Bài 1:

Các hàm và thủ tục trong Pascal

<u>Yêu cầu: Dùng các phím F1, Alt F1, Ctrl + F1 xem cấu trúc và các thông số đầu vào của từng lệnh trong Pascal và ghi vào</u> dòng phía dưới của các câu lệnh.

Ví dụ:

- Window Tạo một cửa sổ màn hình mới

☞ ***Procedure Window (X1,Y1,X2,Y2:Byte);***

Bạn có thể Copy các ví dụ của Pascal và thử chạy nó để hiểu rõ hơn về cách sử dụng của các hàm và các thủ tục...

I. Một số Unit hay dùng trong Pascal for DOS

Khi sử dụng một Unit nào đó bạn cần phải khai báo tên Unit sau từ khoá uses. Các Unit có thể đã được Pascal tạo trước đó hoặc do chính bạn định nghĩa.

1. Unit System:

Không phải khai báo sau từ khoá uses, nó là thư viện chứa các hằng, biếnn thủ tục, hàm để xuất, nhập, xử lý dữ liệu, cấp phát, quản lý bộ nhớ.

Các thủ tục và hàm trong Unit system:

Các thủ tục (Procedure):

- Append Mở một tập tin văn bản để ghi thêm dữ liệu.
- Assign Gán tên một tập tin ngoại trú vào biến tập tin.
- Close Đóng biến tập tin.
- Exit Thoát chương trình hoặc vòng lặp.
- FillChar Điền một số byte có giá trị vào một biến.
- Halt Kết thúc ngang chương trình.
- Inc Tăng giá trị một biến.
- Randomize Khởi tạo chế độ tạo số ngẫu nhiên.
- Reset Mở một tập tin đã có để đọc
- ReWrite Tạo mới và mở tập tin.
- Read Gán giá trị cho một hay nhiều biến
- Readln Gán giá trị cho một hay nhiều biến
- Str Đổi giá trị kiểu số sang dạng chuỗi ký tự.
- Val Đổi giá trị kiểu chuỗi sang dạng số.
- Write Xuất dữ liệu ra tập tin, màn hình.
- Writeln Xuất dữ liệu ra tập tin, màn hình.

Các hàm (Function):

- ABS Lấy giá trị tuyệt đốI
- Arctang Lấy giá trị Arctang của một giá trị lượng giác.
- Cos Lấy giá trị lượng giác Cosin của một góc.
- Chr Lấy ký tự trong bảng mã ASCII
- Concat Nối các chuỗi.
- Eof Kiểm tra trạng thái kết thúc của tập tin.
- Eoln Trạng thái kết thúc dòng của tập tin văn bản.
- Exp Lấy luỹ thừa cơ số e của một số.
- Length Lấy chiều dài của một chuỗi
- Ord Lấy thứ tự của một ký tự trong bản mã ASCII.
- Paramcount Tổng số tham số truyền vào của chương trình.
- ParamStr Nội dung của tham số.
- Pi Giá trị 3,1416...
- Pos Vị trí của một chuỗi con trong chuỗi lớn.
- Random Tạo một số ngẫu nhiên.
- Round Làm tròn một số.
- Sin Cho giá trị lượng giác Sinus của một góc.
- Sqr Bình phương của một số
- Sqrt Căn bậc hai của một số.
- Upcase Đổi ký tự thường thành ký tự in.

2. Unit Crt:

Là thư viện chứa các hằng, biến, thủ tục, hàm dùng để xuất dữ liệu dạng văn bản trên màn hình, các thủ tục có liên quan đến âm thanh. Unit Crt gồm có:
Các hằng liên quan đến mầu:

Blank	0	Blue	1	Green	2
Syan	3	Red	4	Magenta	5
Brow	6	LightGray	7	DarkGray	8
LightRed	12	LightMagent	13	Yellow	14
White	15	Blink	128		

Các biến:

Winmax: Word Giá trị đỉnh dưới phải của cửa sổ.
Winmin: Word Giá trị đỉnh trái trên của cửa sổ.

Các hàm và thủ tục:

- Clrscr: Xoá và tô mầu cho cửa sổ hiện tại
- Delay: Dừng chương trình trong một thời gian
- Gotoxy: Đưa con trỏ tới toạ độ màn hình.
- NoSound: Tắt âm thanh
- Sound: Phát âm thanh với một tần số.
- Textbackground: Định mầu nền.
- Textcolor Định mầu chữ
- Window Tạo một cửa sổ màn hình mới
- Keypressed Kiểm tra nếu có một phím được nhấn
- ReadKey Lấy giá trị của một phí m đ ược bấm
- WhereX Vị trí cột của con trỏ
- WhereY Vị trí hàng của con trỏ.

3. Unit Dos:

Là thư viện chứa các hằng, biến, thủ tục dùng để truy xuất đến các tập tin trên đĩa, truy xuất bộ nhớ máy tính... Unit DOS gồm có:

- GetDate Lấy ngày của hệ thống máy tính - GetTime Lấy giờ của hệ thống máy tính
- SetDate Đặt lại ngày cho máy tính - SetTime Đặt lại giờ cho máy tính
- DiskFree Cho số byte còn trống trên đĩa - DiskSize Cho dung lượng của đĩa.

4. Unit Graph:

Là thư viện chứa các hằng, biến, thủ tục, hàm dùng để truy xuất dữ liệu ra màn hình trong chế độ đồ hoạ. Unit Graph gồm có:

- CloseGraph Đóng chế độ đồ hoạ
- DetectGraph Xác định Card và Mode đồ hoạ
- Initgraph Khởi tạo chế độ đồ hoạ
- GraphResult Kết quả khởi động đồ hoạ
- ClearDevice Xoá màn hình đồ hoạ
- SetViewport Định cửa sổ trên màn hình
- ClearViewport Xoá cửa sổ tạo bởi Viewport
- GetMaxX Lấy giá trị lớn nhất của tạo độ X trong chế độ đồ hoạ
- GetMaxY Lấy giá trị lớn nhất của tạo độ Y trong chế độ đồ hoạ
- GetX Lấy giá trị hiện hành của tạo độ X trong chế độ đồ hoạ
- GetY Lấy giá trị hiện hành của tạo độ Y trong chế độ đồ hoạ
- Moveto Di chuyển co trỏ tới một toạ độ mới
- MoveRel Di chuyển con trỏ ngầm định đến một khoảng cách.
- SetBkColor Đặt mầu nền trong chế độ đồ hoạ
- SetColor Đặt mầu vẽ
- GetBkColor Lấy mầu nền trong chế độ đồ hoạ
- GetColor Lấy mầu vẽ
- PutPixel Vẽ một điểm ảnh tại toạ độ

- GetPixel Lấy mầu của điểm ảnh tại toạ độ

**** Các kiểu biến, hằng, hàm, thủ tục liên quan đến cách viết chữ trong đồ hoạ:***

+ Hằng liên quan đến Font chữ:

```
Const
    DefaultFont = 0;      TriplexFont = 1;      SmallFont = 2;
    SansSerifFont = 3;    GothichFont = 4;
```

+ Hằng liên quan đến hướng chữ:

```
Const
    HorizDir = 0        VertDir = 1
```

