TUYÊN BỐ BẢN QUYỀN

LỜI GIỚI THIỆU

Giáo Trình Lập trình căn bản C được biên soạn nhằm đáp ứng yêu cầu học tập của sinh viên bước đầu làm quen với công việc lập trình, đồng thời giúp cho sinh viên có một tài liệu học tập, rèn luyện tốt khả năng lập trình, tạo nền tảng vững chắc cho các Môn học tiếp theo.

Giáo trình không chỉ phù hợp cho người mới bắt đầu mà còn phù hợp cho những người cần tham khảo. Nội dung của giáo trình được chia thành 7 chương:

Chương 1: Tổng quan về ngôn ngữ C

Chương 2: Các thành phần cơ bản

Chương 3: Các lệnh có cấu trúc

Chương 4: Hàm

Chương 5: Mảng

Chương 6: Chuỗi ký tự

Chương 7: Con trỏ

Khi biên soạn, chúng tôi đã tham khảo các giáo trình và tài liệu giảng dạy Môn học này của một số trường Cao đẳng, Đại học để giáo trình vừa đạt yêu cầu về nội dung vừa thích hợp với đối tượng là sinh viên của các trường Cao đẳng.

Chúng tôi hy vọng sớm nhận được những ý kiến đóng góp, phê bình của bạn đọc về nội dung, chất lượng và hình thức trình bày để giáo trình này ngày một hoàn thiện hơn.

Cần Thơ, ngày….....tháng…..... năm 2018

Tham gia biên soạn

1. Chủ biên: Lư Thục Oanh

MỤC LỤC

TRANG

LẬP TRÌNH CĂN BẢN

Tên Môn học/Môn học: LẬP TRÌNH CĂN BẢN

Mã Môn học/Môn học: MĐ10

Vị trí, tính chất, ý nghĩa và vai trò Môn học:

- Vị trí: Môn học được bố trí sau khi sinh viên học xong các Môn Học chung.
- Tính chất: Là Môn học lý thuyết cơ sở nghề.
- Ý nghĩa và vai trò: Là Môn học cơ sở, giúp sinh viên bước đầu làm quen với công việc lập trình.

Mục tiêu của Môn học:

- Trình bày được khái niệm về lập máy tính;
- Mô tả được ngôn ngữ lập trình: cú pháp, công dụng của các câu lệnh;
- Phân tích được chương trình: xác định nhiệm vụ chương trình;
- Thực hiện được các thao tác trong môi trường phát triển phần mềm: biên tập chương trình, sử dụng 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.
- Bố trí làm việc khoa học đảm bảo an toàn cho người và phương tiện học tập.

Số TT	Tên các bài trong mô đun	Thời gian (giờ)			
		Tổng số	Lý thuyết	Thực hành	Kiểm tra* (LT hoặc TH)
I.	**Chương 1: Tổng quan về ngôn ngữ C**	**4**	**2**	**2**	**0**
	Giới thiệu ngôn ngữ C	0.5	0.5	0	0
	Các thao tác cơ bản	3	1	2	0
	Sử dụng trợ giúp	0.5	0.5	0	0
II.	**Chương 2: Các thành phần cơ bản**	**8**	**4**	**4**	**0**
	Từ khóa và kí hiệu	1	0.5	0	0
	Các kiểu dữ liệu sơ cấp	1.5	0.5	1	0
	Biến, hằng, biểu thức	1.5	0.5	1	0
	Cấu trúc chương trình	2	1	1	0
	Câu lệnh	1.5	1	0.5	0

	Thực thi chương trình	1	0.5	0.5	0
III.	**Chương 3: Các lệnh có cấu trúc**	**22**	**6**	**15**	**1**
	Cấu trúc rẽ nhánh	7	2	5	0
	Cấu trúc lựa chọn	6	1	5	0
	Cấu trúc lặp	6	2	4	0
	Các lệnh Break, Continue	2	1	1	0
	Kiểm tra	*1*	*0*	*0*	*1*
IV.	**Chương 4: Hàm**	**14**	**4**	**9**	**1**
	Khái niệm hàm	1	1	0	0
	Xây dựng hàm	3	1	2	0
	Các tham số của hàm	5	1	4	0
	Hàm đệ quy	4	1	3	0
	Kiểm tra	*1*	*0*	*0*	*1*
V.	**Chương 5: Mảng**	**16**	**4**	**11**	**1**
	Khái niệm mảng	1	1	0	0
	Khai báo mảng	2	1	1	0
	Truy xuất mảng	12	2	10	
	Kiểm tra	*1*	*0*	*0*	*1*
VI.	**Chương 6: Con trỏ**	**14**	**6**	**8**	0
	Khái niệm con trỏ và địa chỉ	1	1	0	0
	Khai báo và sử dụng con trỏ	6	2	4	
	Con trỏ và mảng 1 chiều	4	2	2	0
	Con trỏ và hàm.	3	1	2	0
VII	**Chương 7: Chuỗi ký tự**	**12**	**4**	**7**	**1**
	Khái niệm	0.5	0.5	0	0
	Khai báo	1.5	0.5	1	0
	Các thao tác trên chuỗi	9	3	6	0
	Kiểm tra	*1*	*0*	*0*	*1*
	Cộng	**90**	**30**	**56**	**4**

CHƯƠNG 1: TỔNG QUAN VỀ NGÔN NGỮ C

Mã chương: MĐ10-01

Mục tiêu:

Sau khi học xong bài này sinh viên có khả năng:

- Hiểu được lịch sử phát triển của ngôn ngữ C.
- Biết được ngôn ngữ này có ứng dụng như thế nào.
- Biết cách khởi động được và thoát khỏi chương trình.
- Sử dụng hệ thống trợ giúp từ help file.

1. Giới thiệu về lịch sử phát triển của ngôn ngữ, sự cần thiết phải học ngôn ngữ C hiện nay.

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 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 C dựa trên ngôn ngữ BCPL (Basic Conbined Programming Languages) do Martin Richards đưa 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.Từ trước tới nay, ngôn ngữ C là ngôn ngữ lập trình được nhiều người sử dụng, vì C được xây dựng trên những thành tựu mới nhất của tin học.

Hình 1: (Dennis Ritchie(trái) và Ken Thompson trước hệ thống PDP-11với 2 text-terminal (1972))

Hình 2: (Dennis Ritchie (trái) và Kernighan)

Hiện nay ở các nước Âu, Mỹ ngôn ngữ lập trình dùng để giảng dạy ở bậc trung học và đại học là ngôn ngữ C và C++. Trong tương lai ngôn ngữ C sẽ trở thành ngôn ngữ quốc tế trong tin học cũng giống như ngày nay người ta sử dụng tiếng Anh trong giao tiếp.

Ngôn ngữ C có một thư viện khổng lồ các hàm (function) mà không có một ngôn ngữ nào sánh kịp. Lập trình viên khi sử dụng các hàm của C chỉ cần truyền tham số sẽ nhận được ngay kết quả mà không cần phải mất nhiều thời gian để viết chức năng tương tự.

Vì vậy công việc lập trình trở nên dễ dàng, nhanh chóng, ngắn gọn và tạo ra nhiều sản phẩm là những chương trình ứng dụng trong mọi lĩnh vực có chất lượng cao. Ngôn ngữ C rất thích hợp để giải quyết các bài toán kỹ thuật có những công thức và thuật toán phức tạp, vì ngôn ngữ C có những toán tử điều khiển rất mạnh mẽ.

Ngoài ra khái niệm cấu trúc trong C cho phép mô tả các khối dữ liệu lớn, vì vậy ngôn ngữ C cũng có thể được sử dụng để giải quyết các bài toán quản lý và xử lý các mô hình lựa chọn tối ưu. Một chương trình viết bằng ngôn ngữ C có thể chạy nhanh gần bằng Assembler.

Ngoài ra ngôn ngữ C là ngôn ngữ dễ học, chương trình viết ngắn gọn, súc tích có cấu trúc rõ ràng, dễ phát hiện sai lầm nếu có. Vì thế ngày càng được nhiều người đặt biệt ưa chuộng đặt biệt là các bạn học sinh và sinh viên. Ngôn ngữ C có những đặc điểm cơ bản như sau:

+ Tính cô động (compact): C chỉ có 32 từ khóa chuẩn và 40 toán tử chuẩn, nhưng đều được biểu diễn bằng những chuỗi ký tự ngắn gọn.

+ Tính cấu trúc (structured): C có một tập hợp những chỉ thị của lập trình như cấu trúc lựa chọn, lặp,… Từ đó các chương trình viết bằng C được tổ chức rõ ràng, dễ hiểu.

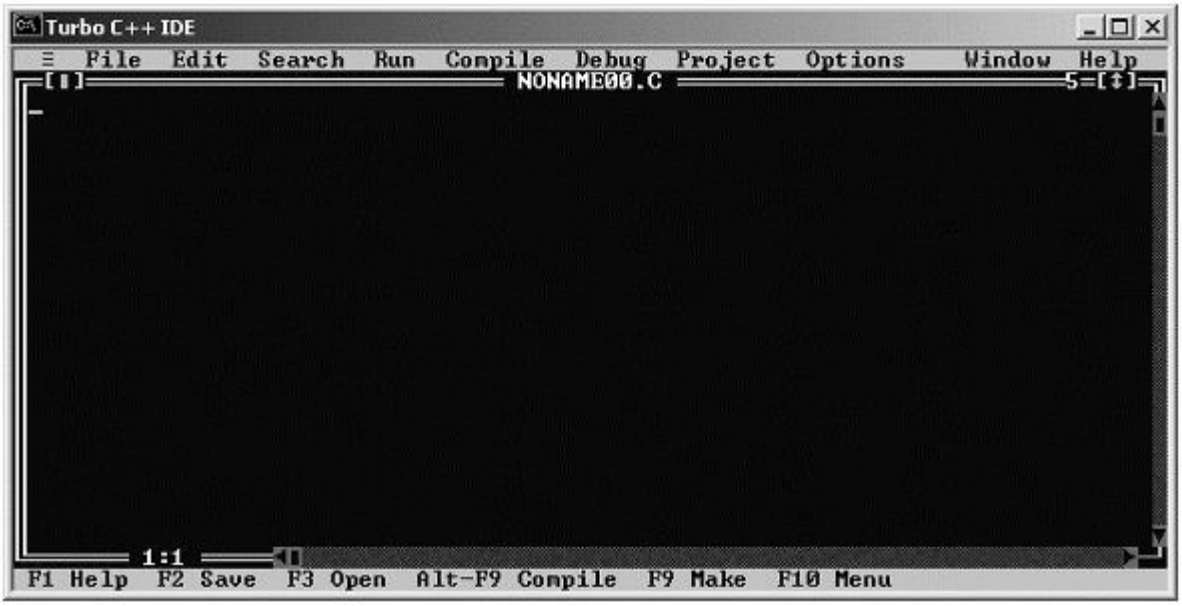
- Tính tương thích (compatible): C có một tiền xử lý và một thư viện chuẩn vô cùng phong phú nên khi chuyển từ máy tính này sang máy tính khác các chương trình viết bằng C vẫn hoàn toàn tương thích.
- Tính linh động (flexible): C là một ngôn ngữ rất uyển chuyển và cú pháp, chấp nhận nhiều cách thể hiện, có thể thu gọn kích thước của mã lệnh, làm chương trình chạy nhanh hơn.
- Biên dịch (compile): Cho phép C biên dich nhiều tập tin chương trình riêng lẽ 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 (executable) thi được thống nhất.

2. Cách khởi động và thoát chương trình (BorlandC):

Giới thiệu môi trường làm việc của Turbo C:

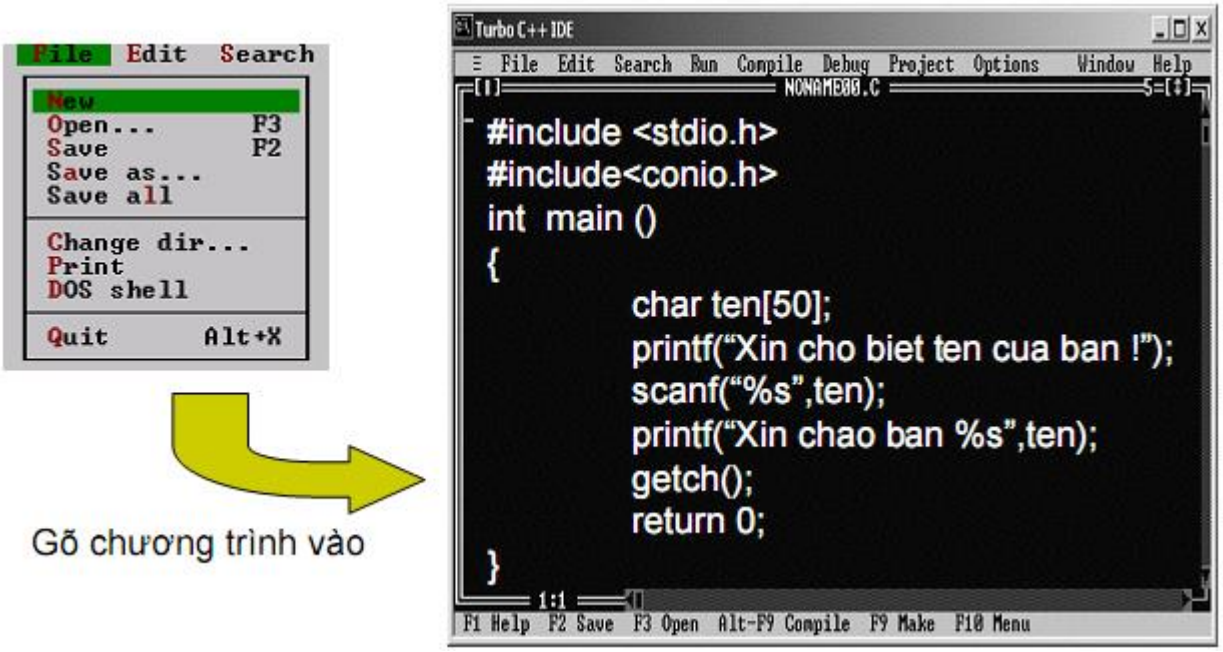
- Turbo C do hãng Borland cung cấp.

- Gồm các chức năng sau: soạn thảo chương trình, dịch, thực thi chương trình,…
- Phiên bản được sử dụng ở đây là Turbo C 3.0.
- Đây là màn hình gọi Turbo C:



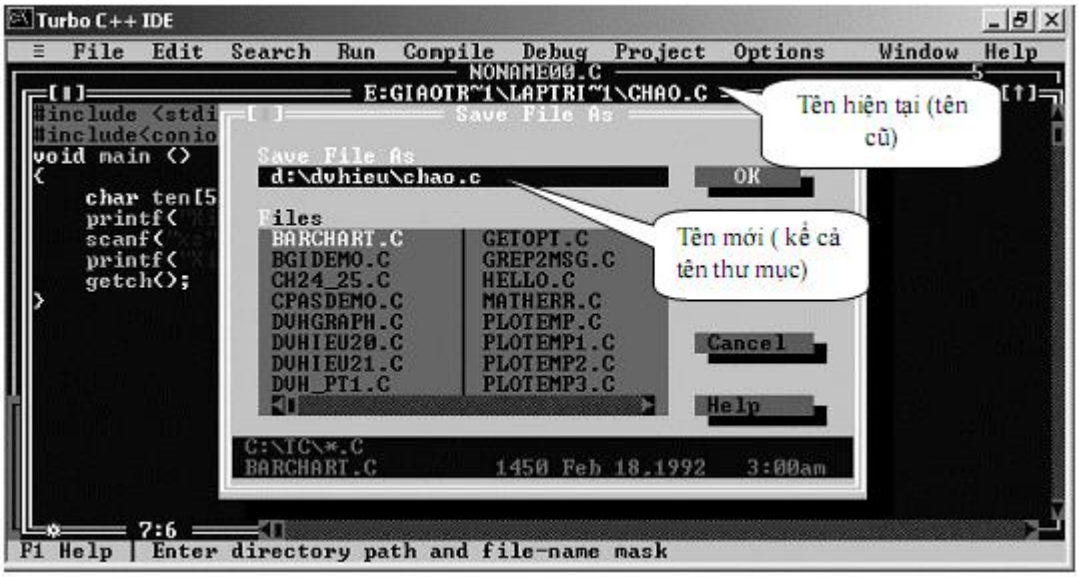
Hình 3: Màn hình Turbo C

- Giao diện soạn thảo chương trình mới: vào menu File ->New



Hình 4: Giao diện soạn thảo chương trình mới

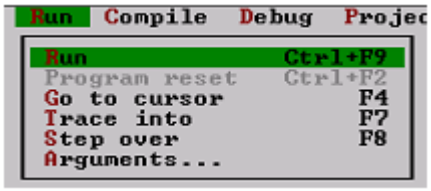
- Ghi chương trình đang soạn thảo vào đĩa:
- Sử dụng File->Save hoặc gõ phím F2.
- Lệnh Save As để lưu lại chương trình với tên khác.



Hình 5: Giao diện lưu chương trình mới

- Thực hiện chương trình, mở một chương trình và thoát:

+ Thực hiện chương trình: Nhấn Ctrl-F9 hoặc vào menu Run->Run.



Hình 6: Giao diện thực hiện chương trình

+ Mở một chương trình có sẵn trên đĩa: Vào menu File->Open hoặc F3.

Hình 7: Giao diện mở một chương trình có sẵn trên đĩa

+ Thoát khỏi Turbo C.

Vào menu File->Exit hoặc Alt-X.

- Các lệnh nằm trên menu Option:

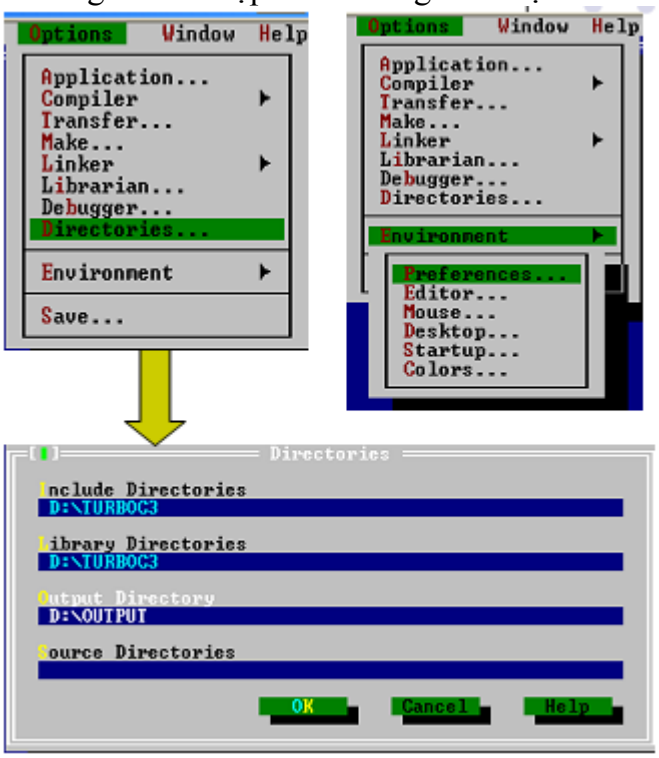
+ Directories:

▪ Include directories: chứa các tập tin muốn đưa vào chương trình (file .h trong dòng # include)

▪ Library directories: Chứa tập tin thư viện (file .lib).

- Output directory: Chứa các tập tin ”đối tượng “ .obj và .exe sau khi chạy chương trình.
- Source directories: Chứa các tập tin nguồn (.obj và .lib).

+ Environment: dùng để thiết lập môi trường làm việc.



Hình 7: Giao diện thiết lập môi trường làm việc

2.1. Khởi động:

- 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 trong thư mục BORLANDC nằm ở ổ đĩa nào. Sau đó ta gõ lệnh sau:

< ổ đĩa>:\BORLANDC\BIN\BC (Enter)

Nếu bạn muốn khởi động BC vừa soạn thảo chương trình với một tập tin có tên do 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 cho có sẽ được tạo mới.

- Khởi động tại Window**:** 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.

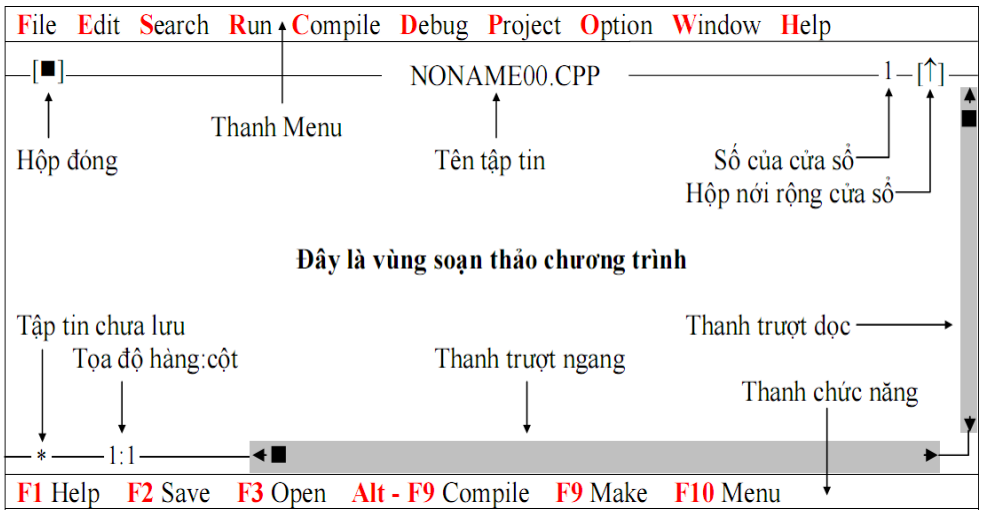
BORLANDC, vào thư mục BIN, khởi động tập tin BC.EXE

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.

Màn hình sau khi khởi động thành công:



Hình 8: Màn hình sau khi khởi động thành công

2.2. Thoát:

- Ấn phím **F10** (kích hoạt menu), chọn menu **File**, chọn **Quit**;
- Hoặc ấn tổ hợp phím **Alt-X**.

2.3. Các ví dụ đơn giản:

Ví dụ 1:

Dòng	File Edit Search Run Compile Debug Project Option Window Help
1	/* Chuong trinh in ra cau bai hoc C dau tien */
2	#include <stdio.h>
3	
4	void main(void)
5	{
6	printf("Bai hoc C dau tien.");
7	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

Hình 09: Màn hình ví dụ 1:

Kết quả in ra màn hình:

```
Bai hoc C dau tien. _
```

Hình 10: Màn hình kết quả ví dụ 1:

+ Dòng thứ 1: bắt đầu bằng /* và kết thúc bằng */ cho biết hang này diễn giải (chú thích). Khi dịch và chạy chương trình, dòng này không được dịch và cũng không thi hành lệnh gì cả. Mục đích của việc ghi chú này giúp chương trình rõ rang hơn. Sau này bạn đọc lại sẽ biết chương trình làm gì.

+ Dòng thứ 2: chứa phát biểu tiền xử lý **#include<stdio.h>.** Vì trong chương trình này ta sử dụng hàm thư viện của C là **printf**, do đó cần phải có khai báo của hàm thư viện này để báo cho chương trình biên dịch C biết nếu không khai báo chương trình sẽ thông báo lỗi.

+ Dòng thứ 3: hang trắng viết ra với ý đồ làm cho bảng chương trình thoáng, dễ đọc.

+ Dòng thứ 4: void main (void) là thành phần chính của chương trình C (bạn có thể viết main() hoặc void main() hoặc main(void)). Tuy nhiên bạn nên viết void main() thì chương trình sẽ rõ ràng hơn. Mọi chương trình C đều bắt đầu thi hành từ hàm main(). Cặp dấu ngoặc () cho biết đay là khối hàm (function). Hàm void main(void) có từ khóa void đầu tiên cho biết hàm này không trả về giá trị, từ khóa void trong dấu ngoặc đơn cho biết hàm này không nhận vào đối số.

+ Dòng thứ 4, 7: cặp dấu ngoặc nhọn {} là giới hạn thân của hàm. Thân hàm bắt đầu bằng dấu { và kết thúc bằng dấu }.

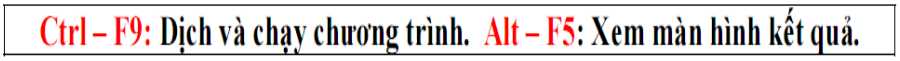
+ Dòng thứ 6: printf("Bai hoc C dau tien.");, chỉ thị cho máy in ra chuỗi ký tự nằm trong dấu nháy kép (""). Hàng này được gọi là một câu lệnh, kết thúc câu lệnh C phải là dấu chấm phẩy (;).

↳ Chú ý:

- Các từ include, stdio.h, void, main, printf phải viết bằng chữ thường.
- Chuỗi nháy kép cần in ra bạn có thể viết chữ hoa hay chữ thường tùy ý.

+ Kết thúc câu lệnh phải có dấu chấm phẩy.
+ Kết thúc tên hàm không có dấu chấm phẩy hay bất cứ dấu gì.
+ Ghi chú phải đặt trong dấu /*…*/.
+ Thân hàm phải được bao trong cặp {}.
+ Các câu lệnh trong thân hàm phải viết thụt vào.

➢Bạn nhập đoạn chương trình trên vào máy. Dịch, chạy và quan sát kết quả.



Hình 11: Màn hình dịch, chạy chương trình

➢ Sau khi nhập đoạn chương trình vào máy.Ấn giữ phím Ctrl, gõ F9 để dịch và chạy chương trình. Khi đó bạ thấy chương trình chớp rất nhanh và không có kết quả gì cả. Bạn ấn giữ phím Ctrl, gõ F5 để xem kết quả, khi xem xong, ấn phím bất kỳ để quay về màn hình soạn thảo chương trình.

➢Bây giờ bạn sửa lại dòng thứ 6 bằng câu lệnh printf("Bai hoc C dau tien.\n");, sau đó dịch, chạy chương trình và quan sát kết quả.

→ Kết quả in ra màn hình.

Hình 12: Màn hình kết quả chương trình

Ở dòng bạn vừa sửa có them \n, \n là ký hiệu xuống dòng sử dụng trong lệnh printf. Sau đây là một số ký hiệu khác:

+ ***Các kí tư điều khiển***:

\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.

+ ***Các kí tư đặc biệt***:

\\ : In ra dấu \
\" : In ra dấu "
\' : In ra dấu '

Hình 13: Một số ký hiệu khác

➢ Bây giờ bạn sửa lại dòng thứ 6 bằng câu lệnh printf("\tBai hoc C dau tien.\n"); sau đó dịch, chạy chương trình và quan sát kết quả.
→ Kết quả in ra màn hình.

```
        Bai hoc C dau tien.
_
```

Hình 14: Màn hình kết quả chương trình

Khi chạy chương trình bạn nghe tiếng bip phát ra từ loa.
⇨ Mỗi khi chạy chương trình bạn cảm thấy bất tiện trong việc xem xét kết quả phải ấn tổ hợp phím Alt-F5. Để khắc phục tình trạng này bạn sửa lại chương trình như sau:

```
Dòng  File  Edit  Search  Run  Compile  Debug  Project  Option  Window  Help
   1  /* Chuong trinh in ra cau bai hoc C dau tien */
   2  #include <stdio.h>
   3  #include <conio.h>
   4
   5  void main(void)
   6  {
   7     printf("\t\tBai hoc C \rdau tien.\n");
   8     getch();
   9  }
      F1 Help  Alt-F8 Next Msg  Alt-F7 Prev Msg  Alt - F9 Compile  F9 Make  F10 Menu
```

Hình 15: Màn hình chương trình

→ Kết quả in ra màn hình
dau tien Baihoc C.

