THẠC BÌNH CƯỜNG - LÊ QUỐC TRUNG

GIÁO TRÌNH

LẬP TRÌNH PASCAL

NHÀ XUẤT BẢN GIÁO DỤC

THẠC BÌNH CƯỜNG – LÊ QUỐC TRUNG

Giáo trình
LẬP TRÌNH PASCAL

(Dùng cho các trường đào tạo hệ Cao đẳng và Trung cấp chuyên nghiệp)

NHÀ XUẤT BẢN GIÁO DỤC

17 – 2007/CXB/59 – 2217/GD

Mã số : 6E008M7 - DAI

Lời nói đầu

Cuốn "Giáo trình Lập trình Pascal" trình bày các vấn đề cơ bản về tin học và ngôn ngữ lập trình Pascal. Các nội dung được biên soạn kỹ nhằm đáp ứng nhu cầu của các bạn sinh viên chuyên ngành Điện, Điện tử, Tự động và Công nghệ thông tin các trường Cao đẳng, Trung cấp chuyên nghiệp trên cả nước. Toàn bộ nội dung cuốn "Giáo trình Lập trình Pascal" được chia làm 8 chương:

Mở đầu : Giới thiệu Turbo Pascal

Chương 1 : Các thành phần cơ bản

Chương 2 : Các kiểu dữ liệu chuẩn

Chương 3 : Biểu thức

Chương 4 : Cấu trúc của một chương trình Pascal

Chương 5 : Các lệnh của Turbo Pascal

Chương 6 : Các kiểu dữ liệu mở rộng

Chương 7 : Chương trình con

Chương 8 : Đồ hoạ

Cuốn sách giới thiệu về các thành phần cơ bản trong ngôn ngữ lập trình Pascal, các khái niệm cơ bản và các chương trình ví dụ ứng dụng linh hoạt đã được biên soạn và thử nghiệm cùng nhiều bài tập bổ sung đã được biên soạn nhằm giúp các bạn học viên có thể thực tập.

Hy vọng cuốn "Giáo trình Lập trình Pasclal" sẽ mang lại nhiều kiến thức bổ ích cho bạn đọc trong quá trình nghiên cứu và học tập. Chúc các bạn thành công ! Rất mong nhận được những ý kiến đóng góp, phê bình cho những khiếm khuyết còn tồn tại để cuốn sách sẽ hoàn thiện hơn trong các lần xuất bản sau.

Những ý kiến góp ý xin gửi về địa chỉ : *Công ty cổ phần sách Đại học - Dạy nghề, 25 Hàn Thuyên, Hà Nội.*

CÁC TÁC GIẢ

Mở đầu

GIỚI THIỆU TURBO PASCAL

Pascal là một ngôn ngữ lập trình cấp cao do Niklaus Wirth, giáo sư Điện toán Trường Đại học Kỹ thuật Zurich (Thụy sĩ), đề xuất vào năm 1970 với tên Pascal để kỷ niệm nhà toán học và triết học nổi tiếng Blaise Pascal (người Pháp).

Lúc đầu mục đích của Wirth thiết kế Pascal là để giảng dạy lập trình. Pascal có các đặc điểm : ngữ pháp, ngữ nghĩa đơn giản và có tính logic ; cấu trúc chương trình rõ ràng, dễ hiểu (thể hiện tư duy lập trình cấu trúc); dễ sửa chữa, cải tiến.

Trong quá trình phát triển, Pascal đã phát huy được ưu điểm của mình và tỏ ra hơn hẳn nhiều ngôn ngữ cao cấp khác, Pascal đã trở thành một ngôn ngữ mạnh được ứng dụng trong rất nhiều lĩnh vực khác nhau. Các tổ chức và công ty chuyên về máy tính dựa trên Pascal chuẩn đã phát triển thêm và tạo ra các chương trình dịch ngôn ngữ Pascal với nhiều phần thêm bớt khác nhau. Ví dụ: TURBO PASCAL của hãng Borland (Mỹ), QUICK PASCAL của hãng Microsoft, UCSD PASCAL (University of California at San Diego), ANSI PASCAL (American National Standard Institut), ...

So với nhiều sản phẩm Pascal của nhiều tổ chức và công ty khác nhau xuất bản, Turbo Pascal tỏ ra có nhiều ưu điểm nhất và hiện nay đã trở thành một ngôn ngữ lập trình cao cấp phổ biến nhất trên thế giới, được sử dụng trong lĩnh vực giảng dạy và lập trình chuyên nghiệp. Chỉ trong vòng mấy năm, Turbo Pascal được cải tiến qua nhiều phiên bản: 1.0, 2.0, 3.0, 4.0, 5.0, 5.5 (1989), 6.0 (1990), 7.0 (1992).

Chương 1

CÁC THÀNH PHẦN CƠ BẢN

1.1. CÁC KÝ HIỆU CƠ BẢN

Cũng giống như mọi ngôn ngữ khác, TURBO PASCAL có bộ chữ cái riêng và chia thành 2 nhóm như sau:

– Các ký tự chữ và số:

+ 26 chữ cái a, b, c, ..., z.

+ 10 chữ số thập phân 0, 1, 2, ..., 9.

– Các ký tự đặc biệt:

+ Ký tự trống ' '.

+ Dấu bằng '='.

+ Dấu cộng và dấu trừ '+', '-'.

+ Dấu nhân và dấu chia '*', '/'.

+ Dấu lớn hơn và dấu nhỏ hơn '>', '<'.

+ Dấu lớn hơn hay bằng, dấu nhỏ hơn hay bằng '>=', '<='.

+ Dấu đô la '$'.

...

Đây là tập hợp các ký tự được dùng để viết các chương trình, không được dùng các ký hiệu khác ngoài các ký hiệu nói trên (ví dụ : α, β, Ψ, Σ, ω, φ...).

Các ký tự trên được nhóm lại theo các kiểu khác nhau để tạo thành một số từ nhất định gọi là *từ khoá*. Đến lượt chúng, các từ khoá lại liên kết với nhau theo những quy tắc cần phải tôn trọng (gọi là *cú pháp*) để tạo thành các câu lệnh mà chúng ta sẽ dần dần làm quen trong quá trình giới thiệu ngôn ngữ này. Một chương trình bao gồm nhiều câu lệnh diễn đạt một thuật toán nào đó.

1.2. CÁC TỪ KHOÁ

Các từ khoá là một bộ phận không thể tách rời của TURBO PASCAL, được định nghĩa trước với ý nghĩa xác định và *không thể* định nghĩa lại. Các từ khoá dùng để khai báo các kiểu dữ liệu (ví dụ: interger, real...), viết các toán tử (ví dụ: mod, div...), diễn đạt các câu lệnh (ví dụ if, then...), các từ khoá phải được dùng đúng cú pháp, không được dùng vào việc khác, người lập trình không được phép đặt tên mới trùng với các từ khoá.

Sau đây là một số từ khoá:

array	do	begin
end	case	of
for	to	downto
const	type	var
div	set	else
and	or	if
repeat	while	record

1.3. TÊN (định danh)

Tên theo định nghĩa là một dãy ký tự dùng để chỉ tên biến, tên hằng, tên kiểu dữ liệu, tên chương trình con... Tên bắt đầu bằng chữ cái, sau đó có thể là chữ cái hoặc chữ số, dấu gạch nối. Tên không được chứa các ký tự đặc biệt như dấu trống ' ', dấu ':', dấu '?', dấu '!'....

Sau đây là một số tên hợp lệ: K40, BachKhoa, CNTT... còn các tên sau đây là không hợp lệ: 12ANH (vì bắt đầu bằng số), kien truc (vì có chứa dấu trống).

Độ dài cực đại của tên là 63 ký tự, nếu ta đặt một tên dài hơn 63 ký tự thì chỉ có 63 ký tự đầu tiên có nghĩa.

TURBO PASCAL *không phân biệt* chữ hoa với chữ thường. Như vậy, các tên ABC, abc, Abc là giống nhau. Khi lập trình người ta thường đặt tên sao cho nó phản ánh được nội dung của đối tượng. Việc đặt tên theo kiểu diễn nghĩa như vậy rất có ích khi bảo trì chương trình.

Các ví dụ đúng về tên :	PT_BAC_2 ; Delta;	a_1_b
Các ví dụ sai về tên :	PT BAC2	(có ký tự trống)
	3ABC	(ký tự đầu tiên là chữ số)
	g(x)	(sử dụng dấu ())
	Label	(trùng với từ khoá)
	x-1	(dùng dấu trừ)

1.4. CÁC TÊN CHUẨN

TURBO PASCAL xác định một số tên chuẩn cho các kiểu hằng, biến và thủ tục. Các tên chuẩn này có thể định nghĩa lại nhưng điều này dễ gây ra sự nhầm lẫn, vì vậy tốt nhất là không nên sử dụng tên chuẩn để đặt tên cho các đối tượng riêng của mình.

Sau đây là một số tên chuẩn:

sin	cos	tan
exp	false	boolean
char	odd	copy
...		

Chú ý: Sự khác nhau giữa tên chuẩn và từ khoá là người lập trình có thể định nghĩa lại tên chuẩn và dùng vào việc khác nếu muốn, còn từ khoá phải dùng đúng với quy định của Turbo Pascal. Một ví dụ đơn giản là ta có thể định nghĩa một hàm có tên là Sin(x) để tính căn bậc ba của x. Song ta không thể dùng từ khoá array để làm bất cứ việc gì khác ngoài công dụng đã được quy định sẵn của nó là dùng để khai báo các mảng.

1.5. CÁC DÒNG CHƯƠNG TRÌNH

Độ dài tối đa của một dòng chương trình là 127 ký tự.

BÀI TẬP

1.1. Hãy đặt 10 tên hợp lệ trong TURBO PASCAL.

1.2.Các tên sau đây, tên nào hợp lệ trong môi trường TURBO PASCAL :

a) tinh dientich
b) Tinh_dien_tich
c) bcd
f) 12a4H
g) K56
h) k46
i)f1.txt
k)122thnl

Chương 2

CÁC KIỂU DỮ LIỆU CHUẨN

Một kiểu dữ liệu là một tập hợp các giá trị mà một biến thuộc kiểu đó có thể nhận. Mỗi biến trong chương trình đều phải kết hợp với một và chỉ một kiểu dữ liệu. Một kiểu dữ liệu trong TURBO PASCAL có thể phức tạp nhưng trong mọi tình huống nó đều được cấu thành từ các kiểu dữ liệu chuẩn như interger, real, char, boolean. Ta lần lượt khảo sát các kiểu này.

2.1. KIỂU SỐ NGUYÊN (integer)

Trong TURBO PASCAL, mọi biến kiểu integer đều có thể nhận được các giá trị nguyên nằm trong đoạn [-32768, 32767]. Mỗi giá trị integer chiếm 2 byte bộ nhớ.

Tuy vậy, việc thực hiện các phép toán với các kiểu dữ liệu integer dẫn đến kết quả vượt quá phạm vi nói trên sẽ không được thông báo, vì vậy chúng ta phải đặc biệt lưu ý tới vấn đề này. Ví dụ, biểu thức 1000*100/50 sẽ không cho kết quả 2000 vì tích 1000*100 đã cho kết quả vượt quá 32767.

Vì kiểu integer chỉ biểu diễn được các số nguyên khá bé nên từ TURBO PASCAL 4.0 trở đi người ta định nghĩa thêm các kiểu nguyên khác:

Tên kiểu	Số byte chiếm trong bộ nhớ	Miền giá trị
Byte	1 byte	[0,255]
ShortInt	1 byte	[-128,127]
Word	2 bytes	[0,65535]
Integer	2 bytes	[-32768, 32767]
LongInt	4 bytes	[-2147483648,2147483647]

2.2. KIỂU SỐ THỰC (real)

Một biến kiểu real có thể lấy các giá trị hàm nằm trong khoảng $[2.9 \times 10^{-39}, 1.7 \times 10^{38}]$. Một giá trị kiểu real chiếm 6 byte bộ nhớ.

Khi thực hiện các phép toán với các giá trị real, nếu kết quả vượt ra ngoài khoảng trên, chương trình sẽ dừng lại, máy tính báo lỗi tràn ô nhớ. Còn nếu giá trị tuyệt đối của kết quả quá bé, thì sẽ được xem bằng 0.

Các giá trị thực được biểu diễn theo hai cách: dạng dấu phẩy tĩnh và dạng dấu phẩy động.

– Cách viết số thực theo dạng dấu phẩy tĩnh: viết dạng thập phân bình thường. Ví dụ: 2.12, +4, -25.345768.

– Cách viết số thực theo dạng dấu phẩy động: số được tách thành hai phần là định trị và bậc. Phần định trị là một số nguyên hay số thực viết dưới dạng dấu phẩy tĩnh. Phần bậc là một số nguyên. Hai phần cách nhau bởi chữ E hay e.

Ví dụ :

123.456E-4 : Biểu diễn số 0.0123456

0.12E+5 : Biểu diễn số 12000.0

-52.4e2 : Biểu diễn số -5240.0

2E8 : Biểu diễn số 200000000.0

2.3. KIỂU LOGIC (boolean)

Một giá trị kiểu boolean chỉ có thể là một trong hai giá trị TRUE hoặc FALSE, các giá trị này được xếp thứ tự như sau: FALSE < TRUE. Một giá trị kiểu boolean chiếm 1 byte bộ nhớ.

Các phép toán gồm có: AND, OR, XOR, NOT, tác dụng thể hiện qua bảng sau :

P	Q	NOTP	P AND Q	P OR Q	P XOR Q
TRUE	TRUE	FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	FALSE	TRUE	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE

Trong đó P và Q là hai giá trị kiểu Boolean.

2.4. KIỂU KÝ TỰ (char)

Kiểu ký tự bao gồm 256 ký tự trong bảng mã ASCII. Một ký tự được viết trong hai dấu nháy đơn, ví dụ: 'T', 'a'. Mỗi giá trị kiểu char chiếm 1 byte bộ nhớ.

Một ký tự được biểu diễn trong bộ nhớ bởi giá trị của nó trong bộ mã ASCII. Ví dụ ký tự 'A' có mã ASCII là 65, sẽ được biểu diễn trong bộ nhớ bằng một byte có giá trị là 65.

Các ký tự có thể so sánh với nhau. Sự so sánh được tiến hành theo giá trị của chúng trong bảng mã ASCII.

Ví dụ: 'A' < 'B' vì 65 (là mã ASCII của ký tự 'A') < 66 (là mã ASCII của ký tự 'B').

Trong TURBO PASCAL có hai thủ tục:

– Ord(c): cho ta giá trị tương ứng của ký tự c trong bộ mã.

Ví dụ: ord('A') = 65.

– Chr(i): thủ tục này cho ta ký tự tương ứng với vị trí i trong bộ mã, chẳng hạn chr(65) = 'A'.

2.5. KIỂU XÂU KÝ TỰ (String)

Một giá trị kiểu String là một dãy ký tự bất kỳ đặt trong hai dấu nháy đơn. Số ký tự của dãy không quá 255. Xâu không có ký tự nào cả gọi là xâu rỗng.

Ví dụ : 'Ho va Ten': xâu gồm 9 ký tự

'' : xâu rỗng

BÀI TẬP

2.1. Viết các số sau ra dạng thông thường:

3.5E+2 -4.4E-15

2.2. Các giá trị sau đây thuộc kiểu chuẩn nào?

a) 1/4 b) 2002 c) sqrt(25.0) d) 5>8

e) 6/3 g) sqrt(36) h) 100,000 i) sqrt (34)

2.3. Khi cần tính các số có giá trị từ 0 đến 255 ta nên dùng dữ liệu kiểu gì?

Chương 3

BIỂU THỨC

Biểu thức bao gồm các toán hạng (biến, hằng, lời gọi hàm) kết hợp với nhau bằng các toán tử.

Các toán tử trong TURBO PASCAL được chia thành 5 lớp có thứ tự ưu tiên như sau:

3.1. TOÁN TỬ ĐẢO DẤU

Đây chính là phép trừ không có số bị trừ. Toán tử này cho phép đổi dấu của toán hạng thuộc kiểu integer hay real đặt ngay sau nó.

3.2. TOÁN TỬ NOT

Toán tử này phủ định giá trị của một toán hạng (biểu thức) boolean sau nó.

3.3. CÁC TOÁN TỬ THUỘC LỚP NHÂN

Toán tử	Phép toán	Kiểu toán hạng	Kiểu kết quả
*	nhân	real	real
*	nhân	integer	integer
*	nhân	real, integer	real
/	chia	real	real
/	chia	real, integer	real
/	chia	integer	real
Div	chia nguyên	integer	integer
Mod	lấy phần dư	integer	integer
And	và logic	boolean	boolean

3.4. CÁC TOÁN TỬ THUỘC LỚP CỘNG

Toán tử	Phép toán	Kiểu toán hạng	Kiểu kết quả
+ (-)	cộng (trừ)	real	real
+ (-)	cộng (trừ)	integer	integer
+ (-)	cộng (trừ)	real, integer	real
Or	hoặc logic	boolean	boolean

3.5. CÁC TOÁN TỬ QUAN HỆ

Các toán tử này tác dụng lên tất cả các kiểu dữ liệu chuẩn. Một điều nên nhớ là *chỉ có thể so sánh các toán hạng có cùng kiểu*. Có một ngoại lệ là các toán hạng kiểu integer và real cũng có thể so sánh với nhau bằng các toán tử quan hệ. Kết quả của phép so sánh *bao giờ cũng có kiểu boolean*. Các toán tử quan hệ bao gồm: =, >, <, >=, <=, <>.

3.6. CÁC THỦ TỤC

PASCAL thiết kế sẵn một số thủ tục (ta sẽ dần làm quen với chúng ở những phần sau). Sau đây là một số thủ tục hay dùng:

Toán học	TURBO PASCAL
x^2	sqr(x)
$\sqrt{2}$	sqrt(x)
cosx	cos(x)
sinx	sin(x)
lnx	ln(x)
...	

BÀI TẬP

3.1. Viết các khai báo cho yêu cầu sau:

a) x, y là các biến thực

b) biến k sẽ nhận các giá trị trong khoảng [0, 255]

c) z, t là các biến nguyên trong khoảng [-32768, + 32767].

d) k, j, i là các biến có thể nhận các giá trị nguyên trong khoảng [0,65535]

e) ch là biến có thể nhận các ký tự.

3.2. Chuyển các biểu thức viết trong PASCAL sau đây thành biểu thức toán học thông thường

a) sqr (b) – 4*a*c

b) (a+b)/2/a

c) (a*a +b*b)/2* x

d) (x+y)/ (2*a)

3.3. Cho biết kết quả và kiểu dữ liệu của các biểu thức sau:

a) 3 + 5.0

b) 6/3 + 2 div 3

c) (10* ((45 mod 3)) + 12)) /6

d) (5<=3) And (Not True And (12 Div 3<=1))

Chương 4

CẤU TRÚC CỦA MỘT CHƯƠNG TRÌNH PASCAL

Cấu trúc của một chương trình Pascal bao gồm 3 phần:

– Phần tiêu đề của chương trình.

– Phần khai báo dữ liệu, chương trình con.

– Phần thân chương trình.

Ví dụ 4.1

Một chương trình PASCAL hoàn chỉnh tính diện tích hình tròn bán kính 5.

```
Program ViDu4_1;
Const
   pi = 3.141592653;
Var
   s,r : real;
BEGIN
   r := 5;
   s := pi * r * r;
END.
```

Ta lần lượt đề cập tới từng phần của cấu trúc chương trình:

4.1. PHẦN TIÊU ĐỀ

Phần này không nhất thiết phải có nhưng nó chứa tên chương trình thông thường, tên được đặt sao cho gợi nhớ đến nội dung chương trình, do đó giúp ta phần nào phân biệt được các chương trình khác nhau.

Phần này bắt đầu bằng khoá từ Program.

```
Program ten_chuong_trinh ;
```

4.2. PHẦN KHAI BÁO

PASCAL yêu cầu người lập trình phải liệt kê các đối tượng sẽ được sử dụng trong chương trình. Điều này có nghĩa là tất cả các kiểu dữ liệu, các biến, các thủ tục phải khai báo trước. Một chương trình PASCAl có thể sử dụng các dữ liệu là hằng số hay biến số.

– *Hằng số* là dữ liệu mà giá trị của nó do người lập trình xác định, giá trị này không thay đổi trong suốt quá trình thực hiện chương trình.

– *Biến số* là dữ liệu mà giá trị của nó do chính chương trình xác định, giá trị này có thể thay đổi trong quá trình thực hiện chương trình.

Phần này bao gồm nhiều mục, tuy nhiên tuỳ theo từng chương trình cụ thể, phần khai báo có thể thiếu một hay nhiều mục, thậm chí không mục nào. Nhưng phần này cũng có thể dài hơn phần thân chương trình. Ta lần lượt xét tất cả các mục này.

4.2.1. Phần khai báo Unit

Đây là một nét mới tạo sức mạnh của TURBO PASCAL từ thế hệ thứ tư trở đi, được thể hiện ở chỗ nó chia chương trình ra thành nhiều môđun (1 môđun, hay còn gọi là 1 unit, được hiểu là các dữ liệu + các chương trình con có liên quan) và biên dịch chúng một cách riêng rẽ. Làm như vậy khi thay thế hay sửa chữa một môđun, chúng ta không cần biên dịch lại toàn bộ chương trình. Mặt khác một unit có thể được sử dụng bởi nhiều chương trình khác nhau. Nếu một chương trình nào đó cần một unit cụ thể chỉ cần khai báo nó trước.

Phần khai báo này nếu có thì phải là mục tiêu đầu tiên trong phần khai báo và bắt đầu bằng từ khoá **uses** tiếp theo là các danh sách unit cần khai báo. Ví dụ:

Uses graph, crt, dos;

– Có hai loại unit:

+ Loại do chính người lập trình tạo ra.

+ Loại unit chuẩn.

– Có tất cả 7 loại unit chuẩn:

+ DOS là unit chứa các thủ tục và hàm liên quan tới các thủ tục và hàm của Hệ điều hành DOS.

+ CRT là unit cung cấp các phương tiện xử lý màn hình, bàn phím.
+ PRINTER chứa các dữ liệu về chương trình con giúp chúng ta sử dụng máy in.
+ SYSTEM là thư viện chuẩn.
+ GRAPH cung cấp các thủ tục về đồ hoạ.
+ TURBO3 và GRAPHE3 cung cấp các phương tiện tương thích với TURBO PASCAL 3.0.

Để sử dụng được các thủ tục và dữ liệu trong các unit thì chỉ cần khai báo. Ví dụ, muốn vẽ đồ thị của một thủ tục số, ta cần tới các thủ tục và thủ tục đồ hoạ, muốn vậy phải phai báo như sau:

uses graph;

4.2.2. Phần khai báo kiểu dữ liệu

Đây cũng là một điểm mạnh của TURBO PASCAL, nó cho ta khả năng tự thiết kế ra kiểu dữ liệu mới thuận tiện cho công việc lập trình của mình. Lẽ dĩ nhiên là bất cứ kiểu mới nào cũng được thiết kế từ những kiểu sẵn có của TURBO PASCAL. Một kiểu mới sẽ được mô tả cụ thể bắt đầu bằng từ khoá type.

Type

Tên kiểu = mô tả xây dựng kiểu;

Sau đó có thể khai báo các biến thuộc kiểu dữ liệu mới này.

Ví dụ:

```
Type
   Songuyen = integer;
   Tuoi_tho = 0..100;  //* Miền giá trị của tuổi thọ từ
0 đến 100*
Var
   i: Songuyen;
   tuoi: Tuoi_tho;
```

4.2.3. Phần khai báo hằng

Hằng là đại lượng có giá trị xác định và không thay đổi trong suốt chương trình. Hằng được khai báo bằng từ khoá const như sau:

Const

Tên_hằng = giá_trị_hằng;

Hay

Const

Tên_hằng = biểu thức hằng;

Ví dụ:

```
Const
   Max = 150;          {hằng nguyên}
   L = False;          {hằng logic}
   A= (5*7)/4;         {hằng thực}
   Ch= 'Y';            {hằng ký tự}
   Ho= 'Quoc Trung'; {hằng xâu ký tự}
```

Chúng ta sử dụng các hằng để chương trình được rõ ràng và dễ sửa đổi.

4.2.4. Phần khai báo biến

Biến là một đại lượng có thể thay đổi giá trị, giống như khái niệm biến của toán học. Biến của một chương trình thực chất là tên của một ô nhớ. Tất cả các biến của một chương trình phải được khai báo ở đầu chương trình, sau từ khoá **var**.

Cách khai báo như sau :

```
var
  tb1:k1;
  tb2:k2;
  ...
  tbn:kn;
```

Ở đây các tbi là các tên, còn các ki là các kiểu dữ liệu sẵn có hoặc là các kiểu dữ liệu vừa được định nghĩa trong mục type.

Chú ý: Nếu có nhiều biến cùng kiểu ta có thể khai báo trên cùng một dòng, như trong ví dụ 4.1:

```
Var
  r, s : real;
```

4.2.5. Phần khai báo chương trình con

Mục này bắt đầu bằng từ khoá **function** hay **procedure**. Đó chính là mục khai báo các chương trình con mà chúng ta sẽ nói kỹ ở phần sau.

4.3. THÂN CHƯƠNG TRÌNH

Phần này nằm trọn trong cặp từ khoá **begin ... end** đầu tiên.

Xét lại ví dụ trên:

```
Program ViDu4_1;
Const
  pi = 3.141592653;
Var
  s,r : real;
BEGIN
  r := 5;
  s := pi * r * r;
END.
```

Chương 5

CÁC LỆNH CỦA TURBO PASCAL

5.1. CÁC LỆNH ĐƠN GIẢN

5.1.1. Lệnh gán

Đây là một lệnh cơ bản của TURBO PASCAL, dùng để truyền giá trị của một hằng, của một biến, của một biểu thức vào một biến. Phép gán được biểu diễn bằng ký hiệu ":=".