+ Hằng liên quan đến độ lớn chữ:

```
    UserCharSize = 0;
```

+ Hằng liên quan đến cân chỉnh chữ:

```
    CenterText = 1; LeftText = 0; RightText = 2;
    BottomText = 0; TopText = 2;
```

+ Các thủ tục liên quan:

- GetTextSettings Lấy cách định chữ
- OutText Viết một chuỗi tại toạ độ
- OutTextXY Viết một chuỗi tại toạ độ
- SetTextJustify Đặt cách căn chuỗi
- SetTextStyle Đặt kiểu chữ
- TextHight Lấy độ cao của chuỗi
- TextWidth Lấy độ rộng của chuỗi

**** Các kiểu biến, hằng, thủ tục liên quan đến vẽ đường:***

+ Hằng liên quan đến loại đường

```
Const
    Solidln = 0;    Dottedln = 1;    CenterLn = 2;   DashedLn = 3
```

+ Các thủ tục liên quan:

- GetLineSettings Lấy thông số thiết kế đường
- SetLineStyle Đặt kiểu đường
- SetWritemode Định Mode vẽ đường
- Line Vẽ một đoạn thẳng
- LineRel Vẽ đoạn từ vị trí con trỏ cách một đoạn
- LineTo Vẽ đoạn thẳng từ vị trí con trỏ đến vị trí mới
- DrawPoly Vẽ hình đa giác
- Circle Vẽ đường tròn
- Arc Vẽ cung tròn
- Ellipse Vẽ cung Ellipse
- Rectangle Vẽ hình chữ nhật

**** Các hằng, biến, thủ tục liên quan đến vẽ hình:***

- SetFillStyle Đặt kiểu tô
- SetFillPattern Đặt mẫu tô
- FloodFill Tô một vùng
- Bar Vẽ hình chữ nhật có tô
- Bar3D Vẽ hình khối chữ nhật 3 chiều

- FillPoly Vẽ đa giác có tô
- PieSlice Vẽ hình quạt Ellipse
- Sector Vẽ hình rẻ quạt tròn
- FillEllipce Vẽ hình Ellipce

**** Các thủ tục đặc biệt:***

+ Const:

NormalPut = 0; CopyPut =0
XORPut = 1; ORPut = 2;
ANDPut = 3 NOTPut = 4;

+ Thủ tục:

- GetImage Lưu trữ một vùng màn hình
- PutImage Vẽ lại vùng màn hình đã lưu trữ
- SetActivePage Đặt tran vẽ hoạt động
- SetVisualPage Dặt trang hiển thị
- ImageSize Lấy độ lớn vùng màn hình
- Sector Vẽ hình rẻ quạt tròn
- FillEllipce Vẽ hình Ellipce

**** Các thủ tục đặc biệt:***

+ Const:

NormalPut = 0; CopyPut =0
XORPut = 1; ORPut = 2;
ANDPut = 3 NOTPut = 4;

+ Thủ tục:

- GetImage Lưu trữ một vùng màn hình
- PutImage Vẽ lại vùng màn hình đã lưu trữ
- SetActivePage Đặt tran vẽ hoạt động
- SetVisualPage Dặt trang hiển thị
- ImageSize Lấy độ lớn vùng màn hình
- Sector Vẽ hình rẻ quạt tròn
- FillEllipce Vẽ hình Ellipce

II. Cấu trúc một chương trình viết bằng Pascal

Một chương trình máy tính là một dãy lệnh nhằm chỉ thị, hướng dẫn máy tính thực hiện một thao tác nào đó, thực hiện các phép tính trên các dữ liệu. Cấu trúc chung nhất của Pascal gồm 3 phần:

1. Phần tiêu đề
2. Phần khai báo
3. Phần thân chương trình

Chúng được quy định bằng cú pháp sau:

Program ...; ***Đặt tên cho chương trình***
Uses ...; ***Khai báo các Unit sử dụng trong chương trình***
Const ...; ***Khai báo các hằng sử dụng trong chương trình***
Type ...; ***Định nghĩa các kiểu biến***
Var ...; ***Khai báo các biến sử dụng trong chương trình***
Procedure ...; ***Các chương trình con sử dụng trong chương trình***
Function ...; ***Các hàm sử dụng trong chương trình***
Begin
Statement; ***Các thủ tục và lệnh***
End.

1. Phần tiêu đề của chương trình:

Bắt đầu bằng từ khoá PROGRAM tiếp đó là tên của chương trình do bạn đặt ra (Tên chương trình không có ký tự trống).

VD: **Program** Chao_Cac_Ban;

2. Phần khai báo:

Mô tả các kiểu dữ liệu, các biến, các hằng, các chương trình con ...

USES **Dùng khai báo các Unit (nếu có), các Unit cách nhau bởi dấu phẩy (,), cuối khai báo là dấu chấm phẩy (;).**

VD: **Uses** Crt,Graph;

Const **Từ khoá để khai báo hằng số**

VD: **Const** Max=40;

TYPEDùng khai báo các kiểu dữ liệu do bạn định nghĩa

Var **Dùng khai báo các biến của chương trình**

VD: **Var**
M,N:integer;
ST:String;

Functiong **Để khai báo và triển khai hàm tự tạo do bạn tạo ra.**

Procedure **Khai báo triển khai chương trình con do bạn tạo ra.**

3. Phần thân chương trình:

Gồm hai từ khoá BEGIN và END bao các lệnh của chương trình. Sau từ khoá END là dấu chấm (.) báo hiệu chấm dứt chương trình. Các câu lệnh trong phần thân chương trình sẽ được thực hiện tuần tự, lệnh nào nằm trước sẽ được thực hiện trước, lênh nào nằm sau sẽ được thực hiện sau.

Bài 2

Khai báo biến các kiểu dữ liệu chuẩn

1. Khai báo biến:

Bất kỳ một biến nào khi sử dụng trong chương trình đều phải khai báo, việc khai báo biến gồm hai phần:

- Khai báo tên biến (tên biến do bạn đặt).
- Khai báo kiểu dữ liệu, là tên các kiểu dữ liệu chuẩn, dữ liệu không chuẩn.

Phần tên và phần dữ liệu cách nhau bởi dấu hai chấm (:). Các biến khai báo được bắt đầu bằng từ khoá Var , các biến cùng kiểu cách nhau bằng dấu phẩy (,), các biến khác kiêu cách nhau bằng dấu chấm phẩy (;).

Var
<Biến 1>, <Biến 2> : <Kiểu 1>;
<Biến 3> : <Kiểu 2>;

2. Khai báo Hằng và hằng biến:

Hằng cũng giống như biến nhưng trong nội dung của hằng sẽ không thay đổi trong quá trình thực hiện chương trình. Việc khai báo hằng bắt đầu bằng từ khoá Const

Const
<Hằng> = <Giá trị>;

Hằng được gán thẳng một giá trị mà không cần khai báo kiểu.

