôn ngữ C có thể kể tên như:

- Turbo C++ và Borland C++ của Borland Inc.
- MSC và VC của Microsoft Corp.
- GCC của GNU project.
- Free C

…

1.1.2. Các bước cơ bản khi viết chương trình trên C

Để giải bài toán bằng chương trình thực hiện theo các bước sau:

1. Xác định đối tượng của chương trình
2. Xác định phương pháp và thuật giải
3. Viết chương trình (lập trình)
4. Chạy chương trình và kiểm tra kết quả.

Như vậy ta thấy chu trình phát triển một chương trình như sau:

1. Soạn thảo chương trình nguồn

Chúng ta có thể sử dụng một trình soạn thảo văn bản chuẩn (ASCII) nào đó để soạn thảo chương trình, sau đó ghi vào file chương trình nguồn (ngầm định với phần mở rộng là .C).

Do C cũng như hầu hết các ngôn ngữ lập trình phổ biến đều sử dụng bảng chữ cái ASCII nên bạn có thể sử dụng bất kỳ một hệ soạn thảo văn bản chuẩn để viết chương trình, tuy nhiên hầu hết các trình biên dịch của C trên môi trường MS-DOS hoặc WINDOWS đều có tích hợp trình soạn thảo và bạn nên sử dụng trình soạn thảo tích hợp này sẽ thuận lợi hơn.

2. Biên dịch chương trình nguồn

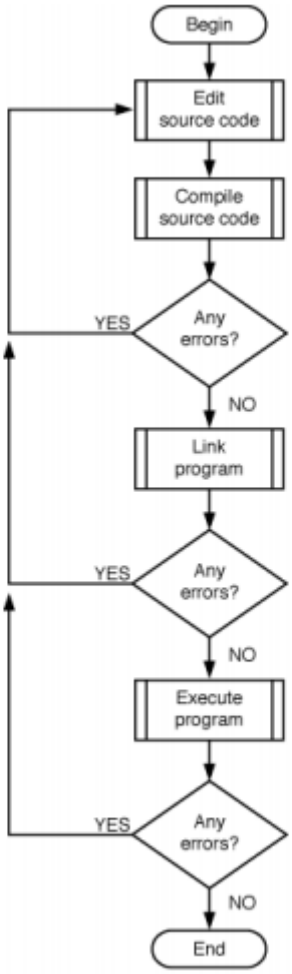
Hiện nay có rất nhiều chương trình dịch cho C như: Turbo C, BC, Microsoft C,..mục đích của bước này là chuyển chương trình nguồn thành chương trình mã đối tượng (object). Sau bước này (nếu thành công) chúng ta thu được file chương trình đối tượng (có phần mở rộng là .OBJ)

3. Liên kết chương trình

Sau bước biên dịch hoàn thành ta có chương trình đối tượng, đây chưa phải là chương trình có thể chạy được trên máy tính, bước này chúng ta phải sử dụng một trình liên kết để liên kết các hàm thư viện với chương trình đối tượng để tạo ra chương trình đích. Bạn có thể sử dụng trình liên kết độc lập nào đó, nhưng với các trình biên dịch của C trên môi trường DOS hay WINDOWS đều có sẵn trình liên kết.

4. Chạy và kiểm tra kết quả chương trình

Khi đã có chương trình đích, chúng ta cần phải kiểm tra tính đúng đắn của nó.Bạn chạy chương trình với các bộ dữ liệu mẫu và kiểm tra kết quả có như dự kiến hay không, nếu có sai sót thì phải xác định nguyên nhân gây lỗi và quay lại bước 1 để hiệu chỉnh. Và chúng ta lặp lại quá trình này cho tới khi được chương trình giải đúng bài toán mong đợi.



Các bước phát triển chương trình

Hiện nay có rất nhiều chương trình dịch cho C và hầu hết (trên nền DOS hoặc Windows) trong đó được tích hợp cả trình soạn thảo, biên dịch, liên kết - gọi là môi trường tích hợp. Trong giáo trình này chúng ta sử dụng BC (Borland C) hoặc turbo C làm môi trường lập trình.