Cú pháp lệnh:

```
tb := bth;
```

Trong đó:

– tb là tên một biến đã được khai báo ở mục **var**.

– bth là một biểu thức có cùng kiểu với biến tb ở vế trái.

Ví dụ:

```
x := x + 1;
y := x;
z := 3.4;
...
```

Xét thêm một ví dụ sau:

Sau khi đã khai báo:

```
Var
c1, c2: char; i,j: integer; x, y:real;
```

có thể thực hiện các phép gán sau:

```
c1:= 'B'; c2:= chr(7);
i:= (23+6)* mod 3; j:= Round (20/3);
x:= 0.5; y:=1;
```

5.1.2. Các thủ tục vào ra dữ liệu

a) Thủ tục hiển thị dữ liệu ra màn hình

Có 3 dạng viết như sau:

– Write (X1, X2, ..., Xn);

– WriteLn (X1, X2, ..., Xn);

– WriteLn;

Trong đó: Xi là các biểu thức mà giá trị của nó cần hiển thị ra màn hình.

Ví dụ 5.1

Trong ví dụ 4.1, để hiển thị giá trị của diện tích ra màn hình ta cần thêm vào lệnh write như sau:

```
Program ViDu5_01;
Const
  pi = 3.141592653;
Var
  s,r : real;
BEGIN
  r := 5;
  s := pi * r * r;  // công thức tính diện tích
   write( s );
END.
```

Khi cho chạy (thực hiện) chương trình, ta nhìn thấy trên màn hình kết quả diện tích:

7.8539816325E+01

Sự khác nhau giữa hai lệnh write và writeln là ở chỗ vị trí con trỏ màn hình sau khi kết thúc lệnh. Đối với lệnh writeln, sau khi hiển thị các giá trị con trỏ sẽ chuyển xuống đầu dòng tiếp theo. Còn đối với lệnh write, sau khi hiển thị các giá trị, con trỏ sẽ không chuyển xuống đầu dòng tiếp theo.

Còn thủ tục WriteLn sẽ chỉ làm một động tác đơn giản là đưa con trỏ màn hình chuyển xuống dòng tiếp theo.

** Hiển thị có quy cách*

Khi hiển thị kết quả ra màn hình (trong trường hợp giá trị thực) hệ thống thể hiện dưới dạng dấu phẩy động rất khó phân biệt. Vì vậy TURBO PASCAL thiết kế thêm một cơ chế hiển thị có quy cách ra màn hình. Cách viết như sau:

```
Write(X:m:n);
```

Trong đó:

– X là biểu thức có giá trị thực.

– m và n là các số nguyên dương.

Tác dụng của lệnh như sau: hiển thị giá trị của biểu thức biểu thức ra màn hình trên tất cả m cột trong đó có n cột cho phần thập phân. Trong ví dụ 5.1, nếu chúng ta sửa lệnh write(s) thành lệnh write(s:5:2) thì chúng ta sẽ thu được số 78.54 trên màn hình.

```
Program ViDu5_01;
Const
  pi = 3.141592653;
Var
  s,r : real;
BEGIN
  r := 5;
  s := pi * r * r;  // công thức tính diện tích
   write( s:5:2 );
END.
```

Khi cho chạy (thực hiện) chương trình, ta nhìn thấy trên màn hình kết quả diện tích:

78.54

Đối với các giá trị nguyên, cách hiển thị có quy cách được viết như sau:

```
Write(X:m);
```

Trong đó:

– X là biểu thức có giá trị nguyên.

– m là các số nguyên dương, lúc đó máy sẽ dành m chỗ (cột) để hiển thị giá trị của X, nếu còn thừa chỗ, nó sẽ chèn các dấu trắng vào bên trái.

Ví dụ:

```
Var
    I: integer; R: Real; Ch: Char; B:Boolean;
BEGIN
I:= 123; R:=123.456;Ch:= 'D'; B:=2<5; Z:= 543621.342;
    Writeln (I:8);                    {1}
    Writeln (-23564:8);               {2}
    Writeln (R:12:6);                 {3}
    Writeln (35.123456789:12:6);      {4}
    Writeln (R:12)                    {5}
    Writeln (Ch:5)                    {6}
    Writeln ('ABC':5);                {7}
    Writeln (B:7);                    {8}
    Writeln (Z:1:2);                  {9}
END.
```

Cách viết quy cách sẽ căn lề theo bên phải, nếu thừa chỗ thì phần bên trái bỏ trắng.

* *Hiển thị không có quy cách*

Ví dụ:

```
Uses crt;
Var
I: integer; R: Real; Ch: Char; B:Boolean;
BEGIN
       I:= 123; R:= 123.456; Ch:='D'; B:= 2<5;
       Writeln (I);          {1}
       Writeln (R);          {2}
       Writeln (3.14);       {3}
       Writeln (20*2.5)      {4}
       Writeln;
       Writeln (Ch);         {5}
       Writeln (B);          {6}
       Writeln (#7)          {7}
END.
```

Ta sẽ thấy cách viết không có quy cách sẽ căn theo lề bên trái.

b) Thủ tục vào dữ liệu qua bàn phím

Như ta đã thấy, để vào dữ liệu có thể dùng lệnh gán, nhưng trong thực tế nếu dùng lệnh gán để vào dữ liệu thì sẽ rất bất tiện. Vì thế, để vào dữ liệu ta hay dùng lệnh Read và ReadLn. Có 3 dạng viết như sau:

```
Read(b1, b2, ..., bn);
ReadLn(b1,b2, ..., bn);
ReadLn;
```

Trong đó: bi chỉ có thể là các biến.

Hai dạng đầu giúp cho ta gán giá trị cho các biến khi chạy chương trình. Ví dụ, nếu muốn gán cho x giá trị 5, y giá trị 3.45 và z giá trị 12.4 thì lúc đó ta viết trong chương trình lệnh sau:

```
Read(x, y, z);
```

Khi chạy chương trình, lúc gặp lệnh này, máy sẽ dừng lại, đợi ta nhập dữ liệu thông qua bàn phím, cần gõ:

5 3.45 12.4 ↵

Lưu ý là các cụm dữ liệu tương ứng với các biến phải cách nhau ít nhất một dấu cách và phải phù hợp về kiểu cũng như số lượng.

Khi gặp lệnh ReadLn, chương trình sẽ dừng lại và chỉ chạy tiếp khi người sử dụng gõ phím ENTER. Do vậy, nó thường được dùng để dừng màn hình để xem kết quả thực hiện chương trình.

Ví dụ:

Với a, b là hai biến nguyên, x là biến thực. Xét đoạn chương trình:

```
Readln (a,b);
Readln(x);
```

Nếu gõ các phím

```
2 24 6.5 14 <Enter>
```

thì a nhận giá trị 2, b nhận giá trị 24. Các ký tự còn lại bị bỏ qua và không được xét trong thủ tục Readln(x) tiếp theo. Như vậy máy dừng lại ở câu lệnh: Readln(x) để đợi nạp số liệu.

Ví dụ:

Giả sử đã khai báo: Var s1, s2, s3: string [5];

Xét câu lệnh: Readln (s1,s2,s3);

Nếu ấn phím Enter thì cả ba biến s1, s2, s3 đều là xâu rỗng.

Nếu gõ: ABCDE1234567 và phím Enter thì: s1:='ABCDE', S2='12345', S3='67'

c) Kết hợp giữa Write và ReadLn để tạo hội thoại người máy

Trong các ví dụ dùng Read và ReadLn vừa xét thể hiện một yếu điểm là không có chỉ dẫn trên màn hình để báo cho ta biết đang cần đưa giá trị vào cho biến nào. Vì vậy, ta nên kết hợp hai thủ tục Write và ReadLn để tạo một giao diện thuận tiện khi vào dữ liệu.

Ví dụ, ta nên thay lệnh Read(x,y,z) bằng cụm lệnh sau:

```
Write('x = '); ReadLn(x);
Write('y = '); ReadLn(y);
Write('z = '); ReadLn(z);
```

Khi đó lúc chạy chương trình ta thu được màn hình như sau:

```
x = 5 ↵
y = 3.45↵
z = 12.4 ↵
```

Rõ ràng khi tiến hành như vậy sẽ ít phạm phải sai lầm.

Ví dụ 5.2

Viết chương trình nhập họ tên, bậc lương, ngày công trong tháng, phụ cấp, hệ số trách nhiệm, tạm ứng. In ra màn hình họ tên, tiền lương và tiền còn được lĩnh; biết rằng công thức toán như sau:

Tiền lương = Bậc lương/30*Ngày công*Hệ số trách nhiệm+ phụ cấp

Tiền còn được lĩnh = Tiền lương – Tạm ứng.

```
Program ViDu5_02;
   Var
      Ten: String;
      nc, pc, tam: integer;
      bl,hs,tt,cl: Real;
BEGIN
   Writeln ('CHUONG TRINH TINH LUONG');
   Writeln ('----------------------------------------');
   Write ('Cho biet ho ten:');
   Readln (ten);
   Write ('Cho biet bac luong:'); Readln (bl);
   Write ('Cho biet ngay cong:'); Readln (nc);
   Write ('Cho biet he so trach nhiem:'); Readln(hs);
   Write ('Cho biet phu cap khu vuc='); Readln(pc);
   Write ('Cho biet so tien da tam ung ky 1='); Readln
(tam);
      tt:= ((bl/30*nc*hs)+ pc);
      cl:= tt-tam;
   Writeln;
   Writeln ('+ Ong (Ba): ', ten:24);
   Writeln ('+Tien luong trong thang= ',tt:10:2,'dong');
   Writeln ('+So tien con linh = ', cl:10:2,' dong');
   Writeln ('Ban phim <Enter> de ket thuc');
   Readln;
END.
```

d) Lệnh in dữ liệu ra máy in (thủ tục đầu ra dữ liệu)

Để đưa dữ liệu ra máy in ta dùng các lệnh Write và WriteLn với tham số đầu tiên là LST. Vì biến LST chỉ được khai báo trong Unit Printer, cho nên cần khai báo unit này trong mục khai báo **uses**, ví dụ:

```
Program ViDu5_02;
Uses printer;
BEGIN
  Write(lst, 'x = ', 12, ',  y = ', 3.55:4:1);
END.
```

Chương trình này sẽ in ra giấy nội dung sau:

```
x = 12,  y =  3.6
```

5.2. MỘT SỐ HÀM VÀ THỦ TỤC TRÌNH BÀY MÀN HÌNH TRONG PASCAL

Các hàm và thủ tục này nằm trong Unit CRT, vì vậy để sử dụng nó trước hết chúng ta phải có khai báo sau:

Uses CRT;

Sau đây là một số hàm và thủ tục hay dùng:

- GotoXY(x, y) thủ tục này di chuyển con trỏ màn hình đến cột x dòng y, lưu ý rằng màn hình được chia thành 25 dòng và 80 cột, cột đầu tiên được đánh số 1 và dòng đầu tiên cũng được đánh số 1. Toạ độ (1,1) là vị trí góc trên bên trái của màn hình.
- ClrScr là thủ tục xoá toàn bộ màn hình và đưa con trỏ về dòng 1, cột 1 của màn hình.
- WhereX là hàm trả về một giá trị kiểu byte, cho biết con trỏ đang ở cột nào.
- WhereY là hàm trả về một giá trị kiểu byte, cho biết con trỏ đang ở dòng nào.
- Windows(x1, y1, x2, y2) là thủ tục thiết lập cửa sổ hoạt động trên màn hình có góc trái trên ở toạ độ (x1, y1), góc trái dưới ở toạ độ (x2, y2). Khi đã thiết lập cửa sổ hoạt động các toạ độ phải tính lại theo quy tắc:

  – Hoành độ mới = hoành độ cũ - x1 + 1

  – Tung độ mới = tung độ cũ - y1 + 1

Ví dụ muốn di chuyển con trỏ đến dòng 8 cột 10 trên màn hình sau khi đã thực hiện lệnh windows(5,7,75,15) ta viết gotoXY(6,2).

Cách đặt màu nền và màu chữ

Để xác định màu nền ta dùng thủ tục:

```
TextBackground(color);
```

Để xác định màu chữ ta dùng thủ tục:

```
TextColor(color);
```

Trong đó: color là mã của màu. Mã màu là các hằng được định nghĩa trong unit CRT.

Biến color trong thủ tục Text Background có thể nhận 8 giá trị đầu của bảng dưới đây, còn biến color của thủ tục Text color có thể nhận được cả 16 giá trị:

Black = 0 (đen)	DarkGray = 8 (xám đậm)
Blue = 1 (xanh da trời)	LightBlue = 9 (xanh da trời nhạt)
Green = 2 (xanh lá cây)	LightGreen = 10 (xanh lá cây nhạt)
Cyan = 3 (xanh lơ)	LightCyan = 11 (xanh lơ nhạt)
Red = 4 (đỏ)	LightRed = 12 (đỏ nhạt)
Magenta = 5 (tím)	LightMagenta = 13 (tím nhạt)
Brown = 6 (nâu)	Yellow = 14 (vàng)
LightGray = 7 (xám nhạt)	White = 15 (trắng)

Ví dụ các lệnh sau đây sẽ hiển thị lên màn hình dòng chữ Đại Học Bách Khoa màu đỏ:

```
Textcolor(4);
Write('Đại Học Bách Khoa');
```

Chú ý: Thủ tục TextBackground sẽ làm mất hiệu lực của thủ tục TextBackground trước đó. Thủ tục này thường đi kèm với các thủ tục window và clear. Để minh hoạ cách sử dụng ta xét ví dụ sau:

Giả sử ta muốn tạo một cửa sổ làm việc màu xanh da trời với góc trên trái có toạ độ (10,5), góc dưới phải có toạ độ (70,20) ta sử dụng đoạn lệnh sau:

```
Textbackground(1);
Windows(10,5,70,20);
ClrScr;
```

Bây giờ nếu muốn chuyển sang cửa sổ màu tím ta viết tiếp 2 lệnh sau:

```
Textbackground(5);
ClrScr;
```

5.3. CÂU LỆNH ĐIỀU KHIỂN

5.3.1 Câu lệnh IF ... THEN ... ELSE

- *Dạng không đầy đủ:* if bt then lenh;
- *Dạng đầy đủ:* if bt then lenh1
  else lenh2;

Trong đó:

– if, then, else là các từ khoá.

– bt là một biểu thức điều kiện có kết quả là một giá trị có kiểu boolean.

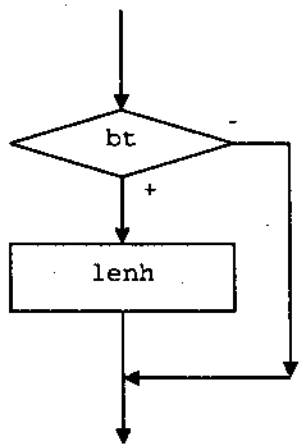
– lenh, lenh1, lenh2 là các lệnh của PASCAL.

Bộ vi xử lý sẽ hoạt động ra sao khi gặp lệnh này?

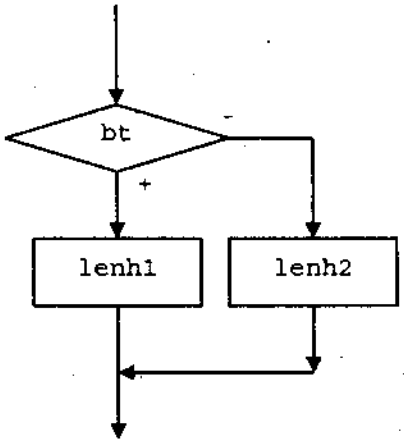
Đối với *dạng không đầy đủ*, nếu *bt* có giá trị đúng thì thực hiện *lenh*; trong trường hợp ngược lại thì bỏ qua không thực hiện *lenh*.

Đối với *dạng đầy đủ*, nếu *bt* có giá trị đúng thì thực hiện *lenh1*; trong trường hợp ngược lại thì thực hiện *lenh2*.

Ta có thể minh hoạ hai dạng này bằng sơ đồ khối sau:



Dạng không đầy đủ



Dạng đầy đủ

Nói tóm lại, câu lệnh if chỉ ra rằng nếu biểu thức cho giá trị đúng thì câu lệnh đứng sau từ khoá then được thực hiện, ngược lại thì hoặc là không có câu lệnh nào hoặc là câu lệnh đứng sau else được thực hiện.

Chú ý: không có dấu ';' trước từ khoá else.

Những câu lệnh if có thể lồng vào nhau nên ta có thể gặp tình huống sau:

```
if bt1 then
if bt2 then lenh1
else lenh2;
```

Lúc đó hệ thống sẽ giải quyết bằng cách tự động hiểu như là đã viết:

```
if bt then begin
if bt2 then lenh1
else lenh2;
end;
```

Điều này có nghĩa là else luôn luôn kẹp đôi với if gần nhất chưa có else.

Sau đây là một số ví dụ về cách sử dụng lệnh if ... then:

Cho hàm số

$$f(x) = \begin{cases} x + \sin x & \text{khi } x \geq 0 \\ x & \text{khi } x < 0 \end{cases}$$

Các câu lệnh sau đây sẽ tính giá trị của hàm sau khi nhận được một giá trị của x đọc vào máy thông qua bàn phím:

```
Write('Hay nhap vao mot gia tri: ');
ReadLn(x);
if x < 0 then f := x
else f := x + sin(x);
```

Với những kiến thức đã học, ta hoàn toàn có thể viết được một chương trình hoàn thiện cho phép tính giá trị của hàm trên.

Ví dụ 5.3

Viết chương trình nhập vào một ký tự viết hoa sau đó hiển thị dạng viết thường của nó:

```
Program ViDu5_3
var
   ch: char;
BEGIN
       write ('nhap chu viet hoa');
       readln (ch); // doc du lieu vao tu ban phim
   if ('a' < ch) and (ch < 'z') then
   begin
      writeln ('chu viet thuong tuong ung la:');
      write (# ord(ch)+32));
   end
      else write ('khong phai la chu cai thuong:');
readln ();
END.
```

Ví dụ 5.4

Chương trình cho phép tìm giá trị lớn nhất trong 3 giá trị đọc vào từ bàn phím.

```
Program ViDu5_4a; { Tim gia tri lon nhat trong 3 so
doc tu ban phim }
Var
     a,b,c : real;
      max : real;
BEGIN
        Write('Hay cho gia tri cua so thu nhat: ');
ReadLn(a);
        Write('Hay cho gia tri cua so thu hai : ');
ReadLn(b);
       Write('Hay cho gia tri cua so thu ba   : ');
ReadLn(c);
    if a > b then
      if a > c then max := a
      else max := c
    else
     if b > c then max := b
      else max := c;
    WriteLn('Gia tri lon nhat trong 3 so la: ',
max:7:2);
    ReadLn;
END.
```

Lưu ý về ký pháp *lùi vào* (cách viết có cấu trúc nhô ra lùi vào): khi viết một chương trình, sử dụng ký pháp này sẽ giúp chúng ta dễ dàng hiểu được ý đồ của thuật toán hay đoạn chương trình. Đây là một tiêu chuẩn cần phải có cho một chương trình nói chung và cho chương trình được viết bằng TURBO PASCAL nói riêng. Các cặp begin ... end tương ứng bao giờ cũng gióng thẳng hàng với nhau.

Tất nhiên là chương trình ở ví dụ 5.4 có thể được viết theo khuôn dạng như sau (không tuân thủ ký pháp *lùi vào*) mà vẫn đảm bảo chạy đúng:

```
Program ViDu5_4b; { Tim gia tri lon nhat trong 3 so
doc tu ban phim }
Var
        a,b,c : real;
        max : real;
```

```
BEGIN
    Write('Hay cho gia tri cua so thu nhat: '); ReadLn(a);
    Write('Hay cho gia tri cua so thu hai : '); ReadLn(b);
    Write('Hay cho gia tri cua so thu ba  : '); ReadLn(c);
          if a > b then
             if a > c then max := a
          else max := c
             else
          if b > c then max := b
          else max := c;
       WriteLn('Gia tri lon nhat trong 3 so la: ',
max:7:2);
    ReadLn;
    END.
```

Hay thậm chí:

```
Program ViDu5_4c; { Tim gia tri lon nhat trong 3 so
doc tu ban phim } Var
  a,b,c : real;  max : real; BEGIN Write('Hay cho
gia tri cua so thu nhat: '); ReadLn(a); Write('Hay cho
gia tri cua so thu hai : '); ReadLn(b); Write('Hay cho
gia tri cua so thu ba  : '); ReadLn(c);
  if a > b then if a > c then max := a else max := c
else if b > c then max := b else max := c;
  WriteLn('Gia tri lon nhat trong 3 so la: ',
max:7:2); ReadLn; END.
```

Tuy nhiên, chúng ta thấy ngay rằng cách viết chương trình trong hai trường hợp b,c thực sự *không sáng sủa* và *không đẹp mắt*. Vì vậy, **tuyệt đối không nên** viết chương trình theo hai cách này.

Ví dụ 5.5

Lập chương trình giải bất phương trình bậc nhất ax+b >0

```
Program ViDu5_05;
   Var a,b: real;
BEGIN
```

```
      Write ('vao a, b =');
  readln (a,b);
      If a = 0 then
         If b >0 then
      Writeln ('Bat phuong trinh dung voi moi x')
         else writeln ('Bat phuong trinh vo nghiem');
      If a>0 then
      Writeln ('nghiem bat phuong trinh la x>:',
-b/a:10:2);
      If a<0 then
      Writeln ('nghiem bat phuong trinh la x<:',
-b/a:10:2);
      Readln;
   END.
```

5.3.2 Câu lệnh lựa chọn CASE

Như đã thấy nếu dùng lệnh if ta chỉ có thể chọn một trong hai khả năng. Trong tình huống cần chọn một trong nhiều khả năng nếu ta dùng lệnh if sẽ có nhiều bất tiện, trong trường hợp này tốt nhất dùng lệnh case. Lệnh này có hai dạng như sau:

Dạng 1:

```
case bt of
  h1: lenh1;
  h2: lenh2;
  ...
  hn: lenhn;
end;
```

Dạng 2:

```
case bt of
  h1: lenh1;
  h2: lenh2;
  ...
  hn: lenhn;
else lenhn+1
end;
```

Trong đó: bt là biểu thức mà kết quả của nó là một giá trị có kiểu là một trong những kiểu sau: integer, byte, char, liệt kê, miền con, nhưng không là kiểu thực.

Ví dụ 5.6

Viết chương trình tính số ngày của một tháng.

```
Program ViDu5_6; { Tinh so ngay cua 1 thang }
Var
  SoNgay, Thang, Nam : integer;
BEGIN
  Write('Hay cho gia tri cua thang : '); ReadLn(Thang);
  Write('Hay cho gia tri cua nam   : '); ReadLn(Nam);
  case Thang of
    4,6,9,11          : SoNgay := 30;
          2           : case (Nam mod 4) of
                0 : SoNgay := 29;
                    else SoNgay := 28;
                    end;
    1,3,5,7,8,10,12 : SoNgay := 31;
  end; { case }
  WriteLn('Thang ', Thang,' cua nam ', Nam,' co ',
SoNgay,' ngay.');
  ReadLn;
END.
```

Ví dụ 5.7

Viết chương trình nhập vào một điểm kiểm tra từ bàn phím và in kết quả xếp loại: loại yếu (dưới 5 điểm), loại khá (7, 8 điểm), loại giỏi (8, 9 điểm).

```
Program ViDu5_7
Var
diem : Integer
BEGIN
          Write ('Cho biet diem:');
          Readln (diem);
   Case diem of
      0..4: Writeln ('Xep loai yeu');
```

```
            5,6: Writeln ('Xep loai trung binh');
            7,8: Writeln ('Xep loai kha');
            9,10: Writeln ('Xep loai gioi')
        Else
            Writeln ('Diem nhap vao sai');
        End;
    Readln;
    END.
```

5.4. CÁC CÂU LỆNH LẶP

Dùng để thực hiện lặp đi lặp lại nhiều lần một (hoặc một số) câu lệnh nào đó, nếu số lần lặp biết trước ta sử dụng lệnh for, nếu không ta dùng lệnh while hay repeat.

5.4.1. Câu lệnh FOR

Câu lệnh for chỉ ra rằng lệnh thành phần phải được thực hiện lặp đi lặp lại nhiều lần chừng nào giá trị của biến điều khiển vẫn còn nằm giữa giá trị đầu và cuối.

Cú pháp lệnh for có 2 dạng sau:

```
For bdk := gtd to gtc do lenh;
For bdk := gtc downto gtd do lenh;
```

Trong đó:

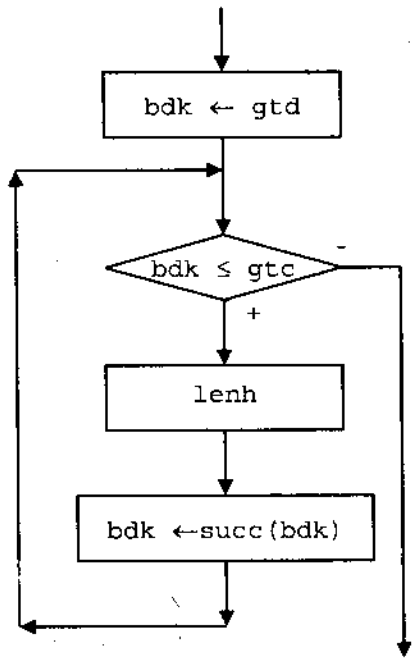
– for, to, downto, do là các từ khoá.

– bdk là biến điều khiển vòng lặp có thể có kiểu integer, char, boolean, miền con, liệt kê, nhưng không được là kiểu thực.