VD:
N =20
Str = String;

```
Thoat=True;
```

3. Dữ liệu kiểu số:

a. Các phép toán trên dữ liệu kiểu số nguyên:

Trong Pascal định nghĩa kiểu số nguyên chuẩn như sau:

Kiểu (type)	Phạm vi (Range)	Độ lớn
ShortInt	-128 ... 128	1 Byte
Integer	-32768 ... 32767	2 Byte
LongInt	-2147483648 ... 2147483647	4 Byte
Byte	0 ... 255	1 Byte
Word	0 ... 65535	2 Byte

* Phép toán số học:

Chức năng	Ký hiệu
Phép cộng	+
Phép trừ	-
Phép nhân	*
Phép chia nguyên	Div
Phép lấy phần dư	Mod

Các phép toán này tác động nên kiểu dữ liệu số nguyên cho dữ liệu kiểu nguyên.

* Hàm, thủ tục trên dữ liệu số nguyên:

- PRED(x) Cho phần tử đứng trước x
- SUCC(x) Cho ra phần tử đứng sau x
- INC(x,r) Tăng giá trị của x nên r đơn vị. Tương đương với x:=x+r.
- DEC(x,r) Giảm giá trị của x xuống r đơn vị. Tương đương với x:=x-r.

(Thủ tục INC(x), DEC(x) xem như tăng, giảm x một đơn vị).

* Phép so sánh Logic:

Các phép so sánh như:

Chức năng	Ký hiệu
Bằng nhau	=
Khác nhau	<>
Nhỏ hơn	<
Lớn hơn	>
Lớn hơn và bằng	>=
Nhỏ hơn và bằng	<=

tác động lên kiểu dữ liệu nguyên cho dữ liệu kiểu Boolean có giá trị là đúng (True) hoặc sai (False).

Ví dụ:

```
Var     a,b:Integer;
        Kiemtra:Boolean;
Begin
        a:=10;
        b:=20;
        Kiemtra:=a<b;
...
End.
```

Kết quả của phép gán Kiemtra:=a<b sẽ cho kết quả là True dữ liệu kiểu Boolean và gán cho biến Kiemtra.

b. Kiểu số thực:

Trong Pascal định nghĩa kiểu số thực như sau:

Kiểu (type)	Phạm vi (Range)	Độ lớn
Real	$2.9e^{-39}$ ... $1.7e^{38}$	6 Byte
Single	$1.5e^{-47}$ ... $3.4e^{38}$	4 Byte
Double	$5e^{-324}$ ... $1.7e^{380}$	8 Byte
Extended	$3.4e^{-4932}$ ... $1.1e^{4932}$	10 Byte
Comp	$-9.2e^{18}$ ... $9.2e^{18}$	8 Byte

Chú ý: Muốn dùng kiểu Single, Double, Extended, Comp phải đổi từ SoftWare sang 8087/80287 trong menu Option/Compiler và máy tính phải trang bị mạch coprocessor 80287.

* Các phép tính trên dữ liệu kiểu số thực:

+ Phép tính toán học:

Chức năng	Ký hiệu
Phép cộng	+
Phép trừ	-
Phép nhân	*
Phép chia	/

Các phép tính này tác động nên dữ liệu kiểu số thực và số nguyên cho dữ liệu kiểu số thực. Trong một biểu thức tính nếu các số hạng kiểu số nguyên sẽ tự động biến đổi kiểu thành số thực để các toán tử trên thực hiện phép tính.

+ Phép so sánh: Sử dụng giống hệt các phép so sánh của số nguyên.

+ Các trên kiểu số nguyên và thực:

Hàm	Kiểu tác động	Cho ra kiểu	Chức năng
Round(x)	Real	LongInt	Làm tròn x
Trunc(x)	Real	LongInt	Lấy phần nguyên
Int(x)	Real	Real	Lấy phần nguyên
Frac(x)	Real	Real	Lấy phần lẻ
ABS(x)	Real/Nguyên	Real/Nguyên	Lấy giá trị tuyệt đối
Arctang	Real/Nguyên	Real	Cho arctang của x
Cos(x)	Real/Nguyên	Real	Cho Cosin của x (tính theo radian)
Sin(x)	Real/Nguyên	Real	Cho Sin của x (tính theo radian)
Exp(x)	Real/Nguyên	Real	Hàm số mũ của x theo cơ số e
Pi		Real	Cho hằng 3,1416....
Sqr(x)	Real/Nguyên	Real	Bình phương của x
Sqrt	Real/Nguyên	Real	Căn bậc 2 của x
Ln(x)	Real/Nguyên	Real	Logarit Neper của x

4. Kiểu dữ liệu Boolean:

Đây là một kiểu mà phạm vi chỉ có hai giá trị là True và False

Kiểu (type)	Phạm vi (Range)	Độ lớn
Boolean	True/False	1Byte

* Gán một giá trị kiểu boolean bạn có thể gán một giá trị hay kết quả của một biểu thức logic cho một biến kiểu Boolean bằng lệnh gán:

Ví dụ:

```
Var     Ketqua,thoat:Boolean;
        a,b:integer;
Begin
        a:=10;
        b:=20;
        Thoat:=False;{gán giá trị False cho biến thoat}
        Ketqua:=a<b; {Biểu thức so sánh a<b cho giá trị True vì thế Ketqua có giá trị là True}
....
```

* Các phép Logic trên dữ liệu kiểu Boolean:

Các phép logic như And, Or, Xor, Not tác động lên dữ liệu kiểu Boolean cho kết quả như sau:

A	B	A And B	A or B	Not A	A Xor B

True	True	True	True	False	False
True	False	False	True	False	True
False	True	False	True	True	True
False	False	False	False	True	False

Các dữ liệu kiểu Boolean thường được dùng trong các câu lệnh có điều kiện hoặc các câu lệnh lặp

5. Dữ liệu kiểu ký tự:

a. Kiểu Char:

Là một kiểu biến chiếm 1 byte bộ nhớ mà phạm vi là 256 ký tự trong bảng mã ACSII.

b. Các hàm và thủ tục tác động lên dữ liệu kiểu Char:

- ord(ch:Char):Byte Cho số thứ tự trong bảng mã ACSII của ký tự ch
- Chr(N:byte):Char Cho ký tự thứ N trong bảng mã ACSII. Có thể dùng ký kiệu # để mô tả ký tự N (VD: Chr(65) hoặc #65 đều cho ký tự A).
- Pred(ch:char):char Cho ký tự đứng trước ký tự Ch
- Succ(ch:char):char Cho ký tự đứng sau ký tự ch

Ví dụ: Cho Ch và St là dữ liệu kiểu Char

```
Ch:='C';
St:=Prec(Ch);          Kết quả St=□B□
St:=Succ(Ch);          Kết quả St=□D□
- Upcase(Ch:Char):Char Đổi chữ thường thành chữ in
Ví dụ:  ch:=a;
        st:=Upcase(ch); Cho kết quả A
```

Chú ý: Muốn dùng kiểu Single, Double, Extended, Comp phải đổi từ SoftWare sang 8087/80287 trong menu Option/Compiler và máy tính phải trang bị mạch coprocessor 80287.

* Các phép tính trên dữ liệu kiểu số thực:

+ Phép tính toán học:

Chức năng	Ký hiệu
Phép cộng	+
Phép trừ	-
Phép nhân	*
Phép chia	/

Các phép tính này tác động nên dữ liệu kiểu số thực và số nguyên cho dữ liệu kiểu số thực. Trong một biểu thức tính nếu các số hạng kiểu số nguyên sẽ tự động biến đổi kiểu thành số thực để các toán tử trên thực hiện phép tính.