1.2. Các phần tử cơ bản của C

1.2.1. Tập ký tự cơ bản

Tập kí tự sử dụng trong ngôn ngữ lập trình C gồm có:

1.2.2. Từ khóa (Keyword)

26 chữ cái hoa:	A B C ... X Y Z
26 chữ cái thường:	a b c … x y z.
10 chữ số:	0 1 2 3 4 5 6 7 8 9.
Các kí hiệu toán	+ - * / = <>
Các dấu ngăn cách:	. ; , : space tab
Các dấu ngoặc:	( ) [ ] { }
Các kí hiệu đặc	_ ? $ & # ^ \ ! ‘ “ ~ .v.v.

Từ khóa (*Keyword*) là những từ có sẵn của ngôn ngữ và được sử dụng dành riêng cho những mục đích xác định.

Một số từ khóa hay dùng trong Turbo C

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

Chú ý: Tất cả các từ khóa trong C đều viết bằng chữ thường.

Các từ khóa trong C được sử dụng để

- Đặt tên cho các kiểu dữ liệu: **int**, **float**, **double**, **char**, **struct**, **union**…
- Mô tả các lệnh, các cấu trúc điều khiển: **for**, **do**, **while**, **switch**, **case**, **if**, **else**, **break**, **continue**…

1.2.3. Định danh (Tên) - Identifier

Định danh (*Identifier* – hoặc còn gọi là *Tên*) là một dãy các kí tự dùng để gọi tên các đối tượng trong chương trình. Các đối tượng trong chương trình gồm có biến, hằng, hàm, kiểu dữ liệu… ta sẽ làm quen ở những mục tiếp theo.

Khi đặt tên cho định danh trong C, người lập trình cần tuân thủ các quy tắc sau :

1. Các kí tự được sử dụng trong các định danh của ngôn ngữ C chỉ được gồm có: chữ cái, chữ số và dấu gạch dưới "_" (*underscore*).
2. Bắt đầu của định danh phải là chữ cái hoặc dấu gạch dưới, không được bắt đầu định danh bằng chữ số.
3. Định danh do người lập trình đặt không được trùng với từ khóa.
4. Turbo C++ không giới hạn độ dài của định danh, nhưng chỉ 32 kí tự đầu của định danh được chương trình biên dịch sử dụng (khi định danh có độ dài lớn hơn 32 kí tự thì Turbo C++ sẽ tự động cắt bỏ, không xem xét các kí tự cuối bắt đầu từ kí tự thứ 33).

Một số ví dụ về định danh:

i, x, y, a, b, _function, _MY_CONSTANT, PI, gia_tri_1…

Ví dụ về định danh không hợp lệ

1_a, 3d, 55x	bắt đầu bằng chữ số
so luong, ti le	có kí tự không hợp lệ (dấu cách – *space*) trong tên
int, char	trùng với từ khóa của ngôn ngữ C

1.2.4. Kiểu dữ liệu

Dữ liệu là đối tượng được lưu trữ và xử lý trong máy tính. Dữ liệu trong máy tính lại không phải tất cả đều giống nhau. Có dữ liệu là chữ viết, có dữ liệu là con số, lại có dữ liệu khác là hình ảnh, âm thanh… Ta nói rằng các dữ liệu đó thuộc các kiểu dữ liệu khác nhau.

Kiểu dữ liệu trong C có thể phân loại thành 3 nhóm:

- Kiểu đơn giản (cơ bản/cơ sở) gồm kiểu số nguyên,số thực,ký tự
- Kiểu dữ liệu có cấu trúc gồm kiểu mảng, xâu (chuỗi), struct (bản ghi),tệp.
- Kiểu con trỏ

Một cách hình thức, kiểu dữ liệu có thể được định nghĩa gồm 2 điểm như sau:

Một kiểu dữ liệu là một tập hợp các giá trị mà một dữ liệu thuộc kiểu dữ liệu đó có thể nhận được (dải giá trị/miền giá trị).

Trên một kiểu dữ liệu ta xác định một số phép toán đối với các dữ liệu thuộc kiểu dữ liệu đó (tập thao tác/ phép toán trên tập giá trị).