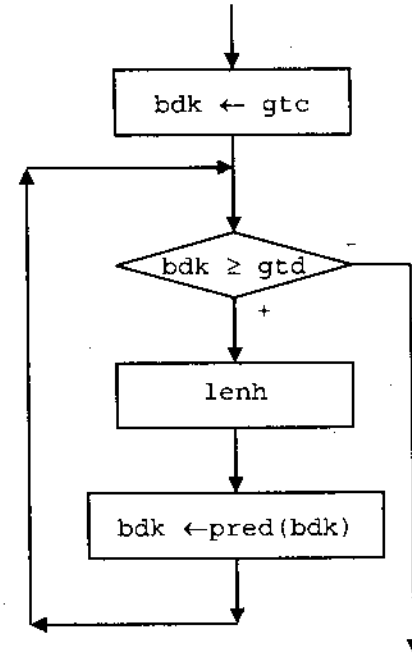
– gtd, gtc là các biểu thức có giá trị có kiểu giống như kiểu của biến điều khiển.

– lenh là những câu lệnh đơn của PASCAL, trong trường hợp có nhiều lệnh cùng được thực hiện, ta phải đặt trong khoá begin ... end.

Ta có thể mô tả cơ chế hoạt động của lệnh for bằng sơ đồ khối như sau:



Dạng 1



Dạng 2

Ví dụ 5.8

Tính giá trị của tổng S = 1 + 1/2 + 1/3 + ... + 1/n, sử dụng lệnh FOR.

```
Program ViDu5_8;
Var
  i,n : integer;
  Tong : real;
BEGIN
       Write('Hay cho gia tri cua n : '); ReadLn(n);
  Tong := 0;
        for i := 1 to n do
   Tong := Tong + 1/i;
       WriteLn('Gia tri cua tong: ', Tong:7:2);
  ReadLn;
END.
```

Một số điểm cần lưu ý:

– gtd, gtc chỉ tính một lần

– Đối với dạng thứ nhất, nếu gtd > gtc lệnh `for` sẽ không thực hiện.

– Đối với dạng thứ hai nếu gtd < gtc lệnh `for` sẽ không thực hiện.

– Biến điều khiển không bị câu lệnh sau do thay đổi giá trị.

– Câu lệnh `for` có thể lồng vào nhau.

Ví dụ 5.9

Dùng vòng FOR để in ra màn hình nội dung sau:

```
1 2 3 4 5 6 5 4 3 2 1
  2 3 4 5 6 5 4 3 2
    3 4 5 6 5 4 3
      4 5 6 5 4
        5 6 5
          6
```

```
Program ViDu5_9;
Uses crt;
Var
  i,j : integer;
BEGIN
  ClrScr;
  for i := 1 to 6 do begin
    gotoxy( 35 + i - 1, 10 + i - 1);
    for j := i to 6 do write(j:2);
    for j := 5 downto i do write(j:2);
    WriteLn;
  end;
  ReadLn;
END.
```

5.4.2. Câu lệnh WHILE

Cú pháp lệnh như sau:

```
while bt do lenh;
```

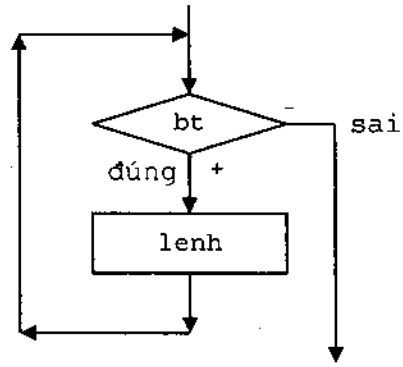
Trong đó:

– while, do là các từ khoá.

– bt là biểu thức logic, biểu thức này sẽ được đánh giá trước khi lặp (vì vậy lệnh while còn gọi là lệnh lặp với điều kiện trước).

– lenh là những câu lệnh đơn của PASCAL, trong trường hợp có nhiều lệnh cùng được thực hiện, ta phải đặt trong khoá begin ... end. lenh sẽ thực hiện lặp lại nếu giá trị của biểu thức điều khiển bt vẫn còn đúng. Tất nhiên nếu giá trị của biểu thức bt này sai ngay từ đầu thì lenh sẽ không thực hiện một lần nào cả.

Ta có thể mô tả cơ chế hoạt động của lệnh while bằng sơ đồ khối như sau:



Sau đây là chương trình tính tổng đã nói đến ở mục trước, nhưng sử dụng lệnh while.

Ví dụ 5.10

Tính giá trị của tổng S = 1 + 1/2 + 1/3 + ... + 1/n, sử dụng lệnh WHILE.

```
Program ViDu5_10;
Var
  i,n : integer;
```

```
  Tong : real;
BEGIN
   Write('Hay cho gia tri cua n : '); ReadLn(n);
        Tong := 0;
        i := 1;
  while i <= n do begin
        Tong := Tong + 1/i;
        i := i + 1;
  end;
  WriteLn('Gia tri cua tong: ', Tong:7:2);
  ReadLn;
END.
```

Ví dụ 5.11

Tính hàm số sinx theo công thức sau:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + ... + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

Số phần tử được chọn cho tới khi đạt được độ chính xác:

$$\left| (-1)^n \frac{x^{2n+1}}{(2n+1)!} \right| < \varepsilon$$

Các giá trị của x và ε đều được nhập vào từ bàn phím.

Để thực hiện phép tính được nhanh chóng và dễ dàng, ta tìm mối quan hệ giữa các số hạng ở vế phải. Nhận thấy rằng nếu gọi các số hạng này là $T_0, T_1, ..., T_n$ ta có:

$$T_o = x$$

$$T_1 = -\frac{x^2}{2k(2k+1)} T_{k-1} \qquad k = 1,2,...,n$$

Sử dụng công thức truy hồi này, ta có chương trình sau:

```
Program ViDu5_11; { Tinh sin cua 1 so }
Var
     eps, x, S, T : real;
     k : integer;
BEGIN
      Write('Hay cho gia tri cua x: '); ReadLn(x);
      Write('Hay cho gia tri cua do chinh xac
epsilon: '); ReadLn(eps);
       S := x;
      T := x;
       k := 1;
  while abs(T) > eps do begin
       T := (-1) * T * sqr(x) / ( 2 * k * (2 * k + 1) );
       S := S + T;
          k := k + 1;
  end;
      WriteLn('Gia tri cua ham sin tai diem ',
x:7:2,' la: ', S:7:5);
      ReadLn;
END.
```

Ví dụ 5.12

Kiểm tra một số nguyên đọc vào từ bàn phím có là một số nguyên tố không.

```
Program ViDu5_12; { Kiem tra xem 1 so co la so
nguyen to khong }
Var
     x,i : integer;
BEGIN
     Write('Hay cho 1 so tu nhien: '); ReadLn(x);
      if x <= 1 then WriteLn('Khong xet!')
   else
```

```
begin
        i := 2;
        while x mod i <> 0 do
       i := i + 1;
   if i = x then WriteLn('So ', x,' la so nguyen to.')
    else WriteLn('So ', x,' KHONG la so nguyen to.');
  end;
  ReadLn;
END.
```

5.4.3. Câu lệnh REPEAT

Cú pháp của lệnh này như sau:

```
repeat
  lenh1;
  ...
  lenhn;
until bt;
```

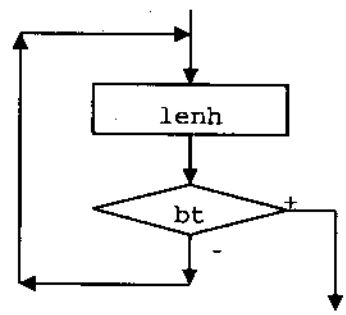
Trong đó:

– repeat, until là các từ khoá.

– lenhi là những câu lệnh của PASCAL.

– bt là biểu thức logic. Câu lệnh repeat sẽ được thực hiện lặp đi lặp lại cho đến khi biểu thức logic này trở nên đúng.

Ta có thể mô tả cơ chế hoạt động của lệnh repeat bằng sơ đồ khối như sau:



Ví dụ 5.13

Tính giá trị của tổng S = 1 + 1/2 + 1/3 + ... + 1/n, sử dụng lệnh REPEAT.

```
Program ViDu5_13;
Var
     i,n : integer;
   Tong : real;
BEGIN
    Write('Hay cho gia tri cua n : ');
    ReadLn(n);
    Tong := 0;
    i := 1;
  repeat
     Tong := Tong + 1/i;
     i := i + 1;
  until i > n;
  WriteLn('Gia tri cua tong: ', Tong:7:2);
  ReadLn;
END.
```

Ví dụ 5.14

Viết chương trình tính n! = 1*2*3*...*n.

```
Program ViDu5_14; { Tinh n giai thua }
Var
  i,n : integer;
  GiaiThua : LongInt;
BEGIN
  Write('Hay cho gia tri cua n : '); ReadLn(n);
   GiaiThua := 1;
    i := 0;
  repeat
      i := i + 1;
    GiaiThua := GiaiThua * i;
```

```
  until i = n;
    WriteLn('Giai thua cua ', n,' la: ', GiaiThua);
  ReadLn;
END.
```

Ví dụ 5.15

Lập trình yêu cầu vào đúng mật khẩu là "phuong" thì mới thoát khỏi chương trình:

```
Uses crt;
Var mk: string[6];
Begin
       Repeat write ('Vao mat khau'); readln (mk);
Until mk = 'phuong';
Writeln ('Ban da vao dung mat khau'); delay (2000)
End.
```

Chú ý:

Khi dùng Repeat và While cần chú ý đến các điểm khác nhau sau đây:

– Trong Repeat, <lệnh> thực hiện trước, kiểm tra điều kiện sau. Còn trong while <lệnh> thực hiện sau, kiểm tra điều kiện trước.

– Trong Repeat, <lệnh> được thực hiện ít nhất một lần, còn trong while có thể không thực hiện.

– Trong Repeat, <lệnh> còn thực hiện khi điều kiện còn sai. Còn trong while, <lệnh> còn thực hiện khi điều kiện còn đúng.

– Trong Repeat, các lệnh không cần đặt giữa Begin và End, còn trong while các lệnh bắt buộc phải để giữa Begin và End nếu có từ hai lệnh trở lên.

BÀI TẬP

5.1. Hãy viết ra màn hình một ký tự, sau đó là mã số ASCII của nó theo dạng sau '0': 48.

5.2. Cho biến x là một số phức. Hãy tìm cách biểu diễn nó trong PASCAL. Hãy cho nó giá trị và viết ra màn hình dạng a+jb nếu phần ảo là dương hoặc a-jb, nếu phần ảo âm.

5.3. In ra các cách để có 200 tờ tiền với 3 loại giấy bạc: 2000đ, 5000đ và 10.000 đ.

5.4. Lập chương trình xếp các dấu * thành tam giác cân n dòng với n đọc từ bàn phím.

Chương 6

CÁC KIỂU DỮ LIỆU MỞ RỘNG

Ngoài các kiểu chuẩn, PASCAL còn có một số kiểu dữ liệu mở rộng.

6.1. KIỂU DỮ LIỆU LIỆT KÊ

PASCAL cho phép người lập trình định nghĩa kiểu dữ liệu mới bằng cách liệt kê các thành phần của dữ liệu, danh sách các thành phần được đặt trong cặp ngoặc đơn, mỗi thành phần cách nhau một dấu phẩy ','. Danh sách đó được mô tả bằng một tên kiểu. Dữ liệu được định nghĩa như vậy được gọi là kiểu liệt kê. Ví dụ:

```
Type
  Mau = (den, trang, xanh, vang);
  ThucPham = (ca, thit, trung, tom);
```

Lúc đó một biến kiểu liệt kê có thể khai báo thông qua tên kiểu đã được định nghĩa trong phần type như sau:

```
Var
   MauSac : Mau;
   ThucAn : ThucPham;
```

Khi đó trong thân chương trình các lệnh sau đây là hợp lệ:

```
   MauSac := xanh;
   ThucAn := thit;
```

Ta có thể khai báo các biến MauSac, ThucAn lại như sau:

```
Var
  MauSac : (den, trang, xanh, vang);
  ThucAn : (ca, thit, trung, tom);
```

Chú ý:

1. Có thể thực hiện phép gán trên các trị kiểu liệt kê.

Ví dụ: Mua:= Quit; Lam:=Nghi; Color:= Red;

2. Các giá trị của các kiểu liệt kê có thể so sánh với nhau theo quy định: trị đi trước nhỏ hơn. Toán tử duy nhất dùng cho kiểu liệt kê là toán tử so sánh.

Ví dụ: Thu<Fri cho kết quả True.

Red>= Blue cho kết quả False.

3. Các hàm chuẩn áp dụng cho kiểu liệt kê.

– *Hàm Ord*: cho thứ tự trị của đối số trong kiểu liệt kê.

Ví dụ: Ord (Sun)=0, Ord(Tuc) = 2

– *Hàm Pred*: cho trị đi trước của đối số trong kiểu liệt kê.

Ví dụ: Pred(Sat) = Fri, Pred (Lambai) = Dihoc

Pred (Sun): cho lỗi thời gian thi hành.

– *Hàm Succ*: cho trị đi sau của đối số trong kiểu liệt kê.

Ví dụ: succ (Fri) = Sat,

Succ (Sat): cho lỗi thời gian thi hành.

4. Không thể nhập, xuất với dữ liệu kiểu liệt kê.

5. Giá trị thuộc kiểu liệt kê thường được dùng để làm chỉ số cho vòng lặp For, các trường hợp chọn trong lệnh Case, chỉ số cho các mảng (array).

Ví dụ 6.1

Chương trình sau đổi thứ ra số. Chủ nhật ứng với số 0, thứ hai đứng với số 1,....

```
Type
   Thu= (ChuNhat, ThuHai, ThuBa, ThuTu, ThuNam,
ThuSau, ThuBay);
Var
   NgayThu: Thu;
```

```
Begin
    Writeln ('Chuong trinh doi thu ra so');
   For NgayThu:= ChuNhat to ThuBay Do
  Writeln (Ord (NgayThu));
  Readln;
  End.
```

Ví dụ 6.2

```
Type
    Ten_thang = (Gieng, Hai, Ba, Tu, Nam, Sau, Bay,
Tam, Chin, Muoi, Muoi_mot, Muoi_hai);
```

Hãy viết chương trình in ra tên tháng bằng Tiếng Anh.

```
Program ViDu6_02;
Type
        Ten_thang = (Gieng, Hai, Ba, Tu, Nam, Sau, Bay,
Tam, Chin, Muoi, Muoi_mot, Muoi_hai); // kiểu liệt kê
Var
    Thang:Ten_thang;
BEGIN
    Writeln ('IN RA TEN THANG BANG TIENG ANH')
    Writeln ('Kieu liet ke ten thang');
    Writeln ('--------------------------------');
    For Thang:= Gieng to Muoihai do
        Case Thang Of
            Gieng  :Writeln ('January');
            Hai    :Writeln ('February');
            Ba     :Writeln ('March');
            Tu     :Writeln ('April');
            Nam    :Writeln ('May');
            Sau    :Writeln ('June');
            Bay    :Writeln ('July');
```

```
            Tam     :Writeln ('August');
            Chin    :Writeln ('Septemeber');
            Muoi        :Writeln ('October');
            Muoimot     :Writeln ('November');
            Muoihai     :Writeln ('December');
        End;
    Writeln;
    Write ('Go phim <Enter> de ket thuc');
    Readln;
END.
```

6.2. KIỂU DỮ LIỆU MIỀN CON (KHOẢNG CON)

Kiểu này cho phép tạo ra những loại dữ liệu mang sắc thái chuyên biệt (chẳng hạn phù hợp với đời thường...) cách định nghĩa như sau:

```
type
  TenKieu = cd..ct;
```

Trong đó:

– TenKieu là một tên tuỳ ý, đặt theo quy định của PASCAL.

– cd và ct là các hằng có cùng kiểu giá trị (integer, char, boolean, liệt kê) thoả mãn điều kiện: cd < ct.

Khi đó các giá trị kiểu miền con sẽ được xác định trong khoảng từ cd đến ct.

Ví dụ:

```
type
  TuoiThanhNien = 15..28;
  TocDoOto = 0..300;
var
  tuoi : TuoiThanhNien;
  VanToc : TocDoOto;
```

Khi đó các phép gán sau là *không hợp lệ*:

```
tuoi := 12;
VanToc := 400;
```

Nhận xét: Kiểu miền con có 2 tác dụng chính:

– Tiết kiệm bộ nhớ.

– Khi chạy chương trình, máy có thể tự động kiểm tra xem biến có vượt ra ngoài giới hạn của nó không.

Ví dụ 6.3

Viết chương trình tính giai thừa của số nguyên n

```
Program ViDu6_03
Type
    So = 1..16;   // kiểu miền con
Var
    n, k: So;
    Gt: LongInt;
Begin
    Writeln ('Tinh giai thua cua n tu 1 den 16');
    Writeln ('Su dung kieu mien con');
    Writeln ('.....');
Repeat
    Write ('Nhap so n (so 0 de dung chuong trinh): ');
    Readln (n);
    Gt:= 1;
    For k:= 1 to n do
    Gt:= Gt*k;
    Writeln ('        ', n ,'! = ',Gt);
    Until n = 0;
End.
```

6.3. KIỂU DỮ LIỆU TẬP HỢP (SET)

Một tập hợp bao gồm một số đối tượng nào đó liên quan với nhau. Một đối tượng trong một tập như vậy gọi là phần tử của tập hợp. Trong tập hợp quan hệ, thứ tự giữa các phần tử không được tính đến.

6.3.1. Định nghĩa kiểu tập hợp

Trong toán học, kiểu của một phần tử trong một tập hợp là bất kỳ, nhưng trong PASCAL không phải như vậy. Các phần tử của một tập hợp phải cùng một kiểu gọi là kiểu cơ sở. PASCAL chỉ chấp nhận các tập không quá 256 phần tử. Điều đó có nghĩa là trong khai báo về phần tử của tập hợp, không thể cho phần tử đó chạy trên một tập nền có quá 256 phần tử.

Một kiểu tập hợp được khai báo như sau:

```
tth : set of kcs;
```

Trong đó:

– tth là một tên tuỳ ý, đặt theo quy định của PASCAL.

– set of là từ khoá.

– kcs chỉ có thể là một trong các kiểu sau: byte, char hoặc là một trong các kiểu liệt kê đã khai báo trong mục type.

Ví dụ:

```
type
  SoChan = set of byte;
  Mau = set of (den, trang, xanh, vang);
  HoaQua = set of (tao, le, nho, chuoi, oi, roi);
var
  t1, t2 : SoChan;
  r1, r2 : HoaQua;
  m1, m2 : Mau;
```

Các khai báo như sau là *không hợp lệ*:

```
type
  SI = set of integer;
  SR = set of real;
  SS = set of string;
```

Để chỉ rõ một tập hợp, cần liệt kê các phần tử của nó trong một cặp ngoặc vuông '[' và ']'. Tập rỗng được biểu diễn là [].

Ví dụ, có thể thực hiện các phép gán:

```
t1 := [1, 2, 3];
t2 := [];
r1 := [tao, le];
```

Ví dụ 6.4

Viết chương trình nhập vào một số nguyên bất kỳ, máy sẽ in ra màn hình số đó có bao nhiêu chữ số:

```
Program ViDu6_04;
Var
    Thuong: Set of 0..9;
    Sodu: 0..9;
    So, Soluu, Sohang: Integer;
BEGIN
    Writeln ('SO CHU SO CUA MOT SO NGUYEN');
    Writeln ('-----------------------------------');
    Write ('- Nhap so nguyen bat ky:  ');
    Readln (So);
    Soluu:=So;
   Thuong:=[];
Sohang:=0;
 Repeat
    Sodu:= So Mod 10;
    Is Not (Sodu In thuong) Then
       Sohang:= Sohang + 1;
    Else
       BEGIN
       Sohang:= Sohang+1;
          Thuong:= Thuong + [Sodu];
       END;
    So:= So Div 10;
 Until So = 0;
 Writeln;
 Writeln ('*So', Soluu, 'co', Sohang, 'chu so');
 Writeln;
 Write ('Go <Enter> de ket thuc');
 Readln;
 END.
```

6.3.2. Các phép toán trên tập hợp

a) Các phép toán quan hệ

– Phép = cho giá trị là TRUE nếu hai tập hợp bằng nhau.

– Phép <> cho giá trị là TRUE nếu hai tập hợp khác nhau.

– Phép <= cho giá trị là TRUE nếu toán hạng bên trái được bao trong toán hạng bên phải.

– Phép >= cho giá trị là TRUE nếu toán hạng bên trái bao toán hạng bên phải.

– Phép toán *in* là toán tử kiểm tra sự có mặt của một phần tử trong một tập hợp. Ví dụ: 'a' in ['a', 'd', 'h'] cho giá trị TRUE, còn 'u' in ['a', 'k'] cho giá trị FALSE.

b) Các phép toán hợp, giao và hiệu

Giả sử A và B là hai tập hợp có cùng kiểu, vậy:

– Toán tử cộng + : A + B là một tập hợp mà các phần tử của nó hoặc thuộc A hoặc thuộc B hoặc thuộc cả hai.

– Toán tử trừ –: A – B là một tập hợp mà các phần tử của nó thuộc A nhưng không thuộc B.

– Toán tử giao * : A * B là một tập hợp mà các phần tử của nó thuộc cả A và B.

c) Phép gán tập hợp

Toán tử gán cũng được dùng để gán kết quả tính toán biểu thức tập hợp cho các biến tập hợp.

Ví dụ 6.5

Viết chương trình nhập vào một chữ cái. Xem xét chữ cái đó là nguyên âm hay phụ âm?

```
Var
   ChuCai, NguyenAm: set of char; Ch:char;
Begin
   ChuCai:= ['A'..'Z', 'a'..'z'];
   NguyenAm:= ['A', 'E', 'I', 'O', 'U'];
Repeat
```

```
   Write ('Nhap mot chu cai:'); readln (Ch);
   Until Ch in ChuCai;
      If upcase (Ch) In NguyenAm then writeln (Ch,' La
nguyen am')
      Else
  Writeln (Ch,' La phu am');
Readln;
End.
```

6.4. KIỂU DỮ LIỆU MẢNG (ARRAY)

Mảng là một cấu trúc bao gồm một số hữu hạn các phần tử có cùng kiểu. Số phần tử của mảng được xác định khi khai báo. Như vậy, một mảng phải có một số cố định các phần tử và được sắp thứ tự: có phần tử thứ nhất, phần tử thứ 2... Ví dụ mảng số nguyên, mảng số thực, mảng xâu...

Phép toán cơ bản liên quan tới mảng là phép truy nhập trực tiếp tới các phần tử của mảng để có thể tìm ra phần tử của mảng nằm ở vị trí đó, hay có thể lưu trữ dữ liệu tại vị trí đó. Phép truy nhập trực tiếp này được thể hiện thông qua tên mảng cùng với chỉ số nằm trong cặp ngoặc vuông []. Ví dụ: x[5], a[2,3]...

6.4.1. Mảng một chiều

Một khai báo mảng phải được đặc tả bởi hai khía cạnh của mảng: *kiểu các phần tử của mảng* và *kiểu của chỉ số*. Khai báo mảng một chiều trong PASCAL có cú pháp như sau:

```
tm : array[csd..csc] of kpt;
```

Trong đó:

– array, of là các từ khoá.

– tm là tên biến mảng cần khai báo.

– csd, csc là chỉ số đầu và cuối của mảng. Chúng phải có cùng kiểu và là một trong những kiểu vô hướng như integer, char, boolean...

– kpt là kiểu của các phần tử của mảng. Kiểu này có thể là bất kỳ kiểu dữ liệu nào của PASCAL, nhưng không được là kiểu file (kiểu này sẽ được đề cập tới sau).

Ví dụ, để khai báo một mảng có tên là a, là mảng gồm 20 số nguyên, ta viết như sau:

```
var
  a : array[1..20] of integer;
```

Hoặc dài dòng hơn:

```
type
  DaySo = array[1..20] of integer;
var
  a : DaySo;
```

Khi đó, trong chương trình, các lệnh gán sau là hợp lệ:

a[1] := 3;

a[2] := 2 + a[1];

a[3] := a[1] + a[2];

a[1] := a[a[1]] + a[2 + 3];

Cách gán giá trị cho biến kiểu mảng:

– Nếu có hai mảng a và b có cùng kiểu như nhau, và ta đã biết giá trị của b thì có thể thực hiện phép gán như sau:

```
a := b;
```

– Ta thường gán giá trị cho từng phần tử của một mảng bằng cách dùng lệnh lặp

Ví dụ, để nhập một dãy 100 số nguyên vào mảng a, có thể dùng nhóm lệnh và khai báo như sau:

```
var
  a : array[1..100] of integer;
  i : integer;
...
  for i := 1 to 100 do begin
    Write('Nhap gia tri cho phan tu thu ',i,' cua
mang: ');
    ReadLn( a[i] );
  end;
```

Sau đây là một số ví dụ minh hoạ cho việc sử dụng mảng để giải quyết một số bài toán:

Nhập một mảng:

Trong đó có:

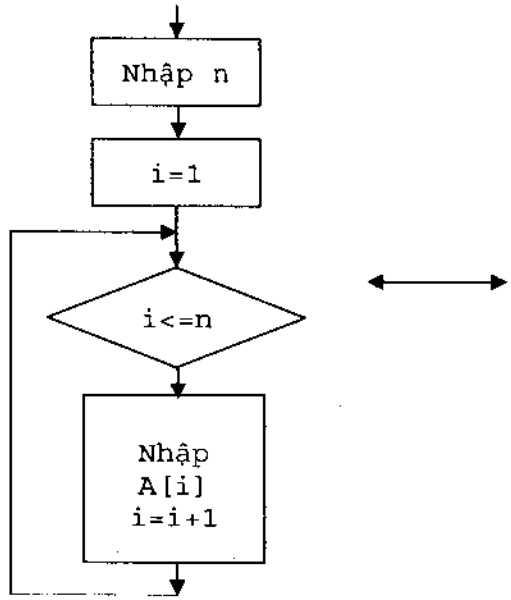
– Nhập một mảng gồm mảng biết trước số phần tử của mảng.