+ Phép so sánh: Sử dụng giống hệt các phép so sánh của số nguyên.
+ Các trên kiểu số nguyên và thực:

Hàm	Kiểu tác động	Cho ra kiểu	Chức năng
Round(x)	Real	LongInt	Làm tròn x
Trunc(x)	Real	LongInt	Lấy phần nguyên
Int(x)	Real	Real	Lấy phần nguyên
Frac(x)	Real	Real	Lấy phần lẻ
ABS(x)	Real/Nguyên	Real/Nguyên	Lấy giá trị tuyệt đối
Arctang	Real/Nguyên	Real	Cho arctang của x
Cos(x)	Real/Nguyên	Real	Cho Cosin của x (tính theo radian)
Sin(x)	Real/Nguyên	Real	Cho Sin của x (tính theo radian)

Exp(x)	Real/Nguyên	Real	Hàm số mũ của x theo cơ số e
Pi		Real	Cho hằng 3,1416....
Sqr(x)	Real/Nguyên	Real	Bình phương của x
Sqrt	Real/Nguyên	Real	Căn bậc 2 của x
Ln(x)	Real/Nguyên	Real	Logarit Neper của x

6. Cấu trúc câu lệnh:

Chương trình của Pascal là tập hợp các câu lệnh tuần tự tuân theo cú pháp của Pascal. Đó là các chỉ thị cho máy tính thực hiện, cá chỉ thị này được máy tính thi hành tuần tự câu lệnh ghi trước sẽ được thực hiện trước, câu ghi sau sẽ được thực hiện sau.

Một câu lệnh trong chương trình Pascal có thể là:

- assignment(:=)	Phép gán
- Procedure call	Lời gọi một chương trình con
- Begin ... End	Khối câu lệnh
- if ... then ... else	Câu lệnh có điều kiện
- Case ... of ... else ... end	Câu lệnh rẽ nhánh
- Repeat ... Until	Lệnh lặp Repeat
- While ... do	Lệnh lặp While
- For ... to/downto ... do	Lệnh lặp For
- With ... do	Câu lênh With

a. Câu lệnh đơn, câu lệnh ghép:

Về phương diện cú pháp chương trình của Pascal được xem như là một tập hợp liên tiếp các lệnh đơn. Các câu lệnh này chỉ thị cho máy tính thực hiện một thao tác nào đó. Nếu như một thao tác không thể mô tả qua một câu lệnh đơn mà cần phải kết hợp nhiều câu lệnh đơn ghép lại thì các câu lệnh ghép này được đưa về một câu lệnh đơn bằng cách đặt tập hợp các lệnh đơn đó trong cặp từ khoá Begin ... End.

b. Câu lệnh gán:

Dùng để gán giá trị hay kết quả của một biểu thức hay nội dung của một biến cho một biến.

<Biến> := <Giá trị>;
<Biến> := <Biến>;
<Biến> := <Biểu thức>;

c. Lời gọi một thủ tục:

Các thủ tục (Procedure) là các chương trình con. Các chương trình con này do Turbo tạo ra hay do người lập trình tạo ra với mục đích chỉ thị máy tính thực hiện một công việc nào đó. Các thủ tục chuẩn được Turbo đặt trong các Unit, các thủ tục do người lập trình tạo ra được đặt trong chương trình hoặc trong các Unit do người dùng định nghĩa. Các thủ tục này có một tên. Trong chương trình khi cần đến chức năng nào tương ứng với thủ tục thì tên thủ tục đặt trong chương trình coi nhiư là một lệnh. Tuỳ theo yêu cầu của thủ tục mà lện đó có tham đối hay không. Các tham đối được đặt trong dấu ().

Ví dụ: Writeln là một thủ tục trong Unit System của Turbo Pascal làm chức năng xuất các tham đối ra màn hình. Trong chương trình cần chức năng này ta gọi thủ tục:

Writeln(□Tong so là: □,S);

□Tong so là: □ và S là các tham đối của lệnh Writeln như vậy Writeln(□Tong so là: □,S); là một câu lệnh đơn, đó là lời gọi một thủ tục.

Các thủ tục do người lập trình tạo ra phải khai báo nội dung của thủ tục đó trong chương trình hay khai báo tên. Nếu các thủ tục đó được đặt trong Unit do người lập trình là bạn tạo ra thì bạn phải khai báo như sau: Unit Ctr, MyUnit;

d. Câu lệnh điều kiện:

Câu lệnh điều kiện là câu lệnh làm cho chương trình:

- Chọn lựa thực hiện một trong hai câu lệnh
- Thực hiện hay không một câu lệnh tuỳ thuộc vào điều kiện bạn đưa ra có thoả mãn hay không.

* Cấu trúc:

+ Lệnh điều kiện thiếu:

```
if <điều kiện> then <lệnh>;          Sử dụng khi chỉ thực hiện một lệnh đơn
if <điều kiện> then
Begin
 <Lệnh1>;
<Lệnh 2>;
......
End;                                 Sử dụng khi thực hiện nhiều lệnh đơn.
```

+ Lệnh điều kiện đủ:

```
if <điều kiện> then
 <lệnh1>
Else
<lệnh 2>;                            Sử dụng khi chỉ thực hiện một lệnh đơn

if <điều kiện> then
Begin
 <Lệnh1>;
 <Lệnh 2>;
......
End
Else
Begin
 <Lệnh1>;
<Lệnh 2>;
......
End;                                 Sử dụng khi thực hiện nhiều lệnh đơn.
```

Ghi chú: Trước Else không có dấu chấm phẩy (;).

Trong đó:

* <Điều kiện> có thể:

- Một biến kiểu Boolean:

Ví dụ: Nhập hai số, tìm số nhỏ nhất.

Thuật toán để giải có thể như sau:

+ Nhập số thứ nhất; Nhập số thứ hai.
+ So sánh số thứ nhất và số thứ hai, nếu số thứ nhất <= số thứ hai thì so sánh là đúng nếu không thì sai. Đoạn chương trình có thể như sau:

```
Var a,b:Integer;
     Nho:Boolean;
Begin
Write('Nhap a:'); Readln(a);
```

```
Write('Nhap b:'); Readln(b);
Nho:=a<=b;
If Nho then
  Writeln('So nho nhat laf a')
Else
  Writeln('So nho nhat la b');
Readln;
End.
```

- Kết hợp các biểu thức logic thông qua các phép logic And, or, Xor, Not.

Ghi chú: Nếu trong điều kiện có từ hai câu lệnh đơn trở nên bạn cần phải đặt chúng trong khối Begin...End; khi đó có thể coi các câu lệnh đơn trong khối Begin ... End; là một câu lệnh đơn (có thể gọi là nhóm các câu lệnh thành một câu lệnh).

e. Câu lệnh rẽ nhánh.

Câu lệnh rẽ nhánh là câu lệnh làm chương trình thực hiện việc chọn lựa một trong nhiều hành động để thực hiện.

