

SỞ GIÁO DỤC VÀ ĐÀO TẠO HÀ NỘI
TRƯỜNG TRUNG CẤP CÔNG NGHỆ HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN

BÀI GIẢNG MÔN LẬP TRÌNH PASCAL

Giáo viên: Đặng Thị Phước

Phuocdt.gdvn@gmail.com

Hà Nội , năm 2009

Bài giảng môn Lập Trình Căn Bản



I. CÁC PHẦN TỬ CƠ BẢN CỦA NGÔN NGỮ PASCAL

1. BỘ CHỮ VIẾT CỦA PASCAL

a. Bộ chữ cái La tinh

Gồm 26 chữ cái tiếng Anh in hoa A-Z và in thường a-z. Ký tự gạch nối _ cần phân biệt với dấu -

b. Bộ chữ số

Gồm các chữ số thập phân: 0, 1, ... , 9. Để tránh lẫn 0 (chữ số không) và O (chữ O) TP quy định gạch chéo trong chữ số không.

c. Những dấu phép toán số học

+ (cộng), - (trừ), * (nhân), / (chia)

d. Các dấu so sánh

= (bằng) , > (lớn hơn) , < (nhỏ hơn),

>= (lớn hơn hoặc bằng), <= (nhỏ hơn hoặc bằng), <> (khác)

e. Những kí hiệu khác:

. ,,: ' "!:! @ # \$ % \ ^ & () [] { }



I. CÁC PHẦN TỬ CƠ BẢN CỦA NGÔN NGỮ PASCAL

2. TÊN – ĐỊNH DANH

◆ Khái niệm Tên - Định danh

- Là tên của các đối tượng khác nhau trong lập trình, dùng để phân biệt giữa đối tượng này với đối tượng khác.
- Các đối tượng thường được đặt tên bằng danh hiệu: biến, hằng, chương trình con,

◆ Qui tắc ngữ pháp của tên:

- Bắt đầu bằng chữ cái (A-Z, a-z) hay dấu gạch dưới (_)
- Theo sau là chữ cái, dấu gạch dưới hay chữ số.
- Với Pascal không phân biệt CHỮ HOA hay chữ thường
- Một số ngôn ngữ khác có phân biệt như Java,...
 - ◆ Ví dụ: X , BienDem, Bien_dem, X1 , X2 , X3 , x1,x2,x3
 - ◆ Ví dụ sai: 101X3, (X1), Bien Dem



I. CÁC PHẦN TỬ CƠ BẢN CỦA NGÔN NGỮ PASCAL

2. TÊN – ĐỊNH DANH

- ◆ **Tên** gồm 2 loại:
 - Tên thuộc ngôn ngữ (Pre-defined)
 - ◆ Do ngôn ngữ quy định trước ý nghĩa của nó.
 - ◆ Được dùng cho các đối tượng có sẵn trong ngôn ngữ
 - Ví dụ: Integer, Readln,sqrt, real,...
 - Tên do người sử dụng đặt ra (user defined)
 - ◆ Do người sử dụng tự qui ước và qui định ý nghĩa của nó trong chương trình nguồn (source code)
 - Ví dụ: abc, xyz1, xyz2, delta, namsinh, tinh_giai_thua
- ◆ **Từ dành riêng (từ khóa):** Là những từ do ngôn ngữ quy định sẵn như là một bộ phận cấu thành ngôn ngữ đó.
 - Ví dụ: begin, if, then, program, array, procedure (trang 24)



I. CÁC PHẦN TỬ CƠ BẢN CỦA NGÔN NGỮ PASCAL

2. TÊN – ĐỊNH DANH

◆ Qui tắc đặt tên

- Tuân thủ quy tắc ngữ pháp của tên
- Không được trùng lắp với tên thuộc ngôn ngữ hoặc đã được định nghĩa.
- Nên sử dụng các tên gợi nhớ

◆ Tên gợi nhớ?

- Tên mà khi đọc đến sẽ giúp ta biết được ý nghĩa của đối tượng mang tên đó.
- Lợi ích của tên gợi nhớ: giúp chương trình dễ đọc, dễ hiểu & dễ kiểm tra.

If ABC < 0 then write('Phuong trinh vo nghiem') ABC không gợi nhớ

If Delta < 0 then write('Phuong trinh vo nghiem') Delta là tên gợi nhớ



I. CÁC PHẦN TỬ CƠ BẢN CỦA NGÔN NGỮ PASCAL

3. Dấu chấm phẩy, lời giải thích

- ◆ **Dấu chấm phẩy** dùng để ngăn cách các lệnh của TP và không thể thiếu trong các câu lệnh.
- ◆ **Lời giải thích** – Các lời giải thích, bình luận có thể đưa vào bất kì chỗ nào trong chương trình để cho chương trình dễ đọc, dễ hiểu mà không làm ảnh hưởng đến các phần khác. Lời giải thích được đặt trong dấu { và } hoặc giữa cụm (* và *)
 - Thí dụ (* day la mot chuong trinh*)
{ day la mot chuong trinh}



II. Cấu trúc chương trình Pascal

◆ Phần Khai báo

Chứa các khai báo tài nguyên sẽ sử dụng

Các khai báo nào không cần thiết có thể bỏ đi.

◆ Phần Thân Chương trình

Nội dung các câu lệnh mô tả công việc sẽ được thực hiện.

Program

- ◆ Uses lời gọi sử dụng các đơn vị chương trình
- ◆ Label
- ◆ Const
- ◆ Type
- ◆ Var
- ◆ *Khai báo chương trình con*
- ◆ Begin
các phát biểu, câu lệnh;
- ◆ End.

Ví dụ



Khai báo trong PASCAL

- ◆ Program tenchuongtrinh; { có thể có hoặc không}
Program Chuongtrinhinhtron;
- ◆ Uses crt, printer; (*khai báo sử dụng các đơn vị chương trình*)
- ◆ Const tênhằng=giátrịhằng;
Const pi=3.14159; tentruong='Dai Hoc Bach Khoa'
- ◆ Type tênkiểudữliệu= mô tả xây dựng kiểu
Type diemso=1..10; chucai='a'..'A'
- ◆ Biến và khai báo biến
 - Biến: là ô nhớ lưu trữ dữ liệu và thay đổi được. Có kiểu dữ liệu tương ứng.
var tên biến: kiểu dữ liệu;
- ◆ Procedure... (*khai báo thủ tục hoặc hàm*)
- ◆ Function ...



Phần thân chương trình

◆ Phần thân chương trình:

Phần này bao giờ cũng nằm gọn giữa hai từ khóa BEGIN và END. Sau từ khóa END là dấu chấm để báo kết thúc chương trình. Phần này bắt buộc phải có đối với một chương trình, nó chứa các lệnh để xử lí các đối tượng, số liệu đã được mô tả ở phần khai báo.



Ví dụ một chương trình

```
PROGRAM Hello;           { Dòng tiêu đề }
USES Crt;                { Lời gọi sử dụng các đơn vị chương trình }
VAR Name : string; { Khai báo biến }
PROCEDURE Input;         { Có thể có nhiều Procedure và Function }

Begin
    ClrScr;          { Lệnh xóa màn hình }
    Write('Hello ! What is your name ?... ');
    Readln(Name);
End;
BEGIN
    Input;
    Writeln ('Welcome to you, ', Name) ;
    Writeln ('Today, we study PASCAL PROGRAMMING ... ');
    Readln;
END.
```

III. Môi trường làm việc của Turbo pascal (TP)

bao gồm những phần việc sau:

- ◆ Trước hết là soạn thảo chương trình. Trong TP, một chương trình là một tệp (file) văn bản được soạn thảo theo đúng các quy định của TP. Có thể dùng một hệ soạn thảo văn bản nào đó để soạn thảo. TP có sẵn chức năng soạn thảo và chúng ta nên khai thác khả năng này.
- ◆ Sau khi chương trình đã soạn thảo xong, ta dùng TP để kiểm tra xem trong chương trình đó có lỗi hay không. Nếu có lỗi thì TP sẽ thông báo vị trí xảy ra sai sót và đưa ra dự đoán nguyên nhân, giúp ta cách thức sửa chữa.
- ◆ Khi không còn các thông báo lỗi, nghĩa là chương trình đã đúng về mặt cú pháp, ta có thể chạy chương trình, nạp dữ liệu và thu nhận kết quả.
- ◆ Như vậy, công việc đầu tiên là phải biết cách viết đúng chương trình trên TP. Để làm được điều đó ta cần tìm hiểu một số khái niệm cơ bản trong TP.