– Nhập mảng chưa biết trước số phần tử của mảng.

**** Nhập mảng biết trước số phần tử của mảng***

Bài toán đặt ra là nhập một số nguyên từ n từ bàn phím rồi nhập n giá trị A1, A2...An từ bàn phím.

Sơ đồ khối của bài toán:




```
write (' Nhap n: ');
readln (n);
      for i: = 1 to
n do
begin
      write
('a[',i,'] = ');
readln (a[i]);
end;
```

Ví dụ 6.6

Viết chương trình nhập một mảng số nguyên A gồm n phần tử.

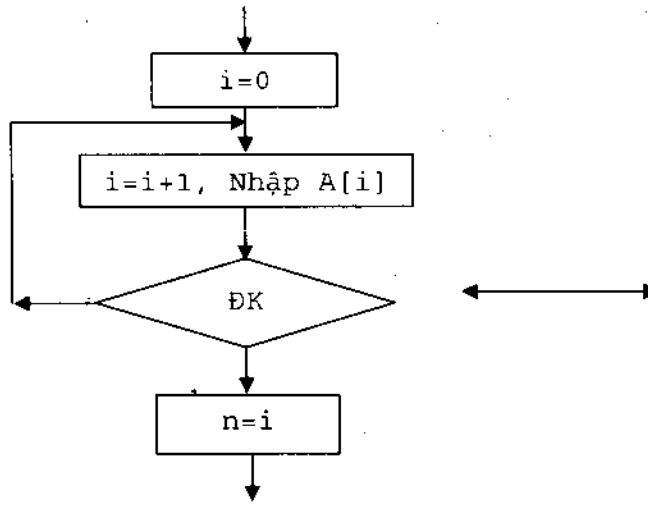
```
Uses crt;
Var a: array [1...100] of integer;
     i,n: integer;
Begin
    Clrscr;
    write (' Nhap n: ');
    readln (n);
```

```
    for i: = 1 to n do
        begin
          write ('a[',i,'] = ');
          readln (a[i]);
      end;
End.
```

**** Nhập mảng chưa biết trước số phần tử của mảng***

Bài toán đặt ra là nhập một dãy số A1, A2...cho đến khi các phần tử thoả mãn một điều kiện nào đó.

Sơ đồ khối của bài toán:




```
i: = 0;
repeat
      i: = i+1;
      write
('a[',i,'] = ');
       readln
(a[i]);
until (ĐK);
n:=i;
```

Chú ý: Phần tử cuối của mảng có thể được xoá hoặc có thể giữ lại tuỳ thuộc vào bài toán cụ thể. Nếu bài toán muốn xoá phần tử cuối cùng thì ta chỉ cần giảm n đi 1 là được.

Ví dụ 6.7

Viết chương trình nhập một mảng số thực A cho đến khi gặp một phần tử âm.

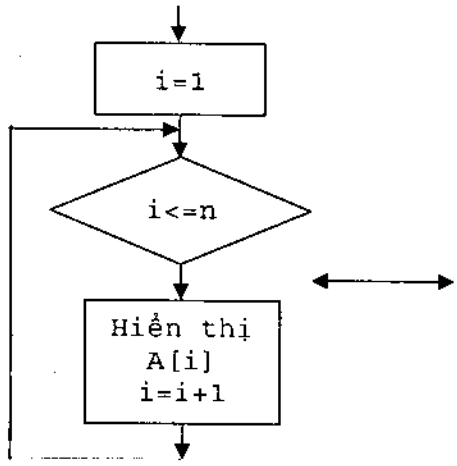
```
Uses crt;
Var a: array [1...100] of integer;
     i,n: integer;
```

```
Begin
      Clrscr;
      i: = 0;
      repeat
           i: = i+1;
           write ('a[',i,'] = ');
           readln (a[i]);
     until (a[i]<0);
     n:=i;
End.
```

**** Hiển thị một mảng***

Khi hiển thị một mảng ta đã biết chính xác số phần tử của mảng, do đó bài toán đặt ra là đã có mảng A gồm n phần tử, hiển thị ra màn hình A1, A2,...An.

Sơ đồ khối của bài toán:




```
for i: = 1 to n do
write (A [i] : 8);
```

Ví dụ 6.8

Viết chương trình nhập vào mảng số thực A gồm n phần tử từ bàn phím. Sau đó hiển thị mảng A đã nhập ra màn hình.

```
Uses crt;
Var a: array [1…100] of real;
      i,n: integer;
```

```
Begin
     Clrscr;
     write (' Nhap n: ');
     readln (n);
     for i: = 1 to n do
        begin
          write ('a[',i,'] = ');
          readln (a[i]);
        end;
    writeln (' Mang A vua nhap la:');
    for i:=1 to n do
      write (A[i]:8:2);
      readln;
End.
```

Chú ý: Thao tác hiển thị thông thường sau các thao tác làm thay đổi giá các phần tử của mảng A như là thao tác nhập, sắp xếp, chèn và xoá....

Ví dụ 6.9

Viết một chương trình thực hiện các công việc sau:

– Đọc một dãy số bao gồm n số thực (n<=50) vào máy thông qua bàn phím.

– Tính tổng các số âm trong dãy này.

```
Program ViDu6_9;
Var : array [1..50] of real;
     i,n : integer;
tong : real;
BEGIN
Repeat
      Write('Cho so phan tu cua day: ');
  ReadLn(n);
until (0 < n) and (n <=50)
 for i:=1 to n do begin
```

```
            Write('Hay nhap phan tu thu ', i,': ');
          ReadLn(a[i]);
        end;
    tong := 0
      for i:=1 to n do
        if a[i] < 0 then
          tong := tong + a[i];
      WriteLn('Tong cua cac so am trong day la: ',
tong:7:2);
      ReadLn;
    END.
```

Ví dụ 6.10

Cho một dãy gồm n số nguyên dương (n ≤ 250). Viết một chương trình thực hiện các công việc sau:

– Đọc số phần tử trong dãy n và dãy số đã cho vào máy.

– Hãy cho biết dãy đã cho có bao nhiêu giá trị khác nhau và mỗi giá trị đó là giá trị của các số hạng nào trong dãy.

```
Program ViDu6_10;
Var
    a : array[1..300] of byte;
  n : integer;
  i,j,k : integer;
    dem : integer;
     kn : set of byte;
    skn : integer;
BEGIN
    repeat
          Write('Cho so phan tu cua day: ');
ReadLn(n);
  until (0 < n) and (n <=250);
      for i:=1 to n do begin
```

```
            Write('Hay nhap phan tu thu ', i,': ');
ReadLn(a[i]);
      end;
           kn := [];
          skn := 0;
      for i:=1 to n do
          if not (a[i] in kn) then begin
           kn := kn + [a[i]];
           skn := skn + 1;
       end;
      WriteLn('Co tat ca ', skn,' gia tri khac nhau:');
dem := 0;
       for i:=1 to n do begin
      if a[i] in kn then begin
           dem := dem + 1;
           WriteLn('Gia tri khac nhau thu ', dem,' la: ',
a[i], ' va la cac so hang o cac vi tri:');
           for j := i to n do
     if a[j] = a[i] then write(j:5);
           WriteLn;
           kn := kn - [a[i]];
        end;       {if}
      end;         {for}
      ReadLn;
    END.
```

Ví dụ 6.11

Cho một dãy số thực $x_1, x_2, ..., x_{100}$. Hãy viết chương trình sắp xếp lại dãy này theo thứ tự tăng dần.

```
Program ViDu6_11; { Sap xep tang dan }
Var
  x : array[1..100] of real;
  i, j : integer;
  tg : real;
```

```
BEGIN
  for i:=1 to 100 do begin
    Write('Hay nhap phan tu thu ', i,': ');
    ReadLn(x[i]);
  end;
  { sap xep day theo thu tu tang dan }
  for i:=1 to 99 do
    for j:=i+1 to 100 do
      if x[i] > x[j] then begin
        tg := x[i];
        x[i] := x[j];
        x[j] := tg;
      end;
WriteLn('Day sau khi sap xep:');
  for i:=1 to 100 do begin
    write(x[i]:7:2);
  end;
  ReadLn;
END.
```

6.4.2. Mảng nhiều chiều

Ngoài mảng một chiều PASCAL còn cho phép làm việc với mảng nhiều chiều vì chúng tỏ ra rất có ích. Ví dụ, mảng hai chiều đặc biệt thích hợp cho việc xử lý dữ liệu được sắp thành hàng và thành cột như khi ta làm việc với ma trận.

Một mảng hai chiều được khai báo như sau:

```
tm : array[csd1..csc1, csd2..csc2] of kpt;
```

Trong đó:

– array, of là các từ khoá.

– tm là tên biến mảng cần khai báo.

– csd1, csc1 là chỉ số đầu và cuối cho hàng của ma trận.

– csd2, csc2 là chỉ số đầu và cuối cho cột của ma trận.

– csd1, csc1, csd2, csc2 đều có cùng một kiểu là một trong những kiểu vô hướng như integer, char, boolean...

– kpt là kiểu của các phần tử của mảng. Kiểu này có thể là bất kỳ kiểu dữ liệu nào của PASCAL, nhưng không được là kiểu file (kiểu này sẽ được đề cập tới sau).

Ví dụ, để khai báo một ma trận 3 hàng 4 cột gồm toàn các số thực ta sử dụng đoạn lệnh như sau:

```
Var
  a : array[1..3, 1..4] of real;
```

Ví dụ 6.12

Chương trình sau đây thực hiện công việc cộng hai ma trận

```
Program ViDu6_12; { Cong hai ma tran }
Var
  a,b,c : array[1..4, 1..5] of real;
  i,j : integer;
BEGIN
  for i:=1 to 4 do
    for j:=1 to 5 do begin
      Write('Hay nhap a[',i,',',j,']: ');
      ReadLn(a[i,j]);
    end;
  for i:=1 to 4 do
    for j:=1 to 5 do begin
 Write('Hay nhap b[',i,',',j,']: ');
      ReadLn(b[i,j]);
    end;
  for i:=1 to 4 do
    for j:=1 to 5 do begin
      c[i,j] := a[i,j] + b[i,j];
    end;
```

```
  for i:=1 to 4 do begin
    for j:=1 to 5 do write(c[i,j]:5:2);
    WriteLn;
  end;
END.
```

Ví dụ 6.13

Viết chương trình thực hiện các công việc sau:

– Đọc hai ma trận vào máy.

– Nhân hai ma trận này với nhau ; hiển thị lên màn hình.

Program ViDu6_13 ;

```
Const
    Max= 50 ;
    Type
    Matran = Array [1..Max, 1.. Max] Of Real ;
Var
    A,B,C : Matran ;
    hangA, cotA, hangB, cotB: integer;
    i,j,k: integer;
Begin
        Repeat
        Writeln ('NHAN 2 MA TRAN');
        Writeln ('Phai nhap cotA = hangB');
        Writeln ('------------------------------');
        Writeln;
        Writeln ('+Ma tran A:');
        Write ('   -So hang: ');
        Readln (hangA);
        Write ('   - So cot:   ');
        Readln (cotA);
```

```
        Writeln ('+Ma tran B:');
        Write ('   -So hang: ');
        Readln (hang B);
        Write ('   - So cot:   ');
        Readln (cotB);
    If CotA <> hangB then
Begin
    Writeln ('Khong nhan duoc, go <Enter> de nhap lai');
Readln;
End;
    Until CotA = hangB;
        Writeln ('NHAP MA TRAN A');
        For i:=1 to hangA do
        For j:= 1 to cotA do
    Begin
        Write ('+Phan tu A [',i,',',j,'] = ');
        Readln (A[i,j]);
    End;
        Writeln ('NHAP MA TRAN B');
        For i:= 1 to hangB do
            For j:= 1 to cotB do
    Begin
        Write ('+phantuB [',i,',',j,'] = ');
    Readln (B [i,j]);
    End;
        Writeln;
        For i:= 1 to hangA do
            For j:=1 to cotB do
    Begin
        C[i,j]:= 0;
        For k:=1 to cotA do
        C[i,j]:= C[i,j] + A [i, k]* B[k,j];
    End;
```

```
        Writeln ('MA TRAN C');
        For i:=1 to hangA do
    Begin
        For j:= 1 to CotB do
    Write (C[i,j]: 6);
    Writeln;
    End;
Readln;
End.
```

6.5. KIỂU XÂU KÝ TỰ (STRING)

6.5.1. Khai báo một kiểu xâu

Để khai báo kiểu xâu, tiến hành như sau :

```
type
  tk = string[n];
```

Trong đó:

– tk là tên kiểu xâu mà do người lập trình đặt.

– string là từ khoá.

– n báo cho máy biết số ô nhớ (cụ thể là byte) tối đa cần dành sẵn cho mỗi biến thuộc kiểu này..

Ví dụ:

```
type
  str10 = string[10];
var
  s1, s2 : str10;
```

Hoặc ngắn gọn hơn:

```
var
  s1, s2 :
string[10];
```

Lúc đó s1, s2 sẽ được cấp phát 1+10 byte bộ nhớ liên tục, ô đầu tiên để lưu trữ độ dài thực tế của xâu, các ô còn lại lưu trữ các ký tự của xâu. Chẳng hạn nếu ta có phép gán:

S1 := 'TIN YEU';

Thì s1 có dạng như sau:

#8	T	I	N			Y	E	U		

Một xâu có thể coi là một mảng các ký tự, ta có thể truy xuất đến từng ký tự của string giống như truy xuất đến từng phần tử của mảng bằng cách dùng đến chỉ số:

Tbx[cs]

Trong đó: Tbx: tên biến xâu; cs: giá trị chỉ số xác định ký tự trong xâu.

Để xử lý xâu, có các phép xử lý sau đây:

Phép gán (:=);

Phép cộng (+) để ghép hai hoặc nhiều xâu với nhau.

6.5.2. Một số hàm và thủ tục xử lý xâu thường dùng

– *Hàm LENGTH(X)* : cho độ dài (số ký tự) của xâu X

Khai báo: `Function Length(S: String): Interger;`

Chiều dài này được chứa trong ký tự đầu tiên của xâu:

`Length(S) = ord(S[0]).`

– *Hàm POS*

Khai báo: `Function Pos(SubStr, S:String): Byte;`

Mô tả: Hàm POS dùng để tìm kiếm một xâu con SubStr trong một xâu lớn S. Hàm này trả lại một giá trị nguyên là chỉ số của ký tự đầu tiên của SubStr trong S. Nếu không tìm thấy SubStr, Pos sẽ trả lại giá trị Zero.

Ví dụ:

```
Var S: String;
Begin
   S:= '123.5';
   While Pos(' ', S) > 0 do {chuyển dấu cách thành zero}
      S[Pos(' ', S)] := '0' ;
End.
```

– Hàm COPY

Khai báo: `Function Copy(S:String; I:Interger; N:Interger) : String;`

Mô tả: Hàm Copy trả lại một xâu ký tự con được lấy ra từ xâu ký tự gốc. S là một xâu ký tự. Hàm này trả lại một xâu có N ký tự bắt đầu từ vị trí thứ I trong xâu S. Nếu I lớn hơn chiều dài của S, hàm trả lại một xâu rỗng. Nếu N chỉ ra nhiều hơn tổng số ký tự tính từ I đến hết xâu ký tự S thì toàn bộ các ký tự đó được trả lại.

Ví dụ:

```
Var S: string;
Begin
    S:= 'ABCDEF';
    S:=  Copy(S,2,3);    {S thành BCD }
End.
```

– Hàm CONCAT

Khai báo: `Function Concat(S1, S2,...Sn ): String;`

Mô tả: Hàm concat sẽ ghép nối tất cả các xâu ký tự S1, S2,....Sn thành một xâu theo thứ tự đã viết.

Lưu ý:

+ Số lượng đối số của hàm Concat phải lớn hơn hoặc bằng 2.

+ Nếu tổng số chiều dài của các xâu lớn hơn 255 thì máy sẽ báo lỗi.

+ Có thể dùng phép (+) để ghép nối xâu ký tự.

– Thủ tục DELETE

Khai báo: `DELETE (St, Pos, Num)`

Mô tả: Đây là thủ tục xoá đi một số ký tự Num (Num là viết tắt của từ tiếng Anh Number: số) kể từ vị trí (Position: vị trí) trong xâu St.

Ví dụ:

Nếu St:= 'ABCDEFG' thì

Delete (St,2,4) làm cho St:= 'AFB'

Delete(St, 2,10) làm cho St:= 'A'

Delete (St, 9,3) làm cho St: = 'ABCDEFG'

– Thủ tục INSERT

Khai báo: `INSERT(S1, S2, Pos)`

Mô tả: Thủ tục này dùng để chèn xâu S1 vào xâu S2 ở vị trí Pos.

Ví dụ:

```
Sn = "Student"; Obj = '12';
```

Sau khi gọi Insert(Obj, Sn, 4);

Hoặc Insert ('12', Sn, 4);

Sn sẽ có dạng như sau: 'Stu12ent';

Lưu ý: Nếu Length(Obj) + Length (Sn) vượt quá độ dài cực đại cho phép thì chỉ những ký tự nào nằm trong khoảng độ dài cực đại cho phép mới được giữ lại.

Ví dụ 6.14

Lập trình nhập một xâu từ bàn phím, rồi hiển thị xâu đảo ngược ; ví dụ nhập vào chuỗi 'ĐHBK' sẽ nhận được 'KBHĐ'.

```
Program Vidu6_14 ;
Var
    St : String ;
    i: Byte;
Begin
        Writeln ('dao chuoi');
        Write ('- Nhap chuoi:');
        Readln(St);
        Writeln;
    For i:= Length (st) downto 1 do
        Write (St [i]);
        Writeln;
        Writeln ('Go phim <Enter> de ket thuc');
        Readln;
End.
```

Ví dụ 6.15

Lập trình đọc một câu vào từ bàn phím sau đó đưa ra màn hình dưới dạng một cột. Ví dụ đọc vào BAI TAP TIN HOC, đưa ra màn hình:

BAI
TAP
TIN
HOC

```
Program ViDu6_15;
Uses crt;
Var x, y, cau: string [30];
a, i: byte;
BEGIN
    Clrscr;
    Write ('Doc vao mot cau:'); readln (cau);
    a:= length (cau);
    for i:= 1 to a do
    Begin
        x:= copy (cau, i); y:= copy (cau, i+1, 1);
        if (x = ' ' ) and (y <> ' ') then writeln
        else write (x);
    End;
    Writeln ; writeln;
    Writeln ('Go ENTER de ket thuc');
    Readln;
END.
```

6.6. KIỂU BẢN GHI (RECORD)

Các kiểu dữ liệu đã xét như mảng (array), tập hợp (set) chỉ mô tả một nhóm các phần tử của cùng một kiểu. Trong các bài toán quản lý xuất hiện nhiều đối tượng cần quản lý mà chúng ta phải dùng nhiều kiểu dữ liệu để mô tả chúng. Để phục vụ mục đích này PASCAL có một kiểu dữ liệu phức hợp đó là kiểu record.

Kiểu dữ liệu record là cấu trúc gồm nhiều thành phần, trong đó các thành phần không nhất thiết phải cùng một kiểu, các thành phần của một record gọi là trường.

6.6.1. Khai báo kiểu record

Một kiểu record được định nghĩa sau từ khoá type theo mẫu sau:

```
type
  tbg = record
          t1 : k1;
          t2 : k2;
          ...
          tn : kn;
        end;
```

Trong đó:

– tbg là tên kiểu bản ghi mà do người lập trình đặt.

– t1, t2, .., tn là tên các trường.

– k1, k2, .., kn là các kiểu dữ liệu sẵn có của PASCAL hoặc là một kiểu dữ liệu vừa mới định nghĩa ở trên.

Ví dụ

```
type
  DiaChi = record
             SoNha : integer;
             Pho   : string[15];
             Quan  : string[15];
             ThanhPho : string[15];
           end;
```

Để mô tả một địa chỉ, cần phải cung cấp hai kiểu dữ liệu khác nhau (integer và string).

Chú ý:

– Nếu các trường trong record thuộc cùng một kiểu, có thể gộp chúng lại, chẳng hạn như trường hợp trên, có thể khai báo lại như sau:

```
type
  DiaChi = record
             SoNha : integer;
             Pho, Quan, ThanhPho : string[15];
           end;
```

– Các record có thể lồng nhau: chẳng hạn tiếp theo khai báo DiaChi ở trên, chúng ta có thể khai báo như sau:

```
type
  LyLich = record
             HoTen     : string[25];
             ChucVu    : string[15];
             Luong     : real;
             ChoO      : DiaChi;
             NgaySinh  : record
                           ngay  : 1..31;
                           thang : 1..12;
                           nam   : integer;
                         end;
           end;
```

6.6.2. Truy nhập vào các trường của một record

Trường của một record đóng vai trò như một biến hay một phần tử của mảng, bởi vậy phép toán nào thực hiện trên các biến thì cũng áp dụng lên trường.

Để truy nhập vào một trường, dùng cú pháp sau:

`tbg1.tbg2...tbg3.tt`

Trong đó:

– tbgi là tên bản ghi trong khai báo.

– tt là tên trường cần truy cập.

Ví dụ, tiếp theo khai báo kiểu lý lịch trên chúng ta có thể khai báo biến của kiểu này:

```
var
  ng1, ng2 : LyLich;
```

Lúc đó trong chương trình các lệnh sau đây sẽ là hợp lệ:

```
ng1.HoTen   := 'Nguyen Van A';
ng1.ChucVu  := 'Giam doc';
ng1.Luong   := 1200000;
ng1.ChoO.SoNha := 45;
ng1.ChoO.Pho   := 'To Hien Thanh';
...
```

hoặc

```
ReadLn(ng1.HoTen);
WriteLn(ng2.ChoO.Quan);
```

Chú ý: các biến cùng kiểu record có thể gán giá trị cho nhau, ví dụ

```
ng1 := ng2;
```

lệnh này giúp ta tránh được một loạt các lệnh gán như sau:

```
ng1.HoTen   := ng2.HoTen;
ng1.ChucVu  := ng2.ChucVu;
ng1.ChoO.SoNha := ng2.ChoO.SoNha;
...
```

6.6.3. Lệnh with ... do ...

Để truy nhập tới một trường, không những phải chỉ tên của trường mà còn phải chỉ ra tên các bản ghi chứa trường này. Điều này gây ra sự mất công và tẻ nhạt trong lập trình. Để khắc phục nhược điểm này ta sử dụng lệnh with:

```
with tbg do begin
  ...
end;
```

Trong khoảng begin ... end, chỉ cần nêu tên các trường là đủ.

Ví dụ, có thể viết lại đoạn lệnh trên như sau:

```
with ng1 do begin
  HoTen  := 'Nguyen Van A';
  ChucVu := 'Giam doc';
  Luong  := 1200000;
  ChoO.SoNha := 45;
  ChoO.Pho   := 'To Hien Thanh';
  ...
end;
```

Hoặc để đơn giản hơn nữa, có thể viết:

```
with ng1, ChoO do begin
  HoTen:= 'Nguyen Van A';
  ChucVu:= 'Giam doc';
  Luong:= 1200000;
  SoNha:= 45;
  Pho:= 'To Hien Thanh';
  ...
end;
```

Theo cách trên, rõ ràng chúng ta có thể truy nhập vào các trường giống như truy cập vào các tên biến thông thường.

6.6.4. Mảng các bản ghi

Giả sử, cần xử lý lý lịch của 100 người, lúc đó tiếp theo khai báo bên trên ta khai báo một mảng:

```
var
  DanhSach : array[1..100] of LyLich;
```

Mỗi phần tử của mảng chính là một biến tương đương với biến được khai báo đơn (như ng1 hay ng2). Ta có thể có các phép truy nhập các biến của mảng như sau:

```
DanhSach[1].HoTen  := 'Tran Ngoc Tri';
DanhSach[1].ChucVu := 'Ky su';
...
```

hay

```
with DanhSach[1] do begin
  HoTen  := 'Tran Ngoc Tri';
  ChucVu := 'Ky su';
  ...
end;
```

Xét ví dụ hoàn chỉnh như sau:

Ví dụ 6.16

Lập chương trình nhập từ bàn phím một danh sách sinh viên gồm: Họ và tên, năm sinh, giới tính, quê quán. Đưa ra màn hình danh sách này và danh sách các sinh viên nữ và sinh sau năm 1975;

```
Program ViDu 6_16;
Uses crt;
Type tulieu = record
    ten:string;
    namsinh: word;
    gioitinh: string;
    quequan: string;
end;
var sinhvien: array [1..50] of tulieu;
n,i, tt: integer;
BEGIN
Clrscr;
Write (' So luong sinh vien = '); readln (n);
Writeln ('Nhap du lieu');
For i:= 1 to n do
with sinhvien [i] do
    begin
    write ('Ho ten sinh vien thu ',i,':'); readln (ten);
    write ('nam sinh');readln(namsinh);
    write ('gioi tinh (nam/nu):'); readln (gioitinh);
    write ('que quan: (tinh)'); readln (quequan);
    end;
```

```
clrscr;
gotoxy(10,2); write ('DANH SACH SINH VIEN');
gotoxy (10, 4); write ('-------------------------');
gotoxy (5,8); write ('TT');
gotoxy (10,8); write ('HO VA TEN');
gotoxy (40,8); write ('NAM SINH');
gotoxy (55,8); write ('GIOI TINH');
gotoxy (70,8); write ('QUE QUAN');
for i:= 1 to n do
with sinhvien [i] do
   begin
      gotoxy (5, 8+i); write ('   ',i);
      gotoxy (10, 8+i); write ('  ', ten);
      gotoxy (40, 8+ i); write (' ', namsinh);
      gotoxy (55, 8+i); write ('  ', gioitinh);
      gotoxy (70, 8+i); write ('  ', quequan);
   end;
readln;
clrscr;
gotoxy(10,2);
write ('DANH SACH SINH VIEN NU SINH SAU 1975');
gotoxy(5,8); write ('TT');
gotoxy(10,8); write ('HO VA TEN');
gotoxy(40,8); write ('NAM SINH');
tt:=0;
for i:= 1 to n do
with sinhvien[i] do
begin
if (gioitinh = 'nu') and (namsinh >1975) then
```

```
        begin
        tt: = tt+1;
        gotoxy (5,8+tt); write ('       ',tt);
        gotoxy (10,8+tt); write ('      ',ten);
        gotoxy (40, 8+tt); write ('     ', namsinh);
        end;
    readln;
END.
```

6.7. KIỂU TỆP (FILE)

Trong những chương trước đã trình bày nhiều phương thức để tạo lập biến và danh sách. Tuy vậy tất cả phương các này đều chịu nhược điểm: khi tắt máy hay mất điện thì toàn bộ dữ liệu sẽ mất. Trong mục này sẽ trình bày cách khắc phục nhược điểm trên nhờ tạo lập các file (tệp tin).