Câu trúc:

```
Case <Biến, biểu thức> of
<Giá trị 1>:<Lệnh 1>;
<Giá trị 2>:<Lệnh 2>;
....
Else
<Lệnh N>;
End;
```

Trong đó:

- Biến, biểu thức là một biến hay biểu thứccho ra dữ liệu rời rạc, đếm được như kiểu Nguyên(Integer), Char, kiểu liệt kê. Kiểu Real không dùng làm biến điều khiển rẽ nhánh.
- Giá trị phải cùng kiểu với biến, các giá trị trong câu lệnh không được trùng nhau, các giá trị trong một trường hợp cách nhau bởi dấu phẩy (,).

Ví dụ: Var Ch:char;

```
Begin
Write('Nhan mot phim:'); Readln(ch);
Case Ch of
'A'..'Z','a'..'z':Writeln('ban nhap mot ky tu');
'0'..'9': Writeln('ban nhap mot so');
else
Writeln('Mot ky tu dac biet');
Readln;
End.
```

f. Câu lệnh lặp For:

Trong chương trình lệnh lặp For làm chương trình thực hiện lặp lại một số lần nhất định.

Cấu trúc:

```
For <Biến >:=<Giá trị đầu> to <giá trị cuối> do <lệnh>;
For <Biến >:=<Giá trị đầu> Downto <giá trị cuối> do <lệnh>;
```

Câu lệnh For chỉ thị máy tính thực hiện <lệnh> một số lần nhất định từ giá trị đầu đến khi vượt qua giá trị cuối . Sau mỗi lần thực hiện <lệnh> thì <Biến> sẽ được tăng hoặc giảm một đơn vị.

Ví dụ: Liệt kê các ký tự trong bảng mã ACSII

```
Var i:byte;
Begin
```

```
    For i:=0 to 255 do Write(i:6,Chr(i):2);
Readln;
End.
```

g. Vòng lặp Repeat

Câu lênh lặp Repeat ... Until là lệnh yêu cầu chương trình lặp lại nhiều lần một hành động cho đến khi thoả mãn một điều kiện để thoát ra khỏi vòng lặp.

Cấu trúc:

```
    Repeat
    <Lệnh 1>;
    <Lệnh 2>;
    ....
    Until <Điều kiện thoát>;
```

Lệnh Repeat ... until lặp lạ các lệnh cho tới khi <Điều kiện thoát> cho kết quả True.

Ví dụ: Tính giai thừa

```
Var  N,i:Integer;
     GT:LongInt;
Begin
Writeln(Tinh giai thua cua mot so nguyenÒ);
Write(INhap N= voi N<17);Readln(N);
GT:=1; i:=1;
Repeat
     GT:=GT*i;
     i:=i+1;
Until i>N;
Writeln(IN =Ò,GT);
Readln;
End.
```

Toàn bộ cấu trúc repeat ... Until là một câu lệnh đơn do đó bạn có thể sử dụng Repeat là một câu lệnh đơn của câu lệnh khác. Hoặc bạn có thể sử dụng nhiều câu kệnh Repeat ... Until lồng nhau.

g. Lệnh lặp While ... do

Cấu trúc:

```
    While <điều kiện> do <lệnh>;
```

Câu lệnh While lặp lại <lệnh> cho tới khi <điều kiện> nhận giá trị sai (False). Bản thân <điều kiện> là một bước kiểm tra trước khi vào vòng lặp While. Nếu thoả mãn <điều kiện> thì <lệnh> được thực hiện nếu không thoả mãn sẽ bỏ qua vòng lặp này. Khác với vòng lặp Repeat ... Until là các lệnh trong nó ít nhất cũng được thực hiện một lần.

* Vòng lặp vô tận và cách thoát khỏi nó:

Vòng lặp While <điều kiện> do <lệnh> sẽ lặp lại <lệnh> vô tận khi <điều kiên> luông bằng True.

Cấu trúc: While True do <lệnh>;

Vòng lặp Repeat ... Until <điều kiện> lặp lại vô tận khi <điều kiện> luông sai (False).

Cấu trúc: Repeat ... Until False;

Trong chương trình khi muốn thoát ra khỏi vòng lặp vô tận bạn có thể kết hợp với câu lệnh điều kiẹen để kiểm tra khi nào thì cho phép thoát ra khỏi vòng lặp vô tận. Thường sử dụng

```
    if <điều kiện> then exit; hoặc if <điều kiện> then Halt;
```

Bài 3

Một số cách sử dụng về thuật toán

<u>Yêu cầu: Phân tích từng dòng lệnh trong các thuật toán dưới đây:</u>

I. Sử dụng vòng For:

1. Dừng màn hình khi số dòng quá 25:

```
Var i:byte
Begin
      for i:=0 to 200 do
      if (i+1) mod 24 =  0 then
      Begin
           Writeln(ỊNhan Enter de tiep tuc ...Ò);
           Readln;
      end
      else
           writeln(ỊDong: Ị,i);
Readln;
end.
```

2. Tính trung bình của 10 số nhập từ bàn phím:

```
Const     Solap=10;
Var       Ketqua,so:real;
          i:Byte;
Begin
      Ketqua:=0;
      For i:=1 to Solap do
           Begin
                Write(ỊCho so thu nhatÒ,i:2);Readln(So);
                Ketqua:=Ketqua+So;
           End;
      Ketqua:=Ketqua/Solap;
      Writeln(ỊTri trung binh cua 10 so vua nhapÒ,Ketqua:6:2);
      Readln;
End.
```

3. Vẽ hình chữ nhật bằng dấu Ị*Ò:

```
Var   d,r:byte;
      i,j:byte;
Begin
      Write(ỊChieu rong cua HCN Ò);readln(r);
      Write(ỊChieu cao cua HCN Ị);readln(d);
      For i:=1 to d do
           Begin
                For j:=1 to r do write(Ị*Ò);
           writeln;
           End;
Readln;
End.
```

4. Các đoạn chương trình sau làm gì:

a. Sử dụng cách viết quy cách:

```
var i,j:byte;
Begin
      for i:=1 to 8 do
      Begin
           for j:=1 to i do write(i:i);
      Writeln;
      Writeln;
      End;
End.
```

b. Có bao nhiêu dấu □*□ hiện ra trong 2 đoạn chương trình sau:

Đoạn 1:

```
For i:=1 to 3 do
      for j:=1 to 3 do
           for k:=1 to 3 do writeln('*');
```

Đoạn 2:

```
For i:=1 to 3 do
      For j:=3 to 6 do
           for k:+i to j do Writeln('*');
```

? Thử viết chương trình hiện bảng cửu chương lên màn hình

? Tạo dãy Fibonaci biết số thứ nhất và thứ 2 của dãy bằng 1 vào từ số thứ 3 trở đi được tính bằng tổng của hai số trước nó.