III. Môi trường làm việc của Turbo pascal (TP)

1. Khởi động TURBO PASCAL

- ◆ Để sử dụng TURBO PASCAL ta cần tối thiểu là hai tệp: TURBO.EXE và TURBO.TPL.
- ◆ Khởi động TURBO PASCAL
 - Đối với hệ điều hành DOS, giả sử ta đang ở thư mục có hai tệp nói trên ta gõ TURBO tiếp theo là phím ENTER.
 - Đối với hệ điều hành Windows, nếu trên màn hình Windows chúng ta thấy biểu tượng của TURBO PASCAL thì ta chỉ cần kích đúp chuột vào đó hoặc gõ đầy đủ đường dẫn vào hộp thoại của lệnh RUN – ví dụ c:\TP70\turbo



III. Môi trường làm việc của Turbo pascal (TP)

2. Màn hình soạn thảo trong TURBO PASCAL

File Edit Search Run Compile Debug Option Window Help

-

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

13



III. Môi trường làm việc của Turbo pascal (TP)

3. Soạn thảo trong TURBO PASCAL

- ◆ PASCAL dùng bộ từ vựng tiếng Anh. Turbo Pascal không chỉ cho phép ta làm việc với những chương trình theo ngôn ngữ Pascal mà còn là một hệ soạn thảo khá mạnh. Điều đó tạo thuận lợi cho việc soạn chương trình. Sau đây là một số thao tác soạn thảo thông dụng:



III. Môi trường làm việc của Turbo pascal (TP)

3. Soạn thảo trong TURBO PASCAL

3.1. Dịch chuyển con chạy

- 4 phím mũi tên.

3.2. Sửa chữa văn bản

- Phím Del để xoá một kí tự bên phải con chạy.
- Phím Backspace xoá đi một kí tự bên trái con chạy.
- Phím INSERT để chọn chế độ chèn hoặc đè.
- Ctrl-Y. Xoá cả dòng đang chứa con chạy.
- Ctrl-Q Y. Xoá từ vị trí con chạy đến cuối dòng
- Ctrl- Q A. Tìm kiếm một dãy kí tự và thay thế.



III. Môi trường làm việc của Turbo pascal (TP)

3. Soạn thảo trong TURBO PASCAL

3.3. Làm việc với khối dòng

- Ctrl-K B. Đánh dấu đầu khối.
- Ctrl-K K. Đánh dấu cuối khối.
- Ctrl-K Y. Xoá khối dòng đã đánh dấu.
- Ctrl-K C. Sao chép khối tới vị trí mới của con chạy.
- Ctrl-K V. Chuyển khối tới vị trí mới của con chạy.
- Ctrl – Insert Sao chép khối vào trong bộ nhớ.
- Shift _ Insert Dán khối trong bộ nhớ vào cửa sổ.
- Ctrl-K W. Ghi khối dòng vào một tệp.
- Ctrl-K R. Đọc một tệp từ đĩa vào và xen vào chỗ con chạy.



III. Môi trường làm việc của Turbo pascal (TP)

4. Môi trường của TURBO PASCAL

- ◆ Môi trường trên giúp ta làm việc với TURBO Pascal: Soạn chương trình (Edit), thực hiện chương trình (Run), ghi chương trình vào đĩa, gọi chương trình từ đĩa (File) v.v... Muốn chọn công việc nào, ta dùng một trong các cách sau:
 - ◆ Nhấn phím F10 để vào menu, di vệt sáng đến chức năng cần chọn rồi gõ ENTER.
 - ◆ Nhiều công việc ghi trên menu còn có thể thực hiện bằng cách gõ phím chức năng tương ứng. Ví dụ F3 để mở tệp, Alt-F3 để đóng tệp, F9 để dịch chương trình, Ctrl-F9 để thực hiện chương trình, F2 để ghi tệp lên đĩa với tên đã có, Alt-X để kết thúc làm việc với TURBO PASCAL...



IV. Kiểu dữ liệu (data type)

1. Kiểu dữ liệu là gì?

- Một kiểu dữ liệu là một qui định về hình dạng, cấu trúc, miền giá trị, cách biểu diễn và các phép toán để xử lý một loại dữ liệu thực tế nào đó **trong máy tính**.

Kiểu INTEGER biểu diễn số nguyên từ -32767 đến 32768 và thực hiện được các phép toán cộng, trừ, nhân, chia, div, mod

Kiểu CHAR biểu diễn các ký tự và biểu diễn giữa cặp dấu nháy đơn. ‘A’

Có thể thực hiện phép so sánh, không thể cộng, trừ, nhân, chia

- ◆ Mọi dữ liệu muốn được xử lý bằng máy tính thì phải quy về một kiểu dữ liệu nào đó mà ngôn ngữ lập trình đó hiểu được.



2. Các kiểu dữ liệu

- ◆ Các kiểu dữ liệu đơn giản chuẩn
 - Kiểu liên tục: Real (một số tên ở ngôn ngữ khác float, double)
 - Kiểu rời rạc: Integer, char, boolean, byte, word, liệt kê, miền con.
 - string
- ◆ Các kiểu dữ liệu có cấu trúc/Kiểu do người dùng định nghĩa
 - Array (dãy, mảng)
 - Record (bản ghi, mẫu tin, mục tin)
 - Set (tập hợp)
 - File (tập tin, tệp)
 - String (chuỗi, xâu)
- ◆ Kiểu pointer (con trỏ, chỉ điểm)



3. Kiểu dữ liệu đơn giản chuẩn

3.1 Kiểu số nguyên - Integer: Chiếm 2 byte trong bộ nhớ.
Miền giá trị trong phạm vi từ -32768 đến +32767

Các phép toán số học đối với số nguyên

TÙ KHÓA	SỐ BYTE	PHẠM VI
BYTE	1	0 .. 255
SHORTINT	1	- 128 .. 127
INTEGER	2	- 32768 .. + 32767
WORD	2	0 .. 65535
LONGINT	4	- 2147483648 ... 2147483647

KÝ HIỆU	Ý NGHĨA
+	Cộng
-	Trừ
*	Nhân
/	Chia cho kết quả là số thực
DIV (5 div 2)	Chia lấy phần nguyên
MOD (6 mod 4)	Chia lấy phần dư
SUCC (n)	n + 1
PRED (n)	n - 1
ODD (n)	TRUE nếu n lẻ và FALSE nếu n chẵn



3. Kiểu dữ liệu đơn giản chuẩn

3.2 Kiểu số thực - Real: biểu diễn các số thực dạng dấu phẩy tĩnh hoặc dấu phẩy động. Chiếm 6 byte trong bộ nhớ. Miền giá trị (dương) nhỏ nhất đến ($1.9E-39$) và lớn nhất đến ($1.7E+38$)

Tên	Kích thước	Khoảng biểu diễn	Số chữ số đáng tin
Real	6	$\pm 2,9 \cdot 10_{-39} \dots \pm 1,7 \cdot 10_{38}$	11-12
Single	4	$\pm 1,5 \cdot 10_{-45} \dots \pm 3,4 \cdot 10_{38}$	7-8
Double	8	$\pm 5,0 \cdot 10_{-324} \dots \pm 1,7 \cdot 10_{308}$	15-16
Extended	10	$\pm 3,4 \cdot 10_{-4932} \dots \pm 1,1 \cdot 10_{4932}$	19-20