File của PASCAL là một cấu trúc dữ liệu gồm nhiều phần tử cùng kiểu được lưu trữ ở thiết bị nhớ ngoài, số lượng phần tử là không hạn chế.

Chương trình khi chạy thường sinh ra những khối lượng dữ liệu lớn mà sau này sẽ còn sử dụng lại nhiều lần. Vì vậy việc lưu trữ chúng là hoàn toàn cần thiết, muốn vậy ta ghi chúng ra bộ nhớ ở ngoài (ổ đĩa, băng từ...) nhờ cấu trúc file (tệp). Các tệp phân biệt với nhau bởi tên của chúng, mỗi tệp bao gồm một dãy nhiều phần tử thuộc cùng một kiểu, số lượng và nguyên tắc là không hạn chế, điều này có nghĩa là số lượng các phần tử không cần xác định khi khai báo. Thay vào đó PASCAL theo dõi các thao tác truy nhập file thông qua một cơ chế đặc biệt đó là con trỏ tệp: ở một thời điểm nhất định con trỏ tệp luôn luôn chỉ vào một phần tử xác định, mỗi khi phần tử nào đó của tệp được đọc từ tệp hoặc ghi lên tệp, con trỏ sẽ tự động chuyển sang phần tử tiếp theo và chỉ có phần tử đang được con trỏ tệp định vị thì mới truy nhập được. Do tất cả các phần tử của tệp đều có độ dài như nhau cho nên hoàn toàn có thể tính được vị trí của mỗi phần tử riêng biệt, cũng chính vì thế mà ta có thể chuyển con trỏ tệp tới bất kỳ phần tử nào trong file.

6.7.1. Khai báo tệp

Kiểu tệp được định nghĩa bằng:

```
var
  tt : file of kft;
```

Trong đó:

– *file of* là các từ khoá.

– *tt* là tên biến tệp cần khai báo.

– *kft* là kiểu dữ liệu của các phần tử của nó.

Tên biến tệp được khai báo bằng cách sử dụng một kiểu file đã định nghĩa trước đó hoặc bằng cách khai báo kiểu trực tiếp trong dòng khai báo, ví dụ:

```
type
  TI = file of integer;
  TR = file of real;
  NhanSu = record
             Ten      : string[20];
             NgaySinh : integer;
             Luong    : real;
           end;
  HoSo = file of NhanSu;
   var
  f1, f2, f3 : TI;
  f4 : TR;
  f5 : HoSo;
hay
var
  f1, f2, f3 : file of integer;
  f4 : file of real;
  f5 : file of record
                 Ten      : string[20];
                 NgaySinh : integer;
                 Luong    : real;
               end;
```

6.7.2. Các thao tác trên file

a) Tạo tệp để ghi dữ liệu

Để mở một tệp nhằm lưu cất dữ liệu ta dùng cặp lệnh sau:

```
Assign(bt, TenTep);
Rewrite(bt);
```

Trong đó:

– assign, rewrite là các từ khoá.

– bt là biến tệp đã được khai báo trong phần var của chương trình.

b) TenTep là tên tệp do người lập trình đặt (theo quy định của DOS), khi cần thì kèm cả đường dẫn.

Ví dụ:

```
Assign (f1,'SONGUYEN.DAT');
Rewrite (f1);
```

Lệnh đầu gán tệp có tên là SONGUYEN.DAT cho biến tệp f1. Sau khai báo này, về sau mọi thao tác trên f1 sẽ được cập nhật vào tệp trên đĩa với tên là SONGUYEN.DAT.

Lệnh thứ hai là mở tệp để ghi, sau khi thực hiện lệnh này, trên đĩa có một tệp rỗng (không có dữ liệu) có tên là SONGUYEN.DAT, con trỏ tệp chỉ vào đầu tệp.

Sau khi đã mở tệp để ghi bằng hai lệnh trên, ta dùng lệnh sau để ghi dữ liệu vào tệp:

```
Write (bt, X1, X2,..., Xn);
```

Trong đó:

– *bt* là biến tệp đã được khai báo trong phần var của chương trình.

- *Xi* hoặc là một giá trị cụ thể, hoặc là một biến, hoặc một biểu thức có kết quả với kiểu là kiểu thành phần của tệp bt trong khai báo.

Các giá trị của *Xi* sẽ lần lượt được ghi vào tệp ở trên đĩa và sau mỗi lần ghi thì con trỏ của tệp sẽ chuyển tới thành phần tiếp theo, ví dụ:

```
Write (f1, x, x + 1);
for i := 1 to 12 do write (f1,i);
```

Cuối cùng, sau khi kết thúc các truy nhập tệp, ta đóng tệp bằng lệnh:

```
close(bt);
```

Lệnh này sẽ đóng tệp có tên đã gán cho *bt* và cập nhật tệp để phản ảnh trạng thái mới của tệp.

Sau đây là một chương trình hoàn chỉnh nhằm tạo ra một tệp với tên là SONGUYEN.DAT chứa 10 số tự nhiên đầu tiên.

Ví dụ 6.17

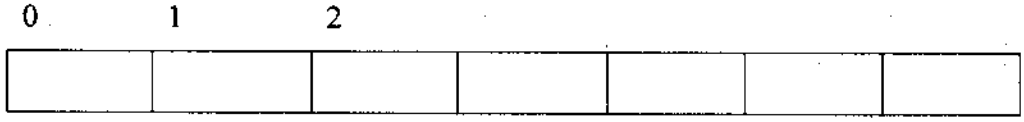
```
Program ViDu6_17;
Var
  i : integer;
  f : file of integer;
BEGIN
  assign(f, 'SONGUYEN.DAT');
  rewrite(f);
  for i:=0 to 9 do write(f, i);
  close(f);
  ReadLn;
END.
```

** Mở tệp để đọc dữ liệu*

Một chương trình muốn sử dụng dữ liệu đang lưu trữ trong một tệp tin, trước hết phải thực hiện thao tác mở tệp để đọc như sau:

```
assign(bt, TenTep);
reset(bt);
```

Lệnh thứ nhất đã được cắt nghĩa, lệnh thứ hai có ý nghĩa như sau: tệp trên đĩa (có tên là TenTep) vừa gán cho biến tệp bt được mở ra để sẵn sàng chờ xử lý, con trỏ tệp chỉ vào phần tử đầu tiên của tệp (phần tử đầu tiên mang chỉ số 0).



Sau khi đã mở tệp để đọc, ta dùng lệnh sau để đọc các phần tử của tệp vào bộ nhớ:

```
Read (bt, X1, X2, ..., Xn);
```

Trong đó:

– *bt* là biến tệp đã được khai báo trong phần var của chương trình.

– *Xi* chỉ có thể là một biến có kiểu là kiểu thành phần của tệp bt trong khai báo.

Các giá trị trong tệp sẽ được đọc lần lượt từ đĩa và gán vào các biến Xi, sau mỗi lần đọc, con trỏ tệp sẽ tự động chuyển tới phần tử tiếp theo. Ví dụ, giả sử tệp SONGUYEN.DAT đã được mở, lúc đó lệnh:

```
read(f, x, y, z);
```

sẽ đọc giá trị 0 và gán vào biến x, 1 vào y và 2 vào z.

Cuối cùng, sau khi đã tiến hành các thao tác, phải đóng tệp bằng lệnh:

```
close(bt);
```

Trong khi tiến hành đọc một tệp, ta hay dùng hàm EOF(bt) để kiểm tra xem con trỏ tệp đã chỉ vào cuối tệp chưa. Đây là một hàm, trả về giá trị có kiểu boolean, cho kết quả là TRUE nếu con trỏ tệp đã chỉ vào vị trí kết thúc của tệp, trong trường hợp ngược lại, nó trả về giá trị FALSE.

Sau đây là một chương trình sử dụng hàm EOF để đọc tệp SONGUYEN.DAT và tính tổng của tất cả các phần tử của nó:

Ví dụ 6.18

```
Program ViDu6_18;
Var
      i,s : integer;
      f : file of integer;
BEGIN
     assign(f,'SONGUYEN.DAT');
     reset(f);
     s := 0;
while not eof(f) do begin
    read(f, i);
    s := s + i;
  end;
```

```
  close(f);
  WriteLn('Tong la: ',s);
  ReadLn;
END.
```

Ví dụ 6.19

(mô tả việc đọc và ghi nội dung vào tệp)

Viết một chương trình thực hiện các công việc sau:

– Đọc vào máy 45 số thực

– Ghi các số vừa đọc vào một tệp có tên "THUC. DAT".

```
Program ViDu6_19;
   var
      i: integer;
      f: file of real;
      x: array [1..45] of real;
BEGIN
   {Mo tep de ghi}
      assign (f,'THUC.DAT');
      rewrite (f);
{Doc so lieu}
for i:= 1 to 45 do
      readln (x[i]);
{Ghi vào tệp}
      for i:=1 to 45 do
write (f, x[i]);
close (f);
END.
```

6.7.3. Truy nhập trực tiếp vào tệp

Như ta đã thấy, trong một thời điểm chỉ có một phần tử của tệp có thể được truy xuất, đó chính là phần tử mà con trỏ tệp chỉ vào. Vì vậy, ta hoàn toàn

có khả năng truy cập vào bất cứ phần tử nào của tệp bằng cách đặt con trỏ tệp chỉ đúng vào phần tử đó. Lệnh sau giúp thực hiện điều này:

```
seek(bt,n);
```

Trong đó:

– seek là một lệnh chuẩn (một thủ tục) có sẵn của PASCAL.

– bt là biến tệp đại diện cho một tệp tin.

– n là một biểu thức có kết quả có kiểu nguyên.

Lệnh này đặt con trỏ chỉ vào phần tử mang số hiệu n (lưu ý là phần tử đầu tiên mang số hiệu 0). Ví dụ đoạn lệnh sau sẽ đọc phần tử thứ 6 trong tệp f vào biến i:

```
seek(f,5);
read(f,i);
```

Bây giờ giả sử đã lưu trữ 10 số thực vào một tệp có tên là SOTHUC.DAT. Chương trình sau sẽ kiểm tra xem các phần tử của nó có đúng không, nếu sai sẽ cho phép người sử dụng sửa phần tử đó:

Ví dụ 6.20

```
Program ViDu6_20;
Var
          i : real;
          j : integer;
          c : char;
          f : file of real;
BEGIN
       assign(f, 'SOTHUC.DAT');
        reset(f);
        j := 0;
 while j <= 9 do begin
    seek(f,j);
    read(f,i);
 WriteLn('Phan tu co so hieu ',j,' la:', i:10:2);
   Write('Co sua khong (C/K):'); ReadLn(c);
```

```
            if c in ['c','C'] then begin
          seek(f, j);
       Write('Cho gia tri moi:'); ReadLn(i);
            write(f, i);
       end;
            j := j + 1;
    end;
    close(f);
    ReadLn;
END.
```

6.7.4. Các hàm và thủ tục xử lý tệp

a) Các hàm

1. FileSize(bt) – hàm này cho biết số lượng phần tử có trong tệp. Khi tệp rỗng (không có dữ liệu), sẽ trả về giá trị 0.
2. FilePos(bt) – hàm này cho biết vị trí hiện thời của con trỏ tệp.

b) Các thủ tục

1. rename(bt, TenMoi) – Thủ tục này cho phép đổi tên tệp đang kết hợp với biến bt bằng một tên mới chứa trong xâu TenMoi.
2. erase(bt) – Thủ tục này cho phép xoá tệp đang kết hợp với biến bt.

Ví dụ 6.21

Xét bài toán: người ta cần quản lý một lớp có tối đa 60 học sinh. Với mỗi học sinh cần quản lý các thông tin sau: họ và tên, tuổi, địa chỉ, điểm toán, điểm văn và phân loại, được tiến hành như sau:

– Nếu điểm (Toán + Văn) < 10 thì phân loại 'kém'.

– Nếu (Toán + Văn < 14) và (Toán + Văn ≥ 10) thì phân loại 'trung bình'.

– Nếu (Toán + Văn < 18) và (Toán + Văn ≥ 14) thì phân loại 'khá'.

– Nếu (Toán + Văn ≥ 18) thì phân loại 'giỏi'.

Sau đây là chương trình:

```
Program ViDu6_21;
Type
  HocSinh = record
              HoTen : string[30];
              Tuoi : byte;
              DiaChi : string[35];
              DiemToan, DiemVan : real;
              PhanLoai : string[10];
            end;
Var
     n, i, j : integer;
     x : HocSinh;
    f : file of HocSinh;
BEGIN
  repeat
    Write('Doc so luong hoc sinh: '); ReadLn(n);
  until (n > 0) and (n <= 60);
  assign(f,'LOP.DAT');
  rewrite(f);
        for i:=1 to n do begin
            WriteLn('Cho cac so lieu ve hoc sinh thu ',i);
       with x do begin
             Write('Ho va ten:'); ReadLn(HoTen);
            Write('Tuoi:'); ReadLn(Tuoi);
            Write('Dia chi:'); ReadLn(DiaChi);
            Write('Diem toan:'); ReadLn(DiemToan);
          Write(' Diem van:'); ReadLn(DiemVan);
           if DiemToan + DiemVan < 10 then PhanLoai:='kem';
            if (10 <= DiemToan + DiemVan) and
          (DiemToan + DiemVan < 14) then PhanLoai:='trung binh';
           if (14 <= DiemToan + DiemVan) and
       (DiemToan + DiemVan < 18) then PhanLoai:='kha';
     if (18 <= DiemToan + DiemVan) then PhanLoai:='gioi';
```

```
    end;
    write(f, x);
  end;
  close(f);
  ReadLn;
END.
```

Ví dụ 6.22

Hãy viết một chương trình cho phép sao chép một tệp nguồn bất kỳ sang tệp đích. Tên tệp nguồn và tên tệp đích được đọc vào từ bàn phím.

Sau đây là chương trình:

```
Program ViDu6_22; { Sao chep tep }
Uses Crt;
Var
     fNguon, fDich : file of byte;
    b : byte;
    TenNguon, TenDich : string[20];
BEGIN
   Write('Cho ten tep nguon: '); ReadLn(TenNguon);
   Write('Cho ten tep dich : '); ReadLn(TenDich);
    assign(fNguon, TenNguon);
    reset(fNguon);
  if IOResult <> 0 then begin
    WriteLn('Mo tep nguon co loi! Thoat.');
    exit;
  end;
     assign(fDich, TenDich);
    rewrite(fDich);
    while not EOF(fNguon) do begin
    read(fNguon, b);
    write(fDich, b);
  end;
```

```
  close(fNguon);
  close(fDich);
  ReadLn;
END.
```

6.8. TỆP VĂN BẢN

Tệp văn bản là tệp mà các phần tử của tệp là các ký tự, nhưng được tổ chức thành dòng với độ dài của các dòng không nhất thiết phải bằng nhau.

Mỗi dòng được kết thúc bằng dấu EOLN (end of line). Trong PASCAL, dấu này được hình thành bởi hai ký tự CR (#10) và LF(#13).

Tệp văn bản kết thúc bởi dấu EOF (end of file). Trong PASCAL, dấu này chính là ký tự Ctrl-Z có mã ASCII 26 (#26).

Ví dụ đoạn văn bản sau:

Tệp văn bản

Rất hay dùng

Được bố trí như sau ở bộ nhớ ngoài:

Tệp văn bản	CR LF	Rất hay dùng	EOF

Chú ý:

– Tệp văn bản được tổ chức theo dòng, nên việc ghi và đọc theo dòng có thể được thực hiện nhờ lệnh WriteLn và ReadLn.

– Tuy tệp văn bản chứa các ký tự, nhưng các lệnh WriteLn, ReadLn và Write, Read vẫn có khả năng ghi và đọc các dữ liệu kiểu integer, real, string nhờ sự chuyển đổi thích hợp (do PASCAL đảm nhiệm).

6.8.1. Khai báo tệp văn bản

Các biến tệp văn bản được định nghĩa theo mẫu sau:

```
var
  bt : text;
```

Trong đó:

– file of là các từ khoá.

– bt là tên biến tệp văn bản cần khai báo.

6.8.2. Ghi vào tệp văn bản

Để ghi vào tệp văn bản, trước hết phải dùng cặp lệnh mở tệp để ghi (như đối với tệp có kiểu):

```
assign(bt, TenTep);
rewrite(bt);
```

Sau đó, sử dụng một trong ba thủ tục sau:

```
- Write(bt, X1, X2, ..., Xn);
- WriteLn(bt, X1, X2, ..., Xn);
- WriteLn(bt);
```

Trong đó:

– bt là tên biến tệp văn bản.

– Xn hoặc là một giá trị cụ thể, hoặc là một biến, hoặc một biểu thức có kết quả có kiểu là một trong những kiểu: integer, real, char, string, boolean.

Dạng thứ hai chỉ khác dạng đầu một chi tiết là sau khi đưa các giá trị vào tệp, nó chèn thêm ký hiệu eoln vào tệp.

Dạng thứ ba chỉ thực hiện việc đưa dấu hết dòng vào tệp.

Ví dụ:

```
WriteLn(f, 4, -2.5,'C','Xau ky tu',TRUE, i + 3 * 3.4);
```

6.8.3. Đọc dữ liệu từ tệp văn bản

Để đọc từ tệp văn bản, trước hết ta phải dùng cặp lệnh mở tệp để đọc (như đối với tệp có kiểu):

```
assign(bt,TenTep);
reset(bt);
```

Sau đó, chỉ có thể đọc tuần tự từ tệp, sử dụng một trong ba thủ tục sau:

```
- Read(bt, X1, X2, ..., Xn);
- ReadLn(bt, X1, X2, ..., Xn);
- ReadLn(bt);
```

Trong đó:

– bt là tên biến tệp văn bản.

– Xi chỉ có thể là biến có kiểu là một trong những kiểu: integer, real, char, string.

Dạng thứ nhất: sau khi đọc hết các giá trị vào các biến từ tệp, con trỏ tệp không tự động chuyển xuống dòng tiếp theo.

Dạng thứ hai: sau khi đọc hết các giá trị vào các biến từ tệp, con trỏ tệp được chuyển xuống dòng tiếp theo.

Dạng thứ ba: chỉ đơn giản là chuyển con trỏ tệp xuống dòng tiếp theo.

PASCAL hỗ trợ hai hàm kiểu boolean để kiểm tra vị trí con trỏ trong tệp văn bản: hàm EOF và EOLN.

Ví dụ, để đọc liên tục các ký tự (các giá trị) cho tới khi gặp dấu hết dòng, ta sử dụng:

```
while not EOLN(bt) do ...
```

và để đọc liên tục cho tới khi hết tệp:

```
while not EOF(bt) do ...
```

6.8.4. Mở tệp văn bản để ghi thêm vào cuối tệp (lệnh append)

Đây là lệnh chỉ dùng riêng cho tệp văn bản, cú pháp như sau:

```
assign(bt,TenTep);
append(bt);
```

Cặp lệnh này mở tệp văn bản đã có trên đĩa và định vị con trỏ tệp ở cuối tệp đã mở. Khi đó, mỗi lần dùng lệnh Write hay WriteLn, thì dữ liệu sẽ được ghi thêm vào ở cuối của tệp này.

Ví dụ 6.23

Giả sử đã có tệp văn bản SO.TXT chứa 50 số nguyên từ 1 đến 50. Lúc đó, để thêm các số từ 51 tới 100 vào cuối tệp văn bản này, ta sử dụng chương trình sau:

```
Program ViDu6_23;
Var
      f : text;
      i : integer;
BEGIN
      assign(f, 'SO.TXT');
     append(f);
```

```
  WriteLn(f,'Cac dong moi them:');
  for i:=51 to 100 do
    WriteLn(f, i);
   close(f);
  ReadLn;
END.
```

Ví dụ 6.24

Viết chương trình đếm xem trong một tệp văn bản có bao nhiêu dòng.

```
Program ViDu6_24; {Dem so dong trong mot tep van ban}
Var
     f : text;
     SoDong : integer;
      TenTep : string;
BEGIN
     Write('Cho ten tep van ban: '); ReadLn(TenTep);
     assign(f, TenTep);
     reset(f);
    SoDong := 0;
  while not EOF(f) do begin
     ReadLn(f);
     SoDong := SoDong + 1;
  end;
  WriteLn('So dong trong tep nay la: ', SoDong);
  close(f);
  ReadLn;
END.
```

6.9. CÁC TỆP TIN THIẾT BỊ CỦA DOS

Trong TURBO PASCAL, các loại tệp được chia thành hai nhóm:

1. Loại các tệp như đã trình bày trong các phần trước.

2. Loại các tệp tin thiết bị: chính là các thiết bị vào ra như bàn phím, màn hình, máy in, các cổng.

Sau đây là bảng tên các tệp tin thiết bị đã được định nghĩa sẵn trong TURBO PASCAL:

Tên	Chức năng	Vào	Ra
CON	Màn hình, bàn phím	x	x
LST	Cổng máy in số 1		x
PRN	Cổng máy in số 1		x
LPT1	Cổng song song số 1		x
LPT2	Cổng song song số 2		x
LPT3	Cổng song song số 3		x
COM1	Cổng nối tiếp số 1	x	x
COM2	Cổng nối tiếp số 2	x	x

Ví dụ 6.25

Chương trình in ra máy in tất cả các số nguyên từ 1 tới 50.

```
Program ViDu6_25; {In ra may in cac so nguyen tu 1 den 50}
Var
  MayIn : text;
  i : integer;
BEGIN
  assign(MayIn, 'PRN'); { Noi may in vao bien tep }
  rewrite(MayIn);
  for i:=1 to 50 do begin
    WriteLn(MayIn, i);
  end;
  WriteLn(MayIn, #12); { Day giay ra khoi may in }
  close(MayIn);
  ReadLn;
END.
```

BÀI TẬP

6.1. Lập một chương trình thực hiện các công việc sau:

a) Đọc từ bàn phím một danh sách học sinh gồm họ tên, giới tính, năm sinh.

b) Ghi dữ liệu ra đĩa với tên tệp đọc từ bàn phím.

c) Tìm các học sinh là nữ và sinh trước 1980 rồi đưa các kết quả ra màn hình.

6.2. Cho ma trận nguyên a = (Aij)4,8. Viết chương trình thực hiện các công việc sau:

a) Đọc ma trận vào máy qua bàn phím; hiển thị ma trận lên màn hình.

b) Đếm xem ma trận trên có bao nhiêu phần tử lớn hơn 10 và chia hết cho 4, đồng thời tính tổng của các phần tử đó, hiển thị lên màn hình.

Chương 7

CHƯƠNG TRÌNH CON (CTC)

Trong chương trình, chúng ta thường gặp hiện tượng có những đoạn chương trình nào đó được thực hiện lặp đi lặp lại nhiều lần, tuy dữ liệu khác nhưng bản chất các công việc lại giống nhau. Do vậy, có thể viết gộp những đoạn chương trình đó lại thành một chương trình con mà khi cần chỉ cần truyền dữ liệu cho nó. Tư tưởng đó cũng dẫn tới việc chúng ta chia một chương trình lớn thành nhiều phần nhỏ rồi giải quyết từng phần. Sau đó ráp nối chúng lại thành một chương trình lớn, các chương trình này sẽ chính là các chương trình con.

Như vậy khái niệm chương trình con cho ta hình ảnh về một dây chuyền sản xuất mang tính công nghiệp cao. Mỗi công đoạn thực hiện một sản phẩm, cuối cùng chúng được ráp nối lại với nhau và sản phẩm mới sẽ ra đời.

Trong PASCAL có hai loại chương trình con: *function* và *procedure*. Như đã nói trước đây, chương trình con phải khai báo ở mục khai báo cuối cùng của phần khai báo.

7.1. HÀM (FUNCTION)

7.1.1. Cấu trúc của một hàm

Cấu trúc của một hàm như sau:

```
function TenHam( DanhSachCacThamSo ) : KieuHam;
  {Phần khai báo}
 ...
  Begin
    {Các lệnh của hàm}
  ...
  End;
```

Trong đó:

– TenHam là tên hàm cần khai báo (do người lập trình tự quyết định, nhưng phải theo quy định đặt tên của PASCAL).

– DanhSachCacThamSo là danh sách các nhóm tham số hình thức, các nhóm này cách nhau bởi một dấu chấm phẩy ';'. Trong cùng một nhóm, các tham số có cùng kiểu thì phân cách nhau bởi một dấu phẩy ','.