II. Sử dụng Repeat ... Until:

1. Lặp lại một bài toán:

```
Var   r:real;
      Traloi:Char:
Begin
Repeat
      Write('Cho ban kinh : ');Readln(r);
      Writeln('Dien tich cua hinh tron: ',Pi * r * r);
      Writeln;
Write('ban co lam tiep khong (C/K), Nhan K de thoi ');
Until Upcase(Traloi)='K';
End.
```

2. Tạo Menu chọn đơn giản:

```
Var   Chon:Byte;
      Thoat:Boolean;
      a,b,c:real;
Begin
Repeat
Writeln('1. Phep cong');
Writeln('2. Phep tru');
Writeln('3. Phep nhan');
Writeln('4. Phep chia');
Writeln('5. Ket thuc');
Write('Cho hai so a,b :');Readln(a,b);
Writeln;
Thoat:=False;
Repeat
Write('Ban thuc hien phep tinh nao voi a,b ? ');Readln(chon);
Until (Chon>=1) and (chon<=5);
```

```
Case Chon of
      1: C:=a+b;
      2: C:=a-b;
      3: C:=a*b;
      4: C:=a/b;
      5: Thoat:=True;
End;
Writeln('Ket qua tinh la: ',C:6:2);
Readln;
Until  Thoat;
End.
```

III. Sử dụng While … do:

1. Giả sử bài toán giai thừa như sau:

P=1*2*3*...*N.

Dùng cú pháp lệnh lặp While ... do nào đúng:

a.

```
i:= 1;
P:=1;
While i<=N do;
P:=P*i;
i:=i+1;
Writeln(P);
```

b.

```
i:=1;
P:=1;
While i<=N do
    P:=P*i;
    i:=i+1;
Writeln(P);
```

c.

```
i:=1;
P:=1;
While i<=N do
    Begin
        P:=P*i;
        i:=i+1;
    End;
Writeln(P);
```

2. Khi nào thì câu □Chào bạn□ hiện ra trong đoạn chương trình sau:

```
Repeat
readln(ch);
Until Upcase(ch)='*';
Writeln(□Chào bạn□);
```

Viết lại đoạn trên sử dụng while ... do.

IV. Cách sử dụng một số hàm và thủ tục:

1. Hàm Chr(ch:Byte):Char

Dùng lấy ký tự thứ Ch trong bảng mã ACSII

```
For ch:=0 to 255 do
Begin
      if (ch mod 10) =0 then writeln
      else write(ch,chr(ch):4);
end;
```

2. Tạo âm thanh:

Một nốt nhạc trong âm nhạc có cao độ và trường độ nhất định Tần số của các nốt nhạc như sau:

Đô	Đô#	Rê	Rê#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
512	542	575	609	645	683	624	767	813	861	912	967

Đây là các nốt nhạc trong quãng tám trung bình. Nếu trong bản nhạc có các nốt cao hơn hoặc thấp hơn bạn hãy gấp đôi hoặc giảm một nửa tần số của nốt nhạc đó ở quãng trung bình.

Sound(Tầnsố:Word); Tạo ra một âm thanh có tần số □Tầnsố□

Delay(Ms:Word); Dừng chương trình một thời gian tính bằng mini giây.

NoSound; Tắt âm thanh được tạo ra từ Sound;

3. Cửa sổ màn hình và mầu:

- Window(x1,y1,x2,y2:Integer); Tạo cửa sổ riêng để làm việc
  x1,y1: Toạ độ đỉnh trái trên của màn hình
  x2,y2: Toạ độ góc phải dưới của màn hình

Thường được kết hợp như sau:

```
Window(x1,y1,x2,y3);
TextbackGround(Maunen); Tạo mầu nền
Clrscr; Tô mầu nền cho cửa sổ.
```

- GotoXY(X,Y:Integer); Di chuyển con trỏ tới toạ độ X,Y của màn hình.
- Keypressed:Boolean; Kiểm tra xem trên màn hình có một phím nào bấm hay không

Thường được dùng trong câu lệnh: if, Repeat, While:

```
VD: Repeat
        Các câu lệnh;
    Until Keypressed; Lặp lại tới khi bạn nhấn một phím bất kỳ.
```

- Readkey:Char; Nhận giá trị của một phím ký tự được nhấn nhưng không hiện ra màn hình.

Nếu so sánh Readkey và readln với biến Ch:Char ta thấy:

+ Readln(Ch); Màn hình dừng lại và chờ bạn nhấn một phím, ký tự nhập vào được hiện lên màn hình. Sau khi nhấn Enter kết quả sẽ gán vào biến Ch;
+ Ch:=Readkey; Màn hình sẽ dừng và chờ bạn nhấn một phím, ký tự nhập không hiện lên màn hình và được gán ngay vào biến Ch không cần bạn phải nhấn Enter;

4. Tạo các số ngẫu nhiên:

Randomize; Khởi tạo chế độ tạo số ngẫu nhiên.
Random(N:Word):word; tạo một số nguyên dương bất kỳ từ 0 (N-1

```
uses Crt;
Begin
Randomize;
Repeat
    GotoXY(Random(80)+1,Random(25)+1);
    TextColor(Random(15)+1);
    Write(Chr(Random(256)));
Until Keypressed;
End.
```

Bài 4

Cách sử dụng một số kiểu dữ liệu

1. Sử dụng kiểu dữ liệu String:

String là một dạng kiểu dữ liệu không chuẩn gồm một chuỗi các ký tự trong bảng mã ACSII. Bạn có thể khai báo các biến kiểu String theo một trong hai cách sau:

```
    Var     Biến:String;
            Biến:String[N];
Ví dụ: Var  Hoten:String[25];
            Thongbao:String;
```

a. Truy xuất kiểu dữ liệu String:

Với kiểu dữ liệu String bạn có thể sử dụng phép gán, sử dụng các thủ tục Writeln, Write, Readln để truy xuất các biến dạng này.

Ví dụ:
```
Var     Hoten:String;
Begin
Write('Cho biet ho va ten: ');Readln(Hoten);
End.
```
* Pascal cho phép truy xuất đến từng ký tự của dữ liệu kiểu String bằng cách sử dụng cú pháp Biến[i]. Với i là thứ tự của ký tự đó trong chuỗi được xem như dữ liệu kiểu Char và có thể tác động bằng các toán tử, hàm trong dữ liệu kiểu Char.

b. Các hàm và thủ tục trên dữ liệu kiểu String:

Phép cộng chuỗi:

+ Các chuỗi nối với nhau bằng phép cộng (+).

Ví dụ: St1:=□Turbo□; St2:=□Pascal□;St:=St1+St2; khi đó biến St có nội dung □TurboPascal□

+ Sử dụng hàm cộng chuỗi của Pascal: Concat(st1,st2,...stn:String):String;

Ví dụ:

St1:=□ECS □;st2:=□Tin hoc tre□;st:=concat(st1,st2); có nội dung □ECS Tin hoc tre□

+ Lấy chiều dài chuỗi:

Hàm Length(st:string):integer;

Cho chiều dài thực của chuỗi st, giá trị này được ghi trong biến st tại vị trí đầu tiên của chuỗi. Nghĩa là để biết chiều dài thực của chuỗi st bạn có thể dùng truy xuất ORD(st[0]);

+ Vị trí của chuỗi con trong một chuỗi:

Hàm POS(Chuoicon:String;Chuoi:String):Byte;

Hàm POS tìm chuỗi Chuoicon trong Chuỗi. Nếu Chuoicon nằm trong Chuoi thì hàm POS trả về vị trí của Chuoicon trong Chuoi, nếu không sẽ trả giá trị bằng 0.

Ví dụ: St:=□Turbo Pascal□; thì POS(□ascal□,st) cho giá trị là 8.

+ Lấy một chuỗi con từ một chuỗi:

Hàm Copy(st:String; vitri:integer; Sotu:Integer):String;

Hàm Copy cho một chuỗi ký tự con của chuỗi st. chuỗi này được lấy từ vitri với Sotu.