3. Kiểu dữ liệu đơn giản chuẩn

3.2 Kiểu số thực - Real:

Một số hàm toán học

KÝ HIỆU	Ý NGHĨA
ABS (x)	$ x $: lấy giá trị tuyệt đối của số x
SQR (x)	Lấy bình phương trị số x
SQRT(x)	Lấy căn bậc hai của x
SIN(x)	$\sin (x)$: lấy sin của x
COS (x)	$\cos (x)$: lấy cos của x
ARCTAN (x)	arctang (x)
LN (x)	$\ln x$: lấy logarit nepe của trị x (e (2.71828)
EXP (x)	e^x
TRUNC (x)	lấy phần nguyên lớn nhất không vượt quá trị số x
ROUND (x)	làm tròn giá trị của x, lấy số nguyên gần x nhất



3. Kiểu dữ liệu đơn giản chuẩn

3.3 Kiểu kí tự - Char: biểu diễn cho dữ liệu ký tự. *Chiếm 1 byte trong bộ nhớ. Miền giá trị theo bảng mã ASCII. Biểu diễn bằng ký tự nằm giữa hai dấu nháy đơn 'A', 'a', ...*

KÝ HIỆU	Ý NGHĨA
ORD(x)	Cho số thứ tự của ký tự x trong bảng mã
CHR(n) hay #n	Cho ký tự có số thứ tự là n
PRED(x)	Cho ký tự đứng trước x
SUCC(x)	Cho ký tự đứng sau x



3. Kiểu dữ liệu đơn giản chuẩn

3.4 Kiểu logic - Boolean: biểu diễn cho giá trị luận lý FALSE và TRUE. Qui ước FALSE < TRUE. *Chiếm 1 byte trong bộ nhớ.*

A	B	NOT A	A AND B	A OR B	A XOR B
TRUE	TRUE	FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	FALSE	TRUE	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE

Các phép toán
quan hệ cho kết
quả kiểu Boolean

KÝ HIỆU	Ý NGHĨA
< >	khác nhau
=	bằng nhau
>	lớn hơn
<	nhỏ hơn
> =	lớn hơn hoặc bằng
< =	nhỏ hơn hoặc bằng



3. Kiểu dữ liệu đơn giản chuẩn

3.5 **Kiểu chuỗi - String:** biểu diễn cho chuỗi ký tự. Chiếm $n+1$ byte trong bộ nhớ với n là số ký tự có trong string. Các ký tự có chỉ số từ 1. Vị trí chỉ số 0 chứa một ký tự có giá trị có mã ASCII là số ký tự n của string. Ví dụ chuỗi 'Truong Dai Hoc bach khoa Rat noi tieng' được lưu trong bộ nhớ là

9Truong Dai Hoc bach khoa rat noi tieng

Chuỗi trên có 38 ký tự và chiếm 39 byte với byte 0 chứa ký tự '9'

String rỗng '' chứa không ký tự. String có thể so sánh theo từng ký tự từ trái sang phải đến khi có sự khác biệt. Ví dụ 'ABCDEFG' < 'ABcD'

Phép ghép string + : 'Truong Dai Hoc' + 'Bach Khoa'
=> 'Truong Dai HocBach Khoa'



3. Kiểu dữ liệu đơn giản chuẩn

3.6. Các kiểu dữ liệu tự tạo:

Pascal cho phép lập trình viên dựa trên những kiểu dữ liệu cơ sở tạo ra những kiểu dữ liệu mới. Quá trình đó theo hai chiều hướng:

- 1- thu hẹp khoảng biểu diễn hay thay tên gọi của phần tử ta được *kiểu khoảng con* và *kiểu liệt kê*.
- 2- xây dựng kiểu mới có thành phần là các kiểu đã biết. Ta gọi chúng là *kiểu dữ liệu có cấu trúc* (chúng gồm kiểu mảng (array), kiểu bản ghi (record), kiểu tập hợp(set), kiểu đối tượng (object))... Chúng ta sẽ nghiên cứu chúng trong phần sau.

Khai báo các kiểu dữ liệu tự tạo (custom data type) ta dùng từ khoá type như sau:

Type

```
<tên kiểu>=<mô tả>;
```

Type

```
Chu_so = '0'..'9';
```



V. Khai báo hằng, biến, biểu thức, câu lệnh

1. Hằng (const)

Hằng là một đại lượng có giá trị không đổi trong quá trình chạy chương trình. Ta dùng tên hằng để chương trình được rõ ràng và dễ sửa đổi.

Cách khai báo

CONST

<Tên hằng> = <giá trị của hằng>;

Ví dụ **CONST**

Siso = 100; chuoi = 'xxx';

2. Biến (variable)

Biến là một cấu trúc ghi nhớ dữ liệu vì vậy nó phải tuân theo qui định của kiểu dữ liệu : một biến phải thuộc một kiểu dữ liệu nhất định

Cách khai báo

VAR

<Tên biến> : <Kiểu biến>;

Ví dụ : **VAR**

a : Real ;

b, c : Integer ;

TEN : String [20]

X : Boolean ;



V. Khai báo hằng, biến, biểu thức, câu lệnh

3. Biểu thức (Expression)

Biểu thức là một công thức tính toán để có mtj giá trị theo qui tắc toán học nào đó. Một biểu thức bao gồm toán tử, toán hạng. Các phần tử của biểu thức có thể là số hạng, thừa số, biểu thức đơn giản, hàm...

Ví dụ CONST

$3 + pi * sin(x);$

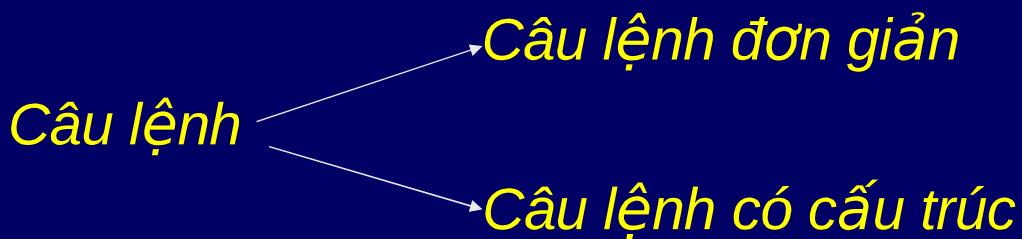
Mức độ thứ tự ưu tiên phép toán

(...) ; NOT, - (cho phép toán có một toán hạng); * , /, DIV, MOD, AND; + , -, OR, XOR; =, <>, <=, >=, <, >, IN.



V. Khai báo hằng, biến, biểu thức, câu lệnh

4. Câu lệnh (*Statement*)



Câu lệnh đơn giản: Là các câu lệnh không chứa các lệnh khác như read, write...