Bảng sau liệt kê các kiểu dữ liệu đơn giản trong C

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 $1.1 * 10^{4932}$

Ví dụ:

Trong ngôn ngữ C có kiểu dữ liệu **int**. Một dữ liệu thuộc kiểu dữ liệu **int** thì nó sẽ là một số nguyên (*integer*) và nó có thể nhận giá trị từ - 32,768 ($- 2^{15}$) đến 32,767 ($2^{15} - 1$). Trên kiểu dữ liệu **int** ngôn ngữ C định nghĩa các phép toán số học đối với số nguyên như

Tên phép toán	**Kí hiệu**
Đảo dấu	-
Cộng	+
Trừ	-
Nhân	*
Chia lấy phần nguyên	/
Chia lấy phần dư	%
So sánh bằng	= =
So sánh lớn hơn	>
So sánh nhỏ hơn	<
…	

Trong máy tính, việc phân biệt kiểu dữ liệu là cần thiết vì qua kiểu dữ liệu máy tính biết được đối tượng mà nó đang xử lí thuộc dạng nào, có cấu trúc ra sao, có thể thực hiện các phép xử lí nào đối với đối tượng đó, hay là cần phải lưu trữ đối tượng đó như thế nào…

1.2.5. Hằng

Hằng (*constant*) là đại lượng có giá trị xác định và không thay đổi trong chương trình. Để giúp chương trình dịch nhận biết hằng ta cần nắm được cách biểu diễn hằng trong một chương trình C.

Biểu diễn hằng số nguyên

Trong ngôn ngữ C, một hằng số nguyên có thể được biểu diễn dưới những dạng sau

- Dạng thập phân: đó chính là cách viết giá trị số đó dưới hệ đếm cơ số 10 thông thường.
- Dạng thập lục phân: ta viết giá trị số đó dưới dạng hệ đếm cơ số 16 và thêm tiền tố **0x** ở đầu.
- Dạng bát phân: ta viết giá trị số đó dưới dạng hệ đếm cơ số 8 và thêm tiền tố **0** ở đầu.

Ví dụ

Giá trị thập phân	Giá trị hệ bát phân	Giá trị hệ thập lục phân
2007 396	03727 0614	0x7D7 0x18C

Biểu diễn hằng số thực

Có 2 cách biểu diễn hằng số thực:

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

Ví dụ:

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

Biểu diễn hằng kí tự

Có 2 cách biểu diễn hằng kí tự:

- Bằng kí hiệu của kí tự đó đặt giữa 2 dấu nháy đơn.
- Bằng số thứ tự của kí tự đó trong bảng mã ASCII (và lưu ý số thứ tự của một kí tự trong bảng mã ASCII là một số nguyên nên có một số cách biểu diễn).

Ví dụ

Kí tự cần biểu diễn	Cách 1	Cách 2		
Chữ cái A	‘A’	65	hoặc 0101	hoặc 0x41
Dấu nháy đơn ‘	‘\’’	39	hoặc 047	hoặc 0x27
Dấu nháy kép “	‘\”’	34	hoặc 042	hoặc 0x22
Dấu gạch chéo ngược \	‘\\’	92	hoặc 0134	hoặc 0x5c
Kí tự xuống dòng	‘\n’			
Kí tự NUL	‘\0’	0	hoặc 00	hoặc 0x0
Kí tự Tab	‘\t’	9	hoặc 09	hoặc 0x9

Biểu diễn hằng xâu kí tự

Hằng xâu kí tự được biểu diễn bởi dãy các kí tự thành phần có trong xâu đó và được đặt trong cặp dấu nháy kép.

Ví dụ:

"ngon ngu lap trinh C", "tin hoc dai cuong"…

Để sử dụng, hằng phải được đặt tên. Tên hằng đặt theo quy tắc định danh

1.2.6. Biến

Biến (*variable*) là đặc trưng cho một đại lượng được xử lý trong bài toán, đại lượng này có giá trị có thể thay đổi trong chương trình. Tại một thời điểm xác định, giá trị của biến là một hằng.

Trong chương trình, hằng và biến được sử dụng để lưu trữ dữ liệu, và dữ liệu lưu trữ trong biến, hằng phải thuộc một kiểu dữ liệu nào đó.

Biến và hằng đều phải được đặt tên để khi cần thì có thể gọi đến. Tên biến và hằng được đặt theo quy tắc đặt tên cho định danh.