+ Dòng thứ 3: chứa phát biểu tiền xử lý #include <conio.h>.Vì chương trình này ta sử dụng hàm thư viện của C là getch. Do đó bạn cần phải khai báo hàm thư viện này để báo cho chương trình dịch C biết. Nếu không khai báo chương trình sẽ thông báo lỗi.

+ Dòng thứ 8: getch();, chờ nhận một ký tự bất kỳ từ bàn phím, nhưng không in ra màn hình. Vì thế ta sử dụng hàm này để khi ta chạy chương trình xong sẽ dừng lại ở màn hình kết quả, sau đó ta ấn phím bất kỳ để quay về màn hình soạn thảo.

+ Bạn nhập đoạn chương trình trên vào máy chạy và quan sát kết quả.

Ví dụ 2:

Dòng	File Edit Search Run Compile Debug Project Option Window Help
1	/* Chuong trinh nhap va in ra man hinh gia tri bien*/
2	#include <stdio.h>
3	#include <conio.h>
4	
5	void main(void)
6	{
7	int i;
8	printf("Nhap vao mot so: ");
9	scanf("%d", &i);
10	printf("So ban vua nhap la: %d.\n", i);
11	getch();
12	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

Hình 16: Màn hình chương trình

☞ ***Kết quả in ra màn hình***

```
Nhap vao mot so: 15
So ban vua nhap la: 15.

_
```

Hình 17: Màn hình kết quả chương trình

+ Dòng thứ 7: int i; là lệnh khai báo, mẫu tự I gọi là tên biến. Biến là một vị trí trong bộ nhớ dùng để lưu trữ giá trị nào đó mà chương trình sẽ lấy để sử dụng. Mỗi biến thuộc một kiểu dữ liệu. Trong trường hợp này ta sử dụng biến i kiểu số nguyên (integer) viết tắt là int.

+ Dòng thứ 9: scanf("%d", &i). Sử dụng hàm scanf để nhận từ người sử dụng một trị nào đó. Hàm scanf trên có hai đối mục. Đối mục "%d" được gọi là chuỗi định dạng, cho biết loại dữ kiện mà người sử dụng sẽ nhập vào. Chẳng hạn ở đây phải nhập vào là số nguyên. Đối mục thứ 2 là &i có dấu & đi đầu gọi là address operator, dấu & phối hợp với tên biến cho hàm scanf biến đem trị gõ từ bàn phím lưu vào biến i.

+ Dòng thứ 10: printf("So ban vua nhap la: %d.\n", i);. Hàm này có 2 đối mục. Đối mục thứ 1 là chuỗi định dạng có chứa chuỗi văn bản So ban vua nhap la: và % d (ký hiệu khai báo chuyển đổi dạng thức) cho biết số nguyên sẽ được in ra. Đối mục thứ 2 là i cho biết giá trị lấy từ biến i để in ra màn hình.

→ Bạn nhập đoạn trên vào máy. Dịch, chạy và quabn sát kết quả.

Ví dụ 3:

Dòng	File Edit Search Run Compile Debug Project Option Window Help
1	/* Chuong trinh nhap vao 2 so a, b in ra tong*/
2	#include <stdio.h>
3	#include <conio.h>
4	
5	void main(void)
6	{
7	int a, b;
8	printf("Nhap vao so a: ");
9	scanf("%d", &a);
10	printf("Nhap vao so b: ");
11	scanf("%d", &b);
12	printf("Tong cua 2 so %d va %d la %d.\n", a, b, a+b);
13	getch();
14	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

Hình 18: Màn hình chương trình

→Kết quả in ra màn hình:

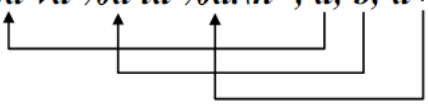
```
Nhap vao so a: 4
Nhap vao so b: 14
Tong cua 2 so 4 va 14 la 18.

_
```

Hình 19: Màn hình kết quả chương trình

+ Dòng thứ 12:

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



→ Nhập đoạn chương trình trên vào máy. Dịch, chạy và quan sát kết quả

Ví dụ 4:

```
Dòng | File  Edit  Search  Run  Compile  Debug  Project  Option  Window  Help
   1 | /* Chuong trinh nhap vao ban kinh hinh tron. Tinh dien tich */
   2 | #include <stdio.h>
   3 | #include <conio.h>
   4 |
   5 | #define PI    3.14
   6 |
   7 | void main(void)
   8 | {
   9 |    float fR;
  10 |    printf("Nhap vao ban kinh hinh tron: ");
  11 |    scanf("%f", &fR);
  12 |    printf("Dien tich hinh tron: %.2f.\n", 2*PI*fR);
  13 |    getch();
  14 | }
     | F1 Help  Alt-F8 Next Msg  Alt-F7 Prev Msg  Alt - F9 Compile  F9 Make  F10 Menu
```

Hình 20: Màn hình chương trình

→ Kết quả in ra màn hình:

```
Nhap vao ban kinh hinh tron: 1
Dien tich hinh tron: 6.28

_
```

Hình 21: Màn hình kết quả chương trình

+ Dòng thứ 5: #define PI 3.14, dùng chỉ thị define để định nghĩa hằng số PI có giá trị 3.14. Trước define phải có dấu # và cuối dòng không có dấu chấm phẩy.

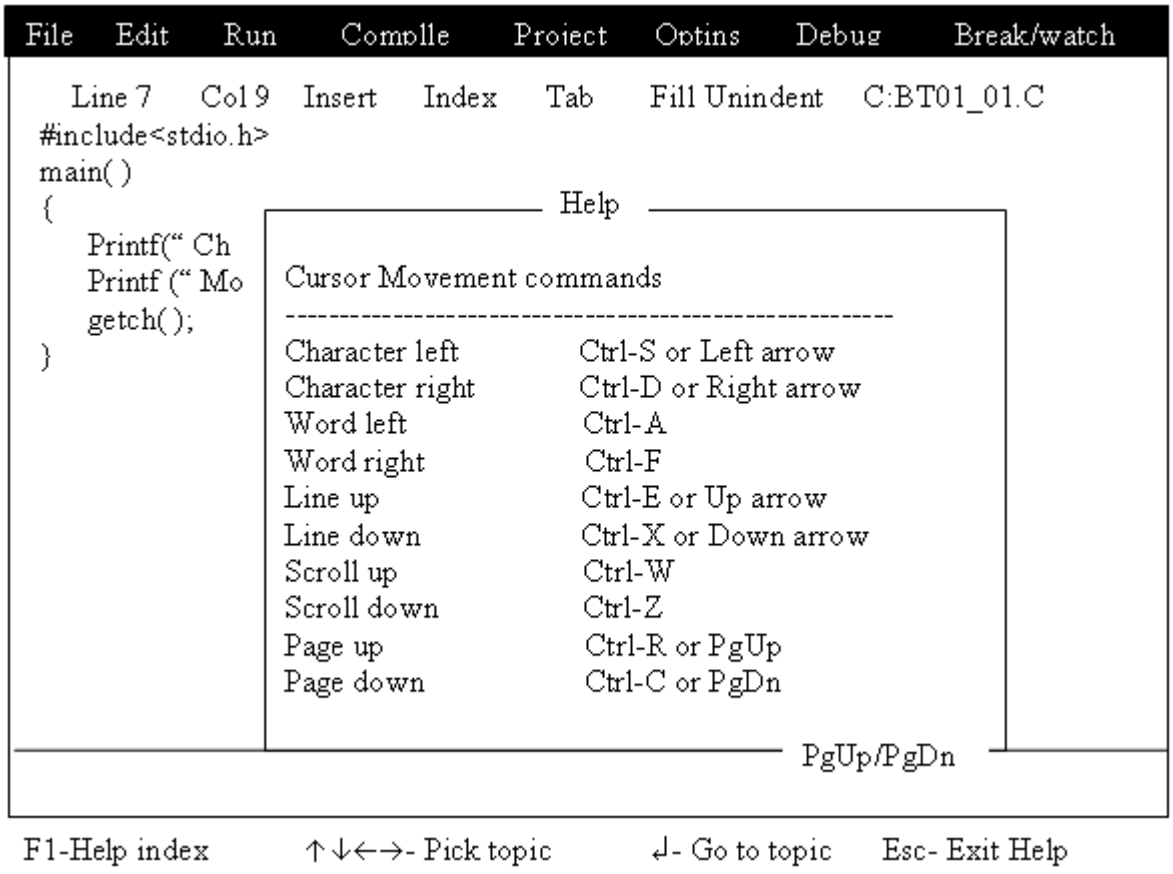
+ Dòng thứ 12: printf("Dien tich hinh tron: %.2f.\n", 2*PI*fR);. Hàm này có 2 đối mục.Đối mục thứ 1 là một chuỗi định dạng có chứa chuỗi văn bản Dien tich hinh tron: và %.2f (ký hiệu khai báo chuyển đổi dạng thức) cho biết dạng số chấm động sẽ được in ra. Trong đó .2 có nghĩa là in ra 2 số lẽ. Đối mục thứ 2 là biểu thức hằng 2*PI*fR;

→ Nhập đoạn chương trình trên vào máy. Dịch, chạy và quan sát kết.

3. Cách sử dụng sự trợ giúp từ helpfile về cú pháp lệnh, về cú pháp hàm, các chương trình mẫu.

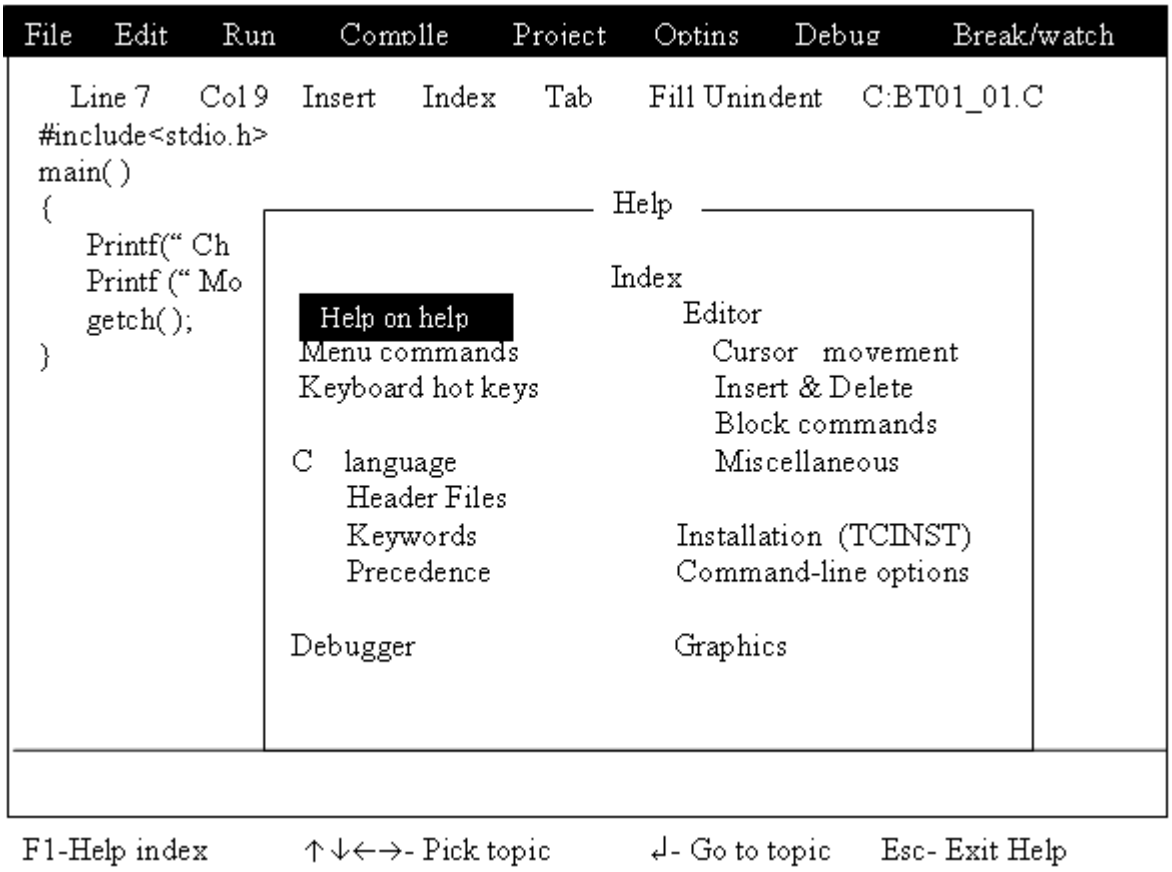
Tương tự như các ngôn ngữ lập trình khác, Turbo C cũng có một hệ thống trợ giúp trong việc học tập sử dụng ngôn ngữ này.

Ở bất kỳ trạng thái làm việc nào, nếu bạn bấm phím **<F1>,** Turbo C sẽ hiển thị màn hình giúp đỡ (Help) tương ứng. Ví dụ bạn đang soạn thảo chương trình, nếu bấm phím **<F1>** sẽ xuất hiện một cửa sổ như hình dưới đây.



Hình 22: Màn hình Help

Muốn hệ thống chỉ mục Help cua Turbo C, bạn bấm tiếp phím **<F1>** sẽ hiển thị ra màn hình sau:



Hình 23: Màn hình Help on help

Bạn có thể thực hiện theo sự hướng dẫn hiển thị trên dòng trạng thái để trang cứu những thông tin cần thiết đối với bạn hoặc giúp bạn học tập cách sử dụng các lệnh, các hàm, các loại toán tủ, các thao tác bằng các phím gõ tắt v.v…
Muốn thoát khỏi môi trường help, bạn chỉ cần bấm phím **<Ecs>.**

Bạn cũng có thể học tập cách sử dụng hệ thống giúp đỡ (Help) của Turbo C mà không cần khởi động Turbo C, có nghĩa là ngay tại dấu nhắc của hệ điều hành MS-DOS bạn vẫn có thẻ tìm hiểu về Turbo C được.

Muốn vậy, bạn gõ vào lệnh sau đây từ dấu nhắc của MS-DOS.

c:\TC>**THELP** ↲

Turbo C hiển thị thông báo sau:

Turbo Help version 1.0 Copyright (C) 1988
Borland International.
To active THELP, press 5 on the numeric keypad

Bạn bấm phím số 5 bên phía bàn phím số (**numeric keypad** ở bên phải của bàn phím), trên màn hình hiển thị giống như hình trên.

Muốn gỡ bỏ trình thường trú **THELP.COM** ra khỏi bộ nhớ, bạn gõ vào lệnh sau đây:

c:\TC>THELP/U ↲

Turbo C hiển thị thông báo.

Turbo Help version 1.0 Copyright (C) 1988
Borland International.
THelp has been removed from memory.

CHƯƠNG 2: CÁC THÀNH PHẦN CƠ BẢN

Mã chương: MĐ10-02

Mục tiêu:

Sau khi học xong chương này sinh viên có khả năng:

- Hiểu và sử dụng được hệ thống kí hiệu và từ khóa.
- Hiểu được các kiểu dữ liệu.
- Hiểu được và vận dụng được các loại biến, hằng biểu thức cho từng chương trình cụ thể.
- Biết, hiểu và so sánh được các lệnh, khối lệnh.
- Thực hiện được việc chạy chương trình.

1. Hệ thống từ khóa và kí hiệu được dùng trong C.

1.1. Bộ chữ viết trong C:

Bao gồm những ký tự, những ký hiệu sau (phân biệt chữ hoa và chữ thường):

26 chữ cái la tinh lớn A,B,C...Z

26 chữ cái la tinh nhỏ a,b,c ...z.

10 chữ số thập phân 0,1,2...9.(chữ số Ả Rập)

Các ký hiệu toán học: +, -, *, /, =, <, >, (,)

Các ký hiệu đặc biệt: :. , ; " ' _ @ # $! ^ [ ] { } ...

Dấu cách hay khoảng trống.

1.2. Các từ khóa trong C:

Từ khóa là các từ dành riêng (reserved words) của C mà người lập trình có thể sử dụng nó trong chương trình tùy theo ý nghĩa của từng từ. Dùng để khai báo dữ liệu, viết câu lệnh.... Ta không được dùng từ khóa để đặt cho các tên riêng của mình. Các từ khóa của Turbo C 3.0 bao gồm:

asm	auto	break	case	cdecl	char
class	const	continue	_cs	default	delete
do	double	_ds	else	enum	_es
extern	_export	far	_fastcall	float	for
friend	goto	huge	if	inline	int
interrupt	_loadds	long	near	new	operator
pascal	private	protected	public	register	return
_saveregs	_seg	short	signed	sizeof	_ss
static	struct	switch	template	this	typedef
union	unsigned	virtual	void	volatile	while

Khác với ngôn ngữ khác, các từ khóa của Turbo C đều viết bằng chữ thường.

Ví dụ: phải viết là if, không được viết là If hoặc IF như Pascal hoặc Foxpro.

1.3. Tên:

Khái niệm tên 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 thường đặt cho hằng, biến, mảng, con trỏ, nhãn,… Chiều dài tối đa của biến 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 tiên phải là chữ hoặc dấu gạch dưới. Khi đặt tên không được trùng từ khóa.

VD1:

↳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 phân biệt chữ hoa chữ thường.

VD2:

number khác Number

case khác Case

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

1.4. Cặp dấu ghi chú thích:

Khi viết chương trình có một vài trường hợp ta phải có một vài lời ghi chú về một đoạn chương trình nào đó để ghi nhớ, dễ hiểu, dễ hiệu chỉnh sau này. Ghi chú thì không thuộc về nội dung của chương trình, khi biên dịch sẽ bị bỏ qua, nội dung sẽ ghi trong cặp dấu chú thích /* và */.

```
#include<stdio.h>
main ()
{
                    /*Bắt đầu khối lệnh*/
                    ....
} /*Kết thúc khối lệnh*/
```

Nếu bạn không viết đúng quy định trên, khi chạy chương trình sẽ bị lỗi.

Ví dụ:

```
#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;
}
```

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 ghi chú một hàng và dạng /* …. */ có thể ghi chú một hàng hoặc nhiều hàng.

1.5. Các ký tự điều khiển:

Turbo C có nhiều ký tự điều khiển việc xuất dữ liệu ra màn hình hoặc máy in hơn hẳn các ngôn ngữ lập trình khác. Bộ ký tự điều khiển của Turbo C bao gồm:

Ký tự điều khiển	Giá trị	Ký tự được hiểu	Tác dụng

\a	0x07	BEL	Phát tiếng còi
\b	0x08	BS	Xóa ký tự bên trái
\f	0x0C	FF	Sang trang
\n	0x0A	LF	Xuống dòng
\r	0x0D	CR	Trở về đầu dòng
\t	0x09	HT	Tab theo cột
\v	0x0B	VT	Tab theo hang
\\	0x05	\	Dấu Backslash
\’	0x2C	‘	Dấu nháy đơn (‘)
\”	0x22	“	Dấu nháy kép (“)
\?	0x3F	?	Dấu chấm hỏi (?)
\ddd	ddd	Ký tự có giá trị theo mã ASCII	cơ số 8 là ddd trong bảng
\xHHH	0xHHH	Ký tự có giá trị theo mã ASCII	cơ số 16 là HHH trong bảng

Bảng 1: Bộ ký tự điều khiển của Turbo C

Ví dụ:

```
#include<stdio.h>
    main()
      {
   Printf(“\nDay la chuong trinh \’ Turbo C\’”);
     …
     …
     …
     }
```

Kết quả sẽ in ra:

Day là chuong trinh ‘Turbo C’

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

2.1. Kiểu số nguyên:

Kiểu số nguyên là kiểu dữ liệu dùng lưu các giá trị nguyên hay còn gọi là kiểu đếm được. Kiểu số nguyên trong C được chia thành các kiểu dữ liệu con, mõi kiểu có một miền giá trị khác nhau.

2.1.1. Kiểu số nguyên 1 byte (8 bits):

Gồm có các kiểu sau:

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

Bảng 2: Kiểu dữ liệu số nguyên 1 byte (8 bits)

Kiểu unsigned char: lưu các số nguyên dương từ 0 đến 255.

- Để khai báo một biến là kiểu ký tự thì ta khai báo biến kiểu unsigned char. Mỗi số trong miền giá trị của kiểu unsigned char tương ứng với một ký tự trong bảng mã ASCII .

Kiểu char: lưu các số nguyên từ -128 đến 127. Kiểu char sử dụng bit trái nhất làm bit dấu.

Nếu gán giá trị > 127 cho biến kiểu char thì giá trị của biến này có thể là số âm.

Ví dụ:

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

2.1.2. Kiểu số nguyên 2 bytes (16 bits):

Gồm có 4 kiểu sau:

STT	Kiểu dữ liệu	Miền giá trị (domain)
1	Enum	Từ -32,768 đến 32,767
2	unsigned int	Từ 0 đến 65,535
3	Short int	Từ -32,768 đến 32,767
4	Int	Từ -32,768 đến 32,767

Bảng 3: Kiểu dữ liệu số nguyên 2 bytes (16 bits)

Kiểu enum, short int, int : Lưu các số nguyên từ -32768 đến 32767. Sử dụng bit bên trái nhất làm bit dấu.

=> Nếu gán giá trị >32767 cho biến có 1 trong 3 kiểu trên thì giá trị của biến này có thể là số âm.

Kiểu unsigned int: Kiểu unsigned int lưu các số nguyên dương từ 0 đến 65535.

2.1.3. Kiểu số nguyên 4 bytes (32 bit):

Kiểu số nguyên 4 bytes hay còn gọi là số nguyên dài (long) gồm có 2 kiểu sau:

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	unsigned long	Từ 0 đên 4,294,967,295
2	Long	Từ -2,147,483,648 đên 2,147,483,647

Bảng 4: Kiểu dữ liệu số nguyên 4 bytes (32 bits)

Kiểu long: Lưu tất cả các số nguyên từ -2147483648 đên 2147483647. Sử dụng bit bên trái nhất để làm bit dấu.

=> Nếu gán giá trị > 2147483647 cho biến có kiểu long thì giá trị biến này có thể là số âm.

Kiểu unsigned long: Lưu các số nguyên dương từ 0 đên 4294967295.

2.2. Kiểu số thực:

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

STT	Kiểu dữ liệu	Kích thước (Size)	Miền giá trị (domain)
1	float	4 bytes	Từ $3.4 * 10^{-38}$ đến $3.4 * 10^{38}$
2	Double	8 bytes	Từ $1.7 * 10^{-308}$ đến $1.7 * 10^{308}$
3	long double	10 bytes	Từ $3.4 * 10^{-4932}$ đến $3.4 * 10^{4932}$

Bảng 5: Kiểu dữ liệu số thực

Mỗi kiểu số thực trên đều có miền giá trị và độ chính xác (số số lẻ) khác nhau. Tùy vào nhu cầu sử dụng mà ta có thể khai báo biến thuộc 1 trong 3 kiểu trên.

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

3. Các loại biến, cách khai báo, sử dụng.

3.1. Biến:

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

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

❄ Tên biến:

Muốn biến sử dụng trong chương trình đều phải đặt tên theo các quy định như sa:

⇨ Tên biến hợp lệ là một chuỗi ký tự liên tục gồm: Ký tự, chữ, số, dấu gạch dưới. Ký tự đầu tiên ,phải là chữ. Khi đăt tên biến không đặt trùng với các từ khóa của Turbo C.

Ví dụ:

bien_1, bien_2, temp22 → hợp lê.

bi&, 2ten, A b, float → không hợp lệ.

⇨ Turbo C phân biệt rất rõ rang giữa chữ in thường và in hoa.

Ví dụ:

BIEN, Bien, bieN: Là những tên biến khác nhau.

If, Goto : Tuy trùng với từ khóa nhưng vẫn hợp lệ.

- Turbo C chỉ phân biệt hai tên hợp lệ với nhau với n ký tự đầu tiên của chúng, (n=8 đến 32 ký tự). Nhu vậy bạn có thể đặt tên biến hợp lệ một cách thoải mái, tuy nhiên tên biến càng dài thì càng khó nhớ.

Ví dụ : Hai tên biến sau đây được coi là hợp lệ và cùng tên nhau :

- Ten_bien_dai_toi_32_ky_tu_dau_tien_1
- Ten_bien_dai_toi_32_ky_tu_dau_tien_2

3.1.1. Cú pháp khai báo biến:

<**Kiểu dữ liệu**> *Danh sách các tên biến;*

Kiểu dữ liệu: một trong các kiểu ở mục

Danh sách các tên biến: gồm các tên biến có cùng kiểu dữ liệu, mỗi tên biến cách nhau dấu phẩy (,), cuối cùng là dâu chấm phẩy (;).