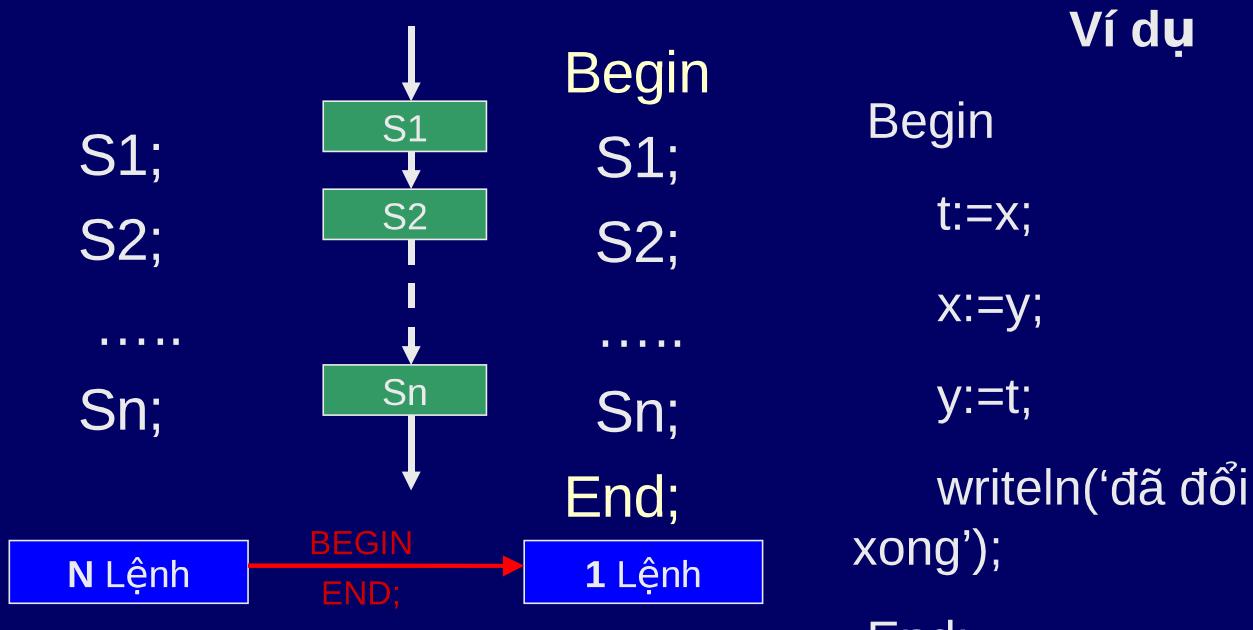
Câu lệnh có cấu trúc: Là các khối lệnh như lệnh thử, rẽ nhánh, lặp....



V. Khai báo hàng, biến, biểu thức, câu lệnh

5. Lệnh hợp thành (Compound Statement)

- ◆ Dùng để ghép nhiều lệnh đơn liên tiếp thành một lệnh.
- ◆ Cú pháp : BEGIN các phát biểu; END;



VI. Lệnh gán và thủ tục xuất nhập

1. Lệnh gán :=

Gán một giá trị của một biểu thức cho một biến

TenBien := Bieuthuc;

2. Thủ tục viết dữ liệu ra màn hình:

- Write(Item1,Item2,...)
- Writeln(Item1,Item2,...)
- Writeln;

Itemi có thể là hằng, biến, biểu thức, hàm, chuỗi kí tự (chuỗi kí tự để trong dấu ‘...’



VI. Lệnh gán và thủ tục xuất nhập

2. Thủ tục viết dữ liệu ra màn hình:

- Viết ra màn hình một chuỗi kí tự
Write('van ban')
Write('van ban':16) *cần phải 16 kí tự*
- Viết ra kiểu số nguyên; với biến A=23123
Write(A); Write(A:8); *cần phải 8 kí tự*
- Viết ra kiểu số thực; VỚI biến A=231.23
Write(A); Write(A:8:3);
Kết quả: 2.312300000E+02 231.230
- Viết dữ liệu ra máy in:
Write(Lst, Item1, Item2,...); Khi sử dụng lệnh này
trong phần khai báo sử dụng lệnh uses printer;



VI. Lệnh gán và thủ tục xuất nhập

2. Thủ tục vào dữ liệu:

- Read(biến1, biến2,...biếnN);
- Readln(biến1, biến2,...biếnN);
- Readln;

Dữ liệu khi gõ vào bàn phím tương ứng với từng biến được phân biệt với nhau bởi dấu cách (ít nhất là 1)

- Trong lập trình người ta thường kết hợp hai lệnh xuất và nhập để đối thoại giữa người và máy.



VI. Lệnh gán và thủ tục xuất nhập

3. Thủ tục trình bày màn hình

- Khi ta khai báo unit CRT với câu lệnh USES CRT; ta sẽ được quyền sử dụng các lệnh sau
- GOTOXY(X, Y); nhảy con trỏ tới vị trí có tọa độ (X, Y)
- ClrScr; Lệnh xóa màn hình.
- ClrEof; Lệnh xóa kí tự phía bên phải con trỏ.
- Textcolor(*con số từ 1 đến 15 hoặc tên màu*); lựa chọn màu cho kí tự.
- Textbackground(*con số từ 1 đến 8 hoặc tên màu*); lựa chọn màu nền.
- LowVideo; làm cho chữ tối hơn.
- NormVideo; làm cho chữ trở lại bình thường.



Các câu lệnh điều khiển

Bài giảng môn Lập Trình Căn Bản



1. Tổng quan

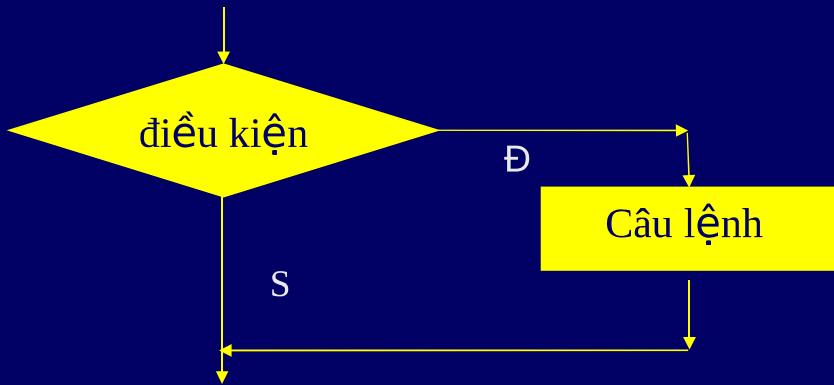
- ◆ **Lệnh điều khiển:** là những dòng lệnh dùng để điều khiển hoạt động của chương trình.
- ◆ **Các lệnh điều khiển cơ bản**
 1. Các lệnh điều khiển (control statements)
 1. Câu lệnh điều kiện **IF**
 2. Câu lệnh điều kiện **CASE**
 3. Câu lệnh lặp **WHILE**
 4. Câu lệnh lặp **REPEAT**
 5. Câu lệnh lặp **FOR**
 6. *Phát biểu GOTO*
 7. Lệnh gọi thủ tục, hàm (**procedure call**): gọi các chương trình con loại procedure, function



2. Câu lệnh điều kiện IF...Then...Else

a. Lệnh rẽ nhánh dạng khuyết

Cú pháp: if <điều kiện> then Câu lệnh;



◆ Ví dụ

```
If Delta > 0 then  
begin  
    X1:= (-b + sqrt(Delta))/2/a  
    X2:= (-b - sqrt(Delta))/2/a  
end;
```

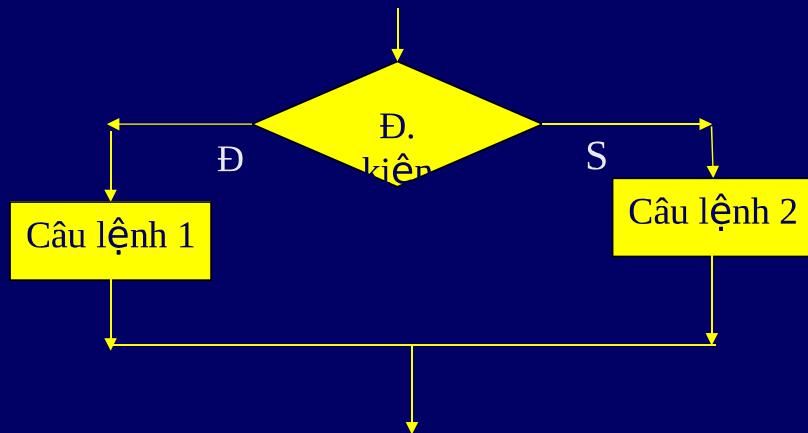
Hoạt động của lệnh IF. Nếu điều kiện đúng thì thực hiện câu lệnh. Nếu sai thì không làm gì



2. Câu lệnh điều kiện IF...Then...Else

b. Lệnh rẽ nhánh dạng đầy đủ

- ◆ Cú pháp if <điều kiện> then câu lệnh1 else câu lệnh2 ;



- ◆ Ví dụ

```
If Delta > 0 then  
begin  
    X1:= (-b + sqrt(Delta))/2/a  
    X2:= (-b - sqrt(Delta))/2/a  
end  
else Writeln('Còn xét tiếp');
```

Hoạt động của lệnh IF dạng này: Nếu điều kiện đúng thì thực hiện Câu lệnh 1 còn (ứng với trường hợp điều kiện sai) thì thực hiện Câu lệnh 2

Học sinh viết chương trình giải phương trình bậc một, hai.