1.2.7. Hàm

Trong lập trình chúng ta rất hay phải tính toán giá trị của một số đại lượng thường gặp như sin(x), cos(x), căn bậc hai, lũy thừa, logarithm…

Ngôn ngữ C cung cấp cho người lập trình một công cụ dùng để tính toán giá trị các đại lượng đó mỗi khi cần trong chương trình, đó là các hàm.

Một số hàm toán học hay được sử dụng trong C

Hàm	**Ý nghĩa**	**Kí hiệu toán học**	**Ví dụ**
sqrt(x)	Căn bậc 2 của *x*	$\sqrt{x}$	sqrt(16.0) bằng 4.0
pow(x,y)	*x* mũ *y*	x^y	pow(2,3) bằng 8
exp(x)	*e* mũ *x*	e^x	exp(1.0) bằng 2.718282
log(x)	logarithm tự nhiên (cơ số e) của x	*lnx*	log(2.718282) bằng 1.0
log10(x)	logarithm cơ số 10 của x	*logx*	log10(100) bằng 2
sin(x)	sin của x	*sinx*	sin(0.0) bằng 0.0
cos(x)	cosin của x	*cosx*	cos(0.0) bằng 1.0
tan(x)	tang của x	*tgx*	tan(0.0) bằng 0.0
ceil(x)	phần nguyên già của x, tức là số nguyên nhỏ nhất không nhỏ hơn x	$\lceil x \rceil$	ceil(2.5) bằng 3 ceil(-2.5) bằng –2
floor(x)	phần nguyên non của x, tức là số nguyên lớn nhất không lớn hơn x	$\lfloor x \rfloor$	floor(2.5) bằng 2 floor(-2.5) bằng –3

1.2.8. Biểu thức

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

Các toán hạng trong biểu thức có thể là biến, hằng, hàm hoặc một biểu thức khác. Bản thân một biến, hằng, hàm đứng độc lập cũng được coi là một biểu thức.

Các toán tử trong biểu thức rất đa dạng như cộng, trừ, nhân, chia, so sánh…

Biểu thức thường là sự thể hiện công thức tính toán giá trị một đại lượng nào đó.

Ví dụ về biểu thức:

chieu_dai * chieu_rong * chieu_cao

Trong biểu thức trên chieu_dai, chieu_rong, chieu_cao là các biến hoặc hằng, * là kí hiệu của toán tử nhân. Nếu chieu_dai, chieu_rong, chieu_cao là các biến (hoặc hằng) lưu trữ giá trị chiều dài, chiều rộng và chiều cao của một khối hộp chữ nhật thì biểu thức trên sẽ tính giá trị thể tích của khối hộp chữ nhật đó.

1.2.9. Câu lệnh

Câu lệnh (*statement*) diễn tả một hoặc một nhóm các thao tác trong giải thuật. Chương trình được tạo thành từ dãy các câu lệnh.

Cuối mỗi câu lệnh đều có dấu chấm phẩy ';' để đánh dấu kết thúc câu lệnh cũng như để phân tách các câu lệnh với nhau.

Câu lệnh được chia thành 2 nhóm chính:

- Nhóm các câu lệnh đơn: là những câu lệnh không chứa câu lệnh khác. Ví dụ: phép gán, phép cộng, phép trừ…
- Nhóm các câu lệnh phức: là những câu lệnh chứa câu lệnh khác trong nó. Ví dụ: lệnh khối, các cấu trúc lệnh rẽ nhánh, cấu trúc lệnh lặp…

Lệnh khối là một số các lệnh đơn được nhóm lại với nhau và đặt trong cặp dấu ngoặc nhọn { } để phân tách với các lệnh khác trong chương trình.

1.2.10. Chú thích

Để giúp việc đọc và hiểu chương trình viết ra được dễ dàng hơn, chúng ta cần đưa vào các lời chú thích (*comment*). Lời chú thích là lời mô tả, giải thích vắn tắt cho một câu lệnh, một đoạn chương trình hoặc cả chương trình, nhờ đó người đọc có thể hiểu được ý đồ của người lập trình và công việc mà chương trình đang thực hiện.