VD1:

```
int   a, b, c;        /*Ba biến a, b,c có kiểu int*/
long int   chu_vi;    /*Biến chu_vi có kiểu long*/
float    nua_chu_vi;  /*Biến nua_chu_vi có kiểu float*/
double  dien_tich;    /*Biến dien_tich có kiểu double*/
```

↳Khi khai báo biến nên đặt theo **quy tắc Hungarian Notation**

VD2:

```
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 tiên biến tuoi 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 di chuyển đến phần khai báo mới biết kiểu của biến này. Tương tự cho biến fTrongluong, bạn nhìn vào và biết ngay biến này có kiểu float.

❄ **Chú ý :**

Việc khai báo biến giữa Turbo C và Pascal có sự khác nhau. Vì vậy đối với những bạn đã học lập trình Pascal khi học Turbo C thường hay bị nhầm, do đó chúng tôi xin lưu ý để tránh nhầm lẫn.

Đối với Turbo C	Đối với Pascal

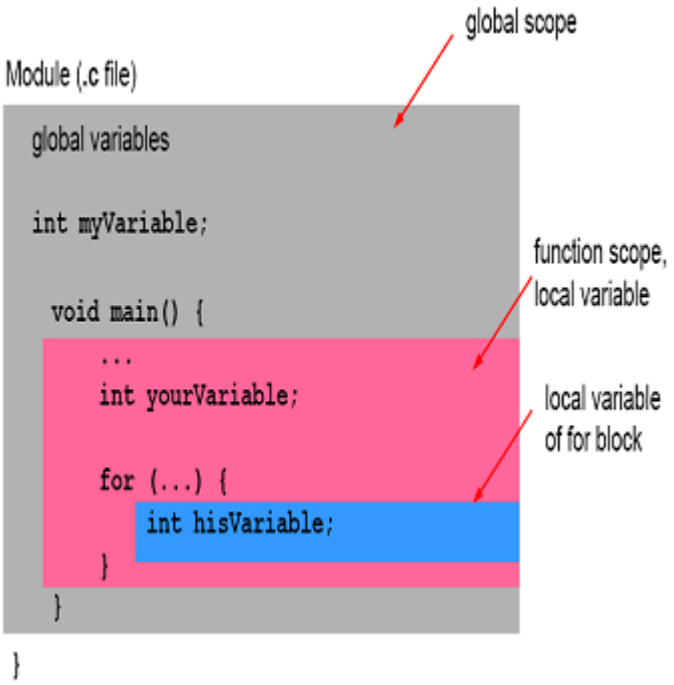
Kiểu *DS biến* ;
Ví dụ :
#include<stdio.h>
main()
{
int a,b,c;
…
}
	DS biến : **Kiểu** ;
	Ví dụ :
	var
	a,b,c :integer ;
	
	begin
	...
	}

Bảng 6: Việc khai báo biến giữa Turbo C và Pascal có sự khác nhau

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

✓ **Khai báo biến ngoài:**

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



Hình 1: Khai báo biến ngoài

Ví dụ 1:

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

Ví dụ 2 :

```
int n, tong ;/*Khai báo biến ngoài*/
main()
{
n = tong  + 10 ;
...
}
```

✓ **Khai báo biến trong:**

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

Ví dụ 1:

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

```
int bienngoai;    /*khai bao bien ngoai*/
int  main ()
{   int j,i; /*khai bao bien ben trong chuong trinh chinh*/
 clrscr();
 i=1; j=2;
 bienngoai=3;
printf("\n Gia  tri cua i la %d",i);
 /*%d là s nguyên, s bit sau */
  printf("\n Gia tri cua j la %d",j);
  printf("\n Gia tri cua bienngoai la %d",bienngoai);
 getch();
 return 0;
}
```

Ví dụ 2:

```
#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;
}
```

3.1.3. Khai báo đối số hàm:

Vị trí biến đặt trong phần định nghĩa tham số của hàm.

Ví dụ:

```
fun2(x,c)
int x; /*Khai báo đối số hàm*/
char c;
{
                x=x-1;
                ...
}
```

❄ **Chú ý:**

Khi khai báo biến trong bạn cần đặt biến ở ngay sau dấu { đầu tiên của thân hàm và cần phải đứng trước các câu lệnh khác. Như vậy, sau một lệnh khác (chẳng hạn như câu lệnh gán) thì không được khai báo biến nữa, nếu không Turbo c sẽ báo sai như đoạn chương trình dưới đây.

Ví dụ:

```
/*Khai báo sai vị trí biến*/
main()
{
int a,b,c;/*khai báo đúng*/
a=35;
int d;/*khai bao sai vi sau cau lenh gan*/
}
```

Sự khai báo biến ở các vị trí khác nhau trong chương trình thì sẽ có ý nghĩa và giá trị khác nhau, bạn phải hết sức lưu ý vấn đề này để khỏi mất sai lầm đáng tiếc.

Cũng do có đặc điểm này, bạn có thể lợi dụng để đặt trùng tên cho các biến sẽ tiết kiệm được bộ nhớ rất nhiều mà không sợ bị sai các giá trị của các biến. Muốn hiểu rõ tác dụng này, bạn hãy nhập và chạy thử đoạn chương trình dưới đây.

Ví dụ:

```
/*Mô tả sự khác nhau về vi trí các biến*/
#include<stdio.h>
main()
{
int i = 3, j= 5;
{
int i = 2, j = 4;
/*Các biến này chỉ tồn tại trong khối*/
printf("KET QUA IN RA\n");
printf("Gia tri i  trong khoi = % d\n",i);
printf("Gia tri j  trong khoi = % d\n",j);
}
printf("Gia tri i ngoai khoi = %d\n",i);
printf("Gia tri j ngoai khoi = %d\n"j);
getch();
}
```

→ Sau khi chay chương trinh ma se co ket qua nhu sau:

KET QUA IN RA

- Gia tri i trong khoi = 2
- Gia tri j trong khoi = 2
- Gia tri i ngoai khoi = 3
- Gia tri j ngoai khoi = 5

3.2. 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.

VD:

↳ Khai báo trước, gán trị sau:

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

↳ Vừa khai báo vừa gán trị:

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

3.3. Biểu thức:

Biểu thức là một sự kết hợp giữa các toán tử (operator) và các toán hạng (operand) theo đúng một trật tự nhất định.

- Mỗi toán hạng có thể là một hằng, một biến hoặc một biểu thức khác.
- Trong trường hợp, biểu thức có nhiều toán tử, ta dùng cặp dấu ngoặc đơn () để chỉ định toán tử nào được thực hiện trước.

Ví dụ 1:

Biểu thức nghiệm của phương trình bậc hai:

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

Trong đó 2 là hằng; a, b, Delta là biến.

Ví dụ 2 :

*a = (i = 10 * a) + sin (x = b + 1) * (b>1) :*

hoặc

x = y / 3 + power (i,2);

3.3.1. Các toán tử số học:

Trong ngôn ng C, các toán tử +, -, *, / làm việc tương tự như khi chúng làm việc trong các ngôn ngữ khác. Ta có thể áp dụng chúng cho đa số kiểu dữ liệu có sẵn được cho phép bởi C. Khi ta áp dụng phép / cho một số nguyên hay một ký tự, bất kỳ phần dư nào cũng bị cắt bỏ. Chẳng hạn, 5/2 bằng 2 trong phép chia nguyên.

Toán tử	Ý nghĩa
+	Cộng
-	Trừ
*	Nhân
/	Chia
%	Chia lấy phần dư
--	Giảm 1 đơn vị
++	Tăng 1 đơn vị

Bảng 7: Các toán tử số học

Tăng và giảm (++ & --)

Toán tử ++ thêm 1 vào toán hạng của nó và – trừ bớt 1. Nói cách khác:

x = x + 1 giống như ++x

x = x – 1 giống như x—

Cả 2 toán tử tăng và giảm đều có thể tiền tố (đặt trước) hay hậu tố (đặt sau) toán hạng. Ví dụ: x = x + 1 có thể viết x++ (hay ++x)

Tuy nhiên giữa tiền tố và hậu tố có sự khác biệt khi sử dụng trong 1 biểu thức. Khi 1 toán tử tăng hay giảm đứng trước toán hạng của nó, C thực hiện việc tăng hay giảm trước khi lấy giá trị dùng trong biểu thức. Nếu toán tử đi sau toán hạng, C lấy giá trị toán hạng trước khi tăng hay giảm nó. Tóm lại:

```
x = 10
y = ++x  //y = 11
```

Tuy nhiên:

```
x = 10
x = x++ //y = 10
```

Thứ tự ưu tiên của các toán tử số học:

++ -- sau đó là * / % rồi mới đến + -

3.3.2. Các toán tử quan hệ và các toán tử Logic:

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

Toán tử	Ý nghĩa
Các toán tử quan hệ	
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
==	Bằng
!=	Khác
Các toán tử Logic	
&&	AND
||	OR
!	NOT

Bảng 8: Các toán tử quan hệ và các toán tử Logic

Ví dụ:

Operator	Meaning	Example
==	equal	x == 3
!=	not equal	x != y
>	greater	4 > 3
<	less	x < 3
>=	greater or equal	x >= y
<=	less or equal	x <= y

Hình 2: Ví dụ các toán tử quan hệ

P	Q	P&&q	P||q	!p
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Bảng 9: Bảng chân trị cho các toán tử Logic:

Các toán tử quan hệ và logic đều có độ ưu tiên thấp hơn các toán tử số học. Do đó một biểu thức như: *10 > 1+12 sẽ đươc xem là 10 > (1+12) và kết quả là sai (0).*

Ta có thể kết hợp vài toán tử lại với nhau thành biểu thức như sau:

10>5&&!(10<9)||3<=4 Kết quả là đúng.

Thứ tự ưu tiên của các toán tử quan hệ là Logic

Cao nhất: !

> >= < =

== !=

&&

Thấp nhất: ||

3.3.3. Các toán tử Bitwise:

Các toán tử Bitwise ý nói đến kiểm tra, gán hay sự thay đổi các Bit thật sự trong 1 Byte của Word, mà trong C chuẩn là các kiểu dữ liệu và biến char, int. Ta không thể sử dụng các toán tử Bitwise với dữ liệu thuộc các kiểu float, double, long double, void hay các kiểu phức tạp khác.

Toán tử	Ý nghĩa
&	AND
|	OR
^	XOR
~	NOT
>>	Dịch phải
<<	Dịch trái

Bảng 10: Các toán tử

Bảng chân trị của toán tử ^ (XOR)

P	Q	p&q
0	0	0
0	1	1
1	0	1
1	1	0

Bảng 11: Bảng chân trị

3.3.4. Toán tử ? cùng với:

C có một toán tử rất mạnh và thích hợp thay thế cho các câu lệnh của If-Then-Else. Cú pháp của việc sử dụng toán tử ? là:

E1 ? E2 : E3

Trong đó E1, E2, E3 là các biểu thức.

Ý nghĩa: Trước tiên E1 được ước lượng, nếu đúng E2 được ước lượng và nó trở thành giá trị của biểu thức; nếu E1 sai, E2 được ước lượng và trở thành giá trị của biểu thức.

Ví dụ:

X = 10

Y = X > 9 ? 100 : 200

Thì Y được gán giá trị 100, nếu X nhỏ hơn 9 thì Y sẽ nhận giá trị là 200. Đoạn mã này tương đương cấu trúc if như sau:

X = 10

if (X < 9) Y = 100

else Y = 200

*3.3.5. Toán tử con trỏ & và *:*

Một con trỏ là địa chỉ trong bộ nhớ của một biến. Một biến con trỏ là một biến được khai báo riêng để chứa một con trỏ đến một đối tượng của kiểu đã chỉ ra nó. Chúng ta sẽ tìm hiểu kỹ hơn về con trỏ trong chương về con trỏ. Ở đây, chúng ta sẽ đề cập ngắn gọn đến hai toán tử được sử dụng để thao tác với các con trỏ.

Toán tử thứ nhất là &, là một toán tử quy ước trả về địa chỉ bộ nhớ của hệ số của nó.

Ví dụ:

Đặt vào biến m địa chỉ bộ nhớ của biến count.

Chẳng hạn, biến count vị trí bộ nhớ 2000, giả sử count có giá trị là100. Sau câu lệnh trên m sẽ nhận giá trị 2000.

*Toán tử thứ hai là *, là một bổ sung cho &; đây là một toán tử quy ước trả về giá trị của biến được cấp phát tại địa chỉ theo sau đó.*

Ví dụ:

*q = *m*

Sẽ đặt giá trị của count vào q. Bây giờ q sẽ có giá trị là 100 vì 100 được lưu trữ tại địa chỉ 2000.

3.3.6. Toán tử dấu phẩy:

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

Ví dụ:

x = (y=3,y+1);

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

3.3.7. Xem các dấu ngoặc đơn và các cặp dấu ngoặc vuông là toán tử:

Trong C, cặp dấu ngoặc đơn là toán tử tăng độ ưu tiên của các biểu thức bên trong nó.

Các cặp dấu ngoặc vuông thực hiện thao tác truy xuất phần tử trong mảng.

3.3.8. Tổng kết về độ ưu tiên:

Cao nhất	() []
	! ~ ++ -- (Kiểu) * &
	* / %
	+ -
	<< >>
	< <= > >=
	&
	^
	|
	&&
	||
	?:
	= += -= *= /=
Thấp nhất	,

Bảng 12: độ ưu tiên toán tử

3.3.9. Cách viết tắt trong C:

Có nhiều phép gán khác nhau, đôi khi ta có thể sử dụng viết tắt trong C nữa.

Chẳng hạn:

x = x + 10 được viết thành x +=10

Toán tử += báo cho chương trình dịch biết tăng giá trị của x lên 10.

Cách viết này làm việc trên tất cả các toán tử nhị phân (phép toán hai ngôi) của ngôn ngữ C.

Tổng quát:

(Biến) = **(Biến)** **(Toán tử)** **(Biểu thức)**

Có thể được viết:

(Biến) **(Toán tử)**= **(Biểu thức)**

3.3.10. Hằng:

Hằng là những giá trị biểu thức cố định được đặt vào chương trình nguồn và không bị thay đổi khi thực hiện chương trình.

Trong Turbo C mỗi hằng hoặc biểu thức hằng đều có một trị và kiểu dữ liệu của chúng.

✓ **Hằng số:**

Hằng số là một số xác định, có thể là số nguyên hay số thực

- **Hằng số thực:**

Hằng số thực bao gồm các số thuộc kiểu **float** và **double** được thể hiện theo hai cách:

- Cách 1 (Ký pháp tự nhiên): Số được viết theo cách thông thường chỉ khác là dấu phẩy thập phân (mà ta quen gọi) được thay bằng dấu chấm.

Ví dụ: 123.45 -678.90 123.0

- Cách 2 (Ký pháp số mũ hoặc khoa học): Số được tách thành hai phần là:

+ Phần định trị: Là số nguyên hoặc số thực được viết theo ký pháp tự nhiên.

+ Phần đặc tính (hay phần bậc): Là số nguyên.

Hai phần này cách nhau bởi ký tự e hoặc E.

Ví dụ:

123.456e – 4 = 0.0123456

0.12E3 = 120

-49.5e – 2 = -0.495

1E8 = 100000000

- **Hằng int:**

Được viết theo kiểu

-48953L hoặc -489351 (thêm L hoặc 1 vào sau số).

Một số nguyên nằm ngoài phạm vi của **int** cũng được coi là một hằng **long**.

45678L và 45678

Là hai hằng **long** có cùng giá trị.

- **Hằng nguyên (int) hệ 8:**

Hằng này được viết theo cách 0X1X2X3...

Trong đó **xi** là một số nguyên nằm trong khoảng từ 1 đến 7. Hằng nguyên hệ 8 luôn luôn nhận giá trị dương.

Ví dụ:

$0345_{(Hệ\ 8)}$ = $229_{(Hệ\ 10)}$

Ta có:

$3 * 8^2 + 4 * 8^1 + 5 * 8^0 = 3 * 64 + 4 * 8 + 5 * 1$

$= 192 + 32 + 5$

$= 229$

- **Hằng nguyên hệ 16:**

Hệ 16 có 16 ký số được quy định như sau:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Hằng số hệ 16 được viết theo cách:

0Xx1x2x3 ... hoặc 0xx1x2x3...

Trong đó. Xi hoặc là một chữ số (từ 0 đến 9) hoặc là một chữ cái từ a(A) đến f(F) (Từ 0 đến 15).

Ví dụ: 0xa9, 0Xa9, 0xA9, 0XA9

Là 4 hằng số hệ 16 có giá trị như nhau.

Khi đổi sang hệ 10 chúng có giá trị là:

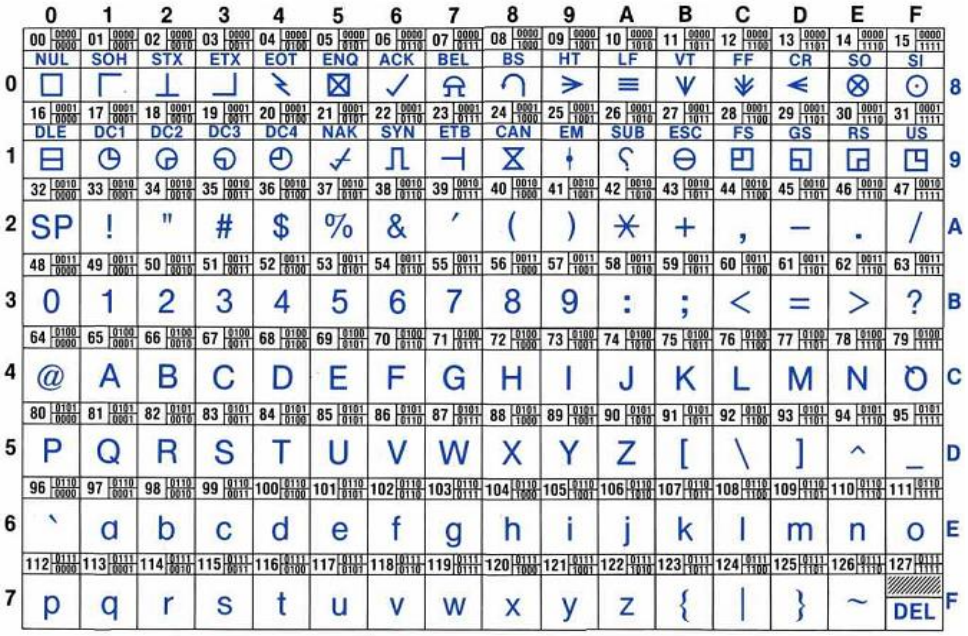
$A * 16^1 + 9 * 16^0 = 10 * 16 + 9 * 1$

$= 160 + 9$

$= 169$

✓ **Hằng ký tự:**

- Hằng ký tự là một ký tự riêng biệt được viết trong hai dấu nháy đơn (‘’).
- Hằng ký tự cũng được xem như trị số nguyên.
- ASCII = American Standard Code for Information Interchange.



Hình 3: Bảng mã ASCII

Hằng ký tự cũng được coi là những giá trị nguyên, vì mỗi ký tự đều được gán bằng một giá trị trong bảng mã ASCII, do đó bạn cũng có thể thực hiện được thực hiện phép toán trong biểu thức sau:

'9' – '0' = 9

Vì:

Số thứ tự mã ASCII của ký số 9 là 57.

Số thứ tự mã ASCII của ký số 0 là 48.

1. - 48 = 9
2. Hằng ký tự được viết theo cách '\x1x2x3'
3. Trong đó xi là một ký số tùy theo từng hệ đếm mà có giá trị khác nhau

Ví dụ:

Ký tự 'a' và 'A' có trị mã ASCII theo từng hệ đếm như sau:

	Trị của mã ASCII		
Ký tự	Hệ 8	Hệ 10	Hệ 16
'a'	141	97	61
'A'	101	65	41

Bảng 13: Bảng mã ASCII

✓ **Hằng chuỗi:**

Hằng chuỗi là một chuỗi bất kỳ đặt trong cặp dấu nháy kép (“”)

Ví dụ:

”**Turbo C**”

Thành Phố Hồ Chí Minh”

” ” /* Chuỗi rỗng */

Các ký tự trong chuỗi có thể gồm cả các ký tự đặt biệt (không có trên bàn phím) được biểu diễn bằng ký tự (\ ) và ký tự theo sau.

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

- **Biểu thức hằng:**

Một biểu thức toán học có thể xem là một biểu thức hằng nếu trong đó các toán hạng là những hằng số hoặc hằng ký tự. Khi đó biểu thức hắng sẽ được trình biên dịch tính toán sử lý trước và lưu trữ kết quả tính được.

Ví dụ:

8 * 20 – 13: Được tính trước để lưu trữ kết quả là 147.

‘ a ’ – ‘ A ’: Được tính trước để lưu trữ kết quả là 32.

Khả năng tính trước này làm giảm thời gian cần thiết để chạy chương trình, vì máy không cần phải tính lại các biểu thức đó nữa.

Ví dụ: Nếu sau đó gặp biểu thức:

’ C ’ + ‘ a ’ – ‘A’

Thì máy chỉ cần đem trị mã ASCII của ‘C’cộng với 32. (Vì ‘a’ – ‘A’ đã tính trước đó).

Tóm lại các hằng **int, long, float, double** thường dùng trong tính toán, còn các **hằng ký tự** và **chuỗi ký tự** thường dùng trong in ấn (đưa ra các dòng thông tin văn bản, điều khiển việc in v.v...)

Để kết thúc phần này đề nghị bạn nhập vào máy và chạy chương trình dưới đây minh họa cách viết các hằng và cách thể hiện chúng trên màn hình. Tin rằng sau khi chạy chương trình, bạn có thể đối chiếu kết quả với từng câu lệnh, từng mã điều khiển v.v...

Ví dụ:

```
/*Minh hoa cach dung cac hang*/
#include ”stdio.h” /*Khong can khai bao cung duoc*/
#include “stdlib.h” /*g can khai bao cung duoc*/
main()
{
printf(“\nhang so thuc: %10.2%f%10.2%f%10.2%f%10.2f”.
        14689.e-4, -0.125666E3.23468E-2, 1e4);
printf(“\nHang nguyen %  101d %  101d % 61%d % 6d % 6d”.
        456398461, 45639846, 35L, 35-123);
        printf(“\nHang nguyen he 8 va he 16 :%7d%7d%7d%7d”.
        0345, 0xa9, 0xA9, 0Xa9, 0XA9);
        /*In hang cac ky tu dung dac  ta % [fw] c*/
printf(“%c%c%6c%6c%6c%6c”,’\n’, ‘\n’, ‘a’\141’,’\”, ‘\”, ‘\\’);
/*in chuoi ky tu dac ta*/
printf(“\n\n % 10c % 10c % 10s “, “,“Chuc may man“ ) ;
getch();
}
```

Sau khi chạy chương trình có kết quả như sau:

Hang so thuc 10000.00	1.47	-125.67		234.68
Hang nguyen	45639846	45639846	35	35
Hang nguyen he 8 va he 16: 229 169	169	169		169
	a	a	‘	“ \
chuc may man				

Bảng 14: Bảng kết quả

4. Lệnh và khối lệnh, lệnh gán, lệnh gộp.

4.1. Khái niệm câu lệnh:

- Một câu lệnh xác định một công việc mà chương trình phải thực hiện.
- Kết thúc bởi dấu;

Phân loại: có 2 loại

+ Lệnh đơn:
  - ↳Không chứa 1 lệnh nào khác.
  - ↳Gồm: lệnh gán, nhập, xuất.
+ Lệnh có cấu trúc:
  - ↳Cấu trúc điều kiện rẽ nhánh.
  - ↳Cấu trúc điều kiện lựa chọn.
  - ↳Cấu trúc lặp.
  - ↳Cấu trúc lệnh hợp thành.

❄Lệnh gán:

Ví dụ:

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

Hình 4: Ví dụ

- Cú pháp:

  <Tên biến>=<Biểu thức>;

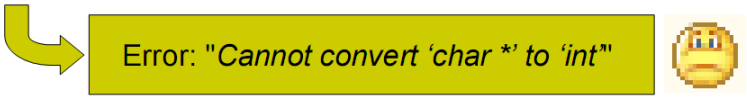
  Ý nghĩa: Gán giá trị cho một biến.
- Gán giá trị ngay tại lúc khai báo.

```
int x = 10, y=x;
```

Hình 5: Gán trị

+ Kiểu của biểu thức và kiểu của biến phải giống nhau.

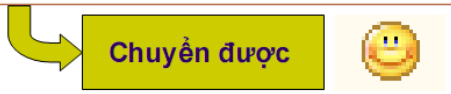
```
int main() {
      int x,y;
      x = 10; //  Gán hằng số 10 cho biến x
      y = "Xin chao";
            //y có kiểu int, còn "Xin chao" có kiểu char*
      return 0;
}
```



Hình 6: Kiểu của biểu thức và kiểu của biến phải giống nhau

+ Thường thì có sự chuyển đổi kiểu tự động nếu có thể.

```
int main() {
    int x,y;
    float r;
    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
    char ch;
    ch = 65.7; //  ch có kiểu char, còn 65.7 có kiểu float
    return 0;
}
```



Hình 7: Chuyển đổi kiểu tự động

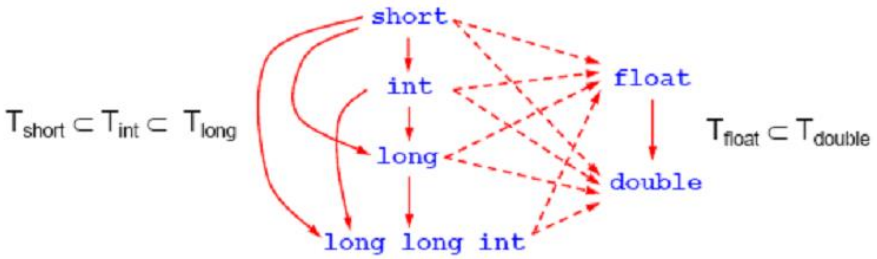
+ Kết quả chương trình sau là gì?