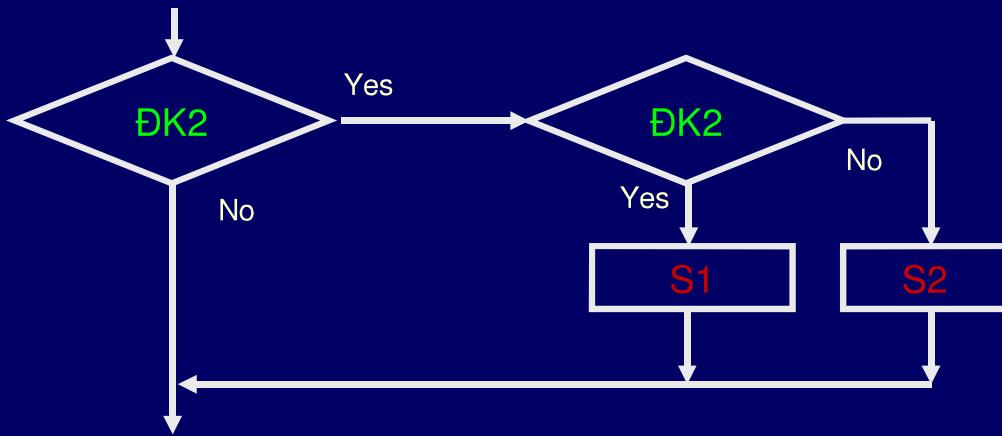


Trường hợp đặc biệt

- ◆ Xét phát biểu sau:
- ◆ **If ĐK1 then if ĐK2 then S1 else S2;**

?
else ? else

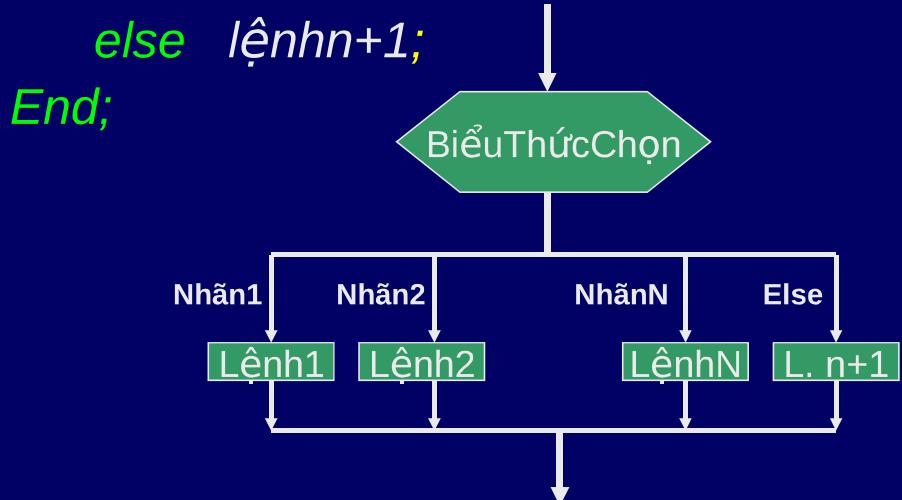
- ◆ ELSE sẽ thuộc về IF nào gần nhất chưa có ELSE



Phát biểu CASE

- ◆ Dùng để chọn một trong số những lệnh để thực hiện tùy theo giá trị của biểu thức chọn.
- ◆ Các nhãn case: chỉ ra các trường hợp phân nhánh.
- ◆ Trong một nhãn có thể có nhiều giá trị phân cách nhau bởi dấu phẩy.
- ◆ ELSE trong phát biểu có thể không có.

Case BiểuThứcChọn of
nhãn1: lệnh1;
nhãn2: lệnh2;
.....
nhãnN: lệnhn;
else lệnhn+1;
End;



Phát biểu CASE

- ◆ Thí dụ cho biết số ngày của một tháng trong năm nhập từ bàn phím:
VAR

 songay, thang, nam: integer

BEGIN

 Write('Thang : '); ReadIn(thang);

 Write('nam : '); ReadIn(nam);

 case thang of

 4, 6, 9, 11: songay:=30;

 2: case nam mod 4 of

 0: songay:=29

 1, 2, 3: songay:=28

 End; {case nam}

 1, 3, 5, 7, 8, 10, 12: songay:=31;

 End;{case thang}

 Writeln('so ngay cua thang ', thang, ' nam ', nam, ' la ', songay);

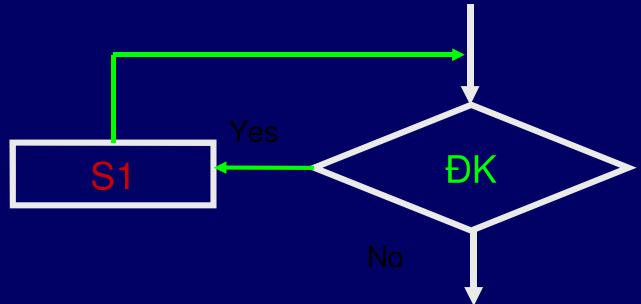
END.



Phát biểu While

- ◆ Dùng để lặp đi lặp lại nhiều lần một công việc nào đó.
- ◆ Cú pháp
While <ĐK> Do câu lệnh
- ◆ While kiểm tra điều kiện trước rồi mới thực hiện phát biểu.
- ◆ Số lần lặp là không biết trước.
- ◆ Số lần lặp tối thiểu là 0 và tối đa là không xác định.
- ◆ Chú ý: Trong thân của while phải có ít nhất một phát biểu có khả năng thay đổi giá trị của điều kiện. Nếu không sẽ lặp vô tận (infinite loop)

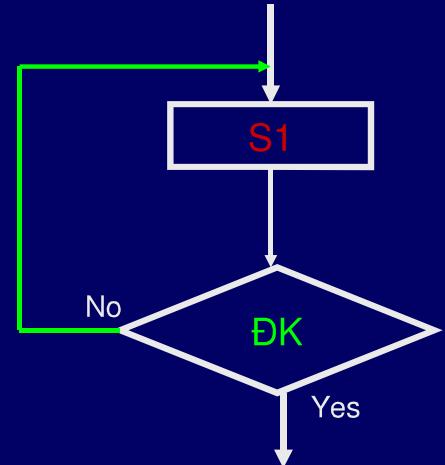
- ◆ Ví dụ:
gt:=1; i:=1
While i<n do
begin
 i:=i+1;
 gt:=gt*i;
end;



Phát biểu Repeat

- ◆ Dùng để lặp đi lặp lại nhiều lần một công việc nào đó.
- ◆ Cú pháp
Repeat câu lệnh until <ĐK>
- ◆ Repeat thực hiện xong các phát biểu rồi mới kiểm tra điều kiện.
- ◆ Số lần lặp là không biết trước.
- ◆ Số lần lặp tối thiểu là 1 và tối đa là không xác định.
- ◆ Chú ý: Trong thân của repeat phải có ít nhất một phát biểu có khả năng thay đổi giá trị của điều kiện. Nếu không sẽ lặp vô tận (infinite loop)

- ◆ Ví dụ:
gt:=1; i:=1
repeat
 i:=i+1;
 gt:=gt*i;
until i>n



While và Repeat

◆ While và Repeat là hai phát biểu có thể chuyển đổi cho nhau.