– Cuối cùng của hàm là một dấu hai chấm ':' và tên kiểu dữ liệu của hàm (chính là kiểu của kết quả trả về từ hàm).

Chú ý: DanhSachCacThamSo có thể không có.

Nhận xét: về phương diện cấu trúc thì hàm cũng giống như chương trình chính.

Ví dụ 7.1

Viết chương trình tính giai thừa của một số nguyên.

```
Program ViDu7_1;
Function GiaiThua(n:integer):real;
 Var
      i : integer;
      k : real;
 Begin
      i := 0;
      k := 1;
 while i <= n do begin
      i := i + 1;
      k := k * i;
   end;
   GiaiThua := k;
 End;
BEGIN
  WriteLn('Giai thua cua 8 la: ', GiaiThua(8):8:0);
  ReadLn;
END.
```

Nhận xét:

– Trong một hàm (function) bao giờ cũng có ít nhất một lệnh gán kết quả tính toán cho tên của hàm (thường là lệnh cuối cùng của hàm).

– function bao giờ cũng cho ta một giá trị.

Ví dụ 7.2

Viết chương trình tìm ước số chung lớn nhất (USCLN) của hai số nguyên, sử dụng hàm.

```
Program ViDu7_2;
Var
  a, b : integer;
Function USCLN(x, y : integer):integer;
 Var
  SoDu : integer;
 Begin
   while y <> 0 do begin
     SoDu := x mod y;
     x := y;
     y := SoDu;
   end;
   USCLN := x;
 End;
BEGIN
  Write('Cho so a: '); ReadLn(a);
  Write('Cho so b: '); ReadLn(b);
  WriteLn('USCLN cua ',a,' va ',b,' la: ', USCLN(a,b));
  ReadLn;
END.
```

Ví dụ 7.3

Viết chương trình để kiểm tra một số nguyên có là số nguyên tố không, sử dụng hàm.

```
Program ViDu7_3;
Var
  a : integer;
Function NguyenTo(x : integer):boolean;
 Var
  i : integer;
 Begin
   i := 2;
   while x mod i <> 0 do
     i := i + 1;
   if i = x then NguyenTo := true
   else NguyenTo := false;
 End;
BEGIN
  Write('Cho 1 so nguyen: '); ReadLn(a);
  if NguyenTo(a) then WriteLn('So ',a,' la so nguyen to.')
  else WriteLn('So ',a,' KHONG la so nguyen to.');
  ReadLn;
END.
```

7.1.2. Lời gọi hàm

Mỗi khi đã khai báo một hàm, chúng ta có thể gọi nó như gọi một hàm chuẩn của PASCAL.

Ví dụ, một lời gọi hàm USCLN vừa được khai báo ở trên, có thể có dạng như sau:

write(USCLN(6, 27));

Chú ý: Trong TURBO PASCAL, kiểu của hàm có thể là integer, real, char, boolean, string, pointer hay kiểu do người lập trình tự định nghĩa nhưng phải dưới dạng tên kiểu.

Ví dụ, các khai báo sau đây đều là **không hợp lệ**:

```
function f(x : integer) : 0..65;
function g(x : integer) : string[50];
```

Khai báo đúng phải là như sau:

```
type
  K1 : 0..65;
  str50 : string[50];
```

và khai báo chương trình con (hàm) như sau:

```
function f(x : integer) : K1;
function g(x : integer) : str50;
```

7.2. THỦ TỤC (PROCEDURE)

7.2.1. Cấu trúc của một thủ tục

Cấu trúc của một thủ tục như sau:

```
procedure TenThuTuc(DanhSachCacThamSo);
  {Phần khai báo}
 ...
  Begin
    {Các lệnh của thủ tục}
  ...
  End;
```

– TenThuTuc là tên thủ tục cần khai báo (do người lập trình tự quyết định, nhưng phải theo quy định đặt tên của PASCAL).

– DanhSachCacThamSo là danh sách các nhóm tham số hình thức, các nhóm này cách nhau bởi một dấu chấm phẩy ';'. Trong cùng một nhóm, các tham số có cùng kiểu thì phân cách nhau bởi một dấu phẩy ','.

– Cuối cùng của thủ tục chỉ là một dấu chấm phẩy ';'.

Chú ý: DanhSachCacThamSo có thể không có.

Nhận xét: về phương diện cấu trúc thì thủ tục cũng giống như chương trình chính.

Ví dụ:

Viết chương trình con (thủ tục) nhập dữ liệu từ bàn phím, cho phép nhập lại nếu dữ liệu vào bị sai.

```
procedure Nhap(Var x,y,z : real);
 var
  TraLoi : char;
  Begin
    repeat
      write('Vao x, y, z: ');
      ReadLn(x, y, z);
      write('Co sua lai khong (C/K): ');
      ReadLn(TraLoi);
    until TraLoi in ['k','K'];
  End;
```

Chú ý:

Trong TURBO PASCAL, kiểu của các tham số hình thức được khai báo trong DanhSachCacThamSo phải là một tên kiểu. Vì vậy, các khai báo sau là **không hợp lệ**:

```
procedure abc(a1 : array[1..10] of integer);
procedure TT1(var s : string[20] );
```

Khai báo đúng phải được viết như sau:

```
type
  Mang : array[1..10] of integer;
  xau20 : string[20];
```

và sau đó, có thể khai báo thủ tục như sau:

```
procedure abc(a1 : Mang);
procedure TT1(var s : xau20);
```

Ví dụ 7.4

Có tất cả các loại tiền 50.000, 20.000, 10.000, 5.000, 2.000, 1.000. Hãy viết thủ tục cho phép đổi một số tiền ra các loại tiền trên sao cho tổng số tờ là ít nhất.

```
Procedure Doi (St: LongInt);
    Const
        Loai: Array [1..6] Of Integer = (50,20,10,5,2,1);
Var
        i, soto: Integer;
Begin
        i:=1;
        Repeat;
        soto:= st Div loai [i];
        if soto <> 0 Then
    Writeln (soto,' to loai', loai[i],' ngan dong');
        st:= st Mod loai [i];
        i:= i+1;
    until st=0;
End;
```

7.3. LƯU Ý VỀ PHONG CÁCH LẬP TRÌNH

– Khi chỉ cần tính một giá trị thì nên dùng function.

– Thường chương trình con giải quyết một công việc nào đó theo một (hoặc một số) tham số, các giá trị nhập (tham số thực) được cung cấp qua tham trị. Các giá trị xuất qua các tham biến hoặc qua kết quả của hàm, do đó:

+ Trong các chương trình con thường không có lệnh Read() và ReadLn(), vì các giá trị nhập được cung cấp cho các tham số hình thức khi gọi chương trình con.

+ Trong các chương trình con cũng thường không có lệnh Write() và WriteLn(), vì các giá trị xuất thường được xuất ra dưới dạng giá trị của hàm hay qua tham biến.

+ Các lệnh xuất, nhập thường để trong chương trình chính.

+ Tuy nhiên, khi sử dụng chương trình con để làm rõ những giai đoạn trong một công việc lớn hoặc kiểm tra một việc gì đó thì trong chương trình con thường có các lệnh xuất nhập riêng cho bản thân nó.

Sau đây là một ví dụ hoàn chỉnh:

Ví dụ 7.5

Viết chương trình giải phương trình bậc hai.

```
Program ViDu7_5; {Giai phuong trinh bac 2}
Var
  x, y, z : real;
Procedure GiaiPTBH(a, b, c : real);
 Var
  delta : real;
  x1, x2 : real;
 Procedure Delta_KhongAm;
   Begin
     x1 := (-b + sqrt(delta)) / (2 * a);
     x2 := (-b - sqrt(delta)) / (2 * a);
     WriteLn('Nghiem x1: ', x1:7:2);
     WriteLn('Nghiem x2: ', x2:7:2);
   End; { Delta_KhongAm }
 Procedure Delta_Am;
   Begin
     WriteLn('Phuong trinh khong co nghiem thuc.');
   End; { Delta_Am }
 Begin { GiaiPTBH }
   delta := sqr(b) - 4*a*c;
   if delta >= 0 then Delta_KhongAm
   else Delta_Am;
 End; { GiaiPTBH }
BEGIN
  Write('Cho gia tri cua x: '); ReadLn(x);
```

```
  Write('Cho gia tri cua y: '); ReadLn(y);
  Write('Cho gia tri cua z: '); ReadLn(z);
  GiaiPTBH(x, y, z);
  ReadLn;
END.
```

Nhận xét: Trong chương trình trên, có ba chương trình con. Chương trình chính gọi chương trình con GiaiPTBH, đến lượt chương trình con GiaiPTBH lại gọi chương trình con của nó là Delta_KhongAm hoặc Delta_Am tuỳ thuộc vào giá trị của biến delta.

Nguyên tắc hoạt động như sau: nếu gọi chương trình con ở lệnh thứ n, thì sau khi thực hiện xong chương trình con, lại quay về lệnh thứ n.

Ở ví dụ 7.5, khi gặp lệnh ReadLn(x), ta phải đưa vào máy qua bàn phím giá trị của x. Tương tự như vậy, cũng phải cung cấp các giá trị cho y và z từ bàn phím khi các lệnh ReadLn(y) và ReadLn(z) được thực hiện. Sau đó chương trình chính gọi chương trình con GiaiPTBH và truyền cho a, b, c ba giá trị tương ứng x, y, z vừa nhập vào. Nhờ đó mà chương trình con GiaiPTBH tính được delta, sau đó tuỳ theo giá trị của delta mà gọi Delta_KhongAm hoặc Delta_Am. Cuối cùng, khi thực hiện xong một trong hai procedure vừa nói, hệ thống trở lại lệnh gọi GiaiPTBH (nằm trong chương trình chính), gặp lệnh ReadLn, lệnh này dừng màn hình cho ta xem kết quả. Để kết thúc chương trình, gõ phím *enter*.

7.4. CÁCH TRUYỀN THAM SỐ

Để xem xét vấn đề này, xét một bài toán như sau: cho 3 số thực, hãy viết một chương trình cho phép tìm số lớn nhất của ba số trên rồi in ra kết quả.

Ví dụ 7.6

```
Program ViDu7_6; { Tim so lon nhat trong 3 so }
Var
  n1, n2, n3 : real;
  m1, m2 : real;
Procedure SoSanh(a, b : real; Var max : real);
```

```
Begin
   if a >= b then max := a
   else max := b;
 End;
BEGIN
  Write('Cho gia tri cua so thu nhat: '); ReadLn(n1);
  Write('Cho gia tri cua so thu hai : '); ReadLn(n2);
  Write('Cho gia tri cua so thu ba  : '); ReadLn(n3);
  m1 := 0;
  m2 := 0;
  SoSanh(n1, n2, m1);
  SoSanh(n3, m1, m2);
  WriteLn('So lon nhat trong 3 so la: ', m2:7:2);
  ReadLn;
END.
```

Chỉ với một ví dụ đơn giản như trên, ta đã thấy việc sử dụng chương trình con có tham số là cách cho phép một thủ tục được gọi nhiều lần trong một chương trình với các kết quả khác nhau tuỳ theo giá trị thật được truyền cho các tham số khi gọi. Những tham số trong phần khai báo của chương trình con được gọi là tham số hình thức. Danh sách các tham số hình thức được khai báo trong cặp ngoặc đơn ngay sau tên của các thủ tục theo quy tắc sau:

1. Các tham số cùng kiểu được khai báo trong cùng nhóm, cách nhau bởi dấu phẩy.
2. Các tham số hình thức cách nhau bởi dấu chấm phẩy.

Khi chương trình con được gọi, các giá trị thật mới được truyền vào thay thế cho tham số hình thức. Vì vậy, hai danh sách tham số (hình thức và thật) phải phù hợp với nhau về cả 3 phương diện: số lượng, kiểu và thứ tự.

Có hai cách truyền tham số cho chương trình con:

1. Truyền bằng trị thông qua tham trị.
2. Truyền bằng biến thông qua tham biến.

Sau đây chúng ta lần lượt khảo sát hai cơ chế này:

– Truyền bằng biến:

Những tham số hình thức thuộc loại này được khai báo trong chương trình con sau từ khoá *var*, các tham số thật tương ứng với chúng phải là các biến. Các tham số loại này được gọi là tham biến, các giá trị thật truyền cho chúng có thể bị thay đổi trong chương trình con và khi ra khỏi chương trình con sự thay đổi này vẫn còn hiệu lực (nghĩa là khi ra khỏi chương trình con, tham biến vẫn giữ nguyên giá trị cuối cùng mà nó nhận được).

– Truyền bằng trị:

Những tham số hình thức thuộc loại này cũng được khai báo ở trong cặp vòng đơn sau tên chương trình con, nhưng không có từ *var* đứng trước. Các giá trị thật mà nó nhận được từ chương trình mẹ có thể thay đổi trong chương trình con. Nhưng trong mọi trường hợp không làm thay đổi giá trị của tham số thật trong chương trình mẹ.

Quay trở lại chương trình trên ta thấy, ở lần gọi đầu: SoSanh(n1, n2, m1), m1 được truyền bằng biến, điều này có nghĩa là khi ra khỏi chương trình con, m1 sẽ nhận được giá trị mới nhận được trong chương trình con. Chính vì vậy mà nhờ lần gọi thứ hai: SoSanh(n3, m1, m2) ta mới thu được kết quả mong muốn m1 = max{n1,n2,n3}.

Vậy điều gì xảy ra khi khai báo thủ tục như sau:

```
Procedure SoSanh(a, b, max : real);
```

Ta thấy ngay sau lần gọi đầu, SoSanh(n1, n2, m1), m1 vẫn giữ giá trị ban đầu của nó (=0). Do đó, kết quả sẽ nằm ngoài sự mong đợi của chúng ta.

Để hiểu rõ hơn về vấn đề này, xét thêm một ví dụ sau:

Ví dụ 7.7

```
Program ViDu7_7;
Var
  x, y : integer;
Procedure ViDu(x1 : integer; Var x2 : integer);
```

```
    Begin
      x1 := x1 + 1;
      x2 := x2 + 2;
      WriteLn('Gia tri trong chuong trinh con la: ',
x1,' va ', x2);
    End;
  BEGIN
    x := 0;
    y := 5;
    ViDu( x, y );
    WriteLn('Gia tri trong chuong trinh chinh la: ',
x,' va ', y);
    ReadLn;
  END.
```

Khi thực hiện chương trình, các kết quả thu được trên màn hình như sau:

```
Gia tri trong chuong trinh con la: 1 7
Gia tri trong chuong trinh chinh la: 0 7
```

Sở dĩ như vậy là vì x1 là tham trị, còn x2 là tham biến. Như vậy, tóm lại, khi truyền một giá trị cho tham trị thì giá trị được truyền sẽ không thay đổi, còn khi truyền một giá trị cho tham biến thì giá trị đó có thể bị thay đổi.

7.5. PHÂN BIỆT THAM TRỊ VÀ THAM BIẾN

7.5.1. Tham trị

– Không có từ khoá *var* đứng trước khai báo.

– Được cấp một ô nhớ riêng khi chương trình con được gọi và bị xoá bỏ khi chương trình con kết thúc.

– Tham số thật tương ứng là một biểu thức.

– Tham trị thực là một biến cục bộ nhận giá trị khởi đầu là trị của tham số thật tương ứng.

– Những thay đổi của tham trị không gây ảnh hưởng tới giá trị của tham số thật tương ứng.

7.5.2. Tham biến

– Đi sau *var* trong khai báo.

– Tham số thật tương ứng phải là biến.

– Thực chất của việc truyền tham số theo biến là một sự truyền địa chỉ.

– Những thay đổi trên tham biến thực chất là được thực hiện trên tham số thật tương ứng.

– Ngoài ra, cần lưu ý thêm một số điểm sau:

+ Sau khi chương trình con được thực hiện xong mọi biến cục bộ (kể cả tham trị) đều bị xoá bỏ trong bộ nhớ.

+ Khi cần truyền tham số một cấu trúc dữ liệu lớn (ví dụ: mảng) cho một chương trình con, chúng ta nên sử dụng kỹ thuật truyền theo biến, vì nếu truyền theo trị, sẽ tốn thêm bộ nhớ để tạo bản sao và thời gian cần thiết để tạo bản sao đó.

7.6. VẤN ĐỀ TẦM VỰC

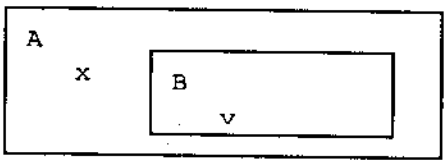
Trong TURBO PASCAL, khi nói đến một khối ta hiểu đó là một chương trình hay một chương trình con.

Trong một khối các đối tượng (nhãn, biến, kiểu, hằng, chương trình con) cần được khai báo trước khi sử dụng. Khai báo một đối tượng là xác nhận sự tồn tại của nó trong một khối.

Vậy một vấn đề cần quan tâm là: một đối tượng có thể được nhận biết ở những nơi nào? Ví dụ ta có khối A (có chứa đối tượng x) chứa khối B (có chứa đối tượng y), khi đó:

– Trong B có quyền truy xuất x không?

– Trong A có quyền truy xuất y không?



Để giải quyết vấn đề này, đưa vào định nghĩa sau: *tầm vực của một đối tượng là vùng nó được biết tới, có thể được sử dụng.*

Ta có quy tắc sau: Tầm vực của một đối tượng trải ra từ chỗ nó được khai báo đến hết khối mà khai báo đó thuộc về, kể cả mọi khối trong khối đó. Ngoại trừ các trường hợp sau:

– Trường hợp có khai báo lại trong một khối con, tức là nếu khối A chứa khối B và trong cả hai khối đều có khai báo x thì trong khối B chỉ có thể truy xuất đến đối tượng x của chính nó (x của A và của B không phải là một đối tượng, chúng chỉ trùng tên). Ta nói đây là hiện tượng bị che.

– Trường hợp nhãn và goto: nếu trong một khối có lệnh goto X thì nhãn X phải được khai báo ngay trong khối đó.

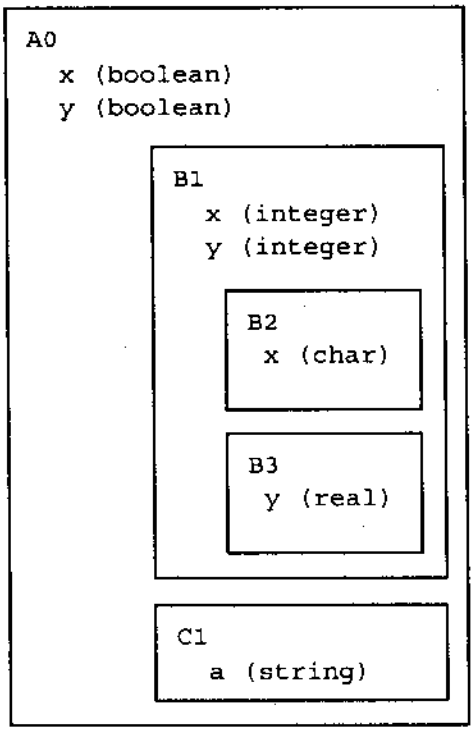
Sau đây sẽ xem xét một vài trường hợp để làm rõ vấn đề này.

7.6.1. Vấn đề tầm vực của một chương trình con

Như ta đã thấy, một chương trình có thể có nhiều chương trình con và trong mỗi chương trình con lại có thể khai báo nhiều chương trình con khác. Vậy với mỗi chương trình con nơi nào có thể gọi nó thực hiện? Đó là vấn đề tầm vực của chương trình con.

Để dễ dàng hiểu vấn đề này, xét ví dụ sau:

```
Program A0;
  var x,y : boolean;
    Procedure B1;
    var x,y : integer;
    Procedure B2;
      var x : char;
      begin
        :
      end; { B2 }
    Procedure B3;
      var y : real;
      begin
        :
      end; { B3 }
```



```
begin { B1 }
      :
    end; { B1 }
  Procedure C1;
    var a,b : string;
    begin { C1 }
      :
    end; { C1 }
BEGIN
  :
END.
```

Chúng ta hãy trả lời những câu hỏi sau:

➢ *Thủ tục B2 có thể được gọi (truy xuất) ở những nơi nào trong chương trình?*

Vì B2 được khai báo trong B1 nên B2 chỉ có thể được truy nhập ở bên trong B1, nghĩa là: có thể gọi B2 từ những nơi sau:

– Trong thân thủ tục B1.

– Trong thân thủ tục B2 (gọi đệ quy).

– Trong thân thủ tục B3.

Như vậy, tầm vực của B2 là toàn bộ B1 (hay nói nôm na là tầm vực của một chương trình con là toàn bộ phạm vi cha của nó). *Chú ý là không thể truy xuất B2 hay B3 từ C1.*

➢ *Thủ tục B1 có thể được gọi (truy xuất) ở những đâu?*

Tầm vực của B1 là A0 cho nên:

– A0 có thể gọi B1.

– C1 có thể gọi B1.

– B1 có thể gọi B1, nghĩa là:

+ Trong thân B1 có thể gọi B1.

+ Trong thân các chương trình con của B1 (B2, B3) có thể gọi B1.

Tương tự, C1 có thể được truy xuất từ trong A0, C1, B1...

7.6.2. Vấn đề tầm vực của biến: biến toàn cục, biến cục bộ, biến không cục bộ

Trở lại với ví dụ trên chúng ta có:

– Những biến khai báo trong B1 là những biến cục bộ trong thân B1, chúng được biết đến trong toàn bộ B1, do vậy có thể được truy xuất trong B2 và B3.

– Trong thân của A0 và thân của C1 những biến cục bộ của B1 không được biết đến.

– Những biến được khai báo trong A0 gọi là biến toàn cục, những biến này có thể được truy xuất từ khắp nơi trong toàn bộ chương trình.

Kết luận: Tầm vực của một biến là phạm vi khối khai báo mà nó thuộc về.

Tuy nhiên, trong B2 có khai báo một biến trùng tên với một biến của B1 (x), cho nên, trong B2 chỉ biết tới biến mà chính nó khai báo (biến x có kiểu char).

Sau đây là một số ví dụ giúp chúng ta nhận thức rõ hơn về các loại biến:

Ví dụ 7.8

```
Program ViDu7_8;
Var
  x : integer;
Procedure ViDu1;
 Var
  x : integer;
 Begin
   x := 1;
 End;
BEGIN
  x := 0;
  ViDu1;
  WriteLn('Gia tri cua x la: ', x); {se in ra gia tri x = 0}
  ReadLn;
END.
```

Khi chạy chương trình, ta sẽ thu được giá trị 0 trên màn hình, vì biến x của chương trình chính và của thủ tục ViDu1 là khác nhau. Sau khi thủ tục ViDu1 chạy xong, các biến của nó sẽ bị xoá khỏi bộ nhớ, vì vậy, không gây ảnh hưởng gì đến cha của nó (là chương trình chính).

Ví dụ 7.9

```
Program ViDu7_9;
Var
  x : integer;
Procedure ViDu2;
 Begin
   x := 1; {x chinh la bien toan cuc}
 End;
BEGIN
  x := 0;
  ViDu2;
  WriteLn('Gia tri cua x la: ', x); {se in ra gia
tri x = 1}
  ReadLn;
END.
```

Khi chạy chương trình, sẽ thu được giá trị 1 trên màn hình.

Chú ý: Trong thân chương trình con, nên hạn chế việc truy xuất đến các biến *không cục bộ*. Sự thay đổi các biến không cục bộ trong thân chương trình con gọi là hiệu ứng lề. Hiệu ứng lề làm chương trình kém trong sáng, khiến khó quản lý các biến và làm cho chương trình chạy sai, nhất là khi dùng biến không cục bộ làm biến điều khiển vòng lặp.

Ví dụ 7.10

```
Program ViDu7_10;
Var
  x : integer;
Procedure ViDu3( x : integer );
 Begin
   x := 1;
 End;
```

```
BEGIN
  x := 0;
  ViDu3( x );
  WriteLn('Gia tri cua x la: ', x);   ReadLn;
END.
```

Khi chạy chương trình, ta sẽ thu được giá trị 0 trên màn hình vì khi gọi ViDu3(x), tham trị x của ViDu3 được cung cấp giá trị khởi đầu là trị của tham số thật x (tức là 0). Sau đó, phép gán x := 1 được thực hiện và ViDu3 hoàn thành nhiệm vụ. Tham trị x thực chất là một biến cục bộ nên bị xoá bỏ khi thoát khỏi chương trình con. Vì vậy, trị của biến toàn cục không hề bị thay đổi và vẫn là 0.

Ví dụ 7.11

```
Program ViDu7_11;
Var
  x : integer;
Procedure ViDu4( Var x : integer );
 Begin
   x := 1;
 End;
BEGIN
  x := 0;
  ViDu4( x );
  WriteLn('Gia tri cua x la: ', x); {se in ra gia tri x = 1}
  ReadLn;
END.
```

Khi chạy chương trình, ta sẽ thu được giá trị 1 trên màn hình, tại sao?