Khi gặp kí hiệu lời chú thích trong chương trình, trình biên dịch sẽ tự động bỏ qua không dịch phần nội dung nằm trong phạm vi của vùng chú thích đó.

Trong C, có 2 cách để viết lời chú thích

Dùng 2 dấu sổ chéo liên tiếp // để kí hiệu toàn bộ vùng bắt đầu từ 2 dấu sổ chéo liên tiếp đó đến cuối dòng là vùng chú thích. Ví dụ:

```
// khai bao 2 bien nguyen
int a, b;
a = 5; b = 3;   // khoi tao gia tri cho cac bien nay
```

Cách này thường dùng nếu đoạn chú thích ngắn, có thể viết đủ trên một dòng.

Dùng 2 cặp kí hiệu /* và */ để kí hiệu rằng toàn bộ vùng bắt đầu từ cặp kí hiệu /* kéo dài đến cặp kí hiệu */ là vùng chú thích. Ví dụ:

```
/* doan chuong trinh sau khai bao 2 bien nguyen va khoi tao gia tri cho 2 bien nguyen nay */
int a, b;
a = 5; b = 3;
```

Cách này thường dùng khi đoạn chú thích dài, phải viết trên nhiều dòng.

1.3. Cấu trúc cơ bản của chương trình C

Về cơ bản, mọi chương trình viết bằng ngôn ngữ C sẽ có cấu trúc gồm các phần có thứ tự như sau:

Phần 1: Khai báo tệp tiêu đề #include
Phần 2: Định nghĩa hằng #define hay const
Phần 3: Định nghĩa kiểu dữ liệu typedef ...
Phần 4: Khai báo các nguyên mẫu hàm
Phần 5: Khai báo các biến toàn cục
Phần 6: Định nghĩa các hàm

Phần 7: Định nghĩa hàm **main()**
main()
{
...
}

Phần 8: Định nghĩa các hàm đã khai báo nguyên mẫu

- Phần 1: Phần khai báo các tệp tiêu đề. Phần này có chức năng thông báo cho chương trình dịch biết là chương trình có sử dụng những thư viện nào (mỗi tệp tiêu đề tương ứng với một thư viện).
- Phần 2: Định nghĩa các hằng mới dùng cho cả chương trình.
- Phần 3: Định nghĩa các kiểu dữ liệu mới dùng cho cả chương trình
- Phần 4: Phần khai báo các hàm nguyên mẫu. Phần này giúp cho chương trình dịch biết được những thông tin cơ bản (gồm tên hàm, dach sách các tham số và kiểu dữ liệu trả về) của các hàm sử dụng trong chương trình.
- Phần 5: Phần khai báo các biến toàn cục.
- Phần 6: Định nghĩa các hàm.
- Phần 7: Phần định nghĩa hàm **main( ).** Hàm **main( )** là một hàm đặc biệt trong C. Khi thực hiện, chương trình sẽ gọi hàm **main( )**, hay nói cách khác chương trình sẽ bắt đầu bằng việc thực hiện các lệnh trong hàm **main( ).** Trong hàm **main( )** ta mới gọi tới các hàm khác.
- Phần 8: Phần định nghĩa các hàm đã khai báo nguyên mẫu. Ở phần 3 ta đã khai báo nguyên mẫu (*prototype*) của các hàm, trong đó chỉ giới thiệu các thông tin cơ bản về hàm như tên hàm, danh sách các tham số và kiểu dữ liệu trả về. Nguyên mẫu hàm không cho ta biết cách thức cài đặt và hoạt động của các hàm. Ta sẽ làm việc đó ở phần định nghĩa các hàm.

Trong các phần trên, thì phần 7 định nghĩa hàm **main( )** bắt buộc phải có trong mọi chương trình C. Các phần khác có thể có hoặc không.

1.4. Cài đặt và sử dụng môi trường turbo C

1.4.1. Cài đặt

Để sử dụng Turbo C++ 3.0 ta cần phải cài đặt nó lên máy. Quá trình cài đặt thực hiện theo các bước sau:

Bước 1: Bạn cần chuẩn bị đĩa chứa bộ cài của Turbo C++ 3.0, kích thước của bộ cài khoảng 4 MB. Hãy copy bộ cài này vào máy của bạn, giả sử vào thư mục *C:\TC_Setup*.