```
#include <conio.h>
#include <stdio.h>
int main(){
        int x,y;
        float r;
        char ch;
        clrscr();
        r=9000;
        x=10;
        y='d';
        r='e';
        ch='A';
        printf("y=%d, r=%f, ch=%c",y,r,ch);
        getch();
        return 0;
}
```

```
Turbo C++ IDE
y=100, r=101.000000, ch=A_
```

Hình 8: Kết quả

+ Trong C, các chuyển đổi sau được làm tự động



Hình 9: Các chuyển đổi được làm tự động

+ Những chuyển đổi trên đảm bảo không làm mất đi sự chính xác (loss of precision).

+ Việc chuyển đổi theo các hướng khác có thể làm mất đi sự chính xác. →Ví dụ:

```
double aDouble = 3.14;
int anInt;
anInt = aDouble;
aDouble = anInt;
```

„loss of precision"

⇒ aDouble = 3.000000

Hình 10: Chuyển đổi làm mất sự chính xác

+ Ép kiểu (casting type):

```
valueInNewType = (NewType)valueInOldType;
```

Hình 11: Ép kiểu

```
int anInt;
double aDouble = 3.14159;

aDouble = (int)aDouble;     // truncation of fraction
printf("%f\n", aDouble);    // --> 3

anInt = aDouble;            // truncation of fraction
aDouble = anInt/2;          // integer division 3/2 = 1
printf("%f\n", aDouble);    // --> 1

aDouble = (double)anInt/2; // floating-point division 3/2
printf("%f\n", aDouble);    // --> 1.50000

aDouble = anInt/2.0;        // floating-opint division 3/2
printf("%f\n", aDouble);    // --> 1.50000
```

Hình 12: Ví dụ ép kiểu

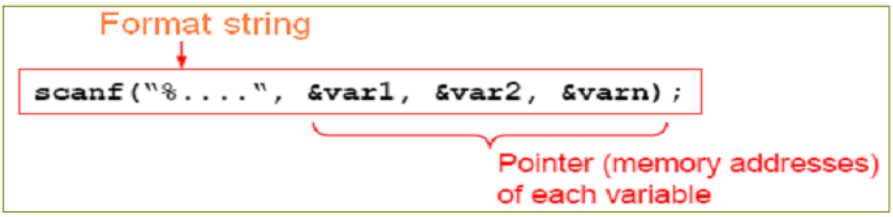
✧ Phép gán được viết gọn:

	short form	expanded form
+=	x += y;	x = x + y;
-=	x -= y;	x = x - y;
*=	x *= y;	x = x * y;
/=	x /= y;	x = x / y;
%=	x %= y;	x = x % y;

Hình 13: Ví dụ phép gán

4.2. Lệnh nhập giá trị từ bàn phím cho biến:

- scanf đọc giữ liệu từ bàn phím và gán giá trị cho biến.



Hình 14: Ví dụ lệnh scanf

- Chuỗi định dạng (format string): để quy định kiểu dữ liệu, cách biểu diễn, độ rộng, số chữ số thập phân,...

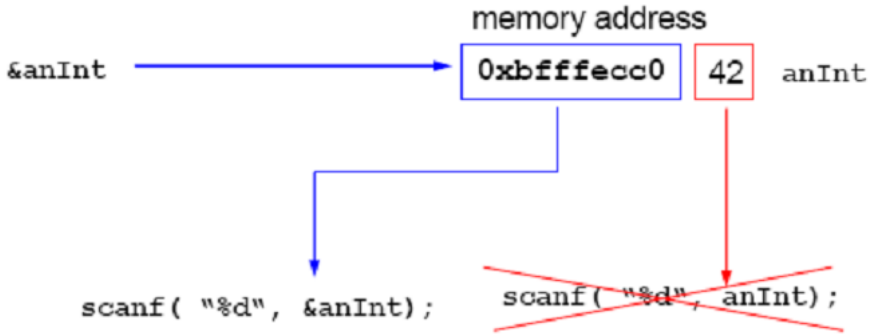
Định dạng	Ý nghĩa
%[số ký số]d	Nhập số nguyên có tối đa <số ký số>
%[số ký số] f	Nhập số thực có tối đa <số ký số> tính cả dấu chấm
%c	Nhập một ký tự

Hình 15: Ví dụ chuỗi định dạng

- scanf phải lưu giá trị vào 1 biến.

+ scanf("%d",anInt): không đúng, vì anInt xác định giá trị hiện hành của một biến.

+ scanf("%d",&anInt): đúng, vì địa chỉ của anInt đã được xác định



Hình 16: Lưu giá trị vào 1 biến

- Ví dụ dùng hàm Standard Input

```
#include <stdio.h>
main()
{
   int age;
   float size;

   printf("your age: ");
   scanf("%d", &age);
   printf("your size in meter: ");
   scanf("%f", &size);

   printf("You are %d years old and %f m.\n",
          age, size);
}
```

Output:

```
your age:                ⇐ 42
your size in meter:      ⇐ 1.76
You are 42 years old and 1.760000 m.
```

Hình 17: Ví dụ dùng hàm Standard Input

5. Thực thi chương trình, nhập dữ liệu, nhận kết quả.

CHƯƠNG 3: CÁC LỆNH CÓ CẤU TRÚC

Mã chương: MĐ10-03

Mục tiêu:

Sau khi học xong chương này sinh viên có khả năng:

- 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.

1. Khái niệm về lệnh cấu trúc.

1.1. Lệnh:

- Một tác vụ, một biểu thức hoặc chỉ là một câu lệnh gán nhưng có dấu chấm phẩy (;) ở cuối dòng đều được gọi là **một lệnh**.

Ví dụ:

```
...
int i,j = 10 /*Khai bao bien va gan gia tri*/
clrscr() ; /*Mot tac vu*/
printf("\nNhap mot so :"); /*In ra man hinh*/
scanf("%d", &a); /*Nhap so lieu tu ban phim*/
...
```

- Mỗi cấu trúc điều khiển trong chương trình theo đúng cú pháp của nó cũng là một lệnh, gọi là **lệnh điều khiển**. Trong Turbo C có ba dạng câu lệnh điều khiển sau:
  + Lệnh rẽ nhánh có điều kiện: if – else; switch – Case
  + Lệnh rẽ nhánh không điều kiện: bread; continue; goto.
  + Lệnh lặp chu trình (chu trình): for, while, do … while

1.2. Khối lệnh:

- Là một dãy các câu lệnh được bọc bởi cặp dấu {} gọi là một khối lệnh.

Ví dụ:

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

viết thụt vô 1 tab so với cặp dấu{}

Một tác vụ mà trong đó một lệnh của Turbo C không thể diễn tả đầy đủ thì phải sử dụng **khối lệnh.** Vậy thực chất khối lệnh cũng được coi là một lệnh riêng rẽ. Nói cách khác thì chỗ nào viết được một câu lệnh thì ở đó cũng có quyền đặt một khối lệnh.

- Nếu trong chương trình có sử dụng các hàm, các biến thì phần khai báo này thường đặt ở đầu của khối lệnh.

Ví dụ:

```
{
int a, b;
float x, y;
a = b =3 /*gan 3 cho a va b*/
x = 5.7;
y = a * x;
z = b * x ;
```

```
printf("\n Ket qua y = % 8.2f \n z = %8.2f", y, z);
getch() ;
}
```

- Khối lệnh lồng nhau: Trong các bài tập của các chương trước, bạn đã thấy bên trong một khối lệnh này lại có thể có khối lệnh khác. Đó chính là cấu trúc của khối lệnh lồng nhau và sự lồng nhau này thì không hạn chế. Có nhiều trường hợp phải sử dụng đến khối lệnh con (tức là khối lệnh nằm bên trong). Chẳng hạn như thân một hàm cũng là một khối lệnh con ở bên trong nó.

Về hình thức, cấu trúc một chương trình có các khối lệnh lồng nhau được trình bày như sau:

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

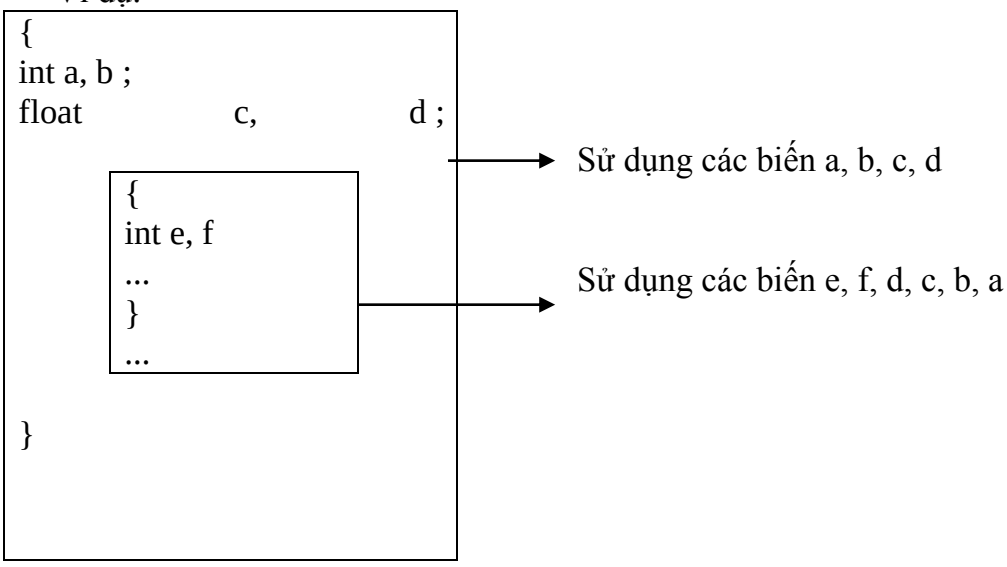
Hình 1: cấu trúc một chương trình có các khối lệnh lồng nhau

Lưu ý:

Khi một chương trình có nhiều khối lệnh và ở mỗi khối lệnh lại có sự khai báo các biến thì sự tác động của các biến trong và ngoài theo nguyên tắc sau:

- Nếu bên trong một khối lệnh bạn khai báo một biến (hay một mảng) có tên là a thì điều này không làm thay đổi giá trị của một biến (hay mảng) khác cũng có tên là a nhưng ở ngoài khối lệnh.
- Nếu một biến khai báo ở ngoài khối lệnh và không trùng tên với các biến ở bên trong khối lệnh thì biến ở bên ngoài khối lệnh cũng được sử dụng ở bên trong khối lệnh. Trường hợp này ta gọi là biến chung, còn điều nói ở trên gọi là biến riêng.

Ví dụ:



Bảng 1: Ví dụ

Để nắm thật chắc thế nào là lệnh, khối lệnh đồng thời để hiểu rõ phạm vi tác động của các biến khai báo ở bên trong và bên ngoài khối, đề nghị bạn nhập chương trình minh họa sau đây và chạy chương trình để xem kết quả trên màn hình.

Ví dụ:

```
/*Chương trinh minh hoa pham vi hoat dong cua mot bien*/
main()
{
int a, b = 20, c, d = 40;
float e = -35.1, x = 23, y, z, t = 36.1;
a = c = 10 ;
y = z = a + b + c + d ;
{
float y, z ;
y = z = e + x + t ;
printf("\nY trong khoi = %8.2f", y );
printf("\nZ trong khoi = %8.2f", z );
}
printf("\nY ngoai khoi = %8.2f", y );
printf("\nZ ngoai khoi = %8.2f", z );
getch();
}
```

Kết quả sau khi chạy chương trình :

Y trong khoi = 24.00
Z trong khoi = 24.00
Y ngoai khoi = 80.00
Z ngoai khoi = 80.00

2. Các lệnh cấu trúc rẽ nhánh như: if, switch

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

2.1. Dạng không đầy đủ:

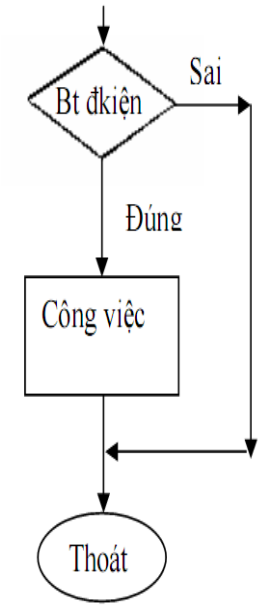
Quyết định sẽ thực hiện hay không thực hiện một khối lệnh.

Cú pháp:

if (<*Biểu thức điều kiện*>)
<*Công việc*>

↳ Từ khóa **if** phải viết bằng chữ thường.
↳ kết quả *công việc* phải là đúng (≠0) hoặc sai (=0).
↳ Nếu khối lệnh bao gồm 2 lệnh trở lên thì phải đặt trong dấu {}.

Lưu đồ cú pháp:



Giải thích:

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

Kiểm tra *Biểu thức điều kiện* trước.

Nếu điều kiện đúng (!= 0) thì thực hiện câu lệnh hoặc khối lệnh liền sau điều kiện.

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

Hình 2: Lưu đồ

Giải thích:

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

Kiểm tra biểu thức điều kiện trước.

Nếu điều kiện đúng (!=0) thì thực hiện câu lệnh hay khối lệnh liền sau điều kiện.

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

Ví dụ:

Yêu cầu người thực hiện chương trình nhập vào một số thực a. In ra màn hình kết quả nghịch đảo của a khi a ≠ 0.

```
#include <stdio.h>
#include <conio.h>
int main ()
{
 float a;
  printf("Nhap a = "); scanf("%f",&a);
  if (a !=0 )
    printf("Nghich dao cua %f la %f",a,1/a);
 getch();
 return 0;
}
```

Giải thích:

Nếu chúng ta nhập vào a ≠ 0 thì câu lệnh printf ("Nghich dao cua %f la %f",a,1/a) được thực hiện, ngược lại câu lệnh này không được thực hiện.

- Lệnh getch() luôn luôn được thực hiện vì nó không phải là "lệnh liền sau" điều kiện if.

- Ví dụ 2: Yêu cầu người chạy chương trình nhập vào giá trị của 2 số a và b, nếu a lớn hơn b thì in ra thông báo "Gia trị của a lớn hơn giá trị của b", sau đó hiển thị giá trị cụ thể của 2 số lên màn hình.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
 int a,b;
  printf("Nhap vao gia tri cua 2 so a, b!");
 scanf("%d%d",&a,&b);
 if (a>b)
{
   printf("\n Gia tri cua a lon hon gia tri cua b");
   printf("\n a=%d,  b=%d",a,b);
 }
 getch();
 return 0;
}
```

Giải thích:

Nếu chúng ta nhập vào giá trị của a lớn hơn giá trị của b thì khối lệnh:

```
{
   printf("\n Gia tri cua a lon hon gia tri cua b");
   printf("\n a=%d,  b=%d",a,b);
}
```

Sẽ được thực hiện, ngược lại khối lệnh này không được thực hiện.

Ghi chú:

↳ Không đặt dấu chấm phẩy (;) sau câu lệnh **if**

Ví dụ: **if**<Công việc>;

→ Trình bien dịch không báo lỗi nhưng khối lệnh không được thực hiện cho dù điều kiện đúng hay sai.

2.2. Dạng đầy đủ:

Cú pháp:

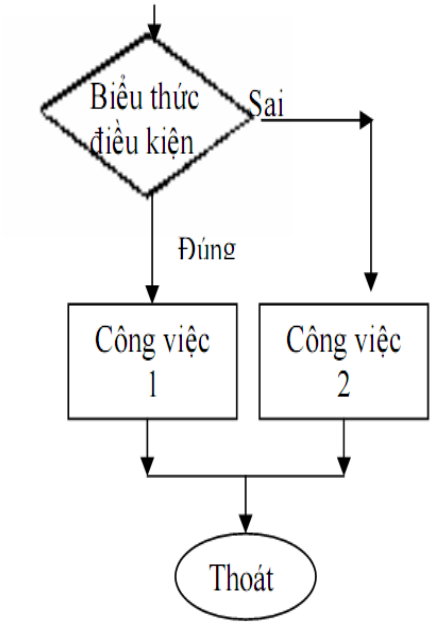
if (*<Biểu thức điều kiện>*
<Công việc 1>
else
<Công việc 2>

Ghi chú:

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

↳ kết quả *Công việc 1, Công việc 2* phải là đúng (≠0) hoặc sai (=0).

Lưu đồ cú pháp:



Giải thích:
Công việc 1, công việc 2 được thể hiện là 1 câu lệnh hay 1 khối lệnh.
Đầu tiên *Biểu thức điều kiện* được kiểm tra trước.
Nếu điều kiện đúng thì thực hiện công việc 1.
Nếu điều kiện sai thì thực hiện công việc 2.
Các lệnh phía sau công việc 2 không phụ thuộc vào điều kiện.

Hình 3: Lưu đồ cú pháp

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

Ví dụ 1: Yêu cầu người thực hiện chương trình nhập vào một số thực a. In ra màn hình kết quả nghịch đảo của a khi a ≠ 0, khi a =0 in ra thông báo "Khong the tim duoc nghich dao cua a"

```
#include <stdio.h>
#include <conio.h>
int main ()
{
 float a;
  printf("Nhap a = "); scanf("%f",&a);
  if (a !=0 )
    printf("Nghich dao cua %f la %f",a,1/a);
 else
    printf("Khong the tim duoc nghich dao cua a");
getch();
 return 0;
}
```

Giải thích:

Nếu chúng ta nhập vào a ≠ 0 thì câu lệnh printf ("Nghich dao cua %f la %f",a,1/a) được thực hiện, ngược lại câu lệnh printf ("Khong the tim duoc nghich dao cua a") được thực hiện.

Lệnh getch() luôn luôn được thực hiện.

Ví dụ 2: Yêu cầu người chạy chương trình nhập vào giá trị của 2 số a và b, nếu a lớn hơn b thì in ra thông báo "Gia trị của a lớn hơn giá trị của b, giá trị của 2 số", ngược lại thì in ra màn hình câu thông báo "Giá trị của a nhỏ hơn hoặc bằng giá trị của b, giá trị của 2 số".

```
#include <stdio.h>
#include<conio.h>
int main ()
{
  int a, b;
  printf("Nhap vao gia tri cua 2 so a va b !");
 scanf("%d%d",&a,&b);
 if (a>b)
  {
     printf("\n a lon hon b”);
     printf("\n a=%d  b=%d ",a,b);
  }
 else
  {
     printf("\n a nho hon hoac bang b");
     printf("\n a=%d  b=%d",a,b);
  }
  printf("\n Thuc hien xong lenh if");
 getch();
 return 0;
}
```

Giải thích:

Nếu chúng ta nhập vào 40 30 thì kết quả hiện ra trên màn hình là

a lon hon b
a=40 b=30
Thuc hien xong lenh if

Còn nếu chúng ta nhập 40 50 thì kết quả hiện ra trên màn hình là

a nho hon hoac bang b
a=40 b=50
Thuc hien xong lenh if

Ví dụ 3: Yêu cầu người thực hiện chương trình nhập vào một số nguyên dương là tháng trong năm và in ra số ngày của tháng đó.

- Tháng có 31 ngày: 1, 3, 5, 7, 8, 10, 12
- Tháng có 30 ngày: 4, 6, 9, 10
- Tháng có 28 hoc 29 ngày : 2

```
#include <stdio.h>
#include<conio.h>
int main ()
{
 int thg;
  printf("Nhap vao thang trong nam !");
 scanf("%d",&thg);
 if (thg==1||thg==3||thg==5||thg==7||thg==8||thg==10||thg==12)
   printf("\n Thang %d co 31 ngay ",thg);
 else if (thg==4||thg==6||thg==9||thg==11)
   printf("\n Thang %d co 30 ngay",thg);
 else if (thg==2)
```

```
            printf("\n Thang %d co 28 hoac 29 ngay",thg);
              else printf("Khong co thang %d",thg);
          printf("\n Thuc hien xong lenh if");
         getch();
          return  0;
        }
```

Giải thích:

Nếu chúng ta nhập vào một trong các số 1, 3, 5, 7, 8, 10, 12 thì kết quả xuất hiện trên màn hình sẽ là:

Thang <số> co 31 ngay
Thuc hien xong lenh if

Nếu chúng ta nhập vào một trong các số 4, 6, 9, 11 thì kết quả xuất hiện trên màn hình sẽ là:

Thang <số> co 30 ngay
Thuc hien xong lenh if

Nếu chúng ta nhập vào số 2 thì kết quả xuất hiện trên màn hình sẽ là:

Thang 2 co 28 hoac 29 ngay
Thuc hien xong lenh if

Nếu chúng ta nhập vào số nhỏ hơn 0 hoặc lớn hơn 12 thì kết quả xuất hiện trên màn hình sẽ là:

Khong co thang <s>
Thuc hien xong lenh if

Trong đó <số> là con số mà chúng ta đã nhập vào.

Lưu ý:

Ta có thể sử dụng các câu lệnh if…else lồng nhau. Trong trường hợp if…else lồng nhau thì else sẽ kết hợp với if gần nhất chưa có else.

Trong trường hợp câu lệnh if "bên trong" không có else thì phải viết nó trong cặp dấu {} (coi như là khối lệnh) để tránh sự kết hợp else if sai.

Ví dụ 1:

```
if ( so1>0)
  if (so2 > so3)
  a=so2;
  else   /*else ca if (so2>so3) */
  a=so3;
```

Ví dụ 2:

```
if (so1>0)
 {
    if (so2>so3)   /*lenh if này không có else*/
   a=so2;
}
else  /*else ca if (so1>0)*/
  a=so3;
```

Ghi chú:

↳ Sau else không có dấu chấm phẩy (;)

VD: else; printf('a khac b\n");

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

3. Các lệnh lặp như for, while, do while

Tuy lệnh if kết hợp với lệnh goto cũng tạo nên một cấu trúc lặp nhưng có nhiều hạn chế. Trong cấu trúc lặp chính thống của Turbo C có 3 câu lệnh sau đây:

- Lệnh for được sử dụng khi cần kiểm tra điều kiện rồi mới thực hiện các câu lệnh. Số lần lặp là biết trước, tương tự lệnh for của Pascal và Foxpro.
- Lệnh while được sử dụng khi cần kiểm tra điều kiện rồi mới thực hiện các lệnh. Sốn lần lặp không thể biết trước, tương tự như lệnh while … do của Pascal hoặc do do while của Foxpro.
- Lệnh do… while được sử dụng khi cần thực hiện các lệnh trước rồi mới kiểm tra điều kiện sau. Số lần lặp không thể biết trước, tương tự như lệnh reapeat … until của Pascal.

3.1. Vòng lặp for:

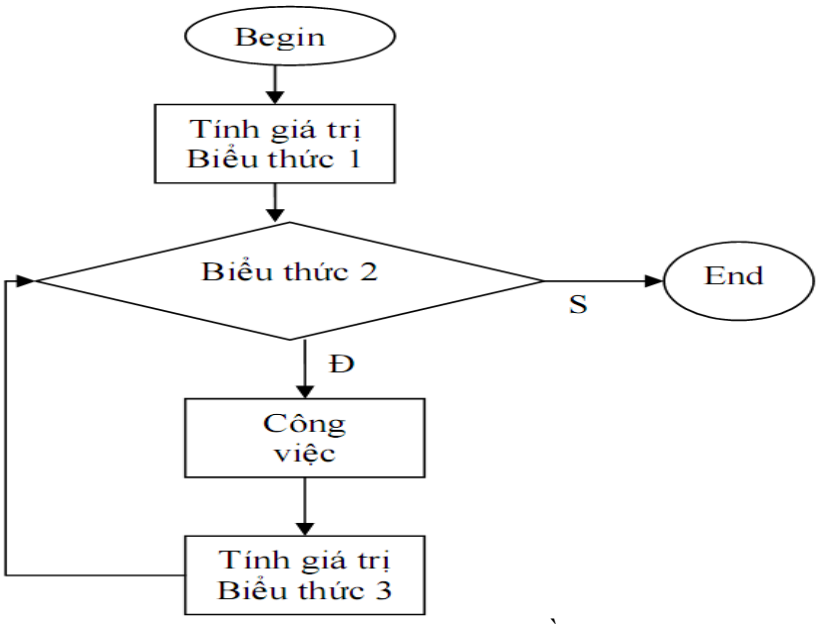
Lệnh for cho phép lập lại công việc cho đến khi điều kiện sai.

Cú pháp:

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

<Công việc>

Lưu đồ:



Hình 4: Lưu đồ

Giải thích:

<Công việc>: được thể hiện là 1 câu lệnh hay 1 khối lệnh. Thứ tự thực hiện các câu lệnh for như sau:

B1: Tính giá trị các biểu thức 1.

B2: Tính giá trị các biểu thức 2.

+ Nếu giá trị các biểu thức 2 là sai (=0): thoát khi câu lệnh for.
+ Nếu giá trị các biểu thức 2 là úng (!=0): <Công việc> được thực hiện.

B3: Tính giá trị các biểu thức 3 và quay lại B2.

Một số lưu ý khi sử dụng câu lệnh for:

Khi biểu thức 2 vắng mặt thì nó được coi là luôn luôn đúng.

Biểu thức 1: thông thường là một phép gán để khởi tạo giá trị ban đầu cho biến điều kiện.

Biểu thức 2: là một biểu thức kiểm tra điều kiện đúng sai để dưng vòng lặp.

Biểu thức 3: thông thường là một phép gán thay đổi giá trị các biến điều kiện.

Trong mỗi biểu thức có thể có nhiều biểu thức con. Các biểu thức con cần phân biệt bởi dấu phẩy.

Lưu ý:

Khi sử dụng câu lệnh for, bạn cần lưu ý những vấn đề sau:

- Khi biểu thức 2 vắng mặt thì nó luôn luôn được xem là đúng trong trường hợp này muốn thoát ra khỏi vòng lặp for phải dùng một trong ba lệnh : break; goto hoặc return.

- Một trong 3 biểu thức của lệnh for lại có quyền viết một dãy biểu thức, nhưng giữa các biểu thức con này phải phân cách nhau dấu (,). Giá trị tính toán xác định giá trị lần lượt thực hiện từ trái sang phải.

Nếu Biểu thức 2 là một dãy biểu thức thì tính đúng hoặc sai của biểu thức này sẽ xác định bởi tính đúng, sai của biểu thức cuối cùng trong dãy này.

- Bên trong thân vòng lặp for này có thể chứa một vòng lặp for khác. Như vậy bạn có quyền thành lập vòng lặp for lồng nhau.

- Khi gặp các câu lệnh break trong thân vòng for máy sẽ ra khỏi thân for sâu nhất chứa câu lệnh này.

- Trong thân vòng lặp for, bạn có thể dùng lệnh goto để nhảy ra khỏi vòng lặp và nhảy tới vị trí mong muốn.

- Bạn cũng có thể dùng lệnh return tring thân for để trở về một hàm nào đó. Đến đây hẳn bạn đọc đều nhận thấy vòng lặp f0or trong Turbo C có nhiều qui định hết sức chặt chẽ nhưng lại hết sức linh hoạt và nếu bạn nào đã học Pascal, Foxpro… thì lệnh for của chúng đơn giản hơn nhiều so với Turbo C.

Ví dụ 1:

```
#include <stdio.h>
main()
{
    int i;        // i is loop variable
          ①     ②     ③
    for (i=1; i<=10; i++) {
        printf("%d: %d\n", i, i*i);
    }
}
```

Hình 5: Ví dụ

- (1) Khởi tạo i giá trị 1.
  - (2) Thoát khỏi vòng lặp nếu i> 10.
  - (3) Tăng i sau mỗi lần lặp.

Ví dụ 2: Viết đoạn chương trình in dãy số nguyên từ 1 đến 10.

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

```
}
```

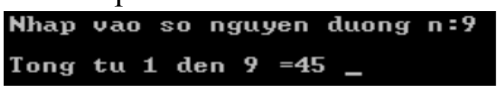
Kết quả chương trình như sau:

Hình 6: Kết quả chương trình

Ví dụ 3: Viết chương trình nhập vào một số nguyên n. Tính tổng của các số nguyên từ 1 đến n.

```
#include <stdio.h>
#include<conio.h>
int main ()
{   unsigned int n,i,tong;
 clrscr();
  printf("\n Nhap vao so nguyen duong n:"); scanf("%d",&n);
 tong=0;
  for (i=1; i<=n; i++)
   tong+=i;
  printf("\n Tong tu 1 den %d =%d ",n,tong);
 getch();
 return 0;
}
```

Nếu chúng ta nhập vào số 9 thì kết quả như sau:



Hình 7: Kết quả chương trình

Ví dụ 4: Viết chương trình in ra trên màn hình một ma trận có n dòng m cột như sau:

```
1  2  3  4  5  6  7
2  3  4  5  6  7  8
3  4  5  6  7  8  9
…
#include <stdio.h>
#include<conio.h>
int main ()
{   unsigned int dong, cot, n, m;
 clrscr();
  printf("\n Nhap vao so dong va so cot :");
scanf("%d%d",&n,&m);
 for (dong=0;dong<n;dong++)
 {
    printf("\n");
    for (cot=1;cot<=m;cot++)
  printf("%d\t",dong+cot);
  }
 getch();
 return 0;
}
```

Kết quả khi nhập 3 dòng 6 cột như sau:

```
Nhap vao so dong va so cot :3 6

1       2       3       4       5       6
2       3       4       5       6       7
3       4       5       6       7       8
```

Hình 8: Kết quả chương trình

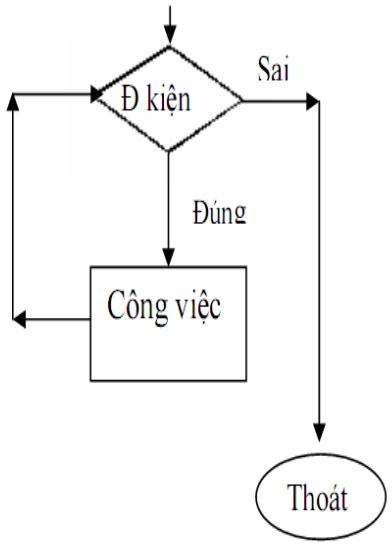
3.2. Vòng lặp while:

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

Cú pháp:

while (Biểu thức điều kiện)
<Công việc>

Lưu đồ:



Giải thích:

- <Công việc>: được thể hiện bằng 1 câu lệnh hay 1 khối lệnh.
- Kiểm tra *Biểu thức điều kiện* trước.
- Nếu điều kiện sai (=0) thì thoát khỏi lệnh while.
- Nếu điều kiện đúng (!=0) thì thực hiện công việc rồi quay lại kiểm tra điều kiện tiếp.