Ví dụ 7.12

Viết thủ tục cho phép sắp xếp một dãy số thực bao gồm tối đa 100 phần tử theo thứ tự tăng dần.

```
Program ViDu7_12;
Type
  Day100 = array[1..100] of real;
Var
  i, SoPT : integer;
  a : Day100;
Procedure SapXepTang( Var x : Day100; SoPT : integer );
 Var
  i, j : integer;
  tg : real;
    Begin
   for i := 1 to SoPT - 1 do
     for j := i + 1 to SoPT do
       if x[i] > x[j] then begin
         tg := x[i];
         x[i] := x[j];
         x[j] := tg;
       end;
 End;
Begin
       Write('Cho so phan tu: '); ReadLn(SoPT);
          for i := 1 to SoPT do begin
        Write('Cho gia tri cua phan tu thu ',i,':');
ReadLn(a[i]);
 end;
  SapXepTang( a, SoPT );
  WriteLn('Day sau khi sap xep la:');
       for i := 1 to SoPT do
Begin
    Write(a[i]:7:2);
 end;
 ReadLn;
END.
```

Ví dụ 7.13

Viết thủ tục tính tích hai ma trận.

```
Program ViDu7_13;
Const
  Max = 10;
Type
  MaTran = array[1..Max, 1..Max] of real;
Var
  m,n,k : integer;
  a,b,c : MaTran;
Procedure NhanMaTran(a, b : MaTran; Var c : MaTran;
m,n,k : integer);
 Var
  i, j, t : integer;
 Begin
   for i := 1 to m do
     for j := 1 to k do begin
       c[i,j] := 0;
       for t := 1 to n do
         c[i,j] := c[i,j] + (a[i,t] * b[t,j]);
       end;
 End;
BEGIN
(* Ta nhap du lieu cho 2 ma tran a va b o day.
  Gia su ma tran a co kich thuoc m x n
  Gia su ma tran b co kich thuoc n x k
  Vay kich thuoc cua ma tran tich c la m x k
  *)
  NhanMaTran( a,b,c,m,n,k );
  (*
  Toi day ta da co ma tran ket qua c la tich cua 2
ma tran a va b.
  *)
  ReadLn;
END.
```

BÀI TẬP

7.1. Viết hàm tính số từ có trong một xâu ký tự.

7.2. Viết thủ tục cho phép tính số từ trong một xâu ký tự.

7.3. Viết chương trình con tính diện tích của các hình vuông, tròn, thang trong một chương trình. Sau đó hỏi và chọn một phương án tính diện tích bằng cách chọn trong bảng chọn lệnh sau:

0. Rời khỏi chương trình. Quay về DOS.

1. Tính diện tích hình vuông.

2. Tính diện tích hình tròn.

3. Tính diện tích hình thang.

Chương 8

ĐỒ HOẠ (GRAPHS)

8.1. CHẾ ĐỘ ĐỒ HOẠ

8.1.1. Khởi tạo và tắt đồ hoạ

Để khởi động chế độ đồ hoạ trước tiên ở mục khai báo uses, phải khai báo unit GRAPH như sau:

```
uses graph;
```

sau đó trong chương trình, trước khi sử dụng các lệnh đồ hoạ, cần khởi động chế độ đồ hoạ bằng cặp lệnh:

```
x: = detect; Init Graph (x, y, path);
```

x,y là các biến integer đã khai báo; còn path là đường dẫn tới thư mục chứa các tệp tin có đuôi là *.BGI.

CloseGraph: chấm dứt chế độ đồ họa, trở về màn hình văn bản.

Ví dụ 8.1

Ví dụ sau minh hoạ cách khởi động và tắt chế độ đồ hoạ:

```
Uses Graph;
    Var
    Gd, Gm, : Integer;
    ErrCode : Integer;
    Begin
    Gd : = Detect;
    InitGraph (Gd, Gm, 'C:\BP\BGI');
    ErrCode : = GraphResult;
    If ErrCode <> grOk then { Thông báo lỗi }
```

```
        Writeln ('Graphics error :', GraphErrorMsg
(ErrCode))
    Else
    Begin { Vẽ đường chéo màn hình }
    Line (0,0, GetMaxX, GetMaxX);
    Readln ;
    CloseGraph;
    End;
End.
```

8.1.2. Chuyển đổi giữa hai chế độ văn bản và đồ họa

Thủ tục CloseGraph chấm dứt chế độ đồ hoạ và thu hẳn về màn hình văn bản. Muốn trở lại chế độ đồ hoạ, phải khởi tạo lại từ đầu, việc này mất nhiều thời gian. Unit Graph cung cấp hai thủ tục RestoreCrtMode và SetGraphMode để chuyển đổi qua lại giữa hai chế độ văn bản và đồ hoạ.

Ví dụ 8.2

Chương trình sau minh hoạ cách chuyển đổi giữa hai chế độ Text và Graphics

```
    Uses Graph;
    Var
        Gd, Gm : Integer;
        Mode   : Integer;
    Begin
        Gd := Detect;
        InitGaph(Gd, Gm, 'C:\BP\BGI');
        If GraphResult <> grOk then Halt (1);
        Out Text ('Bam ENTER de tam thoi thoat khoi che
do Text !');
        Readln;
RestoreCrtMode;
Writeln ('Day la che do Text,');
```

```
            Write ('Bam ENTER de tro lai che do Graphic !')
            Readln;
            SetGraphMode (GetGraphMode);
            OutTextXY(0, TextHeight ('H'), Bam ENTER de
ket thuc chuong trinh');
            Readln;
            CloseGraph;
```

Cuối cùng, ta xét mẫu chung của một chương trình có dùng đồ hoạ là:

```
Uses Graph;
Var
Gd, Gm: integer;
BEGIN
Gd: = Detect;
InitGraph (Gd, Gm, 'C:\TURBO5\BGI');
If GraphResult  <> grOk then Halt (1);
```

(các thủ tục vẽ đồ hoạ để ở đoạn này)

.....

```
Readln;  (Chờ ấn Enter để xem màn hình)
CloseGraph;
END.
```

8.2. VĂN BẢN TRÊN MÀN HÌNH ĐỒ HOẠ

8.2.1. Hiển thị văn bản

Để hiển thị văn bản, ta sử dụng hai lệnh sau:

- Lệnh OutText (v_ban:string);
- Lệnh OutTextXY (x, y: integer; v_ban:string);

Trong đó v_ban là xâu ký tự cần hiển thị, còn tham số x, y: string trong lệnh thứ hai xác định vị trí hiển thị của v_ban;

8.2.2. Phông chữ trên màn hình đồ hoạ

Các phông chữ được lưu trữ bằng các tệp tin có đuôi là *. CHR, các phông này cho ta các kiểu chữ cùng các kích thước khác nhau của chúng. Để nạp phông chữ ta sử dụng thủ tục:

```
SetTextStyle (Font, Dir, Size:word);
```

Ở đây Dir có thể lấy một trong hai hằng giá trị:

0 sẽ cho văn bản nằm ngang màn hình;

1 sẽ cho văn bản nằm dọc màn hình;

còn Size có thể lấy các giá trị từ 0 đến 10 cho kích thước ký tự.

Font có thể lấy các giá trị sau đây:

0 Cho kiểu chữ ngầm định;

1 Cho kiểu chữ TriplexFont;

2 Cho kiểu chữ SmallFont;

3 Cho kiểu chữ SanserifFont;

4 Cho kiểu chữ GothicFont.

Các giá trị do thủ tục này thiết lập sẽ giữ nguyên cho tới khi gọi lại nó với tham số khác hay đóng chế độ đồ hoạ bằng thủ tục CloseGraph.

8.2.3. Bề rộng và chiều cao của văn bản

Để biết rõ số điểm ảnh do ký tự choán chỗ theo bề rộng và chiều cao ứng với FONT chữ hiện hành ta dùng hai hàm sau:

TextHeight (vb:string):word (cho biết số điểm ảnh theo chiều cao).

TextWidth (vb:string):word (cho biết số điểm ảnh theo chiều rộng).

8.3. CÁC THỦ TỤC VẼ HÌNH

8.3.1. Vẽ điểm

Thủ tục Putpixel (x, y: integer): word;

Thủ tục này vẽ một điểm có màu tại điểm có tọa độ x, y trên màn hình đồ họa.

Hàm Getpixel (x,y : integer): word;

Hàm này cho kết quả là màu của điểm tại tọa độ x, y.

8.3.2. Vẽ đường thẳng

Để vẽ đoạn thẳng ta dùng ba lệnh sau:

– line (x1,y1, x2,y2:integer): vẽ đoạn thẳng nối hai điểm có tọa độ là (x1, y1) và (x2,y2);

– lineTo(x,y:integer): vẽ đoạn thẳng nối vị trí hiện hành của CP với điểm có tọa độ là (x, y);

– lineRel(dx,dy: integer); vẽ đoạn thẳng đi từ vị trí hiện hành của CP đến một điểm có độ chênh lệch theo tọa độ với vị trí hiện hành là dx, dy.

8.3.3. Một số hình cơ bản

– Vẽ hình chữ nhật: *rectangle* (x1,y1,x2,y2: integer) lệnh này vẽ lên màn hình hình chữ nhật có góc trên trái tại tọa độ (x1,y1) còn góc dưới phải có tọa độ (x2,y2).

– Vẽ hình chữ nhật đặc *bar* (x1, y1,x2,y2: integer).

– Vẽ đường tròn tâm (x,y) bán kính r: *circle* (x, y,r).

– Vẽ elip hoặc một cung elip: *ellipe* (stAngle, endAngle, xR, yR).

Ví dụ sau đây mô tả vẽ một hình tam giác:

```
Uses Graph;
    Const
        Triangle: array [1...4] of
        PointType = ((X: 50; Y:100), (X: 100; Y:100),
(X:150; Y:150), (X:50; Y:100));
    Var Gd, Gm: Integer;
Begin
    Gd: = Detect;
    InitGraph (Gd, Gm, '');
If GraphResult <> grOK then Halt (1);
    DrawPoly (SizeOf (Triangle)div SizeOf(PointType),
Triangle);
Readln;
CloseGraph;
End.
```

8.4. MÀU SẮC TRONG CHẾ ĐỘ ĐỒ HOẠ

8.4.1. Hằng biểu diễn màu

Mỗi chế độ đồ hoạ đều cho phép vẽ với một số màu khác nhau. Các màu này đã được Turbo PASCAL đặt tên cho dễ nhớ thay vì phải dùng các mã số. Các hằng mô tả màu có giá trị như sau:

Hằng	Giá trị	
Black	Đen	0
Blue	Xanh	1
Green	Xanh lá cây	2
Cyan	Xanh lơ	3
Red	Đỏ	4
Magenta	Tía	5
Brown	Nâu	6
LightGray	Xám nhẹ	7
DarkGray	Xám đậm	8
LightBlue	Xanh nhạt	9
LightGreen	Xanh lá cây nhạt	10
LightCyan	Xanh lơ nhạt	11
LightRed	Hồng	12
LightMagenta	Tía nhạt	13
Yellow	Vàng	14
White	Trắng	15

8.4.2. Bảng màu (Color Palette)

Số màu được hiển thị trên màn hình tuỳ thuộc vào kiểu loại màn hình sử dụng:

Màn hình CG1: 4 màu

Màn hình EGA, VGA: 16 màu (nếu dùng trình điều khiển VGA256.BGI của Borland có thể hiện thị 256 màu).

Bảng màu điều khiển các màu sắc hiển thị trên màn hình, thay đổi bảng màu sẽ dẫn đến thay đổi màu của từng điểm ảnh đã vẽ. Bảng màu trong bộ nhớ thực chất là một mảng chứa các giá trị màu. Có thể dùng chỉ số này để xác định màu cho điểm và đường.

Ví dụ màu xanh trong Graph unit mang giá trị 1 (hay Blue), nhưng màu thật sự trên màn hình còn tuỳ thuộc vào giá trị màu của Palette {1}. Nếu đổi giá trị này thành Red thì tất cả cá điểm có màu xanh đã vẽ và sắp vẽ sẽ trở thành màu đỏ.

Bảng màu được định nghĩa trong Turbo Pascal như sau:

```
Const
    Maxcolor = 15;
Type
    PalettType = Record
        Size : Byte;
End;
```

Size : Số màu tối đa của bảng màu.

Color : Mảng xác định giá trị của bảng màu

Nếu khai báo biến Pcolor có kiểu Palettype thì số màu trong bảng được xác định như sau:

```
Pcolor . Size = GetPalettesize : integer
```

GetMaxColor: xác định giá trị màu lớn nhất trong bảng

Setpalette (ColorNum : Word; Color : ShortInt) định nghĩa mầu ColorNum trong bảng màu.

SetPalette (IP : PaletteType) định nghĩa lại toàn bộ bảng màu theo trị giá xác định trong biến PI.

8.4.3. Xác định màu nền và màu vẽ

Trong màn hình đồ hoạ, đơn vị thể hiện nhỏ nhất là các điểm ảnh (pixel), chúng chỉ có thể xuất hiện ở một màu nhất định. Như vậy, màu vẽ xem như màu của các điểm ảnh tạo nên hình ảnh cần thể hiện và màu nền chính là màu của tất cả các điểm ảnh còn lại trên màn hình.

Hằng giá trị màu giống như trong chế độ Text (Black=0, Blue=1, ..., White =15).

SetColor (Clr : Integer) định nghĩa màu vẽ Clr là chỉ số màu trong bảng palette màu

SetBkColor(Bk : Integer) xác định màu nền.

Ví dụ 8.3

```
Uses Graph;
Var
G1, G2: Integer;
Palette: PaletteType;
Begin
       G1: = Detect;
       InitGaph (G1,  G2, '');
       If GraphResult <> grok then Halt (1);
       Line (0,  0,  GetmaxX, GetmaxY);
   With Palette do.
         Begin
          Size : = 4;
          Colors [0] : = 5; {màu nền}
          Colors [1] : = 3;
Colors [2] : = 1;
Colors [3] : = 2;
SetAllPalette (palette)
       End;
       Readln;
       CloseGraph;
   End.
```

8.5. DANH MỤC CÁC THỦ TỤC VÀ HÀM TRONG THƯ VIỆN ĐỒ HOẠ

Turbo PASCAL có rất nhiều thủ tục và hàm phục vụ cho đồ hoạ. Vì điều kiện khuôn khổ sách có hạn. Chúng tôi xin liệt kê một cách ngắn gọn ra

đây danh sách các thủ tục và hàm cùng các chức năng của chúng. Bạn cần xem chi tiết hơn cách dùng cùng các ví dụ trong cuốn " Turbo PASCAL, Cẩm nang tra cứu".

Arc	Vẽ một cung tròn có gốc bắt đầu, góc kết thúc. Toạ độ tâm.
Bar	Vẽ một hình chữ nhật có tô bên trong với kiểu tô và màu.
Bar3D	Vẽ hình chữ nhật theo không gian 3 chiều có tô bên trong.
Circle	Vẽ hình tròn.
ClearDevice	Xoá màn hình. Đưa vị trí hiện tại về góc trên - bên trái.
ClearViewPort	Xoá khung nhìn.
CloseGraph	Đóng chế độ đồ hoạ.
DetectGraph	Kiểm tra phần cứng và xác định trình điều khiển và chế độ.
Drawpoly	Vẽ đa giác với kiểu nét vẽ và màu hiện tại.
Ellipse	Vẽ một cung elip.
FillEllipse	Vẽ một cung elip có tô.
FillPoly	Tô một đa giác có sử dụng bộ chuyển đổi quét.
FloodFill	Tô một miền bị chặn, dùng mẫu tô và màu hiện tại.
GetArcCoords	Nhận lại toạ độ để vẽ cung.
GetAspectRatio	Trả lại hệ số tương quan tỷ lệ trên màn hình.
GetBkColor	Nhận lại màu nền hiện tại.
GetColor	Nhận lại màu vẽ hiện tại.
GetDefaultPalette	Nhận lại bảng màu ngầm định.
GetDriverName	Nhận lại tên vỉ mạch đồ hoạ.
GetFillPattern	Nhận lại mẫu tô.
GetFillSetting	Nhận lại mẫu tô được thiết lập mới nhất.
GetsGaphMode	Nhận lại chế độ đồ hoạ hiện tại.
GetImage	Cất ảnh bit của một vùng hình vào trong bộ nhớ đệm.
GetLineSettings	Nhận lại kiểu vẽ, nét vẽ, độ dày nét vẽ.
GetMaxColor	Nhận lại giá trị màu lớn nhất có thể có của chế độ đồ hoạ.

GetMaxMode	Nhận lại giá trị chế độ cao nhất có thể có.
GetMaxX	Nhận lại giá trị độ phân giải ngang.
GetMaxY	Nhận lại giá trị độ phân giải dọc.
GetModeName	Nhận lại tên chế độ đồ hoạ.
GetModeRange	Nhận lại chế độ lớn nhất và thấp nhất vỉ đồ hoạ.
GetPalettteSize	Nhận lại kích thước bảng màu.
Getpixel	Nhận lại màu của điểm vẽ.
Getpalett	Nhận lại giá trị bảng màu.
GetTetSettings	Nhận lại giá trị về kiểu chữ, hướng viết, kích thước...
GetViewSettings	Nhận lại thông tin về khung hình và các tham số.
GetX	Nhận lại toạ độ X của vị trí đồ hoạ hiện tại.
GetY	Nhận lại toạ độ Y của vị trí đồ hoạ hiện tại.
GraphDefaults	Đưa vị trí hiện tại về góc trên bên trái, khởi động lại hệ thống đồ hoạ.
GraphErrorMsg	Nhận lại các dấu ký tự thông báo lỗi cho error code.
GraphResult	Nhận lại giá trị báo lỗi của thao tác đồ hoạ cuối cùng.
ImageSize	Trả lại giá trị số byte cần thiết để cất một vùng chữ nhật trên màn hình.
InstallUsesDrive	Cài đặt các trình điều khiển đồ hoạ mới vào bảng BGI.
InstallUsesfont	Cài đặt một phông chữ mới chưa có trong hệ thống BGI.
InitGraph	Khởi tạo để vào chế độ đồ hoạ.
Line	Vẽ một đoạn thẳng giữa hai điểm chỉ rõ.
LineRel	Vẽ một đoạn thẳng với khoảng cách tương đối.
LineTo	Vẽ một đoạn thẳng từ điểm hiện tại tới...
MoveRet	Dịch chuyển vị trí hiện tại tới điểm mới theo toạ độ tương đối.
MoveTo	Dịch chuyển vị trí hiện tại tới điểm mới.
OutText	Viết ra dòng văn bản tại vị trí hiện tại.
OutTextXY	Viết ra dòng văn bản tại vị trí được chỉ ra rõ ràng.
PieSlice	Vẽ một miếng bánh tròn.

PutImage	Nạp hình ảnh bit vào màn hình.
PutPixel	Vẽ một điểm ảnh tại toạ độ X, Y.
Rectangle	Vẽ một hình chữ nhật không tô bên trong với màu và nét vẽ hiện tại.
RegisterBGIdriver	Đăng ký trình điều khiển BGI với hệ thống đồ hoạ.
RegisterBGIfont	Đăng ký phông BGI với hệ thống đồ hoạ.
RestoreCrtMode	Khôi phục lại chế độ màn hình gốc trước khi chế độ được khởi tạo.
Sector	Vẽ và tô một miếng cung hình ellip.
SetActivepage	Thay đổi trang tích cực để cho ra đồ hoạ
SetAllpalette	Thay đổi toàn bộ bảng màu.
SetAspectRatio	Thay đổi tỷ lệ tương quan ngang dọc.
SetBkColor	Đặt màu nền.
SetColor	Đặt màu vẽ hiện tại.
SetFillpattern	Đặt mẫu tô do người sử dụng định nghĩa.
SetFillStyle	Đặt mẫu và tô màu.
SetGraphBufsize	Thay đổi kích thước bộ nhớ đệm đồ hoạ để quét và tô nhúng.
SetGraphMode	Đặt hệ thống tới chế độ đồ hoạ và xoá màn hình.
SetLineStyle	Đặt kiểu nét vẽ.
Setpalette	Thay đổi giá trị bảng màu.
SetRGBPPalette	Thay đổi giá trị bảng màu cho vi mạch IBM8514 và VGA.
SetTextjustify	Đặt chế độ căn lề cho OutText và OutTextXY.
SetTextStyle	Thiết lập phông chữ, hướng, kích thước viết chữ đồ hoạ.
SetUserCharSize	Thay đổi kích thước độ rộng và chiều cao phông vectơ.
SetViewport	Thiết lập khung nhìn đồ hoạ.
SetVisualpage	Thiết lập cách thức ghi lên màn hình vẽ là COPY đè lên hay XOR.
TextHeight	Hàm trả lại độ cao của xâu chữ, tính theo pixel.

MỘT SỐ ĐỀ BÀI KIỂM TRA THAM KHẢO

ĐỀ 1

Bài 1

Một dãy số N số thực $X_1, X_2, \ldots, X_n$ được gọi là dãy không giảm nếu $X_1 \leq X_2 \leq \ldots \leq X_{N-1} \leq X_n$

Viết một chương trình thực hiện lần lượt các việc sau:

a) Lập chương trình con hàm KT để kiểm tra dãy số X có N số thực có là dãy không giảm. Hàm KT nhận giá trị TRUE nếu dãy là không giảm hoặc nhận giá trị FALSE khi ngược lại.

b) Nhập từ bàn phím một số nguyên K, với $50 < K \leq 100$.

c) Nhập từ bàn phím K số thực rồi lưu chúng vào mảng A. Sử dụng chương trình con trên để kiểm tra tính không giảm của dãy số vừa nhập, sau đó đưa ra màn hình một thông báo thích hợp trong các thông báo sau:

- LA DAY SO KHONG GIAM
- KHONG LA DAY SO KHONG GIAM

Bài 2

Tệp định kiểu có tên Phanso.dat được đặt tại thư mục gốc của ổ đĩa cứng C. Mỗi phần tử của tệp là một bản ghi lưu trữ hai số nguyên, chúng lần lượt là tử số và mẫu số của một phân số (mọi mẫu số đều khác không) **Record TS, MS: Integer; end.**

Kích thước của tệp khá lớn nên không thể lưu trữ toàn bộ tệp tại bộ nhớ trong. Viết một chương trình lần lượt làm các việc sau:

a) Lập chương trình con hàm tính ước số chung lớn nhất của hai số nguyên A và B với B luôn khác không. Khai báo hàm là:

Function USCLN (A,B: Integer): Integer;

b) Phân số ở dạng tối giản là phân số có ước số chung lớn nhất của tử số và mẫu số bằng 1. Đọc từng phân số của tệp Phanso.dat. Sử dụng chương trình con trên để rút gọn phân số đó về dạng tối giản rồi lưu trữ vào tệp văn bản có tên PSTG.TXT cũng đặt tại thư mục gốc của ổ đĩa cứng C. Mỗi dòng

của tệp văn bản ghi hai số nguyên là tử số và mẫu số của phân số tối giản theo quy định: mỗi số chiếm 10 vị trí.

c) Đưa ra màn hình số lượng phân số đã ghi lên tệp PSTG.TXT mà giá trị của phân số là một số nguyên.

ĐỀ 2

Bài 1

Một xâu ký tự không rỗng được gọi là xâu chữ nếu ký tự đầu là chữ, các ký tự còn lại là chữ hoặc là dấu cách.

Viết một chương trình lần lượt làm các việc sau:

a) Lập chương trình con hàm xác định một xâu ký tự có là xâu chữ hay không. Khai báo hàm là :

Function XauChu (S:string): boolean;

b) Nhập từ bàn phím hai xâu ký tự không rỗng. Dùng hàm trên để kiểm tra hai xâu rồi đưa ra màn hình một thông báo thích hợp trong các thông báo sau:

- CA 2 XAU DEU LA XAU CHU
- CA 2 XAU DEU KHONG LA XAU CHU
- CHI CO XAU THU NHAT LA XAU CHU
- CHI CO XAU THU HAI LA XAU CHU

Bài 2

Tệp văn bản có tên PHANSO.TXT được lưu tại thư mục gốc của ổ đĩa cứng C. Mỗi dòng của tệp chỉ có hai số nguyên, giữa chúng có ít nhất một dấu cách, đây là tử số và mẫu số của một phân số (mọi mẫu số đều khác không). Riêng dòng đầu tiên của tệp lưu một số nguyên dương là số lượng các phân số đang lưu trên tệp. **Kích thước của tệp khá lớn nên không thể**

lưu trữ toàn bộ tệp tại bộ nhớ trong. Viết một chương trình lần lượt làm các việc sau:

a) Lập chương trình con hàm tính ước số chung lớn nhất của hai số nguyên A và B với B luôn khác không. Khai báo hàm là:

```
Function USCLN (A,B: integer): integer;
```

Hàm này phải được dùng trong chương trình.

b) Phân số ở dạng tối giản là phân số có ước số chung lớn nhất của tử số và mẫu số bằng 1. Giả sử trên tệp có không quá 120 phân số ở dạng tối giản. Đọc tệp để tìm ra các phân số ở dạng tối giản, rồi lưu trữ vào mảng A. Mỗi phần tử của A là một bản ghi gồm hai số nguyên là tử số và mẫu số của một phân số.

- Nếu không tìm thấy thì thông báo ra màn hình "KHONG TIM THAY" và kết thúc chương trình.

- Nếu tìm thấy thì sắp xếp lại mảng A theo thứ tự không giảm của tử số. Đưa mảng A sau khi sắp xếp ra màn hình theo quy định: mỗi dòng màn hình có 5 phân số, mỗi phân số có dạng:

[Tử số: Mẫu số] với tử số và mẫu số đều chiếm 6 vị trí.

ĐỀ 3

Bài 1

Cho hàm số sau với đối số là số thực:

$$g(x)=\begin{cases}0 & x\geq -3\\ \dfrac{5x+4x^2}{9+3x} & x<-3\end{cases}$$

1. Hãy sửa lại các lỗi có trong chương trình con hàm sau đây để nó tính đúng giá trị hàm g(x) nêu trên:

```
Function G (x:real): integer;
```

Begin G:= 0; if x < -3 then G:= 5x + 4x*x div 9 + 3x **End**;

2. Viết một chương trình hoàn chỉnh, có sử dụng chương trình con hàm đã sửa, để thực hiện các việc sau:

– Đọc từ bàn phím hai số thực a và b. Tính và đưa ra màn hình giá trị T = g(a) - g(b)

– Đọc từ bàn phím số thực c rồi đưa ra màn hình thông báo cho biết giá trị g(c) nằm trong hay ngoài khoảng [N, L] với

N = min{g(a), g(b)} L = max {g(a), g(b)}

Bài 2

Trong tệp định kiểu có tên INTERNET.DAT được lưu tại thư mục gốc của ổ đĩa cứng C, đang lưu trữ số liệu thống kê số giờ vào mạng Internet của mỗi sinh viên trong từng tháng. Mỗi phần tử của tệp là một bản ghi tương ứng với một sinh viên trong một tháng, gồm ba trường sau:

- Họ tên sinh viên - là xâu không quá 30 ký tự

-Tên lớp - là xâu không quá 12 ký tự đã ở dạng chữ hoa.