Bước 2: Tìm đến thư mục chứa bộ cài Turbo C++ 3.0 (như giả sử ở trên là *C:\TC_Setup*) và kích hoạt file *INSTALL.EXE* để chạy chương trình cài đặt Turbo C++ 3.0. Chương trình cài đặt Turbo C++ 3.0 ban đầu sẽ yêu cầu bạn chỉ ra ổ đĩa trên đó chứa bộ cài Turbo C++ 3.0

Enter the SOURCE drive to use:

Hãy nhập vào tên ổ đĩa, chẳng hạn C (ta để bộ cài Turbo C++ 3.0 ở thư mục *C:\TC_Setup*).

Bước 3: Sau đó chương trình yêu cầu bạn nhập vào đường dẫn tới thư mục chứa các file của Turbo C++ 3.0

Enter the SOURCE Path:

Thông thường chương trình sẽ tự tìm cho bạn, và bạn chỉ cần ấn Enter để chuyển sang bước tiếp theo.

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

Directories… [C:\TC]

Option… [IDE CMD LIB CLASS BGI HELP EXMPL]

Start Installation

Thư mục cài đặt mặc định sẽ là *\TC* nằm trên thư mục gốc của ổ đĩa chứa bộ cài. Nếu bạn muốn thay đổi thư mục cài đặt thì hãy dùng các phím ↑ và ↓ để di chuyển hộp sáng đến phần Directories, gõ Enter và nhập vào đường dẫn mới, sau đó ấn phím Esc để trở về.

Dùng các phím ↑ và ↓ để di chuyển hộp sáng đến phần Start Installation và ấn Enter.

1.4.2. Sử dụng môi trường Turbo

Sau khi cài đặt xong, bạn có thể tìm đến thư mục *BIN* trong thư mục cài đặt và chạy file *TC.EXE* để khởi động Turbo C++ 3.0.

Sau khi khởi động Turbo C++ 3.0 sẽ xuất hiện màn hình làm việc của Turbo C++ 3.0.

Bạn dùng chuột di chuyển đến menu File (hoặc ấn Alt-F), sau đó chọn mục New để mở cửa sổ soạn thảo mới.

Giờ hãy gõ vào toàn bộ chương trình viết bằng ngôn ngữ C của bạn lên cửa sổ soạn thảo này. Ấn phím F2 để lưu trữ tệp chương trình nguồn trên máy, một cửa sổ cất giữa tệp sẽ hiện ra yêu cầu bạn nhập vào tên mới cho tệp chương trình nguồn (tên mặc

định sẽ là *NONAME.CPP*). Hãy đặt một tên cho tệp rồi chọn OK để lưu tệp chương trình nguồn lại.

Cuối cùng là ấn phím F9 để biên dịch chương trình viết ra. Nếu chương trình của bạn có lỗi thì Turbo C++ 3.0 sẽ báo lỗi và bạn phải sửa lại đến khi không còn lỗi. Nếu chương trình bạn không có lỗi thì Turbo C++ 3.0 sẽ thông báo biên dịch thành công và bạn có thể ấn Ctrl-F9 để chạy chương trình đã biên dịch.

CHƯƠNG 2: KHỞI ĐẦU VỚI LẬP TRÌNH C

MỤC TIÊU CỦA BÀI

Kiến thức:

- Trình bày được các phép toán trong C;
- Trình bày được các lệnh vào ra dữ liệu trong C.

Kĩ năng

Tạo được một chương trình đơn giản nhập và xuất ra màn hình.

Thái độ

Vận dụng tích cực, linh hoạt các kiến thức đã học vào các bài học tiếp theo, áp dụng viết các chương trình cơ bản.