Hình 9: Lưu đồ

Lưu ý:

Lệnh while gồm có biểu thức điều kiện và thân vòng lặp (khi lệnh thực hiện công việc).

Vòng lặp dừng lại khi nào điều kiện sai.

Khi lệnh thực hiện công việc có thể rỗng, có thể làm thay đổi điều kiện.

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

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

```
getch();
return 0;
}
```

Kết quả chương trình như sau:

```
Day so tu 1 den 10 :1 2 3 4 5 6 7 8 9 10
```

Hình 10: Kết quả chương trình

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

```
#include <stdio.h>
#include<conio.h>
int main ()
{
 unsigned int n, i, tong;
 clrscr();
  printf("\n Nhap vao so nguyen duong n:");
 scanf("%d",&n);
 tong=0;
 i=1;
 while (i<=n)
 {
   tong+=i;
   i++;
 }
  printf("\n Tong tu 1 den %d =%d ",n,tong);
 getch();
 return 0;
}
```

Nếu chúng ta nhập vào số 9 thì kết quả như sau:

```
Nhap vao so nguyen duong n:9
Tong tu 1 den 9 =45 _
```

Hình 11: Kết quả chương trình

Ví d 3: Viết chương trình in ra trên màn hình một ma trận có n dòng m cột như sau:

```
1   2   3   4   5   6   7
2   3   4   5   6   7   8
3   4   5   6   7   8   9
…
#include <stdio.h>
#include<conio.h>
int main ()
{
 unsigned int dong, cot, n, m;
 clrscr();
  printf("\n Nhap vao so dong va so cot :");
```

```
scanf("%d%d",&n,&m);
dong=0;
while (dong<n)
{
   printf("\n");
   cot=1;
   while (cot<=m)
  {
   printf("%d\t",dong+cot);
   cot++;
  }
    dong++;
 }
getch();
 return 0;
}
```

Kết quả khi nhập 3 dòng 6 cột như sau

```
Nhap vao so dong va so cot :3 6

1       2       3       4       5       6
2       3       4       5       6       7
3       4       5       6       7       8
```

Hình 12: Kết quả chương trình

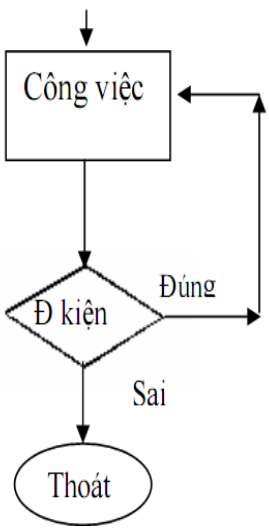
3.3. Vòng lặp do…while:

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

Cú pháp:

do
<Công việc>
while (*<Biểu thức điều kiện>*)

Lưu đồ:



Giải thích:

- <Công việc>: được thể hiện bằng 1 câu lệnh hay 1 khối lệnh.
- Trước tiên công việc được thực hiện trước, sau đó mới kiểm tra *Biểu thức điều kiện*.
- Nếu điều kiện sai thì thoát khỏi lệnh do ... while.
- Nếu điều kiện còn đúng thì thực hiện công việc rồi quay lại kiểm tra điều kiện tiếp.

Hình 13: Lưu đồ

Lưu ý:

Lệnh do…while thực hiện công việc ít nhất 1 lần.

Lệnh do…while thực hiện công việc ít nhất 1 lần.

Khi lệnh thực hiện công việc có thể rỗng, có thể làm thay đổi điều kiện.

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

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

Kết quả chương trình như sau:

```
Day so tu 1 den 10 :1 2 3 4 5 6 7 8 9 10
```

Hình 14: Kết quả chương trình

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

```
#include <stdio.h>
#include<conio.h>
int main ()
{   unsigned int n,i,tong;
 clrscr();
  printf("\n Nhap vao so nguyen duong n:");
 scanf("%d",&n);
 tong=0;
 i=1;
 do
 {
   tong+=i;
   i++;
  } while (i<=n);
  printf("\n Tong tu 1 den %d =%d ",n,tong);
 getch();
 return 0;
}
```

Nếu chúng ta nhập vào số 9 thì kết quả như sau:

```
Nhap vao so nguyen duong n:9
Tong tu 1 den 9 =45 _
```

Hình 15: Kết quả chương trình

Ví dụ 3: Viết chương trình in ra trên màn hình một ma trận có n dòng m cột như sau (n, m>=1):

```
1  2  3  4  5  6  7
2  3  4  5  6  7  8
3  4  5  6  7  8  9
…
#include <stdio.h>
#include<conio.h>
int main ()
{   unsigned int dong, cot, n, m;
 clrscr();
  printf("\n Nhap vao so dong va so
 scanf("%d%d",&n,&m);
 dong=0;
 do
 {
    printf("\n");
    cot=1;
  do
  {
   printf("%d\t",dong+cot
   cot++;
   } while (cot<=m);
    dong++;
  } while (dong<n);
getch();
 return 0;
}
```

Kết quả khi nhập 3 dòng 6 cột như sau

```
Nhap vao so dong va so cot :3 6

1       2       3       4       5       6
2       3       4       5       6       7
3       4       5       6       7       8
```

Hình 16: Kết quả chương trình

3.4. So sánh các vòng lặp:

✓ Vòng lặp for, while:

Kiểm tra điều kiện trước thực hiện công việc sau nên đoạn lệnh thực hiện công việc có thể không được thực hiện .

Vòng lặp kết thúc khi nào điều kiện sai.

✓ Vòng lặp do…while:

Thực hiện công việc trước kiểm tra điều kiện sau nên đoạn lệnh thực hiện công việc được thực hiện ít nhất 1 lần.

Vòng lặp kết thúc khi nào điều kiện sai.

4. Các lệnh đơn nhằm kết thúc sớm vòng lặp

4.1. Lệnh break:

- Lệnh break cho phép thoat khỏi vòng lặp của các lệnh **for, while, do while** và cấu trúc **switch … case** như đã trình bày ở trên. Lệnh break tương tự như lệnh **EXIT** trong **FOXPRO.**
- Khi có nhiều vòng lặp (chu trình) lồng nhau, lệnh break có tác dụng ra khỏi vòng lặp (hoặc lệnh switch) bên trong nhất chứa nó mà không cần kiểm tra điều kiện kết thúc sớm vòng lặp. Nói một cách khác máy sẽ bỏ qua các câu lệnh còn lại để thoát ra khỏi vòng lặp.
- Mọi câu lệnh **break** đều có thể thay thế bằng lệnh **goto** kèm theo nhãn sẽ chuyển tới.

Để hiểu rõ tác dụng của câu lệnh **break**, bạn hãy nhập và chạy chương trình sau:

Ví dụ:

```
main ()
{
int i = 10;
while (1) /*điêu kiên luôn luôn đung*/
{
if(!--i)
break;
printf("i = %d\n", i);
}
}
```

Chú ý:

Lệnh if(!--i) tương đương với if(--i = = 0 nhưng toán tử -- (giảm) có độ ưu tiên cao hơn so với toán tử = = (bằng)).

Ví dụ: Viết chương trình nhập vào một năm, máy sẽ thông báo năm đó là năm nhuận hay không?

```
main()
{
int nam;
clrscr();
printf("\nCho biet nam can xem?");
scanf("%d", &nam);
whitch (nam%4)
{
case 0:
                printf("\nNam %4", nam);
                Printf("\nLa nam nhuan");
                break;
case 1:
```

```
                printf("\nNam ngoai %4", nam - 1);
                printf("\nMoi la nam nhuan");
                break;
        case 2 :
                printf("\nNam kia %4d", nam - 2 ) ;

                printf("\nMoi la nam nhuan");
                break;
        default :
                printf("\nSang nam %4d", nam + 1) ;
                printf("\nMoi la nam nhuan");
        break ;
        }
        }
```

4.2. Lênh continue:

Câu lệnh **break** trình bày ở trên dùng để thoát khỏi một vòng lặp, ngược lại câu lệnh continue dùng để bắt đầu một vòng lặp chứa nó. Khi gặp lệnh continue, máy sẽ bỏ qua các câu lệnh còn lại để trở về đầu vòng lặp tương tự như lệnh **LOOP** trong **FOXPRO.**
Nói cụ thể và chính xác hơn thì:

- Khi gặp lệnh **continue** bên trong vòng lặp for máy sẽ chuyển đến **khởi đầu** lại.
- Khi gặp câu lệnh **continue** bên trong vòng lặp while hoặc do … while, máy sẽ xác định lại giá trị của biểu thức và sau đó tiến hành kiểm tra điều kiện để kết thúc vòng lặp hay không.

Lưu ý:
Lệnh **continue** chỉ áp dụng trong cấu trúc lặp, không áp dụng cho cấu trúc whitch.
Đê minh họa câu lệnh continue, đề nghị bạn nhập và chạy thử chương trình:
Ví dụ:

```
        float a[][4] = {
                                { 25, 0, -3, 5},
                                { -6, 4, 0, -2},
                { 30, -4, 7, -3}
                };
         main( )
        {
        int i, j, k = 0;
        float s = 0, max = 0;
        for(i = 0; i < 3; i++ )
           for(j = 0; j < 4; ++j)
        if(a[i][j] < 0)
        continue;
        s += a[i][j];
        if(max < a[i][j])
        max  =  a[i][j];
        if(a[i][j] > 0)
        ++k;
```

```
}
printf("\nSo phan tu duong la = %d", k);
printf("\nTong cac phan tu duong la = %8.2f", s);
printf("\Phan tu duong lon nhat la = %8.2f", max);
getch() ;
}
```

Kết quả sau khi chay chương trình là:

Số phần tử dương là	= 5
Tổng các phần tử dương là	= 71.00
Phần tử dương lớn nhất là	= 30.00

Bảng 2: Kết quả sau khi chay chương trình

4.3. Lênh goto:

Lệnh **goto** dùng để chuyển quyền điều khiển tới một câu lệnh nào đó được chỉ định bởi nhãn.

Cú pháp:

Goto nhan ;

Bảng 3 : Cú pháp

Trong đó nhãn là một tên hợp lệ và tên này phải đặt trước lệnh mà bạn muốn nhảy đến. Cú pháp hợp lệ là:

nhan: lenh ;

Bảng 4 : Cú pháp hợp lệ

Khi sử dụng lệnh **goto** bạn cần nắm các nguyên tắc sau:

- Nếu câu lệnh **goto** và **nhãn** nằm trong một hàm thì lệnh goto chỉ cho phép nhẩy từ vị trí này sang vị trí khác trong thân của hàm đó thôi (không được nhảy từ hàm này sang hàm khác).
- Không cho phép sử dụng lệnh **goto** từ ngoài vào trong một khối lệnh nhưng lại cho phép nhẩy **từ trong ra ngoài.**

Nếu viết như dưới đây là sai:

```
...
Goto ketqua ;
{
Ketqua : printf("\nNhap vao mot so : ");
…
}
```

Viết như dưới đây mới đúng.

```
ketqua: printf("\nNhap vao mot so : ") ;
…
{
…
Goto ketqua;
…
}
```

- Lệnh goto kết hợp với lệnh if cũng có thể tạo thành một cấu trúc lặp như sau:

laplai = Câu lệnh 1;

Câu lệnh 2;
…
Câu lệnh n;
if (biểu thức)
goto laplai;

Thân cấu trúc gồm n câu lệnh sẽ được thực hiện lặp đi lặp lại nhiều lần đến khi biểu thức trong câu lệnh if vẫn còn đúng.

Ví dụ:

```
main()
{
int i, k;
    tt : printf ("|nNhap vao mot so nguyen");
    scanf("%d", &i);
    switch (i);{

    case 0: printf("\nXin chao quy ba");
            break;
    case 1: printf("\nXin chao quy ong");
                    break;
    case 2: printf("\nXin chao quy co");
                    break;
    case : printf("\nXin chao quy cau");
                    break;
    default: printf("\nXin chao moi nguoi");
    }
    printf("\nTiep tuc nua hay khong?");
     printf("\nBam so (1) de tiep tuc, so (2) de dung");
    scanf("%d", &k);
    if(i = = 1)
    goto tt;  }
```

CHƯƠNG 4: HÀM

Mã chương: MĐ10-04

Mục tiêu:

Sau khi học xong chương này sinh viên có khả năng:

- Hiểu được khái niệm hàm.
- Trình bày được qui tắc xây dựng hàm và vận dụng được khi thiết kế xây dựng chương trình.
- Hiểu được nguyên tắc xây dựng hàm, thế nào là tham số, tham trị.
- Biết cách truyền tham số đúng cho hàm.
- Sử dụng được các lệnh kết thúc và lấy giá trị trả về cho hàm.

1. Khái niệm hàm là gì, tại sao phải xây dựng và sử dụng hàm.

Khái niệm về hàm trong C:

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

Ưu điểm của chương trình con dễ kiểm tra tính đúng đắn khi ráp vào chương trình chính, phát hiện sai sót và hiệu chỉnh trong chương trình chính sẽ tiện lợi hơn.

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

Có hai loại hàm: hàm chuẩn và hàm tự định nghĩa. Trong chương này ta chú ý đến việc định nghĩa hàm & cách sử dụng hàm đó.

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

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

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

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

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

int main()
{
  int a, b, c;
  printf("\n Nhap vao 3 so a, b,c  ");
 scanf("%d%d%d",&a,&b,&c);
  printf("\n So lon la %d",max(a, max(b,c)));
 getch();
```

```
return 0;
}
```

- **Hàm thư viện:**

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

```
#include <tên thư viện.h>
```

Một số thư viện:

```
alloc.h     assert.h    bcd.h       bios.h      complex.h
conio.h     ctype.h     dir.h       dirent.h    dos.h
errno.h     fcntl.h     float.h     fstream.h   grneric.h
graphics.h  io.h        iomanip.h   iostream.h  limits.h
locale.h    malloc.h    math.h      mem.h       process.h
setjmp.h    share.h     signal.h    stdarg.h    stddef.h
stdio.h     stdiostr.h  stdlib.h    stream.h    string.h
strstrea.h  sys\stat.h  sys\timeb.h sys\types.h time.h
values.h.
```

Ý nghĩa của một số thư viện thường dùng:

- **studio.h**: Thư viện chứa hàm vào/ ra chuẩn (standard input/ output) gồm các hàm **printf(), scanf(), getc(), putc(), gets(), puts(), fflush(), fopen(), fclose(), fread(), fwrite(), getchar(), putchar(), getw(), putw()…**
- **conio.h**: Thư viện chứa các hàm vào ra trong chế độ DOS (DOS console). Gồm các hàm **clrscr(), getch(), getche(), getpass(), cgets(), cputs(), putch(), clreol(),…**
- **math.h**: Thư viện chứa các hàm tính toán gồm các hàm **abs(), sqrt(), log().log10(), sin(), cos(), tan(), acos(), asin(), atan(), pow(), exp(),…**
- **alloc.h**: Thư viện chứa các hàm liên quan đến việc quản lý bộ nhớ. Gồm các hàm **calloc(), realloc(), malloc(), free(), farmalloc(), farcalloc(), farfree(), ….**
- **io.h**: Thư viện chứa các hàm vào ra cấp thấp. Gồm các hàm **open(), _open(), read(), _read(), close(), _close(), creat(), _creat(), creatnew(), eof(), filelength(), lock(),…**
- **graphics.h**:Thư viện chứa các hàm lien quan đến đồ họa. Gồm **initgraph(), line(), circle(), putpixel(), getpixel(), setcolor(), …**

…

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

2. Nguyên tắc xây dựng và phân biệt các tham số của hàm:

Do người lập trình tạo ra nhằm đáp ứng nhu cầu của mình.

2.1. Định nghĩa hàm:

Cấu trúc của một hàm tự thiết kế:

```
< Kiểu dữ liệu > Tên hàm ( [ danh sách các tham số ] )
[ Khai báo kiểu dữ liệu cho các tham số ];
{
[Khai báo biến cục bộ và các câu lệnh thực hiện hàm]
[return [<Biểu thức>];]
}
```

Giải thích:

- Kiểu dữ liệu: Nếu hàm phải trả về một giá trị thì trước hết phải khai báo kiểu dữ lệu của hàm.
- Tên hàm: Tên hợp lệ của hàm gồm 8 ký tự viết liền nhau. Nên đặt tên hàm mang tính gợi nhớ để dễ dàng cho việc sử dụng và sửa sau này.
- Danh sách các tham số: Đó là tham số hình thức gọi là các biến hình thức của hàm. Những tham số này sẽ nhận giá trị thực bằng cách truyền tham số mỗi khi hàm này được gọi đến. Nếu có nhiều tham số thì dùng dấu phẩy để phân cách chúng.
- Khai báo kiểu cho các tham số: Nếu hàm có các tham số hình thức thì phải khai báo kiểu dữ liệu cho mỗi tham số tương tự như khai báo kiểu dữ liệu cho các biến. Nếu có nhiều tham số cùng kiểu dữ liệu thì phải có dấu phẩy (,) để phân cách các tham số.
- Thân hàm: Là phần giới hạn bởi cặp dấu ngoặc nhọn {}. Trong thân hàm bao gồm:

+ Khai báo biến cục bộ: các biến cục bộ chỉ có tác động trong thân hàm, nó không có mối liên hệ gì với các biến trong các hàm khác của chương trình. Trái lại tham số hình thức được dùng để trao đổi dữ liệu từ hàm này sang hàm khác.

+ Các câu lệnh: Trong thân hàm tùy theo từng yêu cầu cụ thể, bạn có thể đặt các câu lệnh từ đơn giản cho đến câu lệnh gán, biểu thức tính toán đến các câu lệnh điều kiện và vòng lặp,…

+ Trong thân hàm có thể sử dụng một câu lệnh **return ([biểu thức]);**.Bạn có thể sử dụng nhiều câu lệnh **return** ở những chổ khác nhau và cũng có thể không sử dụng câu lệnh này. Giá trị của biểu thức trong câu lệnh **return** sẽ được gán cho hàm.

Cú pháp:

return ; /*không trả về giá trị*/
return <biểu thức>; /*Trả về giá trị của biểu thức*/
return (<biểu thức>); /*Trả về giá trị của biểu thức*/

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

Ví dụ 1: Viết hàm tìm số lớn giữa 2 số nguyên a và b

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

Ví dụ 2: Viết hàm tìm ước chung lớn nhất giữa 2 số nguyên a, b. Cách tìm: ưu tiên ta gọi số UCLN của hai số là số nhỏ nhất trong hai số đó. Nếu điều đó không đúng thì ta giảm đi một đơn vị và cứ giảm như vậy cho tới khi nào tìm thấy UCLN.

```
int  ucln(int a, int b)
{
 int u;
 if (a<b)
   u=a;
 else
   u=b;
 while ((a%u !=0) || (b%u!=0))
   u--;
 return u;
```

```
}
```

2.2. Sử dụng hàm:

Sau khi đã tạo lập các hàm, trong chương trình chính cần sử dụng hàm, ta thực hiên bằng lệnh gọi hàm, đơn giản chỉ là tên hàm có hoặc không có các tham số thực như sau:
Cú pháp gọi hàm:

<Tên hàm>([Danh sách các tham số])

Lưu ý:

Trước khi sử dụng một hàm thì bạn phải khai báo kiểu giá trị của nó như cú pháp sau đây:

Kiểu tên hàm ();

Ví dụ: Viết chương trình cho phép tìm ước số chung lớn nhất của hai số tự nhiên.

```
#include<stdio.h>
unsigned int  ucln(unsigned  int a, unsigned  int b)
{
  unsigned  int u;
 if (a<b)
   u=a;
 else
   u=b;
  while ((a%u !=0) || (b%u!=0))
   u--;
 return u;
}
int main()
{
unsigned int a, b, UC;
   printf("Nhap a,b: ");scanf("%d%d",&a,&b);
  UC = ucln(a,b);
  printf("Uoc chung lon nhat la: ", UC);
 return 0;
}
```

Lưu ý: Việc gọi hàm là một phép toán, không phải là một phát biểu.

2.3. Nguyên tắc hoạt động của hàm:

Để hiểu thật kỹ về hàm và sử dụng nó, bạn cần nắm vững nguyên tắc hoạt động của hàm để điều khiển nó.

Khi gặp một lời gọi hàm thì hàm bắt đầu thực hiện. Dễ hiểu hơn, khi máy gặp một câu lệnh gọi hàm ở moot6j chỗ nào đó trong chương trình, máy sẽ tạm dời chỗ đó để chuyển đến hàm được gọi. Quá trình tiếp diên ra như sau:

- Nếu là hàm có tham số thì sẽ gán giá trị thực cho tham số hình thức tương ứng.

Máy bắt đầu thực hiện từ câu lệnh đầu tiên đến các câu lệnh khác teo sự sắp xếp các câu lệnh trong thân hàm.

- Khi gặp một câu lệnh return hoặc dấu } cuối cùng của thân hàm, máy sẽ thoát khỏi hàm để trở về chương trình gọi và thực hiện câu lệnh kế tiếp của chương trình này.

3. Truyền tham số

Đối với những hàm có tham số hình thức khi thực hiện phải truyền giá trị thực cho các tham số này. Có hai loại truyền tham số: truyền bằng trị và truyền bằng biến.

3.1. Truyền bằng trị:

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

Ví dụ 1: Giả sử ta muốn in ra nhiều dòng, mõi dòng 50 ký tự nào đó. Để đơn giản ta viết một hàm, nhiệm vụ của hàm này là in ra trên một dòng 50 ký tự nào đó. Hàm này có tên là InKT.

```
#include <stdio.h>
#include <conio.h>
void InKT(char ch)
{
 int i;
 for(i=1;i<=50;i++) printf("%c",ch)
 printf("\n");
}
int main()
{
  char c = 'A';
  InKT('*'); /* In ra 50 dau * */
 InKT('+');
 InKT(c);
 return 0;
}
```

Lưu ý:

Trong hàm InKT trên, biến ch gọi là tham số hình thức được truyền bằng giá trị (gọi là tham trị của hàm). Các tham trị của hàm coi như là một biến cục bộ trong hàm và chúng được sử dụng như là dữ liệu đầu vào của hàm.

Khi chương trình con được gọi thi hành, tham trị được cấp ô nhó và nhận giá trị là bản sao giá trị của tham số thực. Do đó, mặc dù tham trị cũng là biến nhưngviệc thay đổi giá trị của chúng không có nghĩa gì đối với bên ngoài hàm, không ảnh hưởng đến chương trình chính, nghĩa là không làm ảnh hưởng đến tham số thực tương ứng.

Ví dụ 2: Ta xét chương trình sau đây:

```
#include <stdio.h>
#include <conio.h>
int  hoanvi(int a, int b)
{
 int t;
 t=a;  /*doan này hoán vi giá tri cua 2 bien a, b*/
 a=b;
 b=t;
  printf("\Ben trong ham a=%d  , b=%d",a,b);
 return 0;
```

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

Kết quả thực hiện chương trình:

```
Nhap vao 2 so nguyen a, b:6 5

Truoc khi goi ham hoan vi a=6  ,b=5
Ben trong ham a=5  , b=6
Sau khi goi ham hoan vi a=6  ,b=5
```

Hình 1: Kết quả chương trình

Giải thích:

Nhập vào 2 số 6 và 5 (a=6, b=5)

Trước khi gọi hàm hoán vị thì a=6, b=5

Bên trong hàm hoán vị a=5, b=6

Khi ra khi hàm hoán vị thì a=6, b=5

* Lưu ý:

- Trong đoạn chương trình trên, nếu ta muốn sau khi kết thúc chương trình con giá trị của a, b thay đổi thì ta phải đặt tham số hình thức là các con trỏ, còn tham số thực tế là địa chỉ của các biến.
- Lúc này mọi sự thay đổi trên vùng nhớ được quản lý bởi con trỏ là các tham số hình thức của hàm thì sẽ ảnh hưởng đến vùng nhớ đang được quản lý bởi tham số thực tế tương ứng (cần chú ý rằng vùng nhớ này chính là các biến ta cần thay đổi giá trị).
- Người ta thường áp dụng cách này đối với các dữ liệu đầu ra của hàm.

Ví dụ: Xét chương trình sau đây:

```
#include <stdio.h>
#include <conio.h>
long  hoanvi(long *a, long *b)
/* Khai báo tham số hình thức *a, *b là các con trỏ kiểu long */
{
  long  t;
 t=*a;  /*gán nội dung của x cho t*/
 *a=*b;  /*Gán nội dung của b cho a*/
 *b=t;  /*Gán nội dung của t cho b*/
  printf("\n Ben trong ham a=%ld  , b=%ld",*a,*b);
/*In ra nội dung của a, b*/
 return 0;
}
```

```
int main()
{
  long a, b;
 clrscr();
  printf("\n Nhap vao 2 so nguyen a, b:");
 scanf("%ld%ld",&a,&b);
  printf("\n Truoc khi goi ham hoan vi a=%ld  ,b=%ld",a,b);
 hoanvi(&a,&b); /* Phi là a ch ca a và b */
  printf("\n Sau khi goi ham hoan vi a=%ld  ,b=%ld",a,b);
 getch();
 return 0;
}
```

Kết quả thực hiện chương trình:

```
Nhap vao 2 so nguyen a, b: 5 6

Truoc khi goi ham hoan vi a=5  ,b=6
Ben trong ham a=6  , b=5
Sau khi goi ham hoan vi a=6  ,b=5
```

Hình 2: Kết quả chương trình

Giải thích:

Nhập vào 2 số 5, 6 (a=5, b=6)

Trước khi gọi hàm hoán vị thì a=5, b=6

Trong hàm hoán vị (khi đã hoán vị) thì a=6, b=5

Khi ra khi hàm hoán vị thì a=6, b=6

Lưu ý: Kiểu con trỏ và các phép toán trên biến kiểu con trỏ sẽ nói trong phần sau.

3.2. Truyền bằng biến:

Theo thống kê ta nhận thấy đa số các trường hợp truyền tham số trong Turbo C được truyền bằng trị, còn nếu muốn truyền theo biến (truyền theo qui chiếu) tức là muốn truyền cả nội dung và địa chỉ của biến thì phải sử dụng biến con trỏ.

4. Các lệnh đơn nhằm kết thúc hàm và nhận giá trị trả về cho tên hàm:

Trong các hàm tự tạo của Turbo C thường hay sử dụng 2 câu lệnh return và exit.

4.1. Câu lệnh return:

Khi gặp lệnh return máy sẽ chuyển quyền điều khiển ra ngoài, không thực hiện các câu lệnh còn lại của hàm chứa câu lệnh này.

4.2. Câu lệnh exit:

Câu lệnh exit có công dụng chấm dứt thực hiện chương trình, trả điều khiển về hệ điều hành (nếu như chạy tập tin .EXE) hoặc màn hình soạn thảo của Turbo C.

Về câu lệnh return bạn đã thấy trong nhiều chương trình ví dụ, ỏ đay xin giới thiệu với các bạn về câu lệnh exit.

Câu lệnh exit có thể có tham số, nếu có nó sẽ chuyển cho MS-DOS.

❄Chương trình minh họa câu lệnh exit.

Bạn hãy nhập chương trình và đặt dteen tập tin là **THOAT.C** trong thư mục **C:\BAITAP\NGUON** sau đây:

VD06-10