Ví dụ:
```
        St:='Turbo Pascal';
        st1:=Copy(st,7,6);
```
khi đó st1 có giá trị □Pascal□;

+ Xoá một chuỗi trong một chuỗi:

Hàm Delete(Var St:string;Vitri,sotu:integer);

Xoá khỏi chuỗi st sotu trừ vitri còn lại sẽ gắn trở lại chuỗi cho biến St;

Ví dụ: St:=□Turbo Pascal□;
```
                Delete(st,7,6);
                Writeln(st); {câu lệnh này hiện nội dung□Turbo□}
```
+ Chèn chuỗi con vào trong chuỗi:

Thủ tục: insert(chuoicon:String; Var st:string; Vitri:integer);

Chèn Chuoicon vào chuỗi st bắt đầu từ Vitri.

Ví dụ: St:=□Tin hoc tre□;
```
        Chuoicon:='ECS ';
        insert(Chuoicon,st,1);
        Writeln(st); {nội dung của st là □ECS Tin hoc tre□}
```
* Chuyển đổi giữa số và chuỗi:

+ Đổi một số thành chuỗi hàm Str

+ Đổi một chuỗi thành số: hàm Val

c. Các giải thuật cơ bản trên dữ liệu kiểu String:

(Đổi chuỗi ký tự sang ký tự in

```
Var i:byte
        st:string;
đoạn chương trình:
        For i:=1 to length(st) do St[i]:=Upcase(st[i]);
```

(Cắt ký tự trắng bên trái chuỗi ký tự:

Ví dụ chuỗi □ Turbo Pascal□ có một số ký tự trắng bên trái bạn có thể xoá nó để đưa về chuỗi □Turbo Pascal□ bạn bắt đầu tìm từ vị trí đầu tiên của chuỗi, kiểm tra xem có phải là một ký tự trắng hay không nếu đúng xác định vị trí và xoá ký tự trắng đó khỏi dãy.

```
Var     i:byte;
        st:string;
Đoạn chương trình:
i:=1;
While st[i]=#32 do inc(i); {#32 là ký tự trắng □ □}
Delete(st,1,i-1);
```

(Cắt ký tự trắng bên phải:

Bạn dùng vòng While ... do với điều kiện còn tìm thấy ký tự trắng nhưng bắt đầu từ vị trí cuối cùng ngược tới vị trí đầu.

```
Var i:byte; st:string;
Đoạn chương trình:
i:=length(st);
while st[i]=#32 then dec(i);
st[0]:=Chr(i)
```

(Cắt các khoảng trắng ở giữa chuỗi:

```
Var     i:byte;
        st:string;
đoạn chương trình:
        i:=pos(#32#32,st);
        while i<>0 do
                begin
                        Delete(st,i,1);
                        i:=Pos(#32#32,st);
                end;
```

2. Sử dụng dữ liệu kiểu array:

Array là một kiểu dữ liệu có cấu trúc bao gồm một số cố định các thành phần có cùng kiểu dữ liệu. Mỗi thành phần của array được truy xuất thông qua các chỉ số môt tả vị trí thành phần đó trong array.

a. Khai báo kiểu:

Cú pháp:

```
Type    <Tên kiểu> = array[chỉ số] of <Kiểu biến>;
```

Ví dụ:

```
Type
        Tocdo=array[Oto, Tai, Buyt, Dulich] of Integer;
        Hoten=array[1..30] of String;
        Loai = array['A'..'Z'] of Byte;
        OrdType=Array[Char] of Byte;
        AsciiType=Array[Byte] of Char;
```

b. Khai báo biến:

Cú pháp:

```
Var     <Tên biến>:ARRAY[Chỉ số] of <Kiểu biến>;
```

Ví dụ:

```
        Var
                Diem:array[1..30] of Byte;
        Type
                Xe=(Oto, Tai, Buyt, Dulich);
                Tocdo=Array[Xe] of Integer;
```

ý nghĩa: Bạn có thể tưởng tượng một biến như một hộp để chứa dữ liệu và Array là một tập hợp nối tiếp các hộp lại với nhau để chứa dữ liệu có cùng kiểu, mỗi hộp được đánh số thứ tự bằng chỉ số đã khai báo và dãy hộp được đặt chung bằng một tên đó là tên của biến kiểu Array.

Ghi chú: Để khai báo Array nhiều chiều bạn theo cú pháp sau:

```
        Var <Tên biến>:Array[Chỉ số1, chỉ số 2,...] of <Kiểu biến>;
        Ví dụ:Manhinh: Array[1..25,1..80.0..15] of Integer;
```

c. Truy xuất dữ liệu kiểu Array:

(Xuất dữ liệu kiểu Array:

Các thủ tục xuất nhập biến như Writeln, Readln không thể truy xuất thẳng biến Array mà phải thông qua từng thành phần của Array đó.

Ví dụ:

```
        Var     a,b:Array[1..100] of String[30];
                c:Array[1..30,1..4] of String;
```

Bạn không thể viết:

```
        Write(a); Readln(b); Readln(c);
```

Mà bạn phải viết:

```
        Write(a[1]);
        Write(a[2]);
        .........
        Write(a[100]);
        Readln(C[1]);
        ..........
```

Array thường kèm theo biến đếm có cùng kiểu với chỉ số của Array, thay đổi biến đếm của vòng lặp bạn sẽ lần lượt truy xuất hết các thành phần của Array như sau:

```
        For i:=<Chỉ số đầu> to <Chỉ số cuối> do
                Begin
                        Readln(<Tên Biến>[i]);
                        Writeln(<Tên biến>[i]);
                End;
```

Ví dụ:

```
Var     Hoten:Array[1..100] of String;
        i:Byte;
Begin
        For i:=1 to 100 do
                Begin
                        Write('Nhap ho ten: ');Readln(Hoten[i]);
                End;
End.
```

(Gán dữ liệu kiểu Array:

Bạn có thể gán nội dung của hai biến Array cùng kiểu

Ví dụ:

```
Var     a,b:Array[1..10] of Integer;
```

```
        i:Byte;
Begin
        For i:=1 to 10 do Readln(a[i]);
        b:=a;
        For i:=1 to 10 do Writeln(b[i]);
Readln;
End.
```

d. Sắp xếp dữ liệu trên dãy:

Sắp xếp là một quá trình tổ chức lại một dãy các dữ liệu theo một trật tự nhất định. Mục đích của việc sắp xếp là giúp cho việc tìm kiếm dữ liệu trong một dãy được dễ dàng hơn. Nguyễn lý chung của sắp xếp là So sánh và hoán vị.

Sắp xếp bằng phương pháp chọn lựa đơn giản bắt đầu từ thành phần đầu tiên của dãy so sánh với các thành phần đó với nhau. Tiếp tục đến thành phần kế tiếp so sánh với các thành phần còn lại cho tới khi gặp thành phần cuối cùng. Vì sau thành phần cuối cùng không còn thành phần nào nữa để so sánh nên việc sắp sếp đã hoàn tất.