2.1. Một số khai báo cơ bản

2.1.1. Khai báo tệp tiêu đề

Trong chương trình C (trong hàm main cũng như các hàm khác do người lập trình viết) có thể sử dụng các hàm, hằng, kiểu dữ liệu,..(gọi chung là các thành phần) đã được định nghĩa trong thư viện của C. Để sử dụng các thành phần này chúng ta phải chỉ dẫn cho chương trình dịch biết các thông tin về các thành cần sử dụng, các thông tin đó được khai báo trong tệp gọi là tệp tiêu đề (có phần mở rộng là H – viết tắt của header). Cú pháp của một dòng bao hàm tệp:

#include <tên_tệp>

hoặc

#include “tên_tệp”

trong đó tên_tệp là tên có thể có cả đường dẫn của tệp tiêu đề (.H) mà chúng ta cần sử dụng, mỗi lệnh bao hàm tệp trên một dòng.

Ví dụ:

```
#include <stdio.h>
#include <conio.h>
#include “phanso.h”
```

2.1.2. Khai báo hằng

Có 2 cách để khai báo hằng trong C là dùng chỉ thị **#define** hoặc khai báo với từ khóa **const**.

Dùng chỉ thị **#define**

Cú pháp khai báo:

 # define tên_hằng giá_trị

Lưu ý không có dấu chấm phẩy ở cuối dòng chỉ thị.

```
#define MAX_SINH_VIEN  50                    // hằng kiểu số nguyên
```

Dùng từ khóa **const** để khai báo với cú pháp:

const kiểu_dữ_liệu tên_hằng = giá_trị;

Khai báo này giống với khai báo biến có khởi tạo giá trị đầu, tuy nhiên cần lưu ý:

- Do có từ khóa **const** ở đầu cho nên giá trị của đối tượng tên_hằng sẽ không được phép thay đổi trong chương trình. Những lệnh nhằm làm thay đổi giá trị của tên_hằng trong chương trình sẽ dẫn tới lỗi biên dịch.

• Trong khai báo biến thông thường, người lập trình có thể khởi tạo giá trị cho biến ngay từ khi khai báo hoặc không khởi tạo cũng được. Nhưng trong khai báo hằng, giá trị của tất cả các hằng cần được xác định ngay trong lệnh khai báo.

Các khai báo hằng ở ví dụ trước giờ có thể viết lại theo cách khác như sau:

```
const int  MAX_SINH_VIEN  = 50;
```

2.1.3. Khai báo biến

Một biến trước khi sử dụng phải được khai báo. Cú pháp khai báo:

kiểu_dữ_liệu tên_biến;

Ví dụ:

```
float x;        // biến kiểu thực
float y;        // biến kiểu thực
double z;       // biến kiểu thực
int i;   // biến kiểu nguyên
int j;   // biến kiểu nguyên
```

Nếu các biến thuộc cùng kiểu dữ liệu thì C cho phép khai báo chúng trên cùng một dòng:

kiểu_dữ_liệu danh_sách_tên_biến;

Ví dụ:

```
float x, y;
double z;
int i, j;
```

Sau khi khai báo, biến có thể nhận giá trị thuộc kiểu dữ liệu đã khai báo. Chúng ta có thể khởi tạo giá trị đầu cho biến nếu muốn với cú pháp:

kiểu_dữ_liệu tên_biến = giá_trị_đầu;

Ví dụ:

```
int a = 3;       // sau lenh nay bien a se co gia tri bang 3
```

2.1.4. Khai báo hàm main()

Một chương trình C bao hàm nhiều hàm. Trong đó hàm main() là hàm chính,luôn xuất hiện và chỉ xuất hiện 1 lần. Khái niệm về hàm của C như sau:

Hàm main() cũng không có tham số, vì vậy ta có thể gặp cách viết void main() hoặc void main(void). Phần thân của hàm main() gồm:

- Các khai báo hằng,biến,…
- Các chỉ thị thực hiện các thao tác theo nội dung hàm.

Ví dụ:

```
Void main (void)
{
Int in;
Printf(“\n Go vao 1 so nguyen la nam sinh cua ban”);
Scanf(“%d”,&n)’
Printf(“\n ban sinh nam %d”,n);
Getch();
}
```

2.2. Các phép toán trong C

2.2.1. Các phép toán (operators)

Các phép toán trong C được chia thành các nhóm phép toán cơ bản sau: nhóm các phép toán số học,nhóm các phép toán thao tác trên bit,nhóm các phép toán quan hệ,nhóm các phép toán logic.