```
main()
{
int N;
puts("\nMoi ban thu lenh EXIT của Turbo C");
printf("\nKhong thu nua, bam so 0 de dung");
scanf("%d",&N);
exit(N); /*lenh exit co tham so*/
}
```

Nếu trong môi trường Turbo C, bạn bấm tổ hợp phím **<Ctrl-F9>** để chạy chương trình này. Khi bạn nhập số (0) sẽ trở về màn hình soạn thảo.

+ Nhưng ở chương trình này, chúng tôi muốn đề cập đến mã thoát của MS-DOS vì vậy sau khi ghi chương trình vào trong tập tin (ở thư mục **C:\BAITAP\NGUON**), bạn bấm tổ hợp phím **<Alt-X>** để thoát về màn hình màn hình MS-DOS và giả sử thư mục hiện hành của bạn là **C:\BAITAP\ NGUON.**

Từ dấu nhắc của hệ điều hành, bạn dùng lệnh **copy con** để tạo tập tin **kt.bat** có nội dung như sau:

```
C:\BAITAP\NGUON  copy con kt.bat ↲
@ ECHO OFF
: LOOP
@ C:\BAITAP\DICH\VD06-10
IF ERRLEVERL 1 GOTO LOOP
@ ECHO ON
```

Sau đó bạn bấm phím **<F6>** hoặc tổ hợp phím <**Ctrl-Z**> để ghi nội dung trên tập tin KT.BAT vào thư mục NGUON.

Tiếp đến từ dấu nhắc của hệ điều hành bạn gõ vào tên tập tin **kt.bat** sẽ xuất hiện màn hình giống như bạn đã chạy chương trình **THOAT.C** trong môi trường Turbo C lúc trước.

Về công dụng của tập tin .BAT chắc bạn đã biết, ở đây chỉ xin giải thích lệnh **IF ERRLEVERL 1 GOTO LOOP.** Câu lệnh **IF** của MS-DOS cũng khảo sát điều kiện, ở dây là mã thoát **ERRLEVERL.** Nếu mã này là một vị trí khác (0) thì lệnh **GOTO** sẽ chuyển đến nhãn **LOOP** để rồi lại chạy tập tin **VD06-10.EXE** trong thư mục **DICH** như chúng tôi đã yêu cầu bạn tạo lập ngay từ trước. Cho tới khi nào bạn nhập số (0) thì máy mới thoát về dấu nhắc của hệ điều hành.

CHƯƠNG 5: MẢNG

Mã chương: MĐ10-05

Mục tiêu:

Sau khi học xong chương này sinh viên có khả năng:

- Hiểu khái niệm mảng.
- Khai báo được mảng một chiều, mảng hai chiều, mảng nhiều chiều.
- Biết cách gán giá trị cho mảng trực tiếp, gián tiếp.
- Vận dụng được mảng làm tham số cho hàm.
- Sắp xếp được mảng theo thứ tự giảm dần, tăng dần.

1. Trình bày khái niệm mảng trong C

Như bạn đã biết, mỗi biến chỉ có thể biểu diễn một giá trị. Muốn biểu diễn một dãy số hay một bảng số bạn có thể sử dụng nhiều biến nhưng rất bất tiện. Vì vậy hầu hết các ngôn ngữ lập trình đều thiết kế một kiểu dữ liệu chuyên dụng cho loại này gọi là kiểu mảng.

Vậy mảng là một tập hợp nhiều phần tử có cùng một kiểu giá trị và có chung một tên. Mõi phần tử của mảng biểu diễn được một giá trị. Có bao nhiêu kiểu dữ liệu (kiểu giá trị) thì cũng có bấy nhiêu kiểu mảng.

Ví dụ có các mảng sau :

```
int a[10], b[4][2]
Float x[5], y[3][3]
```

❄Giải thích:

- Mảng a[10]: Mảng 1chiều, có 10 phần tử thuộc kiểu số nguyên (int) được đánh số như sau :

a[0], a[1],...a[9].

Như vậy mảng a có thể biểu diễn một dãy 10 phần tử, mỗi phần tử a[1] chứa 1 trị nguyên.

- Mảng b[4][1]: Mảng 2 chiều b, có 8 phần tử thuộc kiểu nguyên (int) gồm 4 dòng, 2 cột được sắp xếp như sau:

b[0][0]	b[0][1]
b[1][0]	b[1][1]
b[2][0]	b[2][1]
b[3][0]	b[3][1]

Mỗi phần tử b[i][j] biểu diễn một giá trị kiểu nguyên.

- Mảng X[5]: mảng 1 chiều X, có 5 phần tử thuộc kiểu số thực (float) được đánh số như sau:

X[0], X[1], X[2], X[3], X[4]

Mỗi phần tử X[i] biểu diễn một giá trị kiểu số thực (float). Mảng X biểu diễn được 5 số thực.

- Mảng y[3][3]: Mảng 2 chiều Y gồm 3 dòng, 3 cột, có 9 phần tử, mỗi phần tử y[i][j] chứa một giá trị kiểu số thực (float) được sắp xếp như sau:

y[0][0]	y[0][1]	y[0][2]
y[1][0]	y[1][1]	y[1][2]
y[2][0]	y[2][1]	y[2][2]

Như vậy để xác định một mảng cần phải định rõ:

+ Kiểu dữ liệu của mảng: như int, float, double.

+ Tên mảng: Là các ký tự từ a đến z, viết thường hoặc viết hoa.
+ Số chiều và kích thước của mỗi chiều đặt trong cặp dấu ngoặc vuông [].

2. Cú pháp khai báo mảng và các cách gán giá trị cho mảng.

2.1. Mảng một chiều:

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

2.1.1. Khai báo mảng với số phần tử xác định (khai báo tường minh):

Cú pháp:

<Kiểu> <Tên mảng ><[số phần tử]>;

Ý nghĩa:

Tên mảng: đây là môt cái tên đặt đúng theo quy tắc đặt tên của danh biểu. Tên này cũng mang ý nghĩa là tên biến mảng.

Số phần tử: là một hằng số nguyên, cho bit số lượng phần tử tối đa trong mảng là bao nhiêu (hay kích thước của mảng là gì).

Kiểu: mọi phần tử của mảng có dữ liệu thuộcc kiểu gì.

Ở đây, ta khai báo một biến mảng gồm có số phần tử phần tử, phần tử thứ nhất là tên mảng [0], phần tử cuối cùng là tên mảng [số phần tử -1]

Ví dụ:

int a[10]; / Khai báo biến mang tên a, phần tử thứ nhất là a[0], phần tử cuối cùng là a[9].*/*

Ta có thể coi mảng a là một dãy liên tiếp các phần tử trong bộ nhớ như sau:

Vị trí	0	1	2	3	4	5	6	7	8	9
Tên phần tử	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

Hình 1: Hình ảnh mảng a trong bộ nhớ

Hình 1: Hình ảnh mảng a trong bộ nhớ

2.1.2. Khai báo mảng với số phần tử không xác định (khai báo không tường minh):

Cú pháp:

<Kiểu> <Tên mảng> <[]>;

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

- **Vừa khai báo vừa gán trị:**

Cú pháp:

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

Nếu vừa khai báo vừa gán giá trị thì mặc nhiên C sẽ hiểu số phần tử của mảng là số giá trị mà chúng ta gán cho mảng trong cặp dấu {}. Chúng ta có thể sử dụng hàm sizeof() để lấy số phần tử của mảng như sau:

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

Ví dụ: vừa khai báo vừa gán trị cho mảng

```
#include<stdio.h>
main()
{
int i, A[] = {23,65,7,89,76};
```

```
clrscr();
for(i=0;i<=4;i++)
printf("A[%d]=%d\t",i,A[i]);
getch();
}
```

Sau khi chạy chương trình, nhan được kết quả sau:

A[0]=23 A[1]=65 A[2]=75 A[3]=89 A[4]=7

- **Khai báo mảng là tham số hình thức của hàm:**

Trong trường hợp này không cần chỉ định số phần tử của mảng là bao nhiêu.

<Kiểu> <Tên hàm> (<kiểu> <Tên mảng[]>,…);

2.1.3. Truy xuất đến từng phần tử của mảng:

Mỗi phần tử của mảng được truy xuất thông qua Tên biến mang theo sau là chỉ số nằm trong cặp dấu ngoặc vuông [ ]. Chẳng hạn a[0] là phần tửu ưu tiên của mảng a được khai báo trên. Chỉ số của phần tử mảng là một biểu thức mà giá trị là kiểu số nguyên.

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

Ví dụ 1:

int a[10];

Trong khai báo này, việc truy xuất các phần tử được chỉ ra trong hình 1. Chẳng hạn phần tử thứ 2 (có vị trí 1) là a[1]…

Ví dụ 2: Vừa khai báo và gán trị cho 1 mảng 1 chiều các số nguyên. In mảng số nguyên này lên màn hình.

Giả sử ta đã biết số phần tử của mảng là n; việc hiển thị 1 giá trị số nguyên lên màn hình ta cần sử dụng hàm printf() với định dạng %d, tổng quát hóa lên nếu muốn hiển thị lên màn hình giá trị của n số nguyên, ta cần gọi hàm printf() đúng n lần. Như vậy trong trường hợp này ta sử dụng 1 vòng lặp in ra giá trị các phần tử.

Ta có đoạn chương trình sau:

```
#include <stdio.h>
#include <conio.h>
int main()
{
 int n,i,j,tam;
 int dayso[]={66,65,69,68,67,70};
 clrscr();
n=sizeof(dayso)/sizeof(int); /*Lấy số phần tử*/
  printf("\n Noi dung cua mang ");
 for (i=0;i<n;i++)
     printf("%d ",dayso[i]);
 return 0;
}
```

Ví dụ 3:

Đổi một số nguyên dương thập phân thành số nhị phân. Việc chuyển đổi này đượ thực hiện bằng cách lấy số đó chia liên tiếp cho 2 cho tới khi bằng 0 và lấy các số đó theo chiều ngược lại tạo thành số nhị phân. Ta sẽ dùng mảng một chiều lưu lại các số đó. Chương trình cụ thể như sau:

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

```
#include<stdio.h>
 int main()
{
  unsigned int N;
  unsigned int Du;
  unsigned int NhiPhan[20],K=0,i;
  printf("Nhap vao so nguyen N= ");scanf("%d",&N);
 do
 {
    Du=N % 2;
   NhiPhan[K]=Du; /* Lưu số dư vào mảng ở vị trí K*/
   K++;  /* Tăng K lên để lần kế lưu vào vị trí kế*/
    N = N/2;
 } while(N>0);
  printf("Dang nhi phan la: ");
 for(i=K-1;i>=0;i--)
   printf("%d",NhiPhan[i]);
 getch();
 return 0;
}
```

Ví dụ 4:

Nhập vào một dãy n số và sắp xếp các số theo thứ tự tăng. Đây là một bài toán có ứng dụng rộng rãi trong nhiều lĩnh vực. Có rất nhiều giải thuật sắp xếp. Một trong số đó được mô tả như sau:

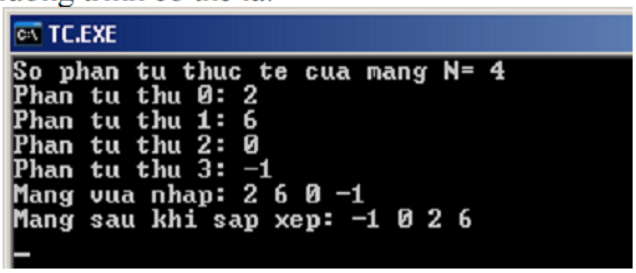
Đầu tiên đưa phần tử thứ nhất so sánh với các phần tử còn lại, nếu nó lớn hơn một phần tử đang so sánh thì đổi chỗ hai phần tử cho nhau. Sau đó tiếp tục so sánh phần tử thứ hai với các phần tử từ thứ ba trở đi ... cứ tiếp tục như vậy cho đến phần tử thứ n-1.

Chương trình sẽ được chia thành các hàm Nhap (Nhập các số), SapXep (Sắp xếp) và InMang (In các số); các tham số hình thức của các hàm này là 1 mảng không chỉ định rõ số phần tử tối đa, nhưng ta cần có thêm số phần tử thực tế được sử dụng của mảng là bao nhiêu, đây là một giá trị nguyên.

```
#include<conio.h>
#include<stdio.h>
void Nhap(int a[],int N)
{
 int i;
  for(i=0; i< N; i++)
 {
    printf("Phan tu thu %d: ",i);scanf("%d",&a[i]);
 }
}
void InMang(int a[], int N)
{
 int i;
  for (i=0; i<N;i++)
   printf("%d ",a[i]);
```

```
 printf("\n");
}
void SapXep(int a[], int N)
{
 int t,i;
 for(i=0;i<N-1;i++)
   for(int j=i+1;j<N;j++)
    if (a[i]>a[j])
    {
     t=a[i];
     a[i]=a[j];
     a[j]=t;
    }
}
int main()
{
  int b[20], N;
  printf("So phan tu thuc te cua mang N= ");
scanf("%d",&N);
 Nhap(b,N);
  printf("Mang vua nhap: ");
 InMang(b,N);
  SapXep(b,N); /* Goi hàm sap xep*/
  printf("Mang sau khi sap xep: ");
 InMang(b,N);
 getch();
 return 0;
}
```

Kết quả chạy chương trình có thể là:



Hình 2: Kết quả chương trình

2.2. Mảng nhiều chiều:

Là mảng có từ hai phần tử trở lên, điều đó có nghĩa mỗi phần tử của mảng là một mảng khác.Dùng để lưu các ma trận, các tọa độ 2 chiều, 3 chiều,Phần dưới đây là các vấn đề liên quan đến mảng 2 chiều; các mảng 3, 4,… chiều thì tương tự (chỉ cần tổng quát hóa lên).

2.2.1. Khai báo mảng hai chiều với số phần tử xác định (Khai báo tường minh):

Cú pháp:

<Kiểu> <Tên mảng><[Số phần tử chiều 1]><[Số phần tử chiều 2]>…;

Ví dụ: Người ta cần lưu trữ thông tin của một ma trận gồm các số thực. Lúc này ta có thể khai báo một mảng 2 chiều như sau:

float m[8][9]; /* Khai báo mảng 2 chiều có 8*9 phần tử là số thực*/

Trong trường hợp này, ta đã khai báo cho một ma trận có tối đa là 8 dòng, mỗi dòng có tối đa là 9 cột. Hình ảnh của ma trận này được cho trong hình 2:

Dòng\Cột	0	1	2	3	4	5	6	7	8
0	m[0][0]	m[0][1]	m[0][2]	m[0][3]	m[0][4]	m[0][5]	m[0][6]	m[0][7]	m[0][8]
1	m[1][0]	m[1][1]	m[1][2]	m[1][3]	m[1][4]	m[1][5]	m[1][6]	m[1][7]	m[1][8]
2	m[2][0]	m[2][1]	m[2][2]	m[2][3]	m[2][4]	m[2][5]	m[2][6]	m[2][7]	m[2][8]
3	m[3][0]	m[3][1]	m[3][2]	m[3][3]	m[3][4]	m[3][5]	m[3][6]	m[3][7]	m[3][8]
4	m[4][0]	m[4][1]	m[4][2]	m[4][3]	m[4][4]	m[4][5]	m[4][6]	m[4][7]	m[4][8]
5	m[5][0]	m[5][1]	m[5][2]	m[5][3]	m[5][4]	m[5][5]	m[5][6]	m[5][7]	m[5][8]
6	m[6][0]	m[6][1]	m[6][2]	m[6][3]	m[6][4]	m[6][5]	m[6][6]	m[6][7]	m[6][8]
7	m[7][0]	m[7][1]	m[7][2]	m[7][3]	m[7][4]	m[7][5]	m[7][6]	m[7][7]	m[7][8]

Hình 2: Ma trận được mô tả là 1 mảng 2 chiều

Hình 3: Ma trận

2.2.2. Khai báo mảng hai chiều với số phần tử không xác định (Khai báo không tường minh):

Để khai báo mảng 2 chiều không tường minh, ta vẫn phải chỉ ra số phần tử của chiều thứ hai (chiều cuối cùng).

Cú pháp:

<Kiểu> <Tên mảng> <[]><[Số phần tử chiều 2]>;

Cách khai báo này cũng được áp dụng trong trường hợp vừa khai báo, vừa gán trị hay đặt mảng 2 chiều là tham số hình thức của hàm.

<Kiểu> <tên mảng> <[][Chỉ số sau]> = {Giá trị gần};

Giá trị gán của các phần tử là các cặp giá trị được rào trong cặp dấu móc {} và phân cách với nhau bởi dấu phẩy (,).

Ví dụ:

Khai báo không tường minh, vừa khai báo vừa gán trị cho mảng.

```
#include<stdio.h>
main()
{
int a[][2] = {{4,5},{6,7},{12,6},{6,9},{5,6}};
int b[][2]= {{6,7},{8,9},{9,6},{5,6},{67,45}};
int k,j;
clrscr();
for(k=0;k<5;k++)
{
for(j=0;j<2;j++)
{
printf("na[%d,%d]=%d",k,j,a[k][j]);
printf("nb[%d,%d]=%d",k,j,b[k][j]);
}
printf("\n");
}
getch();
```

```
}
```
Chạy chương trình được kết quả sau:

a[0,0]=1 a[0,1]=2 a[0,2]=3 a[0,3]=4

a[1,0]=5 a[1,1]=6 a[1,2]=7 a[1,3]=8

2.2.3. Truy xuất đến số phần tử mảng hai chiều:

Ta có thể truy xuất một phần tử của mảng hai chiều bằng cách viết ra tên mảng theo sau là hai chỉ số đặt trong hai cặp dấu ngoặc vuông. Chẳng hạn ta viết m[2][3].

Với cách truy xuất theo cách này, tên mảng[Chỉ số 1][Chỉ số 2] có thể coi là biến có kiểu được chỉ ra trong khai báo biến mảng**.**

Ví dụ 1:

Viết chương trình cho phép nhập 2 ma trận a, b có m dòng n cột, thực hiện phép toán cộng hai ma trận a,b và in ma trận kết quả lên màn hình.

Trong ví dụ này, ta sẽ sử dụng hàm làm ngắn gọn hơn chương trình của ta.Ta sẽ viết các hàm: nhập 1 ma trận từ bàn phím, hiển thị ma trận lên màn hình, cộng 2 ma trận.

```
#include<conio.h>
#include<stdio.h>
void Nhap(int a[][10],int M,int N)
{
 int i,j;
 for(i=0;i<M;i++)
  for(j=0; j<N; j++){
      printf("Phan tu o dong %d cot %d: ",i,j);
scanf("%d",&a[i][j]);
  }
}
void InMaTran(int a[][10], int M, int N)
{
 int i,j;
 for(i=0;i<M;i++){
    for(j=0; j< N; j++)
  printf("%d ",a[i][j]);
  printf("\n");
 }
}
/* Cong 2 ma tran A & B ket qua la ma tran C*/
void CongMaTran(int a[][10],int b[][10],int M,int N,int c[][10]){
 int i,j;
 for(i=0;i<M;i++)
  for(j=0; j<N; j++)
   c[i][j]=a[i][j]+b[i][j];
}
int main()
{
  int a[10][10], b[10][10], M, N;
  int c[10][10];/* Ma tran tong*/
  printf("So dong M= "); scanf("%d",&M);
```

```
 printf("So cot M= "); scanf("%d",&N);
 printf("Nhap ma tran A\n");
Nhap(a,M,N);
printf("Nhap ma tran B\n");
Nhap(b,M,N);
 printf("Ma tran A: \n");
InMaTran(a,M,N);
 printf("Ma tran B: \n");
InMaTran(b,M,N);
CongMaTran(a,b,M,N,c);
 printf("Ma tran tong C:\n");
InMaTran(c,M,N);
getch();
return 0;
}
```

Ví dụ 2:

Nhập vào một ma trận 2 chiều gồm các số thực, in ra tổng của các phần tử trên đường chéo chính của ma trận này.

Ta nhận thấy rằng giả sử ma trận a có M dòng, N cột thì các phần tử của đường chéo chính là các phần tử có dạng: a[i][i] vi i [0…min(M,N)-1].

```
#include<conio.h>
#include<stdio.h>
int main()
{
 float a[10][10], T=0;
 int M, N, i,j, Min;
clrscr();
 printf("Ma tran co bao nhieu dong? ");scanf("%d",&M);
 printf("Ma tran co bao nhieu cot? ");scanf("%d",&N);
 for(i=0;i<M;i++)
 for(j=0; j<N; j++)
 {
    printf("Phan tu o dong %d cot %d: ",i,j);
scanf("%f",&a[i][j]);
 }
 printf("Ma tran vua nhap: \n");
for(i=0;i<M;i++)
{
  for(j=0; j< N; j++)
 printf("%.2f  ",a[i][j]);
printf("\n");
}
 Min=(M>N) ? N: M; /* Tìm giá tr nh nht ca M & N*/
for(i=0;i<Min;i++)
 T=T+a[i][i];
 printf("Tong cac phan tu o duong cheo chinh la: %f",T);
getch();
```

```
return 0;
}
```

3. Mảng và tham số của hàm.

3.1. Địa chỉ các phần tử của mảng:

Như bạn đã biết mỗi phần tử của mảng cũng là một biến; mỗi biến đều xác định bởi địa chỉ và nội dung của nó. Bạn sẽ nghiên cứu vấn đề này ở chương biến con trỏ, nên phần này chỉ giới thiệu sơ lược vài chi tiết mà Turbo C đã quy định để bạn lưu ý trong lúc lập trình.

- Cho địa chỉ của biến **&<biến>.**
- Cho nội dung của biến con trỏ ***<biến>**.

Bạn hãy chạy chương trình ví dụ sau đây để biết cách truy xuất địa chỉ và nội dung của biến con trỏ.

```
#include<stdio.h>
main()
{
int contro=453;
printf("\nVi tri cua bien con tro: %p", &contro);
printf("\nNoi dung cua bien con tro: %d", contro);
getch();
}
```

Sau khi chạy chương trình, bạn sẽ thấy:

Vi tri cua bien con tro: FFD2

Noi dung cua bien con tro: 453

- Đối với mảng thì tên mảng a đồng nghĩa với &a[0] địa chỉ byte đầu tiên của mảng.
- Còn a + i cho ra địa chỉ a[i].

3.2. Truyền tham số là mảng:

Để truyền tham số là mảng cho hàm bạn thực hiện như sau:

- Gọi tên mảng.
- Truyền tham số theo biến vì chứa địa chỉ byte đầu tiên của mảng.

Những câu lệnh dưới đây minh hoạ đều đó.

```
#include<stdio.h>
main()
{
int a;
char b[80];
...
scanf("%d", &a);
...
scan("%s",b);
...
}
```

Bạn cần lưu ý đối với việc truyền cho mảng không có dấu ampersand(&) như truyền cho các biến thông thường mà bạn đã gặp từ trước tới nay.

3.3. Sắp xếp mảng:

Trong thực tế, ta thường gặp những công việc cần phải sắp xếp theo một thứ tự nào đó, chẳng hạn danh sách học sinh trong lớp có thể được sắp xếp theo HO hoặc ĐIỂM TRUNG BÌNH. Vì sắp xếp thứ tự một mảng giúp ta dễ dàng tìm kiếm các phần tử của mảng.

Từ trước tới nay người ta đã tìm ra nhiều phương pháp sắp xếp mảng như sau:

- Phương pháp chèn vào thẳng (straightinsertion).
- Phương pháp chèn vào và chia đôi hay còn gọi là phương pháp chèn vào nhị phân (binaryinsertion).
- Phương pháp chọn lựa thẳng (straightselection)
- Các phương pháp đổi chổ gồm:
  + Phương pháp nổi bọt (subble sort)
  + Phương pháp rủng (shakersort)
- Các phương pháp sắp xếp khác gồm có:
  + Phương pháp shell.
  + Phương pháp Heapsort.
  + Phương pháp quicksort.

CHƯƠNG 6: CHUỖI KÍ TỰ

Mã chương: MĐ10-06

Mục tiêu:

Sau khi học xong chương này sinh viên có khả năng:

- Hiểu được thế nào là chuỗi ký tự.
- Khai báo biến chuỗi.
- Biết cách nhập vào một chuỗi ký tự cho chương trình trước và sau khi runtime.
- Hiểu và áp dụng được các phép toán trên chuỗi.
- Vận dụng được các hàm xử lý chuỗi để xử lý.

1. Khái niệm chuỗi kí tự

Chuỗi ký tự là một dãy gồm các ký tự hoặc một mảng các ký tự được kết thúc bằng ký t ự '\0' (còn được gọi là ký tự NULL trong bảng mã Ascii).
Các hằng chuỗi ký tự được đặt trong cặp dấu nháy kép "".

2. Khai báo biến chuỗi

2.1. Khai báo theo mảng:

Cú pháp:

char <Biến> [Chiều dài tối đa]

Ví dụ: Trong chương trình, ta có khai báo:

char Ten[12];

Trong khai báo này, bộ nhớ sẽ cung cấp 12+1 bytes lưu trữ nội dung của chuỗi ký tự Ten; byte cuối cùng lưu trữ ký tự '\0' chấm dứt chui.

Ghi chú:

- Chiều dài tối đa của biến chuỗi là một hằng nguyên nằm trong khoảng từ 1 đến 255 bytes.
- Chiều dài tối đa không nên khai báo thừa để tránh lãng phí bộ nhớ, nhưng cũng không nên khai báo thiếu.

2.2. Khai báo theo con trỏ:

Cú pháp:

char *<Biến>

Ví dụ: Trong chương trình, ta có khai báo:

*char *Ten;*

Trong khai báo này, bộ nhớ sẽ dành 2 byte lưu trữ địa chỉ của biến con trỏ Ten đang chỉ đến, chưa cung cấp nơi để lưu trữ dữ liệu. Muốn có chỗ để lưu trữ dữ liệu, ta phải gọi đến hàm malloc() hoặc calloc() có trong "alloc.h", sau đó mới gán dữ liệu cho biến.

3. Biến chuỗi và hằng chuỗi, nhập giá trị chuỗi cho chương trình

3.1. Vừa khai báo vừa gán trị:

Cú pháp:

char <Biến>[]=<"Hằng chuỗi">

Ví dụ:

```
#include<stdio.h>
#include<conio.h>
int main()
```

```
{
  char Chuoi[]="Mau nang hay la mau mat em" ;
   printf("Vua khai bao vua gan tr : %s",Chuoi) ;
  getch();
   return 0;
 }
```

3.2. Nhập xuất chuỗi:

3.2.1. Nhập chuỗi từ bàn phím:

Để nhập một chuỗi ký tự từ bàn phím, ta sử dụng hàm gets()

Cú pháp:

gets(<Biến chuỗi>)

Ví dụ:
```
char Ten[20];
gets(Ten);
```

Ta cũng có thể sử dụng hàm scanf() nhập dữ liệu cho biến chuỗi, tuy nhiên lúc này ta chỉ có thể nhập được một chuỗi không có dấu khoảng trắng.

Ngoài ra, hàm cgets() (trong conio.h) cũng được sử dụng để nhập chuỗi.

3.2.2. Xuất chuỗi lên màn hình:

Để xuất một chuỗi (biểu thức chuỗi) lên màn hình, ta sử dụng hàm puts().

Cú pháp:

puts(<Biểu thức chuỗi>)

Ví dụ: Nhập vào một chuỗi và hiển thị trên màn hình chuỗi vừa nhập.

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
int main()
{
 char Ten[12];
 printf("Nhap chuoi: ");gets(Ten);
 printf("Chuoi vua nhap: ");puts(Ten);
 getch();
 return 0;
}
```

Ngoài ra, ta có thể sử dụng hàm printf(), cputs() (trong conio.h) hiển thị chuỗi lên màn hình.