◆ While <ĐK> do statement;

=> If <ĐK> then repeat statement until NOT(<ĐK>);

◆ Repeat statements until <ĐK>;

=> Begin

statements;

while NOT(<ĐK>) do begin

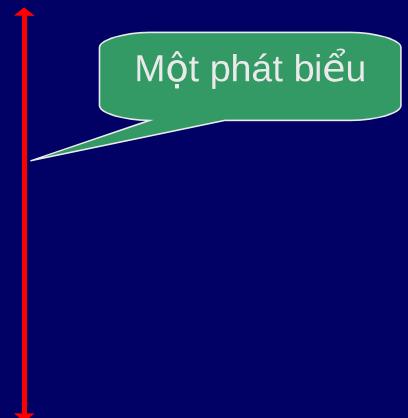
statements;

end;

End;

◆ Chú ý khả năng bị lặp vô tận.

Một phát biểu



Phát biểu FOR

- ◆ Dùng để lặp lại một công việc nào đó với số lần lặp là xác định được.
- ◆ Sử dụng biến đếm và biểu thức cận để xác định số lần lặp lại.
- ◆ For **biendem:=BT1 to BT2 do statement**
 - Số lần lặp là $BT2 - BT1 + 1$;
 - Sau mỗi lần lặp biendem tăng đến giá trị kế tiếp;
- ◆ For **biendem:=BT1 downto BT2 do statement**
 - Số lần lặp là $BT1 - BT2 + 1$;
 - Sau mỗi lần lặp biendem giảm đến giá trị kế tiếp;



Một số chú ý với phát biểu FOR

- ◆ Biến đếm và các biểu thức cận phải thuộc cùng một kiểu rời rạc.
- ◆ Trong thân của FOR, dù cho các thành phần của biểu thức cận thay đổi vẫn không ảnh hưởng đến số lần lặp. Ví dụ đoạn code sau:
a:=5;
For i:=1 to a do a:=a+1;
vẫn chỉ lặp đúng 5 lần dù a bị thay đổi.
- ◆ Giá trị của biến đếm sau vòng lặp FOR là KHÔNG XÁC ĐỊNH. Phụ thuộc vào từng compiler. Do đó không được sử dụng tiếp giá trị này cho các tính toán tiếp theo mà phải gán lại giá trị cụ thể mới.
- ◆ Không được thay đổi giá trị biến đếm trong thân vòng lặp FOR.



Chương 4

Các kiểu dữ liệu phức tạp (do người dùng định nghĩa)

Các kiểu dữ liệu rời rạc

Các kiểu dữ liệu có cấu trúc

Một số giải thuật trên array.

Khái niệm cơ bản về cấu trúc dữ liệu



Kiểu do người dùng định nghĩa

◆ Kiểu do người dùng định nghĩa:

- Được xây dựng từ những kiểu cơ bản.
- Do ngôn ngữ lập trình quy định sẵn cấu trúc và phương thức truy cập
- Người dùng sẽ định nghĩa bằng cách xác định cụ thể các giá trị của các Tham số.

◆ Bao gồm

- Các kiểu rời rạc: liệt kê, miền con.
- Các kiểu có cấu trúc: Array, Record, String



Kiểu miền con



Kiểu liệt kê

- ◆ Kiểu liệt kê được định nghĩa bằng các liệt kê ra các giá trị của kiểu. Các giá trị đó là danh hiệu.
- ◆ Ví dụ: `ngay = (sun, mon, tue, wed, thu, fri, sat);`
Var homqua, homnay, ngaymai : `ngay`;
- ◆ Các phép toán:
 - Có thể gán các biểu thức liệt kê cùng kiểu cho biến tương ứng. Ví dụ: `homqua := mon`;
 - Thực hiện phép so sánh dự trên thứ tự index chính là thứ tự liệt kê bắt đầu từ 0 và tăng dần. Ví dụ `tue < wed`
 - Hàm ORD(), hàm SUCC(), hàm PRED()
 - Không được dùng thao tác xuất nhập với kiểu liệt kê.



Kiểu dữ liệu ARRAY

- ◆ **Định nghĩa:** Array là một dãy gồm nhiều phần tử cùng một kiểu. Hay nói cách khác một dữ liệu kiểu array là một dãy của nhiều dữ liệu thuộc cùng một kiểu.
 - Các phần tử của một dãy phải có cùng kiểu gọi là kiểu cơ sở. Kiểu cơ sở có thể là một kiểu bất kỳ của Pascal.
 - Các phần tử có mối quan hệ về vị trí trong dãy được xác định bằng chỉ số (index). Chính là thứ tự về vị trí của nó trong dãy
- ◆ **Khai báo dãy:** tenkieuday = array[kieuchiso] of kieucoso
 - Số phần tử của dãy chính là số giá trị có thể có của kiểu chỉ số.
 - Index có giá trị từ **giá trị nhỏ nhất** đến **giá trị lớn nhất** của kiểu chỉ số

Ví dụ : daynguyen = array [1..100] of integer;

dayreal = array [char] of real;

Dãy nguyên có 100 phần tử kiểu integer; index từ 1 đến 100.

Dãy dayreal có 256 phần tử kiểu real; index từ ký tự có chr(0) đến chr(255).



Các phép toán trên ARRAY

- ◆ Có thể thực hiện phép gán giá trị của các biến của cùng một kiểu array cho nhau. Ví dụ

var a ,b : daynguyen; ta có thể gán a:= b;

Phát biểu này gán giá trị của từng phần tử trong dãy b sang phần tử tương ứng của dãy a.

- ◆ Có thể truy xuất đến từng phần tử của dãy trực tiếp bằng cách dùng tenbien[index]. Ví dụ: a[5] := b[8]
- ◆ Có thể sử dụng từng phần tử của dãy như là một biến của kiểu dữ liệu cơ sở. Ví dụ
for i:= 1 to 100 do readln(a[i])



Một số đặc tính của kiểu array

- ◆ Số phần tử của kiểu array là cố định ngay từ khi khai báo. Dù là ta dùng bao nhiêu phần tử thì cũng chiếm đúng số bộ nhớ mà dãy đã khai báo.
- ◆ Do đó thông thường phải khai báo dư hơn số thường dùng để tránh bị thiếu => lãng phí bộ nhớ.
- ◆ Các phần tử của dãy được xếp liên tục trong bộ nhớ do đó:
 - Cho phép khả năng truy xuất ngẫu nhiên => nhanh
 - Cần có vùng nhớ trống liên tục đủ lớn khi cấp phát bộ nhớ cho dãy. => khó cấp phát.



Một số giải thuật trên array

- ◆ Khi tổ chức lưu trữ trên array của nhiều phần tử, thao tác thường phải thực hiện là tìm kiếm (search) và sắp xếp (sort) các phần tử trong dãy
- ◆ Việc tìm kiếm (search) dùng để truy vấn thông tin.
- ◆ Việc sắp xếp (sort) dùng để trình bày thông tin và giúp cho thao tác search hiệu quả hơn.
- ◆ Một số giải thuật:
 - Linear search có và chưa sort, Binary search
 - Buble sort, quick sort



Linear search

- ◆ Xem xét từng phần tử xem có phải giá trị cần tìm hay không cho đến khi tìm thấy hoặc hết số phần tử của array thì kết luận có không có.
 - Timthay := 0;
For i:= 1 to n do if a[i]=giatricantim then timthay:= i;
if timthay= 0 then writeln('Không có')
else writeln('Có tại ', timthay);
 - i:=1
while (i<=n)and(a[i]<>giatricantim) do i:= i + 1;
- ◆ Số phần tử tổng quát cần duyệt tối đa là N. Có thể áp dụng cho dây bất kỳ, tuy nhiên nếu dây có thứ tự thì có thể duyệt nhanh hơn “một ít”.