Với phương pháp này hình thành hai vòng lặp. Vòng lặp thứ nhất lấy thành phần thứ i từ 1 tới N-1 so sánh và hoán vị với vòng lặp thứ 2 với j từ i+1 tới N

Giải thuật:

Cho i từ 1 đến n-1 thực hiện
    Cho J từ i+1 đến n thực hiện
        Nếu thành phần tứ i nhỏ hơn thành phần thứ j thì
            Hoán vị hai thành phần thứ i và thứ j.

Ví dụ:

```
var     i,j:Byte;
        Trunggian:Real;
Begin
        For i:=1 to n -1 do
                For j:=i+1 to n do
                        if A[j]<A[i] then {Thực hiện việc so sánh}
                                Begin
                                        Trunggian:=A[j];      {Thực hiện việc hoán vị nếu
sanh sánh}
                                        A[j]:=A[i];           {bằng True}
                                        A[i]:=Trunggian;
                                End;
End;
```

3. Dữ liệu kiểu File:

Các dữ liệu đã khảo sát ở trên đề thực hiện trong bộ nhớ RAM khi khởi động chương trình, khi chấm dứt chương trình các dữ liệu trên bị mất vì vậy việc lưu trữ dữ liệu lâu dài hoặc sử dụng lại dữ liệu đó bạn cần sử dụng dữ liệu kiểu File.

Có 2 loại dữ liệu kiểu File:

- File có kiểu: Dữ liệu được lưu trữ trên đĩa thành một tập tin chứa các Record và được ghi dưới dạng nhị phân
- File dạng Text: Dữ liệu được lưu trữ trên đĩa thành một tập tin chứa các dữ liệu dưới dạng mã ASCII. Bằng các phần mềm soạn thảo văn bản có thể đọc được dữ liệu này.

a. Khai báo dữ liệu kiểu File:

Với dữ liệu File có kiểu:

```
        Var     <Tên biến>:File of <Kiểu biến>
```

Ví dụ:

```
        Var     f:File of Byte;
                S:File of String;
        Type
                Hosocanbo=Record
                        Hoten:String;
                        Diachi:String;
                End;
        Var     F:File of Hosocanbo;
```

Với File dạng Text

Var <Tên biến>:Text;

Khi khai báo tập tin kiểu Text dữ liệu ghi vào đĩa dạng File ASCII như là một tập tin văn bản

b. Các thủ tục chuẩn trên dữ liệu kiểu File:

(Gán tên tập tin cho một biến:

Thủ tục Assign(Var <Tên biến File>,<Tên File>:String);

Thủ tục này nhằm mục đích gán một tập tin trên đĩa cho Tên biến File trong RAM.

Ví dụ:

```
Var     F:Text;
Begin
....
Assign(f,'Chaoban.txt');
....
End;
```

(Mở và tạo mới một tập tin:

Thủ tục: Rewrite(Var <Tên biến File>); (Ví dụ: Rewrite(f);)

Thủ tục này tạo một tập tin trên đĩa có tên đã gán cho Tên biến File bằng lệnh gán Asign đồng thời mở tập tin đó ra để truy xuất dữ liệu.

Chú ý: Khi mở tập tin bằng lệnh Rewrite nếu trên đĩa đã có tập tin trùng với tên bạn đặt thì tập tin trên đĩa sẽ bị xoá thay vào đó là một tập tin trống mà bạn đã gán tên cho Tên biến File. Nên bạn cần cẩn thận khi mở tập tin bằng lệnh Rewrite.

(Mở một tập tin đã có:

Thủ tục: Reset(Var <Tên biến File>);

Mở một tập tin đã gán cho Tên biến File, nạp nội dung của tập tin này vào trong Tên biến File trong RAM để chuẩn bị cho việc truy xuất dữ liệu.

Chú ý: Khi mở một tập tin bằng lệnh Reset nếu tập tin không có trên đĩa sẽ gây lỗi.

(Đóng một tập tin đã mở:

Thủ tục: Close(Var <Tên biến File>);

Thủ tục này chuyển nội dung trong bộ nhớ vào tập tin trên đĩa đồng thời đóng tập tin lại giải toả bộ nhớ dành cho biến tập tin

Chú ý : Các tập tin khi đã mở nếu không đóng lại sẽ mất các dữ liệu truy xuất trên Tên biến File.

(Truy xuất dữ liệu:

Việc xuất nhập dữ liệu trên biến File có kiểu chỉ được thực hiện thông qua từng Record

- Đọc dữ liệu từ tập tin dùng thủ tục Read(<Tên biến File>,<Tên biến>);
- Ghi dữ liệu vào đĩa: dùng thủ tục Write(<Tên biến File>,<Tên biến>);

Chú ý: Không dùng các thủ tục Readln, Writeln vì sau khi đọc, ghi xong một Record con trỏ sẽ tự động nhẩy sang Record tiếp theo.

c. File văn bản:

Để đọc dữ liệu trên File văn bản (Text File) bạn dùng các thrủ tục sau:
- Read(Var f:Text;GT1,GT2...);
- Readln(Var f:Text; v1,v2,...); Đọc dữ liệu trên một dòng và gán cho một hay nhiều biến.

* các hàm tiện ích trên kiểu Text File:
- Eoln(Var f:Text):Boolean; Cho biết tình trạng hiện thời của con trỏ trên dòng đang truy xuất đã hết dòng hay chưa, nếu đã hết dòng hàm Eoln cho giá trị True.
- Eof(Var F:text):Boolean; Cho biết con trỏ hiện thời đã đến cuối tập tin hay chưa, nếu đã ở cuối tập tin hàm Eof sẽ trả về giá trị True.

* Một số ví dụ:

(Hiện toàn bộ nội dung File lên màn hình:

```
Var     f:Text;
        ch:Char;
Begin
{Tên tập tin gõ vào từ dòng lệnh}
if Paramcount=1 then
        Begin
                Assign(f,ParamStr(1));
                Reset(f);
                While not Eof(f) do
                        Begin
                                Read(f,Ch);
                                Write(ch); {Xuất dữ liệu ra màn hình}
                        End;
        End;
End.
```

(Đọc nội dung một File hiện ra màn hình dùng biến Line kiểu String để đọc từng dòng:

```
Var     f:Text;
        Line:String;
Begin
{Tên tập tin gõ vào từ dòng lệnh}
if Paramcount=1 then
        Begin
                Assign(f,ParamStr(1));
                Reset(f);
                Repeat
                        Readln(f,Line);
                        Writeln(Line); {Xuất dữ liệu ra màn hình}
                Until Eof(f);
        End;
End.
```

(Bạn có thể tạo một chương trình phát nhạc từ một File nhạc tạo ra theo cấu truc Cao độ, trường độ theo từng dòng?

Ví du:
{----Lambada.Not----}

```
1318   60
1174   20
1046   20
988    20
....
```
Trong đó số đầu chỉ cao độ, số sau chỉ trường độ.

```
Program Lambada;
Uses Ctr;
Const  Dieuchinh=10;
Var    f:Text;
       Caodo,Truongdo:Word;
Begin
Assign(f,'Lambada.not');
Reset(f);
While (not Eof(f)) and (not Keypressed) do
       Begin
              Readln(f,Caodo,truongdo);
              Sound(Caodo);
              Delay(Dieuchinh*Truongdo);
              Nosound;
       End;
Close(f);
Nosound;
End.
```