4. Các hàm làm việc với chuỗi:

4.1. Cộng chuỗi - Hàm strcat():

Cú pháp:

char *strcat(char *des, const char *source)

Hàm này có tác dụng ghép chuỗi nguồn vào chuỗi đích.

Ví dụ: Nhập vào họ lót và tên của một người, sau đó in cả họ và tên cửa họ lên màn hình.

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
int main()
{
```

```
char HoLot[30], Ten[12];
printf("Nhap Ho Lot: ");gets(HoLot);
printf("Nhap Ten: ");gets(Ten);
strcat(HoLot,Ten); /* Ghep Ten vao HoLot*/
printf("Ho ten la: ");puts(HoLot);
getch();
return 0;
}
```

4.2. Xác nhận độ dài chuỗi - Hàm strlen():

Cú pháp:

```
int strlen(const char* s)
```

Ví dụ: Sử dụng hàm strlen xác nhận độ dài một chuỗi nhập từ bàn phím.

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
int main(){
 char Chuoi[255];
 int Dodai;
 printf("Nhap chuoi: ");gets(Chuoi);
 Dodai = strlen(Chuoi)
 printf("Chuoi vua nhap: ");puts(Chuoi);
 printf("Co do dai %d",Dodai);
 getch();
 return 0;
 }
```

4.3. Đổi một ký tự thường thành ký tự hoa - Hàm toupper():

Hàm toupper() (trong ctype.h) được dùng để chuyển đổi một ký tự thường thành ký tự hoa.

Cú pháp:

```
char toupper(char c)
```

4.4. Đổi chuỗi chữ thường thành chuỗi chữ hoa, hàm strupr():

Hàm struppr() được dùng để chuyển đổi chuỗi chữ thường thành chuỗi chữ hoa, kết quả trả về cửa hàm là một con trỏ chỉ đến địa chỉ chuỗi được chuyển đổi.

Cú pháp:

```
char  *strupr(char *s)
```

Ví dụ: Viết chương trình nhập vào một chuỗi ký tự từ bàn phím. Sau đó sử dụng hàm strupr() để chuyển đổi chúng thành chuỗi chữ hoa.

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
int main()
{
 char Chuoi[255],*s;
 printf("Nhap chuoi: ");gets(Chuoi);
 s=strupr(Chuoi) ;
 printf("Chuoi chu hoa: ");puts(s);
 getch();
```

```
return 0;
}
```

4.5. Đổi chuỗi chữ hoa thành chuỗi chữ thường, hàm strlwr():

Muốn chuyển đổi chuỗi chữ hoa thành chuỗi toàn chữ thường, ta sử dụng hàm strlwr(), các tham số của hàm tương tự như hàm strupr().

Cú pháp:

```
char   *strlwr(char *s)
```

4.6. Sao chép chuỗi, hàm strcpy():

Hàm này được dùng sao chép toàn bộ nội dung cửa chuỗi nguồn vào chuỗi đích.

Cú pháp:

```
char  *strcpy(char *Des, const char *Source).
```

Ví dụ: Viết chương trình cho phép chép toàn bộ chuỗi nguồn vào chuỗi đích.

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
int main()
{
char Chuoi[255],s[255];
printf("Nhap chuoi: ");gets(Chuoi);
strcpy(s,Chuoi);
printf("Chuoi dich: ");puts(s);
getch();
return 0;
}
```

4.7. Sao chép một phần chuỗi, hàm strncpy():

Hàm này cho phép chép n ký tư ưu tiên của chuỗi nguồn sang chuỗi đích

Cú pháp:

```
char *strncpy(char *Des, const char *Source, size_t n)
```

4.8. Trích một phần chuỗi, hàm strchr():

Để trích một chuỗi con của một chuỗi ký tự bắt đầu từ một ký tự được chỉ định trong chuỗi cho đến hết chuỗi, ta sử dụng hàm strchr().

Cú pháp :

```
char *strchr(const char *str, int c)
```

Ghi chú:

- Nếu ký tự đã chỉ định không có trong chuỗi, kết quả trả về là NULL.
- Kết quả trả về của hàm là một con trỏ, con trỏ này chứa đến ký tự c được tìm thấy đầu tiên trong chuỗi str.

4.9. Tìm kiếm nội dung chuỗi, hàm strstr():

Hàm strstr() được sử dụng để tìm kiếm sự xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1.

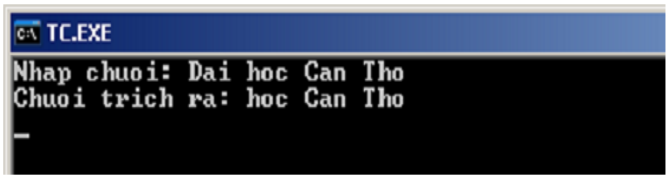
Cú pháp:

```
char  *strstr(const char *s1, const char *s2)
```

Kết quả trả về của hàm là một con trỏ chỉ đến phần tử đầu tiên của chuỗi s1 có chứa chuỗi s2 hoặc giá trị NULL nếu chuỗi s2 không có trong chuỗi s1.

Ví dụ: Viết chương trình sử dụng hàm strstr() để lấy ra một phần của chuỗi gốc bắt đầu từ chuỗi "hoc".

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
int main()
{
 char Chuoi[255],*s;
 printf("Nhap chuoi: ");gets(Chuoi);
 s=strstr(Chuoi,"hoc");
 printf("Chuoi trich ra: ");puts(s);
 getch();
 return 0;
}
```



Hình 1: Kết quả chương trình

4.10. So sánh chuỗi, hàm strcmp():

Để so sánh hai chuỗi theo từng ký tự trong bảng mã Ascii, ta có thể sử dụng hàm strcmp().

Cú pháp:

int strcmp(const char *s1, const char *s2)

Hai chuỗi s1 và s2 được so sánh với nhau, kết quả trả về là một số nguyên (số này có được bằng cách lấy ký tự của s1 trừ ký tự của s2 tại vị trí đầu tiên xảy ra sự khác nhau).

Nếu kết quả là số âm, chuỗi s1 nhỏ hơn chuỗi s2.

Nếu kết quả là 0, hai chuỗi bằng nhau.

Nếu kết quả là số dương, chuỗi s1 lớn hơn chuỗi s2.

4.11. So sánh chuỗi, hàm stricmp():

Hàm này thực hiện việc so sánh trong n ký tự đầu tiên của 2 chuỗi s1 và s2, giữa chữ thường và chữ hoa không phân biệt.

Cú pháp:

int stricmp(const char *s1, const char *s2)

Kết quả trả về tương tự như kết quả trả về cửa hàm strcmp().

4.12. Khởi tạo chuỗi, hàm memset():

Hàm này được sử dụng để đặt n ký tự đầu tiên của chuỗi là ký tự c.

Cú pháp:

memset(char *Des, int c, size_t n)

4.13. Đổi từ chuỗi ra số, hàm atoi(), atof(), atol() (trong stdlib.h):

Để chuyển đổi chuỗi ra số, ta sử dụng các hàm trên.

Cú pháp :

int atoi(const char *s): chuyển chuỗi thành số nguyên.

long atol(const char *s): chuyển chuỗi thành số nguyên dài

float atof(const char *s): chuyển chuỗi thành số thực.

Nếu chuyển đổi không thành công, kết quả trả về của các hàm là 0.

Ngoài ra, thư viện string.h còn hỗ trợ các hàm xử lý chuỗi khác, ta có thể đọc thêm trong phần trợ giúp.

4.14. Bài tập thực hành

❖ ***Mục đích yêu cầu***

Đi sâu vào kiểu dữ liệu chuỗi và các phép toán trên chuỗi.

❖ ***Nội dung***

1. Viết chương trình nhập một chuỗi ký tự từ bàn phím, xuất ra màn hình mã Ascii của từng ký tự có trong chuỗi.

2. Viết chương trình nhập một chuỗi ký tự từ bàn phím, xuất ra màn hình chuỗi đảo ngược của chuỗi đó.

Ví dụ đảo của "abcd egh" là "hge dcba".

3. Viết chương trình nhập một chuỗi ký tự và kiểm tra xem chuỗi đó có đối xứng không.

Ví dụ: Chuỗi ABCDEDCBA là chuỗi đối xứng.

4. Nhập vào một chuỗi bất kỳ, hãy đếm số lần xuất hiện của mỗi loại ký tự.

5. Viết chương trình nhập vào một chuỗi.

- In ra màn hình từ bên trái nhất và phần còn lại của chuỗi.

Ví dụ: "Nguyễn Văn Minh" in ra thành: Nguyễn Văn Minh

- In ra màn hình từ bên phải nhất và phần còn lại của chuỗi.

Ví dụ: "Nguyễn Văn Minh" in ra thành: Minh Nguyễn Văn

6. Viết chương trình nhập vào một chuỗi rồi xuất chuỗi đó ra màn hình dưới dạng mỗi từ một dòng.

Ví dụ: "Nguyễn Văn Minh"

In ra:

Nguyễn

Văn

Minh

7. Viết chương trình nhập vào một chuỗi, in ra chuỗi đảo ngược của nó theo từng từ.

Ví dụ: chuỗi "Nguyễn Văn Minh" đảo thành "Minh Văn Nguyễn"

8. Viết chương trình đổi số tiền từ số thành chữ.

9. Viết chương trình nhập vào họ và tên của một người, cắt bỏ các khoảng trống không cần thiết (nếu có), tách tên ra khỏi họ và tên, in tên lên màn hình. Chú ý đến trường hợp cả họ và tên chỉ có một từ.

10. Viết chương trình nhập vào họ và tên của một người, cắt bỏ các khoảng trắng bên phải, trái và các khoảng trắng không có nghĩa trong chuỗi. In ra màn hình toàn bộ họ tên người đó dưới dạng chữ hoa, chữ thường.

11. Viết chương trình nhập vào một danh sách họ và tên của n người theo kiểu chữ thường, đổi các chữ cái đầu của họ, tên và chữ lót của mỗi người thành chữ hoa. In kết quả lên màn hình.

12. Viết chương trình nhập vào một danh sách họ và tên của n người, tách tên từng người ra khỏi họ và tên rồi sắp xếp danh sách tên theo thứ tự từ điển. In danh sách họ và tên sau khi đã sắp xếp.

CHƯƠNG 7: BIẾN CON TRỎ

Mã bài: MĐ10-07

Giới thiệu

Bài học này cung cấp cho người học những kiến thức sau:

- ✓ Khái niệm về biến con trỏ.
- ✓ Cơ chế làm việc của biến con trỏ với cấu trúc dữ liệu kiểu mảng.
- ✓ Viết chương trình sử dụng biến con trỏ với cấu trúc dữ liệu kiểu mảng.

Mục tiêu:

- Hiểu được biến con trỏ.
- Biết được cách làm việc của biến con trỏ với cấu trúc dữ liệu kiểm mảng.
- Viết được chương trình sử dụng biến con trỏ với cấu trúc dữ liệu kiểu mảng.
- Tính cách suy luận. tư duy logic

1. Biến con trỏ

- Tác dụng của biến con trỏ;
- Khai báo được biến con trỏ;

Một **con trỏ** là một biến, nó chứa địa chỉ vùng nhớ của một biến khác, chứ không lưu trữ giá trị của biến đó. Nếu một biến chứa địa chỉ của một biến khác, thì biến này được gọi là **con trỏ** đến biến thứ hai kia. Một con trỏ cung cấp phương thức gián tiếp để truy xuất giá trị của các phần tử dữ liệu. Xét hai biến var1 và var2, var1 có giá trị 500 và được lưu tại địa chỉ 1000 trong bộ nhớ. Nếu var2 được khai báo như là một con trỏ tới biến var1, sự biểu diễn sẽ như sau:

Vị trí Bộ nhớ	**Giá trị lưu trữ**	**Tên biến**
1000	500	var1
1001		
1002		
.		
.		
1108	1000	var2

Ở đây, var2 chứa giá trị 1000, đó là địa chỉ của biến var1.

Các con trỏ có thể trỏ đến các biến của các kiểu dữ liệu cơ sở như **int**, **char**, hay **double** hoặc dữ liệu có cấu trúc như **mảng**.

Nếu một biến được sử dụng như một con trỏ, nó phải được khai báo trước. Câu lệnh khai báo con trỏ bao gồm một kiểu dữ liệu cơ bản, một dấu *, và một tên biến. Cú pháp tổng quát để khai báo một biến con trỏ như sau:

```
type *name;
```

Ở đó **type** là một kiểu dữ liệu hợp lệ bất kỳ, và **name** là tên của biến con trỏ. Câu lệnh khai báo trên nói với trình biên dịch là **name** được sử dụng để lưu địa chỉ của một biến có kiểu dữ liệu **type**. Trong câu lệnh khai báo, * xác định rằng một biến con trỏ đang được khai báo.

Trong ví dụ của **var1** và **var2** ở trên, vì **var2** là một con trỏ giữ địa chỉ của biến **var1** có kiểu **int**, nó sẽ được khai báo như sau:

```
int *var2;
```

Bây giờ, **var2** có thể được sử dụng trong một chương trình để trực tiếp truy xuất giá trị của **var1**. Nhớ rằng, **var2** không phải có kiểu dữ liệu **int** nhưng nó là một con trỏ trỏ đến một biến có kiểu dữ liệu int.

Kiểu dữ liệu cơ sở của con trỏ xác định kiểu của biến mà con trỏ trỏ đến. Về mặt kỹ thuật, một con trỏ có kiểu bất kỳ có thể trỏ đến bất kỳ vị trí nào trong bộ nhớ. Tuy nhiên, tất cả các phép toán số học trên con trỏ đều có liên quan đến kiểu cơ sở của nó, vì vậy khai báo kiểu dữ liệu của con trỏ một cách rõ ràng là điều rất quan trọng.

2. Con trỏ và mảng một chiều

Tên của một mảng thật ra là một con trỏ trỏ đến phần tử đầu tiên của mảng đó. Vì vậy, nếu **ary** là một mảng một chiều, thì địa chỉ của phần tử đầu tiên trong mảng có thể được biểu diễn là **&ary[0]** hoặc đơn giản chỉ là **ary**. Tương tự, địa chỉ của phần tử mảng thứ hai có thể được viết như **&ary[1]** hoặc **ary+1**,... Tổng quát, địa chỉ của phần tử mảng thứ (i + 1) có thể được biểu diễn là **&ary[i]** hay **(ary+i)**. Như vậy, địa chỉ của một phần tử mảng bất kỳ có thể được biểu diễn theo hai cách:

- Sử dụng ký hiệu & trước một phần tử mảng
- Sử dụng một biểu thức trong đó chỉ số được cộng vào tên của mảng.

Ghi nhớ rằng trong biểu thức **(ary + i), ary** tượng trưng cho một địa chỉ, trong khi **i** biểu diễn số nguyên. Hơn thế nữa, **ary** là tên của một mảng mà các phần tử có thể là cả kiểu số nguyên, ký tự, số thập phân,… (dĩ nhiên, tất cả các phần tử của mảng phải có cùng kiểu dữ liệu). Vì vậy, biểu thức ở trên không chỉ là một phép cộng; nó thật ra là xác định một địa chỉ, một số xác định của các ô nhớ . Biểu thức **(ary + i)** là một sự trình bày cho một địa chỉ chứ không phải là một biểu thức toán học.

Như đã nói ở trước, số lượng ô nhớ được kết hợp với một mảng sẽ tùy thuộc vào kiểu dữ liệu của mảng cũng như là kiến trúc của máy tính. Tuy nhiên, người lập trình chỉ có thể xác định địa chỉ của phần tử mảng đầu tiên, đó là tên của mảng (trong trường hợp này là **ary**) và số các phần tử tiếp sau phần tử đầu tiên, đó là, một giá trị chỉ số. Giá trị của **i** đôi khi được xem như là một **độ dời** khi được dùng theo cách này.

Các biểu thức **&ary[i]** và **(ary+i)** biểu diễn địa chỉ phần tử thứ **i** của **ary**, và như vậy một cách logic là cả **ary[i]** và ***(ary + i)** đều biểu diễn nội dung của địa chỉ đó, nghĩa là, giá trị của phần tử thứ **i** trong mảng **ary**. Cả hai cách có thể thay thế cho nhau và được sử dụng trong bất kỳ ứng dụng nào khi người lập trình mong muốn.

Chương trình sau đây biểu diễn mối quan hệ giữa các phần tử mảng và địa chỉ của chúng.

```
#include<stdio.h>
void main()
{
    static int ary[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int i;
    for (i = 0; i < 10; i ++)
```

```
    {
        printf("\n i = %d , ary[i] = %d , *(ary+i)= %d ", i,
      ary[i], *(ary + i));
        printf("&ary[i] = %X , ary + i = %X", &ary[i], ary + i);
        /* %X gives unsigned hexadecimal */
    }
}
```

Chương trình trên định nghĩa mảng một chiều **ary**, có 10 phần tử kiểu số nguyên, các phần tử mảng được gán giá trị tương ứng là 1, 2, ..10. Vòng lặp **for** được dùng để hiển thị giá trị và địa chỉ tương ứng của mỗi phần tử mảng. Chú ý rằng, giá trị của mỗi phần tử được xác định theo hai cách khác nhau, ary[i] và *(ary + i), nhằm minh họa sự tương đương của chúng. Tương tự, địa chỉ của mỗi phần tử mảng cũng được hiển thị theo hai cách. Kết quả của chương trình trên như sau:

```
i=0   ary[i]=1    *(ary+i)=1    &ary[i]=194   ary+i = 194
i=1   ary[i]=2    *(ary+i)=2    &ary[i]=196   ary+i = 196
i=2   ary[i]=3    *(ary+i)=3    &ary[i]=198   ary+i = 198
i=3   ary[i]=4    *(ary+i)=4    &ary[i]=19A   ary+i = 19A
i=4   ary[i]=5    *(ary+i)=5    &ary[i]=19C   ary+i = 19C
i=5   ary[i]=6    *(ary+i)=6    &ary[i]=19E   ary+i = 19E
i=6   ary[i]=7    *(ary+i)=7    &ary[i]=1A0   ary+i = 1A0
i=7   ary[i]=8    *(ary+i)=8    &ary[i]=1A2   ary+i = 1A2
i=8   ary[i]=9    *(ary+i)=9    &ary[i]=1A4   ary+i = 1A4
i=9   ary[i]=10   *(ary+i)=10   &ary[i]=1A6   ary+i = 1A6
```

Kết quả này trình bày rõ ràng sự khác nhau giữa **ary[i]** - biểu diễn giá trị của phần tử thứ **i** trong mảng, và **&ary[i]** - biểu diễn địa chỉ của nó.

Khi gán một giá trị cho một phần tử mảng như **ary[i]**, vế trái của lệnh gán có thể được viết là **ary[i]** hoặc ***(ary + i)**. Vì vậy, một giá trị có thể được gán trực tiếp đến một phần tử mảng hoặc nó có thể được gán đến vùng nhớ mà địa chỉ của nó là phần tử mảng. Đôi khi cần thiết phải gán một địa chỉ đến một định danh. Trong những trường hợp như vậy, một con trỏ phải xuất hiện trong vế trái của câu lệnh gán. Không thể gán một địa chỉ tùy ý cho một tên mảng hoặc một phần tử của mảng. Vì vậy, các biểu thức như **ary**, **(ary + i)** và **&ary[i]** không thể xuất hiện trong vế trái của một câu lệnh gán. Hơn thế nữa, địa chỉ của một mảng không thể thay đổi một cách tùy ý, vì thế các biểu thức như **ary++** là không được phép. Lý do là vì: **ary** là địa chỉ của mảng **ary.** Khi mảng được khai báo, bộ liên kết đã quyết định mảng được bắt đầu ở đâu, ví dụ, bắt đầu ở địa chỉ 1002. Một khi địa chỉ này được đưa ra, mảng sẽ ở đó. Việc cố gắng tăng địa chỉ này lên là điều vô nghĩa, giống như khi nói

```
x = 5++;
```

Bởi vì hằng không thể được tăng trị, trình biên dịch sẽ đưa ra thông báo lỗi.

Trong trường hợp mảng ary, **ary** cũng được xem như là một **hằng con trỏ**. Nhớ rằng, (ary + 1) không di chuyển mảng **ary** đến vị trí (ary + 1), nó chỉ trỏ đến vị trí đó, trong khi ary++ cố găng dời **ary** sang 1 vị trí.

Địa chỉ của một phần tử không thể được gán cho một phần tử mảng khác, mặc dù giá trị của một phần tử mảng có thể được gán cho một phần tử khác thông qua con trỏ.

```
&ary[2] =       &ary[3];      /* không cho phép*/
ary[2] = ary[3]; /* cho phép*/
```

Nhớ lại rằng trong hàm scanf(), tên các tham biến kiểu dữ liệu cơ bản phải đặt sau dấu (&), trong khi tên tham biến mảng là ngoại lệ. Điều này cũng dễ hiểu. Vì scanf() đòi hỏi địa chỉ bộ nhớ của từng biến dữ liệu trong danh sách tham số, trong khi toán tử & trả về địa chỉ bộ nhớ của biến, do đó trước tên biến phải có dấu &. Tuy nhiên dấu & không được yêu cầu đối với tên mảng, bởi vì tên mảng tự biểu diễn địa chỉ của nó.Tuy nhiên, nếu một phần tử trong mảng được đọc, dấu & cần phải sử dụng.

```
scanf("%d", *ary)    /*      đối với phần tử đầu tiên */
scanf("%d", &ary[2])         /*      đối với phần tử bất kỳ */
```

3. Con trỏ và mảng nhiều chiều

Một mảng nhiều chiều cũng có thể được biểu diễn dưới dạng con trỏ của mảng một chiều (tên của mảng) và một độ dời (chỉ số). Thực hiện được điều này là bởi vì một mảng nhiều chiều là một tập hợp của các mảng một chiều.Ví dụ, một mảng hai chiều có thể được định nghĩa như là một con trỏ đến một nhóm các mảng một chiều kế tiếp nhau. Cú pháp báo mảng hai chiều có thể viết như sau:

```
data_type (*ptr_var)[expr 2];
```

thay vì

```
data_type array[expr 1][expr 2];
```

Khái niệm này có thể được tổng quát hóa cho các mảng nhiều chiều, đó là,

```
data_type (*ptr_var)[exp 2] .... [exp N];
```

thay vì

```
data_type array[exp 1][exp 2] ... [exp N];
```

Trong các khai báo trên, data_type là kiểu dữ liệu của mảng, ptr_var là tên của biến con trỏ, array là tên mảng, và exp 1, exp 2, exp 3, ... exp N là các giá trị nguyên dương xác định số lượng tối đa các phần tử mảng được kết hợp với mỗi chỉ số.

Chú ý dấu ngoặc () bao quanh tên mảng và dấu * phía trước tên mảng trong cách khai báo theo dạng con trỏ. Cặp dấu ngoặc () là không thể thiếu, ngược lại cú pháp khai báo sẽ khai báo một mảng của các con trỏ chứ không phải một con trỏ của một nhóm các mảng.

Ví dụ, nếu **ary** là một mảng hai chiều có 10 dòng và 20 cột, nó có thể được khai báo như sau:

```
int (*ary)[20];
```

thay vì

```
int ary[10][20];
```

Trong sự khai báo thứ nhất, **ary** được định nghĩa là một con trỏ trỏ tới một nhóm các mảng một chiều liên tiếp nhau, mỗi mảng có 20 phần tử kiểu số nguyên. Vì vậy, **ary** trỏ đến phần tử đầu tiên của mảng, đó là dòng đầu tiên (dòng 0) của mảng hai chiều. Tương tự, (ary + 1) trỏ đến dòng thứ hai của mảng hai chiều, ...

Một mảng thập phân ba chiều **fl_ary** có thể được khai báo như:

```
float (*fl_ary)[20][30];
```

thay vì

```
float fl_ary[10][20][30];
```

Trong khai báo đầu, fl_ary được định nghĩa như là một nhóm các mảng thập phân hai chiều có kích thước 20 x 30 liên tiếp nhau. Vì vậy, fl_ary trỏ đến mảng 20 x 30 đầu tiên, (fl_ary + 1) trỏ đến mảng 20 x 30 thứ hai,...

Trong mảng hai chiều **ary**, phần tử tại dòng 4 và cột 9 có thể được truy xuất sử dụng câu lệnh:

```
ary[3][8];
```

hoặc

```
*(*(ary + 3) + 8);
```

Cách thứ nhất là cách thường được dùng. Trong cách thứ hai, (ary + 3) là một con trỏ trỏ đến dòng thứ 4. Vì vậy, đối tượng của con trỏ này, *(ary + 3), tham chiếu đến toàn bộ dòng. Vì dòng 3 là một mảng một chiều, *(ary + 3) là một con trỏ trỏ đến phần tử đầu tiên trong dòng 3, sau đó 8 được cộng vào con trỏ. Vì vậy, *(*(ary + 3) + 8) là một con trỏ trỏ đến phần tử 8 (phần tử thứ 9) trong dòng thứ 4. Vì vậy đối tượng của con trỏ này, *(*(ary + 3) + 8), tham chiếu đến tham chiếu đến phần tử trong cột thứ 9 của dòng thứ 4, đó là ary [3][8].

Có nhiều cách thức để định nghĩa mảng, và có nhiều cách để xử lý các phần tử mảng. Lựa chọn cách thức nào tùy thuộc vào người dùng. Tuy nhiên, trong các ứng dụng có các mảng dạng số, định nghĩa mảng theo cách thông thường sẽ dễ dàng hơn.

BÀI TẬP

1. Nhập một chuỗi kí tự từ bàn phím sau đó hiển thị từ và số lượng nguyên âm.
2. Sử dụng kiểu con trỏ làm lại tất cả các bài tập của chương 5 (mảng)

PHẦN HƯỚNG DẪN LÀM BÀI TẬP

1. Hiển thị từ và số lượng nguyên âm

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char *ptr;
cha word[10];
int I, vowcnt=0;
printf("\Enter a word:");
scanf("%s", word);
ptr = &word[0];
for(i=0; i<strlen(word); i++)
 {
if((*ptr=='a')‖ ((*ptr=='e')‖ ((*ptr=='i')‖ ((*ptr=='o')‖ ((*ptr=='u')‖
((*ptr=='A')‖ ((*ptr=='E')‖ ((*ptr=='I')‖ ((*ptr=='O')‖ ((*ptr=='U'))
vowcnt++;
ptr++;
}
printf("\n The word is: %s \n The number of vowels in the word is: %d", word,
vowcnt);
getch();
}
```

Bài tập thực hành

1. Viết chương trình nhập vào một ma trận (mảng hai chiều) các số nguyên, gồm m hàng, n cột. In ma trận đó lên màn hình. Nhập một số nguyên khác vào và xét xem có phần tử nào của ma trận trùng với số này không ? Ở vị trí nào ? Có bao nhiêu phần tử ?