Binary search

- ◆ Áp dụng cho dãy đã có thứ tự. Nguyên tắc chính là dựa vào tính thứ tự của dãy để thực hiện số phép so sánh tối thiểu bằng cách lấy phần tử giữa so sánh với giá trị cần tìm:
 - nếu bằng có nghĩa là tìm thấy tại vị trí đó.
 - nếu không bằng thì phân nửa số phần tử sẽ được loại bỏ không cần xét vì chắc chắn không thể bằng.
- ◆ Giải thuật này cho tốc độ tìm kiếm rất cao. Ví dụ dãy có 1 triệu phần tử chỉ tốn không đến 20 phép so sánh là đã xác định được trong khi với linear search là 1 triệu phép so sánh.



Binary search – giải thuật

Dãy A có n phần tử từ 1..n . Giá trị cần tìm là X.

```
i:=1; j:=n; co:=false;  
While (i<j) and (not (co)) do  
begin  
    m:=(i+j) div 2;  
    If a[m] = X then co:=true  
    else if a[m] > X then j:=m  
    else i:=m;  
end;  
If co then "writeln('Tim co phan tu', X)  
else writeln('Khong co phan tu', X);
```



Bubble Sort

- ◆ Dùng để sắp thứ tự một dãy.
- ◆ Nguyên tắc :
 - Tìm phần tử lớn nhất đặt vào vị trí cuối cùng A[n] bằng cách so sánh lần lượt các cặp từ $a[j]$, $a[j+1]$ với j từ 1=> n-1. Nếu phần tử $a[i] >a[j]$ thì hoán đổi 2 phần tử này
 - Lặp lâi bước trên với số phần tử cũ dãy còn lại giảm dần từ n -> 2 .
 - Khi hoàn tất dãy sẽ có thứ tự tăng dần.
- ◆ Ưu điểm của giải thuật là đơn giản. Tuy nhiên tốc độ sắp thứ tự không cao.
- ◆ Có một giải thuật khác khá mạnh là QuickSort



Bubble Sort – giải thuật

Giải thuật bubble Sort cho dãy có n phần tử và tăng dần.

$A[j] > A[j+1]$ then

```
Begin  
    t:= a[i];  
    a[j]:=a[j+1];  
    a[j+1]:=t;  
End;
```

So sánh và hoán đổi giá trị 2 phần tử.
Biến t có dùng kiểu dữ liệu với kiểu cơ sở của array



Array nhiều chiều

- ◆ Kiểu cơ sở của array có thể là một array khác, hình thành nên cấu trúc array of array. Trong Pascal hỗ trợ sẵn kiến trúc này với kiểu dữ liệu array nhiều chiều (multi-dimension array).

- ◆ Khai báo

tenkieu= array [index1, index2,..., indexN] of kieucoso;

Ví dụ:

table = array [1..100, 'a'..'z'] of integer;

- ◆ Từng phần tử của array nhiều chiều có thể được truy cập trực tiếp bằng cách chỉ rõ index trên từng chiều.

Ví dụ :

- ◆ Các phần tử này cũng có thể được sử dụng như là một biến kiểu cơ sở.



Kiểu RECORD

- ◆ Record là một kiểu dữ liệu gồm nhiều thành phần, các thành phần có thể thuộc về những kiểu dữ liệu khác nhau.

- ◆ Khai báo

tênkieu = Record

 tênfield:kieudulieu_cua_field;

.....

 tênfield:kieudulieu_cua_field;

end;



Kiểu Record (tt)

◆ Các phép toán

- Có thể gán giá trị các biến record thuộc cùng một kiểu cho nhau. Khi đó sẽ gán từng field tương ứng.
- Có thể truy cập đến từng thành phần (field) của record bằng cách dùng cấu trúc `biênrecord.tênfield`.
- Mỗi field có thể dùng như một biến thuộc kiểu dữ liệu tương ứng.

◆ Record của record

Trong cấu trúc field của record có thể là một record khác.

Khi đó field record tương ứng là một record để có thể truy cập đến field bên trong dạng



Phát biểu WITH

- ◆ WITH là một phát biểu dùng với kiểu dữ liệu record
- ◆ Phát biểu WITH có dạng
 - WITH recordname do Statement;trong đó record name là một biến record, Statement là một phát biểu.
- ◆ Ý nghĩa phát biểu WITH. Trong phần thân của phát biểu WITH, khi muốn truy cập đến các field của record tương ứng ta chỉ cần dùng tên filed mà không cần dùng Tênrecord.tênfield như thông thường.



Kiểu tập hợp

- ◆ **Định nghĩa:** Dữ liệu kiểu tập hợp là một tập hợp của nhiều dữ liệu thuộc cùng một kiểu rời rạc.
- ◆ **Khai báo:**
 - Tênkieu = set of kiểu cơ sở
 - Ví dụ: tapnguyen = setof integer;
- ◆ Một dữ liệu kiểu tập hợp là một tập hợp được biểu diễn dạng [phantu, phantu,...]
- ◆ Có thể gán tập hợp này cho tập hợp kia nếu chúng có cùng kiểu cơ sở.



Các phép toán trên tập hợp

◆ Với các biến kiểu tập hợp ta có các phép toán

- Phép toán = : cho giá trị TRUE nếu hai tập hợp bằng nhau
- Phép toán <>: cho giá trị TRUE nếu hai tập hợp khác nhau
- Phép toán \leq : $A \leq B$ là TRUE nếu A bao hàm trong B
- Phép toán \geq : $A \geq B$ là TRUE nếu A chứa B
- Phép toán IN : X IN A cho giá trị TRUE nếu trong A có phần tử X
- Phép hợp + : $A+B$ là hợp của hai tập A, B
- Phép giao $+^*$: $A*B$ là giao của hai tập A, B
- Phép hiệu - : $A-B$ là hiệu của hai tập A,B



Kiểu tệp (FILE)

1. Tệp (File) định kiểu.

a. Cách khai báo

<tên biến tệp>: FILE OF <kiểu>;

- ví dụ: F: File of Integer;

b. Cách tạo lập.

ASSIGN (<tên biến tệp>, <'tên tệp'>);

REWRiTE (<tên biến tệp>);

.....

WRiTE (<tên biến tệp>, <biến cùng kiểu để ghi vào tệp>);

.....



Ví dụ kiểu tệp

Tạo lập tệp SN.DAT bằng chương trình sau:

```
Uses crt;  
Var F: file of integer; n, l, x : integer;  
BEGIN clrscr;  
Write(' bạn tao tệp gom bao nhieu so'); readln(n);  
ASSIGN (F, 'SN.DAT'); REWRiTE (F);  
For l :=1 to n do  
    Begin  
        Write('nhap so thu ',i,'de ghi vao tep'); readln(x)  
        WRiTEx (F,x)  
    End;  
CLOSE (<tên biến tệp>);
```



Kiểu tệp (FILE)

1. Tệp (File) định kiểu.

c. Cách đọc dữ liệu từ một tệp.

ASSIGN (<tên biến tệp>, <'tên tệp'>);

RESET (<tên biến tệp>);

.....
READ (<tên biến tệp>, <biến cùng kiểu dữ liệu>);

.....
CLOSE (<tên biến tệp>);



Kiểu tệp (FILE)

1. Tệp (File) định kiểu.

* Một số hàm liên quan đến tệp

EOF (<tên biến tệp>); cho giá trị TRUE nếu con trỏ tệp ở cuối tệp.

FILESIZE (<tên biến tệp>); Cho giá trị là số phần tử của tệp.

SEEK (<tên biến tệp>, N); Di chuyển con trỏ tệp đến phần tử thứ N của tệp (phần tử ban đầu tính từ 0).