2.2.1.1. Các phép toán cơ bản

Các phép toán số học (*Arithmetic operators*) gồm có:

Toán tử	**Ý nghĩa**	**Kiểu dữ liệu của toán hạng**	**Ví dụ**
-	Phép đổi dấu	Số thực hoặc số nguyên	int a, b; -12; -a; -25.6;
+	Phép toán cộng	Số thực hoặc số nguyên	float x, y; 5 + 8; a + x; 3.6 + 2.9;
-	Phép toán trừ	Số thực hoặc số nguyên	3 – 1.6; a – 5;
*	Phép toán nhân	Số thực hoặc số nguyên	a * b; b * y; 2.6 * 1.7;
/	Phép toán chia	Số thực hoặc số nguyên	10.0/3.0; (bằng 3.33…) 10/3.0; (bằng 3.33…) 10.0/3; (bằng 3.33…)
/	Phép chia lấy phần nguyên	Giữa 2 số nguyên	10/3; (bằng 3)
%	Phép chia lấy phần dư	Giữa 2 số nguyên	10%3; (bằng 1)

Các phép toán quan hệ (*Comparison operators*) gồm có:

Toán tử	**Ý nghĩa**	**Ví dụ**
>	So sánh lớn hơn giữa 2 số nguyên hoặc thực.	2 > 3 (có giá trị 0) 6 > 4 (có giá trị 1) a > b
>=	So sánh lớn hơn hoặc bằng giữa 2 số nguyên hoặc thực.	6 >= 4 (có giá trị 1) x >= a
<	So sánh nhỏ hơn giữa 2 số nguyên hoặc thực.	5 < 3 (có giá trị 0),
<=	So sánh nhỏ hơn hoặc bằng giữa 2 số nguyên hoặc thực.	5 <= 5 (có giá trị 1) 2 <= 9 (có giá trị 1)
==	So sánh bằng nhau giữa 2 số nguyên hoặc thực.	3 == 4 (có giá trị 0) a == b
!=	So sánh không bằng (so sánh khác) giữa 2 số nguyên hoặc thực.	5 != 6 (có giá trị 1) 6 != 6 (có giá trị 0)

Các phép toán logic (*Logical operators*) gồm có

Toán tử	**Ý nghĩa**	**Kiểu dữ liệu của toán hạng**	**Ví dụ**
&&	Phép VÀ LOGIC. Biểu thức VÀ LOGIC bằng 1 khi và chỉ khi cả 2 toán hạng đều bằng 1	Hai biểu thức logic	3<5 && 4<6 (có giá trị 1) 2<1 && 2<3 (có giá trị 0) a > b && c < d
||	Phép HOẶC LOGIC. Biểu thức HOẶC LOGIC bằng 0 khi và chỉ khi cả 2 toán hạng bằng 0.	Hai biểu thức logic	6 || 0 (có giá trị 1) 3<2 || 3<3 (có giá trị 1) x >= a || x == 0
!	Phép PHỦ ĐỊNH LOGIC một ngôi. Biểu thức PHỦ ĐỊNH LOGIC có giá trị bằng 1 nếu toán hạng bằng 0 và có giá trị bằng 0 nếu toán hạng bằng 1	Biểu thức logic	!3 (có giá trị 0) !(2>5) (có giá trị 1)

Phép toán gán

Phép toán gán có dạng

tên_biến = biểu_thức;

Phép toán gán có chức năng lấy giá trị của biểu_thức gán cho tên_biến. Dấu = là kí hiệu cho toán tử gán.

Ví dụ:

```
int a, b, c;
a = 3;
b = a + 5;
c = a * b;
```

Sau đoạn lệnh trên, biến a có giá trị là 3, b có giá trị là 8 và c có giá trị là 24.

Trong phép toán gán nếu ta bỏ dấu ; ở cuối đi thì ta sẽ thu được biểu thức gán. Biểu thức gán là biểu thức có dạng

tên_biến = biểu_thức;

2.2.1.2. Các phép toán đặc trưng của C

Phép toán gán thu gọn

Xét lệnh gán sau

```
x = x + y;
```

Lệnh gán này sẽ tăng giá trị của biến x thêm một lượng có giá trị bằng giá trị của y. Trong C ta có thể viết lại lệnh này một cách gọn hơn mà thu được kết quả tương đương