2. Viết chương trình để chuyển đổi vị trí từ dòng thành cột của một ma trận (ma trận chuyển vị) vuông 4 hàng 4 cột. Sau đó viết cho ma trận tổng quát cấp m*n.

Ví dụ:

1	2	3	4
2	5	5	8
9	4	2	0
1	5	8	6

1	2	9	1
2	5	4	5
3	5	2	8
4	8	0	6

2. Viết chương trình nhập vào hai ma trận A có cấp m, k và B có cấp k, n. In hai ma trận lên màn hình. Tích hai ma trận A và B là ma trận C được tính bởi công thức:

$c_{ij}= a_{i1}*b_{1j} + a_{i2}*b_{2j} + a_{i3}*b_{3j} + ... + a_{ik}*b_{kj}$ (i=0,1,2,...m-1;j=0,1,2...n-1)
Tính ma trận tích C và in kết quả lên màn hình.

3. Xét ma trận A vuông cấp n, các phần tử a[i, i] (i= 1 ... n) được gọi là đường chéo chính của ma trận vuông A. Ma trận vuông A được gọi là ma trận tam giác nếu tất cả các phần tử dưới đường chéo chính đều bằng 0. Định thức của ma trận tam giác bằng tích các phần tử trên đường chéo chính.

Ta có thể chuyển một ma trận vuông bất kỳ về ma trận tam giác bằng thuật toán:

- Xét cột i (i =0,1...n-2)

- Trong cột i xét các phần tử a[k,i] (k=i+1...n-1)

+ Nếu a[k,i]=0 thì tăng k lên xét phần tử khác

+ Nếu a[k,i] <> 0 thì làm như sau:

Nhân toàn bộ hàng k với - a[i,i]/a[k,i]

Lấy hàng i cộng vào hàng k sau khi thực hiện phép nhân trên.

Đổi chỗ hai hàng i và k cho nhau

Nhân toàn bộ hàng k với -1 sau khi đã đổi chỗ với hàng i

Tăng k lên xét phần tử khác.

Viết chương trình tính định thức cấp n thông qua các bước nhập ma trận, in ma trận, đưa ma trận về dạng tam giác, in ma trận tam giác, in kết quả tính định thức.

4. Viết chương trình thực hiện việc trộn hai dãy có thứ tự thành một dãy có thứ tự. Yêu cầu không được trộn chung rồi mới sắp thứ tự. Khi trộn phải tận dụng được tính chất đã sắp của hai dãy con.

5. Viết chương trình nhập vào hai ma trận A và B có cấp m, n. In hai ma trận lên màn hình. Tổng hai ma trận A và B là ma trận C được tính bởi công thức:

$c_{ij}= a_{ij} +b_{ij}$ (i=0,1,2,...m-1; j=0,1,2...n-1)

Tính ma trận tổng C và in kết quả lên màn hình

BÀI TẬP NÂNG CAO

I/ Câu lệnh điều kiện If:

Bài 1: Viết chương trình nhập vào 3 số nguyên dương từ bàn phím. Xét xem 3 số đó có phải là độ dài ba cạnh của tam giác không ?
Nếu phải thì kiểm tra xem thuộc dạng tam giác gì (vuông cân, vuông, đều, cân hay tam giác thường)

Bài 2: Viết chương trình giải bất phương trình bậc nhất một ẩn ax + b >0 với a, b được nhập từ bàn phím.

Bài 3: Hằng tháng các hộ dân trong thành phố đều nhận được hóa đơn tiền điện. Tiền điện tiêu dùng của mỗi hộ gia đình được tính như sau:
100 số đầu tiên: mỗi số phải trả 1000 đồng;
Từ 101 đến 150 số mỗi số phải trả 2000 đồng;
Từ 151 đến 200 mỗi số phải trả 2500 đồng;
Từ số 200 trở lên mỗi số phải trả 3000 đồng;
Số tiền phải trả là tổng số tiền tính được cộng thêm 10% thuế VAT.
Biết rằng lượng điện tiêu thụ trong một tháng là a (KW), a được nhập từ bàn phím.
Hãy tính số tiền điện phải trả cho một tháng của một gia đình.

Bài 4: Hãy viết chương trình đổi tiền có mệnh giá n đồng ra các tiền 200000, 100000, 50000 đồng sao cho số tờ tiền là ít nhất. n được nhập vào từ bàn phím.

II/ Câu lệnh case:

Bài tập tính tiền điện.
Bài 1: Theo dương lịch, năm được biểu diễn bằng một số nguyên. Theo âm lich, năm được gọi theo can và chi. Ví dụ: năm dương lịch 2006 được gọi theo âm lịch là Bính Tuất, trong đó Bính là can và Tuất là chi.
Có tất cả 10 can: Giáp, Ất, Bính, Đinh, Mậu, Kỉ, Canh, Tân, Nhâm, Quý.
Có 12 chi: Tí, Sửu, Dần, Mão, Thìn, Tỵ, Ngọ, Mùi, Thân, Dậu, Tuất, Hợi.
Hãy lập trình:
Nhập vào từ bàn phím một năm dương lịch.
Đưa ra màn hình tên gọi năm âm lịch tương ứng (tiếng việt không dấu). Chương trình cho phép lần lượt nhập các năm và đưa ra kết quả tương ứng cho đến khi năm nhập vào là số nhỏ hơn hoặc bằng 0 thì kết thúc.
Chú ý: Bài này có thể dùng hai mảng can và chi.
Bài 2: Viết chương trình nhập từ bàn phím một số tự nhiên có 4 chữ số. Hãy in ra màn hình cách đọc tiếng việt (không dấu) của số đó.

Bài 3*: Ngày 01/01/2013 là ngày thứ ba trong tuần. Hội đồng giáo viên họp vào thứ năm đầu tiên hàng tháng trong năm trừ ra các tháng nghỉ hè là tháng 6 và 7.
Hãy in ra tất cả các ngày trong năm 2013 mà Hội đồng giáo viên của trường họp.

III/ Câu lệnh lặp For (dạng tiến và dạng lùi):

Bài 1: Ba số tự nhiên a, b và c được gọi là bộ số Py-ta-go nếu $a^2 + b^2 = c^2$. Viết

chương trình tìm tất cả các số Py-ta-go trong khoảng từ 1 đến n, với n là số nguyên dương được nhập vào từ bàn phím.

Bài 2: Một người gửi tiết kiệm tại một ngân hàng với số tiền ban đầu là a (triệu đồng) có chu kì tính lãi là c (tháng) với lãi suất là k%, biết rằng khi chưa đủ chu kì tính lãi thì không được tính lãi.
Hãy cho biết số tiền người đó rút được sau khoảng thời gian t (tháng). Cho biết phương thức tính lãi lũy kế, nghĩa là lãi suất sau mỗi tháng sẽ được cộng vào số tiền gốc.

BÀI TẬP TỔNG HỢP

1. Viết hàm in ra các ước số của 1 số tự nhiên.
2. Viết chương trình nhân 2 ma trận vuông bậc n: AxB=C với điều kiện

    - N<10 nhập từ bàn phím
    - Các phần tử của A.B cũng nhập từ bàn phím hoặc được gán ngẫu nhiên (xem câu 2)
    - Hiển thị ma trận A.B.C lên màn hình thẳng hàng theo dòng và cột

3. Viết hàm biến đổi một xâu ký tự thành số tự nhiên tương ứng. Ví dụ: "9214" thành 9214. Những giá trị không đúng như "-12", "9abs12", "3.14" … thành -1.
4. Viết hàm tính m^n với m,n là số tự nhiên.
5. Viết hàm tính n!!

    N!!=1.3.5…n nếu n lẻ

    N!!=2.4.6…n nếu n chẵn

6. Viết hàm tìm Max của 3 số a,b,c. Gợi ý: a,b,c là 3 tham số đầu vào, max(a,b,c) là giá trị trả lại của hàm. Viết chương trình chính nhập a,b,c từ bàn phím sau đó gọi hàm.
7. Nâng cấp các hàm trên từ 3 số lên thành dãy n số. Gợi ý: dãy số và số phần tử của nó là các đầu vào của hàm. Viết chương trình chính nhập dãy số từ bàn phím, sau đó gọi hàm. Sau đó sửa lại hàm chính: dãy số không phải nhập bằng tay từ bàn phím mà được gán những giá trị ngẫu nhiên sinh ra bởi hàm random() trong stdlib.h biết rằng hàm random(n) trả lại một số nguyên ngẫu nhiên trong khoảng (0,n-1).
8. Viết chương trình giải phương trình bậc 2.
9. Tìm các số tự nhiên bé hơn 10000 bằng tổng lập phương các chữ số của nó.
10. Tìm các số tự nhiên bé hơn 10000 bằng tổng các ước số của nó.
11. Viết hàm in ra các ước số của 1 số tự nhiên.
12. Viết chương trình nhân 2 ma trận vuông bậc n: AxB=C với điều kiện

    - N<10 nhập từ bàn phím
    - Các phần tử của A.B cũng nhập từ bàn phím hoặc được gán ngẫu nhiên (xem câu 2)
    - Hiển thị ma trận A.B.C lên màn hình thẳng hàng theo dòng và cột

13. Viết chương trình nhập vào n<100 số thực (nhập từ bàn phím hoặc gán ngẫu nhiên). Xếp lại theo thứ tự giảm dần và hiển thị kết quả.
14. Viết lại chương trình trên thành hàm sắp xếp dãy số. Gợi ý: đầu vào là con trỏ đến phần tử đầu dãy. Hàm không cần phải hiển thị kết quả.
15. Viết hàm biến đổi một xâu ký tự thành số tự nhiên tương ứng. Ví dụ: "9214" thành 9214. Những giá trị không đúng như "-12", "9abs12", "3.14" … thành -1.
16. Viết hàm tính m^n với m,n là số tự nhiên.
17. Viết hàm tính n!!

N!!=1.3.5…n nếu n lẻ
N!!=2.4.6…n nếu n chẵn

18. A,B là các ma trận vuông bậc n (n<10). Viết chương trình:

- N được nhập từ bàn phím, các phần tử của A,B nhận những giá trị ngẫu nhiên hoặc cũng được nhập từ bàn phím.
- Hiển thị các phần tử của A,B lên màn hình thẳng hàng theo dòng và cột.
- In ra các phần tử nằm trên đương chéo chính, đường chéo phụ của A
- In ra phần tử lớn nhất của dòng k (k<n)
- In ra phần tử bé nhất của cột p (p<n)
- Hoán vị dòng p và dòng k của ma trận B. Hiển thị kết quả sau khi hoán vị.
- Hiển thị ma trân tổng C=A+B
- Hiển thị ma trận tam giác trên của A
- Tính tổng các phần tử dương của ma trận.

19. Viết hàm đếm số lần xuất hiện của một số nguyên k trong một dãy số. Viết chương trình chính nhập giá trị cho dãy số (tử bàn phím hoặc nhận giá trị ngẫu nhiên) và gọi hàm đó.
20. Viết hàm cho biết vị trí xuất hiện đầu tiên của một số nguyên k trong môt dãy số. Viết chương trình chính nhập giá trị cho dãy số (tử bàn phím hoặc nhận giá trị ngẫu nhiên) và gọi hàm đó.
21. Viết hàm trả lại số phần tử khác nhau của một dãy số.
22. Viết hàm trả lại số phần tử khác nhau của một ma trận.
23. Một xâu ký tự gọi là Palindrome nếu nó không thay đổi khi ta đảo ngược thứ tự các ký tự của nó, chẳng hạn như “ABBA” hay “XyZyX”. Viết hàm kiểm tra một xâu có phải là Palindrome hay không (trả lại 1 nếu đúng). Viết chương trình chính nhập xâu ký tự và gọi hàm đó để kiểm tra.
24. Viết hàm trả lại số ký tự khác nhau của một xâu ký tự.
25. Viết hàm kiểm tra một số có phải là số nguyên tố hay không (trả lại 1 nếu đúng). Viết chương trình nhập số và gọi hàm đó để kiểm tra.
26. Viết chương trình phân tích một số thành các thừa số nguyên tố.

MỘT SỐ ĐỀ THAM KHẢO

Đề 1:

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

Tính chu vi và diện tích của tam giác theo công thức:

Chu vi CV = a+b+c

Diện tích S = sqrt(p*(p-a)*(p-b)*(p-c)) Trong đó: p=CV/2

In các kết quả lên màn hình

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

Đề 2:

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

2. Viết chương trình nhập vào tọa độ của hai điểm (x1, y1) và (x2, y2)

a) Tính hệ số góc của đường thẳng đi qua hai điểm đó theo công thức:

Hệ số góc = (y2 - y1) /(x2 - x1)

b) Tính khoảng cách giữa hai điểm

Đề 3:

1. Viết chương trình nhập vào một ký tự:

a) In ra mã Ascii của ký tự đó.

b) In ra ký tự kế tiếp của nó.

2. Viết chương trình nhập từ bàn phím 2 số a, b và một ký tự ch. (Switch)

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

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

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

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

Đề 4:

Có hai phương thức gửi tiền tiết kiệm: gửi không kỳ hạn lãi suất 2.4%/tháng, mỗi tháng tính lãi một lần, gửi có kỳ hạn 3 tháng lãi suất 4%/tháng, 3 tháng tính lãi một lần.

Đề 5:

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

2. Có hai phương thức gửi tiền tiết kiệm: gửi không kỳ hạn lãi suất 2.4%/tháng, mỗi tháng tính lãi một lần, gửi có kỳ hạn 3 tháng lãi suất 4%/tháng, 3 tháng tính lãi một lần.

Đề 6:

1. Viết chương trình nhập vào ngày, tháng, năm. In ra ngày tháng năm theo dạng dd/mm/yy. (dd: ngày, mm: tháng, yy: năm. Ví dụ: 20/11/99)

2. Trò chơi "Oẳn tù tì": trò chơi có 2 người chơi mỗi người sẽ dùng tay để biểu thị một trong 3 công cụ sau: Kéo, Bao và Búa.

Nguyên tắc: Kéo thắng bao.
Bao thắng búa.
Búa thắng kéo.

Viết chương trình mô phỏng trò chơi này cho hai người chơi. (switch)

Đề 7:

1. Viết chương trình đảo ngược một số nguyên dương có đúng 3 chữ số.

2. Viết chương trình tính tiền điện gồm các khoản sau: Tiền thuê bao điện kế: 1000 đồng / tháng.

Định mức sử dụng điện cho mỗi hộ là 50 Kw

Phần định mức tính giá 450 đồng /Kwh

Nếu phần vượt định mức <= 50 Kw tính giá phạt cho phần này là 700 đồng/Kwh .

Nếu phần vượt định mức lớn 50 Kw và nhỏ hơn 100Kw tính giá phạt cho phần này là 910 đồng/Kwh

Nếu phần vượt định mức lớn hơn hay bằng 100 Kw tính giá phạt cho phần này là 1200 đồng/Kwh .

Với: chỉ số điện kế cũ và chỉ số điện kế mới nhập vào từ bàn phím. In ra màn hình số tiền trả trong định mức, vượt định mức và tổng của chúng. (if)

Đề 8:

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

2. Viết chương trình nhập vào tọa độ của hai điểm (x1, y1) và (x2, y2)

a) Tính hệ số góc của đường thẳng đi qua hai điểm đó theo công thức:

Hệ số góc = (y2 - y1) /(x2 - x1)

b) Tính khoảng cách giữa hai điểm

Đề 9:

1. Viết chương trình nhập vào một ký tự:

a) In ra mã Ascii của ký tự đó.

b) In ra ký tự kế tiếp của nó.

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

Đề 10:

1. . Viết chương trình nhập từ bàn phím 2 số a, b và một ký tự ch. (if)

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

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

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

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

Đề 11:

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

2. Viết chương trình nhập vào 2 số là tháng và năm của một năm. Xét xem tháng đó có bao nhiêu ngày? Biết rằng:

Nếu tháng là 4, 6, 9, 11 thì số ngày là 30.

Nếu tháng là 1, 3, 5, 7, 8, 10, 12 thì số ngày là 31.

Nếu tháng là 2 và năm nhuận thì số ngày 29, ngược lại thì số ngày là 28.

Đề 12:

1. Viết chương trình nhập vào ngày, tháng, năm. In ra ngày tháng năm theo dạng dd/mm/yy. (dd: ngày, mm: tháng, yy : năm. Ví dụ: 20/11/99)

2. Có hai phương thức gửi tiền tiết kiệm: gửi không kỳ hạn lãi suất 2.4%/tháng, mỗi tháng tính lãi một lần, gửi có kỳ hạn 3 tháng lãi suất 4%/tháng, 3 tháng tính lãi một lần.

Đề 13:

1. Viết chương trình đảo ngược một số nguyên dương có đúng 3 chữ số.

2. Trò chơi "Oẳn tù tì": trò chơi có 2 người chơi mỗi người sẽ dùng tay để biểu thị một trong 3 công cụ sau: Kéo, Bao và Búa.

Nguyên tắc: Kéo thắng bao.
Bao thắng búa.
Búa thắng kéo.

Viết chương trình mô phỏng trò chơi này cho người chơi với máy. (switch)

Đề 14:

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

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

Tính chu vi và diện tích của tam giác theo công thức:

Chu vi CV = a+b+c

Diện tích S = sqrt(p*(p-a)*(p-b)*(p-c)) Trong đó: p=CV/2

In các kết quả lên màn hình

Đề 15:

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

2. Viết chương trình nhận vào giờ, phút, giây dạng (hh:mm:ss), từ bàn phím. Cộng thêm một số giây vào và in ra kết quả dưới dạng (hh:mm:ss).

Đề 16:

1. Viết chương trình nhập vào ngày tháng năm của một ngày, kiểm tra nó có hợp lệ không.

2. Viết chương trình nhập vào một ký tự:

a) In ra mã Ascii của ký tự đó. b) In ra ký tự kế tiếp của nó.

Đề 17:

1. Viết chương trình nhập vào ngày, tháng, năm. In ra ngày tháng năm theo dạng dd/mm/yy. (dd: ngày, mm: tháng, yy : năm. Ví dụ: 20/11/99)

2. Kiểm tra một ký tự nhập vào thuộc tập hợp nào trong các tập ký tự sau:

Các ký tự chữ hoa: 'A' ...'Z'

Các ký tự chữ thường: 'a' ... 'z'

Các ký tự chữ số : '0' ... '9' Các ký tự khác.

Đề 18:

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

Đề 19:

1. Viết chương trình nhập vào một ký tự:

a) In ra mã Ascii của ký tự đó.

b) In ra ký tự kế tiếp của nó.

2. Viết chương trình nhập vào ngày tháng năm của ngày hôm nay, in ra ngày tháng năm của ngày mai.

Đề 20:

1. Viết chương trình nhập vào ngày, tháng, năm. In ra ngày tháng năm theo dạng dd/mm/yy. (dd: ngày, mm: tháng, yy : năm. Ví dụ: 20/11/99)

2. Viết chương trình tính các tổng sau:

a) S=1 + 2 + ... + n

b) S=1/2 + 2/3 + ... +n/(n+1)

c) S= - 1 +2 - 3 +4 - ... + $(-1)^n$n

Đề 21:

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

Đề 22:

1. Viết chương trình nhập vào một ký tự:

a) In ra mã Ascii của ký tự đó.

b) In ra ký tự kế tiếp của nó.

2. Tìm số nguyên dương k nhỏ nhất sao cho 2^k > n với n là một số nguyên dương nhập từ bàn phím.

Đề 23:

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

2. Viết chương trình mô phỏng phép chia nguyên DIV 2 số nguyên a và b như sau: để chia nguyên a và b ta tính trị a-b, sau đó lấy hiệu tìm được lại trừ cho b... tiếp tục cho đến khi hiệu của nó nhỏ hơn b. Số lần thực hiện được các phép trừ ở trên sẽ bằng trị của phép chia nguyên.

Đề 24:

1. Viết chương trình nhập vào ngày, tháng, năm. In ra ngày tháng năm theo dạng dd/mm/yy. (dd: ngày, mm: tháng, yy : năm. Ví dụ: 20/11/99)

2. Tìm số nguyên dương N nhỏ nhất sao cho

1+1/2+ ...+1/N > S, với S nhập từ bàn phím.

Đề 25:

1. Viết chương trình nhập vào ngày, tháng, năm. In ra ngày tháng năm theo dạng dd/mm/yy. (dd: ngày, mm: tháng, yy : năm. Ví dụ: 20/11/99)

2. Viết chương trình tính P=2*4*6*...*(2n), n nhập từ bàn phím.

Đề 26:

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

2. Viết chương trình tìm UCLN và BCNN của hai số a và b theo thuật toán sau (Ký hiệu UCLN của a, b là (a,b) còn BCNN là [a,b])

- Nếu a chia hết cho b thì (a,b) = b
- Nếu a = b*q + r thì (a,b) = (b,r)
- [a,b] = a*b/(b,r)

Đề 27:

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

2. Viết chương trình nhập vào một dãy n số, tìm số lớn nhất của dãy và xác định vị trí của số lớn nhất trong dãy.

Đề 28:

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

2. Viết chương trình đếm số chữ số của một số nguyên n.

Đề 29:

1. Viết chương trình nhập vào ngày, tháng, năm. In ra ngày tháng năm theo dạng dd/mm/yy. (dd: ngày, mm: tháng, yy : năm. Ví dụ: 20/11/99)

2. Tìm số nguyên dương k nhỏ nhất sao cho 2^k > n với n là một số nguyên dương nhập từ bàn phím.

Đề 30:

1. Viết chương trình nhập vào ngày, tháng, năm. In ra ngày tháng năm theo dạng dd/mm/yy. (dd: ngày, mm: tháng, yy : năm. Ví dụ: 20/11/99)

2. Viết chương trình in ra số đảo ngược của một số nguyên n, với n nhập từ bàn phím.

BÀI TẬP MẪU

Bài 1

```
#include <stdio.h>
#include <conio.h>
int max(int a,int b,int c);
void main()
{
 int x,y,z;
 clrscr();
 printf("x,y,z="); scanf("%d%d%d",&x,&y,&z);
 printf("Max=%d",max(x,y,z));
 getch();
}
int max(int a,int b,int c)
{
 if (a>b)
     return a>c?a:c;
 else return b>c?b:c;
}
```

Bài 2

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define MAX 100
int maxday(int *p,int len);
void main()
{
int a[MAX],i;
clrscr();
for (i=0;i<10;i++)
     {
     a[i]=random(100); printf("%4d",a[i]);
     }
printf("\nMax day A: %d",maxday(a,10));

getch();
}
int maxday(int *p,int len)
{
int i,gt;
gt=*p;
for (i=0;i<len;i++)
   if (*(p+i)>gt) gt=*(p+i);
return gt;
}
```

Bài 3

```
#include <stdio.h>

void main()
{
  int i,j,s,n=5000;
  for (i=2; i<=n; i++)
  {
     s=1;
    for (j=2; j<=i/2; j++)
      if (i%j == 0)
                s+= j;
       if (s== i)
        printf("\n%d", i);
  }
  getch();
}
```

Bài 4

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <iomanip.h>

const int SIZE = 10+1;

//--------------------------------

void nhap(long n, long a[SIZE][SIZE]) {
    for (long i = 0; i < n; i ++)
        for (long j = 0; j < n; j ++) a[i][j] = random(10);
}

void viet(long n, long a[SIZE][SIZE]) {
    for (long i = 0; i < n; i ++) {
        for (long j = 0; j < n; j ++) cout << setw(4) << a[i][j] << " ";
        cout << endl;
    }
    cout << endl;
}

void nhan(long n, long a[SIZE][SIZE], long b[SIZE][SIZE], long c[SIZE][SIZE])
{
    for (long i = 0; i < n; i ++)
        for (long j = 0; j < n; j ++) {
            c[i][j] = 0;
            for (long k = 0; k < n; k ++) c[i][j] += a[i][k] * b[k][j];
        }
}

//--------------------------------
long main() {
    clrscr();
    long a[SIZE][SIZE], b[SIZE][SIZE], c[SIZE][SIZE], d[SIZE][SIZE];
    long n = 5;
    nhap(n, a);  viet(n, a);
    nhap(n, b);  viet(n, b);
    nhan(n, a, b, c);
    viet(n, c);
    nhan(n, c, c, d);
    viet(n, d);
    getch();
    return 0;
}
```

Bài 5

```
#include <string.h>
#include <stdio.h>
#include <conio.h>
int str2num(char *p);
void main()
{
    clrscr();
    char a[100];
    printf("Enter a string:");
    gets(a);
    printf("%d",str2num(a));
    getch();

}
int str2num(char *p)
{
    int i,x = 0;

    for (i = 0; i < strlen(p); i ++)
            if (*(p+i) >= '0' && *(p+i) <= '9')
                    x = x * 10 + (*(p+i) - '0');
            else return -1;
    return x;
}
```

TÀI LIỆU THAM KHẢO

[1] **Tiến Sĩ Lê Mạnh Thạnh**. *Giáo trình môn lập trình C*. Nhà xuất bản giáo dục Năm 2000.
[2] **Nguyễn Linh Giang, Nguyễn Xuân Thực, Lê Văn Thái**. *Giáo trình kỹ thuật lập trình C*. Nhà xuất bản giáo dục – Năm 2005.
[3] **Nguyễn Đình Tê, Hoàng Đức Hải**. *Giáo trình lý thuyết và bài tập ngôn ngữ C tập I.* Nhà xuất bản Phương Đông.
[4] http://www.haiphongit.com
[5] http://www.toti\el
[6] http://ebook. vinagrid.com
[7] http://www.dvpub.com.vn
[8] http://bachkhoatoanthu.org
[9] http://www.echip.com.vn/echiproot/
[10] http://bkitclub.net/forum/showthread.php?t=5730