- Số giờ vào mạng - là số thực

Kích thước của tệp khá lớn nên không thể lưu trữ toàn bộ tệp tại bộ nhớ trong. Viết một chương trình lần lượt làm các việc sau:

1. Đọc từ bàn phím một xâu ký tự là tên lớp (không quá 12 ký tự), biến đổi thành chữ hoa rồi lưu vào biến LP.

2. Đọc tệp để đưa ra màn hình các thông tin sau:

- Số lượng sinh viên đã vào mạng của lớp đó.

- Thời gian trung bình vào mạng của các sinh viên của lớp đó.

- Thời gian của người vào mạng ít nhất của lớp đó.

3. Ghi họ tên của các sinh viên có thời gian vào mạng ít nhất của lớp đó ra tệp văn bản, mỗi dòng có ba họ tên, chúng cách nhau bởi ba dấu cách. Tên tệp văn bản là INTERNET.TXT và đặt tại mục gốc của ổ đĩa cứng C.

ĐỀ 4

Bài 1

Viết một chương trình thực hiện lần lượt các việc sau:

a) Lập chương trình con hàm tính số hạng thứ N của dãy Fibonacy:

$F_0 = 1, F_1 = 1, F_n = F_{n-1} + F_{n-2}$ với $n \geq 2$

Khai báo hàm là Function Fibo (n: integer): real;

b) Nhập từ bàn phím số nguyên dương K, với K > 1. Sử dụng hàm trên để tính và đưa ra màn hình số hạng lớn nhất của dãy Fibonacy còn nhỏ hơn K.

Bài 2

Viết một chương trình lần lượt làm các việc sau:

a) Lập chương trình con hàm tính giá trị của phân số khi cho tử số và mẫu số. Khai báo hàm là:

```
Function GTPS (tu, mau: real): real;
```

Nếu mẫu số bằng không và tử số lớn hơn không thì hàm nhận số lớn nhất của kiểu số thực. Nếu mẫu số bằng không và tử số nhỏ hơn không thì hàm nhận số nhỏ nhất của kiểu số thực.

b) Nhập từ bàn phím một dãy phân số, mỗi phân số được tổ chức dưới dạng bản ghi gồm có tử số và mẫu số và lưu trữ vào mảng A. Mỗi phần tử của mảng A cũng có cấu trúc bản ghi như vậy và A có không quá 100 phần tử. Dấu hiệu kết thúc việc nhập dữ liệu là một phân số có tử số và mẫu số đều bằng không, phân số này không thuộc dãy.

c) Sắp xếp lại mảng A đó theo quy luật các phân số bé nhất lên đầu dãy, các phân số lớn nhất xuống cuối dãy, các phân số còn lại ở giữa dãy.

d) Ghi mảng A sau khi sắp xếp lên tệp văn bản, tệp này có tên PHANSO.TXT và đặt tại thư mục gốc đĩa C. Mỗi phân số ghi trên một dòng của tệp văn bản và có dạng: Tử số/Mẫu số. Dòng cuối cùng của tệp ghi số lượng phân số đã ghi lên tệp.

ĐỀ 5

Bài 1

Viết một chương trình thực hiện lần lượt các việc sau:

a) Lập chương trình con hàm tính tổng:

$$F = 1 - \frac{x}{2!} + \frac{x^4}{4!} + ... + (-1)^n \frac{x^{2n}}{(2n)!} + ... \text{ cho đến khi } \left|\frac{x^{2n}}{(2n)!}\right| < \varepsilon$$

Khai báo hàm là **Function F (x, epsilon:real;):real;**

b) Nhập từ bàn phím giá trị của x và ε, với $|x| \le 2\pi$ và $0 < \varepsilon < 0,5$

Sử dụng hàm trên để tính và đưa ra màn hình x và F.

Bài 2

Thông tin về các loại thuốc đang có trong kho được lưu trữ tại tệp định kiểu, tệp có tên là THUOC.DAT và đặt tại thư mục C:\TM. Mỗi phần tử của tệp là một bản ghi với các trường

Ten - kiểu xâu có không quá 30 ký tự, lưu trữ tên thuốc

Han - kiểu xâu có 6 ký tự số, lưu trữ hạn sử dụng của thuốc với 2 số đầu là tháng và 4 số cuối là năm.

Luong - kiểu số thực, lưu trữ số thuốc

Gia - kiểu số thực, lưu trữ đơn giá thuốc (giá một đơn vị)

Kích thước của tệp khá lớn nên không thể lưu trữ toàn bộ tệp tại bộ nhớ trong.

Viết một chương trình thực hiện lần lượt các việc sau:

a) Đọc tệp để lấy ra thông tin của các loại thuốc đã hết hạn sử dụng tính đến tháng 12 năm 2006 và lưu trữ vào mảng A. Mỗi phần tử của mảng A cũng có cấu trúc bản ghi như trên. Nếu không có loại thuốc nào hết hạn sử dụng thì thông báo ra màn hình "khong co thuoc het han" và kết thúc chương trình.

b) Nếu có loại thuốc hết hạn sử dụng thì sắp xếp lại mảng A theo thứ tự không tăng của hạn sử dụng. Ghi mảng A ra tệp văn bản, tên tệp này nhập từ bàn phím. Mỗi phần tử của mảng A trên một dòng với quy cách: tên thuốc

chiếm 30 vị trí, hạn sử dụng chiếm 6 vị trí, số lượng và đơn giá đều 10 vị trí trong đó có 3 vị trí phần thập phân. Dòng cuối cùng của tệp ghi tổng giá trị của các thuốc hết hạn $= \sum Luong * Gia$.

ĐỀ 6

Bài 1

Giả sử có khai báo

Var a,b: integer; x:real;

y:array [1..10]of real;

Hãy chỉ ra lệnh nào sai và lỗi sai trong các lệnh sau:

1. a: x+1	2. Write (PT vô nghiệm)
3. Readln (x=10);	4. If a: = 5 Then b:=1
5. If a:=5 Then b:=1	6. y:=a-b
7. Fox x:=1 to 5 Do Write ('Quang Ninh')	8. Read (y[55])
9. While 2<x<6 Do a:=a+1	10. b:=1/3

Bài 2

Viết chương trình đọc vào một dãy gồm n số nguyên bất kỳ (n đọc từ bàn phím).

Thực hiện:

– Đếm số phần tử lẻ của dãy chia hết cho 7 và tính trung bình cộng của các số đó.

– In dãy.

Bài 3

Người ta lưu trữ thông tin về các mặt hàng trong dữ liệu kiểu bản ghi với các thuộc tính sau:

Mã hàng, tên hàng, loại hàng, số lượng, đơn giá, thành tiền và thực hiện các công việc sau:

– Đọc vào n bản ghi (n<=100 đọc từ bàn phím)

– Đếm số các mặt hàng có đơn giá >= 50.000

– In ra màn hình các mặt hàng là loại hàng "Nội" có số lượng >=100.

ĐỀ 7

Câu 1

Trong MS-DOS, tác dụng của lệnh DIR A_BT.* là:

a) Hiển thị các tệp có phần tên là BT của ổ đĩa A.

b) Hiển thị các tệp của ổ đĩa A có phần mở rộng là bất kỳ, phần tên là BT.

c) Các câu trên đều không phải là tác dụng của lệnh.

Câu 2

Trong Turbo Pascal, chọn biểu thức đúng thực hiện việc kiểm tra biến nguyên X có giá trị trong khoảng $5 \leq X \leq 7$

a) (X = 5) and (X = 6) and (X= 7)

b) Not ((X<5) or (X>8))

c) X in [5;6;7]

d) $5 \leq X$ and $X \leq 7$

Câu 3

Hãy chỉ ra giá trị hệ 2 khi lưu trữ giá trị - 17 hệ 10 trong ô nhớ 1 byte:

a) 00010111

b) 11101111

c) 10010001

d) 10010111

e) Các số trên đều không đúng.

Câu 4

Trong Turbo Pascal có lệnh Writeln (F,X) với khai báo F: TEXT. Hãy chọn ra liệt kê chỉ gồm các kiểu dữ liệu được phép của biến X:

a) Real, Integer, Boolean, Char, String

b) Real, Integer, Char, String, Array

c) Real, Integer, Boolean, String, Array, Record

d) Real, Integer, Boolean, Char, String, Array, Record

Câu 5

Trong Turbo Pascal, hãy chỉ ra chương trình con hàm xác định số nguyên N có phải là số chính phương:

a) function cp (n: integer):boolean; var 1: integer;

Begin cp: = false; for i:= 1 to trunc (sqrt (n)) do

if n= i*i then cp:=true else cp:=false; end;

b) function cp (n:integer):boolean

var i:integer;

Begin for i:=trunc (sqrt (N)) downto 1 do

if n=i*i then cp:=true else cp:=false; end

c) function nt (n:integer) : boolean;

Begin cp:= (n>0) and (sqrt (N) = trunc (sqrt (n)); end;

d) Các chương trình con trên đều không là hàm "xác định số nguyên N có phải là số chính phương".

Câu 6

Biến X được khai báo kiểu String. Hãy chỉ ra lệnh gán sai:

a) X:= DELETE ('BACHKKHOA',5,1);

b) X:= 'BACHKHO' + CHR (65);

c) X:= 'BACH' + 'KHOA-K' + '50';

d) X:= COPY('BACHKHOA',5,3)

Câu 7

Trong Turbo Pascal, với khai báo.

Type RHS = reconrd hoten: string [25]; diem: real; end;

Var HS: array [1...25] of RHS;

khi chạy chương trình, HS sẽ chiếm bao nhiêu byte trong bộ nhớ chính:

a) 775

b) 725

c) 800

d) Các số byte đều sai.

Câu 8

Đơn vị nhỏ nhất thường dùng để đo dung lượng bộ nhớ:

a) Kilobyte (KB)

b) Gigabyte (GB)

c) Bit

d) Byte (B)

Câu 9

Cho biết kết quả nhận được khi chạy chương trình sau:

Var x, y: integer;

Proceduce TT (a: integer; var b: integer);

Begin a:= a+2; b:= b+2; end;

Begin x:=3; y:=3; tt (x,y*2); witeln (x:3, y:3); end.

a) Đưa ra màn hình: 3 8

b) Đưa ra màn hình: 3 6

c) Đưa ra màn hình: 5 8

d) Chương trình không chạy vì có lỗi cú pháp

e) Các câu trên đều sai.

Câu 10

Số thứ nhất là 19 ở hệ 16, số thứ hai là 00110110 ở hệ 2. Tổng số của chúng là:

a) 120 hệ 8

b) 80 hệ 10

c) 4F hệ 16

d) Các số trên đều không là tổng số.

ĐỀ 8

Bài 1

Viết chương trình làm các công việc sau:

a) Đọc một số nguyên N, 30>=N>=20; Yêu cầu có kiểm tra các điều kiện của N.

b) Đọc dữ liệu cho mảng P bao gồm N phần tử là số nguyên.

c) Đưa ra màn hình tất cả các phần tử khác nhau trong mảng P và tần xuất (số lần) xuất hiện của chúng trong mảng P, mỗi phần tử chiếm một dòng trên màn hình.

Ví dụ, cho mảng có 10 phần tử [2, 3, 4, 4, 2, 2, 2, 4, 3, 5] thì phải in ra :

2 4 lần

3 2 lần

4 3 lần

5 1 lần

Bài 2

Trong một kỳ thi, kết quả thi của các thí sinh được lưu vào một file định kiểu **KETQUA.DAT** của các bản ghi được khai báo như sau:

```
Type TS = Record
          HoTen: string[50];    (* Họ và tên *)
          SBD:   integer;       (* Số báo danh *)
          Diem:  real ;         (* Điểm thi *)
          End;
```

Giả sử số lượng thí sinh dự thi không vượt quá 100.

Hãy viết chương trình PASCAL thực hiện những công việc sau:

a) Đọc từ tệp KETQUA.DAT thông tin về những thí sinh có điểm thi >=5 rồi lưu vào mảng THIDO – mảng các bản ghi có kiểu TS.

b) Sắp xếp lại mảng THIDO theo chiều giảm dần của điểm thi, các bản ghi có trường điểm thi bằng nhau được sắp xếp theo chiều tăng dần của trường số báo danh.

c) In ra màn hình kết quả thi trong mảng THIDO theo khuôn dạng sau (thông tin về mỗi thí sinh chiếm một dòng màn hình):

Họ tên: 30 ký tự; Số báo danh: 5 ký tự; Điểm thi: 5 ký tự, trong đó có 2 ký tự cho phần thập phân.

d) Tìm trong mảng THIDO và in ra màn hình họ tên và điểm thi của thí sinh theo khuôn dạng: Họ tên: 30 ký tự, Điểm thi : 5 ký tự, trong đó có 2 ký tự cho phần thập phân (số báo danh được nhập từ bàn phím), nếu không tìm thấy thì in ra màn hình thông báo : "Không tồn tại số báo danh như vậy!".

ĐỀ 9

A. PHẦN TRẮC NGHIỆM VÀ TRẢ LỜI NGẮN

Bài 1

Biết rằng biến X có kiểu thực (real). Hãy chỉ ra biểu thức Boolean của PASCAL trong các biểu thức sau:

a) sqr (X * 5 + 2 * X * X) >= 623

b) sqrt (X * X + 9.5) - sin (2 * X + 1)

c) (X <= 5) and (X >= 1)

d) 5 * X * X + 3.1 * (X - 2)

Bài 2

Biết rằng biến X có kiểu nguyên (integer). Hãy chỉ ra biểu thức nguyên của PASCAL trong các biểu thức sau:

a) (X <= 5) and (X >= 1)

b) sqr (X + 1) + 12 * X

c) 2 * X * X + 3 * X - 19

d) Sin (X + 1.5) * 12.9 - 3

Bài 3

Trong chương trình sau, hãy tìm lỗi sai và giải thích lý do:

```
Var   X, Y: real
Begin  X :=185;
  case X * X + 45 of
      3481 : Y := 2 * X + 56
      else Y := X +2
  end;
  writeln (X : 5, Y : 5)
End.
```

Bài 4

Trong chương trình sau, hãy tìm lỗi sai và giải thích lý do:

```
Var X, Y: integer;
Begin  Y := 12
  For X := 1 to Y do ;
  For X := 1 to sqr(x+2) do Y := Y + X;
  Writeln (Y : 6 : 0)
End.
```

B. PHẦN LẬP TRÌNH PASCAL

Bài 1

Đã có sẵn một tệp văn bản tên là **VB.TXT**. Hãy lập một chương trình thực hiện lần lượt các nhiệm vụ sau:

a) Viết khai báo chương trình con.

```
Function DEM (tu, xau : String) : byte;
```

trả về số lần xuất hiện của từ TU trong xâu XAU (từ là một dãy ký tự không chứa dấu cách).

b) Đọc từ bàn phím một từ T.

c) Đọc nội dung của tệp VB.TXT theo từng dòng và dùng hàm DEM vừa khai báo để đếm xem từ T xuất hiện bao nhiêu lần trong tệp đó. Kết quả đưa ra màn hình.

Bài 2

Danh sách mọi người dân thường trú trong một phường đã được cất giữ tại một tệp định kiểu có tên là **DS.DAT**. Mỗi phần tử của tệp này ghi thông tin của một người gồm:họ tên, ngày sinh, địa chỉ. Hãy lập chương trình trong đó khai báo kiểu dữ liệu là:

```
TYPE  dan = record
    ten : string [30];
    sinh : string [8];
    nha : string [32]
     end;
```

để mô tả thông tin của một người, với trường ten để lưu trữ họ tên, trường sinh để lưu trữ ngày sinh, trường nha để lưu trữ địa chỉ. Chương trình lần lượt thực hiện các nhiệm vụ sau:

a) Đọc từ bàn phím một xâu gồm 6 ký tự là tháng tuyển nghĩa vụ quân sự (2 số đầu chỉ tháng, 4 số sau chỉ năm).

b) Đọc tệp nêu trên để đưa ra "Danh sách khám nghĩa vụ quân sự", danh sách này ghi ra một tệp văn bản có tên là **NVQS.TXT**. Danh sách này chứa thông tin của những người dân có tuổi tính đến tháng tuyển nghĩa vụ quân sự là >= 18 tuổi và <= 22 tuổi. Thông tin của mỗi người trên một dòng: từ cột 1 đến cột 30 lưu trữ họ tên, từ cột 32 đến cột 39 lưu trữ ngày sinh, từ cột 41 đến cột 72 lưu trữ địa chỉ. Nếu không có ai trong độ tuổi đó thì đưa thông báo ***Không tìm thấy*** ra màn hình.

ĐỀ 10

Bài 1

Cho hàm số:

$$F(x, y) = Log2(y2+2*x) - x*Cos(2*x+y2)$$

Viết chương trình thực hiện các công việc sau:

a) Xây dựng Function tính F(x, y). Hàm được khai báo như sau:

```
Function F1XY(x: real; y:real): real;
```

b) Nhập một nguyên dương M1<=20 từ bàn phím, có kiểm tra điều kiện của M1. Đưa ra màn hình giá trị của F(x,y) với x chạy từ 1 đến M1 với bước nhảy 0.1 và y=M1.

c) Xác định giá trị lớn nhất của F(x,y) trong câu b. Đưa ra màn hình (chỉ cần đưa 1 giá trị của x) theo dạng:

Giá trị lớn nhất: xxxxxxxx.xx đạt được khi X=xx.x

Bài 2

Để quản lý sổ điện thoại người ta xây dựng tệp định kiểu của các bản ghi được khai báo:

```
DT = record
 HoDem : string [30];    {Họ và đệm}
   Ten : string[15];     {Tên}
 SoDT  : string [15];    {Số điện thoại}
   end;
```

Viết chương trình

a) Tạo một sổ điện thoại trên máy tính vào một tệp định kiểu có tên 'SODT.DAT'. Kết thúc nhập liệu khi nhập '#' vào một trong ba trường Hodem, Ten, SoDT.

b) Đưa ra màn hình danh sách lưu trữ trong sổ điện thoại theo dạng:

STT	Họ và tên	Số điện thoại
(3 vị trí)	(50 vị trí, tên cách họ đệm 5 vị trí)	(15 vị trí)

Tổng số người có trong danh bạ: xxxx

c) Nhập vào một số điện thoại từ bàn phím. Không dùng mảng hãy tìm kiếm trên tệp và đưa ra họ và tên của người có số điện thoại này. Nếu không tìm thấy thì hiển thị thông báo 'Khong tim thay', nếu có nhiều hơn một người như vậy thì phải đưa tất cả.

ĐỀ 11

A. PHẦN TRẮC NGHIỆM VÀ TRẢ LỜI NGẮN

Bài 1

Hãy chỉ ra lỗi trong đoạn chương trình sau:

```
Function (a : integer , b : real ; var c : string[30] );
Begin
    writeln('a= ', a);
    b = a + 5;
    c := 'b=a + 5';
    writeln(c, b);
End.
```

Bài 2

Trong một chương trình viết bằng Turbo Pascal, gọi là chương trình A. Trong A có hai chương trình con B và C. Trong C có hai chương trình con C1 và C2. Hỏi các phát biểu sau đây, phát biểu nào sai:

a) C có thể gọi được B.

b) C1 có thể gọi được B.

c) A có thể gọi được C2.

d) B có thể gọi được C2.

Bài 3

Để lưu trữ số nguyên có dấu trong một byte, người ta dùng bit cao nhất để:

a) Lưu dấu - hoặc +.

b) Lưu giá trị 1 hoặc 0 tương ứng với số âm hay dương.

c) Lưu giá trị 1 hay 0 nhưng giá trị này hoàn toàn tương đương giá trị ở các bit thấp.

d) Không dùng bit này.

Bài 4

Hãy chỉ ra lỗi trong đoạn chương trình sau:

```
var a,b,c: integer;
procedure A(m,n:integer);
Begin
   writeln(m,n);
End;
BEGIN
      a := 3 ; b := 2 ; c:= a/b ;
   A(a,b,c);
   writeln(m,n);
   writeln(a : 5 : 2 ; b : 5 : 2);
END.
```

B. PHẦN LẬP TRÌNH PASCAL

Bài 1

a) Lập chương trình con tính hàm số:

$$Y = F(X) = \begin{cases} 5^{3x} + \sqrt{\left|x + \dfrac{1}{x}\right|} & khi \quad x \neq 0 \\ 1 & khi \quad x = 0 \end{cases}$$

b) Chương trình chính làm các việc sau :

- Tính $S = \frac{\sum_{i=1}^{n} F(x_i)}{n^2}$, biết giá trị x_i tăng từ 1 đến 100 với bước tăng 0,5.

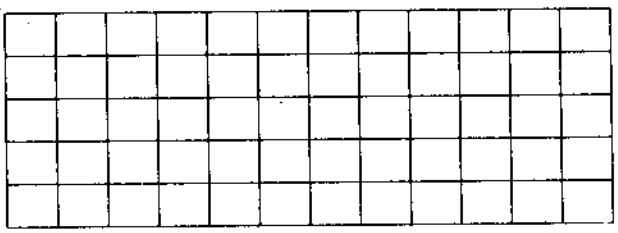
- In ra màn hình giá trị S.

Bài 2

Cho chương trình sau:

```
uses crt;
var a,b,c:integer;
procedure CTC(x:integer;var y,z:integer);
    Begin
    x:=x+3; b:=0; z:=z-3;
    gotoxy(3,2);
    writeln(x:3,y:3,z:3);
    End;
Begin
    clrscr;
    a:=2;b:=5;c:=8;
    CTC(a,b,c);
    gotoxy(3,2);
    writeln(a:3,b:5);
        writeln;writeln(c:3);
        readln;
End.
```

Bạn hãy điền kết quả chạy chương trình vào bảng dưới đây:



Bài 3

Lập chương trình quản lý sách trong thư viện.

a) Nhập các bản ghi về sách bao gồm các thông tin sau : tên sách, tên tác giả, số lượng, năm xuất bản, tiếng. Ghi vào file SACH.DAT. Quá trình nhập liệu kết thúc khi gặp tên sách là '***'.

b) Đọc dữ liệu từ file SACH.DAT. In ra màn hình thông tin sách trong thư viện theo thứ tự ABC của tên sách dưới dạng bảng:

STT	Tên sách	Tên tác giả	Số lượng	Năm xuất bản

c) Tìm tổng số lượng sách của một tác giả nào đó (tên tác giả nhập từ bàn phím).

d) Tìm tất cả các sách xuất bản năm 1999 và đưa vào file 'SACHMOI.DAT'.

MỤC LỤC

Chịu trách nhiệm xuất bản :

Chủ tịch HĐQT kiêm Tổng Giám đốc NGÔ TRẦN ÁI

Phó Tổng Giám đốc kiêm Tổng biên tập NGUYỄN QUÝ THAO

Tổ chức bản thảo và chịu trách nhiệm nội dung :

Chủ tịch HĐQT kiêm Giám đốc Công ty CP Sách ĐH – DN TRẦN NHẬT TÂN

Biên tập nội dung và sửa bản in:

BÙI MINH HIỂN

Trình bày bìa :

BÙI QUANG TUẤN

Chế bản :

ĐAN NGỌC

GIÁO TRÌNH LẬP TRÌNH PASCAL

Mã số : 6E008M7 – DAI

In 1.000 cuốn (QĐ 83), khổ 16 x 24. In tại Nhà in Hà Nam.

Địa chỉ : Số 29, QL 1A, P. Quang Trung, TX. Phủ Lý, Hà Nam.

Số ĐKKH xuất bản : 17 – 2007/CXB/59 – 2217/GD.

In xong và nộp lưu chiểu tháng 10 năm 2007.

CÔNG TY CỔ PHẦN SÁCH ĐẠI HỌC - DẠY NGHỀ
HEVOBCO
25 HÀN THUYÊN – HÀ NỘI
Website : www.hevobco.com.vn

VƯƠNG MIỆN KIM CƯƠNG
CHẤT LƯỢNG QUỐC TẾ

TÌM ĐỌC SÁCH THAM KHẢO MÔN TIN HỌC
CỦA NHÀ XUẤT BẢN GIÁO DỤC

1. Giáo trình Vẽ cơ khí với AutoCAD 2004	*Chu Văn Vượng*
2. Giáo trình Lập trình Pascal (TCCN)	*Thạc Bình Cường - Lê Quốc Trung*
3. Giáo trình Thiết kế Web (TCCN)	*Thạc Bình Cường*
4. Giáo trình Hướng dẫn sử dụng Photoshop (TCCN)	*Nguyễn Phú Quảng*
5. Giáo trình Cad/Cam	*TS. Phan Hữu Phúc*

Bạn đọc có thể mua tại các Công ti Sách - Thiết bị trường học ở các địa phương hoặc các Cửa hàng của Nhà xuất bản Giáo dục :

Tại Hà Nội : 25 Hàn Thuyên; 187B Giảng Võ; 232 Tây Sơn; 23 Tràng Tiền.

Tại Đà Nẵng : Số 15 Nguyễn Chí Thanh; Số 62 Nguyễn Chí Thanh.

Tại Thành phố Hồ Chí Minh : Cửa hàng 451B - 453, Hai Bà Trưng, Quận 3;
240 Trần Bình Trọng – Quận 5.

Tại Thành phố Cần Thơ : Số 5/5, đường 30/4.

Website : www.nxbgd.com.vn

8 934980 763063

Giá : **16.500 đ**