Kiểu tệp (FILE)

2. File văn bản

a. Cách khai báo

<tên biến tệp>: TEXT;

- ví dụ: F: Text;

b. Cách tạo lập.

ASSIGN (<tên biến tệp>, <'tên tệp'>);

REWRiTE (<tên biến tệp>);

.....

WRiTE / WRITELN (<tên biến tệp>, <biến cùng
kiểu ghi vào tệp>);

.....

CLOSE (<tên biến tệp>);



Kiểu tệp (FILE)

2. File văn bản.

c. Cách đọc dữ liệu từ một tệp.

ASSIGN (<tên biến tệp>, <'tên tệp'>);

RESET (<tên biến tệp>);

.....
READ (<tên biến tệp>, biến kiểu CHAR);

.....
CLOSE (<tên biến tệp>);



Kiểu tệp (FILE)

2. File văn bản.

* Một số hàm liên quan đến tệp

SEEKEOF (<tên biến tệp>); cho giá trị TRUE nếu con trỏ tệp ở cuối tệp.

EOLN (<tên biến tệp>); cho giá trị TRUE nếu con trỏ tệp ở cuối dòng.

SEEK (<tên biến tệp>, N); Di chuyển con trỏ tệp đến phần tử thứ N của tệp (phần tử ban đầu tính từ 0).

Ghi tiếp vào file văn bản đã có lệnh APPEND(F); với lệnh này file F sẽ được mở lại sau khi ghi xong ta vẫn phải Close (F) để đóng tệp.



Bài tập

- ◆ Các bài tập trong sách giáo trình
- ◆ Bài tập nhân 2 ma trận.
- ◆ Bài tập xác định ma trận đối xứng
- ◆ Bài tập quản lý điểm sinh viên dùng array và record
- ◆ Các bài tập khác bằng chương trình Pascal



Chương 5

Chương trình con

Chương trình con

Phân loại và khai báo

Tham số: phân loại và ý nghĩa

Biến cục bộ và toàn cục

Tầm vực chương trình con – biến

Đệ quy



Chương trình con

- ◆ Khái niệm: Chương trình con là một đoạn chương trình có tên và được gọi thực hiện ở nhiều nơi trong chương trình chính.
- ◆ Tại sao phải dùng chương trình con:
 - Có công việc cần phải được thực hiện tại nhiều nơi trong chương trình => tách công việc đó thành chương trình con
 - Phân đoạn, module chương trình để thuận tiện trong quản lý, trình bày và phát triển.
- ◆ Các lợi ích của việc sử dụng chương trình con
- ◆ Các loại chương trình con: Procedure & Function



Phương thức thực hiện của chương trình con

- ◆ **Tham số:**
hiện chương trình con, thông thường là những dữ liệu cụ thể cần cho tháo tác sử lý của từng trường hợp gọi chương trình con
- ◆ **Danh sách Tham số**
- ◆ **Phương thức dịch và chuyển điều khiển khi gọi chương trình con**
- ◆ **Một số điểm chú ý trong việc sử dụng chương trình con**
- ◆ **Khai báo chương trình con trong chương trình chính của PASCAL.**



Chương trình con Procedure

Procedure TenChuongTrinhCon(danhsachthongso);

Cont

Type

Var

Khai báo chương trình con

Begin

 Phần thân chương trình con

End;



Chương trình con Function

```
Function TenChuongTrinhCon(danhsachthongso):KieuDuLieuCuaTriTraVe;  
Cont  
Type  
Var  
Khai báo chương trình con  
Begin  
Phần thân chương trình con  
TenChuongTrinhCon:=GiaTriTraVe;**  
End;
```

**



Tham số

◆ Tham số hình thức:

trong danh sách Tham số. Khi chương trình con được gọi thực hiện thì các Tham số này sẽ được truyền những giá trị cụ thể cho chương trình con thực hiện.

◆ Tham số thực:

truyền cho các Tham số hình thức khi chương trình con được gọi là các Tham số thực.

◆ Tham số hình thức có 2 loại:

- Tham số hình thức trị
- Tham số hình thức biến

◆ Tham số thực hợp lệ cho các Tham số hình thức phụ thuộc vào loại của Tham số hình thức



Tham số hình thức trị

- ◆ **Định nghĩa:** Những Tham số hình thức không đi sau từ khoá var trong khai báo danh sách Tham số là thôgn số hình thức trị
- ◆ **Ví dụ:** procedure ABC (A: integer, var B: real, C:string);
Tham số hình thức trị là A và C
- ◆ Khi truyền Tham số, Tham số thực sẽ truyền TRỊ của mình cho Tham số hình thức trị.
- ◆ Mọi sự thay đổi của Tham số hình thức trị trong chương trình con KHÔNG ảnh hưởng gì đến trị của Tham số thực truyền cho nó.
- ◆ Tham số thực cho Tham số hình thức trị là một biểu thức cùng kiểu.



Tham số hình thức biến

- ◆ **Định nghĩa:** Những Tham số hình thức đi sau từ khoá var trong khai báo danh sách Tham số là Tham số hình thức biến.
Ví dụ: procedure ABC (A: integer, var B: real, C:string);
Tham số hình thức trị là A và C
- ◆ Khi truyền Tham số, Tham số thực sẽ truyền địa chỉ của mình cho Tham số hình thức trị.
- ◆ Mọi sự thay đổi của Tham số hình thức trị trong chương trình con SẼ ảnh hưởng trực tiếp và tức thời lên chính ô nhớ của Tham số thực, tức là ảnh hưởng ngay đến chính Tham số thực tương ứng.
- ◆ Tham số thực cho Tham số hình thức trị phải là một biến cùng kiểu.
- ◆ Tham số hình thức biến còn được dùng để trả về các giá trị cần thiết cho chương trình gọi sau khi chương trình con kết thúc.



Cấu trúc khối trong chương trình Pascal

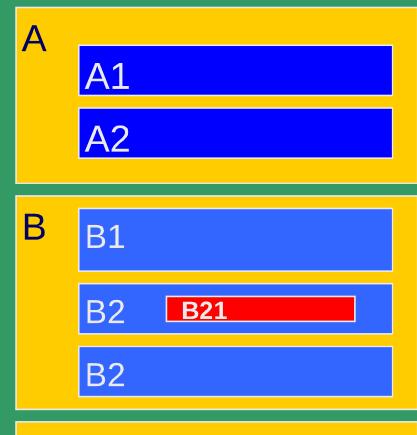
- ◆ Định nghĩa Khối: Một khối (block) gồm 2 phần:

- Phần khai báo với các khai báo: const, type, var, chương trình con.
- Phần thân: bắt đầu bằng BEGIN, ở giữa là các phát biểu và kết thúc bằng END

- ◆ Như vậy:

- Một chương trình là một Block
- Một chương trình con là một Block
- Trong chương trình có chương trình con và trong chương trình con có chương trình con khác -> trong block có block
- Một chương trình là một Block với các Block con lồng vào nhau.

ChuongTrinhChinh



Vấn đề tầm vực

- ◆ **Định nghĩa :** Tầm vực (Scope) của một đối tượng trong chương trình là vùng mà nó được biết đến và có thể được sử dụng.
- ◆ **Tầm vực áp dụng** trên các đối tượng như: biến, hằng, kiểu dữ liệu, chương trình con.
- ◆ **Qui tắc xác định tầm vực:** Tầm vực của một đối tượng được xác định từ vị trí mà nó được khai báo cho đến hết Block chứa khai báo đó, kể cả những Block bên trong của nó. Ngoại trừ trường hợp có sự khai báo lại trong một khối con.
- ◆ **Khai báo lại** Nếu khối A chứa khối B và trong cả 2 khối đều khai báo một đối tượng tên X thì Khối B chỉ có thể truy xuất đối tượng X của chính nó và không thể truy xuất đối tượng X của khối A.

